Brock University

Computer Science Department

COSC 4P02 – Software Engineering 2

PROGRESS REPORT 2

HabitForge

www.habitforge.ca

Authors:

Abhi Patel – 6897334 – ap19wf@brocku.ca

Ameen Khawaja – 6935688 – ak19nu@brocku.ca

Zakir Raza – 6834923 – zr19zt@brock.ca

Nico McFarlane – 7001811 – nm20lw@brocku.ca

Rafael Bocsa – 7035801 – rb20qf@brocku.ca

Ahmed Yaser – 7063944 – am20gy@brocku.ca

Professor: Dr. Naser Ezzati-Jivan

Introduction

HabitForge project has been coming along extremely well. The report consists of work that we did in the past 2 sprints. We have achieved all the main features that we set out too in the beginning and solved most of our user stories. We have added a dashboard that lets users create and track habits that get stored and retrieved from the database. We have created the leaderboards page and added functionality that calculates the ranks and sorts all the users from highest to lowest rank. We also added other features that solves the secondary user stories we set out to solve before.

Sprint 4-5

Dashboard [Abhi]

Created the dashboard that enables users to add and track habits. There is a dialog popup that lets you create habits with the right fields and once created stores it in the firebase database. The dialog asks users for habit name, an icon, colour, goal amount, repeat, and start date. Dashboard pulls all habits from the habits collection in the user's collection. Users are also able to delete habits. I have also created the components for each individual habit that allows you to track them. You can mark them as completed, skipped, or failed.

Audio Inputs [Ahmed]

Audio feedback has been implemented as well as an optional mute button. Each action the user takes triggers the website to play different sounds that are appropriate towards the action.

The different sounds include:

- Create
- Fail
- Skip
- Error
- Success

In this context, we tend to keep it simple as we do not want the app to be cluttered by unnecessary sounds.

Levels/ Rank Function [Raf]

Purpose: To create a ranking system to rank players based on habit tracking performance. Rewarding players who have completed the most habits making sure the system cannot be exploited. All player data will be stored in firebase under each user.

Idea 1: Simply rank players based on how many habits they've completed

Ex:
Player 1:
Completed: 3
Total: 4

Player 2:
Completed: 1
Total: 1

Player 3:
Completed: 2
Total: 3

Ranked: Players 1>> 3>>2

This approach is too simple and doesn't consider various factors of tracking a habit and being consistent with hitting goals.

Idea 2: Rank players based on different factors with varying weights

Weights:
completionRate: 0.6,
goal: 0.2,
skipped: -0.3,
streak: 0.5

Total 100%

NOTE: Keep track of streaks per player if a player logs in consecutive days and completes at least 1 habit.

- First we calculate a completion rate for each player by:

completionRate = Habits Completed / Total Habits

- The we calculate each factor against its weight

completionScore = completionRate * completionRateWeight
goalScore = If player hit goal then goalWeight = 0.2, else = 0
skippedScored = If player skipped then skippedWeight = -0.3 else 0
streakScore = Streak * streakWeight

Then we add up all the scores to get a players final score. Finally, we rank all players based on this final score.

Example:

Player 1:
totalCompleted: 3
totalHabits: 4
goal: 1
skipped: false
streak: 3

Player 2:
totalCompleted: 2
totalHabits: 2
goal: 2
skipped: false
streak: 2

Player 3:
totalCompleted: 2
totalHabits: 2
goal: 1
skipped: true
streak: 1

Total Scores
Player 1: 2.15
Player 2: 1.8
Player 3: 1

Ranked Players 1>>2>>3

Conclusion: Ranking players based on various factors will be the best way to evaluate habit tracking performance and to reward those working hard on multi day streaks and not skipping a single day

Final Notes:

- Keep track of streaks
- Reset “skipped” / have it as skipped in the last week(boolean)
- FailedScore instead of goalScore

Icons [Zakir]

Multiple icons are implemented into the system where when users are creating a habit, they can select the icon they want. Most of the icons available are ones that relate to common habits. For example, working out as habit, you can put a dumbbell as an icon.

Audio To Text [Ameen]

The speech-to-text functionality is now fully implemented and working with the updated HabitForge UI. Alongside integrating the speech-to-text to work with the new UI, a few improvements have been made to it to account for any potential errors that could be made. For example, speech-to-text now only detects a habit name of less than or equal to 25 characters, which is the original constraint we set on the “Habit name” as a title. Additionally, the speech-to-text API is now able to differentiate between integers and words, whereas before it would not be able to differentiate between numbers and integers, such as if the user meant to say “2” it would result in “two”. Lastly, the “Habit repeat” menu component now only allows for the user to say one of either “weekly”, “daily”, or “monthly “. If neither of these words are said, the user is notified and can retry.

Tests [Ameen]

I’ve done research into the type of test cases that can be implemented with the stack of frameworks and tools we used for our projects. Since we are using Next and TypeScript, we are unable to do traditional unit testing as our project is component based. As a result, I’ve narrowed down the two different frameworks that are widely used to test components within our project. The first framework is Jest, which is initially for testing vanilla JavaScript. I’ve implemented Jest with our project, and it appears to work perfectly for component testing. Currently this sprint, around ~25 test cases were written for critical components that are used in the application, such as checking if a user is authenticated, has a habit created, or if the audio functionality is working. In the final sprint before our presentation, we plan to release a few more features. Once these features are released this upcoming sprint, I will also write the test cases for those components.

Motivational System Notifications [Nico]

The previous way we had our system notifications done are:
The motivational notifications quotes are added within the documents in the database. This will send notifications at a specified time everyday which is 12:00 pm. Along this, there are quotes within firebase which are then accessed and retrieved when the motivational quote needs to be called and it will choose a random quote from the database. If no errors are found it will immediately send a windows notification out preventing any memory leaks. Then using the useEffect which will set the function to trigger every 24 hours at specified time. But there was a flaw with this as the useEffect would only trigger if the app was running, i.e. when the tab in the browser is open. This is not ideal as we want to use these

notifications to get users to be motivated and log their habits even when they are not on the website. To fix this we decided to use this:
https://firebase.google.com/docs/cloud-messaging/js/send-with-console

The setup we have now is as follows:

We are using the firebase cloud messaging system. This allows easy access to the firebase cloud messaging with the notifications already created with the appropriate scheduling times and images (optional). This is used by having the payload which will automatically be made when the website is launched and is then put into the FCM registration token. This will allow the notifications to be sent to that device. It will still ask for permission and check permission before sending the notification.

Test Habits Creating/ Tracking [Abhi]

Tested the habits creation and testing so that the data is properly stored and brought in to showcase to the users. They are sorted by the time added (oldest to newest). Fixed a bug where the input fields for the habit creating would lose focus after the first character, was happening because React was rendering the fields each time the state changed, this way at every character press it would re render and lose focus. Fixed it by placing the form as part of the dialog instead of a separate component, now all of it renders at the same time.

Group Habits [Abhi]

Created the ability to create group habits, it has all the same fields as the normal habit dialog, and an extra one that enables you to write an email to add members. After the habit is created, there is a function that goes through all the users and checks if the emails provided in the created habit match to any user. If a match is found, a notification is added that lets user know they are invited. If they accept the notification a duplicate of the created habit is created.

Analytics Components [Zakir]

Implemented the analytics component under dashboard where there are two components. First is where information such as total number of habits completed, weekly progress, and longest streaks will be listed. The second component is for users to track their progress. For example, percent of habits completed. This component was created using functions and using the “react-circular-progressbar", which helps creating the circular progress bar.

Problems

We have not yet encountered any real problems; everyone has been doing the task they were given and get it done before the next sprint has started.

Future

Since most of the main features are added, the future sprints are just going to focus on testing that everything works and adding any features that we missed. We will also soon be starting to work on our final presentation.

Final Conclusions

HabitForge to the moon 🚀