SW Engineering CSC648 / 848 Spring 2022

BusyGator

Team 04		
Role	**Name**	**Email**
Team Lead	Samantha Saxton-Getty	ssaxtongetty@mail.sfsu.edu
Github Lead	Vishal Ramanand Sharma	vsharma5@mail.sfsu.edu
Front End Lead	Elyssa Mari Tapawan	etapawan1@mail.sfsu.edu
Back End Lead	Aaron Carlson	acarlson8@mail.sfsu.edu
Front End	Abdullah Sharaf	fabdullah1@mail.sfsu.edu
Front End	Siqi Guo	sguo4@mail.sfsu.edu
Back End	Janvi Patel	jpatel6@mail.sfsu.edu

March 19, 2022

Milestone 02

Date	Version
03.19.22	Finished Version 01 submitted

Table of Contents

Executive Summary	**3**
Main Data Items and Entities - Glossary / Definitions	**4**
List of Functional Requirements - Prioritized	**6**
UI Storyboards for Each Main Use Case	**8**
High Level Architecture and Database Summary	**12**
Key Risks	**14**
Project Management	**15**

1. Executive Summary

There are many students who cannot find the resources they need to properly study, work out, or research. Our application, BusyGator, provides various materials to help SFSU students find everything they are looking for on campus all online. BusyGator is an e-commerce application that allows students to buy different items, from school materials to gym equipment, as well as list items for others to buy. This will motivate students to use these products offered by the application to better their education and health.

The BusyGator will have numerous functionalities and services. The main service the application will provide is allowing SFSU students and faculty to post and /or buy textbooks, sports gear, research devices, and much more all online. Each product that can be bought will show how much is in stock during the time of browsing, the price of the material, and the student / faculty selling the item. The BusyGator will provide a map pin-pointing various safe areas to pick up the product all around the SFSU campus. Other important functionalities the application will include are a search option and categories to look for more specific items, and a sign up / sign in option to confirm users are SFSU students or faculty. These services will provide users with an easy and accessible way to find various resources all in one place. BusyGator gives SFSU an opportunity to help their community by managing their resources within the application. Instead of going to different locations on campus to see if they can buy materials in person, the BusyGator will make this task a lot easier. It saves students and faculties the time and hassle in their already busy schedules. Also, this allows alumnis to give textbooks they no longer need to newer students who will utilize them more for classes. The unique aspect of BusyGator is that all of the features and services in buying and selling material will be in one place and in one press of a button, where anyone can easily follow.

We are Team 4 from the Software Engineering class of Spring 2022 consisting of Aaron Carlson, Siqi Guo, Janvi Patel, Samantha Saxton-Getty, Abdullah Sharaf, Vishal Ramanand Sharma, and Elyssa Mari Tapawan, with Samantha being the team lead. Other roles consist of Aaron being the back-end lead, Elyssa being the front-end lead, and Vishal being the GitHub lead. The back-end operates the parts of the application that aren’t accessed by a user such as data organization. The front-end works on the parts of the websites that allow users to interact with the web features, such as the application design. The GitHub lead ensures that there are no errors and that the application is functional. We are a group of aspiring students learning the aspects of team software development by making an application ourselves. Our goal is to work together to create the BusyGator to help us experience and prepare ourselves for our future careers.

2. Main Data Items and Entities - Glossary / Definitions

1. **<u>administrator:</u>**
   Definition: stores the login information of administrator accounts
   Sub-Items:
   - **1.1** administrator_id: primary key used for database identification
   - **1.2** email: email address of the administrator used for login and confirmation
   - **1.3** password: protected string created by the user for login

2. **<u>user:</u>**
   Definition: stores the login and personal information of user accounts
   Sub-Items:
   - **2.1** user_id: primary key used for database identification
   - **2.2** first_name: first name of the user
   - **2.3** last_name: last name of the user
   - **2.4** password: protected string created by the user for login
   - **2.5** email: email address of the user used for login and confirmation of official SFSU status
   - **2.6** date_created: date that the user account was created

3. **<u>approve:</u>**
   Definition: stores information required for the approval of posts
   Sub-Items:
   - **3.1** approve_id: primary key used for database identification
   - **3.2** administrator: foreign key identifier of the administrator handling approval
   - **3.3** post: foreign key identifier of the post requiring approval
   - **3.4** approved: flag reporting approval status

4. **<u>post:</u>**
   Definition: stores information regarding product posts made on the application
   Sub-Items:
   - **4.1** post_id: primary key used for database identification
   - **4.2** user: foreign key identifier of the user who created the post
   - **4.3** product: foreign key identifier of the product the post is showcasing
   - **4.4** date_created: date that the post was created

5. **<u>product:</u>**
   Definition: stores descriptive information regarding the application's products

Sub-Items:

5.1 product_id: primary key used for database identification
5.2 category: foreign key identifier of the category the product falls under
5.3 location: foreign key identifier of the location the product is to be exchanged at
5.4 title: name of the product
5.5 description: description given to the product
5.6 image: image showcasing the product
5.7 image_thumnail: image showcasing thumbnail version of product
5.8 price: price value attached to the product

6. **<u>location:</u>**
Definition: stores information regarding the locations used for exchanges
Sub-Items:
6.1 location_id: primary key used for database identification
6.2 name: name of the exchange location

7. **<u>category:</u>**
Definition: stores information regarding the product categories
Sub-Items:
7.1 category_id: primary key used for database identification
7.2 name: name of the product category
7.3 description: description given to the product category

8. **<u>transaction:</u>**
Definition: stores application transaction details
Sub-Items:
8.1 transaction_id: primary key used for database identification
8.2 buyer: foreign key identifier of the product's buyer
8.3 seller: foreign key identifier of the product's seller
8.4 product_id: foreign key identifier of the product involved in the transaction
8.5 date_created: date that the transaction occurred

9. **<u>message:</u>**
Definition: stores information regarding the sending of in-site messages
Sub-Items:
9.1 message_id: primary key used for database identification
9.2 creator_id: foreign key identifier of the message's sender
9.3 parent_message_id: foreign key identifier of the parent to the message

being sent

9.4 subject: subject of the message being sent

9.5 message_body: body of the message being sent

9.6 date_created: date that the message was created

10. message_recipient:

Definition: stores information regarding the recipient of in-site messages

Sub-Items:

10.1 message_repicient_id: primary key used for database identification

10.2 recipient_id: foreign key identifier of the message's recipient

10.3 message: foreign key identifier of the message being received

3. List of Functional Requirements - Prioritized

Priority 1:

Unregistered User:

1.1 Unregistered Users will be able to search posts.

1.2 Unregistered Users shall be able to filter search results based on categories. This function will help to narrow down posts that fit what the user is looking for.

1.3 Unregistered Users shall be able to register for an account. The registration email can only be a SFSU email.

Registered User:

2.1 Registered Users can perform all functions that Unregistered User can along with some additional functionalities.

2.2 Registered Users can log in to their accounts to buy / sell items.

2.3 Registered Users can create posts. This function is required to help Users sell their products.

2.4 Registered Users shall be able to edit the post after it has been submitted.

2.5 If the product is not available anymore then the User can delete the post.

2.6 Registered Users can log out of their accounts. This is to enhance security in case they are accessing the account from a public place.

<u>Priority 2:</u>

Unregistered User:

1.1 Unregistered Users shall not contact the seller. They will not be able message the seller unless they have a registered account.

Registered User:

2.1 Registered Users can contact the seller of the post via in-site messaging after buy. This function is required for transactions between users.

2.2 The product can be reported to the Administrator if the product isn't accurate to the description provided / images displayed. In this case the seller shall be reported to the Administrator.

Administrator:

3.1 Administrators can perform all functions that a Registered User can.

3.2 Administrators shall be able to delete or edit posts made by the User, if they do not follow the terms of service or they are offending in any capacity.

3.3 Administrators will be able to approve a post before it goes live, so that none of the posts go against the terms of service.

<u>Priority 3:</u>

Registered User:

2.1 Registered Users can propose the meeting location and time to complete the transaction.

2.2 Registered Users can request to reset their password in case they forget their password.

2.3 Registered Users can look at their transaction history in case they want to revisit a post.

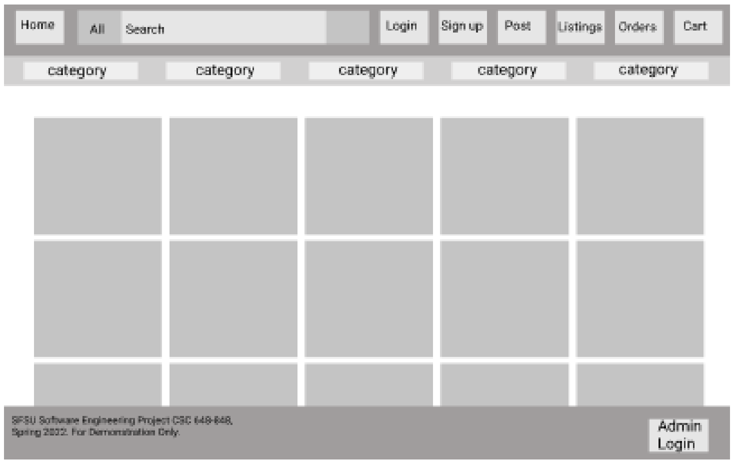
Administrator:

3.1 Administrators will be able to delete / ban a User if they consistently fail to comply with the terms of service.

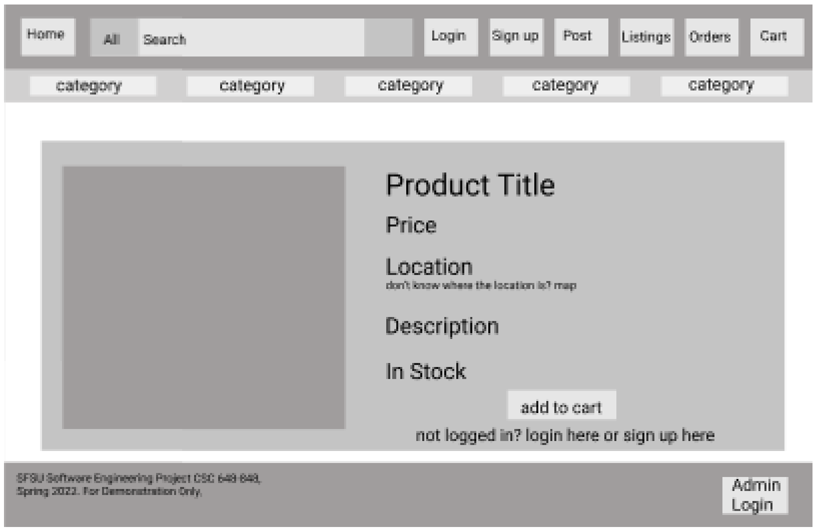
4. UI Storyboards for Each Main Use Case

<u>Use Case 1: SFSU Student/Faculty (Buyer)</u>

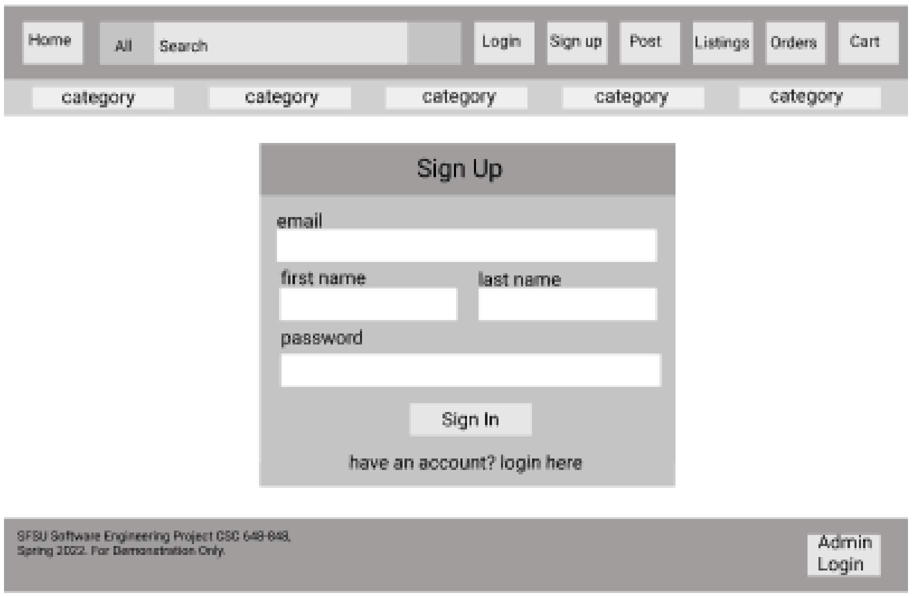
John is a student and a part-time worker at SFSU. He is looking for a textbook for his class on BusyGator. He enters what he is looking for in the search bar. The results show up on the "BusyGator" page.



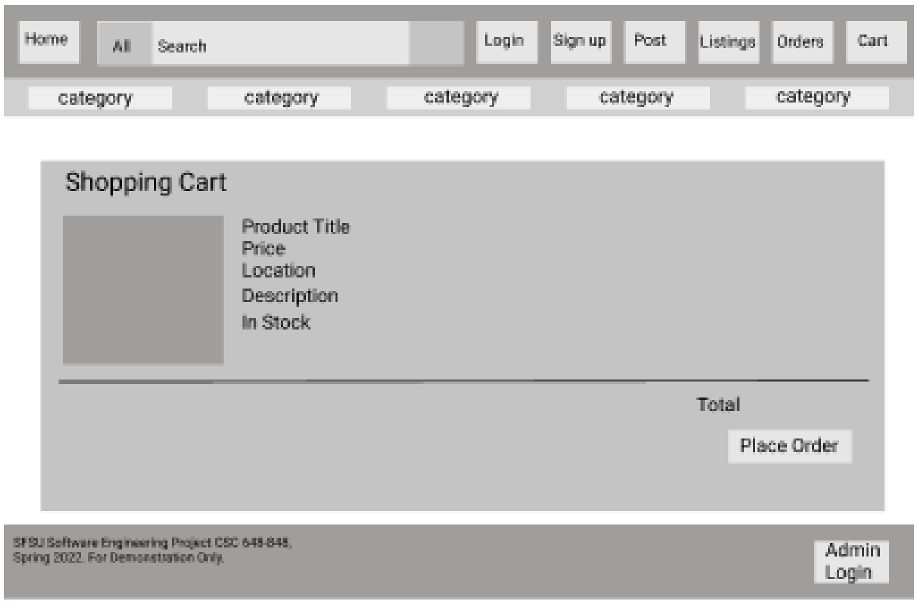
John looks on the page of all the results. He clicks on the product and sees the details of the textbook. He tries to add the item to the cart, but it prompted to log in or sign up for an account before he can add the item to his cart.


Home
All
Search
Login
Sign up
Post
Listings
Orders
Cart
category
category
category
category
category
Product Title
Price
Location
don't know where the location is? map
Description
In Stock
add to cart
not logged in? login here or sign up here
SFSU Software Engineering Project CSC 648-848, Spring 2022. For Demonstration Only.
Admin Login

John doesn't have an account yet, so he clicks on the signup button and creates an account on the website by inputting his SFSU email, first name, last name, and password.

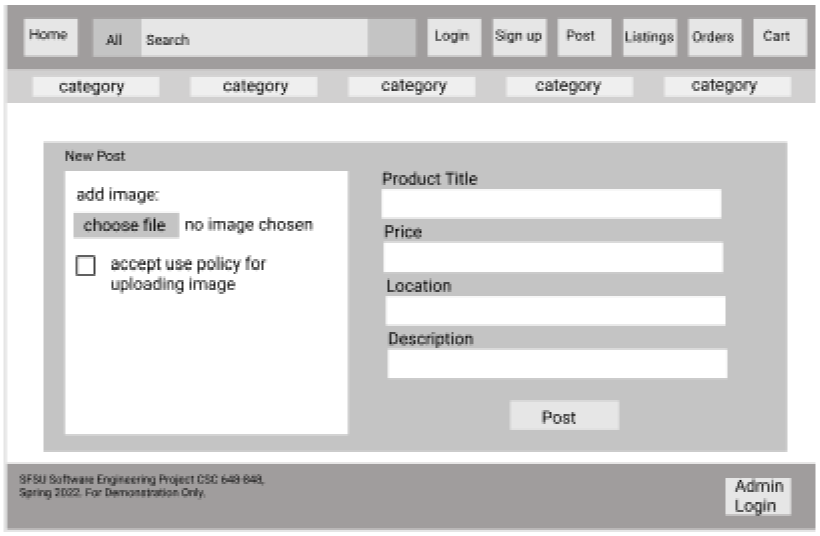


After the account has been made, John logs in and can now add the product to the cart. John proceeds with the buying process.

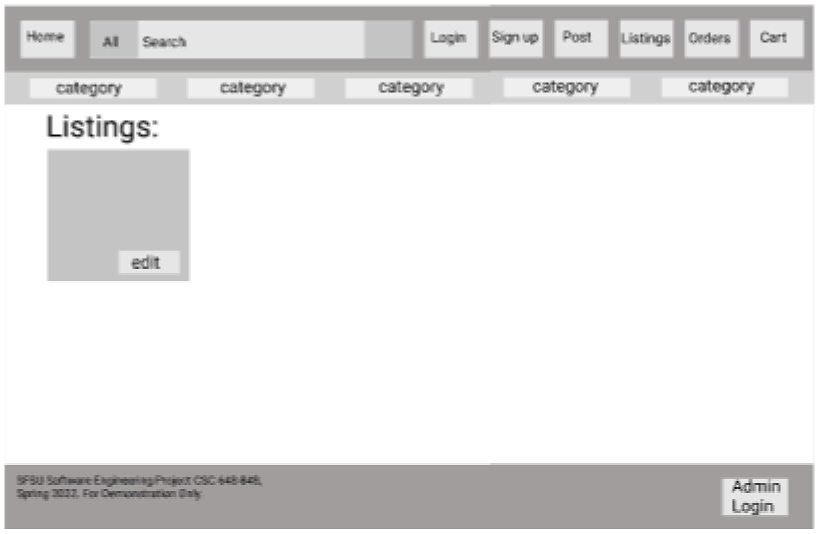


Use Case 2: SFSU Student (Seller)

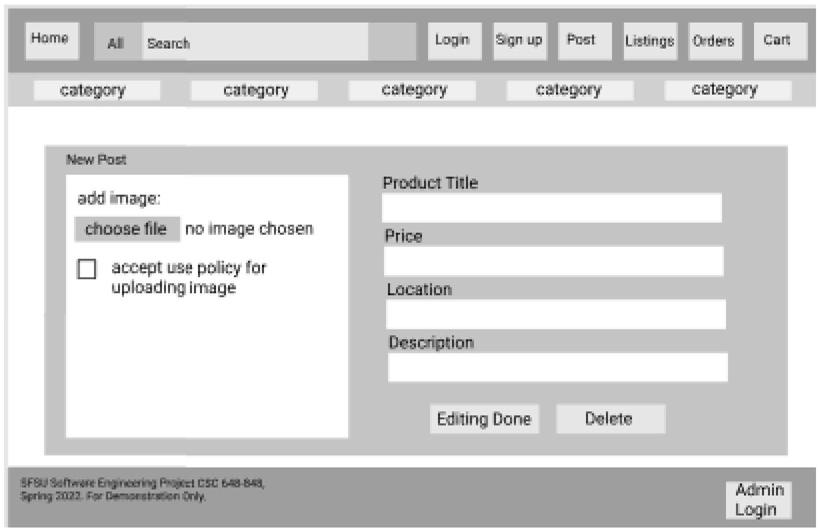
Jennifer is a full-time SFSU student looking to list a textbook to sell. To list an item for sale, Jennifer needs to have an account on the BusyGator. Jennifer repeats the same process as use case 1 to log in, where she can now list a product. To list items, Jennifer needs to fill out the product's description when she is making a new post. After she finishes, she clicks the post button to have it listed.



Jennifer wants to update the price for an item she has already listed. She presses the my listing button on the navigation bar to see her listings and clicks the one she wants to edit. To edit the listing, she presses the edit button to update the prices for the listed item. She updates the price and saves the listing.

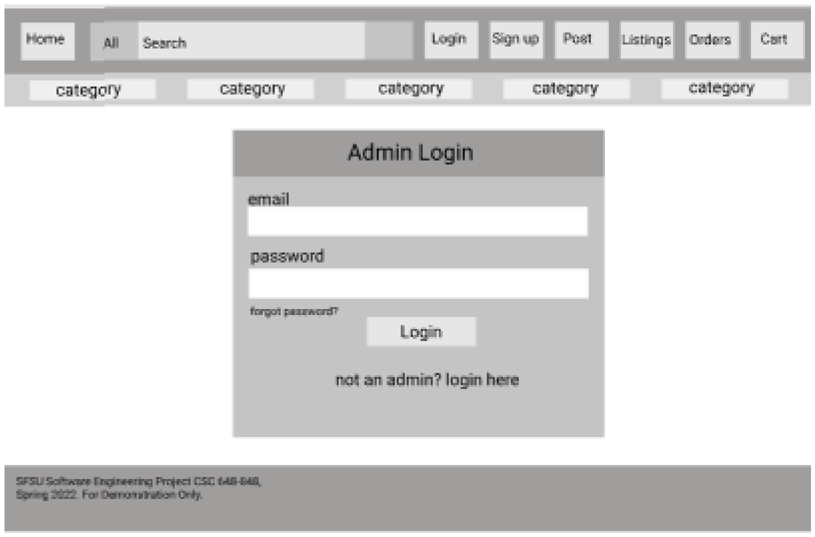


Jennifer finds out that she lost the textbook and wants to close the listed item. She clicks the listings button again on the top. She presses the edit button and presses the delete button where it will prompt "Are you sure you want to delete this listing?" She presses yes and deletes the listing.



Use Case 3: Admin

Mike is an administrator of BusyGator. He logs in through the admin login using his email, name, and password.



5. High Level Architecture and Database Summary

DB Organization:

BusyGators database organization is centered around two account types, user and administrator. Both account type tables hold information regarding the login while the user account also has more in-depth details regarding the users personal information.

The user interacts with products through transactions and posting, while the administrator interacts with products through approving user posts. Products are items with detailed information, including references to tables of location and category. In addition, the user interacts with other users through messages with the message recipient.

The naming convention utilized in this database is keyword based. If there is more than one keyword in the name, it is separated by an underscore. This keyword describes the data variable. For example, *keyword1_keyword2_keyword3, keyword1_keyword2* or *keyword*.

DB Tables:

1. **administrator**
   - **1.1.** administrator_id (PK)
   - **1.2.** email
   - **1.3.** password

2. **user**
   - **2.1.** user_id (PK)
   - **2.2.** first_name
   - **2.3.** last_name
   - **2.4.** password
   - **2.5.** email
   - **2.6.** date_created

3. **approve**
   - **3.1.** approve_id (PK)
   - **3.2.** administrator (FK)
   - **3.3.** post (FK)
   - **3.4.** approved

4. **post**
    - **4.1.** post_id (PK)
    - **4.2.** user (FK)
    - **4.3.** product (FK)
    - **4.4.** date_created

5. **product**
    - **5.1.** product_id (PK)
    - **5.2.** category (FK)
    - **5.3.** location (FK)
    - **5.4.** title
    - **5.5.** description
    - **5.6.** image
    - **5.7.** price

6. **location**
    - **6.1.** location_id (PK)
    - **6.2.** name

7. **category**
    - **7.1.** category_id (PK)
    - **7.2.** name
    - **7.3.** description

8. **transaction**
    - **8.1.** transaction_id (PK)
    - **8.2.** buyer (FK)
    - **8.3.** seller (FK)
    - **8.4.** product_id (FK)
    - **8.5.** date_created

9. **message**
    - **9.1.** message_id (PK)
    - **9.2.** creator_id (FK)
    - **9.3.** parent_message_id (FK)
    - **9.4.** subject
    - **9.5.** message_body
    - **9.6.** date_created

10. **message_recipient**
    - **10.1.** message_recipient_id (PK)
    - **10.2.** recipient_id (FK)
    - **10.3.** message (FK)

Media Storage:

BusyGators media will be stored in a file system. The relative paths for each file will be stored in the image attribute within the product table. There will be a limit of maximum file size.

Search / Filter Architecture and Implementation:

BusyGator will have a search bar that allows the user to search by item title and description. Users are able to then further filter by price and location. These filters will be created using the location and price attributes of the product table. If the user selects a category from the drop down or the sub navbar, that category is included with the search item. We will utilize SQL queries with %like% to match category and search terms together, or match the search term through all items if there is no category selected.

6. Key Risks

Skills Risk:

We will be working on different platforms and using different programs. Some of these programs are new to several of our group members. Our team members are guiding and helping each other knowing how it works.

How to Resolve:

For the members doing it for the first time, it will be difficult but our team shares resources, mini tutorials and guiding through Discord and Zoom meetings. We also do more research while using it.

Schedule Risk:

All the team members have different class schedules and it makes it hard to set up a group meeting outside of class for follow-ups or discussions. So mostly the communication depends on discord.

How to Resolve:

We use When2Meet so everyone can vote for the timings they are available to meet in and we go with the time slot where the majority of people are available. So far Monday's and Friday's around 5 PM - 6 PM have been the fixed check-in times and it has been going good so far.

7. Project Management

As soon as milestone 02 was released and reviewed in class, we immediately reviewed the document as a team to ensure everyone understood the milestone in its entirety. We focused heavily on revising milestone 01 for a smooth transition into milestone 02 since it would be a backbone for a lot of the content in the second milestone. Following this, I made a copy of our milestone 01 Google document and tailored it to the current milestone and its tasks. I added comments to each portion of the document assigning which member would complete each task, some including a brief note on expected work. During our meeting, we reviewed each assigned task and cleared up any questions before each member began their work. I utilized Discord to keep a list of "to-do" items that included meetings, deadlines for deliverables, and notes so that each team member could reference what would be going on during the given week. However, in the future I plan to use Trello as suggested by the professor for a more clean and concise area to divide work and display "to-do" items. Another tool we relied heavily on during meeting times was When2Meet. As a team we utilized this tool to quickly schedule follow up meetings that all team members were able to make in order to check-in on work progress, give feedback, and answer questions that weren't covered over text on Discord. This milestone proved challenging in coordinating the front-end and back-end teams on their deliverables and concepts. Going forward, we will be utilizing Trello for project management and scheduling more frequent meeting times to ensure both teams are on the same page for fluid development.