

BU.ROOMIE4.ME

An Advanced Roommate Search System



Team Members [#8]:

~~Al Hotinski~~ (~~hotinski@bu.edu~~)

Muhammad Aseef Imran (aseef@bu.edu)

Munir Siddiqui (munirsid@bu.edu)

William Chan (willchan@bu.edu)

TABLE OF CONTENTS

TABLE OF CONTENTS.....	1
PROJECT GOALS.....	3
DESIGN PHILOSOPHY.....	4
PROJECT REQUIREMENTS.....	5
Key.....	5
Features Overview.....	5
INTERFACE AND PAGES.....	7
DATA OBJECTS.....	11
roomie-request:.....	11
user:.....	11
error:.....	13
RESTFUL API.....	14
GET /roomie/requests.....	14
Parameters.....	14
Returns.....	14
On Success:.....	14
On Failure:.....	14
GET /roomie/request.....	14
Parameters.....	14
Returns.....	15
On Success:.....	15
On Failure:.....	15
PUT /roomie/create.....	15
Body.....	15
Returns.....	15
On Success:.....	15
On Failure:.....	15
POST /roomie/update.....	15
Body.....	15
Returns.....	15
On Success:.....	15
On Failure:.....	16
DELETE /roomie/delete.....	16
Returns.....	16
On Success:.....	16
On Failure:.....	16
GET /user/myinfo.....	16

Returns.....	16
On Success:.....	16
On Failure:.....	16
GET /user/find.....	16
Parameters.....	16
Returns.....	17
On Success:.....	17
On Failure:.....	17
PUT /user/create.....	17
Body.....	17
Returns.....	17
On Success:.....	17
On Failure:.....	17
POST /user/update.....	17
Body.....	17
Returns.....	17
On Success:.....	17
On Failure:.....	18
DELETE /user/delete.....	18
Returns.....	18
On Success:.....	18
On Failure:.....	18
Parameters.....	19
Returns.....	19
TECHNOLOGY STACK.....	20
DELEGATION OF ROLES.....	21
Al Hotimski.....	21
Muhammad Aseef Imran.....	21
Munir Siddiqui.....	21
William Chan.....	21
TIMELINE.....	22

PROJECT GOALS

- To provide BU Students (and perhaps students from other universities) an effective way to discover and connect with potential roommates.
- To master technologies such as:
 - Git
 - Node JS
 - JS/Typescript
 - Reactas a group
- Master the art of working in a team.
- Learn how to deploy web apps to the internet using modern technologies.

DESIGN PHILOSOPHY

Due to time constraints and uncertainty pertaining to the duration required to implement each feature, the most logical route is to pursue an agile design philosophy. Particularly, aspects of this project are to be broken into independent components that can be optionally completed depending on the available time and resources.

As outlined in the “Project Requirements” section, this project will contain certain “core features” which absolutely must be completed in order to output a functional product and fulfill all project criteria. In addition to these “core features,” certain optional features are also described. Team Members design core features to keep in mind the future vision for the project in order to ease future extension for the project on “day 2” releases.

PROJECT REQUIREMENTS

Key

Core Feature

Important Feature

Non-Important Feature

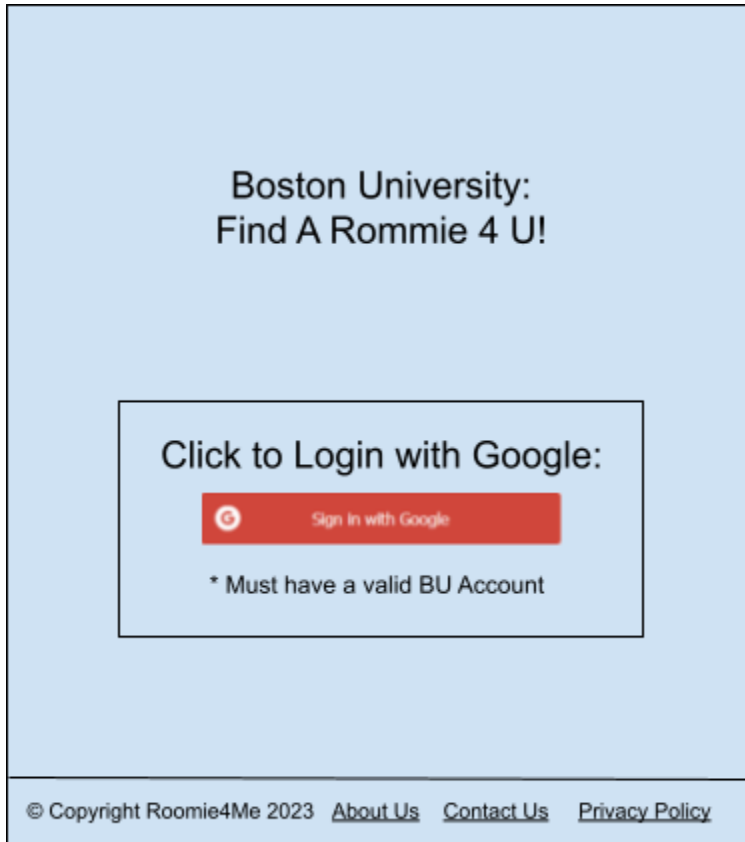
Features Overview

- ☐ Google OAuth login into the site only allowing students with a @bu.edu email to proceed.
 - ☐ The user's account token/cookies are stored in the database so they don't have to login every time.
- ☐ Ability to open a roommate request. This "request" is filed by filling a form. The form ask the following questions which is then used to create a [user](#) object:
 - First Name
 - Last Name
 - Gender
 - Age
 - Phone Number
(The user is notified that this is never shared without their explicit consent)
 - International?
 - Languages
 - Expected Year of Graduation
 - Major
 - On Campus/Off Campus
 - Do you already have a residence?
 - ☐ Enter Address: (*API 1: address validation)
 - About me description
 - ☐ *API 2: Uses AI-Language models to automatically tag text
(used for auto-roommate recommendations)
 - Do you smoke/drink?

- Sleep schedule.
- An algorithm to suggest the most compatible roommate using a weighted similarity metric.
 - Ability to control the “weights” on this similarity matcher
- Once you have created a roommate request, you can also browse other roommate requests and send a pairing request to someone else (or multiple people).
- ***API 3:** Email or Text Message API to reach out to you to send you notifications.
- An account overview page allows you to edit your profile, including your name, DOB, and year of graduation. Allows you to delete your account.
- An IRC panel to chat with your potential roommate right on the site.
 - ***API 4:** Uses an API to implement this functionality:
<https://www.weavy.com/free-chat>

INTERFACE AND PAGES


Login Page



The screenshot shows a light blue login page for Boston University. At the top, the text "Boston University: Find A Rommie 4 U!" is centered. Below this, a white box contains the text "Click to Login with Google:" followed by a red "Sign in with Google" button. Under the button, it says "* Must have a valid BU Account". At the bottom of the page, a footer contains the copyright notice "© Copyright Roomie4Me 2023" and three links: "About Us", "Contact Us", and "Privacy Policy".

Boston University:
Find A Rommie 4 U!

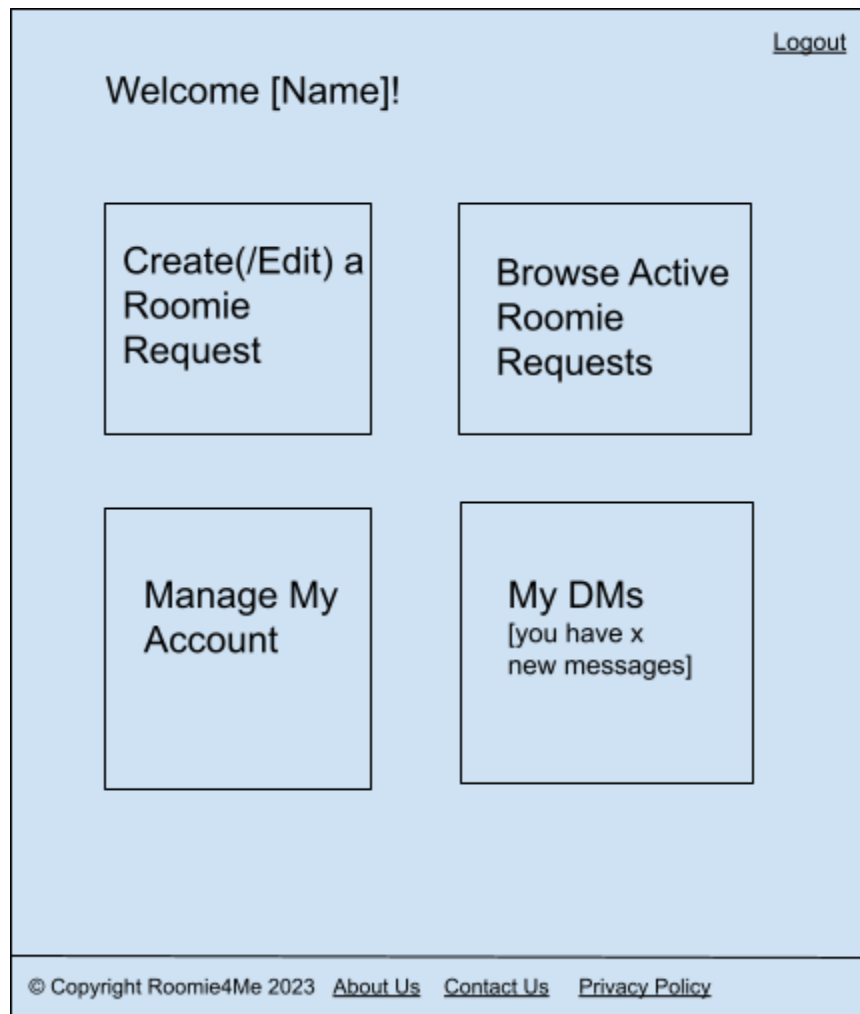
Click to Login with Google:

 Sign in with Google

* Must have a valid BU Account

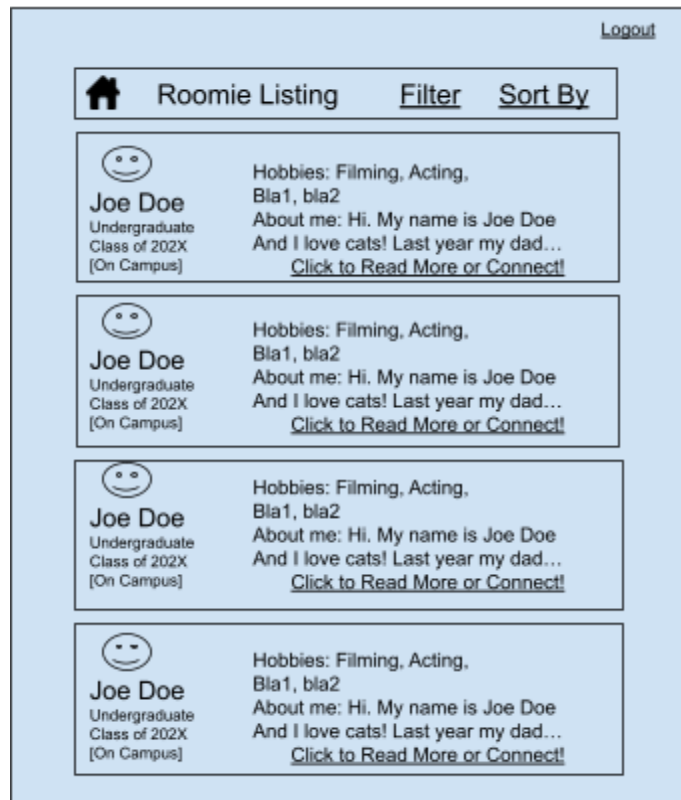
© Copyright Roomie4Me 2023 [About Us](#) [Contact Us](#) [Privacy Policy](#)

- Registration Page
- Main Selection Menu



If you don't have an active Roomie request, the button says "Create a Roomie Request." Otherwise, it says, "Edit a Roomie Request."

- **Active Room Mate Requests Lists**




Note 1: The smile face is substitute for profile picture.

Note 2: Home icon returns user to the home page.

- IRC Messaging Page
- Account Management Page

[Logout](#)

 **My Account**

BU Email: jdoe@bu.edu

First Name: John [edit](#)

Last Name: Doe [edit](#)

Gender: Male [edit](#)


Year of Graduation: 2026 [edit](#)

Major(s): CS, Electrical [edit](#)

Minor(s): Business [edit](#)

...todo

Profile Picture [edit](#)



© Copyright Roomie4Me 2023 [About Us](#) [Contact Us](#) [Privacy Policy](#)

- Roommate Request Forum
- About/Contact Page

DATA OBJECTS

This section describes the objects that make up the site including users, roommate requests, and more.

roomie-request:

- id: the unique integer request id.
- target semester: a string indicating the target semester for which you are looking for a roommate.
- number of roomies: an integer indicating the number of roomies you are looking for.
- housing info: this section indicates where you want to live/are already living. (ie whether you are looking to pull-in someone or you are living on/off campus, etc..)
 - has housing: boolean that is true if the user already has housing
 - address: a validated address of where the housing is (null if has housing is false).
- creation: the timestamp of when this request was created
- request expiry: the time after which, if the request is not fulfilled, it will expire.
- status: an enum(completed, pending, open) indicating whether this request has been fulfilled, is in progress, or is still open.

loggedInUser:

- user: reference to the user object
- registered: a boolean that indicates whether the user is fully registered and has filled out the registration form
- auth token: the oAuth token from Google.

user:

- id: the unique string id for the user from Google
- name
 - first name: a string representing the user's first name

- last name: a string representing the user's last name
- age: an integer representing the user's age
- gender: an enum(male, female, non-binary, other) indicating the user's gender
- languages: an array of strings that each represent a language the user can speak.
- religious affiliation:
 - religion: an enum(christianity, islam, hinduism, buddhism, sikhism, judaism, bahá'í, jainism, shinto, Cao Dai, Zoroastrianism, Tenrikyo, Animism, Neo-Paganism, other, unaffiliated) indicating the user's religious affiliation
 - scale: an integer between 1-5 indicating how religious the user considers themselves to be
- international:
 - is international: true if the user is an international student
 - country: a nullable string array representing the country(ies) the user is from. Null when the user is not an international student.
- degree program:
 - majors: an array of strings
 - minors: an array of strings
 - year of graduation: an integer indicating the expected year of graduation
- drugs:
 - smoke: an enum(often, sometimes, rarely, never) indicating how often the user smokes (including tobacco and weed).
 - vape: an enum(often, sometimes, rarely, never) indicating how often the user vapes.
 - drink: an enum(often, sometimes, rarely, never) indicating how often the user drinks alcohol.
 - other recreational drugs: an enum(often, sometimes, rarely, never) indicating how often the user takes other recreational drugs.
- sleep:
 - weekdays:

- bedtime: a time (with a granularity of every 30 min) indicating when the user usually goes to bed on weekdays.
 - wake time: a time (with a granularity of every 30 min) indicating when the user usually wakes up on weekdays.
- weekends:
 - bedtime: a time (with a granularity of every 30 min) indicating when the user usually goes to bed on the weekends.
 - wake time: a time (with a granularity of every 30 min) indicating when the user usually wakes up on weekends.
- about me: a long string with a user's own description of themselves. This string is then automatically "[tagged](#)."

error:

- reason: An array of strings that represent the reason(s) for the error that will be displayed on the front end.

RESTFUL API

Base URL: <https://bu.roomie4.me/api/v1>

All requests to the Restful API should include the following headers:

- Content-Type: application/json
- Authorization: Bearer {token}

The authorization token is the same as the user's google auth token.

GET /roomie/requests

Retrieves the list of roommate requests

Parameters

page-number (0 or higher): The page number which to get the requests of.

batch-size (1-25): The size of each page.

sort-mode: The enum sorting mode to use.

active-filters: The filters to apply as an integer bit mask

Returns

On Success:

An array of [roomie requests](#)

On Failure:

An [error](#) with the appropriate HTTP status code.

GET /roomie/request

Retrieves the list of roommate requests

Parameters

id: The unique id of the desired roomie request.

Returns

On Success:

The [roomie request](#) with the specified id (if request successful) or error

On Failure:

An [error](#) with the appropriate HTTP status code.

PUT /roomie/create

Create a new roommate request.

Body

A JSON object representing a [roomie request](#).

Returns

On Success:

Empty response (with an HTTP status 200)

On Failure:

An [error](#) with the appropriate HTTP status code.

POST /roomie/update

Updates an existing roommate request

Body

A JSON object representing an updated [roomie request](#).

Returns

On Success:

A JSON object representing a [status](#).

On Failure:

An [error](#) with the appropriate HTTP status code.

DELETE /roomie/delete

Delete the user's current roomie request (if any). We know who the user is using their authorization token.

Returns

On Success:

Empty response (with an HTTP status 200)

On Failure:

An [error](#) with the appropriate HTTP status code.

GET /user/login

Logs the user in and retrieves the API token needed to make API requests to the backend

On Success:

A JSON object that contains a [user](#) as a subdocument and a field called "apiToken".

On Failure:

An [error](#) with the appropriate HTTP status code.

GET /user/myinfo

Returns information about my user (uses authorization key to find who you are)

Returns

On Success:

A JSON object representing a [user](#).

On Failure:

An [error](#) with the appropriate HTTP status code.

GET /user/find

Get info about a specific user

Parameters

id: the user's unique id

Returns

On Success:

Empty response (with an HTTP status 200)

On Failure:

An [error](#) with the appropriate HTTP status code.

PUT /user/create

Creates a new user for me.

Body

A JSON object representing a [user](#).

Returns

On Success:

Empty response (with an HTTP status 200)

On Failure:

An [error](#) with the appropriate HTTP status code.

POST /user/update

Updates my user data.

Body

A JSON object representing an updated [user](#).

Returns

On Success:

Empty response (with an HTTP status 200)

On Failure:

An [error](#) with the appropriate HTTP status code.

DELETE /user/delete

Deletes the user's account and all data associated with it (we figure out who the user is using their authentication token).

Deletes my account.

Returns

On Success:

Empty response (with an HTTP status 200)

On Failure:

An [error](#) with the appropriate HTTP status code.

Parameters

No parameters

Returns

A JSON object representing a [status](#).

TECHNOLOGY STACK

Selected Technology Stack

We intend to use the MERN technology stack for our project alongside Bootstrap Studio. Here is further detail:

- Frontend framework:
 - React.js: Due to its popularity and the number of open jobs requiring knowledge of React, we think React is a worthwhile framework to learn. Moreover certain components like “the footer” are repeated throughout different pages. React has a feature to reuse such components which results in cleaner and more maintainable code. In pure HTML, if we want to make an edit to the footer, we would have to edit every single page. However, react saves us from this.
 - Bootstrap + Bootstrap Studio: Bootstrap is a CSS framework that is responsive and mobile-friendly. Sites made in this framework are designed to automatically adjust to a mobile view thanks to its grid system. To assist with the development, we are also using Bootstrap Studio which is a GUI-based HTML/CSS editor that is designed to be used for designing Bootstrap sites. A GUI program like Bootstrap not only lowers the learning curve of Bootstrap but is simply faster which is why we went with this technology stack. We will then convert the generated HTML and CSS into react components.
- Backend Framework:
 - Node JS + Express: Because it is written right in Javascript/Typescript. And we cannot afford to
- Database
 - MongoDB: Because we will be manipulating JSON data from API's and MongoDB is the most convenient database for this.

Alternative Technology Stack

Our team also considered using the MEAN stack instead of MERN. In other words, we considered using Angular instead of React. However, our research suggested that Angular (despite being more flexible) presents a significantly steeper learning curve. Given our time constraints, we decided to opt for using React. In addition, our research also showed that there are simply more job opportunities for React developers in the current environment. Given this, React for the natural choice.

DELEGATION OF ROLES

Al Hotimski

- Front End
 - Designing the Interface using HTML, CSS, and (possibly some Javascript)

Muhammad Aseef Imran

- DevOps
- Project Specifications
- Tooling
- Domain Routing
- Team Coordination

Munir Siddiqui

- Front End
 - Using react to interface with the Backend

William Chan

- Backend
 - Database connection
 - Restful API (node js)

TIMELINE

Task	Due Date	Assigned To