SW Engineering CSC 648-04

**CSC 648-848 Milestone 2: a) Architecture, UI mock-ups & GUI design and b) vertical SW prototype**

“Track Your Money (TYM)”

Team 6

10 / 17 / 2022

Team members:

Robert Sato (Team Lead),

Siham Argaw (Scrum Master),

Hemanta Thapa (Front-end Lead),

Neel Manthani (Back-end Lead),

Diego Islas (Github Master)

1. Data Definitions V2

The main data structures or collections are going to be users, expenses, and income. Each user will have a username, password, email, and id. Transactions will be a base interface class. It will keep track of data such as date, amount. Then we will have a subclass with expenses and incomes as transactions. They will have separate data fields that are exclusive to their own type. Expenses will have data fields of category of expense, description and everything inherited from transactions. Income will have the data fields of type of method of deposit and everything inherited from the transaction class. We will be using a nosql database so there are no relations. Each transaction will be its own document. And each user will be its own document. Transactions can be an expense or an income. We can also have data fields that would belong to the users collection that track last time paid for subscription and when the subscription payment is coming up.

Users will have the privilege to see user specific information based on the login information they provide. They will only be able to access the information when logged in. They will be able to see that accounts expense and income data fields

The income and expenses data fields will be only user specific. Each expense or income (transaction) will have an id that corresponds to a userid. So only the user that owns those transactions can see their income/expense related data.

Users—------> all attributes can’t be null

. username

. email

. password

. id

Income—-------> all attributes can’t be null

. id

. date

. typeOfPayment

. amount

Expense—-----------> all attributes except image and description can’t be null

. id

. date

. category

. description

. amount

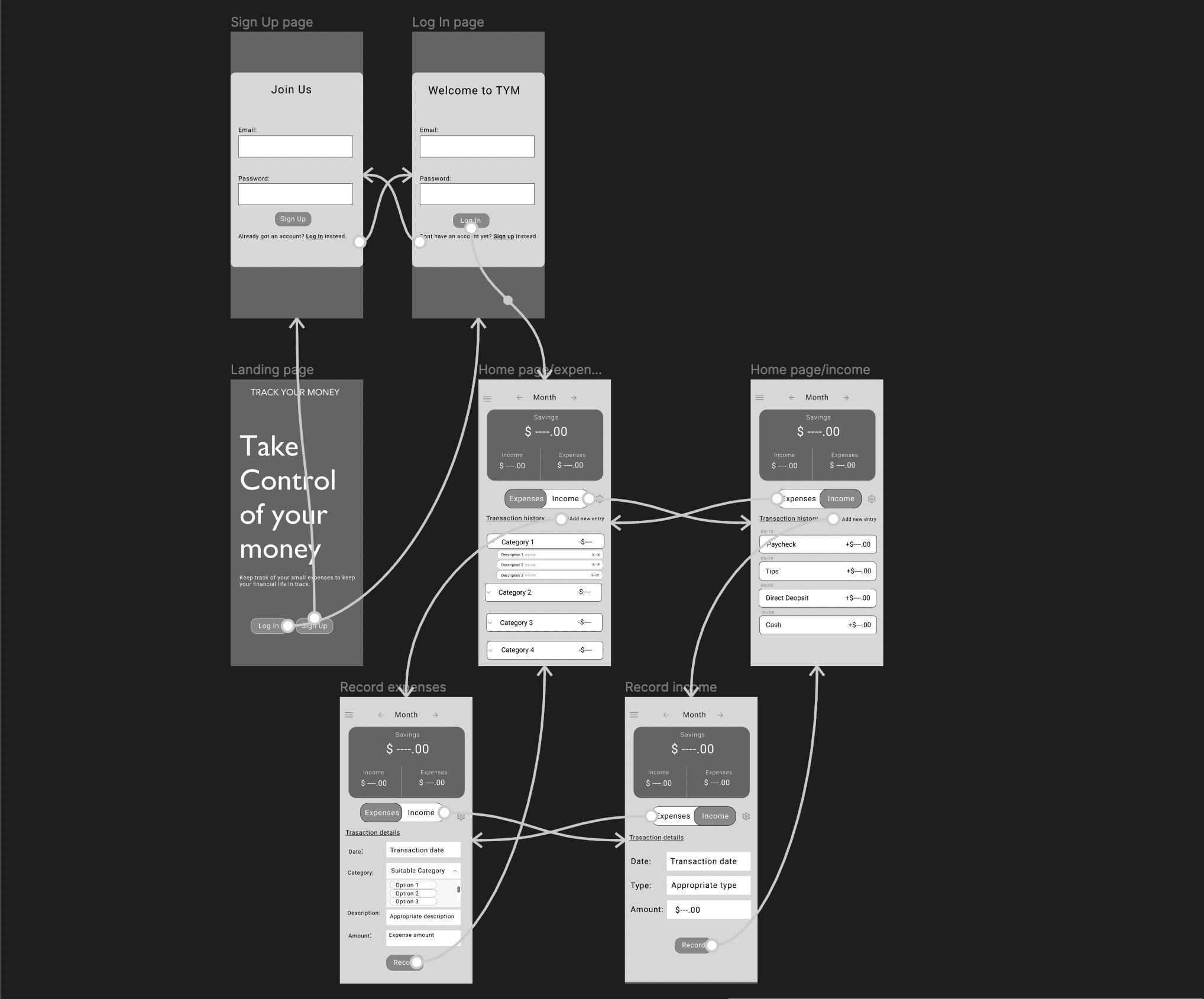
. image—-> of type enumeration Image which allows(png, bmp, svg, jpg) types

# 2. Functional Requirements V2

Functional Requirements:

1. Priority 1
2. Users can add spendings manually
3. Users can add their income manually
4. App keeps a record of spendings and income
5. App calculates savings per month and year and displays it
6. App categorizes spending by type, date spent, amount spent, and description, and these categories can be edited by the user.
7. App categorizes income by paycheck, tip, direct deposit, and cash and these categories can be edited by the user.
8. Users can delete an expense or income they added
9. Users have to login/ sign up to access app
10. Priority 2
11. Allows users to input a certain budget that they want to set for themselves per month and displays a message when they pass that limit. (Displays an alert when savings=0$)
12. App displays graph of income and expenses throughout a month and a year
13. Priority 3
14. App lets users add images of receipts to their spending, and it can be used for reference.

3. UI Mockups and Storyboards (high level only)



4. High level Architecture, Database Organization

High-level NoSQL Database Organization

1. User (id:string, username:string, email:string, password:string)
2. Income(id:string, date:DateTime, typeOfPayment:string, amount:int)
3. Expense(id:string, date:DateTime, category:string, description:string, amount:int, image:Image)

Add/Delete/Search architecture:

1. Users can create and delete an account
2. Users can add 0 to many expenses and incomes
3. Users can delete incomes and expenses
4. Users can search for an expense or income

Users{

id: String;

username: String;

email: String;

password: String;

}

Income{

id: String;

date: DateTime;

typeOfPayment: String;

amount:int;

}

Expense{

id: String;

date: DateTime;

category: String;

description:string;

amount:int;

image: Image;

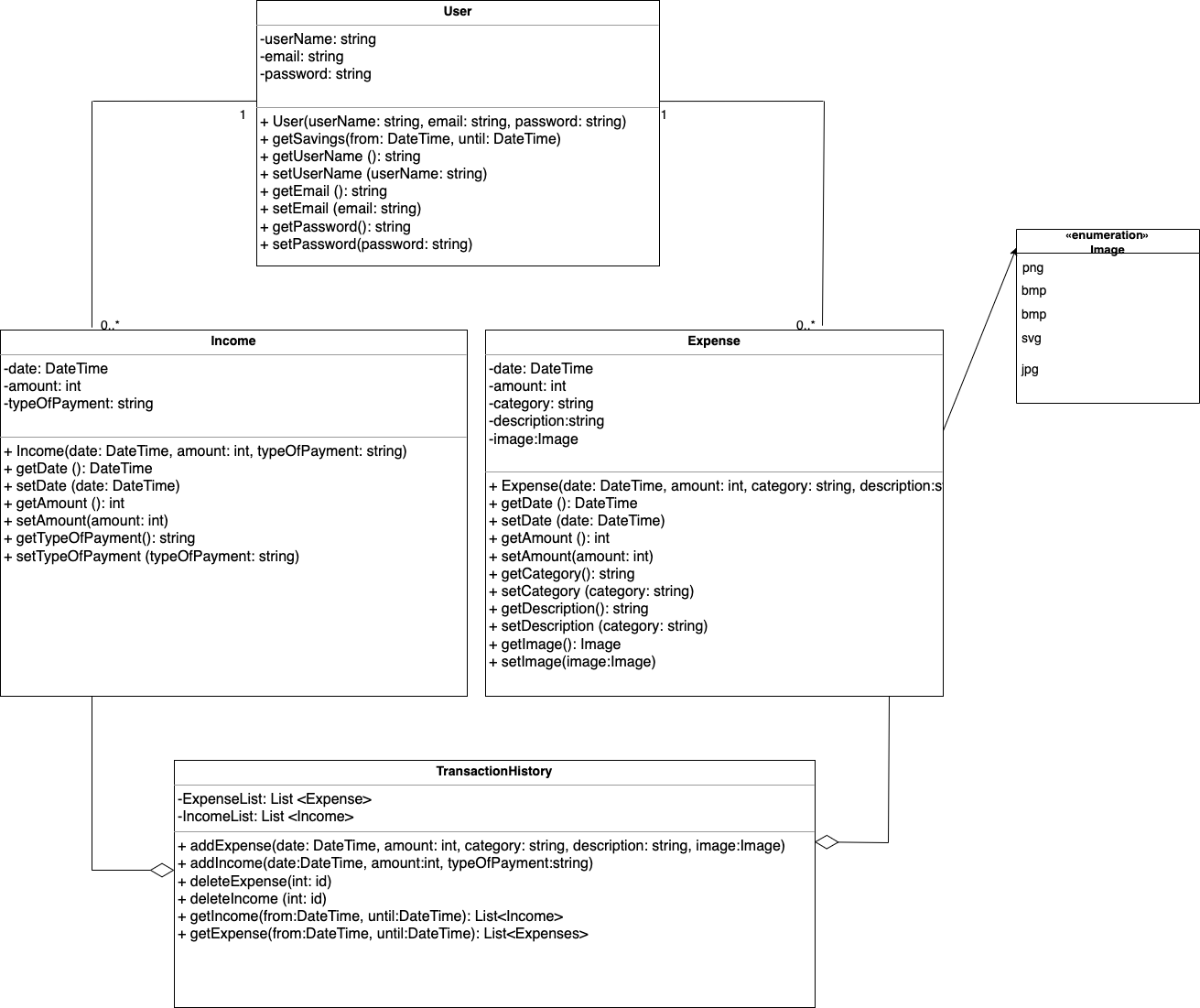
}

* APIs

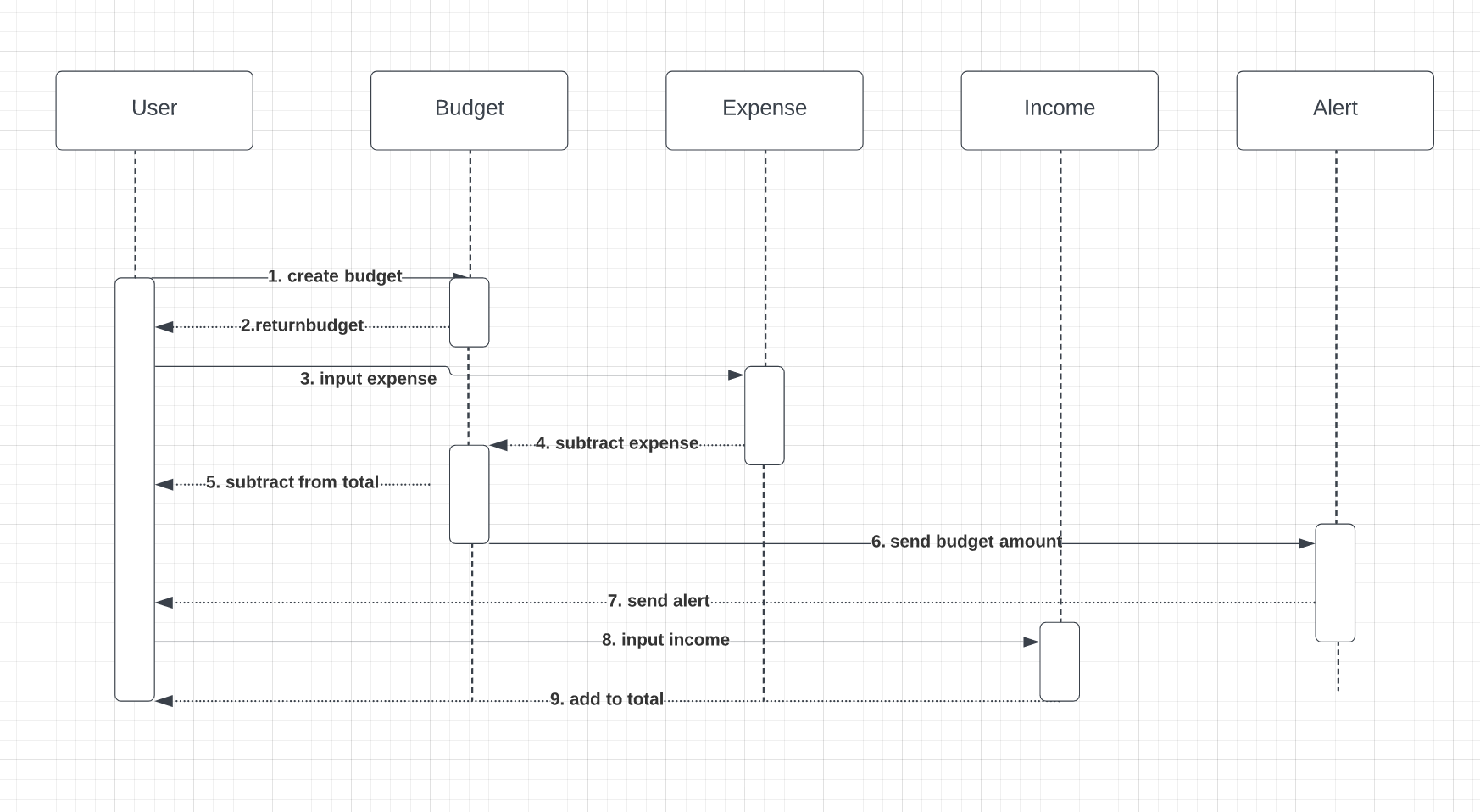
# The team uses the route (website\_name)/api to access the backend from the frontend. Any specific backend routes are appended to this route. There will be specific routes to make queries to find documents that only relate to the relevant user.

# 5. High Level UML Diagrams

UML Class Diagram



High-Level Sequence Diagram



6 .Identify ***actual*** key risks for your project at this time

* **Skills risks**

1. If working with technology selected becomes challenging, we could look for resources to understand the technology better.

* **Schedule risks**

1. If a team member can not complete an assigned task on time, we will look for a team member with similar skills to help out.

* **TeamWork risks**

1. If team members don’t communicate their updates on github, it could lead to merge conflicts, we can resolve that by having meeting regularly and communicating well

7. Project management

Describe in no more than half a page how your team managed M2 tasks including

* how each member’s progress is being shared in scrum meeting? Is it transparently shared?
* Outside the scrum meeting, do you have a tool to manage each member’s task?

During each team meeting, the main goal that needs to be tackled is discussed, as well as a plan for how to accomplish those goals. The team members volunteer for tasks and try to communicate to make sure what they’re doing is important in order for the project to succeed. Progress is transparently shared, and issues are discussed to allow for team members to work through any difficult problems with the group. Outside of scrum meetings, the team uses Discord to periodically inform each other of tasks completed and issues that may have been encountered. This helps team members avoid accidentally working on the same task, when the time could be better spent working independently and then bring their work together.