**Software Requirements and Design Document**

**For**

**Group <18>**

Version 1.0

**Authors**:

Alex Bundy

Kester Mbah

Johnathan Gutierrez-Diaz

Christopher Laughlin

Rodjina Pierre Louis

# Overview (5 points)

Our Project College Lodging is a web application aimed to provide students a platform to discover and identify housing options near their desired college that are suitable to their needs. The system should allow users to search for apartments nearby, filter through them with various aspects such as if it has a pool or not, compare the distances between options to campus, and be able to see all of these apartments on a map provided by the Google Maps API.

This application’s intended use is for students to find a proper apartment near campus. This will include a sign in and login for students to be able to save the apartments they’re interested in for future pursuit. The application will also allow for new apartments to add their information to the site, allowing the apartments to be up to date as new ones are built in the area. Essentially the site will provide an easy way to search through apartments, a way to save these apartments for later, and a way for apartment owner’s to add their specific options to the site.

# Functional Requirements (10 points)

1. Apartment Search: The ability to search through apartments within the application apartment search page with the mouse and scroll wheel. (Priority: High)
2. View Apartment Listings: The ability to search through apartments with given criteria such as: maximum distance to campus, rent, number of bedrooms, etc. given the filters(Priority: High)
3. User Registration: The ability to sign up for an account given basic information like email and password. This would be through the sign in page. (Priority: High)
4. User Login: The ability for the system to take a registered user’s login information and sign them in to their account through the sign in page, bringing them to their account. (Priority: High)
5. Save Apartment Listings: The ability for a user to select an apartment they like and save it in a list for them to refer to later. This would be done by selecting an apartment in the apartment search and marking it with the mouse. This is somewhat the point of the project, being able to create a list of suitable apartments that users may later refer to find their apartment of choice. (Priority: High)
6. Add Apartment: The ability for an apartment owner to get on the site with an account and create an apartment listing through the addapartment page by inputting necessary info and confirming its addition, this is to prepare for future apartments that may be built after the application is in use. (Priority: High)
7. Travel Between Pages: The ability to go from one page of this application to the next through the header, without this actual use of the project will be extremely limited to what screens are able to be accessed (Priority: High)
8. Map functionality: The ability to see a selected apartment on the provided map when selected in the list. (Priority: Medium)
9. Distance Calculation: The ability to calculate and display the distance from a chosen apartment to campus. This is one of various factors a user is meant to be able to use to compare between other apartments to determine what they like once an apartment is selected in the list. (Priority: Medium)
10. Password Recovery: The ability to recover a password through a user’s email once requested. Meant for account security, being able to obtain lost accounts is good practice. (Priority: Medium)
11. Logout Functionality: The ability to logout of a account once signed in for security reasons. (Priority: Medium)
12. Admin Control: The creation of an admin control panel where they will be able to edit information on the site, add, and remove apartments.
13. User Profile Management: The ability for users to change their account information like email or password in case they were compromised. (Priority: Low)
14. User Feedback: The ability for users to submit feedback to the owner’s of the site. (Priority: Low)

# Non-functional Requirements (10 points)

Performance: The application should not take longer given more apartments being added throughout its lifetime, it should be able to adapt to whatever amount is provided and still load in a realistic amount of time.

Reliability: The apartments within the application should remain throughout the lifetime of the program and there should be no downtime for the apartment search.

Scalability: More apartments should be able to be added without inhibiting the application with load times to identify available apartments.

Security: User passwords should be secured properly, with the ability to change passwords and account information being a necessary addition in order to ensure this security. (A secure system is more valuable than a non secure system)

Data Integrity: The database that stores both login information as well as apartment listings should not be able to be lost or deleted without admin approval

Maintainability: There should be no reason why this application can not be changed in order to work for a different school or modified to provide additional information, should be easy to follow and edit.

# Use Case Diagram (10 points)A diagram of a diagram Description automatically generated

# 

|  |  |
| --- | --- |
| Use Case | Textual Descriptions |
| Browse Housing Options | * **Description:** Allows users to view the general list of housing options available on the platform * **Participating Actors:** Unauthenticated User, Authenticated User, Admin. * **Preconditions:** The housing listings must be available in the system. * **Postconditions:** The user views a list of available housing options. * **Special Requirements:** The system should load and display housing options efficiently to ensure a good user experience. |
|
|
|
|
| View Listing Details | * **Description**: Enables users to view detailed information about a specific housing listing. * **Participating Actors**: Unauthenticated User, Authenticated User, Admin. * **Preconditions**: The user has selected a specific listing to view, and the listing exists in the system. * **Postconditions**: Detailed information about the listing is displayed to the user. * **Special Requirements**: Information should be clear and include images, descriptions, and pricing. |
|
|
|
|
|
|
|
| Login | * **Description**: Allows users to authenticate to gain access to additional features. * **Participating Actors**: Unauthenticated User, Authenticated User * **Preconditions**: The user has an existing account in the system. * **Postconditions**: The user is authenticated and can access features for authenticated users. * **Special Requirements**: Secure authentication mechanism to protect user credentials. |
| Save Listing | * **Description:** Enables authenticated users to save specific housing listings for future reference. * **Participating Actors:** Authenticated User * **Preconditions:** The user is authenticated and has viewed a listing they wish to save. * **Postconditions:** The listing is saved under the user’s profile for later viewing. * **Special Requirements:** Persistent storage for saved listings to be accessible in future sessions. |
| Filter Listings | * **Description**: Allows users to apply filters to narrow down the list of housing options based on specific criteria (e.g., price, bedroom). * **Participating Actors**: Authenticated User, Admin. * **Preconditions**: The user has accessed the housing options and wants to refine their search. * **Postconditions**: The system displays a filtered list of housing options that match the user’s criteria. * **Special Requirements**: The filter feature should support multiple criteria and perform efficiently to avoid long wait times. |
| Add & Modify Listings | * **Description:** Allows admins to add new housing listings or edit existing ones * **Participating Actors:** Admin. * **Preconditions:** The admin is authenticated and has access to the listing management feature. * **Postconditions:** New listings are added or existing ones are updated * **Special Requirements:** Admin interface should include input validation to ensure all required listing information is provided and formatted correctly. |
| Modify Users | * **Description:** Allows admins modify details for existing users. * **Participating Actors:** Admin. * **Preconditions:** The admin is authenticated and has permissions to manage users. * **Postconditions:** Existing accounts are updated. * **Special Requirements:** The system should enforce unique email addresses to prevent duplicate accounts. |
| Remove Users | * **Description:** Enables admins to delete user accounts from the system, typically for moderation or maintenance. * **Participating Actors:** Admin. * **Preconditions:** The admin is authenticated and has permissions to remove users. * **Postconditions:** The selected user account is deleted and no longer has access to the system. * **Special Requirements:** System should prompt for confirmation before deletion to avoid accidental removals, and possibly log the deletion action. |
| Admin Dashboard | * **Description:** Provides the admin with an overview of system activities, including user and listing statistics, for effective management. * **Participating Actors:** Admin. * **Preconditions:** The admin is authenticated. * **Postconditions:** The admin accesses tools and analytics to manage the platform effectively. * **Special Requirements:** Dashboard should be intuitive and display key information |

# Class Diagram and/or Sequence Diagrams (15 points)

A screenshot of a computer

Description automatically generated

A diagram of a website

Description automatically generated

# Operating Environment (5 points)

Hardware Platform: Needs to run on a server with sufficient processing power and memory to handle what the application entails, most likely relatively low requirements considering the scope of the project.

Operating System: If we were to deploy this application Ubuntu would most likely be used due to our familiarity with Ubuntu from previous projects.

Web Server: Some type of sever necessary that will be able to run all portions of this project like the Flask and SQL commands.

Database: MySQL Server necessary for storing the user data as well as apartment listings.

# Assumptions and Dependencies (5 points)

# Assumption of User Base: It is assumed that an active user base will be active around Florida State University or a college in general for this application to gain traction. Essentially an audience is assumed for the application.

# Dependence on Google Maps API: For both the map and distance calculations the project is dependent on Google Maps API, if this application were to stop functioning or we were to be barred from using it a lot of the functionality of our program would no longer work.

# Scalability: it is assumed that the server and hardware provided will be able to account for a constantly growing number of apartments, accounts, and lists of apartments within these accounts. If not some type of limit would need to be put into place to ensure the project maintains functionality.

# Database: MySQL Server necessary for storing the user data as well as apartment listings. It will have to be regularly maintained and backed up to prevent data loss. If this database were to corrupt, the project would no longer be useful as no apartment or saved lists of apartments on accounts would be able to be used.

# No Fake Listings: it is assumed that listings made by users will be genuine. There is an ability for the admin to remove entries if they are deemed not real however there is no verification stage where it is checked before going live, leaving it vulnerable to fake listings.

# Device of choice: It is assumed most if not all users will be going through a desktop or laptop rather than a mobile device. Essentially the main device for this application to run has been determined to be the most sensible choice.