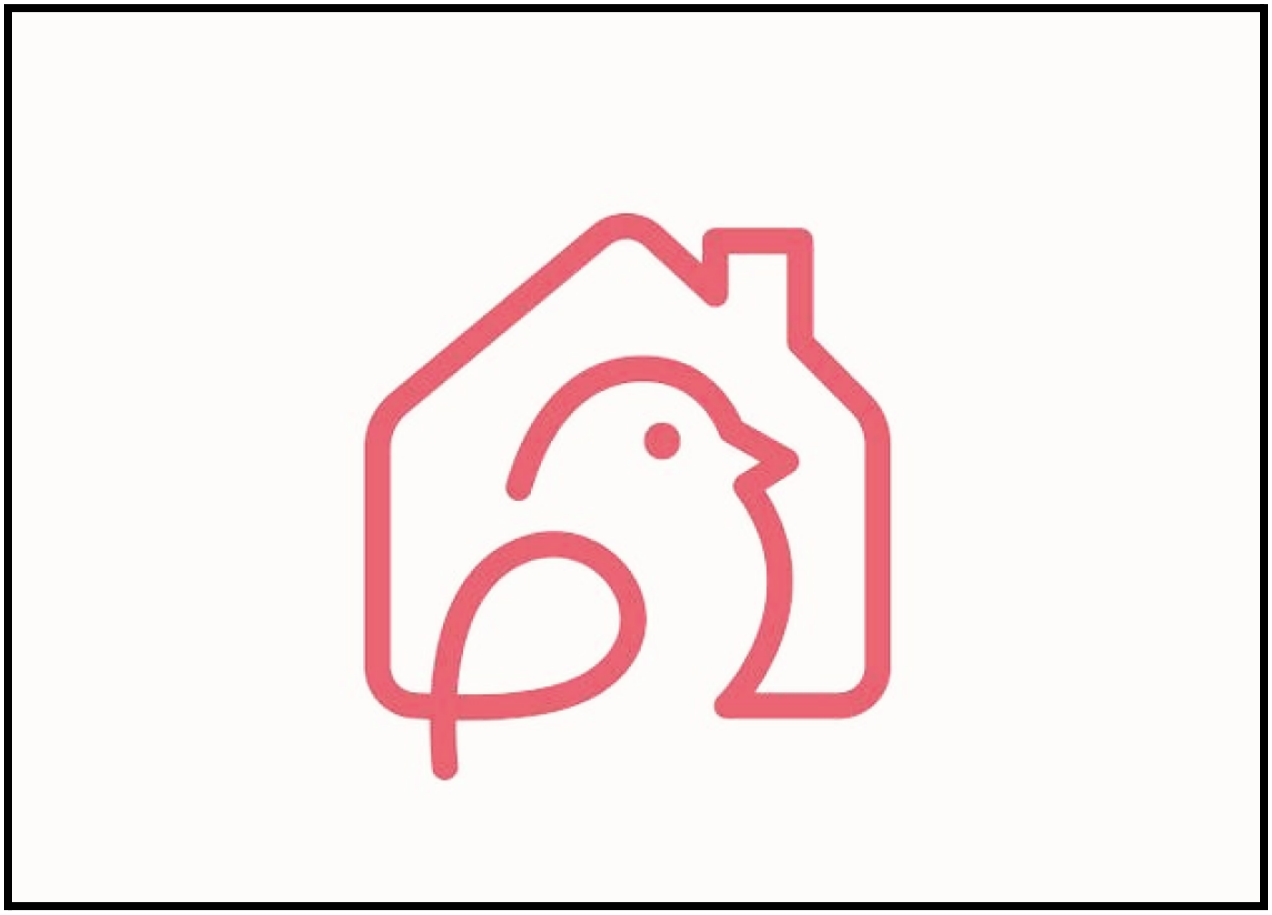
UNEST

CS 307 - Design Document



Team 13

Sudhanva Bharadwaj

Nivedha Kumar

Ram Laxminarayan

Grant McCord

Index

Purpose:

Our project is to develop an online marketplace for university students to post and view available subleases near their campus. Currently, students use forms of social media such as Facebook Marketplace to find subleases, but these platforms are less organized and students cannot directly rent rooms through the site. Also, websites like apartments.com and airbnb only allow property owners to post or they are very expensive due to fees. Our website would be an organized interface displaying information regarding subleases and would allow students to easily identify properties based on their needs.

The home page will have all the listings and from there, if a user clicks on a listing to get further details, it will pop up a user registration page. After signing up or logging in, they will be able to see a new screen with more details about a specific listing and have the option to reserve the space. After deciding to book the space, the subleaser will be able to message the subletter through a chat interface. Subletters will have the additional option to create a new listing by filling out a form on a separate page and clicking post. Subletters will also have access to a page displaying all of their own listings.

Design Outline:

High Level Overview:

We will be using the MERN stack, which includes four languages: mongodb, express.js, react, and node.js. The React layer creates the frontend components that the client interacts with. Express.js and Node.js make up the middle or application layer, and MongoDB will be used to create the database layer.

Our application will use a client-server model with the following components:

1. Client
    a. The client side will be the interface to our product.
    b. The client will send and receive data from the server through HTTP requests.
    c. The data received from the backend will be parsed and formatted to display information clearly to the user.
    d. The client will be a React app. React is a component based framework. Components are reusable pieces of UI and they also make the source code more organized. React is also known for its ability to quickly react to changes in state and render those changes immediately on the screen.
2. Server
    a. The server accepts HTTP requests from the client and processes them.
    b. The server will then query the database to update or retrieve information from the tables.
    c. The server will finally return the data from the database as a JSON object as part of the HTTP response.
    d. The Express Web Framework will run on the Node JS web server in the backend.
    e. Express.js is responsible for much of the API handling and routing. It will be used to do the following steps and transition between them: it dissects the api call url link that was sent to the server and then redirects control to the router, controller, dao, and then database in that order. The router directs to the correct controller that will execute the task that was requested by the user.

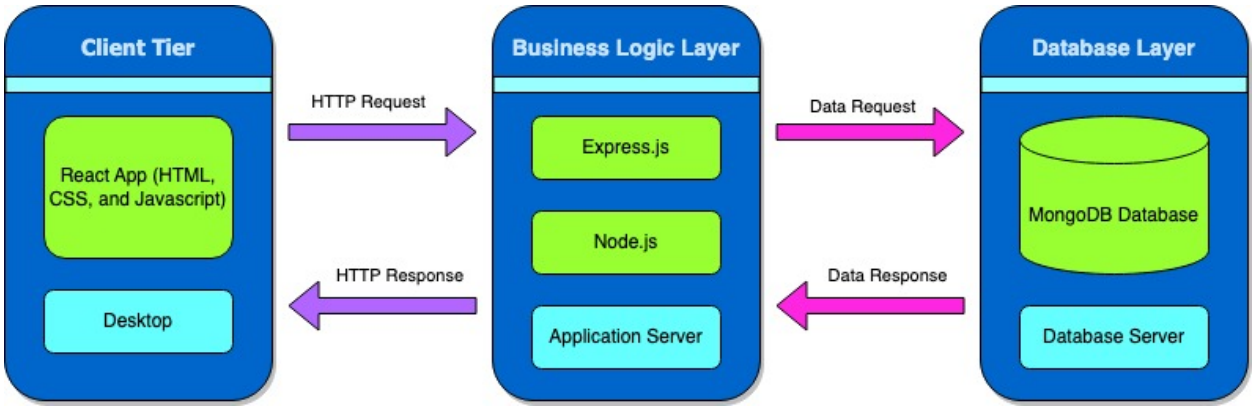
f. Node.js is an open-source JavaScript runtime environment that allows developers to run JavaScript code on the server side.

3. Database
    a. The database will securely store all the data such as user profile and property information for our UNest application
    b. We will be using a MongoDB database. MongoDB is a NoSQL database language, which utilizes JSON-like documents with flexible schemas.
    c. MongoDB Atlas is a MongoDB database service running on the cloud and will securely store our data.

The react app is running on a browser on the user's local machine, the server will be running in a datacenter, and the database will be running in the cloud.

Interactions Between System Components:

Here is a diagram with how the different system components interact with each other:



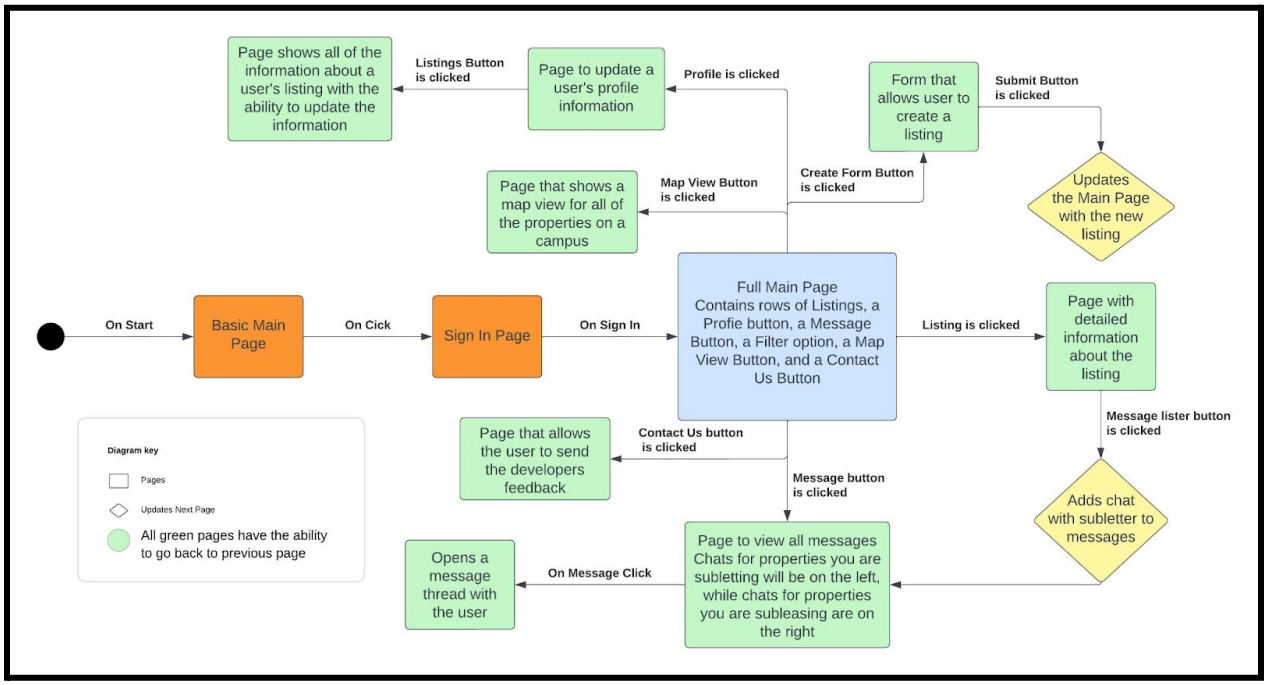
The user first starts the web app. The client interacts with the website by for instance clicking on a button to pull up a page with the information about a particular listing, which will trigger an api call. In that scenario, a GET http request is sent to the server, the server processes the request, and then the server makes a query to the database to retrieve information to put on the detailed information about a listing page. The database sends the information back up to the server,

which then sends the data as an http response to the frontend to be displayed on the page. Similarly, the user can also trigger POST, PUT, and DELETE http requests.

In our application, there will be many client-server interactions. For example, users will post a listing which would add that listing to a listing table in the database. The user could also update the property information in one of their listings. Similarly, when they sign up, a new user profile is added to the database, and the user will have an option to edit that as well. Also, a list of people the user interacts with and the messages exchanged between them will all also be stored in a table in the database. Furthermore, whenever they visit a new page, the information to be displayed on that page will need to be received from the database.

Navigation Flow Diagram:

The diagram illustrates how the user can navigate to the different pages of the application.



Design Issues

Functional Issues

1. What information should users be able to view on the main page of our site without signing in?
   Option 1: They shouldn't be allowed to view any information.
   Option 2: Only basic information about each property on the normal homepage.
   Option 3: They should have full access to the site.
   Choice: We chose option 2.
   Discussion: We thought it would be convenient for users to be able to see basic information about listings without having to sign in, but users will have to sign in to view any detailed information about the property to make sure the only people that can view lister's personal information are students.
2. Where should we include messages from other users about your sublet?
   Option 1: Include them in a separate tab in the messages page.
   Option 2: Include them as additional information to see when you view your listing.
   Choice: We chose option 1.
   Discussion: We thought it would be more convenient to have all of a user's messages in one place, but we were worried things would be too cluttered. We decided to split the page with a section for the properties a user is subletting on the left and a section for the properties the user is subleasing on the right.
3. Should we use different account types for subletters and subleasers?
   Option 1: Both subletters and subleasers use the same account type.
   Option 2: Users must choose their account type when signing up.
   Choice: We chose option 1.
   Discussion: Our original thought was to separate account types, but we figured that some students may want to both sublet and sublease. After discussing, we decided that it would make more sense if everyone had the same access to features, so we blended the two account types and anyone can create a listing and view all available listings.
4. What format should we use if a lister wants to update their listing?
   Option 1: Use the same format as when they created the listing, but auto populate the text fields with the current information.

Option 2: Show the lister what potential tenants see when they view the listing, with the lister being able to modify the fields in that format.

Choice: We chose option 1.

Discussion: We decided to use the same page for creating and updating listings since we would be reusing a lot of the same information anyway. The main differences would be needing to add a delete listing button if it has already been created and to repopulate the text fields with the existing information. We decided against using a similar format to the regular view listing page since we figured that listers would prefer seeing the information in an easy to update format rather than the viewing format.

5. Should we put the total number of bedrooms and bathrooms or the number of private bedrooms and bathrooms on the main page?

   Option 1: Total bedrooms and bathrooms

   Option 2: Private bedrooms and bathrooms

   Option 3: Putting both of them together.

   Choice: We chose option 2.

   Discussion: We decided to put private bedrooms and bathrooms because we decided it was more important information for users to see. We figured that users would care more about what they would get from the sublease for themselves over the number of people they have to live with or the total number of bathrooms in an apartment. We decided against putting both together as we decided it was too confusing with multiple numbers close together seemingly referencing the same thing. By splitting up the total vs. private bedrooms and bathrooms it should make it more readable for the users.

Nonfunctional Issues

1. What programming languages/tools will we use for the project?

   Option 1: MERN tech stack (MongoDB, Express.js, React, Node.js)

   Option 2: ASP.net Core (Microsoft framework)

   Option 3: React, Java Spring Boot, Google Firebase

   Choice: We chose option 1.

   Discussion: We decided to use the MERN tech stack because it was what the team was most comfortable with. Several members on our team had worked with React and MongoDB, but we only had one member who had used Firebase. With MERN tech

stack's rising popularity in the workplace, we decided it would be the best to become more familiar with.

2. How will we develop the backend?

   Option 1: Develop the backend from scratch.

   Option 2: Use a backend API.

   Choice: We chose option 2.

   Discussion: We decided that using a backend API would greatly simplify one of the incredibly important beginning processes for our app, and would allow us to link the different pages of our app together easily. The sign in feature is also not incredibly important for making an original application, so it would be better if we spent more time working on the core features of the web app rather than on a feature the user will interact with minimally.

3. How should we verify student emails?

   Option 1: Use a verification API.

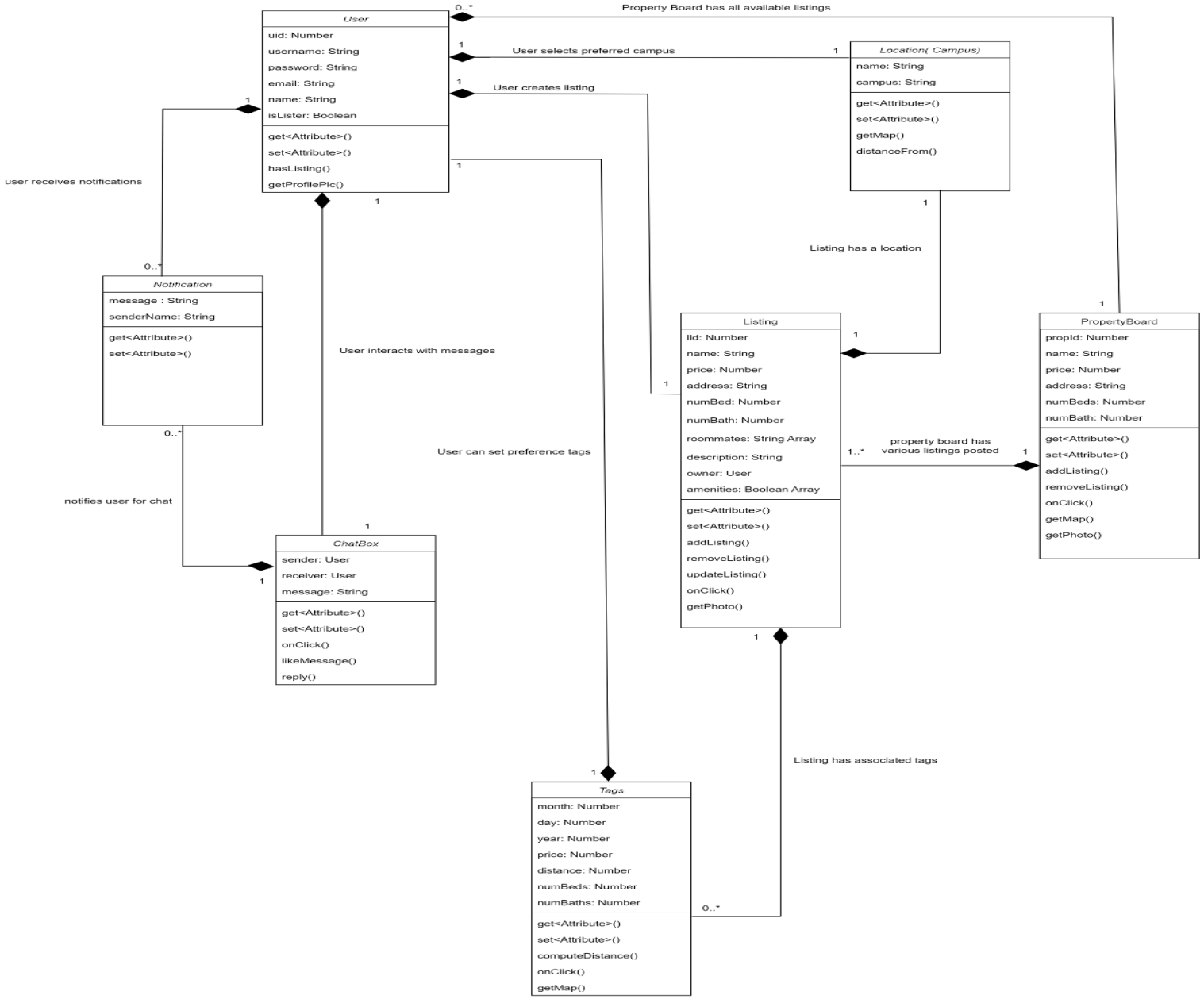
   Option 2: Check for a .edu account and send an email to the account.

   Choice: We chose option 2.

   Discussion: While a verification API sounds more professional, simply checking if the account is an educational account allows us to rely less on API calls and focus more on making most of the core functionality ourselves. By checking for a .edu account and then sending an email to that account we can assure that all users are affiliated with a school since only people affiliated with a school can have a .edu account. If malicious users try to get around it by using a fake .edu account, they will be required to sign in to the account in order to click the verification email, which will stop them from using fake accounts. This should provide us with a good amount of security and allow us to focus our attention on other features more integral to the project.

Design Details

Class Diagram:



Descriptions of Classes and Interaction between Classes

These classes in our class diagram refer to objects in our application that contain attributes and methods that describe the features of each object, and also how they interact with other classes in the application to show how the application functions optimally.

- **Users**
    - Each User Object is created when someone creates an account on our site.
    - Each User creates an account by signing up through email and creating a username and password.
    - Each User also inputs their name when creating an account.
    - After the user's account is created, they are assigned a unique id.
    - Each User will have the ability to update their username and password under their profile settings.
    - Each User can view a list of properties once they sign up and are signed in. A list of properties will be displayed on the homepage.
    - Each User can create a listing by clicking the add listing button. Once a user creates a listing, the user now has extra capabilities and can view and manage listings from a lister and leaser point of view.
    - A User who is also a lister can view and manage their own properties while also viewing other properties whereas a non-lister can only view and show interest in other owned properties.
    - Each User will be able to add, remove, and update their profile picture.
- **Listing**
    - Each Listing Object is created when a user adds a property to the website.
    - Each Listing will be assigned a unique listing id after the property is posted.
    - Each Listing will have a name, price, address, number of beds and bathroom, list of roommates, date available, and description. The name, price, address, data available, number of bedrooms, and number of bathrooms can be seen when it is shown on the homepage or when a user filters their search, but the list of roommates, owner of the property, amenities, and description of the property are only shown when a user clicks on the property.
    - Each Listing will have a photo of the property associated with it.
    - Each Listing's location will be shown on a map near the user's designated campus.

- Each Listing will have an owner designated to it who can be contacted if a user is interested in the property.
- Each Listing can be managed by the property owner, whether it is to edit details about a property, add a property, or remove a property.
- Each Listing can be filtered to best fit a user's needs and essentials in a property such as data available, price, location, and number of bedrooms and bathroom.

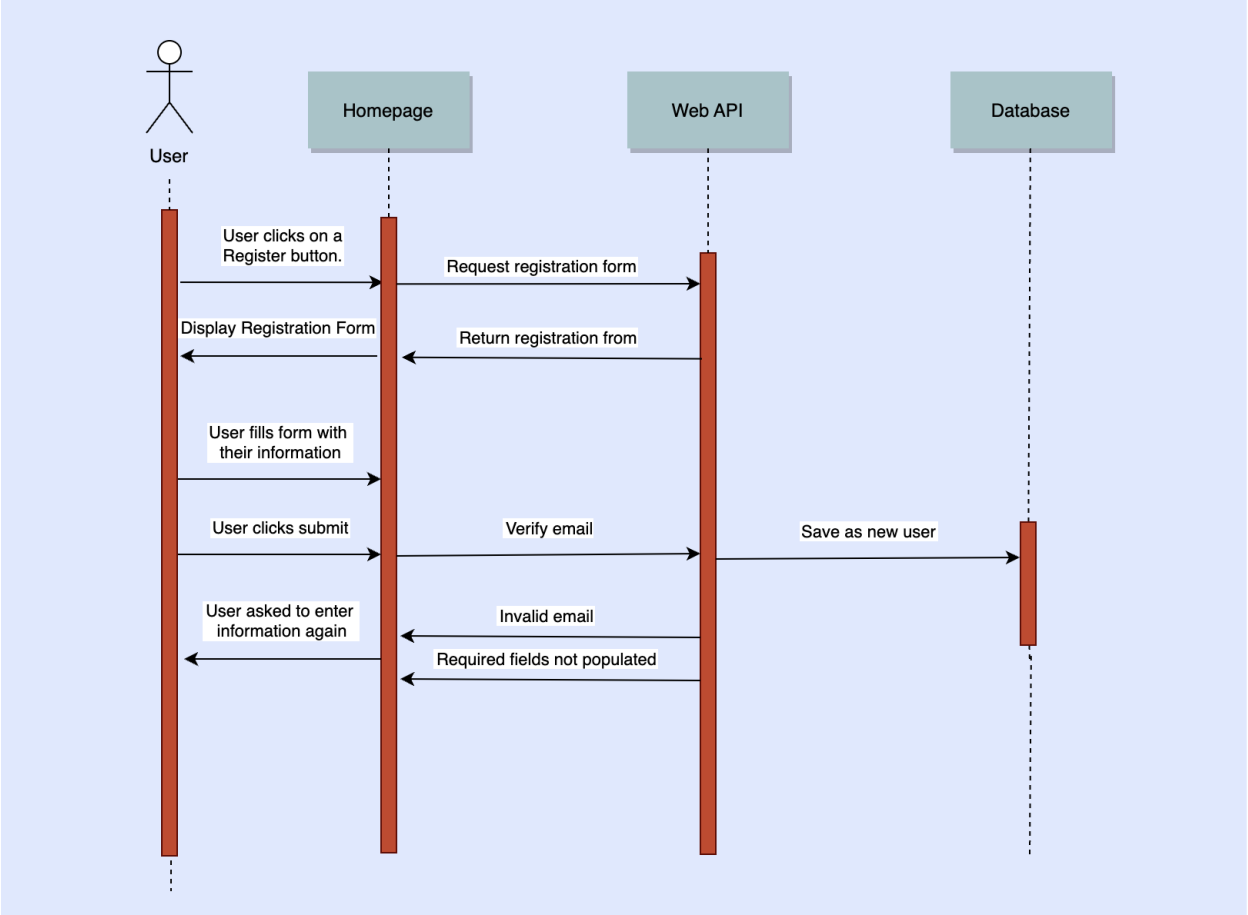
- **ChatBox**
  - A ChatBox object is created when a user receives a message from another user.
  - Each ChatBox is different based on which user is listing the property and which user is a potential tenant. A user will have a different ChatBox for each message they send/receive as a property owner, and each message they send/receive as a potential tenant.
  - Each user can receive chatbox notifications when receiving a message.
  - Each user can click on a chat with other users in the chat box.
  - Users have the option to like a message in the chat box.
- **Tag**
  - A Tag object is created when a user filters their search for properties they are seeking.
  - Each Tag object consists of the number of bedrooms and bathrooms, date range, price range, and distance range. Given a range from these attributes, listings that satisfy this criteria will be displayed.
  - Based on filtered search, a map will be shown based on where the location of most listings are.

- **Location**
  - A location object consists of the name of the location and the designated campus that the location is near.
  - Location is created when listings are created and displayed.
  - Each User can select the designated location where they want to lease a property to see a list of listings nearby on the property board.
  - The location will be displayed on a map given the address.
  - With the coordinates of the listing locations, the distance between the property and the designated campus chosen by the user will be calculated.

- **Notification**
    - Notification object consists of the name of the person who sent the message, and the message sent.
    - A Notification object is created when messages are sent through the chat box from one user to another user.
- **PropertyBoard**
    - A PropertyBoard object is created when a user creates a listing on the website.
    - PropertyBoard displays listings with attributes such as name, price, address, number of bedrooms, and number of bathrooms.
    - Each PropertyBoard object is associated with a unique property board id.
    - Each Property Board displays photos of multiple listings with each of its attributes.
    - Each Property Board displays a map with listings near the designated campus chosen that can be clicked on for more information about the listing.

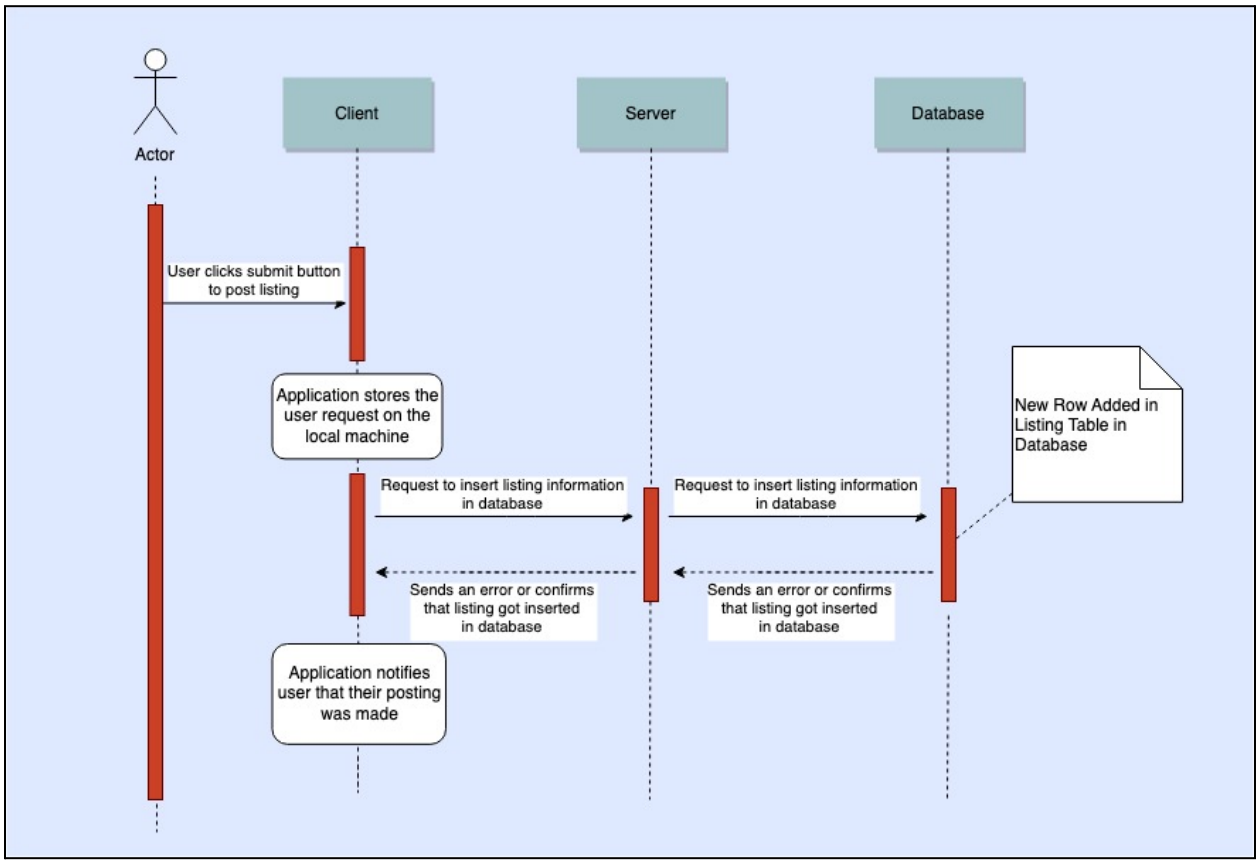
Sequence Diagrams

1. Sequence of events when a user registers



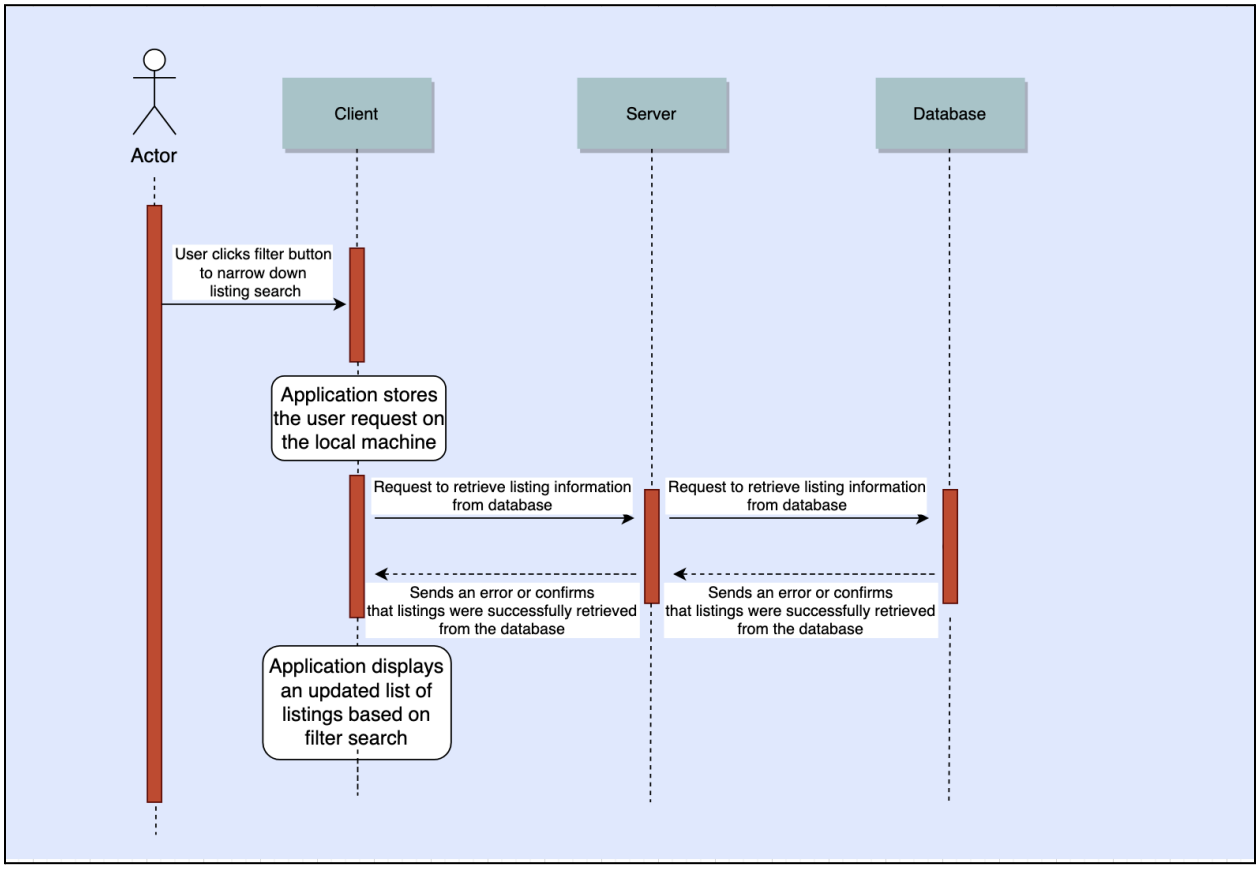
The sequence diagram above outlines the registration process for a new user. The user first clicks the “Register” button at the front end which then prompts the request for the registration page. Once this page is returned, the user will fill out all the required fields. Once the form is complete, the user will have the option to click submit at the bottom which will send a request to the authenticator to verify the email. SInce the website requires a school email, invalid emails will result in invalid field error which will ask the user to enter the information again. This process is repeated until all the fields are filled and the user enters a valid email. Then the information collected from the user is saved into the database as a new user.

2. Sequence of events when a user adds a property listing



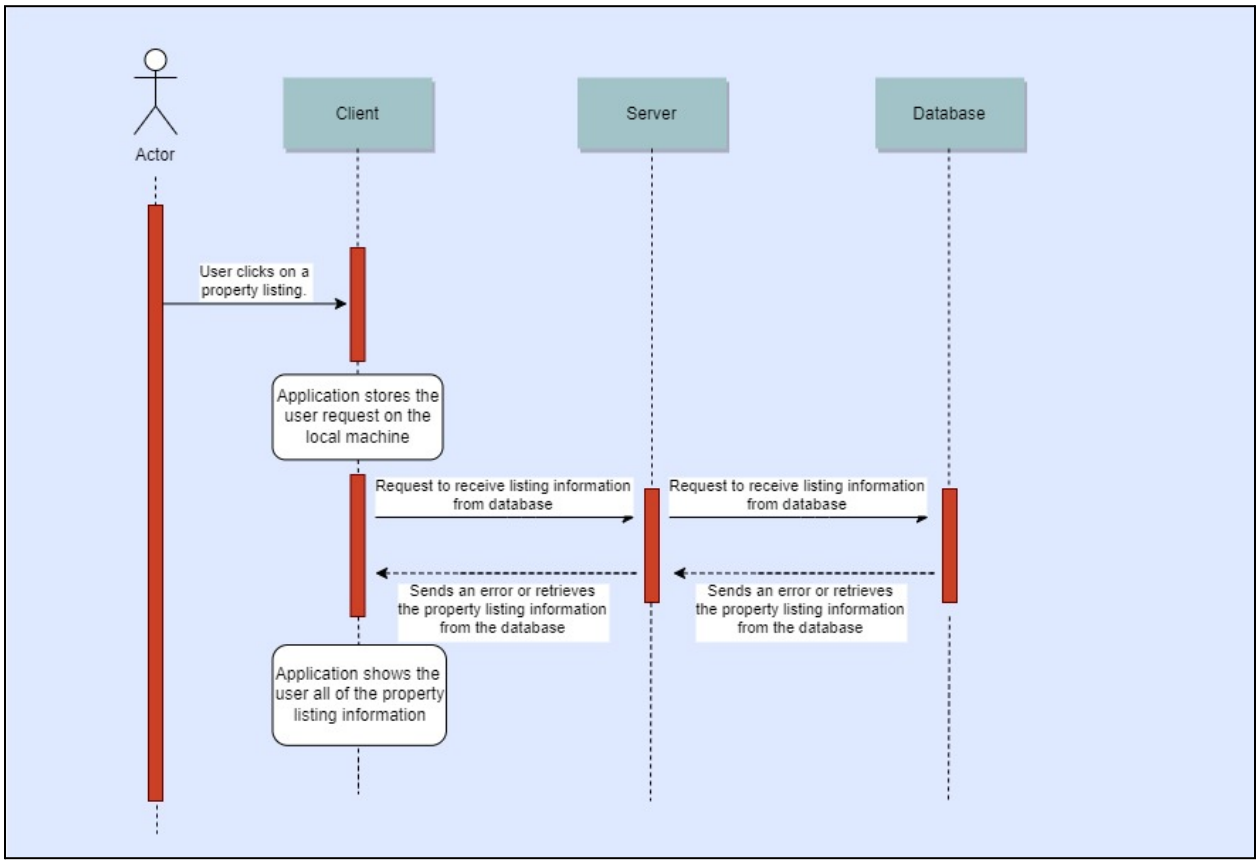
The sequence diagram above shows how the client server model is able to handle the user's request to post a listing. The sequence of events is initiated when an actor clicks the submit button to post a listing. Then the application stores the request on the local machine and sends a request to insert listing information in the table to the server, which then sends the request to the database. If the new row is added to the listing table then the database sends a confirmation message back to the server, and then it is sent back up to the user notifying them that their post was made.

3. Sequence of events when a user filters search



This sequence diagram represents a sequence of events that occur when a user wants to filter their search to find a property listing that best matches their living desires. The event first starts when a user clicks the filter button. When the button is clicked, the request gets stored in the local machine and sent to the server. The client sends the request to the server so that the server can retrieve the listings with the specific filtered search from the database. After the request is handled by the database, the data is sent back to the client and the listings that satisfy the request are displayed on the application.

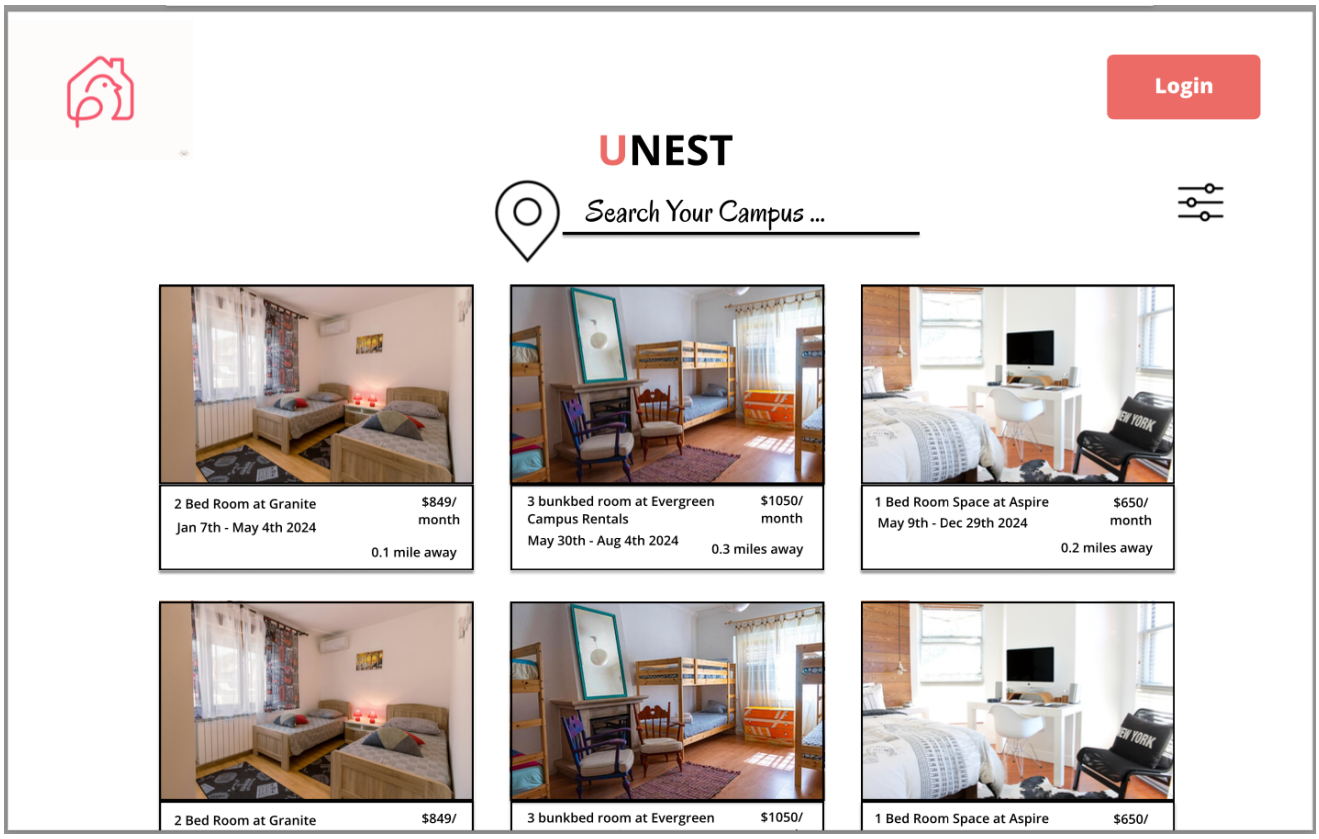
4. Sequence of events when a user clicks on a posted property listing



This sequence diagram shows the sequence of events that happen after a user clicks on a property listing. After clicking the property listing, the application stores the user's request on their local machine. Then, the request for the listing information is received by the server and sent to the database. The database will then return either an error or the property listing information to the server, which then sends the information to the client. Finally, the client's application shows the user all of the property listing's information.

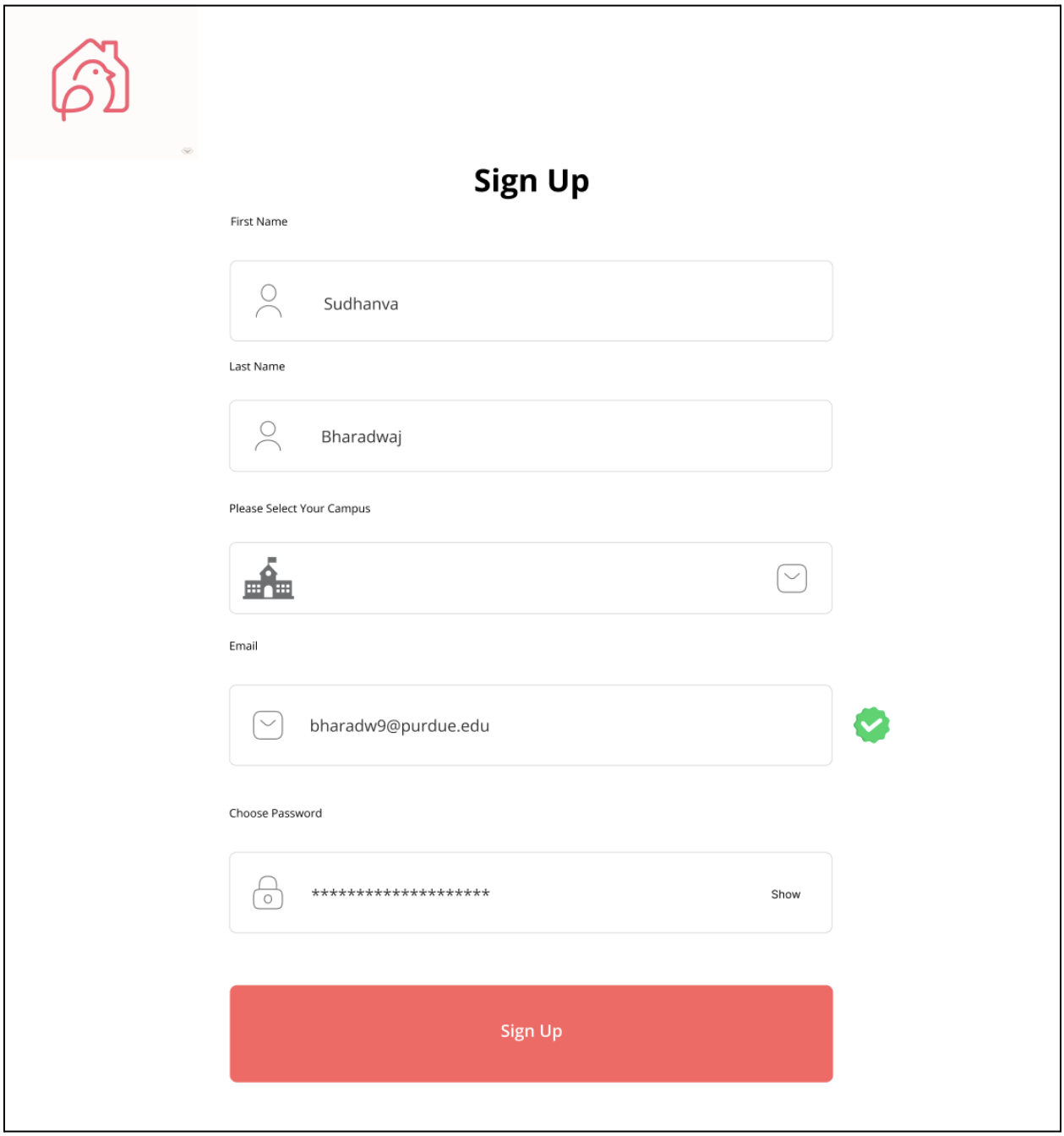
UI Mockups

Logged Out Page



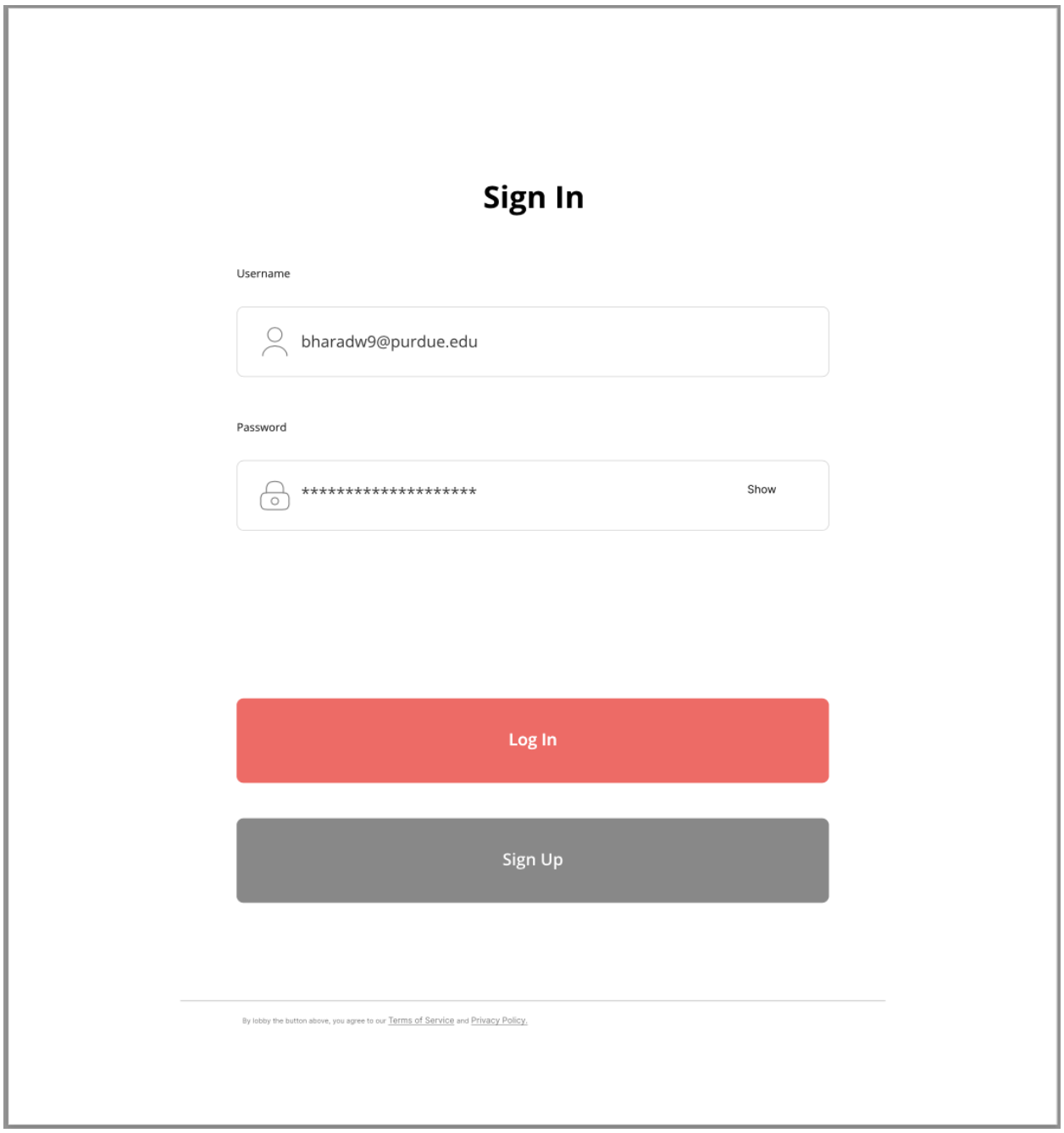
This is the page that the user sees when they first go on the website. A user can see a list of properties when they first go on, but they cannot click on the properties or filter and find properties they are interested in until they login or create an account on this website. This can be done by clicking the login button at the top right, or by clicking anywhere on the page.

Sign up Page



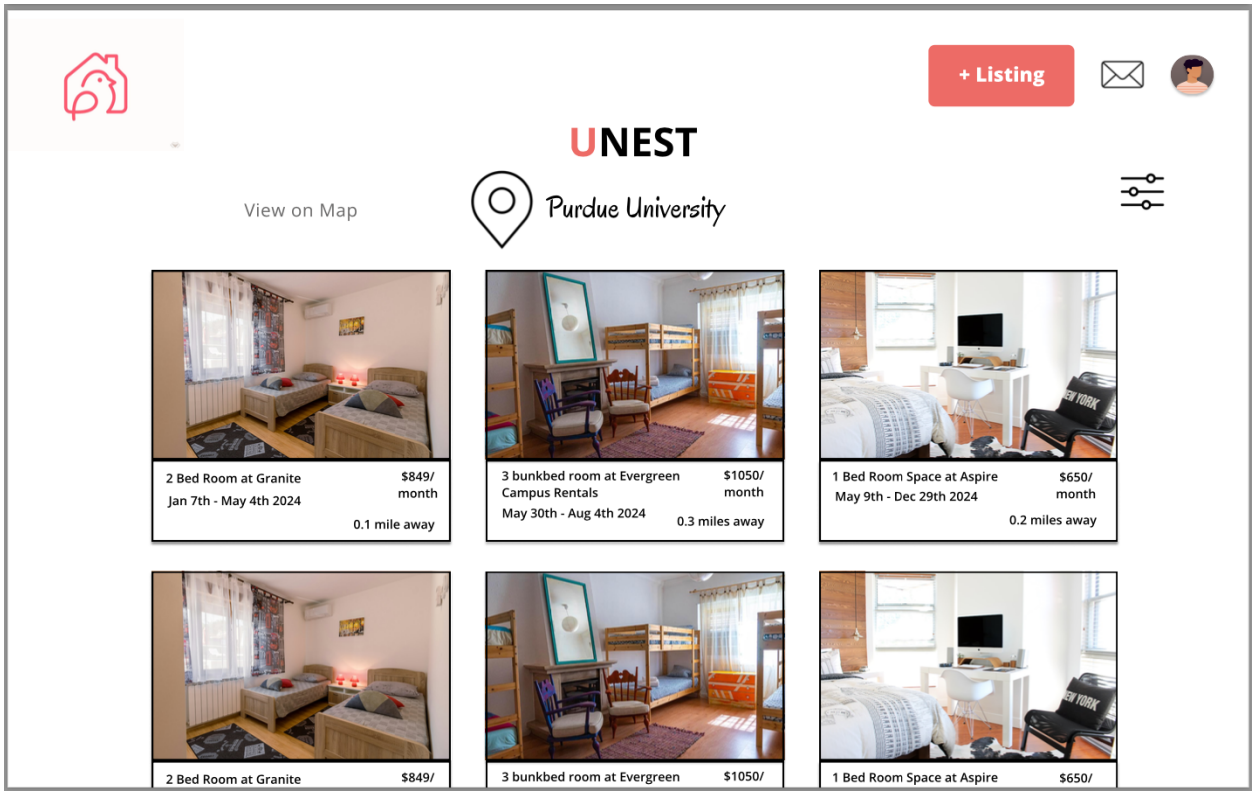
This page is a sign up page that requires the user to input their name, email, university, username, and password. The important feature from this page is the authentication. Users will not be able to click the “Sign Up” button until their email is verified and the green check appears. Once the user finishes filling out the information and has verified their school email, they can click the button at the bottom to officially create their account. The authentication will be done by an external tool so there is no need for a new page.

Login Page



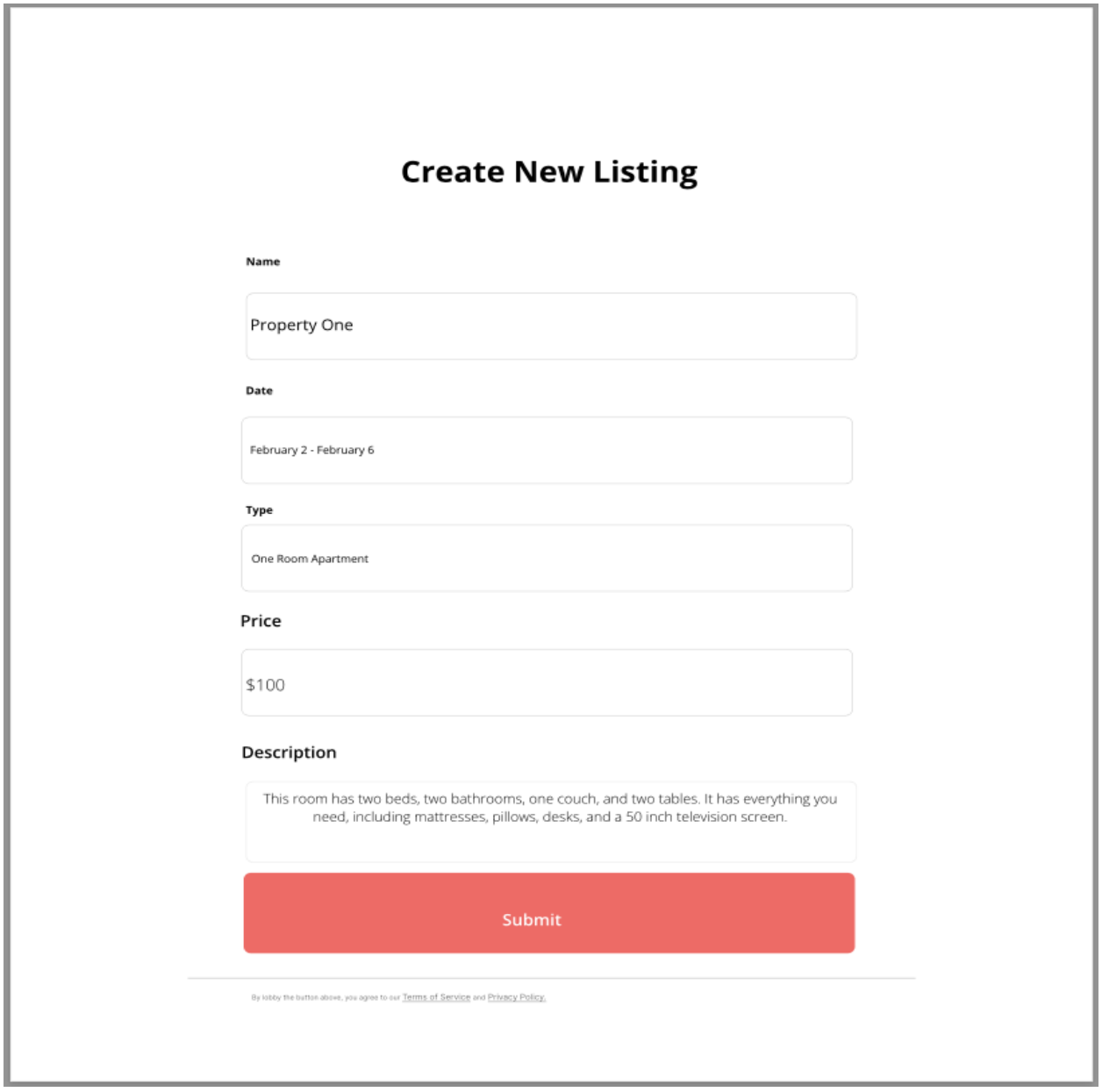
This page is a login page that allows a user who already has an account to input their username or email and password to login. Once that information is entered, the user can click the button at the bottom to officially log in.

Homepage



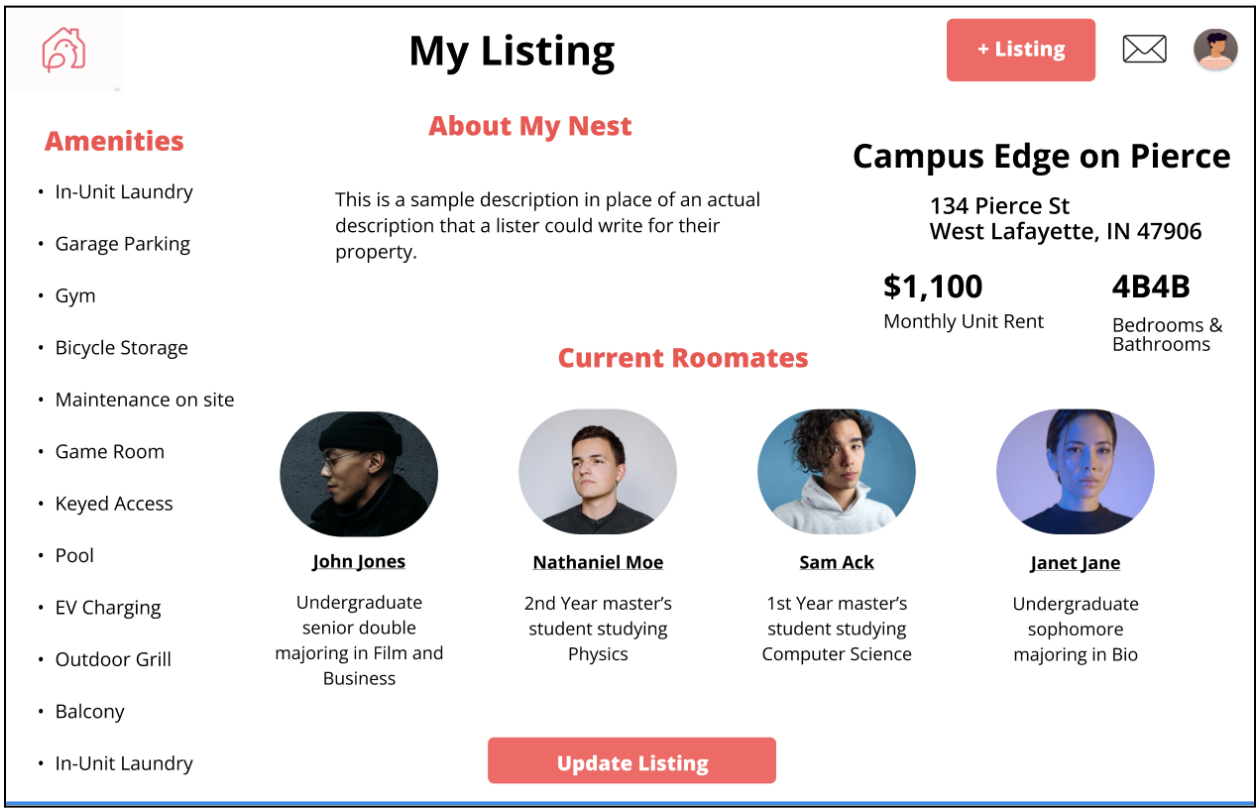
This page is the official homepage of our website when a user is logged in. It contains a list of properties with an image of the property, name of property, dates available, price, and location away from the campus specified at the middle of the page. The user can choose the campus they want to find properties near by clicking on the pin icon and searching. The user can also filter their search on the homepage by clicking on the three lines icon and choosing the price range, date range, number of bedrooms and bathrooms, and mile range accordingly. The user also has the ability to click on the profile icon to look at their profile settings and update their profile, the ability to click on the message icon to send messages to other listers or potential tenants, and also the ability to click on the listings button to add a property to the site and become a lister.

Listing Page



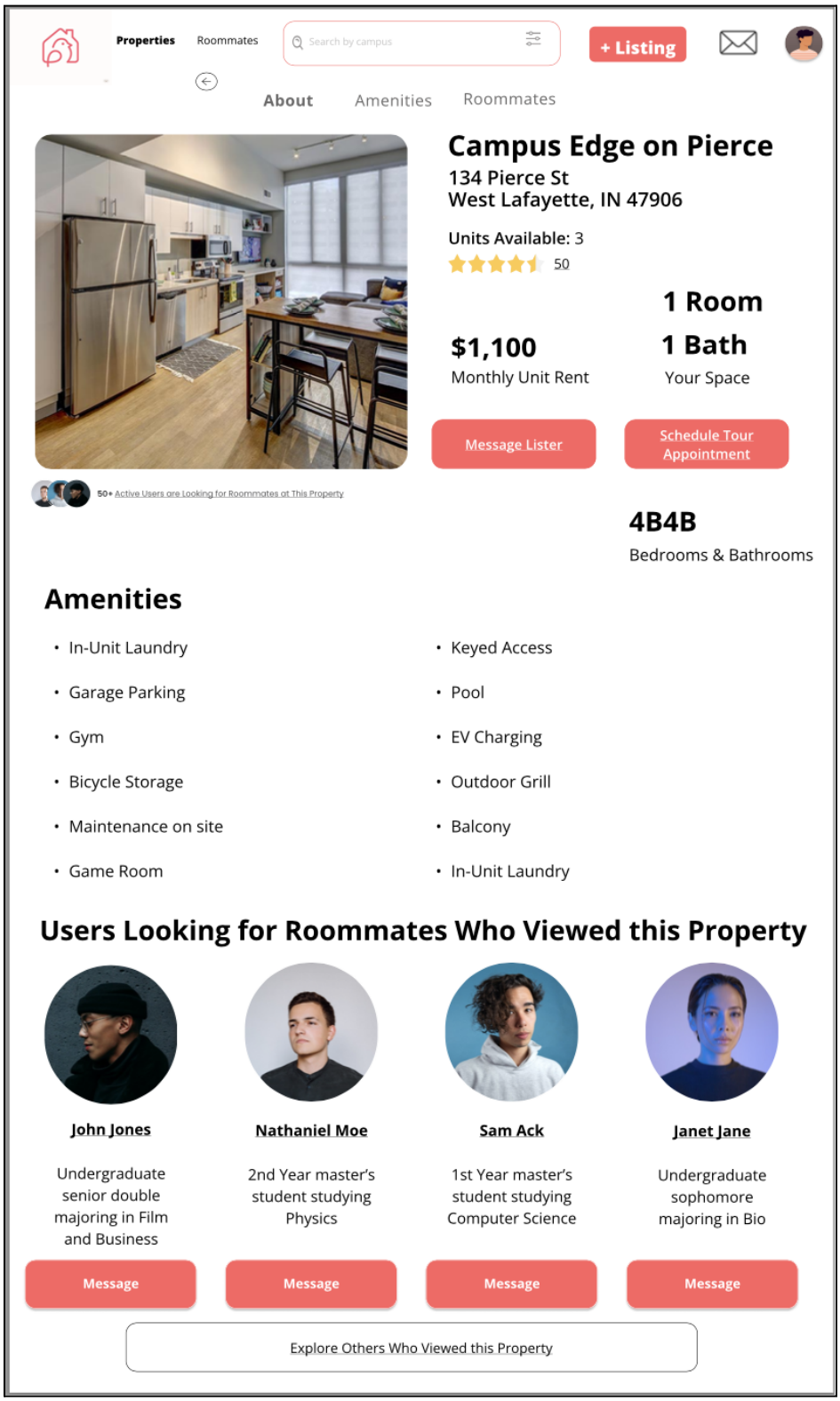
This page allows a user to add their property to the website for potential tenants to show interest in and lease. The user can input the amenities, the name of the property, address of property, dates the property is available for, the price of the property, the number of bedrooms and bathrooms in the property, and a description of the property that they are listing in the blanks provided. Once they input all that information, the user can click the submit button at the bottom of the page to officially post the property on the website.

My Listings Page



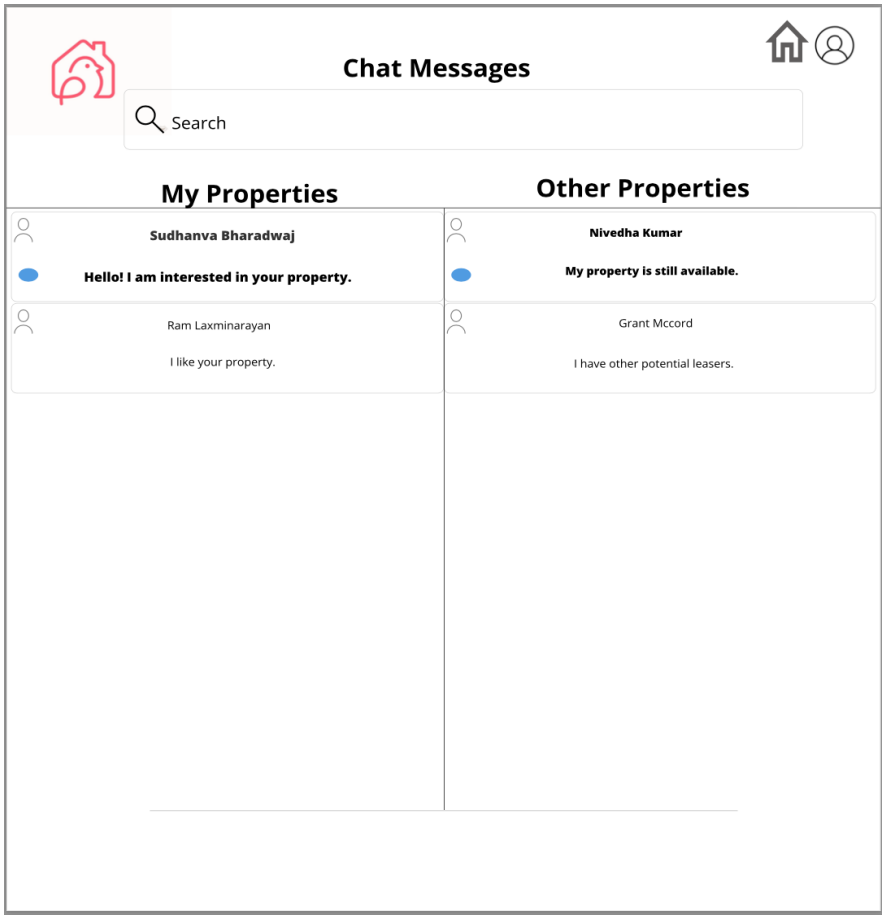
This page allows a user who listed a property to see their property or list of properties that they posted on the website. The user can see all the information that they put about their properties in the my listings page, and they can also see the roommates that they are currently residing with in that property if they have one. The user can click on one of their roommates to see their profile that is viewable to the public. At the bottom of the page, there is an update listings button that allows the user to edit their property information if needed.

Property Listing Page



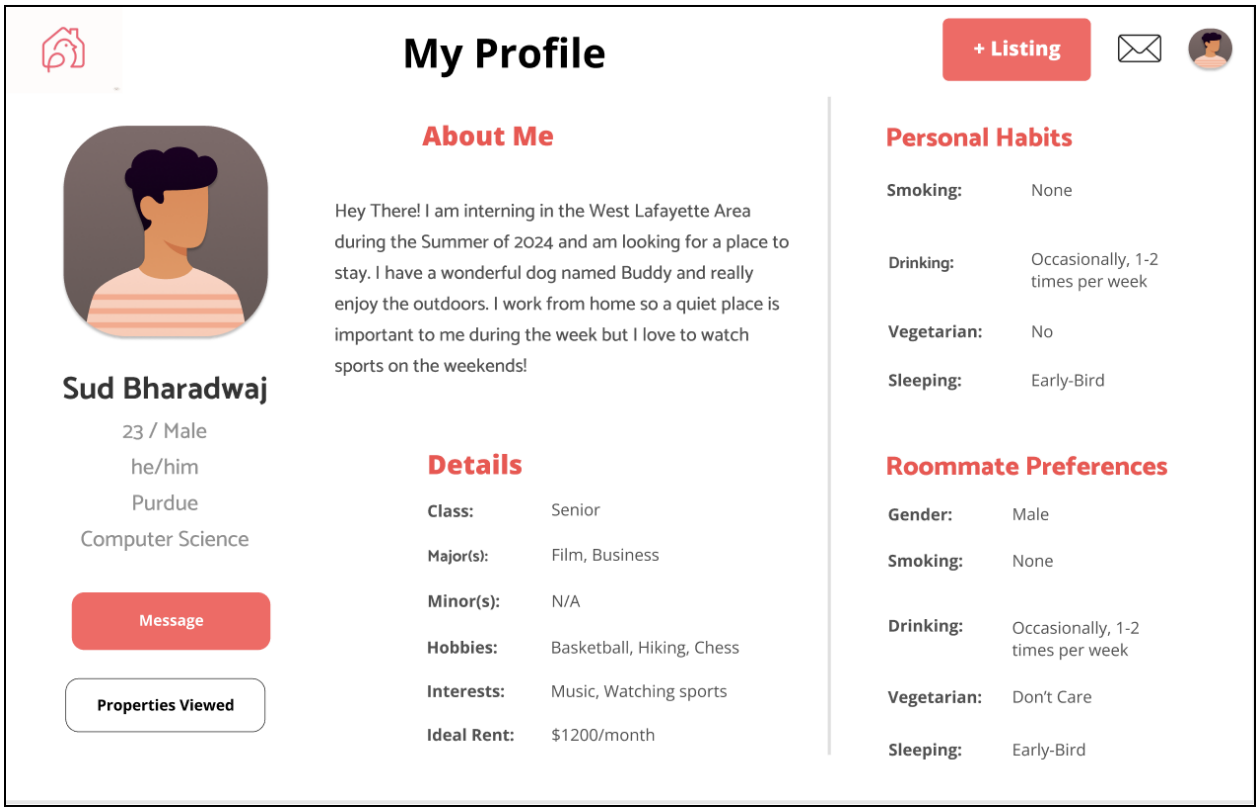
This page allows a user looking for properties to see the full description of a property that is not displayed on the homepage. Details that are provided on this page are amenities, price, location, dates available, units available, and other users interested in the property that are looking for roommates. On the top of this page, there is an option to click on roommates to see current roommates of the property, and there are the other options that are also displayed in the homepage such as an add listing button to add a property to the website, a message icon to send a message to other existing users on the website, and a profile icon to view your own profile. A user can also click on the message lister button to message the person who posted the property, and the schedule tour appointment button to schedule an appointment to see the ins and outs of the property listed. At the bottom of the page, the user can message other users who have viewed the same property and are looking for roommates by clicking on the message button underneath their names, and they can also explore other users by clicking on the Explore Others Who Viewed this Property button all the way at the bottom of the page.

Messages Page



This page allows a user to search for other users that they may want to message by clicking on the search bar. It also allows a user to chat with users that they have previously talked to by clicking on their name. A user can then select their name and send a message to that user.

User Profile Page



This page is created based on the information collected during signup of the user. The design of this page is to provide a brief introduction about an user and their priorities so potential subletters can see if they are a good fit. Additionally, this profile page will be unique for each user and users can visit each other's pages by clicking on a profile from messages. Users will be able to edit their profile and update their bios.