SpaceMate

Group #14

Austin Nicholas
Ivey Padgett
Jonathan Warner

April 28th, 2016
Table of Contents

Executive Summary ................................................................................................................. 1
Personnel and Related Work ................................................................................................. 2
Motivation .............................................................................................................................. 4
Broader Impacts .................................................................................................................... 5
Server Setup .......................................................................................................................... 6
Backend Specifications .......................................................................................................... 8
  Language Choice ................................................................................................................ 8
  UML and Class Design ..................................................................................................... 9
  Entity Relationship Diagram ......................................................................................... 12
  API .................................................................................................................................. 16
  Security ........................................................................................................................... 28
Frontend Web Specifications ............................................................................................... 30
iOS Specifications .................................................................................................................. 33
Android Specifications ......................................................................................................... 38
User Flow .............................................................................................................................. 42
  User Looking for Roommate(s) and/or Housing ............................................................. 42
  Housing Professional ..................................................................................................... 55
UI UX .................................................................................................................................. 62
  Material Design ............................................................................................................... 62
  Mobile Application .......................................................................................................... 62
  Web Application ............................................................................................................. 66
Building, Prototyping, and Evaluation Plan ......................................................................... 69
Explicit Design Summary ................................................................................................. 71
  Simplified Use Case Diagram ....................................................................................... 71
  Sequence Diagram ......................................................................................................... 72
Machine Learning .............................................................................................................. 73
  Algorithm ........................................................................................................................ 73
  Tools and Integration ...................................................................................................... 76
  Machine Learning Challenges ....................................................................................... 76
Administrative Content ....................................................................................................... 77
  Budget and Financing ................................................................................................... 77
  Coding Style Guide ....................................................................................................... 78
  Git Flow ........................................................................................................................ 79
  Privacy Policy ................................................................................................................ 82
  Terms and Conditions ................................................................................................. 86
  Milestone Chart ............................................................................................................. 92
References .......................................................................................................................... 93
Executive Summary

Finding a roommate can be a difficult process. Most people have no idea where to start, what to ask, and what to demand. SpaceMate is a new approach to simplifying the problem of searching for a roommate while also reducing the likelihood that you and your new bunkmate will end on bad terms. We plan to make this happen by creating a product that utilizes the user's personality and habits to pair them with potential roommates.

SpaceMate will have a user experience much like that of Tinder. Users will be prompted to complete a guided survey when they first register. The results from this survey will be used to filter out potential matches that have traits that the user absolutely does not want to be paired with. The user will also be able to add traits to their profile that they feel are associated to with themselves. SpaceMate will use these traits to rank matches by relevance using machine learning. When in Discover mode, the user will be presented with potential roommates one at a time. The user will then swipe right, to indicate a potential match, or left, to indicate that they do not wish to be roommates with that person. Users will be given the option to chat if they both swipe right on each other. This option allows the users to further investigate if they want to be roommates. They will then be able to send a roommate request to the other, if they feel like a compatible match has been made.

The overall goal of SpaceMate is to enable users to easily find roommates that they are going to enjoy living with, or at least not fight with. We have set a couple of sub-goals to make sure we meet our overall goal. They are as follows:

- Create Web, Android, and iOS applications that are both easy to use and aesthetically pleasing
- Integrate with Facebook and Google to simplify the process of creating an account
- Utilize machine learning to optimize roommate suggestion and increase the likelihood that users will find compatible matches
- Implement a chat system between matched users to provide a safe place for potential roommates to introduce themselves
- Allow users to specify whether they are open to any housing situation, looking for a match in university residence halls, or have their own place they are looking to share.
- Allow housing professionals to create and manage their brand network of communities allowing users to easily search for roommates.
Personnel and Related Work

Austin Nicholas

Austin has been working with web technologies for two years through course projects and internships. RateMyHousing, his project for Processes of Object Oriented Software, was Austin’s first encounter with web technologies. The project consisted of a web interface and android application that allows college students to search for housing options in the University of Central Florida area. 21st Century Safety, an internship held for a three month span, introduced Austin to many new techniques and best practices through the many web applications that he administered. Austin was given the opportunity to design and develop a streaming video application for 21st Century Safety from the ground up. Though most of Austin’s web experience is in php he is excited to use Javascript in NodeJS and AngularJS.

Ivey Padgett

Ivey has worked on a few simple Android applications, including one for a class where she was one of three people to work on it. She has done quite a bit of Web Development, including AngularJS applications and has 3 years experience with it. Ivey has worked at three different companies doing Web Development for both front-end and back-end. She has also given introductory workshops on Android and Web Development.
Jonathan Warner

Jonathan has worked on a number of Android applications, including a home camera monitor, an application for remote starting his car's air conditioning, and a QR code reader for Google Glass (Android based). While Jonathan's industry experience is typically heavy in web development (a developer at the University of Central Florida doing Angular.js for two years, as well as an intern on the Instagram Web team doing React.js for three months), he has recently been focusing on studying Android Development, and will be interning on the Google Inbox team during Summer 2016 doing work on their Android application. This will be Jonathan's first professional Android development experience, and he hopes to use the experience he gains working on a large application at the company that makes Android to further the design and build quality of the SpaceMate Android application.
Motivation

Austin Nicholas

Team member Austin Nicholas has worked for the Department of Housing and Residence Life (DHRL) at the University of Central Florida (UCF) for two years. In that time he has witnessed countless roommate conflicts that were not mediable. Most, if not all, of those roommate conflicts were the result of the residents choosing to let the University place them with random roommates. Austin hope that SpaceMate will reduce the number of roommate conflicts allowing students and housing staff in colleges to focus their attention elsewhere.

Ivey Padgett

Ivey Padgett spent months before her first semester at UCF trying to find other UCF students who wanted to room in the same dorm as her. She finally found one at last minute that she ended up becoming friends with, but the other two were a nightmare which caused her to break her housing contract and find off campus housing after the first semester. She also struggled to find roommates to fill apartments a few times over the summers as there were requirements some people could or could not fit. SpaceMate would have helped in both of these situations as being able to chat with someone beforehand and find people that fit and/or are okay with certain requirements would have been a breeze.

Jonathan Warner

Jonathan tried using the UCF recommended roommate matching service before the start of his freshman year, and after the software repeatedly failed to match by housing unit, or by interest, and experienced multiple outages, Jonathan put trust in UCF's randomized matching and eventually paid the price by having to move out after a single semester of dealing with a complete personality and noise expectation incompatibility. Building out a system like this would prevent Jonathan's situation from ever happening to another student, which allows students to focus on the things they care about, like academic success.
Broader Impacts

As creators we hope to see all of our products achieve some sort of broader impact. SpaceMate is no exception for any of us. We believe that this project will have significant impacts for housing professionals, residents, and newcomers starting life in a new city.

As a Housing Professional

Housing professionals spend their workdays ensuring that residents will have a comfortable well maintained space to live in. Working for a housing complex means that you will certainly deal with roommate conflicts. Sometimes these conflicts are simple to resolve but often a roommate conflict will take up a lot of time and resources. We hope to build a product that will impact the daily lives of a housing professional by significantly reducing the amount of time that they spend mediating roommate conflicts. The time saved can then be put to better use within the housing community.

As a Resident

Residents have a lot to deal with in their life, Roommate conflicts can add a great deal of stress to residents and can make them feel uncomfortable in their home. This can cause poor daily performance and a general feeling of sadness. Having an easy, intelligent, and widely used roommate matching service would reduce the number of incompatible roommate matches and will allow residents to focus on school, work, living, and being themselves, rather than fighting and coping with their roommates.

As a Newcomer

Moving to a new city is a stressful and anxious process. Newcomers have no firm understanding of what neighborhoods exist, nor do they have any contacts who can recommend housing situations or help the acclimation process. To make matters worse, the newcomer could land in an incompatible rooming situation, which results in a negative experience in their new chapter of life. SpaceMate can help ensure that not only will the rooming situation be compatible and the housing location be in the right neighborhood, but also that the newcomer has a new friend waiting for them when they arrive.
Server Setup

Google Cloud Platform

Google Cloud Platform (GCP) is a cloud computing platform by Google that offers hosting on the same supporting infrastructure that Google uses internally for end-user products like Google Search and YouTube. Cloud Platform provides developer products to build a range of programs from simple websites to complex applications[20]. Of those products we are using Compute Engine, SQL, and Storage.

Compute Engine

Google Compute Engine (GCE) is the Infrastructure as a Service (IaaS) component of Google Cloud Platform which is built on the global infrastructure that runs Google’s search engine, Gmail, YouTube and other services[21]. We are using Google Compute Engine to run one Ubuntu 16.04 LTS Xenial virtual machine. Originally we planned on using Ubuntu 15.10 Wiley, however, with Canonical newest release we feel that it would be better for us to use this LTS version, as it is guaranteed long-term support. The virtual machine running the server application will have the following specifications:

- Ubuntu 16.04 LTS Xenial
- 1 Intel Haswell CPU with 0.6 GB memory
- 10 GB of persistent storage

SQL

Google Cloud SQL is a fully managed MySQL database that lives in the Google Cloud infrastructure[20]. Data storage and reliability is a major priority. Our machine learning algorithms will have to start from scratch if we lose our data due to a corrupt database. Google Cloud SQL gives us security in knowing that our database is online and backed up in case something goes wrong. Google Cloud SQL allows us to focus on building our application while Google takes care of the mundane but necessary tasks like applying patches and updates, managing backups, and configuring replications[36].

Storage

Google Cloud Storage is a RESTful online file storage web service for storing and accessing data on Google's infrastructure[23]. Users of our platform will be asked to upload profile pictures and photo albums to help people determine if they would make a good roommate. This could result in a need for a lot more than 10 GB of storage on our web server. To
fix this we will be using Google Cloud Storage and creating a Durable Reduced Availability bucket for our images. This bucket has no upper limit on size, just a very low price per GB of data.

Security

One advantage to using Google Cloud Platform is that our server is secured by Google well maintained security protocols. Since we are opening network connections to our server to allow users to connect via the internet, we must ensure that we are protected from network based attackers. To do so we will be using Fail2Ban.

Fail2Ban

Fail2Ban is an open source project that scans log files and bans IPs that show the malicious signs[27]. These signs include too many password failures or attempts at known exploits. This software will help prevent automated attacks against the server from bots and prevent people from bruteforcing account passwords.

Server Software

The Apache HTTP Server Project is a collaborative software development effort aimed at creating a robust, commercial-grade, featureful, and freely-available source code implementation of an HTTP (Web) server[35]. Our Linux virtual machine will be running Apache2 to implement our HTTP server. Apache will be configured to run two virtual hosts. One for the main domain of our site (spacemate.xyz) and one for the subdomain that will house our application (app.spacemate.xyz). Apache will forward API requests to the Node.js, while also serving static files off the disk, like the HTML, JavaScript, and CSS files for the web application.

The backend will be built in JavaScript using the Node.js runtime for reasons described in Backend Specifications. Node.js will be installed on the server for executing this code, with the runtime installed through the operating system package manager.

The machine learning system will be built in Python for reasons described in the Machine Learning section. Python 3.4 will be installed through the system package manager with pip used for installing the required Python packages and virtualenv used to maintain our own project's dependencies.
Backend Specifications

Language Choice

The backend will be written in JavaScript using the Node.js runtime. We chose this because we wanted to provide a thin and fast layer on top of the database to handle authentication, authorization and some minor business logic. Since the user facing applications (web, iOS, Android) will all handle the rendering and display of the data, our backend only needs to handle requests to insert and fetch data from the database, and ensure the user has the rights to perform the action they chose.

Node.js fits the bill because it is high performance (see figure below) and allows us to use JavaScript, which eliminates the need to learn another language, a con of candidates such as PHP, Ruby, or Go. Our expertise in building business logic for the JavaScript frontend can be reused in building the backend layer.

![Bubble Sort Times Comparison](image)

*A comparison of bubble sort times with Node.js, Go, and PHP. While Go performs the benchmarks slightly faster than Node.js, we believe the advantage of using a language we are already familiar with far outweighs the performance benefits of Go. It is also shown that Node.js is significantly faster than the traditional web language of PHP. This graph was created by our team when evaluating different language choices.*
UML and Class Design

The backend is to be built in JavaScript, which results in us having no traditional structure for classes [44]. To mitigate this, we are using ECMAScript 6 class support to create reusable classes that can be instantiated based on the results from the database. The UML for the backend is shown below:
The Models section contains all the classes for datatypes in the backend, along with methods to manipulate the data. These models will pull data out of the database tables (see ER diagram) and inflate them with their nested members; for example, Chat objects will have an array of participant User objects that are inflated upon the object instantiation. The models are interfaced with through the HTTP API:

The Helpers will provide methods to deal with session management, creating and verifying password hashes, and links to external systems, such as the database, the bcrypt Node.js module, and Google and Apple push notification systems:
We believe the following tables warrant additional explanation in regards to their rows and function in the application:

User Table

- **id**: The ID of the user
- **name**: String, The legal name of the user
- **facebook_id**: String, Facebook user ID of the user
- **google_id**: String, ID of the Google+ user
- **email**: String, user's email address
- **is_admin**: Bool, whether or not the user is staff
- **last_login**: Date, the last time the user used the app
- **gender**: String, the user's gender identification
- **hashed_password**: String, bcrypt hashed password
- **img_id**: Int, ID of user image from image table
- **apns_token**: String, token used to send notifications to Apple
- **gms_token**: String, token used to send notifications to Google
- **biography**: String, 500 char or less about me of user
- **google_token**: Token for Google Plus used to verify sign in
- **facebook_token**: Token for Facebook used to verify sign in
- **actively_looking**: Bool, whether or not user needs roommates
- **birthday**: Int, birthday in Unix time
- **affiliation**: String, the company or community a user is affiliated with
- **housing_professional**: Bool, whether the account is a professional
- **approved**: Bool, whether or not the professional account is approved
- **phone_number**: String, user's phone number
User Attributes and Traits

The `user_attributes` table contains rows correlating a user with their preferences on a specific attribute. Attributes are dynamic and based on information in the `attribute` table, but the contents of this table can only be modified by an administrator. The `name` and `type` of an attribute describe the thing (i.e. "Preferred temperature") and measurement (i.e. "integer"). The user-facing application will show survey responses based on these attributes, and the results of these flows will be stored in `user_attributes`.

User traits are freeform. The trait table is fed into the typeahead search, and the user input is stored in the `trait_string` field and is tied to the user who input it.
Matching Information

Information about previous user matches are stored for use in the recommendation system. Previous matches and whether or not they were accepted are stored, along with values listing the current cached coefficients for the user. More information about the coefficients table can be found under Machine Learning.
The backend API uses JavaScript Object Notation (JSON) for message passing over HTTPS. This allows us to use the same API for the web, Android, and iOS platforms, and running the service over HTTPS will reduce the chance of tokens or credentials being intercepted in Man-In-The-Middle attacks (see security below).

The API is a RESTful implementation [43] where every resource is defined by its path and the intended operation is specified using the HTTP method: a POST request indicates that something should be created, whereas a GET request means something should be pulled down.

The API is specified as follows:

**Account Management**

Create an account

```json
POST /api/user/signup
{
    first_name: String,
    last_name: String,
    email: String,
    phone: String,
    birthday: Date,
    facebook_id: [Optional] Integer,
    facebook_access_token": [Optional] String,
    google_id: [Optional] Integer,
    google_access_token: [Optional] String,
    password: String // If Facebook or Google IDs are not present
}
```

Result:

```json
{ success: Boolean, error: String }
```
Sign in

```plaintext
POST /api/user/signin
{
    email: String,
    facebook_id: [Optional] Integer,
    facebook_access_token: [Optional] String,
    google_id: [Optional] Integer,
    google_access_token: [Optional] String,
    password: String // If Facebook or Google IDs are not present
}
```

Result:
```json
{ success: Boolean, error: String }
```

Forgot password

```plaintext
POST /api/user/forgot
{
    email: String
}
```

Result:
```json
{ success: Boolean, error: String }
```

Reset password

```plaintext
POST /api/user/reset
{
    email: String,
    reset_token: String
}
```

Result:
```json
{ success: Boolean, error: String }
```
Manage Notification Tokens

**POST /api/user/notifications**

```json
{
    "gms_token": [Optional] String, // Google Messaging Service token
    // for push notifications on
    // Android
    "apns_token": [Optional] String // Apple Push Notification Service
    // token for iOS devices
}
```

Result:

```json
{ success: Boolean, error: String }
```

Get survey questions

**GET /api/user/survey**

Result:

```json
{
    success: Boolean,
    error: String,
    attributes: [Attribute]
}
```

Send survey answer

**POST /api/user/survey**

```json
{
    "attribute_id": Integer,
    "preference": String,
    "own_attribute": String,
    "importance": Integer
}
```

Result:

```json
{ success: Boolean, error: String }
```
Update user profile

```json
POST /api/user/profile
{
    first_name: String,
    last_name: String,
    email: String,
    phone: String,
    birthday: Date,
    password: String, // If created with email
    phone_number: String,
    affiliation: String
}
```

Result:

```json
{ success: Boolean, error: String }
```

Matching System

Get set of potential users

```json
GET /api/matching/bundle
```

Result:

```json
{
    success: Boolean,
    error: String,
    users: [User]
}
```

Accept or Reject a user

```json
POST /api/matching/match/:user_id
{ accept: Boolean }
```

Result:

```json
{ success: Boolean, error: String }
```
List accepted and rejected users

GET /api/matching/matches

Result:

```
{
    success: Boolean,
    error: String,
    Users: [User]
}
```

Star a match

POST /api/matching/match/:user_id/star

Result:

```
{ success: Boolean, error: String }
```

Unstar a match

DELETE /api/matching/match/:user_id/star

Result:

```
{ success: Boolean, error: String }
```

Messaging System

Send a message

POST /api/chat/:chat_id

``` { message: String // The message to send to the chat } ```

Result:

```
{ success: Boolean, error: String }
```
Send an image

```plaintext
POST /api/chat/:chat_id/image
{
  image: Base64Image // The image to send to the chat
}
```

Result:

```json
{ success: Boolean, error: String }
```

Create a new chat

```plaintext
POST /api/chat
{
  participants: [User IDs] // The users to include in the new chat
}
```

Result:

```json
{ success: Boolean, error: String }
```

Star a chat

```plaintext
POST /api/chat/:chat_id/star
```

Result:

```json
{ success: Boolean, error: String }
```

Unstar a chat

```plaintext
DELETE /api/chat/:chat_id/star
```

Result:

```json
{ success: Boolean, error: String }
```
Get latest chat messages

GET /api/chat/chat_id?last_message_id=message_id

```
{  
    last_message_id: [Optional] message_id  
    // The latest message ID in the chat. If omitted, all messages  
    // will be fetched. Otherwise all messages after message_id  
    // will be fetched, if any  
}
```

Result:

```
{  
    "success": Boolean,  
    "error": String,  
    "messages": [Message]  
}
```

Get user chats

GET /api/chat

Result:

```
{  
    "success": Boolean,  
    "error": String,  
    "chats": [Chat]  
}
```

Add chat participants

POST /api/chat/:chat_id/participants

```
{  
    participants: [User IDs]  
    // The users to add  
}
```

Result:

```
{  
    success: Boolean,  
    error: String  
}
```
Remove chat participants

```
DELETE /api/chat/:chat_id/participants
{ participants: [User IDs] // The users to remove }
```

Result:
{ success: Boolean, error: String }

House listing management

Get user's own house listings

```
GET /api/house
```

Result:
{ "success": Boolean,
  "error": String,
  "houses": [House] // The house listings owned by the user
}

Create a new house listing

```
POST /api/house
{ house: House // The house listing to create }
```

Result:
{ success: Boolean, error: String }

Delete a new house listing owned by the user

```
DELETE /api/house/:house_id
```

Result:
{ success: Boolean, error: String }
Update a house listing

```plaintext
PUT /api/house/:house_id
{ house: House // The updated house listing }
```

Result:

```json
{ success: Boolean, error: String }
```

Mark a house as full

```plaintext
POST /api/house/:house_id/full
```

Result:

```json
{ success: Boolean, error: String }
```

Mark a house as no longer full

```plaintext
DELETE /api/house/:house_id/full
```

Result:

```json
{ success: Boolean, error: String }
```

Star a house

```plaintext
POST /api/house/:house_id/star
```

Result:

```json
{ success: Boolean, error: String }
```

Unstar a house

```plaintext
DELETE /api/house/:house_id/star
```

Result:

```json
{ success: Boolean, error: String }
```
Community Safety

Report a user

```javascript
POST /api/community/report
{
    user: User_ID // The users to report,
    reason: String // An explanation for staff to view
}
```

Result:
```
{ success: Boolean, error: String }
```

Block a user

```javascript
POST /api/community/block/user_id
```

Result:
```
{ success: Boolean, error: String }
```

Unblock a user

```javascript
DELETE /api/community/block/user_id
```

Result:
```
{ success: Boolean, error: String }
```

Delete user's own account and all associated data

```javascript
DELETE /api/user
```

Result:
```
{ success: Boolean, error: String }
```
Community Management

Create a community

```plaintext
POST /api/community
{
    name: String,
    location: String,
    amenities: [String],
}
```

Result:

```
{ success: Boolean, error: String }
```

Update a community

```plaintext
PUT /api/community/:community_id
{
    name: String,
    location: String,
    amenities: [String],
}
```

Result:

```
{ success: Boolean, error: String }
```

Get user's communities

```plaintext
GET /api/community
```

Result:

```
{ success: Boolean, error: String, communities: [Community] }
```
Add house to community

```plaintext
POST /api/community/house
{
  house_id: Int,
}
```

Result:

```json
{ success: Boolean, error: String }
```

Add image to community

```plaintext
POST /api/community/:community_id/image
{
  image: Base64Image,
}
```

Result:

```json
{ success: Boolean, error: String }
```

Block user from community

```plaintext
POST /api/community/:community_id/block
{
  user_id: Int,
}
```

Result:

```json
{ success: Boolean, error: String }
```
Unblock user from community

```json
DELETE /api/community/:community_id/block
{
  user_id: Int,
}
```

Result:

```
{ success: Boolean, error: String }
```

Security

To protect the users' data and accounts, we have decided on the following precautions:

Session Tokens

All requests made to the server will include a session token. This token is unique to the user agent making the request, be it on Android, iOS, or web. No two agents will have the same session token, even if they are on the same platform; for example, if Alice has two Android phones with SpaceMate signed in to the same account on each, each phone would be issued a different session token. This allows the system to expire old tokens and tokens no longer in use, while keeping service alive for legitimate users of the application. These session tokens will then be used to verify that the device requesting information from our API is acting on behalf of the correct user.

Password Hashing

All passwords will be hashed using bcrypt at 14 rounds. This decision was arrived at because previously accepted methods, like md5 and sha1, with or without salts, are increasingly easy to brute force, whereas bcrypt is computationally expensive and every user will receive their own salt by design [5].

HTTPS

The use of HTTPS will encrypt all messages to and from the server in transport, which will greatly reduce the risk of Man-In-The-Middle attacks, where the user's password, session token, or personal data are intercepted in transport. Our HTTPS implementation is made possible by certificates issued by Let's Encrypt.
Let’s Encrypt

HTTPS encryption can be expensive for multi domain web sites but it is certainly a necessity to protect the user’s information. This is why we have chosen to use Let’s Encrypt, a new certificate authority that is free, automated, and open source[37].

For information about the security of information at rest, such as the application code and user database, see the Security subsection of Server Setup.
Frontend Web Specifications

The Frontend Website application will be built using Angular 2.0 and Angular Material Design Components. The application will be written in TypeScript, HTML5, and Sass. It will be compiled and minified with Gulp into JavaScript and CSS3.

Setup

Framework

We will be using Angular 2.0 as the main framework for the website because it will simplify things with the ability to bind data between the view and model [50]. Angular Material Design components mixed with Angular will help make the site responsive and attractive along with keeping the style of the website in line with the Material Design Guidelines we have chosen to follow [51]. All members of the group have some experience with Angular as well, which makes the onboarding process for anyone working on the frontend easier.

Architecture

We will be using a Model-View-Controller (MVC) architecture with our frontend with a service as the model, Angular controllers as the controller, and HTML and templates as the view [52]. This is a simple architecture that works very well with how Angular is set up and allows for easy modification and displaying of data.

Model

The website will be written as a single page web application with a static splash page. The simple page web application model allows us to serve HTML only once and decrease the page load times. Having a single page web application means we don’t need to set up a server for just the website to run on [53]. Instead, we can serve the website locally with Gulp and compile everything statically to be put on the main server. AngularJS works really well as a single page application and the team has experience doing so. Single page applications have a downside though, it makes search engine optimization and page crawling work incorrectly as they will only see the Angular templating tags before the JavaScript has loaded [54]. To get around that, the team has decided that the only information a web crawler will need to see is information on the homepage of the site, as the rest of the site will be behind a login screen. This is why the homepage will be a static splash page that contains any information we would want a web crawler to see in addition to the information for new visitors to see.
Routing
Routing for the website will be handled with AngularJS as well. We will setup the Angular Application to serve users certain templates from the website depending on which URL they are currently at [55].

HTTP Requests
We will use Angular’s built in $http library to make any HTTP requests to and from the API. These will be handled in services that will be added to the Angular module for the website. The HTTP requests will be sent to the Backend API as mentioned previously in the document [56].

Spam Prevention
We will use Captchas on the user sign up to ensure that only humans can sign up and create accounts. Since the rest of the website is hidden behind a login, no bots will be able to spam users without getting through the Captchas first [57].

External Dependencies
The following dependencies will be needed on the frontend web application:

Facebook SDK
The Facebook SDK will be used to allow users to log in to the application through Facebook and pull information from their profile (name, gender, etc).

Google Play Services
The Google Play Services SDK will be used to allow users to log in to the application through Google+ and to pull data from their profile (name, email, etc).

Gulp
Gulp will be used to compile and minify the frontend into static content to upload to the server. We will be using Gulp as it is a very popular and well-maintained framework many web developers in the community use [58]. Most of the team has experience both using and setting up Gulp which made the decision to use it easy.
Sass

Our CSS will be compiled from Sass. Sass is a preprocessor for CSS that extends the capabilities of plain CSS. This is done by providing easy, compact, and intuitive syntax to implements features of CSS that otherwise would require large amounts of code as well as adding features that are not provided in CSS. Some of the added features are variables and importing [59].

NPM

NPM will be the package manager we will use for any other external dependencies that may come up over the course of development. Sass and Gulp will both be managed by NPM as well. NPM is the most popular package manager web developers are using right now and it is very well documented and maintained [60].

Testing Frontend Web

The frontend will be tested manually during implementation and nearing the end of the project. We will just ensure everything works as intended and that the system works as a whole.
iOS Specifications

The iOS application will be built in Xcode 7.3 using Apple’s Swift language. This is because the Xcode IDE is at the center of the Apple development experience[7] and Swift is a powerful and intuitive programming language for iOS, OS X, tvOS, and watchOS[8].

Our iOS application will be built targeting iOS devices running iOS 9. At the current time Apple is reporting that eighty four percent of iOS devices are running iOS 9[9]. This percentage is always growing as consumers upgrade their devices forcing them to the latest version of iOS.

When writing an iOS application programmers are given the choice to write their application using Swift or Objective-C. Our group has chosen to use Swift because we recognize Swift as the upcoming industry standard for iOS development. With this in mind we only see it fit that our application should be written in Swift from the very first version.

External Dependencies

External dependencies for our iOS application will be manage using the Cocoapods software package. Cocoapods is a dependency manager for Swift and Objective-C Cocoa projects with over ten thousand libraries[10]. Cocoapods is easily configures and allows for package names to be added to a file, named Podfile. Those packages are then downloaded and installed with a single command.
Facebook SDK

The Facebook SDK for iOS will enable users to sign up for and into our app using their Facebook account. When users log into our app with Facebook they can grant permissions to our app so we can retrieve information or perform actions on Facebook on their behalf[16].

Pros

- Users won’t have to remember a new password if they login with Facebook
- Users will be more familiar with Facebook and feel more comfortable using their login system
- Grants the potential for easily sharing SpaceMate via their Facebook profile
- Grants access to profile data to create their profile easily

Cons

- User information may be outdated or falsified
- User deleting their Facebook account means they also deleted their SpaceMate account

Google Sign-in for iOS

Google Sign-in for iOS will enable users to sign up for and into our app using their Google account. When users log into our app with Google they can grant permission to our app so we can retrieve information or perform actions on Google on their behalf[17].

Pros

- Users won’t have to remember a new password if they login with Google
- Users will be more familiar with Google and feel more comfortable using their login system
- Grants access to profile data to create their profile easily
- Gmail users will always provide a valid email that they currently use

Cons

- User information may be outdated or falsified
- User deleting their Google account means they also deleted their SpaceMate account
IQKeyboardManagerSwift

iOS developers tend to run into a lot of issues with the iOS keyboard. By default the keyboard will slide up and cover the text field that it is modifying. IQKeyboardManagerSwift[11], written by Mohd Iftekhar Qurashi, is a package written in Objective-C and Swift that fixes this issue by sliding up the screen enough to keep the text field being modified in view. This packages also enables the next and previous buttons above the keyboard that can be easily be linked to the next and previous text fields.

Pros

● Works automatically with zero lines of code
● Earily tests resulted in desired outcome

Cons

● Little documentation when it comes to customizing the UI/UX

Alamofire

In order for our iOS SpaceMate application to work it needs to be able to communicate with our API. This could be done using NSURL, NSURLRequest, NSURLSession, or NSURLConnection, however, there is an easier way. Alamofire is an HTTP networking library written in Swift that allows chainable requests[12].

Pros

● Reduces the amount of code needed to implement RESTful api calls
● Suggested by many developers around the internet
● Well documented

Cons

● Alamofire builds are currently failing on select devices
AlamofireImage

Image downloading will play a large role in our application. Each user will have a profile picture and albums of images (i.e. photos of their living space). To handle image downloads and caching we are going to use AlamofireImage[13]. AlamofireImage is an extension of Alamofire that enables Image Response Serialization, Async Remote Downloads with Placeholders, Prioritized Queue Order Image Downloading, Auto-Purging In-Memory Image Cache, UIImage Extensions for Inflation / Scaling / Rounding / CoreImage, and more.

Pros

- Takes the complication out of Image serialization and download
- Works with and extends image caching

Cons

- Although its builds are passing, Alamofire builds are currently failing on select devices.

Freddy

We will need to parse our JSON once we retrieve it from our API. Freddy is a reusable framework for parsing JSON in Swift. It is advertised to have three principle benefits: Freddy provides a type safe solution to parsing JSON in Swift, Freddy provides an idiomatic solution to JSON parsing that takes advantage of Swift's generics, enumerations, and functional features, Freddy provides great error information for mistakes that commonly occur while parsing JSON[14].

Pros

- Adds JSON functionality to Swift
- Uses understandable syntax
- Implemented using do/catch to handle cases where the object doesn’t exist

Cons

- None

Regex

Most of our data validation and sanitation will happen in the API, however, we do want to check that a couple things are in the right format on the application side. One of those items is
the user's email address. Before making our HTTP request we want to verify that the user has entered an email that is at least in a valid format. To do this we must make sure it matches the general email regular expression: `^[A-Z0-9a-z._%+-]+@[A-Za-z0-9.]+\.[A-Za-z]{2,6}$`. The Regex[15] package allows us to easily implement regular expressions in Swift.

**Pros**

- Allows easy implementation of regular expressions in Swift

**Cons**

- Little documentation

The best part about using Cocoapods to manage all of the external dependencies above is that all we need in order to manage them is the following Podfile:

```plaintext
use_frameworks!

target 'SpaceMate' do
  pod 'IQKeyboardManagerSwift'
pod 'Alamofire', '~> 3.3'
pod 'AlamofireImage', '~> 2.0'
pod 'Freddy'
pod 'Regex'
end
```
Android Specifications

The Android application will be built using Android Studio 2.0, which is the latest version of the software and Google's officially supported tool for building Android applications [1].

Our Android application will be built targeting the Android API level 17, which is supported by Android devices using version Jellybean 4.2 and above. At present this API level is supported by 87.3% of Android devices worldwide [2] (see figure). We expect this number to grow over the next semester. As a team, we agreed that the added APIs supported by level 17 (namely Nested Fragments and improved security features regarding exporting services [3]) were a worthwhile compromise for the support of approximately 87% of devices used worldwide.

![Android Platform Version Distribution](image.png)
Development Platform

We will be using Android Studio 2.0 for the development of the Android application. We chose this because it is the latest stable release of the software and is officially supported by both Google and the Android development community. Android Studio is built on top of IntelliJ [39] and thus is a powerful IDE for the development of Java applications. The 2.0 release also supports Instant Run, which allows us to quickly iterate on development in the Android app similar to how we would on the web. This is important because Android builds of large applications in the previous version could take over a minute.

Setup

Architecture

Android typically applications use a Model-View-ViewModel architecture [40], and our application will be no different than the industry standard. This architecture is very similar to the Model-View-Controller architecture: we will have models for the various types of data in the application, like users and matches, along with views constructed in XML that will be inflated with the appropriate data. The ViewModel construct is very similar to a controller, but instead of handling business logic, the ViewModel acts as a binding layer between the data in the model and their correlating data manipulation methods, and the user interface. The ViewModel layer will typically involve code for adding event listeners, inflating views and fragments in the views, and managing navigation to other views.

Framework

We feel the Android SDK is sufficient for developing our application and no additional frameworks will be needed.

Model

The Android application will be written using the Fragments style design: instead of building out individual activities for each user interface, we will have one main activity for the logged-in experience, with a navigation panel that folds out on the side, with the center content related to whatever page the user is currently on. This is similar to the single-page application design model that is used on the web application (see Frontend Specifications). In addition to this, the application will have separate activities for when the user is signed out, including sign in and sign up activities.
We chose the fragments route because it allows us to build reusable components that can be used in various views of the application, and keeps all the navigation logic on the screen for the user at all times. The fragments route is also the community recommended option of best practice [41].

External Dependencies

We have identified the following external dependencies for the Android application:

Facebook SDK

The Facebook SDK will allow us to provide Facebook sign up and login in the application, as well as pull data like name, email, and photo from a user's Facebook profile.

Google Play Services

We will be using Google Play Services to implement push notifications to the end user's device. We will also be using the Google Play Services SDK to add Google+ sign in integration and pull data from a user's Google+ profile.

GSON

We will be using the GSON library by Google [26] for serializing and deserializing JSON data to and from the server API and the Java classes in the application. This is important because Java, the language Android apps are written in, is strongly typed, which prevents us from generating object literals like we would in JavaScript on the web application. While we do have the option of writing out calls to manually construct and deconstruct objects, this is unmaintainable and requires numerous changes any time the API specification is tweaked. Serialization using GSON will allow us to transmit native Java objects to the server and automatically construct Java objects out of the reply.

In addition to serialization, GSON supports creating JSON objects out of stream, which allows us to directly convert the bytes read from the HTTP response into an object, rather than having to build a string out of the response and read the string into a JSON parser.

Picasso

Picasso is an image loading library for Android that will allow us to efficiently load images onto the view [28]. This is important because every user card and profile the user sees will involve a user photo. Without Picasso, loading images would require downloading bytes over HTTP, passing the stream into a BitmapFactory, scaling the image correctly, and loading it
into the appropriate view, while also having to deal with error handling at every step. Picasso simplifies this by accepting HTTP image URLs and an ImageView to render to and handles the grunt-work for us, which reduces the amount of complexity and failure points we must worry about. Using this library makes loading images as simple as it is on the web and on iOS with AlamoFireImage.

OkHttp

OkHttp is an HTTP library for Android [29] that simplifies the complexity of sending data to and receiving data from the server, as well as helps deal with networking issues on the user's device, which is crucial to handle gracefully on a mobile device that can be switching between cellular towers or WiFi networks. This library will support our goals of using a RESTful API and can be combined with GSON to send and receive JSON to the server.

Testing Android application

The Android application brings a set of unique challenges when developing and testing. Because Android supports a variety of screen sizes, resolutions, densities, and aspect ratios, we need to be able to test how the application renders on multiple devices that have significant market share. Since it would be cost prohibitive to purchase every physical device we wish to target, the tool Genymotion will be used to create virtual devices with these screen sizes and pixel densities, along with their respective software versions.

Statistically, the top Android devices in use in the United States are the Samsung Galaxy S5, S6, S4, Core Prime, Note 4, and S3 [4] and as such, we will be testing against these virtual devices.

In addition to testing against different device variations, testing Android applications has the challenge of being prohibitively time expensive, as the build, deploy, and loading on the phone take far longer than would be seen with web or command-line applications. Since our application will be heavily dependent on a web API, we will be using the built-in JUnit integration in Android Studio to rapidly develop and test the business logic of the application without having to constantly redeploy the mobile application. This will also prevent regressions when pushing new code.

As mentioned in Building and Prototyping Plan, we will be using BuddyBuild™ for continuous builds of our Android application. This tool will automate deploying builds to developers and testers, as well as handle sharing of Google Play Services tokens that are required for push notifications and Google+ authentication.
User Flow

There are two different types of user flows. The first being a user looking for a roommate and/or housing and the second being a housing professional looking to list their property. The user flow between the three applications will be the same. There will not be any missing functionality on the platforms when it comes to being a user looking for a roommate or housing. For housing professionals, they will only be able to access their side of the application on the website as it is heavily reliant on filling in information and uploading images.

User Looking for Roommate(s) and/or Housing

Website Splash Page

This is the main page that appears when the user goes to the root domain and is not logged in to an account. It contains information about the website that is important for visitors and any web crawlers. It also allows the user to either sign up or sign in.

Features

- Sign up and Sign in
  - Through email
  - Through Facebook
  - Through Google+
- Housing Professional sign up and sign in
- Description of application and its uses
- Screenshots of the application in use
- FAQs about the application
- Contact information

How to get to this page

- Go to the root domain when not signed in
- Sign out

Where to go from this page

- Create an account page
- Main page for user who is logged in
Mobile Application Home Screen

This is the main page when a user pulls up the application and is not signed in through their phone. It allows the user to sign in or create an account.

Features

● Sign in
● Create an Account
● Sign in with Facebook
● Sign in with Google+

How to get to this page

● Pull up the application when not signed in
● Sign out

Where to go from this page

● Logged in main page

Create an Account Screen

This is the screen a user will see in order to create an account on both mobile and web. The application will ask for basic information such as name, email, and password and allow users to create an account with our service. The application will also ask if the user is affiliated with anything, such as a school or a company. This will allow the system to better match

Features

● A form asking for basic user information in order to create an account
  ○ First and Last Name
  ○ Email
  ○ Phone Number
  ○ Birthday
  ○ Location
  ○ Affiliation
    ■ UCF student
    ■ Company X Intern
  ○ Password
  ○ Password Confirmation
● Buttons to cancel or register
How to get to this page

● Choose to create a new account from either the web splash page or the mobile home screen

Where to go from this page

● Back to the web splash page if on web
● Back to the mobile home screen if on mobile
● The guided survey on both

Guided Survey

This is the screen users will be directed to once they create an account. It will ask the user for their opinions on common roommate issues and use this information to match them with others. The survey will be a list of questions such as “what temperature do you prefer the room to be at” along with how much they care about that issue. It will also ask them where they fall for that issue so they can be matched with people.

Features

● Survey of common roommate issues
  ○ What should the temperature be set to
  ○ How late should people be up
  ○ Stance on overnight visitors
  ○ Noise levels
● Ranking on how much the user cares about the particular issue
  ○ Doesn’t care at all
  ○ Cares a little
  ○ Cares
  ○ Cares a lot
● Preference for the particular issue
  ○ Only if they care at least a little about the issue
  ○ Example would be “no overnight guests”
● User’s position on the particular issue
  ○ Asks the user where they fall themselves regardless of where they want their roommate to fall

How to get to this page

● Create an account
• Choose to modify your answers from the preferences screen

Where to go from this page
• The main signed in screen
• Back to preferences

Sign in Screen
This is the screen that will display when a user chooses to sign in either on the website or the mobile application. It will ask the user for an email and password then sign the user in.

Features
• A form asking for the user’s email and password
• A cancel and submit button
• Forgot password button

How to get to this page
• Choosing to sign in from the web splash screen
• Choosing to sign in from the mobile application home screen

Where to go from this page
• Back to the web splash screen if on web
• Back to the mobile application home screen if on mobile
• The signed in main screen on both
• Forgot password screen

Forgot Password Screen
This is the screen users will be routed to when they have indicated they forgot their password. The screen prompts the user to enter their email address so the system can email them a forgot password link.

Features
• A form asking for email
• Submit button
• Cancel button
How to get to this page

- Selecting “forgot password” from the sign in page

Where to go from this page

- The main screen on mobile
- The splash page on web

Signed in Main Screen

This is the screen a user will see once they are logged into the application on both mobile and web. It will show the user one potential match at a time with a picture, some information about the person, an option to go to the person’s profile, and the accept or reject buttons. The user can also click on a menu in the top left of this screen to bring out a menu drawer to visit other pages of the application.

Features

- View users that are potential matches
  - User’s name and picture
  - Some information about them
  - A link to visit their profile
  - Accept and reject buttons
- Side Menu to reach other pages of the application
  - Turn off searching
  - User Profile
  - Discover Mode
  - Pictures
  - Housing Listings
  - Preferences
  - Messages
  - All matches
  - Starred matches
  - Deleted matches
  - Logout

How to get to this page

- Signing in
- Visiting the website root when already signed in
● Loading up the mobile application when already signed in
● Clicking “Discover Mode” from the sidebar on any other page

Where to go from this page
● User profile
● Pictures
● My Housing
● Preferences
● Messages
● All matches
● Starred matches
● Deleted matches
● Another user’s profile
● Chat with a user
● Mobile application home screen if on mobile and signed out
● Web splash page if on web and signed out

User Profile Page
This is the screen a user will see on both the website and the mobile application when visiting another user’s profile. The screen will give information about the user, the ability to accept or reject, the option to chat if you have accepted one another, the ability to favorite the current user, and the ability to block.

Features
● The name of the user
● The user’s picture
● A biography about the user
● Personal information the user has chosen to disclose such as
  ○ Hometown
  ○ Sexual Orientation
  ○ Religion
  ○ Pets
● Tags that describe the user
  ○ Party animal
  ○ Music lover
  ○ Night Owl
● Accept and reject buttons
● Chat button if you have accepted one another
- Favorite user button
- Block button

How to get to this page
- Choosing to view a user’s profile from the card on the discover screen
- Choosing to view a user’s profile from the card on one of your matches pages
- Clicking on the user in chat and choosing to visit their profile

Where to go from this page
- Back to the signed in main page
- Back to your matches
- To chat with the user
- Any page on the sidebar

Current User’s Profile
This is screen a user will see when they choose to visit their own profile. It will display to them what others can see when visiting their profile and give the user the option to edit their profile.

Features
- View your information that is public to other users
  - Name
  - Picture
  - Biography
  - Tags
  - Personal Information
  - Whether or not the current user has a housing arrangement
- Edit your profile

How to get to this page
- Choose to visit your profile from the side menu

Where to go from this page
- Back to the signed in main page
- Edit profile page
- Any page on the menu sidebar
Edit Profile Page

This is the page the user will see when they choose to edit their profile. They will have the option of updating their profile picture, biography, tags, personal information, and password.

Features

- Change profile picture
- Update biography information
- Update tags
- Update personal information
- Change password
- Update housing arrangement
- Update location
- Update affiliations
  - UCF for example

How to get to this page

- Choose to edit your profile from your profile page

Where to go from this page

- Back to your profile

Messages

This screen shows the user the list of messages they have. It lists out the people in the chat and the last message received. The user can click on a chat to open it up, long click on a chat on mobile to choose to leave the chat, and hover over the chat on the website to choose to leave the chat. The user will also be able to search for chats and filter by favorited chats. The chats are sorted by most recent first.

Features

- List chats the users is involved it
  - Names of users in the chat
  - Last message in the chat
- Long press on mobile to choose to leave chat
- Hover over chat and select leave chat on the website
- Search chats
- Filter by favorited chats
How to get to this page

- The sidebar from any page

Where to go from this page

- A selected chat
- Any page on the sidebar

Chat Screen

This is the screen a user will see when involved in a chat. A chat screen can only be reached when trying to chat with users that are mutual matches, meaning all people have accepted one another. Chats can occur between multiple users, meaning a group of people thinking of rooming together can chat all at once. This view shows the user any messages they have received and sent along with letting the user send a new message. The user can also send pictures, visit another user’s profile from this screen, see the chat history, favorite a chat, and leave a chat.

Features

- Send and receive chat messages
- Send and receive pictures
- Send a message
- Visit a user’s profile
- See chat history
- Favorite a chat
- Leave the chat
  - This then rejects any users in the chat

How to get to this page

- Selecting to chat on a user’s profile in which both users have accepted each other
- Selecting to chat from any of the matches screens
- Clicking on a notification from a chat message
- Selecting the chat from the messages page

Where to go from this page

- Back to the messages page
- Back to the matches
- Back to a user’s profile
Pictures

This is the page users will go to when they want to add or view their photos. Photos they should upload would be related to them or their housing. Users can upload images of themselves or add images of their house to the house’s album.

Features

- View list of images the user has uploaded
- View the album for the housing a user has if it exists
- Add images
- Delete images
- Add images to house album
- Delete images from house album

How to get to this page

- The sidebar

Where to go from this page

- Any page from the sidebar

Preferences

This is the page users will go to when they want to modify their preferences. Preferences include things such as notification settings, visibility, and the guided survey.

Features

- Change notification settings
- Change a user’s visibility
  - Can be matched
  - Cannot be matched
- Update the guided survey responses

How to get to this page

- The sidebar

Where to go from this page

- The guided survey
- Any page from the sidebar

**All Matches**

This page shows the user all mutual matches they have. This page will be set up with user cards in a list that the user can scroll through. The user has all options available to them from the main signed in page when it comes to user cards (visit profile, reject, etc.). The user will also have the ability to chat with any of these users as they are mutual matches.

**Features**

- List of mutual matches for the current user
- Card for each user they are mutually matched with
  - Picture
  - Name
  - Visit Profile
  - Accept/Reject
  - Chat

**How to get to this page**

- The sidebar
- A notification of a mutual match

**Where to go from this page**

- A user profile
- Chat

**Starred Matches**

This is a page that lists out any starred mutual matches a user has. Users can star mutual matches so that they can easily find any mutual matches they are very interested in chatting or trying to make the match permanent. The setup of the page is exactly the same as the ‘all matches’ page but only displayed starred matches.

**Features**

- List of mutual matches that the user has starred
- Card for each match
How to get to this page
  ● The sidebar

Where to go from this page
  ● A user profile
  ● A chat

Deleted Matches

  This page lists out any previous mutual matches that the user has since rejected. This allows users to give prior matches a second chance if needed, and chat with them even though it didn’t work out the first time. The layout is the same as the ‘all matches’ page but only displays matches that were once mutual and no longer are.

Features
  ● List of previously mutual matches
  ● Cards for each match

How to get to this page
  ● The sidebar

Where to go from this page
  ● A user’s profile
  ● Chat

Housing Listings

  This page allows the user to indicate they have housing and add information for it. If the user has housing, they indicate this by toggling they already have housing and the page will be displayed in a way that they can add information about the housing. This information includes pictures, total rooms, available rooms, location, type, and more. The user can then share this page with any matches they have so the match is aware of the housing situation. A user can choose to list their housing on the available housing list for other users to see.

  If a user does not have housing, the page will list out available housing in their area that was put in by the housing professionals who use the application. The list will contain cards of housing information that include the name of the housing, the type, pictures, a chat button to talk
to the professional managing it, and a link to the house’s page with more information. The user can filter by starred housing by clicking the star button at the top

Features

- Toggle housing situation
  - I have housing
  - I do not have housing (default)
- Add information about the housing if toggled
  - Type of housing (apartment, house, etc)
  - Location
  - Pictures
  - Total rooms
  - Available rooms
  - Price
  - Amenities
- Option to list housing on the housing list
- Share housing page with another user
- See a list of housing from housing professionals otherwise
  - Name
  - Location
  - Type
  - Pictures
  - Chat with the housing professional
  - More information
  - Star housing
- Filter by favorited housing

How to get to this page

- The sidebar

Where to go from this page

- The page for the housing
- Chat with a professional
- Chat with another user
  - Shares the housing page with this user
Housing Page

This is the page users will go to in order to see more information about housing that is listed on the application. The page will include pictures, the name of the housing, the type, available units, costs, amenities, and more. This information is all added by housing professionals when listing their units or by users when adding their housing. Users can browse pictures and message the lister from this page along with star the housing to find it later. Users will also be able to report a housing listing if they think it is inappropriate, fake, or spam.

Features

● Pictures of housing
● Type of housing
● Name of the complex if there is one
● Available rooms or units
● Costs
● Amenities
● Rules
  ○ No pets
  ○ No overnight guests
● Contact lister
● Favorite housing
● Report Listing

How to get to this page

● Click on more information about housing from the housing list

Where to go from this page

● Chat

Housing Professional

Sign up

This is the screen housing professionals will use to create an account with the system. The system will ask the user for their name, email, company, work address, phone number, password, and password confirmation.
When a company is creating an account for the first time, they will need to be approved manually by our team and will not be permitted to access the rest of the application until this has happened. Any subsequent sign ups through the same domain will be auto-approved.

Features

- A form asking for required information in order to create a housing professional account
  - First and last name
  - Email
  - Company
  - Work address
  - Phone Number
  - Password
  - Password Confirmation

- Submit and cancel buttons

How to get to this page

- Choose to create a housing professional account from the web splash screen

Where to go from this page

- The company verification page if the company has not been approved
- The main housing professional signed in screen if the company has been approved

Company Verification Page

This is the page users will be routed to when they have signed up as a housing professional with a company that has not been approved yet. There will be instructions on how to get approved which include emailing the team some sort of documentation that proves they are a real company and they are who they say they are.

Features

- Instructions on how to get approved through the application
  - Email or mail the team some proof of business documentation to prove the company is real
    - EIN, articles of incorporation, etc.
  - Contact information for the team

How to get to this page

- Creating an account with a company that is not
Where to go from this page

- Back the web splash page

Main Housing Professional Signed in Screen

This is the screen that a user will see when they are signed in as a housing professional. If the company the user is a member of has any communities added this page will be a list of the communities. Each community will have the community name and the options to edit or delete the community. Clicking on a community will take the user to the community page which lists the housing options for that community. The bottom corner of this page will have an add button that allows the user to add a community. There will be a side navigation bar that lets the user go to the preferences, the communities, the messages, and the network members pages.

Features

- See a list of communities for a company if they exist
  - Edit community
  - Delete community
- Add a new community

How to get to this page

- Sign in to an approved company
- Access the page from the side menu

Where to go from this page

- Edit community page
- Any page on the side menu

Community Page

This is the page the user will go to when they click on a community from the list of communities. It will display the information that is currently input for the community including name, location, amenities, website, and pictures at the top of the page. There is an edit button for the user to choose to edit the community and a delete button.

The rest of the page are the housing cards for housing options that exist in this community. An example would be a floor plan for an apartment complex. The cards will display information about the housing option such as name, picture, price, availability, number of rooms, and more. There is an edit and a delete button for each card. Clicking on these cards will take the
user to the page for the housing option. At the bottom corner of the page there is a button to add a new housing option.

Features

- Information about community
- Edit community
- List of housing cards
  - Name
  - Picture
  - Price
  - Availability
  - Number of rooms
- Ability to delete or edit a housing option
- Add new housing option

How to get to this page

- Clicking on a community from the community list

Where to go from this page

- Edit community
- Housing option page
- Edit housing option
- Back to community list

Edit Community Page

This is the page a user will go to when they are either creating a new community for their company or editing an already existing one. A community would be like Lake Claire within UCF. The community will take a name, location, amenities, website, and pictures.

Features

- Edit the information about a community
  - Name
  - Location
  - Amenities
  - Website
  - Pictures
- Save or cancel
How to get to this page

- Choose to edit a community

Where to go from this page

- Back to the list of communities
- Back to the community page

Housing Page

This is the page users will see when they click on a housing option from a community page. This page will display the information about the housing information which includes name, pictures, price, availability, number of rooms, and more. There will be an option to edit or delete the housing from this page.

Features

- View information about the housing option
  - Name
  - Pictures
  - Price
  - Availability
  - Number of Rooms
- Edit the housing
- Delete the housing

How to get to this page

- Click on a housing option from a community page

Where to go from this page

- Edit housing
- Back to community page

Edit Housing Page

This is the page that a user will go to when they choose to edit the information about a housing option. The page will ask for the information about a housing option including name, pictures, price, availability, number of rooms, and more.
Features

- Fields for information on the housing option
  - Name
  - Pictures
  - Price
  - Availability
  - Number of rooms
- Cancel and save buttons

How to get to this page

- Choosing to edit a housing option from the community page
- Choosing to edit a housing option from the housing page

Where to go from this page

- Back to the housing page

Preferences Page

This page will hold any preferences for a housing professional such as changing their password and notification settings.

Features

- Change password
- Edit notification settings
- Save button

How to get to this page

- Click on the preferences from the sidebar

Where to go from this page

- Anywhere on the sidebar

Messages Page

This page will hold any messages that a company has received so that employees can respond to them. Users will be able to search messages and delete a conversation. Clicking on a conversation will open it.
Features

- List of messages
  - Name of people involved
  - Last message
- Ability to delete conversations
- Search messages

How to get to this page

- Clicking on messages from the sidebar

Where to go from this page

- Individual messages

Network Members

This page will list out any users who are in the network. This includes users who signed up on the roommate matching side as affiliated with the network and housing professionals signed up through the network. Users will be able to suspend, block, and reactivate users in the network.

Features

- List of users in the network
  - Users signed up to look for roommates and stated they are affiliated with the network
  - Users signed up as housing professionals for the network
- Suspend users in the network
- Block users from the network
- Reactivate any previously suspended or blocked users

How to get to this page

- Choose the network members page from the sidebar menu

Where to go from this page

- Any link on the sidebar
UI UX

Material Design

All three of our platforms (Web, iOS, and Android) will use Google’s Material Design guidelines for its User Interface (UI) and User Experience (UX). The next few pages will show you the current mockups for SpaceMate and briefly describe the workflow of the particular view.

NOTE: Although the following mockups for our mobile application are displayed using an android template, these mockups will be used for the iOS native application as well.

Mobile Application

Launch View

- First view a user will see when launching SpaceMate application for the first time
- User will see this screen every time they launch the app unless they select remember user
- Presents options for login with user information, Facebook integration, or Google+
- Presents option to register for new users
Login View

- Users will only see this view if their login credentials have not been stored and they click the login button on the Launch View.
- Email field will checked to ensure a basic email format has been entered on the device.
- Password will be passed as is via SSL to the API.
- User credentials will be verified by the API.

Register View

- Users are directed to the Register View by clicking the Register button on the Launch View.
- All field except password will be checked for basic format on the device.
- API will further sanitize all fields.
- Password will be checked for requirement and be sanitized by the API.
- API will respond as Successful/Unsuccessful along with further details via JSON.
Discover View

- First view a logged in user will see when starting the application
- Users can switch to Discover View at any time via the menu drawer
- Cards containing profile information will be presented to the user in the middle of the screen
- User can click ‘VIEW PROFILE’ to see more details about the user
- The whole card will have the user profile picture in the background with the lower half shadow box over top.
- User can swipe left or click the ‘X’ button to reject a user as a potential roommate
- User can swipe right or click the (Star) button to accept a user as a potential roommate

Profile View

- Profile view has two forms: ‘my profile’ and ‘their profile’
- Edit button will only appear in the top right corner if user is on the ‘my profile’ form
- (Star) button will only appear in the middle right side if the user is on the ‘their profile’ form
- About and Tags are among the fields that will display on this view
- User picture will be in the light blue area at the top of this view
Messages View

- Users can switch to Messages view at any time via the menu drawer
- Users is presented with a list of messages they are currently have
- Messages are sorted by time and data with the most recent being on top
- Users can search for a specific chat or filter by their favorite chats with the buttons in the top right corner
- Users can open a chat by clicking on its entry

Individual Message View

- Users will enter Individual Message view by either selecting a message in Message view or clicking message on a Match Notification (Below)
- Users can view chat history
- Users can send images or text messages
- Users can favorite a chat message to easily find it later
Menu View

- Menu View can be accessed from most every view via the menu draw icon in the top left corner.
- Provides navigation to different views.

Match Notification

- Notification popup when a match between two users is found.
- Notification will display on both user’s devices.
- Users can temporarily ignore the match.
- Users can start a chat with the match.

Web Application

Our mockups have not been created for the web application yet, however, we have a vision in mind for how we want it to look. The following two screenshots contain aspects of what we want the aesthetics to be when using the full web version of SpaceMate.
Google Inbox

We like the way the Google Inbox uses the styles and guidelines set by Material Design with slight modifications to create their web application. We plan to utilize the taskbar and side menu. The middle panel will be where our different views load as they are navigated to through the side menu. The side menu of the web application will have the following Navigation points.

**User looking for roommates/housing**
- Profile
- Discover
- Messages
- Preferences
- All Matches
- Stared Matches
- Deleted Matches
- Logout

**Housing Professionals**
- Communities
- Messages
- Network Members
- Preferences
Facebook Messenger

Facebook messenger doesn’t use Material Design, however, we really like the way they have implemented a panel interface for their chat service. We plan to create a similar view using Material Design components with the removal of the right panel. Once the right panel is removed we will extend the chat panel to the right wall. When a user needs to access the side menu it will slide over the left chat messages panel.
Building, Prototyping, and Evaluation Plan

We plan to rapidly prototype our application on all three platforms. We already have detailed plans about our application and backend infrastructure, as well as user interface designs. However, we want to ensure we make revisions as early as possible when features do not perform as expected in front of end users. To achieve the ability to rapidly prototype the application, we will be using a combination of continuous integration, continuous deployment, and real life beta testers. Our beta testers will consist primarily of friends, family, and other Senior Design students who agree to trade feedback with their own projects.

To assist with rapid prototyping, we have identified a number of tools that will automate various development tasks:

Travis CI

Travis CI is a continuous integration platform that integrates directly with GitHub. This tool will allow us to automatically run unit tests on all backend code we attempt to merge into the main codebase, which will help us detect regressions or cases where the new code does not fit the style guide. This tool is mostly free for educational use, see Administrative Content for details.

BuddyBuild

BuddyBuild is a tool that does continuous integration, continuous delivery, and a feedback management system for iOS and Android applications. BuddyBuild will watch out SpaceMate git repositories and, with every push to the master branch, it will build a deliverable version of our app. A download link to that deliverable versions can then be sent out to specified users in order for them to test out the latest beta version of our application. BuddyBuild even automatically adds functionality to these beta version builds that allow users to easily submit feedback by taking a screenshot of any bugs and adding a message to that screenshot.

Evaluation Plan

To ensure our solution adequately solves the problems with finding housing identified in Executive Summary and Broader Impacts, we plan to use the user feedback from the BuddyBuild system, as well as in-person interviews with testers, to ensure that the user experience achieves our goals and that no software bugs or glitches prevent the system from achieving its goals. It is also critical that the application intuitive out of the box so that any user can hop on board and start finding housing; we consider it a failure if users need to read a manual or get training in
order to use the system. At the end of the day, all the feedback received from the testers will be inconcluded in our revision plans and integrated in as rapidly as possible.
Explicit Design Summary

Simplified Use Case Diagram
Machine Learning

The application will use machine learning to improve the order of relevance that potential matches are shown in. This requires an algorithm that ranks the strength of any potential match. This is known as a recommendation problem, and can be seen in examples like Netflix recommending movies similar to those the user has watched, or Amazon trying to display products that are most relevant to the user's search query.

Our goal is to sort users based on a feedback loop from their selected matches. To keep track of characteristics of users, every user has the ability to add freeform traits to describe themselves. Example traits include activities like "studying," "partying," "drinking," and "singing," as well as hobbies like "video games," "guitar," and "Breaking Bad." To keep similar traits consistent, typeahead autocomplete will be used to suggest traits to be input, and disambiguation logic will be handled by the backend (see below.)

When a user likes a user with a certain trait, all users with that trait should be ranked higher, and when a user dislikes a user with a certain trait, all users with that trait should be ranked lower. An example of this is as follows:

1. Joe likes Nickelback
2. You swipe left on Joe, indicating you are not interested in rooming with Joe
3. People who like Nickelback will be sorted lower
4. This trend can be reversed by swiping right on people who like Nickelback

Algorithm

The algorithm we have derived comes in two main parts: trait scoring and user trait feedback similarity.

Trait Scoring

Trait scoring gives potential matches a score based on an equation built out of the total coefficients for all of the user's liking and skipping history. Every user that is liked will contribute a +1 for every trait they have, and every user that is skipped over will received a -1 for every trait they have.

These traits are summed up to create coefficients, which are then used to score new possible matches based on what traits they have.

The pseudocode version of the algorithm is included below for reference:
rated_users = [{ user objects containing:
    name: String,
    attributes: String[],
    rating: Int,
}]

function createUserCoefficients() {
    var coeffs = {};
    rated_users.forEach(function(user) {
        user.attributes.forEach(function(attr) {
            if (!coeffs[attr]) coeffs[attr] = 0;
            coeffs[attr] += user.rating;
        });
    });
    return coeffs;
}

function scoreUsers(users, coeffs) {
    users.forEach(function(user) {
        user.score = 0;
        user.attributes.forEach(function(attr) {
            user.score += coeffs[attr]
        });
    })
    return users
}

var coeffs = createUserCoefficients();
Echo "Based on people you like/dislike, this is what we think is important to you:"
Echo coeffs

users = scoreUsers(rated_users, coeffs);
Echo "This is how the users stack up:"
Echo users

sorted = users.sort((a, b) ->
    return b.score - a.score;
);

Echo "The users will appear in the following order:"
Echo sorted.map((user) -> return user.name)
In an example run of the above algorithm, the user would have an equation of coefficients generated for themselves based on the previous match behaviour. New, unseen traits will not weighted in the ranking, but if potential matches with the trait "smoking" are consistently skipped, these matches will drop to the bottom of the stack. No matches will be removed from the potential match stack, but the order of matches is important for allowing users to find other good matches quickly, or the platform usability will suffer for them.

It is important to note that these rankings are only on a single-user basis, which means that no other users will be able to influence the ranking that the user sees. Logically, the preferences of the population user base should not influence the preferences of an individual user. In addition, this will prevent users from attempting to spam the system and train it to rank in manners they prefer. This is important to consider, as Internet audiences have found entertainment in attempting to troll machine-learning systems, as seen with Microsoft's Tay experiment [45].

For performance reasons, these coefficients will be stored in a table and updated with each transition. We believe it would not be scalable to recalculate the coefficients with every rank request, as this could involve fetching the trait rows for hundreds of previous matches. Instead, adding or subtracting 1 to previous traits, or adding new traits to this table, should be more efficient than computing on the fly.

User trait feedback similarity

The second method of ranking is more pure in its description as a machine-learning algorithm. We believe that users who like and dislike the same kinds of people will be logically good matches, and as such, will attempt to rank users based on how similarly they like and dislike other users. For example, if Sam and Cindy both dislike people who party, but both like people who play piano, Sam and Cindy are a close match.

As mentioned above, this is a recommendation problem. Specifically, we are creating an item-item content recommender, where we score items (other users) with features (traits) based on if they like or dislike that trait. This is accomplished using a K-means clustering algorithm [46] given a matrix where the rows are users and the columns are traits, and the content consists of 0's and 1's to indicate whether or not the particular row typically likes or dislikes that trait. By typically we mean whether that coefficient is positive or negative in the previous algorithm. This K-means algorithm is provided through scikit-learn (see tools below).

These two ranking methods are weighted into a score that is used to sort the potential matches. We intend to run the first trials with a 50/50 weight and refine the weights from there based on user feedback. We will also have to tweak our K-means cluster size depending on user
feedback, and possibly even try other algorithms like Euclidean distance, depending on the feedback users give about the ranking relevance. See *Evaluation* for more details.

Tools and Integration

The machine learning portion of the application will be written in Python. We have decided to do this as much of the machine learning community uses this language for data science, and various libraries have been written for solving machine learning problems in Python. This is an important point, as the backend API of the application will be written in Node.js for reasons described in *Backend Specifications*. To interface with the data in the machine learning component, the Node.js backend binds to the service over a Unix socket. It is believed that this will not cause significant latency to the service, but in the event that it does, we may rebuild the math logic into the Node.js layer, or run the Python code as an offline batch job or asynchronous task.

However, we believe it will be helpful for future maintainability to keep the machine learning logic as a separate component of the application. This will allow us to switch out the Python code for a larger system, like Apache Spark backed by Hadoop, if we find we wish to do more machine learning challenges, or that the Python solution does not scale as we hoped.

We will be using the Scikit-Learn library that provides methods for running recommendation problems and k-means clustering, which limits the number of wheels we must reinvent. Scikit-Learn is also very well documented [47].

Machine Learning Challenges

The biggest challenge we expect to face is effective ranking on cold start. When a user requests their first set of matches, we have no ranking information to feed them, and thus no way of knowing if the set they are presented with is remotely compatible with their preferences. Furthermore, the liking and skipping behavior of the first set of users will influence the next batch the most, since this will be the only data the algorithm has to work off of.

This will be mitigated by the filtering that occurs based off the user preference survey (see *User Flow*), but the extent to how relevant these matches are will depend on user feedback.
Administrative Content

Budget and Financing

SpaceMate is student-sponsored project, and thus has no provided sponsor money. All costs with development of the application, such as server costs and developer licenses, will be paid for by the students on the project.

Google Cloud Platform

Google Cloud Platform (GCP) is a cloud computing platform by Google that offers hosting on the same supporting infrastructure that Google uses internally for end-user products like Google Search and YouTube. Cloud Platform provides developer products to build a range of programs from simple websites to complex applications[20]. Of those products we are using Compute Engine, SQL, and Storage.

Compute Engine

Google Compute Engine (GCE) is the Infrastructure as a Service (IaaS) component of Google Cloud Platform which is built on the global infrastructure that runs Google’s search engine, Gmail, YouTube and other services[21]. We are using Google Compute Engine to run one virtual machine. Google Cloud Platform estimates the cost of that machine to be $5.79 per month.

SQL

Google Cloud SQL is a fully managed MySQL database that lives in the Google Cloud infrastructure[20]. Google Cloud Platform estimates the cost of their smallest tier to be $15.12 per month[22]. This expense may be removed during development by hosting our one MySQL database on our Compute Engine virtual machine.

Storage

Google Cloud Storage is a redundant RESTful online file storage web service for storing and accessing data on Google's infrastructure[23]. This service will be used for user image storage. Google Storage comes in three tiers: Standard, Durable Reduced Availability (DRA), and Nearline. We will be using DRA as it is slightly cheaper than Standard and only had 0.9% less reliability than the Standard tier. Google Cloud Platform estimates the cost of DRA to be $0.02 per GB per month. For development we plan to use less than 1 GB of storage costing up $0.02 per month.
Travis CI Costs

Travis CI is free with the GitHub Student Developer Pack[19], but only up to 100 runs per month before we have to email support and request more. If Travis ends up being cost-prohibitive, it is possible to run the unit tests and linting on our own server.

BuddyBuild Costs

BuddyBuild is currently free but their website says that pricing is coming soon. Currently we have unlimited usability but when they do switch to a pricing model we may be limited in our usability. There is a note on the BuddyBuild site that says “we’ll always offer a free tier.” [18] On one hand it could be the case that their free tier allows us to do everything we need but on the other hand we could be restricted from doing what we need. We will discontinue use of BuddyBuild in our development of the SpaceMate apps if the latter is the case.

Coding Style Guide

Enforcing a coding style guide will ensure that best practice is followed and that all team members can collaborate on the same code without issue. The styleguides will be enforced in a linting style provided by the Travis CI build checks, and these checks must pass before merging code into the master or develop branches.

Android

The Android code (Java) will follow the style guide enforced for contributions to the Android operating system code [6]. While we are clearly not contributing to Android itself, the style rules were created by individuals with significant experience in writing maintainable Java code, and we believe this style guide sets a reasonable standard for our own codebase.

iOS

The iOS code (in Swift) will follow GitHub's style guide [48] for Swift code. These rules will be automatically enforced by SwiftLint [49] to ensure the code meets best practices and all contributed code is consistent between all developers.

Frontend Web

The frontend web application will be written in HTML, CSS, and JavaScript. The JavaScript will be following the rules used in TSLint [30] which will be checked with pre-commit hooks and integrations in text editors. This will be used it is recommended that
Angular 2 applications are written in TypeScript [31] and TSLint is the most well-known linter for TypeScript. The directory structure and Angular components will follow John Papa’s Angular 2 style guide which is currently being added to the official Angular docs [32].

As for the SASS and CSS files, they will follow the Scss-lint style guide [33], which is community backed and the most popular SASS linter. We will connect this with a pre-commit hook as well. SASS will automatically compile CSS files so these will follow the style guide set out by the SASS compiler.

**Backend (Node.js)**

The backend API of the server application will be written in JavaScript and follow the Node.js Style Guide [24] by Felix Geisendörfer. ESLint [25] will be used to enforce the style in a pre-commit hook and in the Travis CI checks. We chose this style because it has nearly 3,000 stars on GitHub, making it a popular community choice.

**Git Flow**

To ensure that all developers can contribute to the same projects with minimal impact to each other, we have decided to adopt the Git Flow model [42]. Our rules for branching, committing, and pushing work as follows:

1. All projects will have *production* and *develop* branches
   a. All development work must branch off the *develop* branch
   b. The production branch holds the code that is deployed to the end users through BuddyBuild and other continuous deployment systems

2. When a developer wants to start a new change, they must switch to the *develop* branch, pull the latest changes, and branch off into their own branch
   a. New branches should be prefixed with *feature/* or *fix/*, depending on what the purpose of the new branch is
   b. Each branch should be used for one feature or one fix only, unless necessary to deviate from this process

3. When preparing to push changes, the developer should pull the latest remote changes from *develop* into their own branch
   a. If any conflicts are found, these should be resolved before pushing

4. New changes must be pushed to the remote as a pull request
   a. Developers should not merge their own changes in
b. Other developers should review their changes and ask for clarification if needed
c. The branch is merged if all changes look good

5. When merging the branch, the squash feature of GitHub should be used
   a. This feature does an interactive Git rebase and turns several commits into one
   b. Each fix or change should only be one commit
      i. This keeps history clean and makes reverting bad changes easier

Instead of merging in two commits with their own messages, a single commit is created with a message that explains the combination of the two changes

Commit Messages

Developers working on the project should follow Tim Pope's advice on writing commit messages [34]. Specifically:

- The first line of each commit message should be a short summary of the change
  - The summary should be written in the imperative: Fix, not Fixed or Fixes, and Merge, not Merged or Merges
  - Be descriptive enough that other developers know the gist of the change without having to read the body or diff
- The second line must be blank
- The following lines should explain why the change was made if the first line is not abundantly clear
Following these conventions will ensure that all developers are able to collaborate on projects together without causing a mess of merge conflicts or being unsure why certain changes were made or if they were intentional.
Privacy Policy

Protecting your private information is our priority. This Statement of Privacy applies to the http://spacemate.xyz and https://app.spacemate.xyz and SpaceMate and governs data collection and usage. For the purposes of this Privacy Policy, unless otherwise noted, all references to SpaceMate include http://spacemate.xyz and https://app.spacemate.xyz and SpaceMate. SpaceMate is a predictive roommate matching service site. By using the SpaceMate website, you consent to the data practices described in this statement.

Collection of your Personal Information

SpaceMate may collect personally identifiable information, such as your name, address, birthday, email address, and telephone number. SpaceMate may also collect anonymous demographic information, which is not unique to you, such as your age, gender, political affiliation, race and religion. We may gather additional personal or non-personal information in the future.

Information about your computer hardware and software may be automatically collected by SpaceMate. This information can include but is not limited to: your IP address, browser type, domain names, access times and referring website addresses. This information is used for the operation of the service, to maintain quality of the service, and to provide general statistics regarding use of the SpaceMate website.

SpaceMate encourages you to review the privacy statements of websites you choose to link to from SpaceMate so that you can understand how those websites collect, use and share your information. SpaceMate is not responsible for the privacy statements or other content on websites outside of the SpaceMate website.

Use of your Personal Information

SpaceMate collects and uses your personal information to operate its website(s) and mobile application(s) and deliver the services you have requested.

SpaceMate may also use your personally identifiable information to inform you of other products or services available from SpaceMate and its affiliates. SpaceMate may also contact you via surveys to conduct research about your opinion of current services or of potential new services that may be offered.

SpaceMate does not sell, rent or lease its customer lists to third parties.
SpaceMate may share data with trusted partners to help perform statistical analysis, send you email or postal mail, provide customer support, or arrange for deliveries. All such third parties are prohibited from using your personal information except to provide these services to SpaceMate, and they are required to maintain the confidentiality of your information.

SpaceMate may keep track of the websites and pages our users visit within SpaceMate, in order to determine what SpaceMate services are the most popular. This data is used to deliver customized content and advertising within SpaceMate to customers whose behavior indicates that they are interested in a particular subject area.

SpaceMate will disclose your personal information, without notice, only if required to do so by law or in the good faith belief that such action is necessary to: (a) conform to the edicts of the law or comply with legal process served on SpaceMate or the site; (b) protect and defend the rights or property of SpaceMate; and, (c) act under exigent circumstances to protect the personal safety of users of SpaceMate, or the public.

Use of Cookies

The SpaceMate website may use "cookies" to help you personalize your online experience. A cookie is a text file that is placed on your hard disk by a web page server. Cookies cannot be used to run programs or deliver viruses to your computer. Cookies are uniquely assigned to you, and can only be read by a web server in the domain that issued the cookie to you.

One of the primary purposes of cookies is to provide a convenience feature to save you time. The purpose of a cookie is to tell the Web server that you have returned to a specific page. For example, if you personalize SpaceMate pages, or register with SpaceMate site or services, a cookie helps SpaceMate to recall your specific information on subsequent visits. This simplifies the process of recording your personal information, such as billing addresses, shipping addresses, and so on. When you return to the same SpaceMate website, the information you previously provided can be retrieved, so you can easily use the SpaceMate features that you customized.

You have the ability to accept or decline cookies. Most Web browsers automatically accept cookies, but you can usually modify your browser setting to decline cookies if you prefer. If you choose to decline cookies, you may not be able to fully experience the interactive features of the SpaceMate services or websites you visit.
Security of your Personal Information

To secure your personal information from unauthorized access, use or disclosure, SpaceMate uses Fail2Ban.

When personal information (such as a credit card number) is transmitted to other websites, it is protected through the use of encryption, such as the Secure Sockets Layer (SSL) protocol.

Minors Under Thirteen

SpaceMate does not knowingly collect personally identifiable information from Minors under the age of thirteen. Minors under the age of thirteen are restricted from using SpaceMate.

Minors Under Eighteen

SpaceMate does not knowingly collect personally identifiable information from Minors under the age of eighteen and over the age of thirteen without the Minor first gaining permission from their parent or guardian. Minors under the age of eighteen and over the age of thirteen are restricted from using SpaceMate without permission of a parent or guardian.

Disconnecting your SpaceMate Account from Third Party Websites

You will be able to connect your SpaceMate account to third party accounts. BY CONNECTING YOUR SPACEMATE ACCOUNT TO YOUR THIRD PARTY ACCOUNT, YOU ACKNOWLEDGE AND AGREE THAT YOU ARE CONSENTING TO THE CONTINUOUS RELEASE OF INFORMATION ABOUT YOU TO AND FROM OTHERS (IN ACCORDANCE WITH YOUR PRIVACY SETTINGS ON THOSE THIRD PARTY SITES). IF YOU DO NOT WANT INFORMATION ABOUT YOU, INCLUDING PERSONALLY IDENTIFYING INFORMATION, TO BE SHARED IN THIS MANNER, DO NOT USE THE THIS FEATURE. You may disconnect your account from a third party account at any time through Facebook or Google.

Opt-Out & Unsubscribe

We respect your privacy and give you an opportunity to opt-out of receiving announcements of certain information. Users may opt-out of receiving any or all communications from SpaceMate by contacting us here:
- Web page: http://spacemate.xyz/contact
- Email: spacemateapp@gmail.com
Changes to this Statement

SpaceMate will occasionally update this Statement of Privacy to reflect company and customer feedback. SpaceMate encourages you to periodically review this Statement to be informed of how SpaceMate is protecting your information.

Contact Information

SpaceMate welcomes your questions or comments regarding this Statement of Privacy. If you believe that SpaceMate has not adhered to this Statement, please contact SpaceMate at:

Email Address:
spacemateapp@gmail.com

Effective as of May 01, 2016
Terms and Conditions

Agreement between user and spacemate.xyz

Welcome to spacemate.xyz. The spacemate.xyz website and mobile applications herin (the "SpaceMate Application") is comprised of various web pages operated by SpaceMate ("SpaceMate"). The SpaceMate Application is offered to you conditioned on your acceptance without modification of the terms, conditions, and notices contained herein (the "Terms"). Your use of the SpaceMate Application constitutes your agreement to all such Terms. Please read these terms carefully, and keep a copy of them for your reference.

The SpaceMate Application is a Social Networking Site to help users find roommates.

Privacy

Your use of the SpaceMate Application is subject to SpaceMate's Privacy Policy. Please review our Privacy Policy, which also governs the SpaceMate Application and informs users of our data collection practices.

Electronic Communications

Visiting the SpaceMate Application or sending emails to SpaceMate constitutes electronic communications. You consent to receive electronic communications and you agree that all agreements, notices, disclosures and other communications that we provide to you electronically, via email and on the SpaceMate Application, satisfy any legal requirement that such communications be in writing.

Your account

If you use this site, you are responsible for maintaining the confidentiality of your account and password and for restricting access to your computer, and you agree to accept responsibility for all activities that occur under your account or password. You may not assign or otherwise transfer your account to any other person or entity. You acknowledge that SpaceMate is not responsible for third party access to your account that results from theft or misappropriation of your account. SpaceMate and its associates reserve the right to refuse or cancel service, terminate accounts, or remove or edit content in our sole discretion.

SpaceMate does not knowingly collect, either online or offline, personal information from persons under the age of thirteen. If you are under 18 and over 13, you may use the
SpaceMate Application only with permission of a parent or guardian. If you are under the age of 13 you may not use the SpaceMate Application.

Links to third party sites/Third party services

The SpaceMate Application may contain links to other websites ("Linked Sites"). The Linked Sites are not under the control of SpaceMate and SpaceMate is not responsible for the contents of any Linked Site, including without limitation any link contained in a Linked Site, or any changes or updates to a Linked Site. SpaceMate is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement by SpaceMate of the site or any association with its operators.

Certain services made available via the SpaceMate Application are delivered by third party sites and organizations. By using any product, service or functionality originating from the SpaceMate App, you hereby acknowledge and consent that SpaceMate may share such information and data with any third party with whom SpaceMate has a contractual relationship to provide the requested product, service or functionality on behalf of the SpaceMate Application’s users and customers.

No unlawful or prohibited use/Intellectual Property

You are granted a non-exclusive, non-transferable, revocable license to access and use the SpaceMate Application strictly in accordance with these terms of use. As a condition of your use of the SpaceMate Application, you warrant to SpaceMate that you will not use the Site for any purpose that is unlawful or prohibited by these Terms. You may not use the SpaceMate Application in any manner which could damage, disable, overburden, or impair the SpaceMate Application or interfere with any other party's use and enjoyment of the SpaceMate Application. You may not obtain or attempt to obtain any materials or information through any means not intentionally made available or provided for through the Site.

All content included as part of the Service, such as text, graphics, logos, images, as well as the compilation thereof, and any software used on the SpaceMate Application, is the property of SpaceMate or its suppliers and protected by copyright and other laws that protect intellectual property and proprietary rights. You agree to observe and abide by all copyright and other proprietary notices, legends or other restrictions contained in any such content and will not make any changes thereto.

You will not modify, publish, transmit, reverse engineer, participate in the transfer or sale, create derivative works, or in any way exploit any of the content, in whole or in part, found on the SpaceMate Application. SpaceMate content is not for resale. Your use of the SpaceMate Application does not entitle you to make any unauthorized use of any protected content, and in
particular you will not delete or alter any proprietary rights or attribution notices in any content. You will use protected content solely for your personal use, and will make no other use of the content without the express written permission of SpaceMate and the copyright owner. You agree that you do not acquire any ownership rights in any protected content. We do not grant you any licenses, express or implied, to the intellectual property of SpaceMate or our licensors except as expressly authorized by these Terms.

Third Party Accounts

You will be able to connect your SpaceMate account to third party accounts. By connecting your SpaceMate account to your third party account, you acknowledge and agree that you are consenting to the continuous release of information about you to others (in accordance with your privacy settings on those third party sites). If you do not want information about you to be shared in this manner, do not use this feature.

International Users

The Service is controlled, operated and administered by SpaceMate from our offices within the USA. If you access the Service from a location outside the USA, you are responsible for compliance with all local laws. You agree that you will not use the SpaceMate Content accessed through the SpaceMate Application in any country or in any manner prohibited by any applicable laws, restrictions or regulations.

Indemnification

You agree to indemnify, defend and hold harmless SpaceMate, its officers, directors, employees, agents and third parties, for any losses, costs, liabilities and expenses (including reasonable attorney's fees) relating to or arising out of your use of or inability to use the SpaceMate Application or services, any user postings made by you, your violation of any terms of this Agreement or your violation of any rights of a third party, or your violation of any applicable laws, rules or regulations. SpaceMate reserves the right, at its own cost, to assume the exclusive defense and control of any matter otherwise subject to indemnification by you, in which event you will fully cooperate with SpaceMate in asserting any available defenses.

Arbitration

In the event the parties are not able to resolve any dispute between them arising out of or concerning these Terms and Conditions, or any provisions hereof, whether in contract, tort, or otherwise at law or in equity for damages or any other relief, then such dispute shall be resolved only by final and binding arbitration pursuant to the Federal Arbitration Act, conducted by a single neutral arbitrator and administered by the American Arbitration Association, or a similar
arbitration service selected by the parties, in a location mutually agreed upon by the parties. The arbitrator's award shall be final, and judgment may be entered upon it in any court having jurisdiction. In the event that any legal or equitable action, proceeding or arbitration arises out of or concerns these Terms and Conditions, the prevailing party shall be entitled to recover its costs and reasonable attorney's fees. The parties agree to arbitrate all disputes and claims in regards to these Terms and Conditions or any disputes arising as a result of these Terms and Conditions, whether directly or indirectly, including Tort claims that are a result of these Terms and Conditions. The parties agree that the Federal Arbitration Act governs the interpretation and enforcement of this provision. The entire dispute, including the scope and enforceability of this arbitration provision shall be determined by the Arbitrator. This arbitration provision shall survive the termination of these Terms and Conditions.

Class Action Waiver

Any arbitration under these Terms and Conditions will take place on an individual basis; class arbitrations and class/representative/collective actions are not permitted. THE PARTIES AGREE THAT A PARTY MAY BRING CLAIMS AGAINST THE OTHER ONLY IN EACH'S INDIVIDUAL CAPACITY, AND NOT AS A PLAINTIFF OR CLASS MEMBER IN ANY PUTATIVE CLASS, COLLECTIVE AND/ OR REPRESENTATIVE PROCEEDING, SUCH AS IN THE FORM OF A PRIVATE ATTORNEY GENERAL ACTION AGAINST THE OTHER. Further, unless both you and Employer agree otherwise, the arbitrator may not consolidate more than one person's claims, and may not otherwise preside over any form of a representative or class proceeding.

Liability disclaimer

THE INFORMATION, SOFTWARE, PRODUCTS, AND SERVICES INCLUDED IN OR AVAILABLE THROUGH THE SITE MAY INCLUDE INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN. SPACEMATE AND/OR ITS SUPPLIERS MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE SITE AT ANY TIME.

SPACEMATE AND/OR ITS SUPPLIERS MAKE NO REPRESENTATIONS ABOUT THE SUITABILITY, RELIABILITY, AVAILABILITY, TIMELINESS, AND ACCURACY OF THE INFORMATION, SOFTWARE, PRODUCTS, SERVICES AND RELATED GRAPHICS CONTAINED ON THE SITE FOR ANY PURPOSE. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, ALL SUCH INFORMATION, SOFTWARE, PRODUCTS, SERVICES AND RELATED GRAPHICS ARE PROVIDED "AS IS" WITHOUT WARRANTY OR CONDITION OF ANY KIND. SPACEMATE AND/OR ITS SUPPLIERS HEREBY DISCLAIM ALL WARRANTIES AND CONDITIONS WITH REGARD TO THIS
INFORMATION, SOFTWARE, PRODUCTS, SERVICES AND RELATED GRAPHICS, INCLUDING ALL IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL SPACEMATE AND/OR ITS SUPPLIERS BE LIABLE FOR ANY DIRECT, INDIRECT, PUNITIVE, INCIDENTAL, SPECIAL, CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF USE, DATA OR PROFITS, ARISING OUT OF OR IN ANY WAY CONNECTED WITH THE USE OR PERFORMANCE OF THE SPACEMATE APPLICATION, WITH THE DELAY OR INABILITY TO USE THE SPACEMATE APPLICATION OR RELATED SERVICES, THE PROVISION OF OR FAILURE TO PROVIDE SERVICES, OR FOR ANY INFORMATION, SOFTWARE, PRODUCTS, SERVICES AND RELATED GRAPHICS OBTAINED THROUGH THE SPACEMATE APPLICATION, OR OTHERWISE ARISING OUT OF THE USE OF THE SPACEMATE APPLICATION, WHETHER BASED ON CONTRACT, TORT, NEGLIGENCE, STRICT LIABILITY OR OTHERWISE, EVEN IF SPACEMATE OR ANY OF ITS SUPPLIERS HAS BEEN ADVISED OF THE POSSIBILITY OF DAMAGES. BECAUSE SOME STATES/JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU. IF YOU ARE DISSATISFIED WITH ANY PORTION OF THE SPACEMATE APPLICATION, OR WITH ANY OF THESE TERMS OF USE, YOUR SOLE AND EXCLUSIVE REMEDY IS TO DISCONTINUE USING THE SITE.

Termination/access restriction

SpaceMate reserves the right, in its sole discretion, to terminate your access to the SpaceMate Application and the related services or any portion thereof at any time, without notice. To the maximum extent permitted by law, this agreement is governed by the laws of the State of Florida and you hereby consent to the exclusive jurisdiction and venue of courts in Florida in all disputes arising out of or relating to the use of the SpaceMate Application. Use of the SpaceMate Application is unauthorized in any jurisdiction that does not give effect to all provisions of these Terms, including, without limitation, this section.

You agree that no joint venture, partnership, employment, or agency relationship exists between you and SpaceMate as a result of this agreement or use of the SpaceMate Application. SpaceMate's performance of this agreement is subject to existing laws and legal process, and nothing contained in this agreement is in derogation of SpaceMate's right to comply with governmental, court and law enforcement requests or requirements relating to your use of the SpaceMate Application or information provided to or gathered by SpaceMate with respect to
such use. If any part of this agreement is determined to be invalid or unenforceable pursuant to applicable law including, but not limited to, the warranty disclaimers and liability limitations set forth above, then the invalid or unenforceable provision will be deemed superseded by a valid, enforceable provision that most closely matches the intent of the original provision and the remainder of the agreement shall continue in effect.

Unless otherwise specified herein, this agreement constitutes the entire agreement between the user and SpaceMate with respect to the SpaceMate Application and it supersedes all prior or contemporaneous communications and proposals, whether electronic, oral or written, between the user and SpaceMate with respect to the SpaceMate Application. A printed version of this agreement and of any notice given in electronic form shall be admissible in judicial or administrative proceedings based upon or relating to this agreement to the same extent and subject to the same conditions as other business documents and records originally generated and maintained in printed form. It is the express wish to the parties that this agreement and all related documents be written in English.

Changes to Terms

SpaceMate reserves the right, in its sole discretion, to change the Terms under which the SpaceMate Application is offered. The most current version of the Terms will supersede all previous versions. SpaceMate encourages you to periodically review the Terms to stay informed of our updates.

Contact Us

SpaceMate welcomes your questions or comments regarding the Terms:

Email Address:
spacemateapp@gmail.com

Effective as of May 01, 2016
Milestone Chart

Our development process will be split into two segments: design and development. Over the summer months our team will work to put together the front end of our three platforms. Our goal is to come back on August 22nd with the look and feel our applications put together. From there, when we are all back together, we will focus on the functionality of our site. Below is a table of our Summer milestones.

<table>
<thead>
<tr>
<th>Date</th>
<th>Milestone</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 5&lt;sup&gt;th&lt;/sup&gt; - May 30&lt;sup&gt;th&lt;/sup&gt;</td>
<td>Transition to summer arrangements and adjust.</td>
</tr>
<tr>
<td>June 1&lt;sup&gt;st&lt;/sup&gt; - June 7&lt;sup&gt;th&lt;/sup&gt;</td>
<td>Frontend Development Phase 1</td>
</tr>
<tr>
<td></td>
<td>● Platform leaders</td>
</tr>
<tr>
<td></td>
<td>○ Set up their projects</td>
</tr>
<tr>
<td></td>
<td>○ Define tasks to be completed</td>
</tr>
<tr>
<td></td>
<td>○ Assign tasks to individuals</td>
</tr>
<tr>
<td>June 8&lt;sup&gt;th&lt;/sup&gt; - July 30&lt;sup&gt;th&lt;/sup&gt;</td>
<td>Frontend Development Phase 2</td>
</tr>
<tr>
<td></td>
<td>● Complete assigned frontend development tasks</td>
</tr>
<tr>
<td></td>
<td>● Document all work in it’s respective GitHub Wiki</td>
</tr>
<tr>
<td>August 1&lt;sup&gt;st&lt;/sup&gt; - August 19&lt;sup&gt;th&lt;/sup&gt;</td>
<td>Frontend Quality Assurance Testing</td>
</tr>
<tr>
<td></td>
<td>● Conduct individual QA tests</td>
</tr>
<tr>
<td></td>
<td>● Make notes of</td>
</tr>
<tr>
<td></td>
<td>○ Things you like</td>
</tr>
<tr>
<td></td>
<td>○ Things you would change</td>
</tr>
<tr>
<td></td>
<td>○ Things that need to be changed</td>
</tr>
<tr>
<td>August 22&lt;sup&gt;nd&lt;/sup&gt;</td>
<td>Fall Planning Team Meeting</td>
</tr>
<tr>
<td></td>
<td>● Discuss progress from summer</td>
</tr>
<tr>
<td></td>
<td>● Go over everyone's QA notes</td>
</tr>
<tr>
<td></td>
<td>● Create a development plan for the Fall</td>
</tr>
<tr>
<td></td>
<td>Things to be completed in the Fall</td>
</tr>
<tr>
<td></td>
<td>● Backend API</td>
</tr>
<tr>
<td></td>
<td>● Peer-to-Peer Chat</td>
</tr>
<tr>
<td></td>
<td>● Machine Learning</td>
</tr>
</tbody>
</table>
References

[10] https://cocoaapods.org/
[12] https://github.com/Alamofire/Alamofire
[13] https://github.com/Alamofire/AlamofireImage
[14] https://github.com/bignerdranch/Freddy
[16] https://developers.facebook.com/docs/facebook-login/ios
[18] https://buddybuild.com/
[22] https://cloud.google.com/sql/pricing
[26] https://github.com/google/gson
[33] https://github.com/brigade/scss-lint
[34] http://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html
[36] https://cloud.google.com/sql/faq#benefits
[37] https://letsencrypt.org/
[40] https://labs.ribot.co.uk/approaching-android-with-mvvm-8ceec02d5442#.2p5ig08px
[48] https://github.com/github/swift-style-guide
[49] https://github.com/realm/SwiftLint
[50] https://angular.io/features.html
[51] https://material.angularjs.org/latest/
[52] https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
[55] https://docs.angularjs.org/api/ngRoute/service/$route
[56] https://docs.angularjs.org/api/ng/service/$http
[57] http://www.captcha.net/
[58] http://gulpjs.com/
[60] https://www.npmjs.com/