**Software Requirements and Design Document**

**For**

**Group <18>**

Version 1.0

**Authors**:

Alex Bundy

Kester Mbah

Johnathan Gutierrez-Diaz

Christopher Laughlin

Rodjna 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 Florida State University 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. (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. (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 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, 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. 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. (Priority: Medium)
9. 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)
10. Logout Functionality: The ability to logout of a account once signed in for security reasons. (Priority: Medium)
11. User Profile Management: The ability for users to change their account information like email or password in case they were compromised. (Priority: Low)
12. User Feedback: The ability for users to submit feedback to the owner’s of the site. (Priority: Low)

*List the* ***functional requirements*** *in sentences identified by numbers and for each requirement state if it is of high, medium, or low priority. Each functional requirement is something that the system shall do. Include all the details required such that there can be no misinterpretations of the requirements when read. Be very specific about what the system needs to do (not how, just what). You may provide a brief design rationale for any requirement which you feel requires explanation for how and/or why the requirement was derived.*

# 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)

# Use Case Diagram (10 points)

A diagram of a diagram

Description automatically generated

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

*This section presents a high-level overview of the anticipated system architecture using a* ***class******diagram*** *and/or* ***sequence diagrams****.*

*If the main* ***paradigm*** *used in your project is* ***Object Oriented*** *(i.e., you have classes or something that acts similar to classes in your system), then draw the* ***Class Diagram******of the entire system and Sequence Diagrams for the three (3) most important use cases in your system.***

*If the main* ***paradigm*** *in your system is* ***not Object Oriented*** *(i.e., you* ***do not*** *have classes**or anything similar to classes in your system) then only draw* ***Sequence Diagrams****,* ***but for all the use cases of your system.*** *In this case, we will use a modified version of Sequence Diagrams, where instead of objects, the lifelines will represent the functions in the system involved in the action sequence.*

***Class Diagrams*** *show the* ***fundamental objects/classes*** *that must be modeled with the system to satisfy its requirements and* ***the relationships*** *between them. Each class rectangle on the diagram* ***must also include the attributes and the methods of the class*** *(they can be refined between increments). All the* ***relationships between classes and their multiplicity*** *must be shown on the class diagram.*

*A* ***Sequence Diagram*** *simply depicts* ***interaction******between objects*** *(or* ***functions -*** *in our case - for non-OOP systems) in a sequential order, i.e. the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.*

# 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.

*Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.*

# Assumptions and Dependencies (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.

*List any assumed factors (as opposed to known facts) that could affect the requirements stated in this document. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project.*