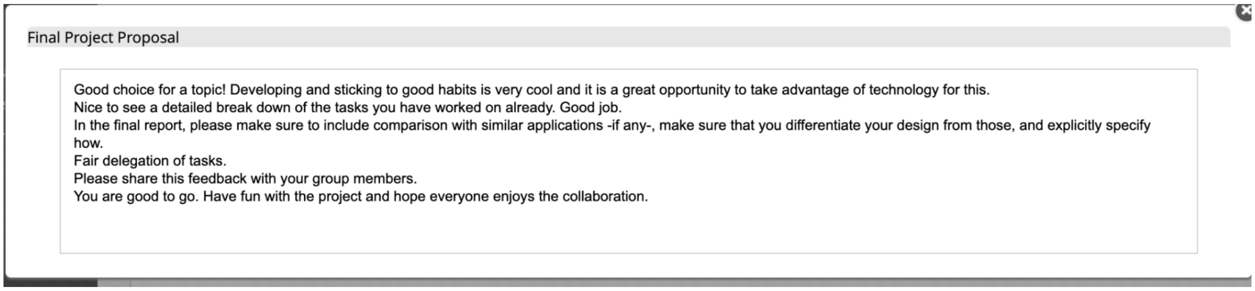
## CS3354 Software Engineering
## Final Project Deliverable 2

# Habit

Sharmin Gaziani, Abdurrehman Zulfiqar, Jaggan Jestine, Sam Williford,
John Bui, Peter Shen, Winifred Ojo.

**<u>Project Deliverable 1 content:</u>**

1. **[5 POINTS] Please attach here the Final Project draft description (that contains the instructor's feedback)**.



Final Project Proposal

Good choice for a topic! Developing and sticking to good habits is very cool and it is a great opportunity to take advantage of technology for this.
Nice to see a detailed break down of the tasks you have worked on already. Good job.
In the final report, please make sure to include comparison with similar applications -if any-, make sure that you differentiate your design from those, and explicitly specify how.
Fair delegation of tasks.
Please share this feedback with your group members.
You are good to go. Have fun with the project and hope everyone enjoys the collaboration.

Above we have attached a copy of the feedback we received from the Final Project Proposal. The feedback we received was largely positive. The ideas and task delegation for our project passed the professor's approval and we look forward to implementing it in a way that emulates professional enterprise software engineering. We'll be sure to get familiar with any and all similar products in the market and seek to make our own project unique.

2. **[10 POINTS] Setting up a GitHub repository.**

   Group Repository URL: https://github.com/sharmingaz/3354-60-Days

3. **[5 POINTS] Delegation of tasks:**

   Sharmin: For the project I'm tasked with gathering all the materials from team members to submit them, creating the GitHub/adding all the members to it, and building the project schedule.

   John: I am tasked with getting information to list the software requirements for the project clearly. I will also be responsible for evaluating the personnel estimation.

   Jaggan: For the project, I am tasked with creating the UML class diagram, as well as creating a software test plan that may be used when the project is ready for testing.

   Sam: I will look at similar projects to compare designs and create a sequence class diagram.

   Abdurrehman: I'll be in charge of creating our first README - aside from that, my main roles will be coming up with some pricing estimation figures as well as working on architectural design for our project.

   Winifred: I am tasked with making a deliverable conclusion on the project as well as creating a use case diagram.

   Peter: I will be creating our project scope commit and doing research on the hardware/software estimation for our product.

4. **[5 POINTS] Which software process model is employed in the project and why.**

   The waterfall model is the software process model employed in this project. This is because of its linear sequence lifecycle model that adopts a simple structure of phases in which the result of each phase affects and falls into the next phase of development. The waterfall model helps keep everyone involved in the project up to speed on the latest development involving the project. This is because technical documentation is necessary for the initial phase of development. That way everyone knows what they are doing and their goals and objectives. By defining all requirements needed for a project, the costs can be estimated with a high level of accuracy preventing extra costs from being imposed later during development. The waterfall model makes testing easier and transparent as the test scenarios would have been explained in the functional specification of the requirement phase [2].

5. **[15 POINTS] Software Requirements including 5.a.) [5 POINTS] Functional requirements.**

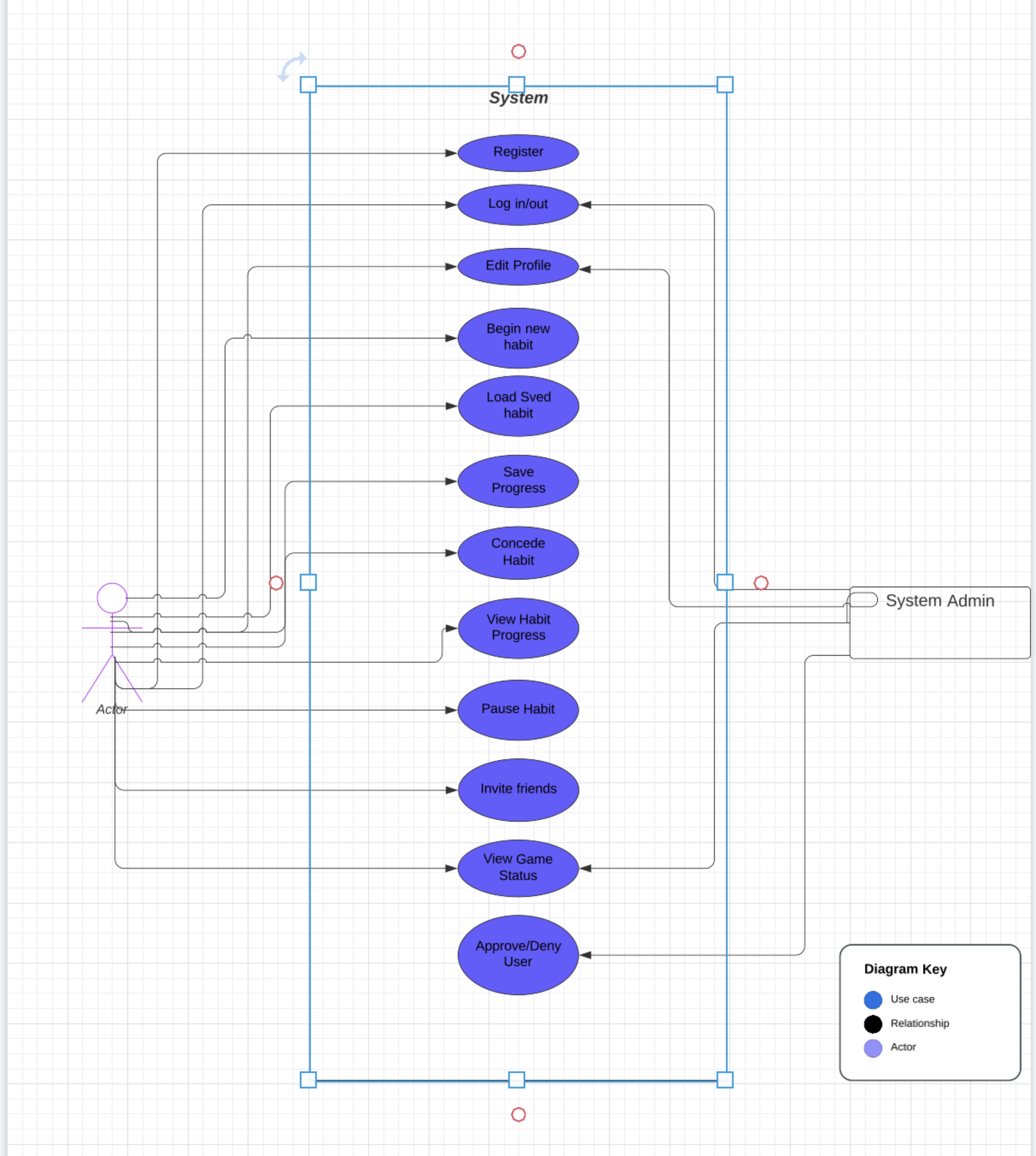
- The system must be able to authenticate the user's login credentials.
- The system must send a lock screen notification to remind the user at a specific set time every single day for 60 days of that certain habit.
- The system must be able to let the user set multiple habits at the same time.
- The system must be able to reward the user for maintaining progress shown on a progress bar.
- The system must be able to deplete your progress bar for missing days and a missed day will add a day to build towards the habit
- The system must be able to let you add your friends and view their profiles to see their completed habits

**5.b.) [10 POINTS] Non-functional requirements**

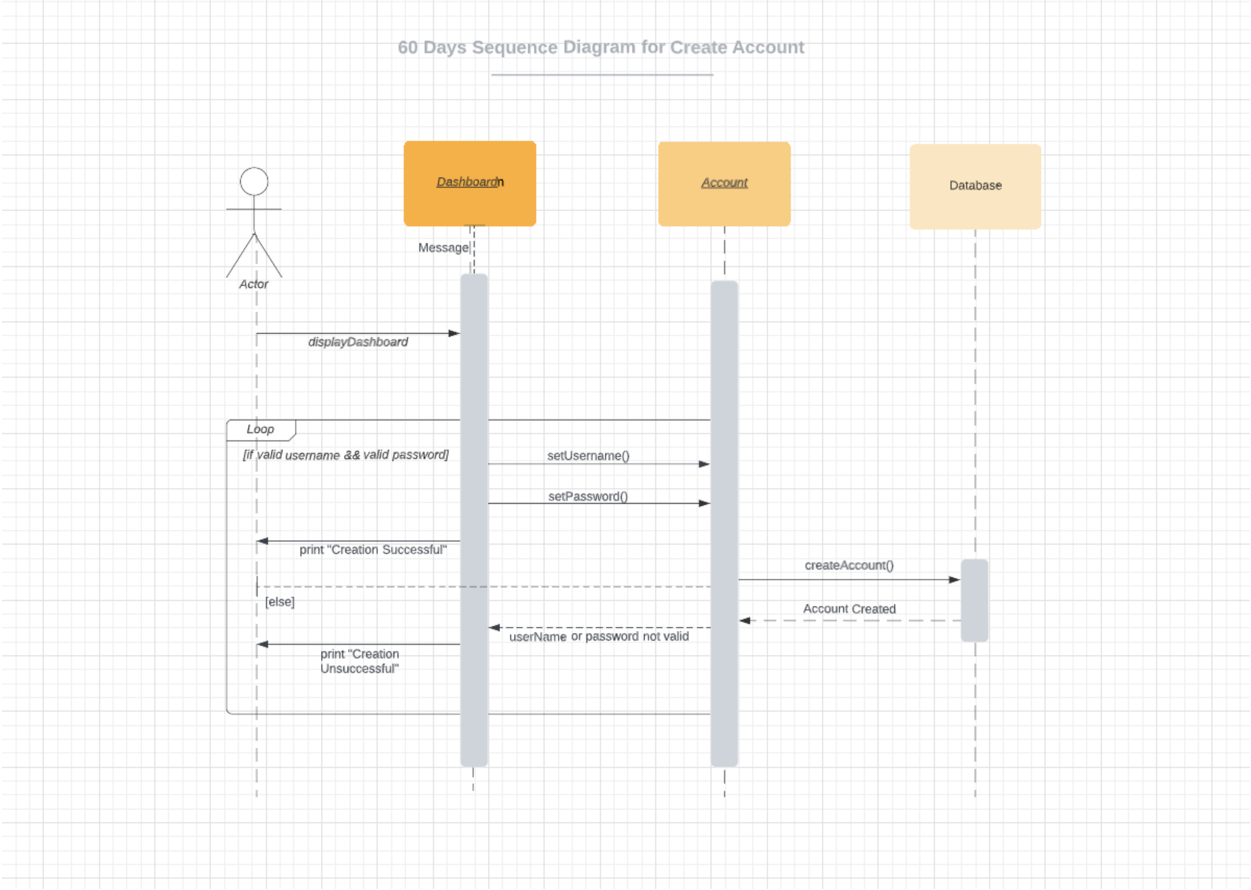
- Usability requirements: The interface must be easy to learn and navigate, and functions are easy to understand and simple to interact with and with few or no errors.
- Performance requirements: The system shall respond in no more than two seconds, and must be sustainable with multi-user concurrent execution where the load on server resources may not be more than 75 percent on average.
- Space requirements: The space needed for the software shall be within 100MB, any update size should be under 30MB.
- Dependability requirements: The system shall be available to the user 24/7 and downtime at in anytime shall not exceed three seconds.
- Security requirements: The system's firewall shall be able to prevent threats from malicious software. The server will cut its internet connection whenever attacks are detected.
- Environmental requirements: The system shall be working under ambient temperature, and the hardware should be kept between 50 to 90 degrees Fahrenheit.
- Operational requirements: A total load of habits shall not exceed 12 hours each day. Users should log in to the system with their own accounts to record the progress of their habits.
- Development requirements: The system must be adjustable and implementable so that it can be maintained and developed for further updated packages.

- Regulatory requirements: The software must be consistent with the regulation under all circumstances, such as Telephone Consumer Protection Act (TCPA), and General Data Protection Regulation (GDPR).
- Ethical requirements: The software must be ethical with the user’s privacy and data, the user data should only be used in the system, and any type of data leak is prohibited.
- Accounting requirements: The system must be performed by rules under U.S. GAAP basis financial statements.
- Safety/security requirements: The source code of the software will not be open source to prevent damage to the database and protect the user’s privacy

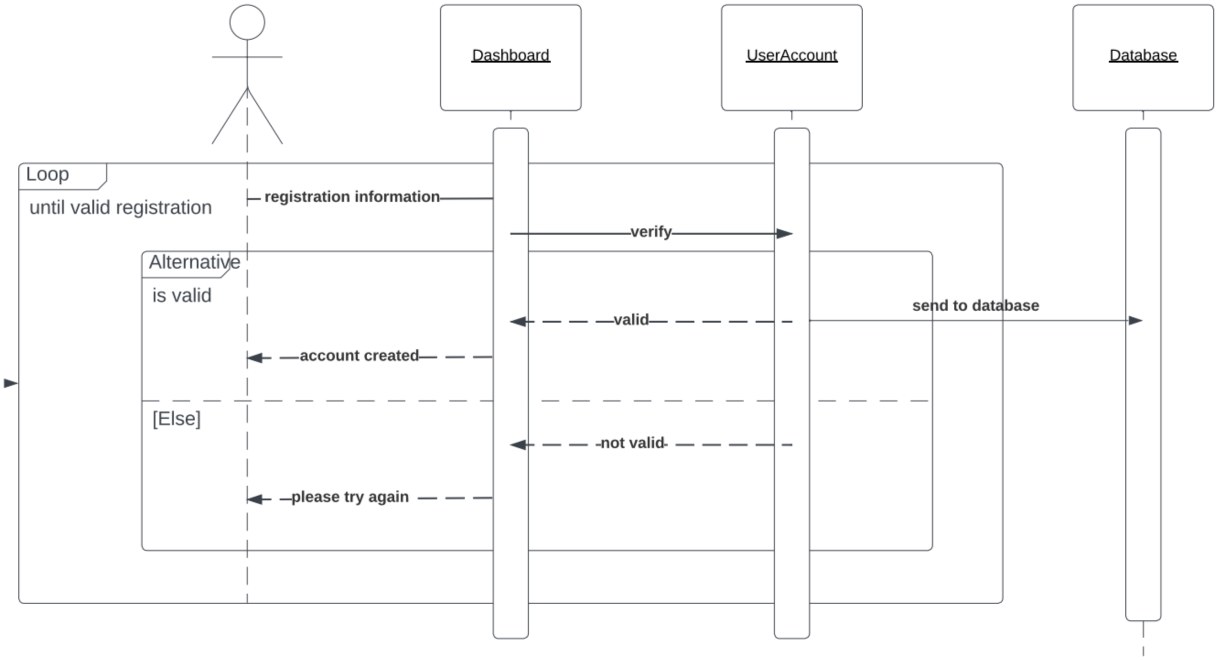
## 6. [15 POINTS] Use case diagram –



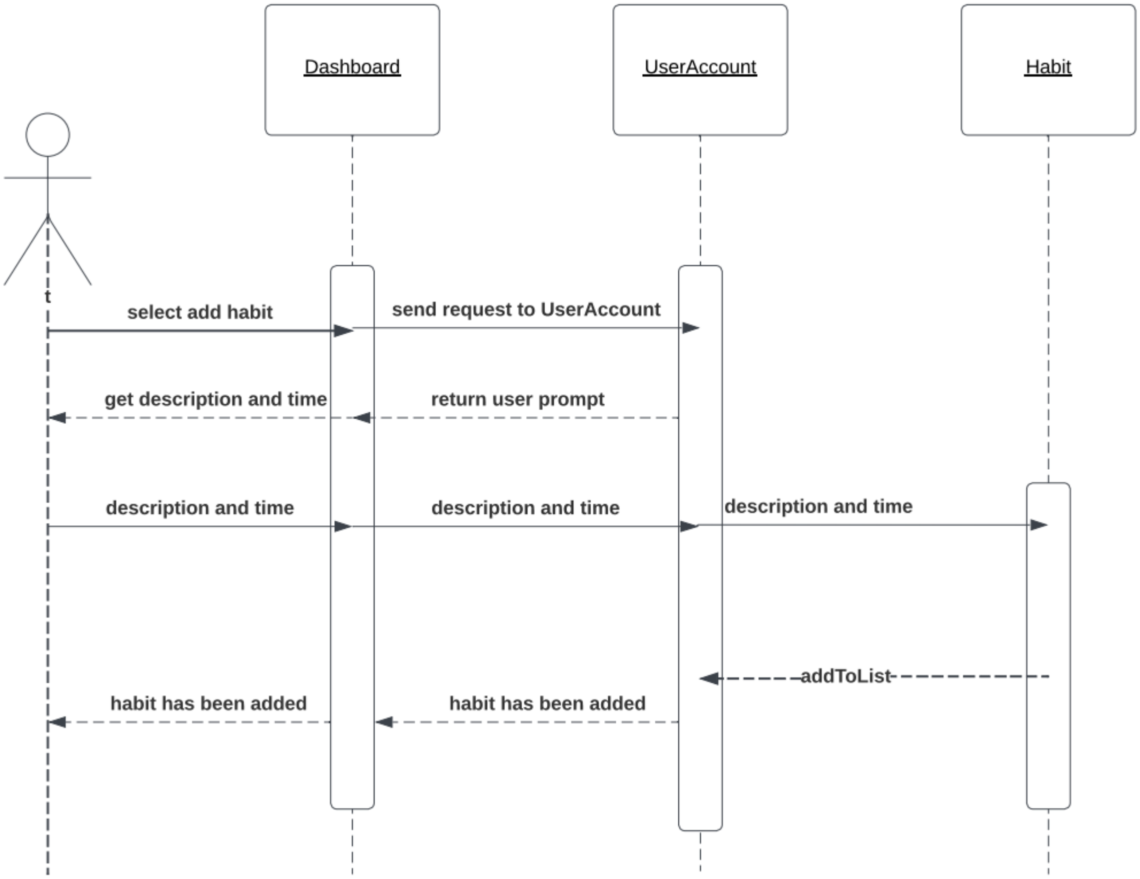
7. **[15 POINTS] Sequence diagram –**


60 Days Sequence Diagram for Create Account
Dashboardn
Account
Database
Actor
Message
displayDashboard
Loop
[if valid username && valid password]
setUsername()
setPassword()
print "Creation Successful"
createAccount()
[else]
Account Created
userName or password not valid
print "Creation Unsuccessful"

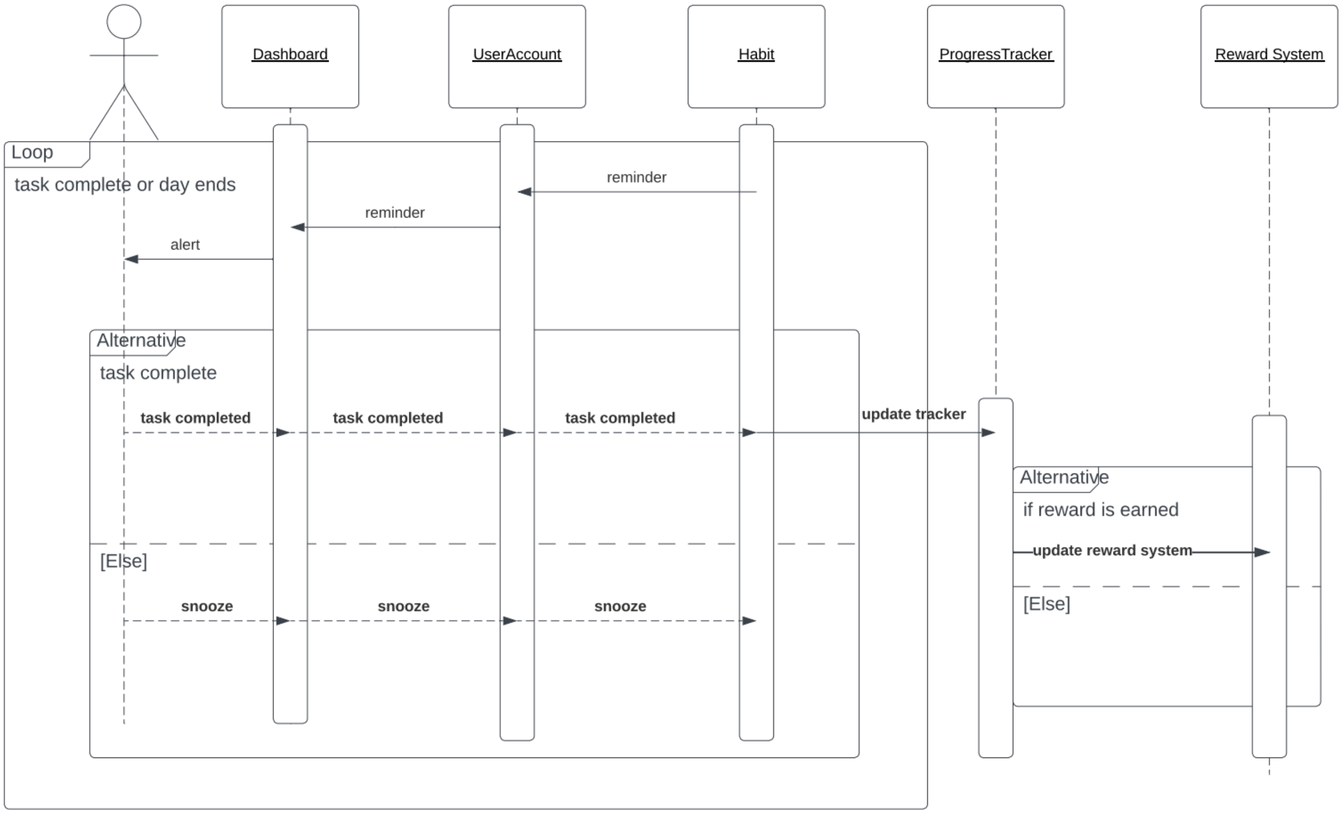
User Registration
Sequence Diagram



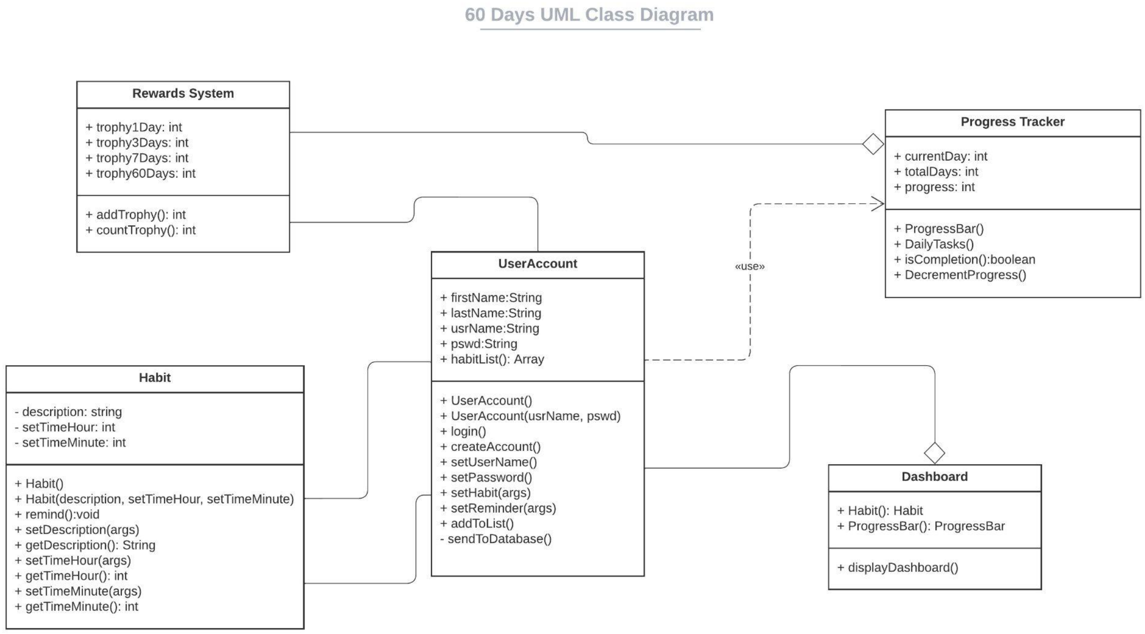
Add a Habit
Sequence Diagram



Daily Use Sequence Diagram

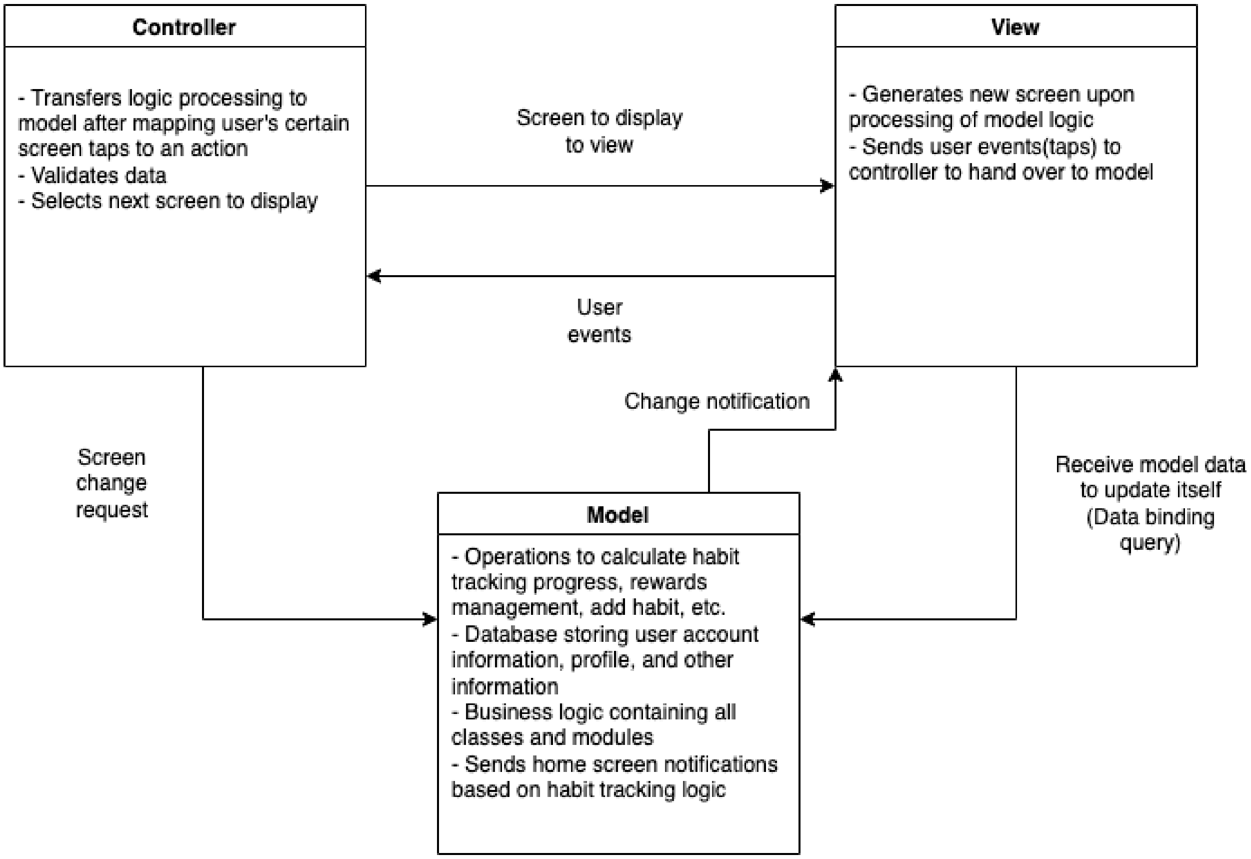


8. **[15 POINTS] Class diagram –**



9. **[15 POINTS] Architectural design –** Provide an architectural design of your project. Based on the characteristics of your project, choose and apply only one appropriate architectural pattern from the following list: (Ch 6 section 6.3) 9.1. Model-View-Controller (MVC) pattern (similar to Figure 6.6) 9.2. Layered architecture pattern (similar to Figure 6.9) 9.3. Repository architecture pattern (similar to Figure 6.11) 9.4. Client-server architecture pattern (similar to Figure 6.13) 9.5. Pipe and filter architecture pattern (similar to Figure 6.15)

The architectural pattern we'll be using in our project will be the Model-View-Controller pattern (MVC). We chose MVC for the specific use case of our system - in it there exist multiple views as well as multiple ways to interact with data. Additionally, future requirements for data interaction and data presentation are not yet known to us - however, with how mobile applications are built, it is natural to have updated releases in the future. Moreover, users may create certain potential use cases that may require additional requirements in the future. The MVC model is good for adapting to these new requirements, as it allows the data to change independently of its representation and vice versa. Changes made in one representation will be made in all of them. Additionally, our project setup closely fits the MVC model in its design. For one, the Model component is carried out on the length of a habit that has been set so far and the progress building up to the end of the 60 days. The View component is represented through our app dashboard, and the Controller component handles when the user carries out key actions such as creating or deleting a hobby, and subsequently passes these interactions to the View and the Model. What is done with the interactions depends on the component they are passed to - either the results are client-facing, or they are done on the data and backend [3].


Controller
- Transfers logic processing to model after mapping user's certain screen taps to an action
- Validates data
- Selects next screen to display
Screen to display to view
View
- Generates new screen upon processing of model logic
- Sends user events(taps) to controller to hand over to model
User events
Change notification
Screen change request
Receive model data to update itself (Data binding query)
Model
- Operations to calculate habit tracking progress, rewards management, add habit, etc.
- Database storing user account information, profile, and other information
- Business logic containing all classes and modules
- Sends home screen notifications based on habit tracking logic

## Project Deliverable 2 content:

1. **[5 POINTS] Well-described delegation of tasks:**

   Sharmin: For the project I was tasked with gathering all the materials from team members to submit them, creating the GitHub/adding all the members to it, and building the project schedule.

   John: I was tasked with adding the project scope to our Github repository.

   Jaggan: For the project, I was tasked with creating the UML class diagram, as well as creating a software test plan that may be used when the project is ready for testing.

   Sam: I was tasked with writing the conclusion for deliverable 2 and creating a sequence class diagram.

   Abdurrehman: I was in charge of creating our first README - aside from that, my main roles were coming up with some pricing estimation figures as well as working on architectural design for our project.

   Winifred: I was tasked with creating a comparison of works as well as creating a use case diagram.

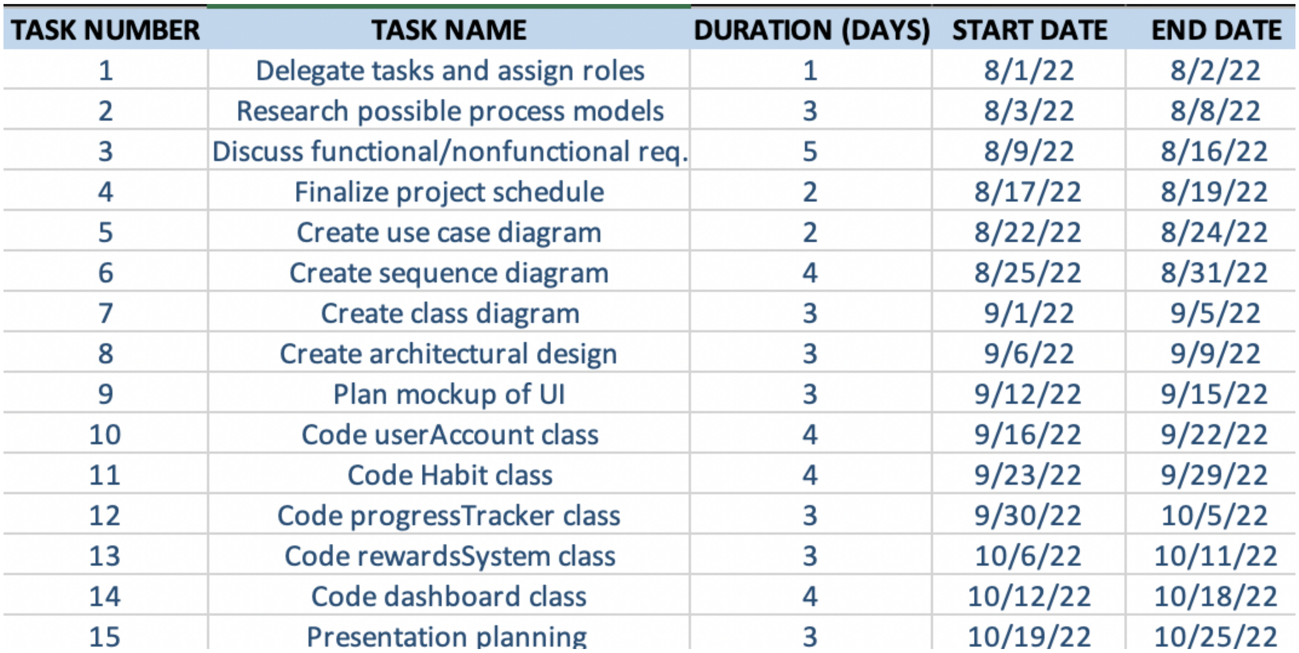
   Peter: For my part, I created hardware and software estimation plans as well as our function and non-functional requirements.

2. **[10 POINTS] Everything required and already submitted in Final Project Deliverable 1.**
   Final Project Deliverable 1 content is displayed above.

3. **[35 POINTS]** Project Scheduling, Cost, Effort and Pricing Estimation, Project duration and staffing: Include a detailed study of project scheduling, cost and pricing estimation for your project. Please include the following for scheduling and estimation studies:

### 3.1. **[5 POINTS] Project Scheduling.**

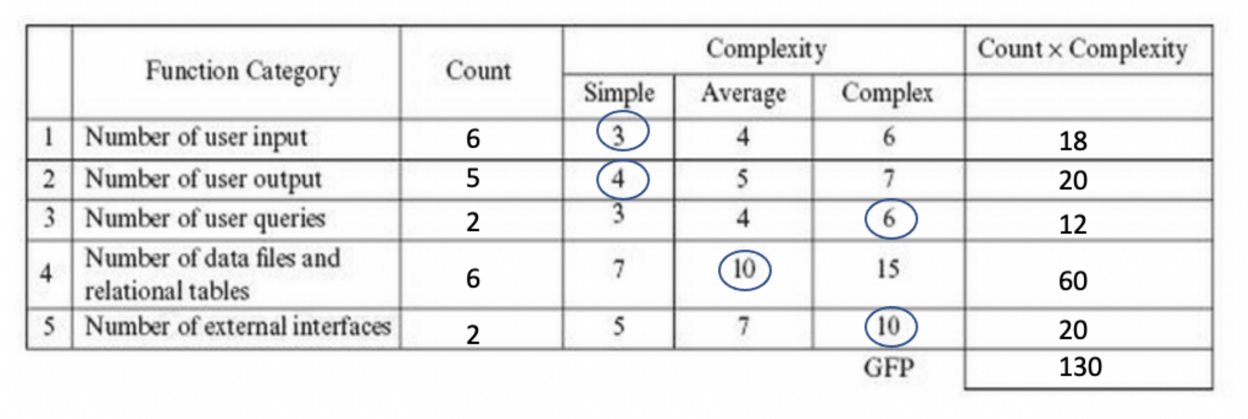
Below I have attached a project schedule including task names, duration days, and our start/end dates. Our schedule reflects a work week of Monday through Friday with an average of 30-40 hours per week. Our team decided to follow this schedule to allow employees time away from the project, similar to a corporate schedule.



| TASK NUMBER | TASK NAME | DURATION (DAYS) | START DATE | END DATE |
|---|---|---|---|---|
| 1 | Delegate tasks and assign roles | 1 | 8/1/22 | 8/2/22 |
| 2 | Research possible process models | 3 | 8/3/22 | 8/8/22 |
| 3 | Discuss functional/nonfunctional req. | 5 | 8/9/22 | 8/16/22 |
| 4 | Finalize project schedule | 2 | 8/17/22 | 8/19/22 |
| 5 | Create use case diagram | 2 | 8/22/22 | 8/24/22 |
| 6 | Create sequence diagram | 4 | 8/25/22 | 8/31/22 |
| 7 | Create class diagram | 3 | 9/1/22 | 9/5/22 |
| 8 | Create architectural design | 3 | 9/6/22 | 9/9/22 |
| 9 | Plan mockup of UI | 3 | 9/12/22 | 9/15/22 |
| 10 | Code userAccount class | 4 | 9/16/22 | 9/22/22 |
| 11 | Code Habit class | 4 | 9/23/22 | 9/29/22 |
| 12 | Code progressTracker class | 3 | 9/30/22 | 10/5/22 |
| 13 | Code rewardsSystem class | 3 | 10/6/22 | 10/11/22 |
| 14 | Code dashboard class | 4 | 10/12/22 | 10/18/22 |
| 15 | Presentation planning | 3 | 10/19/22 | 10/25/22 |

3.2. **[15 POINTS] Cost, Effort, and Pricing Estimation.**

- Function Point (FP)



| | Function Category | Count | Complexity | | | Count × Complexity |
|---|---|---|---|---|---|---|
| | | | Simple | Average | Complex | |
| 1 | Number of user input | 6 | [✓ 3] | 4 | 6 | 18 |
| 2 | Number of user output | 5 | [✓ 4] | 5 | 7 | 20 |
| 3 | Number of user queries | 2 | 3 | 4 | [✓ 6] | 12 |
| 4 | Number of data files and relational tables | 6 | 7 | [✓ 10] | 15 | 60 |
| 5 | Number of external interfaces | 2 | 5 | 7 | [✓ 10] | 20 |
| | | | | | GFP | 130 |

The essential process among 14 questions is the communication between data and the application designed to facilitate change and ease of use by the user.

Thus, PC = PC1 + PC2 + ... + PC14 = 12 × 3 + 2 × 5 = 46.

The processing complexity adjustment PCA is then computed:

PCA = 0.65 + 0.01 × 46 = 1.11

From these, the function points FP are calculated:

FP = GFP ×PCA = 130 × 1.11 = 144.3 FP

The estimated effort is obtained as we assume the productivity from each person is 60:

E = FP/productivity = 144.3/60 = 2.45 ≈ 3 person-weeks

If team size is 3, then project duration is:

D = E / team size = 3/3 = 1 week

3.3. **[5 POINTS] Estimated cost of hardware products (such as servers, etc.)**

The average cost to rent a small business dedicated server is $100 to $200/month. But we can also set up a cloud server for about $40/month to have adequate resources. Therefore, the estimated cost of hardware products is approximately $150 per month.

3.4. **[5 POINTS] Estimated cost of software products (such as licensed software, etc.)**

On average, Microsoft 365 and Google Workspace business and enterprise licenses cost $10-$30 for each user every month. Therefore, the estimated cost of software products is approximately $100 per month.

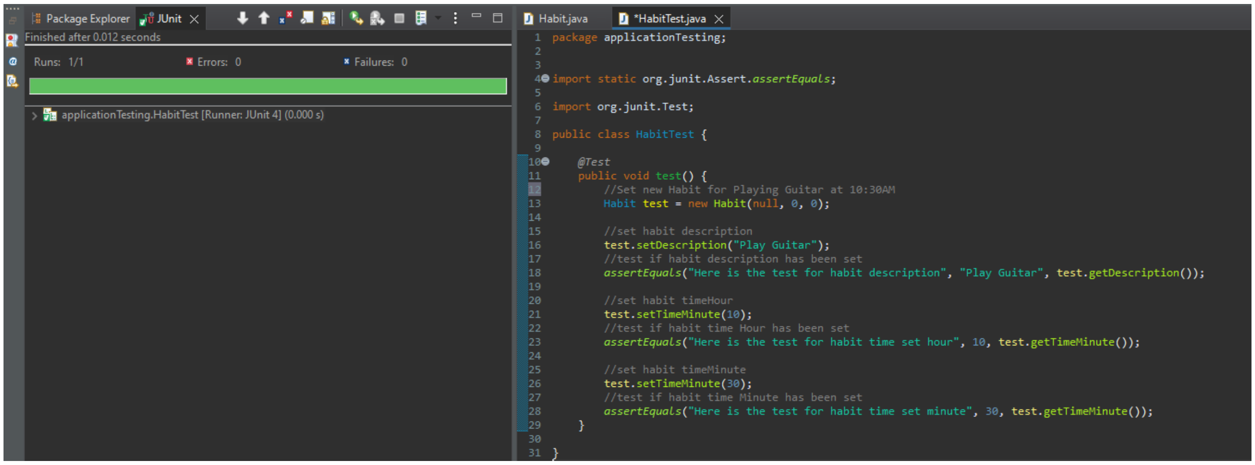
3.5. **[5 POINTS] Estimated cost of personnel (number of people to code the end product, training cost after installation)**

According to the previous estimation the number of people for our development team is 3. The estimated cost of personnel will be $1500 × 3 ≈ $5000. This makes sense as most figures citing enterprise class contract-based software developers list the cost as being around $850, and with an extremely short period of time of 1 week as outlined by the previous estimation, we can expect to stack on top of that initial amount until it gets to around $1500 per developer. [4]

## 4. [10 POINTS] A test plan for your software:

In our software, we plan on testing it through the use of JUnit. The main functions we would need to test for the functionality of the software would be the createAccount function, login function, setHabit function, setReminder function, isCompletion boolean, and the ProgressBar. The createAccount and login function can be tested simultaneously with automation testing scripts that autofill the fields and confirm that accounts were created and were able to log in successfully. This allows us to test the setUsername and setPassword functions. Once login is successful, we can test the setHabit and setReminder functions with a few JUnit test cases, as seen below. This almost completes the functional testing requirements. The final functional test would be the ProgressBar class, we would have to test the isCompletion boolean, by simply checking if the Daily Task was completed and setting the boolean variable based on the result. Once we confirm these functionalities work, we can continue testing the non-functional requirements, as these functions rely on the functional requirements to run. The Rewards system class and the Dashboard class can be tested through JUnit, they are simple classes that use the values from the Progress Bar class, the User Account class, and the Habit class. The Dashboard class represents the overall UI of the software, so this would be the last requirement we would test and it could be achieved through preview app deployment testing, which is possible through XCode IDE and Android IDE. Finally, the reward system class is used by the tested simply through the addTrophy function and the countTrophy function, this is achieved by a JUnit test to check if the function properly adds a trophy to the user's account and then calling the counting function to return the count of the trophies.

JUnit Screenshot:



Finished after 0.012 seconds

Runs: 1/1 Errors: 0 Failures: 0

applicationTesting.HabitTest [Runner: JUnit 4] (0.000 s)

```
package applicationTesting;


import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class HabitTest {

    @Test
    public void test() {
        //Set new Habit for Playing Guitar at 10:30AM
        Habit test = new Habit(null, 0, 0);

        //set habit description
        test.setDescription("Play Guitar");
        //test if habit description has been set
        assertEquals("Here is the test for habit description", "Play Guitar", test.getDescription());

        //set habit timeHour
        test.setTimeMinute(10);
        //test if habit time Hour has been set
        assertEquals("Here is the test for habit time set hour", 10, test.getTimeMinute());

        //set habit timeMinute
        test.setTimeMinute(30);
        //test if habit time Minute has been set
        assertEquals("Here is the test for habit time set minute", 30, test.getTimeMinute());
    }

}
```

5. **[10 POINTS] Comparison of your work with similar designs.**

    The main point of our app is its simplicity. Compared to other habit-tracking apps on the market such as Habitshare, Habitica, and StickK, our app is designed to help cultivate new habits individually or with friends and family while providing a reward system to motivate users to hit the 60 days mark.

    Habitica is an RPG video game-inspired app that treats its real life like a game. To motivate users, one's customized avatar levels up when a task is completed. One can also unlock in-game features such as battle armor and quests. Like our app, one can be joined by friends to complete quests and missions. Habitica stores data persistently in a database management system using MongoDB. The architecture is structured nicely into different layers with a single page application for the client side and backend services accessible through an API.

    Habitshare is a habit tracker that also doubles as a social network. It has messaging options that enable you to talk and communicate with people you know. With Habitshare, your individual goals transform into meaningful team efforts. However, Habitshare doesn't feature scheduling tasks monthly.

    StickK is a habit-tracking app that influences behavior change through loss aversion and accountability. Users are required to make a commitment contract and set a costly deadline. Users bet money on themselves and can assign someone to check the data to ensure that one is on the right track to reaching their goals. if they fail in their goals, it costs them money. This app has more than 10 communities with like-minded goal setters to interact, offer support, and share best practices.

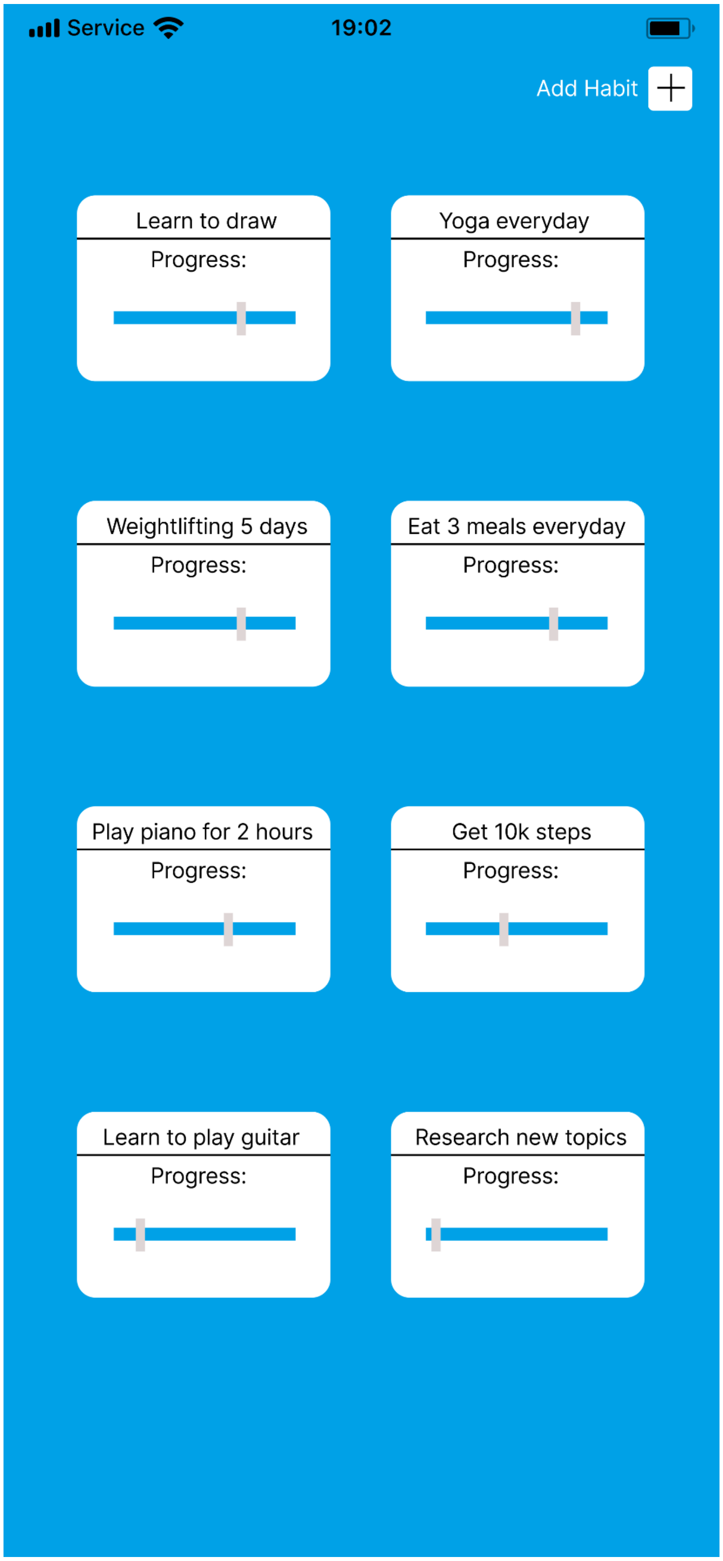
6. **[10 POINTS] Conclusion**

    Our project started out with a brainstorming session where each team member suggested ideas and then voted on which idea he or she liked the best. We ultimately decided to design an application that would help people build good habits and track their progress. As our team worked throughout the semester, we realized that we had slightly different ideas about the design and details of how the end product would function. Throughout the semester, as we studied and learned about different aspects of software engineering, we created detailed requirements and diagrams, and our project transformed from a vague idea into something concrete that could actually be implemented. Overall, it was a valuable experience as we all were able to increase our understanding of the software development process.

7. **[5 POINTS] References:**
    References are listed on the last page.

8. **[10 POINTS] Presentation slides**.
    Presentation slides are included in the submitted zip folder as well as the GitHub.

9. **OPTIONAL PART [POSSIBLE EXTRA CREDIT UP TO 10 POINTS].**



**10. [5 POINTS] GitHub requirement:**

Group Repository URL: https://github.com/sharmingaz/3354-60-Days

# REFERENCES

[1] M. J. Basavaraj and K. C. Shet, “Software estimation using Function Point Analysis: Difficulties and research challenges,” *Innovations and Advanced Techniques in Computer and Information Sciences and Engineering*, pp. 111–116, 2007.

[2] K. Petersen, C. Wohlin, and D. Baca, “The waterfall model in large-scale development,” *Lecture Notes in Business Information Processing*, pp. 386–400, 2009.

[3] A. Majeed, “MVC Architecture: A Detailed Insight to the Modern Web Applications Development,” pp. 1–10, Sep. 2018.

[4] “Software development price guide & hourly rate comparison,” *RSS*. [Online]. Available: https://www.fullstacklabs.co/blog/software-development-price-guide-hourly-rate-comparison. [Accessed: 18-Nov-2022].