Software Requirements and Design Document

For

Group <7>

Version 1.0

Authors:

Jacqueline Vermette

Meghan Cox

Madelyn Yarber

1. Overview (5 points)

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

Currently our system runs on an API localhost. You are able to use the login and register pages which restricts you otherwise from entering the webpage. After you've logged in, you are then taken to the dashboard. Here you will see the main page with daily, weekly and monthly habits. On top of that, you are able to see a rotation of motivational quotes and an interactive notes page. You can access the task manager and select daily, weekly or monthly habits. What you will see here is what you have added from the edit habits tab. If you go to the edit habits tab you can create, edit, or delete a habit. The data page is to store all your habits and the calendar simply states what the date is.

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 which you feel requires explanation for how and/or why the requirement was derived.*

1) Database fixing: high
2) Allow user to delete account : high
3) Saving the data of editing , deleting, and creating habits: high
4) Display the created daily, weekly, monthly habits in the habits tab : high
5) Rotate motivational quotes: Medium
6) Allow user to add notes: medium
7) Getting the user editing sections to work and save the data : high

3. Non-functional Requirements (10 points)

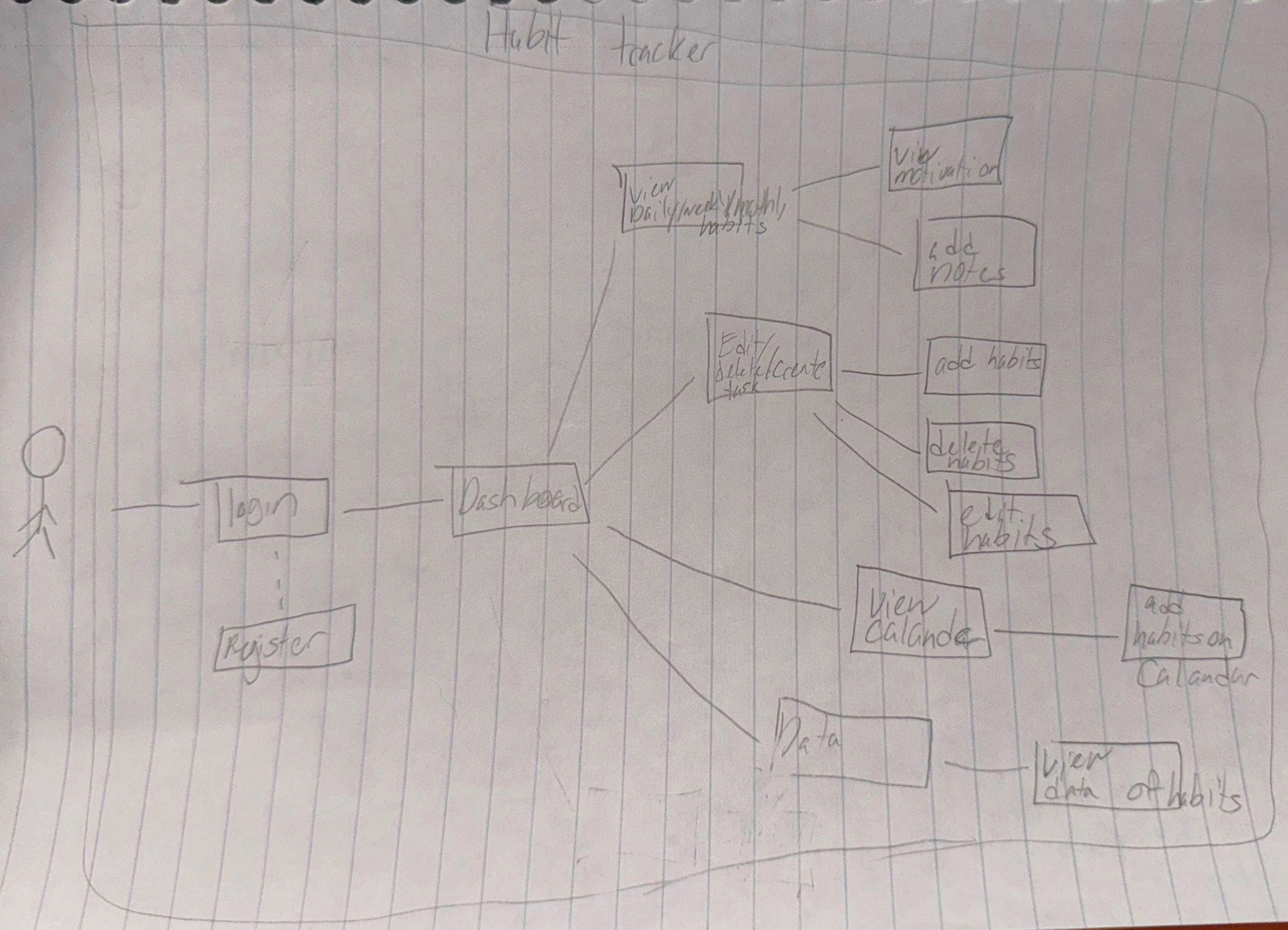
*List the **non-functional requirements** of the system (any requirement referring to a property of the system, such as security, safety, software quality, performance, reliability, etc.) You may provide a brief rationale for any requirement which you feel requires explanation as to how and/or why the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

1) Creating functionality of user data to login page and register page: High
2) Making login page first thing you see : medium
3) Restricting users from accessing web page until logged in: high
4) Remote database access: High
5) Pull data appropriately: High

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.*

***Textual descriptions of use cases**: For the first increment, the textual descriptions for the use cases are not required. However, the textual descriptions for all use cases discovered for your system are required for the second and third iterations.*


Habit tracker
login
Register
Dashboard
View daily/weekly/monthly habits
View motivation
add notes
Edit/delete/create task
add habits
delete habits
edit habits
View Calander
add habits on Calandar
Data
View data of habits

login - User is prompted to login or else they are restricted.
Register - If user does not have login they can register.
Dashboard - once logged in you can access dashboard and task manager

View daily/weekly/monthly: View what habits you are tracking
Motivational: Cycle through motivational quotes
Add notes: Add notes you'd like to remember
Edit/delete/add habits: This allows you to add, delete or edit a task/habit.
View Calander: View calander. Hoping to add functionality to add existing habits to it
Data: View your data of habits

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.***

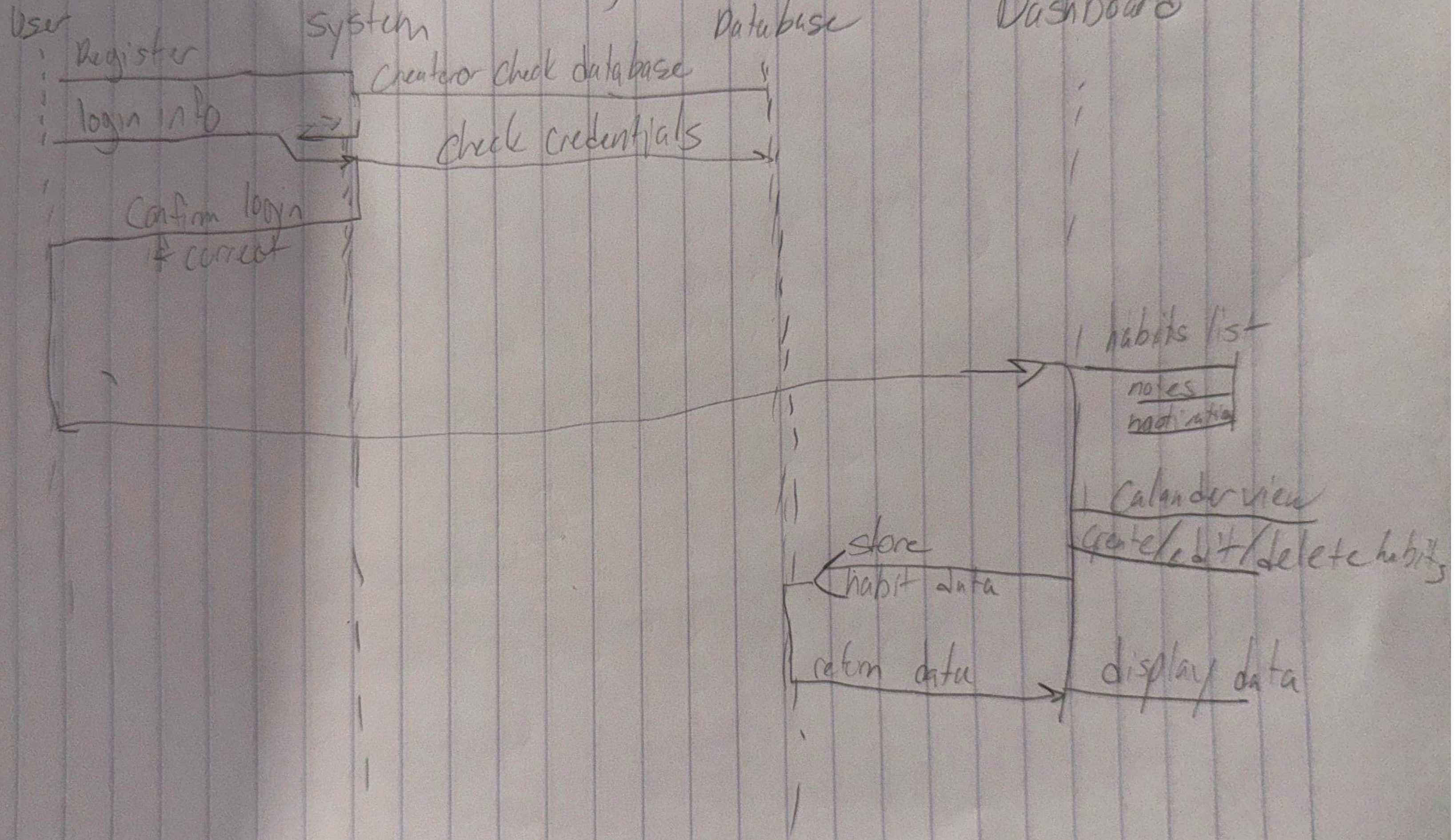
o

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.*

Habit tracker



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.

Currently our webpage is run on Windows 11 version 24h2. We are using an API database.

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.

Some assumed factors could be that the login isn't redirecting properly when you first open the application. Another factor could be that the database is not storing user data properly.

Dependencies would be the loading screen with the small dots.