Group 15 Software Engineering

The user of our website will be a UCSD student.

User Stories :

1. **Profile** - As a student, I want to have a profile, so that I can keep track of the places I have seen and type my personal preferences.
2. **Data** - As a student, I want to save my search progress, so I do not need to redo my previous preferences and interests.
3. **Login** - As a student, I want to login via Gmail, so that my profile is secure and won't have to keep track of another password. (UCSD students are provided with a Gmail account immediately upon acceptance.)
4. **Search** - As a student I want to search, so that I can look up potential housing places
5. **Filter** - As a student I want my housing to meet certain criteria I may have, such as location, commute time, type of apartment/house, types of roommates, range of housing price, neighborhood, number of bedrooms and bathrooms, size of apartments in sqft, ranking and reviews of apartments etc.
6. **Details -** As a student, I want to click on a property and see details, so that I can get further information.
7. **Ratings & Reviews -** As a student, I want to see ratings and reviews of a place, so that I can get a sense of previous tenants' experiences.
8. **Tenant Input -** As a student, I want to leave rating and review on the apartment I am living in and input my rent, so that I can share my experience.
9. **Prices** - As a student, I want to see current and previous prices paid by actual tenants, so I can know the real cost, including utilities, deposits, etc.
10. **See Filters Used** - As a student I want to go back to see the other properties I searched so I don’t have to search again with the same filters
11. **Modification** - As a student, I must always have the option to modify my search to fit in a new criteria
12. **Sort** - As a student, I want to sort the property posts by ratings and prices, so that I see the posts most relevant to me first.
13. **Map** - As a student, I want to see the place on a map, so that I can visualize the location, neighborhood, and relative distance to stores, school, etc.
14. **Likes** - As a student I want to be able to save the properties I liked, so I can rejoin the website and look them again

User Stories : Time Estimate and Priority

User stories have been ranked from 1 to 4. 1 being the highest priority, and 4 being the lowest. Rank also represents the sprint that the user story has been assigned.

| ***User Story*** | **Time Estimate** | **Rank and Sprint** |
| --- | --- | --- |
| Database | 40.0 | 1 |
| Details | 14.6 | 2 |
| Filter | 24.3 | 2 |
| Likes | 12.0 | 3 |
| Login | 26.3 | 2 |
| Map | 12.7 | 4 |
| Prices historical | 8.3 | 4 |
| Profile | 32.1 | 3 |
| Rating | 13.5 | 3 |
| Search | 34.2 | 1 |
| Sort | 13.4 | 4 |
| Tenant Input | 16.5 | 4 |
| **Grand Total** | **248.0** |  |

Tasks and Tests :

|  | **Avg**  **Time**  **Estimate** | **Related User Story** | **Task or Test** |
| --- | --- | --- | --- |
| 1 | 5.7 | Login | Login - Create a login page |
| 2 | 1.0 | Login | Test 1: The page should contain a Login and Get Started Button visible in Google Chrome browser |
| 3 | 1.0 | Login | Test 2: The page should contain “Helping Students Find Home Away From Home!” Heading. |
| 4 | 6.0 | Login | Login - Integrate login with only @ucsd.edu mail id |
| 5 | 1.6 | Login | Test 1: a ucsd email tries to login and it enters the website |
| 6 | 1.2 | Login | Test 2: a non-ucsd email tries to login and fails |
| 7 | 1.2 | Login | Test 3: If user Authentication failed, allow the user at max two more re-tries to authenticate, if still fails, then show contact IT page. |
| 8 | 1.9 | Login | Login - Write more tests |
| 9 | 1.7 | Login | Login - Run tests |
| 10 | 5.1 | Login | Login - Debugging |
| 11 | 6.2 | Profile | Profile - Create user profiles with only information to be used when matching to a property. (Age, name, contact information, email, phone number, gender, etc) + there is a button to save edited information. |
| 12 | 1.0 | Profile | Test 1: user can input their information into |
| 13 | 1.0 | Profile | Test 2: user can’t change his email |
| 14 | 1.2 | Profile | Test 3: the information get saved and can be shown after the page get refreshed |
| 15 | 1.7 | Profile | Test 4: The system should perform input field validators so it will only allow valid information to be pushed into the DB |
| 16 | 1.2 | Profile | Test 5: If the information is invalid, some warning/notification is shown to the user “please check your input, value X is invalid”. |
| 17 | 1.2 | Profile | Test 6: if the information is valid, a success message is displayed |
| 18 | 3.5 | Profile | Profile - Account settings. |
| 19 | 1.0 | Profile | Test 1: User can access their profile page by clicking the icon. |
| 20 | 1.2 | Profile | Test 2: Fields of Name, Gender, Age, Phone can be editted and saved. |
| 21 | 4.1 | Profile | Profile - Store for user profile input into database. |
| 22 | 1.3 | Profile | Test 1: The system should perform input field validators so it will only allow valid information to be pushed into the DB. |
| 23 | 1.7 | Profile | Test 2: When the user inputs his profile information/updates it, the database should be updated with his latest information only. |
| 24 | 1.3 | Profile | Profile - Write more tests |
| 25 | 1.1 | Profile | Profile - Run tests |
| 26 | 3.6 | Profile | Profile - Debugging |
| 27 | 6.1 | Database | Database Design - Entity relation model for database |
| 28 | 2.1 | Database | Test1: Put some fake data in an excel spreadsheet and ask these questions to the DB: |
| 29 | 3.0 | Database | Query of active users, can it be retrieved? |
| 30 | 3.0 | Database | Query of particular id of housing, can it be retrieved? |
| 31 | 2.7 | Database | Query of |
| 32 | 6.2 | Database | Database - Implement it on PostgreSQL |
| 33 | 1.9 | Database | Test: Tables are created based on the entity relation model |
| 34 | 5.9 | Database | Database - Make the connection between the database and the frontend |
| 35 | 2.6 | Database | Database - Save information for filters. |
| 36 | 1.5 | Database | Database - Write more tests |
| 37 | 1.3 | Database | Database - Run tests |
| 38 | 3.8 | Database | Database - Debugging |
| 39 | 5.6 | Search | Search and Sort - Design a search bar with filter and sort. |
| 40 | 1.4 | Search | Test 1: The search bar section contains a search button, a filter section with a number of filters and an apply button, and a sort button. |
| 41 | 2.9 | Search | Search - define the query to send to the database with the selected filters |
| 42 | 1.6 | Search | Test 1: The query sent to the database matches the content in search and filter. |
| 43 | 2.5 | Search | Search - send the query to the database |
| 44 | 1.4 | Search | Test 1: The database does have the query being sent. |
| 45 | 3.6 | Search | Search - return the query results to the frontend. Do a single inquiry for search results. 10 per page. Get all 10 property information. |
| 46 | 1.7 | Search | Test 1: All of the query results and information are sent to the frontend correctly. |
| 47 | 1.1 | Search | Test 2: All of the results are displayed with a button to show detailed information and each page has 10 apartments. |
| 48 | 3.7 | Search | Search - Display the searched results on the frontend 10 at a time as lists. |
| 49 | 0.9 | Search | Test1: click the search button and the result shows in the webpage. |
| 50 | 0.9 | Search | Test2: the information is displayed clearly with no overlapping |
| 51 | 1.3 | Search | Search - Write more tests |
| 52 | 1.1 | Search | Search - Run tests |
| 53 | 4.6 | Search | Search - Debugging |
| 54 | 5.4 | Filter | Filter - Apply preferences in searching (# of bedrooms, # of bathrooms, price, commute time to UCSD, size of apartment [sqft] and rating) |
| 55 | 1.3 | Filter | Test: saving |
| 56 | 2.8 | Filter | Filter - Implement button to clear the current filter |
| 57 | 1.7 | Filter | Test: when having a set of filter active and clicking the “clear” button, all the filters disappear and the webpage looks like the homepage |
| 58 | 3.4 | Filter | Filter - Add values. Tentative list : Location, commute time, type of apartment/house, types of roommates, range of housing price, neighborhood, number of bedrooms and bathrooms, size of apartments in sqft, ranking and reviews of apartments |
| 59 | 1.7 | Filter | Test 1 : Validate filter input |
| 60 | 2.9 | Filter | Filter - Design how to display filtered values |
| 61 | 1.3 | Filter | Filter - Write more tests |
| 62 | 1.3 | Filter | Filter - Run tests |
| 63 | 2.6 | Filter | Filter - Debugging |
| 64 | 6.1 | Map | Map - Integrate google map |
| 65 | 1.6 | Map | Test: when clicking in the button a new window appears with the google map website displaying the location of the property and its address |
| 66 | 1.6 | Map | Map - Write more tests |
| 67 | 1.3 | Map | Map - Run tests |
| 68 | 2.2 | Map | Map - Debugging |
| 69 | 6.6 | Details | Details - Display picture, blurb, website link, email, phone number, filter values, prices, past prices, ratings, reviews. |
| 70 | 1.9 | Details | Test: After clicking on one apartments, you can see in a new webpage the picture, website link of the owner, email of the owner, phone number of the owner, price of the property, past prices in a table, average rating and reviews. |
| 71 | 1.5 | Details | Details - Write more tests |
| 72 | 1.3 | Details | Details - Run tests |
| 73 | 3.3 | Details | Details - Debugging |
| 74 | 5.3 | Rating | Rating & Reviews - display average rating for the clicked property, and display all the reviews and each review’s associated rating for the property. |
| 75 | 1.2 | Rating | Test1: average rating is displayed and computed correctly of the property. |
| 76 | 1.2 | Rating | Test2: All reviews are displayed in the correct format, and each review has its associated rating. |
| 77 | 1.3 | Rating | Rating & Reviews - Write more tests |
| 78 | 1.1 | Rating | Rating & Reviews - Run tests |
| 79 | 3.5 | Rating | Rating & Reviews - Debugging |
| 80 | 3.5 | Prices historical | Prices - display a table that shows the current and previous rating of the apartment. |
| 81 | 1.2 | Prices historical | Test: current and previous ratings are displayed in the correct format. |
| 82 | 1.0 | Prices historical | Prices - Write more tests |
| 83 | 0.9 | Prices historical | Prices - Run tests |
| 84 | 1.8 | Prices historical | Prices - Debugging |
| 85 | 4.1 | Tenant Input | Tenant Input Prices, & Reviews- The student inputs a review, then to assembly the query to send to the DB |
| 86 | 3.0 | Tenant Input | Tenant input & Reviews- Send the input review + rating to the database from the assembled query |
| 87 | 2.8 | Tenant Input | Tenant Input - Update the database with the review and the rating sent |
| 88 | 1.4 | Tenant Input | Test1: updates can be seen in the tables |
| 89 | 1.4 | Tenant Input | Tenant Input - Write more tests |
| 90 | 1.3 | Tenant Input | Tenant Input - Run tests |
| 91 | 2.6 | Tenant Input | Tenant Input - Debugging |
| 92 | 3.2 | Sort | Sort - sort the displayed apartments based on the average rating. |
| 93 | 1.2 | Sort | Test1 : apartments displayed are in descending order based on the average rating. |
| 94 | 3.2 | Sort | Sort - sort the displayed apartments based on the price(high to low/low to high). |
| 95 | 1.2 | Sort | Test1 : apartments displayed are in ascending order or descending order based on the price. |
| 96 | 1.2 | Sort | Sort - Write more tests |
| 97 | 1.1 | Sort | Sort - Run tests |
| 98 | 2.4 | Sort | Sort - Debugging |
| 99 | 4.5 | Likes | Likes - Create a star (favorite) list for users. |
| 100 | 1.2 | Likes | Test 1: webpage of star list is created and can be accessed by users. |
| 101 | 3.2 | Likes | Likes - User can star apartments that they like and find them in their own star list. |
| 102 | 1.2 | Likes | Test 1: User can see the apartments page that they star in the list. |
| 103 | 2.0 | Likes | Likes - Write more tests |

Design Mockup :





