Study Group Application Final Design Document
By: Sammy Israwi, Martha Cornejo, Michael Garber
and Gabriel Babilonia
Senior Design I
04/26/2016
Contents

Contents .......................................................................................................................................... 3
Executive Summary ........................................................................................................................ 8
Broader Impacts ............................................................................................................................ 9
Individual Motivation ..................................................................................................................... 10
Gabriel ....................................................................................................................................... 10
Martha ....................................................................................................................................... 10
Michael ..................................................................................................................................... 10
Sammy ...................................................................................................................................... 11
Specifications and Backlog ........................................................................................................... 12
Technical Design .......................................................................................................................... 16
Architecture Overview .............................................................................................................. 16
Android Overview ..................................................................................................................... 19
Client ..................................................................................................................................... 19
Cordova ................................................................................................................................... 19
Web Technologies .................................................................................................................... 19
iOS Overview ............................................................................................................................ 19
API Overview ............................................................................................................................ 19
API Design ............................................................................................................................... 20
Swagger UI ............................................................................................................................... 25
Authentication ............................................................................................................................ 26
Azure Environment Overview .................................................................................................. 29
App Service Overview .............................................................................................................. 29
SQL Azure Overview ................................................................................................................. 30
SendGrid Overview .................................................................................................................... 31
Azure Web Jobs .......................................................................................................................... 32
Notifications ............................................................................................................................... 33
Environment Setup ................................................................................................................... 34
Scalability Plan ........................................................................................................................... 37
Application Workflow .................................................................................................................. 40
Create/Edit/Delete Study Session ............................................................................................. 40
Choosing Session to Attend ...................................................................................................... 42
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Searching Study Sessions</td>
<td>44</td>
</tr>
<tr>
<td>Cordova</td>
<td>46</td>
</tr>
<tr>
<td>Why did we decide for a hybrid mobile application?</td>
<td>46</td>
</tr>
<tr>
<td>The reach of hybrid applications</td>
<td>46</td>
</tr>
<tr>
<td>Different Hybrid Approaches</td>
<td>47</td>
</tr>
<tr>
<td>Engine limitation of Ionic</td>
<td>47</td>
</tr>
<tr>
<td>Performance analysis of hybrid applications</td>
<td>48</td>
</tr>
<tr>
<td>Ionic’s work on performance</td>
<td>49</td>
</tr>
<tr>
<td>Quality of Life improvements with Ionic and ngCordova</td>
<td>49</td>
</tr>
<tr>
<td>Ionic CLI</td>
<td>49</td>
</tr>
<tr>
<td>Ionic Labs</td>
<td>50</td>
</tr>
<tr>
<td>ngCordova Native Functionality Plugins</td>
<td>52</td>
</tr>
<tr>
<td>Budget</td>
<td>54</td>
</tr>
<tr>
<td>Costs from Azure Environment</td>
<td>54</td>
</tr>
<tr>
<td>Risks</td>
<td>56</td>
</tr>
<tr>
<td>Testing</td>
<td>58</td>
</tr>
<tr>
<td>Unit Testing with Visual Studios</td>
<td>59</td>
</tr>
<tr>
<td>Unit testing with Jasmine and AngularJS</td>
<td>61</td>
</tr>
<tr>
<td>ASP.NET</td>
<td>62</td>
</tr>
<tr>
<td>Agile Development and Scrum Approach</td>
<td>64</td>
</tr>
<tr>
<td>Roles in Development</td>
<td>69</td>
</tr>
<tr>
<td>Gabriel Babilonia</td>
<td>69</td>
</tr>
<tr>
<td>Project Management</td>
<td>69</td>
</tr>
<tr>
<td>Authentication and Security</td>
<td>69</td>
</tr>
<tr>
<td>Web App Lead</td>
<td>69</td>
</tr>
<tr>
<td>Martha Cornejo</td>
<td>69</td>
</tr>
<tr>
<td>Database</td>
<td>69</td>
</tr>
<tr>
<td>iOS Development</td>
<td>70</td>
</tr>
<tr>
<td>Michael Garber</td>
<td>70</td>
</tr>
<tr>
<td>Visual Design</td>
<td>70</td>
</tr>
<tr>
<td>Authentication and Security</td>
<td>70</td>
</tr>
<tr>
<td>Quality Assurance</td>
<td>70</td>
</tr>
</tbody>
</table>
Study Group Application Final Design Document

Sammy Israwi ........................................................................................................................... 70
Design Guidelines.................................................................................................................... 70
Cordova...................................................................................................................................... 70
Android Development ............................................................................................................. 71
Application Roles and Administration....................................................................................... 72
User ........................................................................................................................................... 73
Attendee .................................................................................................................................... 73
Scheduler ................................................................................................................................... 73
Administrator ............................................................................................................................ 74
Milestones ..................................................................................................................................... 75
First Semester ............................................................................................................................ 75
Second Semester ....................................................................................................................... 76
Deployment ................................................................................................................................... 79
Android ..................................................................................................................................... 79
Gather Materials ....................................................................................................................... 80
Configure Application ............................................................................................................. 80
Build the Application .............................................................................................................. 80
Prepare Remote Servers ......................................................................................................... 81
Testing the Applications ......................................................................................................... 81
Releasing to Google’s Play Store ............................................................................................ 81
Updating the Application ....................................................................................................... 83
iOS ............................................................................................................................................. 84
Steps for deployment in iOS environment:................................................................................ 84
Web API ..................................................................................................................................... 87
Deployment ............................................................................................................................... 87
Database .................................................................................................................................... 91
Deploying Local Instance: ....................................................................................................... 91
Deploying Azure SQL (Non Local Instance) ........................................................................... 91
Web App .................................................................................................................................... 93
SendGrid ..................................................................................................................................... 95
Azure Blob Storage .................................................................................................................. 95
Wireframes ................................................................................................................................. 96
Mobile Platform........................................................................................................................ 96
Web Platform .......................................................................................................................... 100
Design Guidelines ....................................................................................................................... 105
Material Design ....................................................................................................................... 105
  Principles ............................................................................................................................. 105
Icons ........................................................................................................................................ 106
Logo ........................................................................................................................................ 107
  Study Worms ...................................................................................................................... 107
  Blue Circles ......................................................................................................................... 107
Table With Books ............................................................................................................... 108
Table With Books and Students .......................................................................................... 108
Book With Pencil ................................................................................................................ 109
Database ...................................................................................................................................... 110
  Database Schema for Study Group Application .................................................................... 113
  Database SQL scripts .............................................................................................................. 114
High level description of database tables ................................................................................ 117
  Tbl_Attendee ....................................................................................................................... 117
  Tbl_Organizer ..................................................................................................................... 117
  Tbl_Session ......................................................................................................................... 117
  Tbl_Comment ..................................................................................................................... 118
  Tbl_Mention ....................................................................................................................... 118
  Tbl_Tag ................................................................................................................................... 118
  Tbl_SessionTags ............................................................................................................... 118
  Tbl_SysBadges ................................................................................................................... 119
  Tbl_BadgesHeld ................................................................................................................. 119
  Tbl_Complaint .................................................................................................................... 119
  Tbl_Admin .......................................................................................................................... 119
  Tbl_User ............................................................................................................................. 119
  Tbl_SysInstitution ............................................................................................................... 119
  Tbl_SysCourse .................................................................................................................... 120
Technologies Used ...................................................................................................................... 121
  Git and Github .................................................................................................................... 121
Executive Summary

Group Study aims to bring students together by breaking down common social barriers. Typically, students from a particular course at a university will find it difficult to find classmates to set a study session last minute or even well in advance. Often times, multiple courses could build on each other and the inherent benefit from collaborating and meeting to discuss ideas is lost.

As universities impose the importance of teamwork and communication, it becomes increasingly difficult to find a safe, focused, and simple system that would aid such efforts. Many students turn to well-known social media platforms such as Facebook and Twitter to develop connections amongst students, but inadvertently fail to reach other students who view such platforms as inadequate for educational purposes. Moreover, the Facebook groups feature has become a must have tool for collaboration amongst students taking a particular course, but if a student is not part of the main host group under which a course group has been created, they completely miss the benefits of such feature. Searching relevant groups that pertain to a particular course a student is taking becomes increasingly difficult.

The goal of this project is to build an application that is easily accessible and simple to use, where students are encouraged to create, find and join study groups. The application is composed of a web, Android, and iOS clients. Moreover, student safety is at the core of Group Study. Group Study will employ email verification to enable only university students with an “.edu” account to register. In addition to ease of accessibility and safety, Group Study will be developed with scalability and budget effective principles, by leveraging a flexible cloud hosting environment and building a lean ecosystem of supporting components.
Broader Impacts

As college students, between work, jobs and extracurricular activities our schedules can make it tough to personally find classmates that can meet outside of class to study. It could also be challenging to find study partners or study session at an unplanned time. This project will set a platform exclusively for setting up or joining study groups, this meetings could be set at a specified location or online at a specified time. This will solve the problem of finding or setting up a study session at the last minute and letting other students interested in the same subjects to simply join a study session they are interested in. This project has the ability to impact students who are seeking to benefit from studying in groups by effortlessly allowing the creation and sharing of study sessions across mobile and web applications.

Group Study would be the first step in aligning students around a collaborative body. Bringing together elements such as a simple design, cross-platform support, and the assurance of knowing that every member of the application is a student, the aim would be to reach more individuals whom are looking for a reliable network of fellow students that aspire to succeed in their college careers.

Inherent benefits from organizing and attending study sessions throughout the semester could possibly include an improvement in student engagement. Engagement usually leads to interest and a positive outlook in any one task, thus improved GPAs and an overall sense of a strong community are some of the goals that an application such as Group Study aims to create.

Moreover, we understand that attending study sessions and/or organizing sessions is something that takes initiative and some amount of effort from students. Oftentimes, as students, we tend to not pay too much attention to any one thing that is not directly related to the grade on our course or to the impact on our GPA. Thus, by employing gamification techniques into Group Study we aim to enhance the experience of the application. Allowing students to earn points for attending or organizing study sessions. Elevating a student’s motivation and sense of accomplishment by awarding badges based on a point system.

These are just the many ways in which Group Study aims to bring a community of students together. Breaking down barriers that may come in the way of reaching out to fellow students for help, and motivating students to go above and beyond in order to get the most out of their educational experience.
Individual Motivation

Gabriel

The idea of developing a platform that would unite students and would break social barriers is something that has always been in the back of my mind. As a Computer Science student, upon arriving to UCF, I wasn’t aware of just how difficult it would be to interact with other fellow students. The pressure of knowing exactly what to say, how to come across, and just being social in general was rather difficult. Professors always pushed for teams to form in order to collaborate around a particular assignment, but with classroom capacities of over 200 students, any shy student would find the idea of walking over and talking to fellow classmates almost an impossible task.

We then were taught by our parents to see Facebook as a tool of wasting time, that we should be focused on our school work and avoid distractions during valuable study time. These types of conflicts are imposed onto us as students on a daily basis, and even more so as we become more dependent on technology and develop an increasing online social presence. Group Study would be my small contribution to solving the problem stated above.

As a member of this group, I would like to employ these collaborative techniques throughout the development of the application. Enabling my fellow team mates to bring their own personal ideas and experiences to the table, and making Group Study the best application it could be. Moreover, I would like to learn new technologies while developing Group Study. Moreover, taking an application from start to finish and standing behind a finished and polished product could be considered my overall goal for this project.

Martha

My personal interest to work on this project comes from past experiences where finding a group to study for tests, finals or the foundation exam would have been beneficial but hard to accomplish due to scheduling difficulties. Computer Science students have a great and active Facebook page but Facebook is not a formal or serious place to announce and keep track of study meetings. Today everyone is connected through social media, but none of this platforms are geared towards creating or joining study sessions unless this meeting are within personally known class mates. I believe that by developing a platform uniquely gear towards finding and creating study meetings, it will allow students to get the positive benefits that come from studying with a group. I also believe that this project gives me the opportunity to work with different languages and technologies that I’m interested in.

Michael

My motivation behind this project is to help build an application with a great group of individuals in hopes that more people will study. Also, I hope to gain knowledge and experience in working with API calls and database communication. The people are great, the idea is great, and I am happy to be part of the team. I am interested in the ionic framework in particular because of the clean look and feel it gives to the developer. Many applications utilize the same
ideas and this framework bundles many of the popular ones together so we don't have to spend an exorbitant amount of time re-inventing the wheel like with many native apps out there.

By working on this project, I hope to help create something people like and want to use. Ultimately, if people use this app the way we intend, more people will be studying which means more people will be gaining knowledge. The only way we can get better is to educate ourselves. This application will facilitate this which is why I am excited to be a part of the team.

**Sammy**

UCF being the size that it is, it can be remarkably difficult to find individual personality in the students around you if you never meet them personally. Because of that, approaching a fellow student to invite them over for a study session is no different than approaching a stranger for the same reason. On top of that, it can already be challenging for some students in particular to approach anyone, or to admit that they need help. Because of this issue, many students never really study fully for their classes, and their GPAs suffer because of that.

Our Study Group app looks to solve that issue by making it easier for everyone to create, organize, and attend study sessions.

By working on this project, we are providing a platform for students to use when they need to look for a study session, or to schedule one in a more organized matter.

More personally, this project allows me to learn mobile development, and to delve deeper into different web technologies. By the end of the project development cycle, and by working in this group, I would have learned much more than what I would have done if I wouldn’t have otherwise.
Specifications and Backlog

The application will be developed using the Scrum Agile framework. Our team will hold multiple sprints throughout Senior Design II, each of about 2 weeks in length. Before each sprint, the team will agree on the number of user stories and complexity points assigned to each. After the planning and agreement has been secured, the sprint will begin, and daily standup meetings will be held to go over any roadblocks, progress updates, or questions the team may have.

Below is a list of the product backlog for the application. Some of the user stories below are listed as stretch goals. The team will evaluate the stretch goal stories after the development of the required user stories if time allows.

Please refer to the Application Roles and Administration section for more information regarding the roles listed on the user stories.

<table>
<thead>
<tr>
<th>Story #</th>
<th>Type of User</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>User</td>
<td>As a user, I may only sign up to the application if I am a university students (.edu account), so that I could ensure proper safety among users.</td>
<td>Required</td>
</tr>
<tr>
<td>2.</td>
<td>User</td>
<td>As a user, I want to enter my username, @edu email, and a password into a login screen, in order to be able to utilize the application.</td>
<td>Required</td>
</tr>
<tr>
<td>3.</td>
<td>User</td>
<td>As a user, I want to be able to verify my student status, by accessing my @edu email and confirming through an easy to follow link present on my email in order to be activated as a valid user.</td>
<td>Required</td>
</tr>
<tr>
<td>4.</td>
<td>Scheduler</td>
<td>As a scheduler, after the meeting is over, I must be able to confirm present users in order for attendees to receive reputation points.</td>
<td>Required</td>
</tr>
<tr>
<td>5.</td>
<td>Attendee</td>
<td>As an attendee, if I get confirmed by a scheduler that I attended a session, I must be able to get reputation points.</td>
<td>Required</td>
</tr>
<tr>
<td>6.</td>
<td>Administrator</td>
<td>As an administrator, I must have the ability to deactivate users.</td>
<td>Stretch</td>
</tr>
<tr>
<td>7.</td>
<td>Scheduler</td>
<td>As a scheduler, I must have the ability to set a meeting location using Google Maps visualization map.</td>
<td>Stretch</td>
</tr>
<tr>
<td>8.</td>
<td>Attendee/User</td>
<td>As a user or attendee, if there is a Google map location defined by the scheduler, I must have the ability to click on the map, and have Google maps open.</td>
<td>Stretch</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>User</td>
<td>As a user, I must have the ability to confirm attendance to a scheduled study session.</td>
<td>Required</td>
</tr>
<tr>
<td>10.</td>
<td>User</td>
<td>As a user, I must have the ability to view the rating (reputation points) of the scheduler and everyone whom has confirmed their attendance to the meeting.</td>
<td>Stretch</td>
</tr>
<tr>
<td>11.</td>
<td>User</td>
<td>As a user, I must have the ability to view anyone that has confirmed attendance to a scheduled study session.</td>
<td>Stretch</td>
</tr>
<tr>
<td>12.</td>
<td>User</td>
<td>As a user, I must be able to view who has scheduled the study session</td>
<td>Required</td>
</tr>
<tr>
<td>13.</td>
<td>User</td>
<td>As a user, I must have the ability to tentatively confirm attendance to a scheduled study session.</td>
<td>Stretch</td>
</tr>
<tr>
<td>14.</td>
<td>Scheduler</td>
<td>As a scheduler, I must be able to tell apart which users have tentatively confirmed and which ones have confirmed attendance to a scheduled session.</td>
<td>Stretch</td>
</tr>
<tr>
<td>15.</td>
<td>User</td>
<td>As a user, I must be able to view whom has tentatively confirmed and whom has confirmed attendance to a scheduled session alongside their reputation points.</td>
<td>Stretch</td>
</tr>
<tr>
<td>16.</td>
<td>User</td>
<td>As a user, I must have the ability to report a user to the administrators.</td>
<td>Stretch</td>
</tr>
<tr>
<td>17.</td>
<td>Attendee</td>
<td>As an attendee, I must be able to leave comments on the scheduled study session. The comment should state my username, time of the comment sent, and the ability to mention another user.</td>
<td>Required</td>
</tr>
<tr>
<td>18.</td>
<td>User</td>
<td>As a user, I must be able to view the status of a scheduled session, whether or not is in progress, not started, ended, or canceled.</td>
<td>Required</td>
</tr>
<tr>
<td>19.</td>
<td>Attendee</td>
<td>As an attendee, I must be able to get notified of changes made to a study session, either by email or in app notifications.</td>
<td>Required</td>
</tr>
<tr>
<td>20.</td>
<td>User</td>
<td>As a user, I must have the ability to specify whether or not I want to receive email notification for study session changes.</td>
<td>Required</td>
</tr>
<tr>
<td>21.</td>
<td>Attendee</td>
<td>As an attendee, I must be able to receive text message notifications following changes to the study session.</td>
<td>Stretch</td>
</tr>
<tr>
<td>22.</td>
<td>Attendee</td>
<td>As an attendee, I must be able to choose my notification type, whether email, in app notification, or text message notification.</td>
<td>Required</td>
</tr>
<tr>
<td></td>
<td>Role</td>
<td>Requirement Description</td>
<td>Required/Stretch</td>
</tr>
<tr>
<td>---</td>
<td>--------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>23.</td>
<td>Scheduler</td>
<td>As a scheduler, I must be able to schedule a study session, and be able to define a name (optional), topic (required), time (required), location (required - either online, or physical location), course (required), major, and school for a session.</td>
<td>Required</td>
</tr>
<tr>
<td>24.</td>
<td>Scheduler</td>
<td>As a scheduler, if I am scheduling an online study session, I must be able to place a link to that session (skype, hangouts etc.).</td>
<td>Required</td>
</tr>
<tr>
<td>25.</td>
<td>User</td>
<td>As a user, I must have the ability to favorite/bookmark any user, so that I may get notifications if they schedule a session.</td>
<td>Stretch</td>
</tr>
<tr>
<td>26.</td>
<td>Scheduler</td>
<td>As a scheduler, I must be able to cancel or postpone the scheduled session.</td>
<td>Required</td>
</tr>
<tr>
<td>27.</td>
<td>User</td>
<td>As a user, I must be able to view badges for reputations earned. Schedulers may have a scheduler badge, and a great user could have a great attendee badge.</td>
<td>Stretch</td>
</tr>
<tr>
<td>28.</td>
<td>Administrator</td>
<td>As an administrator, I must be able to upload new badges, set the rating for the badges, and state the name of the badge.</td>
<td>Stretch</td>
</tr>
<tr>
<td>29.</td>
<td>User</td>
<td>As a user, I must be able to submit a request to be an administrator.</td>
<td>Stretch</td>
</tr>
<tr>
<td>30.</td>
<td>User</td>
<td>As a user, I must be able to search and filter based on school, major, topic/subject, course, scheduled times, user (schedulers), and location (online/physical).</td>
<td>Required</td>
</tr>
<tr>
<td>31.</td>
<td>User</td>
<td>As a user, I must be able to define my profile where I could specify my major, my topics of interest, courses that I am currently taking, and institutions of attendance.</td>
<td>Required</td>
</tr>
<tr>
<td>32.</td>
<td>User</td>
<td>As a user, I would like the ability to disable automatic filtering based on my profile if necessary.</td>
<td>Stretch</td>
</tr>
<tr>
<td>33.</td>
<td>User</td>
<td>As a user, I must be able to save up to 5 preferred searches not including my profile automatic search.</td>
<td>Stretch</td>
</tr>
<tr>
<td>34.</td>
<td>User</td>
<td>As a user, I must be able to access a page that displays a leader board of uses and their reputation and badges.</td>
<td>Stretch</td>
</tr>
<tr>
<td>35.</td>
<td>User</td>
<td>As a user, I must be able to get notified if another user has mentioned my user name on a scheduled study session (comments section).</td>
<td>Stretch</td>
</tr>
<tr>
<td></td>
<td>Role</td>
<td>Requirement</td>
<td>Level</td>
</tr>
<tr>
<td>---</td>
<td>--------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>36.</td>
<td>User</td>
<td>As a user, I must be able to access the application on a mobile web app, iOS, and Android device.</td>
<td>Required</td>
</tr>
<tr>
<td>37.</td>
<td>User</td>
<td>As a user, if confirmed to a scheduled, and the session loses the scheduler, I must be able to be notified, and must have the ability to become the scheduler for the session.</td>
<td>Required</td>
</tr>
<tr>
<td>38.</td>
<td>User</td>
<td>As a user, I must have the ability to share a scheduled session on Facebook, Google+ and Twitter to allow users to be able to sign up to the app and attend the session.</td>
<td>Stretch</td>
</tr>
<tr>
<td>39.</td>
<td>Scheduler</td>
<td>As a scheduler, I must be able to upload attachments/content on a study session for attendees to view. The attendee must be able to download the file.</td>
<td>Required</td>
</tr>
<tr>
<td>40.</td>
<td>User</td>
<td>As a user, I must have the ability to set my preferred general notification schedule, either daily or weekly from within my user profile.</td>
<td>Required</td>
</tr>
<tr>
<td>41.</td>
<td>User</td>
<td>As a user, I must be able to receive a digest of study sessions that match my profile or have a special mention from the administrators.</td>
<td>Stretch</td>
</tr>
<tr>
<td>42.</td>
<td>User</td>
<td>As a user, I must be able to view charts depicting most popular study sessions, topics, and an overview of the sessions that I have attended.</td>
<td>Stretch</td>
</tr>
</tbody>
</table>
Technical Design

Architecture Overview

The application has 6 main towers. Each tower relates to a technology stack and component of the application.

The high level architecture outlines the cloud hosting service, mobile application framework, API layer, database, and code repository respectively. The tower is considered team led, as it spans all technology stacks.

The database tower outlines the overall database architecture, from the Azure SQL hosted database to the Azure Storage Blob used. This effort is led by Michael.

The API tower consist of the 3 main instances of the Web API for the application. The API is based on the ASP.NET Web API 2 framework with Entity Framework for the data access layer. This effort is led by Gabriel.

The Web App tower consist of the 3 main instances of the web app hosted on Azure. The instances Dev, Test, and Prod will be used to better adhere to the software life cycle and proper quality assurance. The web app will be developed using HTML5, CSS3, and JavaScript best practices.

Both of the mobile towers will be very similar in structure because of the hybrid nature of our front end application. Our application will be built on Cordova using the bet practices of web development.
Figure 1 - Application Technical Overview Part 1
Figure 2 - Application Technical Overview Part 2
Android Overview
Because of the hybrid nature of our application, the Android and iOS Overview will be overlapping on most parts. Some of the few sections in which there will not be overlap is on individual styling. Each platform will have its own minor design details.

Client
The client for this platform will be the Android application. Depending on the stage of development, the client will be consuming from either the development API, testing API, or production API. However, all the client cares about is that it is consuming from an API. It should virtually not care if it is development, testing, or production back end.

Since we are not having a lot of native functionality, our target and minimum Android platforms will not be as relevant. However, as expected, the client will have the best performance on the newest platforms, since they have the most advanced Webviews.

Cordova
Our application will use the powerful Cordova development platform to take our web application and run it on Android’s Webview, having access to native mobile functionality when necessary.

Web Technologies
Since our application will be built with Cordova, we will use staple web technologies such as HTML5, CSS3, and JavaScript to build our application. We will also use Angular as a front end framework. For styling, we will be using either Bootstrap, or Angular’s Material. It is important to point out that if we decide to go for the Material Design’s guidelines, it is more likely than not that we will be using Angular’s Material.

iOS Overview
As mentioned above, the Android and iOS overviews will be mostly overlapping. This is because of the hybrid nature of our application.

API Overview
The Web API is built using the ASP.NET Web API Framework (Microsoft, 2015). This framework will allow for the leveraging of powerful API functionality such as custom routing and secure authentication. Moreover, the design of the API will consist of four main components, the authentication, business logic, data, and reporting layers.

The API will be hosted on Azure (Microsoft, n.d.), levering its API hosting features. Azure’s API App Service (Microsoft, 2016) will then allow us the use of the Swagger (Swagger.IO, 2016) API self-documentation and exposing that documentation for future API consuming clients. The service will also facilitate the deployment of the API by using built in Visual Studio 2015 API publishing functionality such as deployment scripts, reports, and billing reports.
API Design
Below is a high level design of the centralized API. The API will be the true source for application data. From the API, HTTP REST requests will be served to both Android, iOS and Web App clients. Moreover, data will be pulled from both the SQL Azure and Storage Blob and will be delivered to the clients.

Figure 3 - High Level Overview of Centralized API
Routing:
The Study Group API will have a base URL of “studygroup” for general requests concerning sessions, reputation points, and user profiles amongst others. Sub-routes such as “search” will then target specific functionality. Sample routes could be of the following form: “api/studygroup/getallupcomingsessions” or “api/studygroup/search/getsessions/{query}”. These will be compartmentalized as web API controllers.

Controllers:
API Controllers are .cs classes that inherit from the API Controller interface. Once defined, a controller will handle HTTP requests made by the client requestor. Each major feature of the application is represented by a controller for better readability and unit testing. Each method defined on the controller will be documented by Swagger and tested using the Swagger UI functionality.

SessionController:
This controller will serve all content related to a session, including the creating, deleting, and editing of a session. The API route to this controller will be restricted, only authenticated and verified users will be able to access this route. If the client does not send a valid bearer token alongside their request to this route, the API will send an unauthorized 401 message to the client.

The controller will also have an entity connection to the studygroup database. Using Entity Framework, all requests will be queried instantly from the database. Each method will run simple request validation, such as checking for valid IDs, usernames, and foreign key database enforcement.

Exception handling will also be defined per the methods of this controller.
Methods under this controller include:

- getAllSessions() – Returns a list of all the sessions, filtered on title, sorted by created date.
- getSession(int id) – Returns the session matching provided id value.
- createSessions() – Accepts a session object, returns a success message back to the client upon creation on the database.
- editSession(int id) – Accepts a provided session object, it first queries for the session object using Entity Framework, once the session has been pulled to the controller, basic logic will match the provided object for the edit with the one on record, it will then save the delta’s and will return a success message back to the client.
- deleteSession(int id) – Accepts an id for the session, checks that the session exists, removes the session from the main sessions table, archives the session, sets the necessary flags to notify the users of the deletion if they’ve signed up for this session.
- auxValidationMethod(Session obj) – Returns true if the object is valid else returns false. This method checks for field content and special character clean up.
AuthenticationController:
This controller handles the initial handshake from the requesting of tokens and email verification. The “authentication/token” route will provide the client with a bearer token after accepting the login credentials from the requesting client. The following routes will handle the authentication flow:

- token() - Receives the client credentials from the requesting client, validates if the user is an active member and has been validated, returns the bearer auth token to the client along with a success message.

ReportingController:
This controller is authenticated. Handles all reporting services such as sending key metrics back to clients. The purpose for this feature is to enhance the use of the application by providing insights on the sessions a user has attended, along with information around a particular session such as the topics, course, and location type amongst other metrics.

The reporting controller will also deliver reports in CSV format.

AttachmentsController:
This controller is authenticated. The purpose of this controller is to accept post attachments from the client. Validation of the attachment will include naming, size, and type. Only attachments of type pdf, jpeg, docx, doc, ppt, excel, png and txt.

All attachments will be uploaded to the Azure production blob. A confirmation will be sent to the client. Methods under this controller will include:

- uploadAttachment(Object obj) – This method is of http post. The object will be validated to ensure proper type, name and size. The method will then pull login credentials from app.config for authenticating against Azure blob storage. Once the file has been uploaded, the method will return a success status back to the client.
- getAttachment(int id) – This method is of http get type. It will require an ID to be provided by the requestor. The method will pull the login credentials for Azure blob storage from app.config. Once the file has been retrieved, it will return the file to the client via a stream of bytes with the type of file, the name, and the size. If the file is not found, it will return a not found http status to the client for proper handling.
- deleteAttachment(int id) – This method is of http get type. It will require the ID of the file attachment. It will validate that the file exists on the blob storage, it will then delete the file and return a success message back to the client. If the file is not found, a not
CommentsController:
This controller will be authenticated. It will pull the comments for the corresponding session. It will also handle all of the CRUD functionality for the comments. Some of the methods will include:

- `getAllComments(int ID)` – This method will require the id of the session. It will then query the `tbl_SessionComments` for the list of comments. This method will take the query returned from Entity Framework and will create an enumerable query of type `tbl_SessionComments`. This object will be returned to the client via a stream of bytes. If no comments are found, it will simply return an empty list back to the client.
- `saveComment(Object obj)` – This method will require than object is sent. The method is of http post type. The method will validate the comment by ensuring that no special characters or attachments are sent on the free text field. It will then open a connection to the database by using Entity Framework. The object should also contain the ID of the session. The comment will be uploaded to the database using the session ID. It will then return a success message back to the client.

UserProfileController:
This controller will be authenticated. The purpose of this controller is to pull all the details regarding user profile, such as university, reputation points, profile picture and other personalized items. Some of the methods include:

- `getUserProfile(string email)` – This method will require authentication. It will query from the `tbl_User` for profile properties. The method will require that the user email is passed as a parameter for the Entity Framework query. This method will return an object of type `tbl_User` with only profile related parameters, it will ignore the password salt and authentication properties such as email notification sent bool field etc.
- `editUserProfile (Object obj)` – This method will require authentication. It will query the user profile on the `tbl_User` by using the user’s email address (included on the object). The object will be matched for deltas and Entity Framework will save the alterations to the `tbl_User`. The method will return a success message back to the client after successful completion of the database commit.

AdminController:
This controller will be authenticated against the `tbl_Admins`. The purpose of the controller is to handle administrator functionality, such as taking actions on reported users, canceling of users, running user reports, create and manage institutions, courses, and topics. Some of the methods include the following:
• createInstitution(Object obj) – This method will require authentication. It will accept an institution that will have the basic parameters of name, location, set of courses associated with the institution. The method will return a success message back to the client upon successful commit to the database.

• editInstitution (Object obj) – This method will require authentication. It will accept an institution object. The method will grab the delta from the two objects on the database and the one provided, it will then save the changes to the database. It will return a success message back to the client once the operation has been successfully committed to the database.

• addCourse(Object obj) – This method will require authentication. It will accept a course object which will contain course name, the name of the corresponding institution and the abbreviation string. The method will return a success message back to the client once the operation has been successfully committed to the database.

• editCourse (Object obj) - This method will require authentication. It will accept a course object. The method will grab the delta from the two objects on the database and the one provided, it will then save the changes to the database. It will return a success message back to the client once the operation has been successfully committed to the database.

Note: this method does not employ deletion methods, as an active Boolean is used on each of the institution and course system tables. If a course needs to be disabled, the Boolean field could be used to remove the course from the application. Deletion could cause data issues as these are tagged to sessions. Once a course has been disabled, any corresponding sessions will be canceled. Same operation applies to the institution disabling.
Swagger UI
This module will allow us to test the API methods defined by each controller. It also serves as a means of documenting the API while development takes place. The team can use this UI to unit test a specific method of a controller using authentication and actual parameters for end to end simulation. An example of Swagger UI is shown on the figure below.

Figure 4 - Swagger IO UI Example  (Getting Started with ASP.NET Web API 2 (C#), 2015).
Authentication

The authentication is handled by the ASP.NET Identity 2.0 framework (Microsoft, 2016). The framework allows for individual logging accounts to be created within a databases schema already defined by the framework. Moreover, the framework also includes Oath 2.0 authentication protocol bootstrap code, which substantially reduces the development effort for the authentication component of the application. The authentication flow is described on figure 1 below:

Each client, including the web application, iOS, and Android will interact with the API authentication module before any resources (sessions, modifying user profile, viewing reputation points etc.) are accessed.

If a user from either client is not authenticated, the API will send back a 401 message: 
*Authorization has been denied for this request.* It is the responsibility of the client to redirect the user to the login screen for proper authorization. If the user hasn’t already created an access account, the client will need to direct an API request to the “/Account/Register” to register the user. Once the registration is completed, the server will then initiate the email verification module using SendGrid. An email is then sent to the registering user, at which point an unique link will be provided for verifying the account. Each unique verification link will also contain an authentication bearer token, this allows the user to verify the account and also login for the first time, they won’t be required to go back to the login screen again after their email verification, this authentication flow will only occur during the user’s first sign up.
If the user already has an account registered, the client must direct a request to the following API route: “/Token”. Once the response is sent to the API, it will then provide the client with a bearer token that will contain the following information:

```
{
  "access_token":"",
  "token_type":"bearer",
  "expires_in":1209599,
  "userName":"johnsmith@universitycollege.edu",
  "issued":"Wed, 01 Oct 2014 01:22:33 GMT",
  "expires":"Wed, 15 Oct 2014 01:22:33 GMT"
}
```

The client will then be responsible for sending the bearer token alongside any requests made to an authenticated route on the API.

Logging out of the application is not mandatory, it is expected that the client will handle this action. However, the API will set all awarded bearer tokens to expire within 24 hours of creation, this will help reduce any unwanted access.

External login services will not be used, and it is not in the imminent scope of the application. The main concern behind restricting to only local accounts, is ensuring that users have an @edu email account and the verification process behind the account.

The image below shows the user login flow. The first step the user sees when they access the application is the log in screen. If a user has previously registered and have already verified their @edu account by clicking the verification link on the email, they'll be able to jump to the enter credentials screen. They'll enter their email and their password. Once the system verifies the credentials, the user will be directed to the main menu screen.

If the user has not previously registered, the system will require their registration and will proceed to sending the user with the @edu verification via email. If the user never verifies, they'll restricted from accessing the application as they wouldn't be able to pass after credentials screen.

If the user verifies after clicking the email link, they'll be sent to the main menu as stated above.
Figure 6 - Authentication Flow
Azure Environment Overview

The Azure environment is composed of multiple platform as a service components. Our goal as a team is to reduce the number of components that would require infrastructure as a service due to the added costs and maintenance overhead it would bring to scope of the project. The competitive pricing and student promotions offered by Azure makes the platform an obvious choice for hosting our API, database, and reporting components.

App Service Overview

To host our API, Azure’s API Apps makes it easy to configure and deploy any of our API instances. Moreover, instance management and performance monitoring services are available through the API App. Documentation using Swagger is also another important feature the team plans on leveraging for ease of consumption and for maintainability. The API will be deployed on the East 1 Azure data center.
SQL Azure Overview

![SQL Azure Logo](image)

Figure 9 - SQL Azure

Our three main database instances will be hosted using the SQL Azure service. SQL Azure allows us to host SQL database on the cloud as a service removing the overhead of maintaining our own server instance or virtual machine. All three instances (test, development, and production) will be hosted on one server for ease of management. Moreover, the server will be deployed on the East 1 Azure data center. The only pitfall to using Azure SQL is the lack of native SQL Server Studio features, such as quick create tables and other common scripting tasks using the Studio UI controls. All other native SQL queries are available on the platform.

To connect to these databases, Azure SQL Studio must be used. Once the server, username, and password are provided.

For more information regarding the Azure SQL implementation, refer to the Database section of the document.
SendGrid Overview

SendGrid is a third party service offered by Azure on its marketplace. The service is considered platform as a service, it allows us to send emails from the Web API to registered users. SendGrid will integrate with the Web API registration module, an email will be sent to the user to verify their account before they could proceed to use the application. Moreover, SendGrid will enable us to leverage email notifications for the application, such as the canceling of a scheduled study session, a weekly digest of scheduled study sessions etc.

SendGrid's plan allows us to send about 100k emails in any given month for about $19.99. This cost will be absorbed by our monthly Azure credit services through the Azure BizSpark program. (Reitan, 2015) For more information, please refer to the Budget section of the document.
Azure Web Jobs

Figure 11 - Azure WebJobs

Web jobs from Azure allows us to run automated scripts or c# console applications created on Visual Studio. Moreover, background functionality such as database backups or table data processing is possible by leveraging web jobs. Jobs may run at a scheduled time on a defined reoccurrence or run continuously.

Web jobs may interface with the application database layer using Entity Framework for data processing as stated above.

Web jobs are of no additional cost, they attach to any Azure Web App. (Dykstra, 2016)
Notifications

The Study Group application will leverage user based notifications. Notifications may be sent out regarding study session updates if the attendee has chosen to receive notifications, or general digest notifications from created sessions if the user has chosen to receive notifications at the general level on their user profile.

Each user has the opportunity to choose the level of occurrence notifications are sent. They have two options, either daily or weekly if they have signed up for a general notification or study group notification.

Notifications will be handled in the form of push notifications by the API. The API will employ a Web Job that will run in the background. The responsibility of the Web Job will be to check if a new user is marked as "recentlyJoined", this table row is set to true when they join a new session, the job will then run in the background, it will look at this table and then notify other attendees of the newly joined user if their "wantNotifications" button is set to true. (We will avoid notifying the person that just joined that they have just joined themselves).

Moreover, notifications will be sent to users if the session has been deleted or altered, this includes the changing of an organizer. If a user is deleted, it will set the tbl_User_email to null. The scheduled job will then look on this table for any null entries, and then use that as a flag to notify attendees of a vacant organizer spot.

For changes made to the session, the webjob will look at the tbl_Changes for any items that have been recently changed for a particular study session. A notification of the changes will then be sent to the attendees that have confirmed their attendance.

Notifications will be made in the form of an email, push notification (For iOS and Android) or text message notifications for users that have opted for the option and have verified their phone numbers within the application.
Environment Setup

The Azure dev, test, and production environments are compartmentalized using Azure Resource Management (ARM) (FitzMcken, 2016). ARM allows us to leverage environment templates in the form of JSON objects for quick deployment of platform services such as databases, web applications, API, or server instances.

The JSON object templates for the environment will have the following naming conventions: `nameofservice-dev`, `nameofservice-tst` and `nameofservice-prd`.
The deployment of the script is done by accessing the Azure Portal (portal.azure.com), navigating to Templates:

![Azure Templates](image1.png)

*Figure 12 - Azure Templates*

Once within the Templates blade, choose the dev instance of group study, then deploy:

![Template Environment Deployment](image2.png)

*Figure 13 - Template Environment Deployment*
Dev Environment:
The development environment is composed of the following components under the dev_environment resource group. The resource group is hosted on the East 1 Azure region to reduce latency to US East users (Refer to Scalability Plan) for more details regarding scaling and geo redundancy.

1. groupstudydev – Web application, will host the AngularJS client side component of the application for desktop users. This is part of the appdevplan App Service Plan, it’s set to the “Free” tier.
2. groupstudydb_dev – Azure SQL database, this is part of the appdevplan App Service Plan, under the tier.
3. groupstudystoragedev – Azure Storage account for blob storage. Initial blob size is of 1 Terabyte. This is also set on free/shared performance and locally-redundant storage data structure.
4. appdevplan – The App dev plan is the pricing tier and scalability architecture for the Web App and API App components. Based on this initial setting, both components will scale accordingly. The App Service Plan allows us to streamline the pricing tier of our apps on Azure (Tardif, 2016). This could potentially be managed on a per App service, but the convenience of having all of the settings in one place reduces complexity for maintainability and administration purposes.

Note: For more information regarding the App Service Plan and the pricing tier, please refer to the Budget section of the documentation.

Test Environment:
The test environment is a copy of the dev_environment resource group, but instead named test_environment. The applications will be recreated on this environment from the same specifications used on the dev_environment resource group.

1. groupstudyapitst – Application insights component for traffic and app performance analytics. This component will be heavily used during stress testing.
2. groupstudytst – Web application, will host the AngularJS client side component of the application for desktop users. This is part of the apptstplan App Service Plan, it’s set to the “Standard” tier. This is needed for stress testing prior to production deployment.
3. groupstudydb_tst – Azure SQL database, this is part of the apptstplan App Service Plan, under the tier.
4. groupstudystoragetst – Azure Storage account for blob storage. Initial blob size is of 1 Terabyte. This is also set on standard performance and locally-redundant storage data structure.
5. apptstplan – The App test plan is the pricing tier and scalability architecture for the Web App and API App components. Based on this initial setting, both components will scale accordingly. The App Service Plan allows us to streamline the pricing tier of our apps on Azure (Tardif, 2016). This could potentially be managed on a per App
service, but the convenience of having all of the settings in one place reduces complexity for maintainability and administration purposes.

**Production Environment:**
The production environment will consist of the same applications and services from the test_environment resource group. Differences are only in the pricing and scalability of the services.

1. `groupstudyapiprd` – Application insights component for traffic and app performance analytics. This component will be heavily used during stress testing.
2. `groupstudyprd` – Web application, will host the AngularJS client side component of the application for desktop users. This is part of the `apptstplan` App Service Plan, it’s set to the “Standard” tier. This is needed for stress testing prior to production deployment.
3. `groupstudydb_prd` – Azure SQL database, this is part of the `apptstplan` App Service Plan, under the tier.
4. `groupstudystorageprd` – Azure Storage account for blob storage. Initial blob size is of 1 Terabyte. This is also set on standard performance and locally-redundant storage data structure.
5. `appprdplan` – The App test plan is the pricing tier and scalability architecture for the Web App and API App components. Based on this initial setting, both components will scale accordingly. The App Service Plan allows us to streamline the pricing tier of our apps on Azure. This could potentially be managed on a per App service, but the convenience of having all of the settings in one place reduces complexity for maintainability and administration purposes.

Note: Scalability plan will only apply on the production environment and not on test or dev environments. Please refer to the [Scalability Plan](#) section for more details.

**Scalability Plan**
Azure offers a wide range of scalability options for their platform as a service offering. Group Study will begin on a basic pricing tier plan, offering competitive high performance while the API or database demands remain in low.

Should demand increase, Azure offers automatic scaling of the API App and Azure SQL databases based on CPU usage and HTTP requests made. Our plan would be to implement a low cost auto scaling workflow, where if the CPU usage rise above 70% for 10 minutes, our production API will deploy additional computing cores (up to 3 in order to stay within budget and pricing limit). These features are native to Azure, they allow us to either setup an auto-scale plan, or implement an auto-scale based on a set schedule. For a set schedule auto-scaling, the SQL Azure, Web API, and Web App would all belong to their own scaling containers. This allows us to scale only one service at a time, or pick and choose which services to increase in available Azure resources. An example case could involve a high demand in session scheduling and document attachments from the iOS and Android clients, in this case, the auto-scale plan would increase the instances of the Web API by adding a set fixed additional CPU cores, the
Azure SQL database would then increase its DTU to a fixed number as well. Both of these auto-scaling procedures would then increase the delivery of data and the read/write ability of the database. In the case of the Web App, auto-scaling would add additional cores if needed based on traffic monitoring and performance of requests made to clients.

Figure 14 - Azure Auto Scaling based on CPU and Performance Usage
The following shows an example of scaling the API App alongside any linked resources such as databases or storage accounts (Blob Storage) etc.

<table>
<thead>
<tr>
<th>linked resources</th>
</tr>
</thead>
</table>
| **MYDB2**  
SQL DATABASE | Manage scale for this database |
| **MYDB1**  
SQL DATABASE | Manage scale for this database |

*Figure 15 - Scaling of related linked items, database, and storage*
Application Workflow

The Study Group application will employ 3 main workflows, that of creating a study session, attending a study session, and searching for sessions to attend. Each workflow below depicts the higher level “flow” of the feature. More in depth design will follow prior to implementing the feature.

Create/Edit/Delete Study Session

A study session could be created by any authenticated user of the application. The user will be able to define the title of the session, topic by means of hashtags, schedule user name is applied automatically, a brief description (180 characters max), start and end times (including hourly definitions), location type, location, and any relevant attachments (up to 2MB in size).

Required fields are: Title, topic hashtags, start and end dates, and location type.

After the session has been created, the user will be regarded as a scheduler for the particular session. They will be redirected to application’s main menu, shortly after an alert will notify the user/organizer that the session has been successfully scheduled.

Should an organizer decide to update a session, they will have the ability to alter any field of the session, including the organizer itself (Reassigning of the organizer role for the session to another user). After changes are committed, the application will notify users of the change.

If an organizer deletes the session, the application will notify the attendees of the cancelation. An organizer may cancel a session at any given moment until 15 minutes prior to the starting time of the session.

If an organizer decides to leave the role of organizer for a session, the application will notify all confirmed attendees (Not including tentative attendees). Any confirmed attendee has the opportunity to take on the role of the organizer.
<table>
<thead>
<tr>
<th>Creating Study Session</th>
<th>Edit/Delete Study Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fills form</td>
<td>View Study Session</td>
</tr>
<tr>
<td>Submits form</td>
<td>No, Edit</td>
</tr>
<tr>
<td>Redirect to Dashboard</td>
<td>Yes, View</td>
</tr>
<tr>
<td></td>
<td>No, Notify Attendees</td>
</tr>
<tr>
<td></td>
<td>Yes, Ask for new organizer from host list</td>
</tr>
<tr>
<td></td>
<td>No, Cancel Session</td>
</tr>
<tr>
<td></td>
<td>Yes, New Organizer</td>
</tr>
<tr>
<td></td>
<td>View Study Session</td>
</tr>
</tbody>
</table>

*Figure 16 - CRUD Study Session*
Choosing Session to Attend

Any authenticated user is able to attend a study session. A user is considered an attendee when they have either confirmed attendance or confirmed a tentative attended for a particular study session.

A user may find a study session to attend either by searching, clicking on a suggestion session based on user profile, or simply by browsing any session on the main dashboard amongst other methods.

Any user may open a study session for viewing. For any particular session, a user may favorite the session by pressing on the heart button, or click on the confirm attendance button (check mark).

Once the user has either confirmed attendance (including tentative attendance), they’ll be able to access the comments section of the study session. However, they may view any attachments created by the organizer prior to taking action.

The attendee will then be able to reference the study session on the dashboard, the session will appear under the Going tab.
Figure 17 - Attend a Study Session
Searching Study Sessions

Any authenticated user is able to search for study sessions. Study sessions may be searched based on a specified criteria set, or by searching for a text that fits a particular group of study sessions.

For searching based on criteria, the user may define fields such as scheduler, tags, starting or ending times, or a particular location type. At least one criteria must be selected before searching.

Searching based on text does not require any criteria to be entered other than a piece of text. The application will then match the text to any data field from a study session. Only the top items that match the search query will be brought to the top of the search results. A cap of 100 study sessions will be placed if the query made is too general.

Moreover, a user may have suggested study sessions based on their profile data matching created study sessions. The settings for the suggested sessions could be changed by altering profile criteria such as favorite sessions or topics, institution, major or preferred times.
Figure 18 - Searching Study Sessions
Cordova

Cordova is the hybrid development solution managed by Adobe that, as of right now, the team has decided to use for the development of our mobile solutions.

Why did we decide for a hybrid mobile application?

During our first planning stages at the beginning of the Spring 2016 semester, we asked ourselves which platforms we will be developing our project on. When the project was first presented to the class by Gabriel Babilonia, Web and Mobile were mentioned. However, once we started going into the planning stages, we needed to iron out the details on them.

Regarding smartphone development, we faced the question on whether we will build a mobile website, or create an application using a native or a hybrid approach. There were advantages and disadvantages for each approach.

Native applications are developed for a specific mobile operating system (iOS, Android, Mobile Windows 10, and so on) by using their own separate development kits. For example, we would need to use the Java programming language or Kotlin (a trending development solution as of recently) for Android, Objective C or Swift for iOS, and C# for Windows phones. If we instead choose to develop a hybrid application using web technologies that we already know, we will develop one application to work on all intended platforms.

The reach of hybrid applications

Developing a hybrid application will also mean that we will reach a wider share of the mobile market by developing a single application, compared to the section of the mobile market we would reach if we developed for only one smartphone platform, as evident on the following table.

<table>
<thead>
<tr>
<th>Region</th>
<th>2015 Shipment Volumes</th>
<th>2015 Market Share</th>
<th>2015 YoY Growth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>1,161.1</td>
<td>81.2%</td>
<td>9.5%</td>
</tr>
<tr>
<td>iOS</td>
<td>226.0</td>
<td>15.8%</td>
<td>17.3%</td>
</tr>
<tr>
<td>Windows Phone</td>
<td>31.3</td>
<td>2.2%</td>
<td>-10.2%</td>
</tr>
<tr>
<td>Others</td>
<td>11.3</td>
<td>0.8%</td>
<td>-16.8%</td>
</tr>
<tr>
<td>TOTAL</td>
<td>1,429.8</td>
<td>100.0%</td>
<td>9.8%</td>
</tr>
</tbody>
</table>

Table 1: Worldwide Smartphone Forecast by OS – Shipments, Market Share, Year-Over-Year Growth (shipments in millions) (International Data Corporation, 2015)

Native applications, however, have better performance on their each intended platform. Since native functionality is more easily accessible, and since performance on native apps is greatly improved, some projects will rightfully decide to use native development approach. Computationally-heavy applications, or applications that make a heavy use of native mobile functionality will have an easier development if using native approach.
Once we made the decision on developing a hybrid application instead of one or two native ones, the next question to ask what tool we will use to develop our native application.

**Different Hybrid Approaches**

The two first options are PhoneGap and Cordova. Both stemmed from the original Nitobi’s PhoneGap developed back in 2009. As it stands right now, Cordova is the core open source hybrid development tool managed by Apache, and PhoneGap is Cordova with Adobe’s hand and software thrown in the mix. Core Cordova has a higher acceptance rate than Adobe’s PhoneGap.

On top of that, one can use the Ionic framework with Cordova for a better development cycle, and the ability to use one’s knowledge of Angular when developing the application. With Cordova and PhoneGap one can also use ngCordova, which allows the developer easier access to native phone functionality.

Facebook has also recently thrown its hat on the mobile hybrid ring with React Native. And with Xamarin’s recent purchase by Microsoft, hybrid mobile applications now not only have to be developed with web technologies, but now with C# too.

<table>
<thead>
<tr>
<th>Cordova with tools</th>
<th>React Native</th>
<th>Xamarin</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pros</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Ability to use web technologies, such as HTML, CSS, Javascript, and Angular</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Very widely used.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Easy to develop in.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Backed by technology giant, Facebook.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Based on successful web framework, React.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• In our case, both backend and frontend would be on the same language.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Boasts native-like performance.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Backed by technology giant, Microsoft.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Cons</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Performance wise, it is still a web app that runs within the web-view, calling on native functions when necessary.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Very early in development, therefore very volatile in features.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Because of the recent purchase, there may be some big changes coming that can destabilize our development cycle.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table 2: Comparison Table between Cordova, React Native, and Xamarin*

Since the biggest con on Cordova’s side is only its performance, we opted for using it instead of Facebook’s React and Microsoft’s Xamarin.

There are also some other smaller hybrid development platforms, like Appcelerator and Sencha. Given our tight timeframe and milestones, we decided against spending time to properly research these tools.

**Engine limitation of Ionic**

We mentioned before that we were going to use Ionic. However, we didn’t mention that we made the decision of using Cordova instead of PhoneGap before learning about the existence and
advantages that Ionic offers. Ionic uses the Cordova platform at its core. However, it could use PhoneGap just as well without having significant changes.

**Performance analysis of hybrid applications**

Performance depends on the underlying hardware, the individual platform, and the version of the platform, because it’s highly dependent upon the quality of the app host and hardware that’s running the code.

Since hybrid applications run on the mobile phone’s Webview, all hybrid applications are limited by the performance and the capabilities of the Webview of that hardware, that individual platform, and the version of the platform.

On Windows 8.1 and Windows Phone 8.1, you’re running a native app (no webviews), and because Microsoft has put a lot of performance work into the IE engine on which the app host is built, JavaScript/Cordova apps run quite well. Compared to native applications, they run 15% to 50% slower.

On Android, iOS, and Windows Phone 8, Cordova apps run JS inside a Webview, and thus are very much subject to Webview performance. The most recent versions of the operating systems have greatly improved the performance of the Webview compared to their last platform versions. The newer the platform, the better and faster the Webview is, and the closer it is in performance to native applications.

*Figure 19: General comparison of mobile development tools*
That being said, running JavaScript in a Webview on platforms like iOS isn't going to match a native iOS implementation. But since we're not needing that kind of performance, Cordova can deliver a great experience for our users without you having to go native.

Ionic’s work on performance

Ionic is also responsible for the performance of our hybrid application. The creators of Ionic take pride in the fantastic performance of their apps. Some of the more apparent ways Ionic ensures fast applications is with minimal DOM manipulation, zero jQuery, and hardware accelerated transitions. Ionic’s implementation of Angular also allows for short load times on mobile.

Quality of Life improvements with Ionic and ngCordova

Ionic CLI

Building applications using the Ionic Command Line Interface is surprisingly easy. Once Ionic is installed, you start a new project using the command:

```
ionic start <app name> <template>
```

To add a new platform to our project (like Android or iOS), use the following command for each platform:

```
ionic platform add <platform>
```

To build the application to deploy it to a platform, use the following command for each platform:

```
ionic build <platform> (--options)
```

If you want to test your application on an actual device, you have to plug in your device and run the following command:

```
ionic run <platform> (--options)
```

If a phone is not plugged in, the above command will run an emulator on your computer. If one wants to bypass a plugged in device or an installed emulator like Genymotion, run the above command with an added flag, like this:
One of the ways to emulate our application is to run it raw on a browser. The way to do it is simply to run the following command:

```bash
ionic serve --options
```

To use Ionic Labs to troubleshoot and test our applications, we modify the command above. Like this:

```bash
ionic serve -l (or --lab)
```

There are other commands that will be useful when developing our application later on, they can all be found on the Ionic docs.

### Ionic Labs

In the last section we mentioned Ionic Labs. During development we will make extensive use of Ionic Labs, which runs on our browser of choice.

Testing UI elements and navigation is fine when using Ionic Labs. However, when using native plugins and ngCordova, we need to test our application on an actual phone or in a better emulator since the native code will not run on the default Ionic emulator, or Ionic Labs.

More advanced functionality is found on the application called Ionic Lab. Ionic Labs is not only used for testing, but also for management, organizing platform and plugins and more.

When using Ionic Labs to emulate your application, a localhost web server is created and served on port 8100 by default. Once one accesses the page, you are presented with a side by side emulation of your application on both Android and iOS.
Figure 20: Browser presentation of Ionic Labs testing environment
ngCordova Native Functionality Plugins

There are numerous ways to use native functionality on a hybrid application using plugins. One of the easier ways to install and use them is with ngCordova. ngCordova is a collection of 70+ AngularJS extensions on top of the Cordova API that make it easy to build, test, and deploy Cordova mobile apps with AngularJS. The documentation of each plugin makes it easy to use and customize plugins.

Using a native functionality plugin through ngCordova is as easy as declaring dependencies on AngularJS modules and calling the correct functions.

ngCordova is made by the same people that developed Ionic plus the help of additional developers, therefore it seamlessly integrates into any Ionic project, like ours.
However, it is also possible to use ngCordova on Cordova or Phonegap projects that do not use Ionic. Documentation on the ngCordova website details how to use each plugin with, and without Ionic.
Budget

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS Developer Program</td>
<td>$99/year</td>
</tr>
<tr>
<td>Android Developer</td>
<td>$25 – one-time fee</td>
</tr>
<tr>
<td>GitHub Private Code Repository</td>
<td>Free (Through GitHub student pack)</td>
</tr>
<tr>
<td>Apache Cordova</td>
<td>Free</td>
</tr>
<tr>
<td>SendGrid</td>
<td>Absorbed by Azure credits</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>$124.00</strong></td>
</tr>
</tbody>
</table>

*Table 3: High level budget table*

Costs from Azure Environment

Using Azure BizSpark we can abstain from any costs incurred from Azure cloud services as long as our use costs stay below the $150.00 per month.

Our code repository will be provided free of cost through GitHub’s student pack.

Azure BizSpark is a program provided by Microsoft, as an incentive for startups and local small development shops to leverage Azure cloud services. The current membership for our group is valid until 1/5/2017. Our plan would be to either apply for another year of membership to Azure BizSpark, or begin a crowd funding initiative to launch the application for use by other educational institutions.

Through the BizSpark Azure monthly credits, we’ll be using the left over balance after applying the credits towards the platform as a service components.
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Category</th>
<th>Service</th>
<th>Region</th>
<th>Quantity</th>
<th>Usage</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Study App - Cloud Environment</strong></td>
<td>Web + Mobile</td>
<td>Azure App Service Basic Small</td>
<td>US East</td>
<td>1 Instances</td>
<td>744 Hours</td>
<td>Web App - Prod Instance</td>
<td>$55.80</td>
</tr>
<tr>
<td></td>
<td>Web + Mobile</td>
<td>Azure App Service Shared Small</td>
<td>US East</td>
<td>1 Instances</td>
<td>1 Hours</td>
<td>Web App - Tst Instance</td>
<td>$0.01</td>
</tr>
<tr>
<td><strong>Data + Storage</strong></td>
<td>Sql Basic, Standard, Premium Databases Basic B1</td>
<td>US East</td>
<td>1 units</td>
<td>744 Hours</td>
<td>API Db - Tst Instance</td>
<td>$5.13</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sql Basic, Standard, Premium Databases Basic B1</td>
<td>US East</td>
<td>1 units</td>
<td>744 Hours</td>
<td>API Db - Dev Instance</td>
<td>$5.13</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Azure App Service Shared Small</td>
<td>US East</td>
<td>1 Instances</td>
<td>1 Hours</td>
<td>API - Dev Instance</td>
<td>$0.01</td>
<td></td>
</tr>
<tr>
<td><strong>Data + Storage</strong></td>
<td>Sql Basic, Standard, Premium Databases Basic B1</td>
<td>US East</td>
<td>1 units</td>
<td>744 Hours</td>
<td>API Db - Prod Instance</td>
<td>$5.13</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Azure App Service Basic Small</td>
<td>US East</td>
<td>1 Instances</td>
<td>744 Hours</td>
<td>API - Prod Instance</td>
<td>$55.80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Azure App Service Shared Small</td>
<td>US East</td>
<td>1 Instances</td>
<td>1 Hours</td>
<td>API - Tst Instance</td>
<td>$0.01</td>
<td></td>
</tr>
<tr>
<td><strong>Data + Storage</strong></td>
<td>Block Blob Locally Redundant</td>
<td>US East</td>
<td>2 GB</td>
<td>-</td>
<td>API Blob Storage</td>
<td>$0.05</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Azure App Service Shared Small</td>
<td>US East</td>
<td>1 Instances</td>
<td>1 Hours</td>
<td>Web App - Dev Instance</td>
<td>$0.01</td>
<td></td>
</tr>
<tr>
<td><strong>SendGrid</strong></td>
<td>Third Party</td>
<td>Third Party Email Service</td>
<td>US East</td>
<td>3 Instances</td>
<td>744 Hours</td>
<td>Third Party Email Service</td>
<td>$19.99</td>
</tr>
</tbody>
</table>

**Support Level: Included** Billing and Community Support

**Azure BizSpark Monthly Credit** $150.00/Month

**Cost Estimate** $146.99/Month

**Total Credits Left Over after Monthly Cost Applied** $3.01/Month

Prices as of 02/22/2016

Note: Azure App Service Shared Small is defined as usage for 1 hour due to the free version that will be utilized. The cost of $0.01 is unavoidable. These instances are considered “Free”.
Risks

The risks of this project are not great. Minor risks include conflicting schedules and learning curves. We expect these things will happen. If something can go wrong it will go wrong. The things that go wrong cannot be totally foreseen, however, we may assume some things out of paranoia because it is better to be proactive than reactive. Servers may go down, code might get lost or deleted and members might not be available to meet for every meeting.

Our database may not always work when we make changes to our code and our code might not work when we push it to testing because of conflicting function calls, variables or things of the nature. We will outline all the major problems we expect to arise and will tackle these problems by continuously having dialogue and mitigating a solution. Gabe, the great team leader he has been, will help bring us all together so we are up to speed and on the same page. He has the most experience, so he knows how to help us all get to the point we need to be.

As far as technical resource difficulties, we have a monthly spending limit for our cloud services which exceeds our needs, so we don’t have to worry about servers being down or not paid for. We have a strong team that works well together and conflict is minimal. One of our goals is to turn this app into something that people want to use. To do this, we meet and talk about everything on our minds with regard to the development of this app. Even if an idea is stupid, we bring it to the table because there is a chance the idea needs a minor tweak from the groups input to be great. We don’t want to miss out on opportunities like this.

In planning for this app, we have proposed several solutions to make the study process easier and more accessible. Some of these options might not have as positive of an effect as we may think, so we run a risk of adding things to the app that may not be necessary. However, through testing and meeting, we hope to iron out this product so people will want to use it. The idea behind the app itself isn’t complicated, so we don’t have much risk in the way completing everything we have come up with collectively thus far. Also, the response from our users might not be exactly what we think it will be. When we create this app and push it to the internet we will run surveys to see how the app is faring for its users. Based on that information, we can add or remove things from the perspective of the user. In this way, we can maximize the apps approval rating and minimize the chance a targeted user will not like it.

We believe that listening to the user is the most important way to develop superior software. Once people start using this app and recommending it to other people, we will be able to quantify the impact our app has in education. When more people study they become proficient. If we can see our app being adopted by the scholarly masses, we will have overcome our biggest risk by creating stronger, more proficient students. Now, to recap, we will list the apps associated risks in an organized fashion and beside it, we will propose our proactive solutions.

1. **Server down or internet down** -> Have a local copy on our computers to develop locally in dev branch. If our computers break for some cataclysmic reason, we can meet at the Senior Design Lab to complete individual pushes.

2. **Learning curves** -> Reading tutorials relating to our apps functionality and become familiar with specific implementations. Michael will be reading and practicing using
Cordova to make simple apps over the summer so he has a better feel of how to implement coding solutions for this project in the Fall. Sam is well up to speed with Cordova and has pointed us in the direction of the Ionic framework that uses Cordova. It is a high level/high functioning API that integrates with the Cordova framework and allows developers to create clean looking solutions.

3. **User approval** -> When we push our code from testing to production, we will promote our app to users at the university for testing and ask willing participants to complete a survey outlining what they did or did not like about the app. Using this information, we can add minor tweaks to the code in the production branch by first pushing it back to the testing branch. After testing, Gabe will sign off on it and it will go back to production for another round of field testing.
Testing

Michael is in charge of the overall testing for the application's development process even though every member is involved. Each person in the group who gets assigned a task to write some code will submit their code to me and I will test it on our testing server. After the testing is done, our team leader, Gabriel, will sign off on each of the tasks' components. These components include the Web App, Cardova and the API. Since all of these things cannot be done at once, we will start with creating simple functionality and database communications. As we develop our code, we will test the small changes to make sure everything is still working together properly. This will help us ensure a working delivery of the app before the deadline.

The testing stages of our software will also be a 3-tiered system to ensure functionality and usability. The first tier is the development tier. Sam, Martha, Gabriel and Michael will all be meeting every week to bounce ideas off of each other and discuss our progress. After these meetings, we will try to implement the things we talked about in code and push the code to our prospective dev branches. Even if the code does not work properly, it will be pushed to our dev branches. Each of us will have our own dev branch to push to in GitHub.

Once it’s in the dev branches we can begin testing and code integration. Once we compile our individual codes together, we will push it to our testing branch. This is our second tier. Once in the testing branch, again it doesn’t have to be fully functional, we will test it. This is when we will work together on the whole code to integrate our solutions together properly and have it running correctly.

Once it’s running properly, we move it to the production branch. This will happen near the end of our development process when we are ready to take on real users. Before this, while in the testing branch, we will create fake accounts and use them as if they were real clients. This will help to put ourselves in the shoes of our users. When we do this, we will have more insight on what needs to be added or removed. It’s easy enough to talk about and develop an idea, but making people want to use it is our main concern. We will work hard in this area before pushing any part of our solution to the production branch.

In the production branch, we hope to have a solid, robust solution that accounts for all known errors. The only way we can achieve this is through rigorous testing in the testing branch once all the dev branches get pushed to it.
Unit Testing with Visual Studios

To continuously debug our application, we will be using unit testing. Unit testing is easily set up in Visual Studio. First we will add a unit test to our existing project like this:

![Figure 22 - Unit Testing](image)

After that we select the newly created Test and add it to the study application we will be creating. We do that as illustrated here in this picture:
Then we add the project reference to the original project. After this is set up, we need to set up the Web API 2 application. We can easily do that by writing some code for a Model. This is an example for our project, so obviously, it's not implemented here in totality:

```csharp
using System;
namespace StudyApp.Models{
    public class Profile{
        public int Id { get; set; }
        public string Name { get; set; }
        public decimal Rating { get; set; }
    }
}
```
Next thing we do is build it and then add our Web API 2 like this:

![Figure 24 - Scaffolding](image)

After setting this up we give a name to our testing controller and add write some code with 2 async calls for passing our application's data to process while testing for functionality. When we do this we run all of the individual codes our team comes up with together all at once instead of individually and separately. Grouping tests together is facilitated with Jasmine, an industry standard, which we will now expound upon.

**Unit testing with Jasmine and AngularJS**

Jasmine has become one of the industry standards for unit testing. It is a behavior driven framework environment that allows our team to group all of our tests together. This is done by calling the describe function and then the individual tests are called via the it function. With Jasmine, we can group related tests together within our describe block and Jasmine then returns back matches which allows us to make assertions. Other function calls we will be using include before() and beforeAll(). These functions will help us share our testing set-ups and minimize the testing run-time and create test cases that are more focused towards our goals.
We are using ASP.NET for our project because it is easy to work with, can be written in C# (our language of choice for this project), and has a large audience that is comfortable with its visual options. One of the things ASP is popular for is dynamic web forms. We are going to be using these web forms for inputting user data and profile information into the system. Our administrator user will have a form which will allow him to manipulate functionality of individual user accounts. The forms we will be using for the scheduler will have fewer options, and we will be using these forms for account creation.

ASP.NET has many web objects like text boxes, buttons, drop down lists, etc. A cool thing about ASP's objects are that they have been programmed to update in real time. For instance, if we have a form that needs a certain format for a password or any data entry for that matter, we can have ASP.NET objects that can recognize "on-the-fly" whether or not the data has been formatted correctly without the user having to submit the form and waiting for it to reload again to tell them if the data was entered correctly. This feature alone is incredibly user-friendly. Here is a visual example of how the forms can communicate with the server and have the on the fly caching capabilities:
The beauty of this is that it works on top of the HTTP protocol. The HTTP protocol is stateless, but ASP.NET is not. ASP consists of 2 states - page and session. In this way, it can operate as a 2-tiers being connected to each other. To get a better understanding of what this means, we can look at the example of a shopping cart. The User adds items to his cart. The item(s) are then selected by the user from the page he is on, let's say the items page, while total collected items/processes are shown on a different page, like the cart page. HTTP cannot keep track of all the information coming from various pages. With ASP.NET, session state and server side infrastructure keeps track of the information collected globally over a session.

The ASP.NET runtime carries the page state to and from the server across page requests while generating ASP.NET runtime codes, and incorporates the state of the server side components in hidden fields.
Agile Development and Scrum Approach

We chose to use a SCRUM approach with the Agile development process because it seems to be a good solution to many professional developers. In fact, it is one of the best agile development practices in use today. Scrums core concepts, like product backlogs, team roles, sprints, burn down charts, and more are all going to be explained in this segmentation of text so get ready to be bombarded with information.

Our team wants to make a group study app. This is the product we want to build. For this product, we get all kinds of feature requests from mostly the teacher. Most of our creative requirements have come from our team leader, Gabe, as well as Sammy and Marta. In Scrum, features are written from the perspective of the end-user, therefore, features are known as user-stories. The user stories we have can be defined as an attendee, a scheduler and the administrator. The collection of all our user-stories is known as the product backlog. Another way to think of the product backlog is to think of it as a wish list of all the things that would make this study application great.

Our wish list is comprised of the absolutely necessary things we need to get done for our application to be considered complete. This includes our major functionalities such as group creation, comments/moderation, group ratings and admin roles. Once we put our wish list or the product backlog together, we need to start planning which specific user-stories we're going to put into the particular release of our product whether it be in the dev, testing or production branch. But we're getting ahead of ourselves. Let's back up a bit. To build this product, we need to have one or more people in our team who are going to play a variety of roles.
First, we need Martha. She plays the role of product owner because she has a good eye for what people like and don’t like. She helps make sure the right features make it into the product backlog representing the users and customers of the product. Martha helps set the direction of the product. Next up is Gabe. He is the Scrum Master and his job is to make sure the project is progressing smoothly and that every member of the team has the tools they need to get their job done. So far, he has been doing this very well. Gabe sets up the meetings, monitors the work being done and facilitates release planning of our study application. He's a lot like a project manager, but that's such a boring title, so we'll call him Scrum Master in this instance.

The rest of the team has similar roles to other development processes. We all will be building the product, but Gabe and Marta have a good eye and experience to make sure everything is working together smoothly. But let’s get back to this idea of “Release Planning.” To plan a release, we start by looking at our product backlog and together we identify user-stories we want to put into our primary release. These user-stories then become part of our release backlog. The team then prioritizes the user-stories and estimates the amount of work involved for each item. In our case, the user stories are: Scheduler, Attendee and Administrator.

Sometimes larger user-stories are broken down into smaller more manageable chunks. In our case, the Admin has a lot of privileges, so when we discuss his user role, we break it down into smaller, more manageable aspects. We can even defer a large chunk of the Admin story to the role of the scheduler because the privileges of the scheduler is basically just a subset of the
administrator’s privileges, the only difference is the admin can ban users and manipulate privileges of all accounts. The collection of all these story/actor estimates provides a rough idea of the total amount of work involved to complete the entire release.

A quick side note about estimates. There are a lot of techniques for creating good estimates. Some prefer estimating in story points where estimates are made relative to building a small component with a known level of difficulty. Unfortunately, story points don’t answer the question of, “When will my project ship?” Gabe has mitigated this through his experience because he has found that the best technique is to estimate work in hours by use of standards in how he knows estimates to be done.

Table 4: Expected points per hours spent on task table

<table>
<thead>
<tr>
<th>Point Value</th>
<th>Time Bucket</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2 point</td>
<td>0-3 hours</td>
</tr>
<tr>
<td>1 point</td>
<td>3-6 hours</td>
</tr>
<tr>
<td>2 points</td>
<td>6-13 hours</td>
</tr>
<tr>
<td>3 points</td>
<td>13-20 hours</td>
</tr>
<tr>
<td>5 points</td>
<td>20-35 hours</td>
</tr>
<tr>
<td>8 points</td>
<td>35-55 hours</td>
</tr>
<tr>
<td>13 points</td>
<td>55-85 hours</td>
</tr>
</tbody>
</table>

For example, Gabriel has said things that take less than a day to complete will be estimated as 1 hour, 2 hours, 4 hours or 8 hours. Every item or task function we implement will fall into one of those buckets. There will be no 3 hour estimates, for example. A 3 hour item would fall into the 4 hour bucket. Larger items will be estimated as 2 days, 3 days, 5 days, or 10 days. Again, all estimates in between will fall into the next larger bucket. Extremely large items are similarly estimated in months: 1, 2, 3 or 6 Months, but the reality is that such items will need to be broken down substantially before work actually begins. A good example of this in our app is the overall design and feel of the app because it’s always going to be changing through minor tweaks. But, enough about estimates. We’ll come back to them in just a minute. For now, let’s go back to this:

The Release Backlog. With a prioritized set of user-stories and the estimated amount of work at hand, we are now ready to plan out several sprints to get the work done. So far we have estimated the workload of our app to be 4 months in total. To complete this work, we will do work in sprints. Sprints are short-duration milestones that allows us to tackle a manageable chunk of the project and get it to a ship-ready state for the class. Sprints generally range from a couple of days to as much as 30 days in length, depending on the product’s release cycles. We plan on doing two week sprints though this may change as the project progresses. We will have a few release cycles before the actual launch. The shorter the release cycles, the shorter each sprint should be. We hope to have at least 2 to as many as a dozen sprints per release. For us, a release will constitute as each of our milestones which are outlined earlier in this paper.

So, at this point, we can take our release backlog and split it up into several Sprint Backlogs. This backlog will help us keep track of what has been done, what works well and what doesn’t so we don’t keep revisiting the same problem out of error. One of the most important things to remember about our sprints is that the goal of each sprint is to get a subset of the release backlog
to a ship-ready state for you, the Professor. So, at the end of each sprint, we will have a fully
tested product with all the features of that sprint 100% complete. Since sprints are a very short,
but a realistic representation of part of the product, a late finish of the sprint is a great indicator
that the project is not on schedule and something needs to be done. Our Scrum Master, Gabe, has
been diligent as making sure we accomplish what needs to be done in a timely fashion.

It's extremely important to monitor the progress of each sprint. To do this, we will be going
along with industry standards: A Burndown Chart. The burndown chart is arguably the number
one reason for Scrum's popularity, and one of the best project visibility tools to ensure a project
is progressing smoothly. Our burndown chart provides a day-by-day measure of the amount of
work that remains in a given sprint or release.

Because historical information is provided in the burndown chart, it will be easy to see if the
team is on the right track. Using the burndown chart, we can quickly calculate this - the slope of
the graph, which is also called the Burndown Velocity. The Burndown Velocity is the average
rate of productivity for each day. For example, let’s say our team’s rate of productivity on a
typical day, finishes approximately 10 hours of work. Knowing that, it's possible to calculate an
estimated completion date for the sprint or even for the entire release, based on the amount of
work remaining. What's great about the burndown chart is that we can compare our actual
velocity and projected completion date to what the team needs to do in order to finish on time.
This is perhaps the most useful piece of knowledge that any team member, product owner or
product executive can have about the project, because knowing whether or not the project is on
track early in the schedule can help teams make the proper adjustments necessary to get the
project on track for later on down the line.

Figure 28: Graph of estimated, and real burndown vs points accomplished
The burndown chart provides empirical proof that the project is on track or if it's going to be late. So, let's talk a little about where the data for this incredibly useful burndown chart comes from. Hopefully, as you recall, part of the release planning process was to create an estimate for each user-story in the release backlog. The collection of these estimates for a given sprint represents the total amount of work that must be done to complete that sprint. As each team member goes through and makes progress on one or more of the user-stories, whether it be the scheduler attendee or the administrator, they simply update the amount of time remaining for each of their own items. So, the total amount of time remaining on the group of user-stories that make up a sprint, changes on a day-by-day basis, hopefully going downward until it hits zero when the sprint is complete. The burndown chart aggregates the remaining work data and shows it visually. It's rather brilliant because it communicates a massive amount of information in just a few seconds.

3 important questions

1. What work did you complete yesterday?
2. What have you planned for today?
3. Are you facing any problems or issues?

Figure 29: Three important questions of daily scrums

And that brings us to this: Our Daily Scrum. The Daily Scrum is an essential tool to having communication flow freely between team members. The idea is to have fast paced stand-up meetings where Martha, Gabriel, Sammy and Michael can quickly list the work we’ve completed since the last meeting, and any obstacles in our way. By meeting daily, it ensures we are always in-sync, and any major issues are dealt with as soon as they are known. Finally, as each sprint comes to a finish, it’s important to have a Sprint Retrospective meeting where we can reflect on what went right and expound upon areas of improvement. After all, Scrum is a flexible, agile development method that needs constant improving and tweaking for every team member.
Roles in Development

There will be a compartmentalization of responsibilities during our development process: each member of the team will be responsible for a different section of the project. Being responsible of a section does not mean that they are the only ones making any effort in that section, but instead that they are responsible that milestones regarding a section are met in a timely manner and report on any significant progress and roadblocks met. It is also important to note that there are some sections of the process that are not the responsibility of any one member, but the group as a whole. Such as the final deliverable, the documentation, and maintaining communication.

Some of the roles include Cordova, Web API, Mockups, Design, Database, Authentication and Security, and so on. In some cases, more than one member will be responsible for a segment.

The roles will be divided the following way:

**Gabriel Babilonia**

**Project Management**

When the projects were first proposed to the class, Gabriel introduced the idea of the Study Group App. Given that it was primarily his idea, it is only natural that Gabriel will act as project manager of our project. Gabriel chose to work as a project manager due to his interest in team dynamics and team leadership, he hopes to take what he has learned throughout this effort and apply it to future career opportunities. He has also demonstrated that he has the ability to lead a project and make decisions when needed be and the experience to know where the development process needs to be aimed at next.

**Authentication and Security**

Gabriel will be overseeing the authentication and security standards of the application. Primarily, the development of the API authentication layer will be conducted by Gabriel, thereafter, he will advise and oversee the client side authentication mechanisms employed by the API consuming application (Android, iOS, and the Web Application). Moreover, Gabriel will be providing final sign off on security testing by Michael.

**Web App Lead**

Gabriel has taken the Web App lead and API due to his past experience with the C# ASP.NET Web API platform. Gabriel intends to continue to build on his skill set by architecting and planning for a maintainable and scalable API that leverages the many benefits of the Azure Cloud.

**Martha Cornejo**

**Database**

Martha has proven to be quite knowledgeable in database design and its rulings during our initial design, which is why she has decided to take responsibility of this crucial section of our project.
iOS Development
Martha’s role in the project is co-developing the mobile applications with Cordova, particularly taking care of any iOS specific component. She took this role because she’s interested in mobile development and wanted to learn more about the languages that are used in the development of our project.

Michael Garber
Visual Design
Michael took this role because he is critical of how things look and feel. People will not use an application even if it has great functionality if the look and feel of the application is poorly designed.

Authentication and Security
Michael has had some experience in the past with setting up Authentication so this job was given to him. He worked on an app with a senior developer that involved location services and was given the task of authenticating a user. Since he has done it before, he was given the task of assisting Gabriel.

Quality Assurance
Michael will be in charge of the quality assurance in the development process to ensure everything is progressing in a manner that promotes pleasing visual aesthetics and a user-friendly interface.

Sammy Israwi
Design Guidelines
First of all, this section differs from the Visual Design section in a matter of topics covered. The visual design section takes the guidelines and designs established on this section and implements in on our project. This section will be responsible to researching and establishing design rules that our application will follow. Sammy’s mentors have always preached about the importance of intelligent product design, and some of that interest and knowledge has made itself evident on Sammy’s product design interests.

Cordova
Sammy has always been passionate about the development of JavaScript as a programming language that can be used on anything. Knowing that, one would understand why he would try to spearhead the development section of our project that will have to deal with using JavaScript on a mobile environment. Sammy has also been on many medium scale projects last year that dealt with different JavaScript frameworks and environment.
Android Development

I used to be an active member of the Association of Computer Machinery’s Mobile Special Interest Group (ACM SIGMobile), where he learned the foundations of mobile development. While his experience is in native Android development, most of the foundations of development translate easily into hybrid development.
Application Roles and Administration

Within our application, roles will be mainly divided into three sections: attendee, scheduler, and administrator. Each application user can be an Attendee or a Scheduler of any given study session, depending on whether or not they are in charge of said study session. An Administrator will be policing the behavior and use of our application. The details and responsibilities of each role will be further explained in the following section.

Figure 30: User case diagram of our project
User

The base role of every person using our application. A User is someone who has sign up for the application and created a profile. This user is not signed up to be attendee at a study session or a scheduler for a study session.

This user can browse for study sessions, can also view individual study session’s information, can also favorite study sessions and be able to view and access this selected favorite sessions from their profile.

Once the user selects a study session as attending, the user becomes an Attendee of that session and gets new privileges relevant to that session.

Similarly, once a user schedules a study session, they become the scheduler of that study sessions and gets new privileges relevant to that study session.

Attendee

An Attendee is the most basic application user. Any user that attends a study session is regarded as an Attendee. As an attendee, a user can comment on, and rate a study session that they are attending.

An attendee can also review or download any study material the Scheduler of the Study Session has uploaded and linked to the session in preparation for it.

When necessary, a user can also report a study session or members of the study session if they are misusing the application, or acting in a malicious or bothersome way.

An Attendee can be reported for actions like participating in a negative way, being loud, distracting, or offensive. They will be reported by any other user that observes bad behavior.

An Attendee can be punished when their bad behavior has been confirmed. After repeated offences, punishment may include suspension from the application.

Scheduler

A Scheduler is a user that has scheduled a study session and has therefore taken responsibility for its organization. The terms Organizer and Scheduler can be used interchangeably when referring to the user that has scheduled and is responsible of organizing a study session.

The Attendee and Organizer titles are not mutually exclusive; an Attendee of a session can be an Organizer of any other session.

A Scheduler can request the creation of an institution if they do not already exist. Schedulers can also request the creation of new Study Topics (e.g. “Foundations Exam” or “Java Certification Exam”) or Classes (e.g. “COT 3100 Introduction to Discrete Structures”) if they don’t already exist. An Administrator has to approve the request of a Scheduler for the Institution, Topic, or Class to be available to the users.
A Scheduler can upload any study material that they consider useful for the study session, so that Attendees can download or review the material to prepare for the study session.

A Scheduler is also responsible for the confirmation of the Study Session Attendees.

When necessary, a Scheduler will report the misbehavior of an Attendee. Such reports will be observed by an Administrator, who will take action when necessary.

A Scheduler can be reported if they have misused the application, taken advantage of their power as Schedulers, or uploaded non-relevant material to a session.

A Scheduler can be punished by removing their ability to schedule sessions, or even suspension of their ability to use the application.

**Administrator**

An Administrator is responsible for policing the behavior of all application users. This includes making sure that the application is being used correctly and the terms of use are not being violated, resolving issued between users, and work as a communications bridge between users and developers.

An Administrator’s area of responsibility is the school they are assigned to. For example, an Administrator of the University of Central Florida is not responsible for the violations of a user of the University of South Florida. However, it is the Administrator’s decision if they want to use their authority to help solve a problem in an area that is not their area of responsibility.

An Administrator has the responsibility of confirming or rejecting the creation of new Institutions, Topics, or Classes, as they were defined in the last section.

In some cases, an Administrator has the ability to merge, or cancel Study Sessions as they see fit.

An Administrator can be chosen from the general user base of an area. Since Administrators are responsible for policing the correct use of our application, they must have a solid understanding of the workings and terms of use of the application.

An Administrator has the ability to remove the ability of a user to schedule sessions, to attend sessions, or even suspend their ability to use the application altogether after repeated offences.
Milestones

These milestones will be used as a guideline for our development process. The deadlines for most of these tasks will be flexible, but the group members should do everything in their power to meet all deadlines.

If the deadlines are followed, by the end of the first semester all documentation and training will be done. In addition to that, the database will be fully designed and will have a connection to the back end Web API.

By the end of the second semester, the mobile application should be deployed to both intended platforms, and the web app and API will be deployed to Azure.

First Semester

The first semester is mostly for planning and training. However, we have decided to add some back end milestones to the first semester, such as design of the database.

<table>
<thead>
<tr>
<th>Milestone</th>
<th>Assignee</th>
<th>Deadline</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete training for C# Web API</td>
<td>Gabriel Babilonia</td>
<td>2/2/2016</td>
<td>Done</td>
</tr>
<tr>
<td>Complete training for Cordova IOS/Android</td>
<td>Sammy Israwi</td>
<td>3/7/2016</td>
<td>Postponed</td>
</tr>
<tr>
<td>Complete training for Azure cloud management essentials</td>
<td>Gabriel Babilonia</td>
<td>Date TBA</td>
<td></td>
</tr>
<tr>
<td>Setup Azure development environment</td>
<td>Martha Cornejo</td>
<td>4/1/2016</td>
<td>Postponed</td>
</tr>
<tr>
<td>Setup Github repository with multiple branches</td>
<td>Martha Cornejo</td>
<td>04/20/2016</td>
<td>Done</td>
</tr>
<tr>
<td>Approve final mock up design for iOS, Android, and Web Application platforms</td>
<td>Michael Garber</td>
<td>Date TBA</td>
<td></td>
</tr>
<tr>
<td>Mobile Wireframe part 1 - Not including Sign up and login page</td>
<td>Martha Cornejo</td>
<td>04/28/2016</td>
<td>Done</td>
</tr>
<tr>
<td>Mobile Wireframe part 2 - Not including Sign up and login page</td>
<td>Sammy Israwi</td>
<td>04/28/2016</td>
<td>Done</td>
</tr>
<tr>
<td>Web App Wireframe - Not including Sign up and login Page</td>
<td>Gabriel Babilonia</td>
<td>04/28/2016</td>
<td>Done</td>
</tr>
<tr>
<td>Login Pages Wireframes - Mobile and Web App</td>
<td>Michael Garber</td>
<td>04/28/2016</td>
<td>Merged with above</td>
</tr>
<tr>
<td>Milestone</td>
<td>Assignee</td>
<td>Deadline</td>
<td>Sprint</td>
</tr>
<tr>
<td>---------------------------------------------------------------------------</td>
<td>---------------------------</td>
<td>--------------</td>
<td>--------</td>
</tr>
<tr>
<td>Final Paper Delivery (Hard Deadline)</td>
<td>Gabriel Babilonia</td>
<td>04/28/16</td>
<td>Done</td>
</tr>
<tr>
<td>Prove of concept for C# Web API with OAuth 2.0 authentication</td>
<td>Gabriel Babilonia</td>
<td>06/15/2016</td>
<td>Done</td>
</tr>
<tr>
<td>Setup base model on API for consumption by clients</td>
<td>Gabriel Babilonia</td>
<td>06/15/2016</td>
<td>In progress</td>
</tr>
<tr>
<td>Prove of concept for Cordova deployment dev to Android and iOS with consumption of Web API</td>
<td>Martha Cornejo</td>
<td>06/15/2016</td>
<td>Done</td>
</tr>
<tr>
<td>Prove of concept for Web Application on dev front end using AngularJS with consumption of Web API</td>
<td>Gabriel Babilonia</td>
<td>06/15/2016</td>
<td>Done</td>
</tr>
<tr>
<td>Setting up Cordova dev environment</td>
<td>Sammy Israwi</td>
<td>04/03/2016</td>
<td>Done</td>
</tr>
<tr>
<td>Design Web API</td>
<td>Gabriel Babilonia</td>
<td>06/15/2016</td>
<td>Done</td>
</tr>
<tr>
<td>Design and implement authentication model</td>
<td>Gabriel Babilonia</td>
<td>06/15/2016</td>
<td>Not started</td>
</tr>
<tr>
<td>Design SQL relational database model</td>
<td>Martha Cornejo</td>
<td>04/28/2016</td>
<td>Done</td>
</tr>
</tbody>
</table>

### Second Semester

The second semester milestones are mostly dedicated to the development and testing of the application. By the end of the semester, the milestones are dedicated to the deployment of the application.

<table>
<thead>
<tr>
<th>Milestone</th>
<th>Assignee</th>
<th>Deadline</th>
<th>Sprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Develop sign up screen for IOS, Web application and Android platforms</td>
<td>Gabriel, Martha, and Sammy</td>
<td>09/04/2016</td>
<td>2</td>
</tr>
<tr>
<td>Develop login screen for IOS, Web application and Android platforms.</td>
<td>Gabriel, Martha, and Sammy</td>
<td>09/04/2016</td>
<td>2</td>
</tr>
<tr>
<td>Develop iOS, Android and Web App attending confirmation screens for study sessions</td>
<td>Gabriel, Martha, and Sammy</td>
<td>09/11/2016</td>
<td>3</td>
</tr>
<tr>
<td>Develop iOS, Android and Web App create, modify and delete study session functionality</td>
<td>Gabriel, Michael, and Martha</td>
<td>09/11/2016</td>
<td>3</td>
</tr>
<tr>
<td>Task</td>
<td>Responsible</td>
<td>Date</td>
<td>Sprint</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>----------------------</td>
<td>------------</td>
<td>--------</td>
</tr>
<tr>
<td>Develop filtering, browsing, and profile management screens for iOS, Web Application, and Android platforms.</td>
<td>Gabriel, Martha, and Sammy</td>
<td>09/18/2016</td>
<td>3</td>
</tr>
<tr>
<td>Develop push and email notifications feature for all platforms.</td>
<td>Gabriel, Martha, and Sammy</td>
<td>09/18/2016</td>
<td>3</td>
</tr>
<tr>
<td>End of sprint testing</td>
<td>Michael</td>
<td>09/21/2016</td>
<td>3</td>
</tr>
<tr>
<td>Develop user bad behavior reporting feature for iOS, Web Application and Android platforms.</td>
<td>Gabriel, Martha, and Sammy</td>
<td>10/02/2016</td>
<td>4</td>
</tr>
<tr>
<td>Develop administrator functionality on the web application.</td>
<td>Gabriel Babilonia</td>
<td>10/02/2016</td>
<td>4</td>
</tr>
<tr>
<td>Develop search strings for iOS, Web Application, and Android platforms.</td>
<td>Gabriel, Martha, and Sammy</td>
<td>10/02/2016</td>
<td>4</td>
</tr>
<tr>
<td>End of sprint testing</td>
<td>Michael</td>
<td>10/05/2016</td>
<td>4</td>
</tr>
<tr>
<td>Develop home page for iOS, Web Application, and Android platforms.</td>
<td>Gabriel, Martha, and Sammy</td>
<td>10/16/2016</td>
<td>5</td>
</tr>
<tr>
<td>Develop reporting screens</td>
<td>Gabriel, Martha, and Sammy</td>
<td>10/16/2016</td>
<td>5</td>
</tr>
<tr>
<td>Develop gamification for study sessions</td>
<td>Michael</td>
<td>10/16/2016</td>
<td>5</td>
</tr>
<tr>
<td>Develop scheduler canceling feature of scheduled session for iOS, Web Application, and Android platforms.</td>
<td>Gabriel, Martha, and Sammy</td>
<td>10/16/2016</td>
<td>5</td>
</tr>
<tr>
<td>Develop scheduler postponing of scheduled session for iOS, Web Application, and Android platforms.</td>
<td>Gabriel, Martha, and Sammy</td>
<td>10/16/2016</td>
<td>5</td>
</tr>
<tr>
<td>End of sprint testing</td>
<td>Michael</td>
<td>10/19/2016</td>
<td>5</td>
</tr>
<tr>
<td>Develop scheduled session comments area/feature for iOS, Web Application, and Android platforms.</td>
<td>Gabriel, Martha, and Sammy</td>
<td>10/30/2016</td>
<td>6</td>
</tr>
<tr>
<td>Develop badge system</td>
<td>Michael</td>
<td>10/30/2016</td>
<td>6</td>
</tr>
<tr>
<td>Apply branding such as images and icons (Finalize)</td>
<td>Sammy, Michael</td>
<td>10/30/2016</td>
<td>6</td>
</tr>
<tr>
<td>Task</td>
<td>Team Member(s)</td>
<td>Date</td>
<td>Notes</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>---------------------------</td>
<td>------------</td>
<td>-------</td>
</tr>
<tr>
<td>End of sprint testing</td>
<td>Gabriel, Martha</td>
<td>11/02/2016</td>
<td>6</td>
</tr>
<tr>
<td>Develop iOS master page</td>
<td>Martha Cornejo</td>
<td>08/28/2016</td>
<td>1</td>
</tr>
<tr>
<td>Develop Android master page</td>
<td>Sammy Israwi</td>
<td>08/28/2016</td>
<td>1</td>
</tr>
<tr>
<td>Develop Web Application master page</td>
<td>Gabriel Babilonia</td>
<td>08/28/2016</td>
<td>1</td>
</tr>
<tr>
<td>End of Sprint Testing</td>
<td>Michael Garber</td>
<td>09/01/2016</td>
<td>1</td>
</tr>
<tr>
<td>Fixing Bugs</td>
<td>Gabriel, Martha, Michael and Sammy</td>
<td>11/18/2016</td>
<td></td>
</tr>
<tr>
<td>Re-Testing</td>
<td>Gabriel, Martha, Michael and Sammy</td>
<td>11/22/2016</td>
<td></td>
</tr>
<tr>
<td>Submit to Google's Play Store</td>
<td>Sammy Israwi</td>
<td>11/23/2016</td>
<td></td>
</tr>
<tr>
<td>Submit to Apple's App Store</td>
<td>Martha Cornejo</td>
<td>11/23/2016</td>
<td></td>
</tr>
<tr>
<td>Deploy API</td>
<td>Gabriel Babilonia</td>
<td>11/29/2016</td>
<td></td>
</tr>
<tr>
<td>Deploy to Google’s Play Store</td>
<td>Sammy Israwi</td>
<td>12/02/2016</td>
<td></td>
</tr>
<tr>
<td>Deploy to Apple’s App Store</td>
<td>Martha Cornejo</td>
<td>12/02/2016</td>
<td></td>
</tr>
</tbody>
</table>
Deployment

By the end of the second semester, we plan to deploy our application on all intended platforms. For this to be able to happen, we need to know the steps and rules that we need to follow to deploy our mobile application to both Google’s Play Store and Apple’s App Store, the web applications, and the back end.

Android

Google has made available a very thorough list to follow to deploy applications on their Play Store and expose it to their user base. Items on the list include creating a signed, release-ready application package (APK), understanding the requirements of the app, and creating the product page and graphic assets for each of your apps (Android Developers, n.d.).

The preparation steps are divided into five sections: gathering materials, configuring the application, building the application, preparing remote servers, testing the application (Android Developers, n.d.).

![Figure 31: Visualization of the points to prepare for release (Android Developers, n.d.)]
Gather Materials
Material such as the EULA, application icon, and most importantly, the cryptographic keys are gathered in this step in preparation to making the final configuration.

Configure Application
In this step, we prepare the Study Group App for the Play Store by cleaning up any debugging code, choosing a package name, cleaning up the directories for dev dependencies, and updating the Android Manifest. The Android Manifest information contains a lot of the metadata of our application. Since we will be using Cordova, most of the reference information for this section will be gathered from the Cordova and Ionic documentation (Apache Software Foundation, n.d.), instead of the Android Developer documentation.

Build the Application
Once all the preparation is ready, we build the release-ready application using the command

```
cordova build --release android
```

The command will generate a release build based on the Android Manifest on our config.xml file. It is important to know that at this point, the generated APK is unsigned. We will sign it in the next step.

To sign our application, we will use the keytool command line tool to generate a RSA key for our application. Here’s an example of how to create an RSA key

```
keytool -genkey -v -keystore my-release-key.keystore -alias alias_name -keyalg RSA -keysize 2048 -validity 10000
```

We then use the jarsigner tool to sign our application with the RSA key that we generated, like this:

```
jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.keystore HelloWorld-release-unsigned.apk alias_name
```

Finally, we run the zipalign CL tool to optimize the APK, like so:

```
zipalign -v 4 HelloWorld-release-unsigned.apk HelloWorld.apk
```
At this point, the generated APK is ready for publishing on the Google Play store (Drifty Co, n.d.).

**Prepare Remote Servers**
On this step, we make sure that our production Azure servers are ready for release, all API endpoints work as expected, and so on.

**Testing the Applications**
Because there is no successful application without testing, we need to test that the final release-ready application is actually ready to be exposed to the Android market.

The Android Developer website contains a list of ‘What to Test’ to make sure that our app is as functional as it can be and up to production standards (Android Developers, n.d.).

Items on the list include, but are not limited to:

- **Visual Design and Interaction**
  - Standard Design
  - Navigation
  - Notifications
- **Functionality**
  - Permissions
  - Install Location
- **Performance and Stability**
  - Battery
  - Media
  - Performance
  - Stability
- **Google Play**
  - App Details
  - User Support
  - Metadata
  - Policies
  - Installation

**Releasing to Google’s Play Store**
This is not a step on the Android Developer’s website. However, the last step on how to deploy an app on the Play store, is to deploy the app on the Play store.

To start, we will need an Android Developer account, which costs $25. We can create one from the Google Play Store Developer Console.

It is important remember that the Google Play Store Developer Console will be the dashboard for our application on the Play store.
Once we have a developer account, we press the button as shown above, and follow the steps to upload and publish the application on the Play store.
Once all those steps are followed, the application has been successfully published to the Google Play store and is being exposed to its user base.

**Updating the Application**

To update an application, almost all steps need to be the same. The developer needs to build a new unsigned APK, then sign it and zip it. However, a new key should not be used to sign the new APK, since this is how the Google Play Store Developer Console will confirm that the app being uploaded is an update of the current app. Instead, one must use the same key that was generated when the app was initially uploaded.

We must also remember to increase the ‘version’ value on the Android Manifest to label the new APK as an update of the application.
iOS

Similarly, to Android, Apple provides a set of guidelines to follow in order to deploy the application to the App Store. We will be going through most of the steps shown below but since we have decided to work with Cordova or development well be in Cordova instead of Xcode.

(Apple, About App Distribution Workflow, 2016)

![Enrollment, Development and Distribute process of iOS app. Instead of Xcode we will be using Cordova](image)

**Steps for deployment in iOS environment:**

1. **Enrolling in the Apple Developer Program.**
   
   Only by enrolling to the Apple Developer Program can we build and test our application on multiple devices, and see how the application works before submitting and releasing the application. Also as a member of the Developer Program, we have access to more features such as iWallet, GameCenter, TestFlight, and different types of application distribution. For the time being we do not plan to use any of the extra features made available for register Apple Developers, therefore we will be joining the Apple Developer Program in the second semester.

2. **Upload metadata about app so the store can present it to customers.**
   
   The design of the application plays a major role on how retaining and gaining users, the application logo and name will be created to make our application memorable. But nearly as equally the profile and design of the application on the App Store plays a big role on the way potential users perceive the application. Before we deploy our application we want to have an attractive application profile on the app store.

3. **Verify that we’ve prepared the app correctly, upload the build and submit it to the store.**
We have planned to test as we develop, but once the application is complete and we have achieved our set developmental goal we can solemnly focus on testing and debugging. Once testing and debugging is finished, we will prepare our app for submission to the app store. Apple provides review guidelines that the app is reviewed on by apple itself before accepting and allowing the app to be published on the app store.

Apple also provides TestFlight (Apple, TestFlight data testing, 2016) Beta Testing, which allows developers to invite up to 2,000 users who are not part of the development team to test the application and provide feedback. Before using TestFlight the application must first meet the review guidelines. When a tester receives an invitation email, they can download the TestFlight Application from the App Store and test the application. After has install the application they will also be able to test new versions that are released. For our project it would be a great benefit to have the time to use TestFlight and invite our classmates to use the application and gather feedback before the application is submitted and published to the App Store.

Also in order to submit to the App store we need to set our final application name, icon, and which app category our application fits the best in, at this moment our main category candidates are Education, Productivity and Social Networking. At the end of the semester we should have a clear idea of where or application will do best and fits best into. Finally, before submitting the application the version needs to be selected, and an apple provided questionnaire about encryption, content rights, export compliance and Advertising Identifier (IDFA)

(Apple, Submitting the App to App Review, 2016)

![Figure 35 Example of screen displayed for application submission for Review.](image)

4. Release and maintain app after submission.

We will be notified if the application is denied or accepted, if the application is accepted we can choose the date to release the application or allow apple to release it. Once the application is published, we can release new versions that contain some of our stretch
goals or improvement made based on the farther testing and simply replace the existing version with the new one. (Apple, Replacing your app with a new version, 2016)

Figure 36 Example of screen displayed when a new version of the application is about being released.
Web API
In this section we detail the steps to deploy our back end Web API from Visual Studio.

Deployment
The deployment procedure of the Web API will require the proper setup of transformable app configuration files. The API will contain multiple deployment configurations, such as debug, development, testing, and production. Before a deployment is conducted to either Azure environment, the following checks must be conducted:

1. Ensure test or prod configurations are present on the transform config file.

![Figure 37 - Web API Configs](image)

2. Check that each configuration file has the correct database connections. Note: It’s easy to miss this check, ensure that the server and database name match your deployment environment. This will avoid a case by which test is deployed using the production database connection.

![Figure 38 – Web API Config File Database Connections](image)

3. Before publishing the API, please run a clean, then build of the project to ensure the build configurations are up to date with the latest changes.

![Figure 39 - Building Project after Config Changes](image)
4. Ensure that relevant controllers have the proper security team. Visit each controller, and check the authorization attributes. This attribute is at the very top of the controller, just after the namespace declaration.

![Figure 40 - Controller Auth](image)

5. Ensure that SSL is enabled on the project. Navigate to the Solution Explorer - > enable the properties window - > click on the project - > set SSL to true.

![Figure 41- Enabling SSL](image)

6. Go to Azure, under the web API app, download the Publish profile. The publish profile holds web deployment credentials such as the publish URL, publish username and password and other details that allow us to run an automated deployment to the web app API instance.
7. Publish to Azure by navigating to the Solutions Explorer - > right click on the project - > click Publish:

8. Once on the publish screen, import the publish profile that was downloaded. Check that the URL matches the name of the instance where the deployment will take place (Ensure that if you are deploying to test, that the tst instance on the URL is present). Other details
that might require revision are the database connections strings under the Settings tab, ensure that none of the options are checked.

The steps above should be followed in order to ensure proper deployment of the web app. They should be the only sequence of procedures that is ran to deploy the Web API.

![Figure 44 - Publish Web App Screen](image)

9. Once all details are checked, click on “Publish”. Visual Studio will then compile the application and publish all files to the web app instance. Once the process has been completed, check the Output window to ensure that a success message is present.
Database
The deployment of the SQL database will depend on the initial development procedure. If a local instance is used to develop the database vs developing the database using the Azure SQL instance.

Deploying Local Instance:
To deploy a local instance of the database please follow the below listed steps:

1. Export the database to a .BACPAC file. You may do this by performing a right click on the database object on SQL Server Management Studio, then clicking on Task – Export Data Tier Application, and following the onscreen prompts that follow thereafter.
2. Log into Azure, and find the Server instance where the Azure SQL database is stored, for this instance, this would more than likely be dev. Once on the server, click on Firewall on the right hand side. A new blade will open to the right. Click on “Add client IP”. This will ensure that your PC is able to log into the database server on Azure.
3. Return to MS SQL Server Management Studio, click Connect - > Connect Database Engine. Once the prompt comes up, enter the server URL, which could be found on the Azure Server blade in the portal, alongside the client user name and credentials.
4. Once the server loads on the studio, right click on the Azure server, then click on Import Data-Tier application. Choose the BACPAC file that had been exported previously, and follow the on screen prompts. Ensure that a success message is show at the end of the prompt, that will conclude the deployment of the database to Azure SQL.

Deploying Azure SQL (Non Local Instance)
Deploying an Azure SQL instance is not required. All changes made to the database will be directly to the hosting server instance. To move databases from one instance to another (between dev, test, and prod), please follow these procedures:

1. Log onto the Azure Portal, navigate to the server that is hosting the database, once on the main blade, click on Copy.
2. Once the Copy blade appears, select the name of the test or prod database for the Database Name. Choose the server corresponding to the deployment instance, then click Ok. The replication will take about 10 minutes, please ensure to check back to the Portal for success message.
Web App

To deploy the web application, we’ll be leveraging Azure’s Git Deployment feature. Through the Azure Portal, we can effectively setup a connection between the Web App instance and our GitHub web app repository. Azure will then compile the application or run any of our grunt.js tasks before the deployment is completed. To complete this deployment, the below steps must be followed:

1. Ensure that the latest version of the application has been committed successfully to the GitHub web app repository.
2. Navigate to the Azure Portal and to then to the Web App instance that the deployment will take place on, for instance, if deployment to the test environment, navigate the groupstudytst web app.
3. On the main blade, click on settings. Under the Settings blade, look for Deployment Source.
4. Under Source, choose GitHub. Select the Authentication tab, and enter the credentials for the GitHub account (Personal credentials).

5. Choose a performance test setup. You may leave the settings as default. The performance test will run as soon as the deployment of the web app has been completed, it will ensure that the app is in a healthy state and ready to handle traffic demands. Please also ensure that a successful message is presented upon completion of the deployment.
SendGrid

SendGrid’s deployment from development to production only involves the changing of the API config file. Ensure that the SendGrid API key is switched to that of the corresponding SendGrid instance (dev, tst, or prd).

```javascript
var apiKey = "your_sendgrid_api_key";
// create a Web transport, using API Key
var transportWeb = new Web(apiKey);
```

Azure Blob Storage

Once a new version of the Web API is ready to be deployed, please ensure that the config files on the API are set to dynamically switch between instance (dev, tst, or prd). Azure Blob Storage’s connection string configuration must change from the corresponding instance where deployment will take place, this is rather similar to SendGrid’s procedure. Please avoid the error of deploying the test API instance with the production Azure Blob Storage.
Wireframes
The goal of our project is to make it easy for users to find and create study sessions. With this in mind, we have created wireframes for our mobile and web applications. Each view displays the fundamental input fields need to find or create a study session and the important details to display information about a study session. The mobile wireframes are explained in detail, and their corresponding website views are also shown below.

As a team we decided to use Material Design for our mobile and web application which allows the users to easily switch between platforms and keep the same design format. Material Design uses bold colors, shapes, shadows and textures to give visual cues about pop-up menus, drop down menus, or simply show or jump to a new view. An example of this is the floating option button, characterize by a colorful round plus sign icon that can be present on either mobile or web view. By selecting the floating option icon, more options are displayed as a list or as a menu bar above or at the bottom of the current screen. The floating option button as well as other features of Material Design allow the applications to be easy to use and visually attractive.

Mobile Platform

Main User View
- View will be displayed when user opens the application.
- User can easily view and access study session that are marked as going, favorite, past attended.
- Floating action button (plus sign), when selected by the user more options are displayed such as create study session, search study session.
- By selecting the top left icon, the Favorite Topic View will be displayed.
- By selecting the top right icon, the user can access application settings.
**Favorite Topic View**
- View will display badges the user has acquired.
- List of Favorite Topics is displayed, list can include class, subject or university user wants to easily search for or be notified about.
- User can swipe left to return to **Main User View**.

**Main User View – Past**
- View displays list of past study sessions that the user attended.
- User can swipe on individual study sessions to show the option to rate study session. Can also select review icon and open a questionnaire to review the session.
- Floating option button when selected shows more option such as create study session, search study session.
Create Study Session View
- Input titles will minimize and move up when the user enters data.
- Description input section size can grow or become scrollable if users data is larger than given space.
- Downward arrows next to date and time input fields will open dialog picker to select the time and date. User can also input the data manually.
- Radio buttons allow the user to pick one option.
- Selecting the check mark icon on the top right corner submits and creates a study session. Return to Main User View.

Figure 51  Create Study Session Mobile View
Study Session Info View

- Floating option button would display more options such as Join Event, Favorite Event, or share event.
- Attending users can be shown on a horizontal scrollable list.
- Date, Location, Location type and comments are displayed.
- Comments section is scrollable.
- Select the upper left corner back arrow to return to previous page where study session was selected from such as Main User View and Search Results View.

Search Study Session View

- Search for study session by scheduler username, tags on location, class name, topic etc.
- Individual selected tags are removable.
- Date and Time can be inputted manually or by selecting the downwards arrow which will open dialog picker for date and time.
- Location type are checkboxes, user can select multiple options.
- Select check mark icon on the top right corner to search study sessions and view results (Search Results View).
Web Platform

The design for the website is similar in look and feel to the mobile version. This is done so that the user can as easily interchange between the mobile application or the web application. The Website is also designed using Material Design, which as previously stated uses colors, texture, shadows and shapes to hint to the user that more content can be revealed with a hover of the mouse or a click. The ultimate goal is allowing the user to easily create and find study sessions.

![Log In page Web View](image-url)
**Main User View**

*Figure 55  Log In page Web View*

**Favorite Topics View**

*Figure 56  Favorite Topics Web View*
Create Study Session View

Figure 57 Create Study Session Web View

Study Session Info View

Figure 58 Study Session Web View
Advance Study Session Search View

Figure 59 Advance Study Search Web View

Search Study Session – Search bar

Figure 59 Search Study bar Web View
### Search Results View

<table>
<thead>
<tr>
<th>Filter Results</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nearest Date</td>
<td>Topic hosted by Scheduler Date</td>
</tr>
<tr>
<td>Latest Date</td>
<td>Topic hosted by Scheduler Date</td>
</tr>
<tr>
<td>Most Attendees</td>
<td>Topic hosted by Scheduler Date</td>
</tr>
<tr>
<td>Least Attendees</td>
<td>Topic hosted by Scheduler Date</td>
</tr>
<tr>
<td>In Person</td>
<td>Topic hosted by Scheduler Date</td>
</tr>
<tr>
<td>Virtual</td>
<td>Topic hosted by Scheduler Date</td>
</tr>
<tr>
<td></td>
<td>Topic hosted by Scheduler Date</td>
</tr>
<tr>
<td></td>
<td>Topic hosted by Scheduler Date</td>
</tr>
</tbody>
</table>
Design Guidelines

In this section, we discuss the various options we have when laying out the design guidelines. From the basic design layout (Material) to the iconography and the application logo sketches that we have come up with so far.

Material Design

Material design is Google’s answer when they first challenged themselves to create a visual language synthesizes the classic principles of good design with the innovation and possibility of technology and science. Material design allows for a unified experience across platforms and devices, which is why it is perfect for both our Mobile and Desktop views.

Material design is well documented online, and therefore the team will have no trouble finding references or guidelines regarding the design itself or its ideology. If the team differs from the basic guidelines, then they will do so in a manner that still follows the principles and goals of Material design.

Principles

Material Design has three ideological principles that it follows:

*Material is the metaphor*

The material is grounded in tactile reality, inspired by the study of paper and ink, yet technologically advanced and open to imagination and magic.

*Bold, graphic, intentional*

The foundational elements of print-based design – typography, grids, space, scale, color, and use of imagery – guide visual treatments.

*Motion provides meaning*

Motion respects and reinforces the user as the prime mover. Motion is meaningful and appropriate. Feedback is subtle yet clear. Transitions are efficient yet coherent.

(Google, n.d.)

These principles allow Material design to have the user as a prime subject of attention, providing them with beautiful but non-distracting design. When developing the UX of our application, we will have to always keep in mind these principles if we are to follow Material Design.
Icons

Since none of the members of our development team is particularly skillful at design, we will use the free collection of icons provided by Font Awesome and Ionicons as we see fit. Most of general use icons, such as Menu, Next, Email, and so on, are available on both collections. Both collections are free for commercial use, so we will not have to worry about that.

Figure 60: Some icons from Font Awesome

Font Awesome is highly used and gives us a wide array of customization available through CSS. Since they are vector images, they scale without any pixilation issues.

Figure 61: Some icons from Ionicons

Ionicons is made by Ionic, therefore it is well integrated with Ionic and Cordova. Ionicons change color automatically when the background they are placed in changes, and Ionic knows where to position and size them if one uses the correct markup when developing with Ionic.
Logo
As mentioned before, neither of the developers is particularly skillful in digital design. However, Sammy managed to sketch up some basic ideas for the main logo design. These are by no means final designs, and should be regarded as very early sketches.

Study Worms
Depicts a group of worms that will study. It is supposed to be a play on the name "Book Worm". The warm colors go against the standard cool colors that are usually used in mobile icon design. This design works well with square frames, but might need cropping for round frames.

![Study Worms Icon](image)

Figure 62: Study Worms Icon

Blue Circles
The circle in the middle depicts an institution, like a university. The groups around it are study groups. This design works well with both round and square frames. The use of cool colors is standard in mobile icon designs.

![Blue Circles Icon](image)

Figure 63: Blue Circles Icon
Table With Books
A table with notes and books, seen from the top. Unlike the next icon idea, this design this table does not have students around it to avoid too much detail. We must remember we are making an icon for displays that generally are not bigger than half inch by half inch. This design would work great with square frames, but not as well on round frames.

![Table With Books](image)

*Figure 64: Table With Books, no students*

Table With Books and Students
Very similar to the design above, but with students sitting around the table. On the final design, there will be less detail on the table to drive more attention to the students around the table. This design would work great with square frames, but not as well on round frames.

![Table With Books and Students](image)

*Figure 65: Table With Books and Students*
Book With Pencil

A book with some notes and a pencil in the middle. The design will have notably different handwriting to show that there are different people working on their notes. This design will need some modifications if it is to work with either square or round frames, since right now it fits neither of them.

Figure 66: Book With Pencil
Database

The database for the Study Group project was designed on Vertabelo, an online database design tool that allows multiple user to edit in real time and easily share and communicate during the design process.

The database was design to conform to the normalization rules, this way allowing for data insertion and retrieval to be faster and easier as well as reducing data redundancy in the database.

The database for our project will be hosted on Azure services. The Azure Storing plan we will be using is a Locally Redundant Storage (LRS), this type of data storage makes three copies of the data within a single datacenter, according to Microsoft this type of data redundancy is mostly used for economical local storage or data governance compliance. With our Azure plan we will have 1 TB of BLOB storage a month. If one storage accounts memory is exceeded, another account can be added to share and expand memory.

Azure also has a Managing portal which shows an overview of the database, gives information about the database for the administrator and has a design option. The administration option shows information about the database, such as when it was created, how many users it has, how many connections it has, the current maximum size, how much space has been used and what percentage is free. The design option works just like SQL where an administrator can add tables, columns or edit the database. Azure Managing portal has many features to interact and modify the database as needed, if a feature is needed and is not in Azure, a SQL server can be hosted on a virtual machine in Azure which allow us to use any feature that SQL server has.

Azure also lets the developers monitor performance for the cloud services, in our case, monitor the performance of our web and mobile applications. By default, some performance monitoring is set and displayed as graphs and tables, such as CPU percentage, Data In, Data Out, Disk Read Throughput and Disk Write Throughput. By customizing the monitoring features we can add and monitor many more important performance data from our applications such as query performance and data traffic. We can also receive notifications when a service such as memory or CPU performance reaches a certain number. The data that is set as default and configured by the developer to be monitored can be set to display in graphs and tables. Azure monitoring service stores the data at time intervals of 5 minutes, 1 hours and 12 hours. The data is stored in one or multiple storage accounts; the data is saved for 10 days before being erased.
Figure 67 shows different default and customer performance data being displayed in charts and tables.

Azure provides Data Services which has several ways to store and manage data, such as SQL Database, Tables, BLOB service and SQL Reporting. For our project we will be using SQL Database and Blobs. SQL Database allows for a new database to be created or for an existing database to be imported and hosted on Azure. For our project we will be importing our database. In order to import a database into Azure SQL database, a BACPAC file is created which contains the database schema and data, a BACPAC file is a zip with an extension .bacpac. Then the BACPAC file will be stored in BLOB container on a selected storage account. Once the BACPAC is saved, a new database can be created from the database saved on the BACPAC file. The administration for the database on Azure SQL database is as follows, SQL database allows the developers to have control over database administration and access, the hard drives, servers and storage is managed by Microsoft.
The second form of storage that we will be using is a BLOB service, which can hold large binary files of data in the cloud. The most common storage usage of BLOB service according to Microsoft Azure documentation are, serving images or documents directly to a browser, storing files for distributed access, Streaming video and audio, performing secure backup and disaster recovery and storing data for analysis by an on-premises or Azure hosted service. For our project we will be using the BLOB storage service to store users uploaded .pdf files, image files, word documents etc. The different types of files stored in the Blob storage are grouped in specified containers inside the Container component, for example the multiple image files stored in the Blob Storage, are grouped in a container specific for storing image type files. All Blobs must be part of a container, and a Container is part of a Storage Account as shown below.

(Get started with Azure Blob storage using .NET , 2016)

Figure 68- Diagram shows an example of the components of Blob storage service.
Database Schema for Study Group Application.

Figure 69: Database visual model
Database SQL scripts

-- Table: tbl_Adfs
CREATE TABLE tbl_Adfs (  
aID int NOT NULL,  
tbl_User_email nvarchar(255) NOT NULL,  
active bit NOT NULL,  
CONSTRAINT tbl_Adfs_pk PRIMARY KEY (aID)  
);

-- Table: tbl_Attendee
CREATE TABLE tbl_Attendee (  
aID int NOT NULL,  
tbl_User_email nvarchar(255) NOT NULL,  
tbl_Session_id int NOT NULL,  
status nvarchar(255) NOT NULL,  
wantNotifications bit NOT NULL,  
recentlyJoined bit NOT NULL,  
CONSTRAINT tbl_Attendee_pk PRIMARY KEY (aID)  
);

-- Table: tbl_Badgesheld
CREATE TABLE tbl_Badgesheld (  
bhID int NOT NULL,  
tbl_SysBadges_sbid int NOT NULL,  
dateAssigned datetime NOT NULL,  
comment nvarchar(MAX) NULL,  
tbl_User_email nvarchar(255) NOT NULL,  
CONSTRAINT tbl_Badgesheld_pk PRIMARY KEY (bhID)  
);

-- Table: tbl_Changes
CREATE TABLE tbl_Changes (  
cID int NOT NULL,  
sid int NOT NULL,  
title nvarchar(255) NULL,  
startDate datetime NULL,  
location nvarchar(255) NULL,  
endDate datetime NULL,  
topic nvarchar(255) NULL,  
description nvarchar(MAX) NULL,  
changeNotificationSent bit NOT NULL,  
CONSTRAINT tbl_Changes_pk PRIMARY KEY (cID)  
);
-- Table: tbl_Comment
CREATE TABLE tbl_Comment (  
    comment nvarchar(max) NOT NULL,  
    tbl_session_sID int NOT NULL,  
    cID int NOT NULL,  
    dateCreated datetime NOT NULL,  
    tbl_Attendee_aID int NOT NULL,  
    CONSTRAINT tbl_Comment_pk PRIMARY KEY (cID) 
);  

-- Table: tbl_Complaint
CREATE TABLE tbl_Complaint (  
    cID int NOT NULL,  
    raisedFor nvarchar(255) NOT NULL,  
    dateCreated datetime NOT NULL,  
    actionTaken nvarchar(255) NULL,  
    status nvarchar(255) NOT NULL,  
    CONSTRAINT tbl_Complaint_pk PRIMARY KEY (cID) 
);  

-- Table: tbl_Mention
CREATE TABLE tbl_Mention (  
    mID int NOT NULL,  
    userMentioned nvarchar(255) NOT NULL,  
    userPost nvarchar(255) NOT NULL,  
    tbl_Comment_cID int NOT NULL,  
    CONSTRAINT tbl_Mention_pk PRIMARY KEY (mID) 
);  

-- Table: tbl_Organizer
CREATE TABLE tbl_Organizer (  
    tbl_User_email nvarchar(255) NOT NULL,  
    oID int NOT NULL,  
    CONSTRAINT tbl_Organizer_pk PRIMARY KEY (oID) 
);  

-- Table: tbl_Session
CREATE TABLE tbl_Session (  
    sID int NOT NULL,  
    title nvarchar(255) NOT NULL,  
    startDate datetime NOT NULL,  
    location nvarchar(255) NOT NULL,  
    endDate datetime NOT NULL,  
    topic nvarchar(255) NOT NULL,  
    description nvarchar(max) NOT NULL,  
    locationType nvarchar(255) NOT NULL,  
    tbl_Organizer_oID int NOT NULL,  
    CONSTRAINT tbl_Session_pk PRIMARY KEY (sID) 
);
CREATE TABLE tbl_SessionTags (  
stID int NOT NULL,  
tbl_Session_sID int NOT NULL,  
tbl_Tag_tID int NOT NULL,  
CONSTRAINT tbl_SessionTags_pk PRIMARY KEY (stID) 
);

CREATE TABLE tbl_SysBadges (  
sbID int NOT NULL,  
title nvarchar(255) NOT NULL,  
criteria nvarchar(255) NOT NULL,  
image_src nvarchar(255) NOT NULL,  
CONSTRAINT tbl_SysBadges_pk PRIMARY KEY (sbID) 
);

CREATE TABLE tbl_Tag (  
tID int NOT NULL,  
tagName nvarchar(255) NOT NULL,  
CONSTRAINT tbl_Tag_pk PRIMARY KEY (tID) 
);

CREATE TABLE tbl_User (  
id int NOT NULL,  
email nvarchar(255) NOT NULL,  
firstName nvarchar(255) NOT NULL,  
lastName nvarchar(255) NOT NULL,  
emailConfirmed bit NOT NULL,  
passwordHash nvarchar(MAX) NOT NULL,  
securityStamp nvarchar(MAX) NULL,  
phoneNumber nvarchar(255) NULL,  
phoneNumberConfirmed bit NOT NULL,  
twoFactorEnabled bit NOT NULL,  
lockoutEndDateTimeUtc datetime NULL,  
lockoutEnabled bit NOT NULL,  
accessFailedCount int NOT NULL,  
displayName nvarchar(255) NOT NULL,  
reputationPoints int NOT NULL,  
hasProfilePicture bit NULL,  
profilePictureSource nvarchar(255) NOT NULL,  
emailNotificationPreference nvarchar(255) NOT NULL,  
badges nvarchar(255) NULL,  
CONSTRAINT tbl_User_pk PRIMARY KEY (email) 
);
High level description of database tables

Tbl_Attendee

Attendee table is a linking table that contains data about which study sessions the user signed up for. The table hold the user’s unique ID that comes as a one-to-many from the User table, since a user can be an attendee for many sessions. The table also holds the session ID that comes from the sessions table as a one to many relation, since one session can have many types of attendees. Another field in the table is Status, which will be set to going, favorite, or past depending on what the user chooses and if the session already passed. The field wantNotifications are fields set by the user if they want to receive notification about the current session. The number of attendees for a session cannot be predicted, neither the number of time a user will become an attendee to different session. Therefore, we use the attendee table to normalize the data and avoid redundancy.

Tbl_Organizer

Organizer table is a linking table that contains data about who is the organizer for a study session. We will be using this table for keeping track of the number of sessions an organizer sets up. The relation between the organizers table and the users table is one-to-many from the users table to organizers table, because one user can be an organizer to more than one session. The relation between the organizers table and the Sessions table is one to many from the organizers table to the sessions table, since each session can only have one organizer and an organizer can have many sessions.

Tbl_Session

The Sessions table, will be populated when an organizer creates a new study session. The fields in the table are title, start date, end date, location, topics, description, location type and
organizers ID. All the fields except description are required when creating a study session. As previously explained the Session table has relations to the Organizers and Attendee table. The Sessions table also has a one-to-many relation to the Comments table, which will store all the comments made by the users on all the study sessions. Another relation, the Sessions table has a one-to-many to the tags table, this is because each session can have any number of tags and in order to conform to the normalization rules, tags need to have its own table that is connected to the sessions table. Lastly, another relation the Sessions table has is a one-to-many relation to table changes, this relation is made to notify users of changes made to the study session.

Tbl_Comment
The Comment table is a linking table between the Session table and Mention table. The comments table is used to store the comments of all the study sessions, since a single study session can have zero to multiple comments it was implemented as a linking table that will store each comment in one entry. The comments are traced back to the study session where they were made by having the sessions ID as a foreign key and an attendee ID that made the comment also as a foreign key. Additionally, the date and time of when the comment was made is saved in the table.

Tbl_Mention
The table Mention is populated when a user creates a comment on a study session and in that comment they mention one or more other users. In order to comply to the normalization rules, more specifically comply to the first normal form which states that each column must contain one value (one mention) and contain no reappearing groups (not reappearing mention columns). User’s mentions are handled on its own table instead of having mentions as a column in the Comment table since an unknown number of mentions can be made in a single comment. Each user mentioned in a comment will have its own entry (row) in the Mention table. The user mentioned will be furthered traced back to the comment by having the comment ID as foreign key in the mentions table.

Tbl_Tag
The columns for the Tag table are the tag name and tag ID. The table Tag will have a single entry for each tag created by the user or admin. The purpose of this tag is to keep a record of all the tags and help us as developers to set up a tag recommendation system. For example, when a scheduler is creating a study session, as they are entering the topics which are hash tagged text, the user will get a list of recommendation for tag completion. If the user enters a new tag not on the tag table, this tag will be inserted to the tag table.

Tbl_SessionTags
The Sessions Tag table was created to handle, one to many tags that can be set as part of the topics section. All the tags that are set for the study session will have their own entry on the table. The tag will be traced back to the session it was create on by having the session ID as a foreign key in the table. The Session tags table is also linked to the tag table by a one to many from the tag table to the sessions tag. This is to verify and only store the tag that exists and to make it more convenient to search.
**Tbl_SysBadges**

The table System Badges has four columns ID, title, criteria, image. The table is used to store the different types of badges that can will be available for the users. The badges will be used as a form of reward system, each badge will have an image and criteria, this can be reaching a certain amount of sessions attended or amount of sessions hosted as scheduler etc. The System Badge table will mostly be used to search and display available badges.

**Tbl_BadgesHeld**

This table has five columns, ID, systemBadgeID, username, comment and date. This table is a linking table between tbl_Users and tbl_SysBadges. This table is implemented as a linking table because the number of badges that can be assignment to a user is unknown and unpredictable. Therefore, to properly handle this problem a single table specifically for badges assigned is created to easily search, insert and display badge records.

**Tbl_Complaint**

This table has five columns, ID, raisedFor, dateCreated, actionTaken, status. This table is used to keep a record of a complain set by a user. The table will store the ID of the user who's the complaint is about. The status value could be, complaint submitted, complaint processes etc. The user who set the complaint could also get a notification once the complaint is resolved. Based on the result of the complaint the account of the user which the complaint is about could have their account removed or suspended.

**Tbl_Admin**

The table admin has three columns, ID, username and active. The table admins is used to store the users who become admins, the status based on its value will indicate if the admin is a current admin

**Tbl_User**

The table user holds the users profile information such as username, name, last name, profile image etc. Aside from the general information needed to create a user account, the table also includes fields such as phone number and phone number confirmed which will be used to confirm and/or reset a user account. AccessFailedCount will be an integer that is incremented every time a user enters the wrong password, the field will be used to lock the account after too many failed log in attempts. The reputationPoints field keeps the number of reputation points earned by the user. In order to keep the users table normalized we created tables for some of the user’s information, such as badgesHeld table, complaints and admin.

**Tbl_SysInstitution**

This is considered a system table, as its only modified by the application process and administrator oversight (Adding of new institution to the application). The table will store the available institutions.
Tbl_SysCourse
This table is considered a system table, as it will only be modified by admins when loading new courses. Each course will contain a name, a code (ie. CAP 5544), whether or not is active by means of an active bit column, and a foreign key constraint that ties to the tbl_SysInstitution. The constraint relationship will be of 1 to many (Many courses match to only one institution).
Technologies Used

This section will be used to list the technologies used on the development of our project. We will list development tools such as Git and Github, to administrative tools such as Asana, design tools such as Mock Ninja, +and product frameworks and tools such as Cordova and Angular.

Git and Github

We will use Git as our version control system and code repository. Unlike SVN and other version control platforms, Git is the most widely used and is the easiest to understand. We will use Github as our Git project repository host service. Since all project members have Github Student accounts, there are more than enough private repositories at their disposal, therefore there will be no issues with code proliferation.

Cordova

Cordova will be the hybrid development engine that will allow us to develop an application on both of our intended mobile platforms using a single code base. Cordova will allow this by grabbing the mobile web application and host it in a Webview in either of our applications of choice. While not all the developers know how to use Cordova, it is relatively easy to pick up, and we will host as many learning sessions as necessary for the required people to learn how to properly use Cordova. For more detailed information, refer to the Cordova section.
Ionic

![Ionic Framework's logo](image)

Ionic is a development framework mobile applications that uses Cordova at its core. It allows for many command line tools that allows easier mobile development. Ionic also implements Angular and includes its own controllers and modules that allow for greater functionality on Cordova web applications.

When we use Cordova we also gain the benefit of the collection of icons Ionicons, and the testing and development suite Ionic Labs. A more thorough description on Ionicons and Ionic Labs are found on the Design Guidelines and Cordova sections respectively.

ngCordova

![ngCordova's logo](image)

ngCordova is a collection of plugins that allow easier native mobile functionality on Cordova applications, such as the use of a device’s camera, GPS tracker, file system, and more. ngCordova can be used with basic PhoneGap and Cordova, and can also be used with Ionic. For a more detailed explanation of ngCordova and how we are planning to use it, refer to the Cordova section.
Angular

AngularJS is a well know, powerful web framework developed by Google and widely used. Angular extends the HTML vocabulary in applications. It is a toolset for building the framework most suited to your application development. It is fully extensible and works well with other libraries. Every feature can be modified or replaced to suit your unique development workflow and feature needs.

Since Ionic uses Angular, we do not have many options regarding the web framework that we would use. However, since we had already decided for using Angular before we knew of the existence of Ionic, so using Angular with Ionic is not a problem.

The development team is currently in talks of moving from Angular to Angular 2, since it boasts increased performance and greater compatibility with mobile. However, it is still unclear how this decision would affect the use of Ionic as a development framework.

Asana

Asana is a website that allows for project organization in different manners. It is a completely online solution that allows us to manage tasks, deadlines, and developers in a streamlined and comprehensible way. Asana has proven to be useful and up to expectations during the first semester, as the team has used it extensively to track progress and stay on top of their deadlines for the first semester of Senior Design using Asana’s tools.

Asana has also a Slack integration that the teams has already made use of.
Figure 76: Example section on Asana that details the tasks assigned to one person on a specific project

Figure 77: Detail of a specific task, on an Asana project showing comments and subtasks
Toggl

Figure 78: Toggl’s logo

Toggl is a tool that the time will use to track the amount of time each member spends on a project. This tool will be particularly useful for the Scrum meetings that the team will be having on a regular basis, since they will have to track and report on their progress.

Figure 79: Screenshot to show Toggl’s functionality

Toggl does have a paid for ‘Pro’ plan that includes features as tracking income based on time spent, and other similar features that would work best for freelancers and contractors. However, the team does not have a need for such features, and therefore will not subscribe to the ‘Pro’ plan.

Visual Studio

Figure 80: Visual Studio’s logo

Visual Studio provides an array of development tools which covers all major phases of application development. The Visual Studio 2015 IDE will be an integral tool during the development of the Study Group API and web application. Moreover, Visual Studio also
provides Git integration and integrated unit testing alongside powerful documentation features and code templates.

The licensing for Visual Studio will be provided by Azure BizSpark Microsoft Developer Network.

**NinjaMock**

![NinjaMock logo](image1)

*NinjaMock's logo*

NinjaMock is a web tool used for mockups for your application. NinjaMock is the tool that the team used for the mockups used in this documentation document. The designs made on NinjaMock are very simple and are used for designing a wireframe, more than creating a visual design.

NinjaMock allows for a ‘Personal’ account that gives the user three projects maximum. The paid accounts are used for commercial enterprises, therefore the team will have no use for it.

**Swagger**

![Swagger logo](image2)

*Swagger's logo*

Swagger is a simple web tool that will allow us to design the project’s RESTful API. When using the web tool, the team will be able to easily design and document the RESTful API endpoints. Some screenshots of Swagger are shown in the documentation.

Swagger is open source, therefore the team can use it freely.
Slack

Figure 83: Slack's logo

Slack is our main communication platform. In the last months, Slack has acquired a lot of attention as a quick communications platform that is tailored for teams. It is particularly useful for development teams, as we have experienced. Slack also allows for plugins (sometimes called “integrations”), which we have used so far to link a Slack channel to our Github repositories, so that we receive a message through Slack when a commit is made, or a task is assigned. We also linked Asana to Slack so that it updates the users when important changes occur on our Asana projects, such as comments, or completion of tasks.

Under our current plan, we have a Slack free of charge but with a limit of 10000 messages in our message history. Therefore, we cannot access any message that is older than 10000 messages. This would be an issue with bigger teams of 50 or more active members, since they would reach the limit of 10000 messages after each mas sent an average of 200 messages. Since we are only four active members, we don’t need to worry about reaching the limit anytime soon. And when we do, the discarded messages will be too old to be relevant.

Vertabelo

Figure 84: Vertabelo's logo

Vertabelo is a database creation, sketching, and visualization solution that the team used in the beginning of the semester to come up with the current schema for the project’s database. Vertabelo not only allows for easy database creation and modification, but also for collaboration: their web client allows for many users to see real time the changes to the database schema and allows them to take control of the schema to make changes one at a time.
On figure 65, we can take a quick glance at the current database schema for the project. Vertabelo allowed the team to come up with the schema in only a few days, and once the design is done, to export the schema to a SQL script that would create the database designed.

The tool, however, is not completely free. The current plan is intended for academic purposes. It allows for the team to have up to 100 tables per database, and up to 20 databases in total.

Jasmine

Jasmine is an behavior driven JavaScript framework geared towards the testing of front end applications. Jasmine allows for the defining of test cases for particular DOM elements or
JavaScript event functionality coupled with an expectant defined behavior. If the test case receives the expected behavior, the module will successfully pass the test. Jasmine will be used mainly during the testing of the web application and for the front end AngularJS components of Cordova.

**Angular Material**

Angular Material is the first of two options of front end UI frameworks that the team has available and is currently considering.

Angular Material is the easiest way to implement Material Design on Angular applications. For developers using AngularJS, Angular Material is both a UI Component framework and a reference implementation of Google's Material Design Specification. This project provides a set of reusable, well-tested, and accessible UI components based on Material Design.

While the teams still needs to analyze both visual frameworks, Angular Material promises a solid implementation of Material on an Angular framework, therefore it has better prospects than Bootstrap.

**Bootstrap**

![Bootstrap's logo](image)

Bootstrap is the second option out of two front end UI frameworks that the team is currently considering.

Bootstrap is widely used and accepted, therefore it does have more popular credibility than Angular Material. Bootstrap is easy to use and allows the use of preprocessors such as LESS and SASS. Bootstrap’s ideology follows the idea that design should be “mobile first” and therefore follow the guidelines set by mobile devices, such as elongated vertical screens that do not allow for much content to be displayed at once.

Bootstrap also allows for visual functionality such as drop down menus, scrolling animations, carousels and more. The learning curve for Bootstrap would also be less steep as some of the team members are already familiar with Bootstrap.
GulpJS

Gulp is a toolkit that will help the team automate painful or time-consuming tasks in our development workflow. Some examples would be helping us in compiling all the source files for our website, or minifying them. Gulp is used for both Node back end and common front end.

We are free to use Gulp in any way, since it is public domain licensed under CC0.

Bower

Bower is a package manager for the web. It is able to download, track, organize, and update packages from all over the web just as a package manager should. Bower packages are mostly for the front of the web, including UI libraries like Bootstrap or collections like ngCordova.

Bower is easy to use and requires minimum installation effort: It can be installed globally using NPM and it is usable immediately after that.
HighchartsJS

HighchartsJS is a javascript charting library. Charts are interactive and responsive based on viewing media. Moreover, charts load and reload data dynamically by means of leveraging ajax. The framework is HTML5 compliant and will integrate well with Angular 1.x and Ionic.

Entity Framework

Entity Framework is ASP.Net’s data relation mapper. Allows for the streamlining of the data layer by leveraging Code First or Database First approaches. This framework eliminates the need to create data access layers manually, as most of the initial setup is taken cared for.
SignalR

SignalR is a framework for ASP.NET that allows for clients to connect to the server and receive real time instant updates. This is similar to that of Socket.IO functionality. Moreover, SignalR allows for high frequency type change to occur without the need to worry about lower level message piping on the server and client.

SignalR will prove helpful during the development of the dashboard components of the application. This will ensure that changes made by other clients are reflected instantaneously for a particular user. It will also avoid the issue of viewing past data, for instance, if a particular session has been canceled, but the client application has not refreshed the view, the user may find errors when attempting to confirm attendance to the already canceled session.

SignalR is free of charge under the Apache 2.0 License.

TypeScript

Typescript is a preprocessor for JavaScript. Meaning that TypeScript runs before JavaScript to turn all TypeScript format files (.ts) into JavaScript files (.js) with a simple console command. To translate a TypeScript file called greeter.ts to JavaScript, use the following command:

```
tsc greeter.ts
```

The output will be a JavaScript file named greeter.js.
The team members that wish to use TypeScript will do so to generate JavaScript files that will work with our application. TypeScript was created, and is managed by Microsoft.

Postman

Postman is a tool that allows us to test our Web API endpoints and record the results. It is available as a Chrome plugin and a download. Postman has a Basic plan for personal use, and a Cloud plan for enterprise use. Since the team will be using Postman as a personal tool, we will have no need for the paid Cloud plan.
Figure 93: Screenshot of Postman's Google Chrome plugin
Bibliography


Guide/Introduction/Introduction.html


https://developer.apple.com/testflight/

http://ionicframework.com/docs/guide/publishing.html

https://azure.microsoft.com/en-us/documentation/articles/web-sites-create-web-jobs/

https://azure.microsoft.com/en-us/documentation/articles/resource-group-overview/

https://www.google.com/design/spec/material-design/introduction.html#introduction-goals


