Software Requirements and Design Document

For

Group <10>

Version 3.0

Authors:
Iskandar V
Panayoti K
Dylan J
Chris H
Blaine T

1. Overview (5 points)
Give a general overview of the system in 1-2 paragraphs (similar to the one in the project proposal).

This project is a fully functional/responsive website that has multifunctionality. Any FSU student can make a request on this website to look for any handyman services they might need in their college life. Once a request and price for the service are posted, another FSU student can fulfill the service by doing the action required.

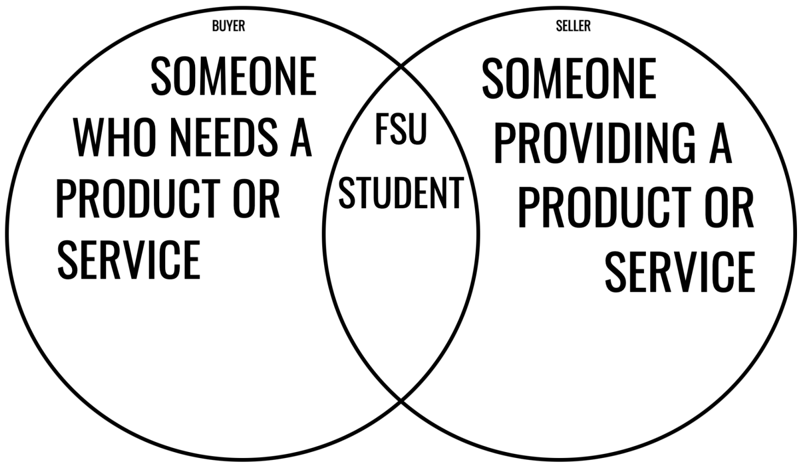
2. Functional Requirements (10 points)
*List the **functional requirements** in sentences identified by numbers and for each requirement state if it is of high, medium, or low priority. Each functional requirement is something that the system shall do. Include all the details required such that there 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 that you feel requires an explanation for how and/or why the requirement was derived.*

- Making listing for services
- Creating an account
- Logging in
- Logging out
- Checking Profiles
- Fulfilling request
- Posting Comments
- Responding to Comments
- providing fast loading speed

3. Non-functional Requirements (10 points)
*List the **non-functional requirements** of the system (any requirement referring to a property of the system, such as security, safety, software quality, performance, reliability, etc.) You may provide a brief rationale for any requirement which you feel requires explanation as to how and/or why the requirement was derived.*

- High security to protect user data
- High-performance website
- Requests such as comments and posts will go through
- Maintaining database reliability
- Server integrity

4. Use Case Diagram (10 points)
*This section presents the **use case diagram** and the **textual descriptions** of the use cases for the system under development. The use case diagram should contain all the use cases and relationships between them needed to describe the functionality to be developed. If you discover new use cases between two increments, update the diagram for your future increments.*


BUYER
SOMEONE WHO NEEDS A PRODUCT OR SERVICE
FSU STUDENT
SELLER
SOMEONE PROVIDING A PRODUCT OR SERVICE

5. Class Diagram and/or Sequence Diagrams (15 points)

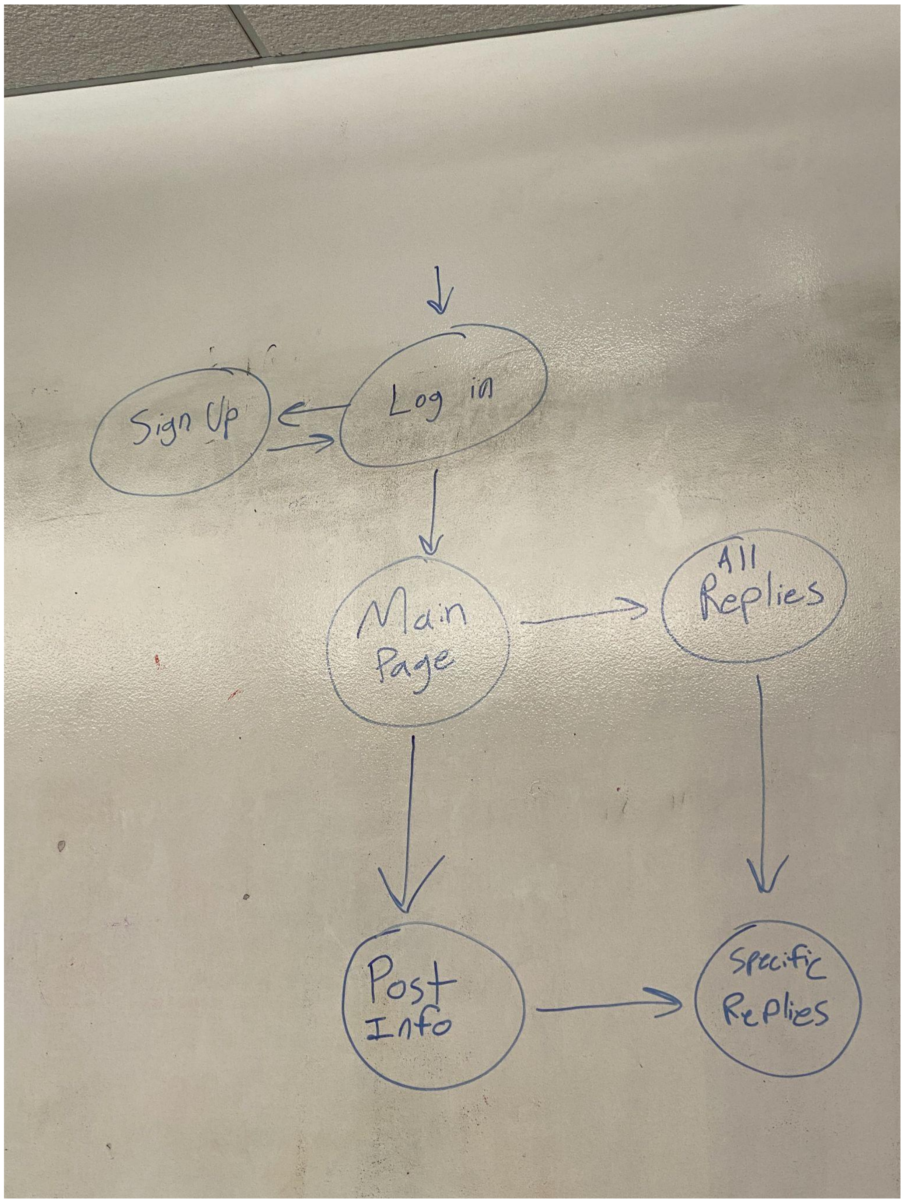
*This section presents a high-level overview of the anticipated system architecture using a **class diagram** and/or **sequence diagrams**.*

*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 **<u>not</u> Object Oriented** (i.e., you **<u>do not</u>** have classes or anything similar to classes in your system) then only draw **Sequence Diagrams**, **but for <u>all</u> 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 <u>functions</u> 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.*



6. Operating Environment (5 points)

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.

The web application will be developed on VS Code which allows us to configure the development environment settings to allow cross-OS development. The website itself will be accessible from any operating environment that supports internet browsing as the website will be hosted through GitHub's server domain service.

7. Assumptions and Dependencies (5 points)

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.

We will reuse multiple components for the website. FSU students will use the website. There is a high demand for this service.