**Team Name:** Orcalearns

**App Name:** Orcalearns

**Proposed Level of Achievement:** Apollo 11

Play
Learn programming in an interactive and gamified manner.
Forum
Ask, vote and comment on questions about the topics learnt or programming in general.
Reward
Collect achievements and gain points to reach higher levels from playing the game or participating in the forum.
Multiplayer
Get on the leaderboard by earning the most points or achievements, and compete with friends in quizzes and challenges.
ORCALEARNS
Learning to Code Made Easy
Flutter
Dart

## Motivation

It would be nice to **learn coding from a young age**, and be interested in further learning as well. Many have come to learn that coding can be very difficult to pick up in our later ages, and that it would have been better to start from a young age.

Many younger children nowadays are also enrolling in centers outside school to learn some programming, mainly as an extracurricular activity, so lessons would only be around 1.5 hours per week. These centers charge rather high fees for classes as well. Therefore, it would be advantageous to have an app that allows these students to **learn and practice at home** for **free** to further solidify their learning. However, as our target users are mainly young students, we would need to make our app exciting for them through **gamification**, or else they would quickly get bored and stop using the app.

## Aim

We hope to facilitate children’s learning of programming using an engaging gamified app.

## User Stories

1. As a young student learning coding, I want an interesting platform so studying is not boring.
2. As the parent of a young child who wants my child to learn some coding, I want to be able to track their progress.
3. As a student or parent, I want to be able to ask questions in a forum if I need any clarifications on certain topics.
4. As an administrator who wants to help answer the aforementioned forum questions and help students in general, I want to be able to answer the questions in a timely and concise manner, as well as police the forums for inappropriate content.

## Scope of Project

Basic gamified learning programme for Arduino, with proper pacing, achievements and advancements based on levels completed.

## Difference from Similar Platforms

- Duolingo
  - Only teaches spoken languages, does not teach coding languages
  - Lack of forum for students to help each other and for certified teachers to assist in issues
- Coursemology
  - Lacks a mobile app, so it is not suitable for students who are looking to learn on the go
  - Only accessible by students taking certain computer science modules in NUS, not for anyone interested in programming

## Tech Stack Choices & Rationale

<u>Front-end:</u>

1. Framework - Flutter
   a. Flutter is Google's free and open-source UI framework for building natively compiled applications for mobile, web, desktop, and embedded devices from a single codebase.
   b. Flutter supports a wide range of fully-customizable widgets that incorporate all critical platform differences such as scrolling, navigation, icons and fonts, which results in fast rendering and flexible designs, which we felt would be

suitable in the implementation of our application. Flutter's Hot Reload function also allows for faster development, which we figured would save us a lot of valuable time during our application development stage. The stability of Flutter code (unaffected by iOS or Android system updates as well as version compatibility with previous Flutter versions) were also appealing to us as it would greatly help simplify the development as well as maintenance of our app in the long run. As Flutter has been rapidly gaining popularity for mobile app-building, even beating our React Native in 2021 with over 109,600 stars on Github, we also felt that learning this new relevant technology would be useful for us in the future.

2. Language - Dart

    a. Dart is Google's client-optimized language for fast apps on any platform, and it is the programming language used to code Flutter apps.

    b. We mainly chose Dart as it is the language used in conjunction with Flutter. However, it has its own fair share of advantages. Dart is optimized for building user interfaces, with features such as sound null safety, the spread operator for expanding collections, and collection if for customizing UI for each platform. As we wanted to develop an app that would be aesthetically pleasing and simple for users to use, we felt that a programming language that was optimised specifically for the user experience would be the best fit. In addition, Dart is described as a language that is incredibly easy to learn, especially for programmers who are already familiar with languages such as Java, Javascript, as the language itself is very straightforward, and well-documented in a detailed, clear and organised manner, which was good for us as we did not have much programming experience before NUS. Finally, using Dart meant we could develop our app for both Android and iOS simultaneously.

3. User Interface Library - Material

    a. Material Design is an adaptable system of guidelines, components, and tools that support user interface design. Backed by open-source code, Material Design streamlines collaboration between designers and developers, and helps teams to quickly build beautiful products.

    b. We decided to use the Material Library in Flutter for the UI of our application. This refers to readily available Flutter widgets that implement Material Design, such as the Appbar or BottomNavigationBar. This is to help speed up the development of our application, as we would not have to spend time designing our own components to use. These components are quite aesthetically pleasing as well, and allow room for customisation should we wish to make any changes.

Back-end:

1. Firebase
    a. Firebase is a platform developed by Google for creating iOS, Android and web applications. It provides tools for tracking analytics, reporting and fixing app crashes, and also a real-time database, as well as user authentication.
    b. We decided to use Firebase as it manages all data real-time in the database, so the exchange of data to and fro from the database is easy and quick, essential for a mobile application that provides good user experience. It is also free, so no extra money is required for a backend server. In fact, no backend server is required. Additional features such as analytics and crash reports are also available.

Resources (Documentations):

Dart Documentation
Flutter Documentation
Material Design Documentation
Firebase Documentation

## Project Management

We implemented the Scrum Agile Framework for our project management. This framework focuses on advancing the project through short-term blocks of work called sprints, which are usually two weeks long. The scrum framework is heuristic, as it is based on constant learning and improvement to the project as new unknown factors come into play. It is described as a framework that acknowledges that the team does not know everything at the start of the project and will evolve through experience.

Our reasons for selecting this framework as opposed to other Agile frameworks are:

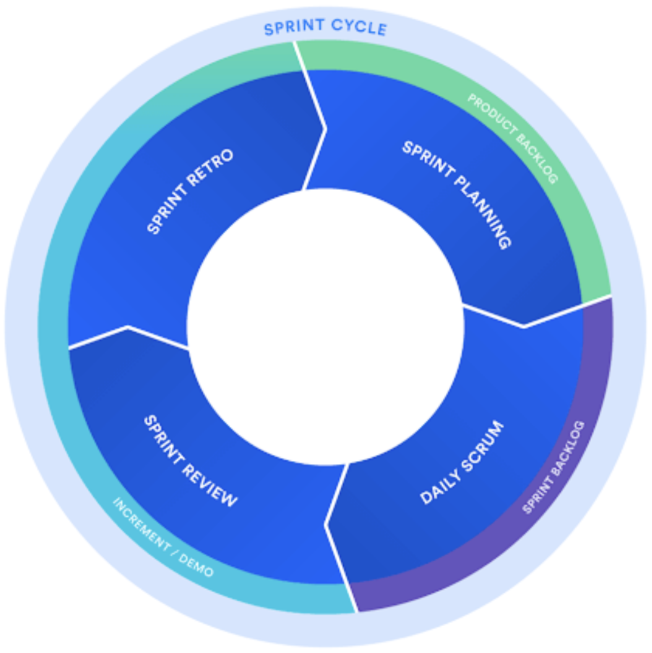
1. This framework is suitable for small self-organising teams (which are all Orbital teams), as opposed to large hierarchical ones.
2. The short period of the sprints means that we can quickly adapt to any changes that need to be made to our development plans, such as fixing certain parts of our application based on feedback or bugs.

3. The frequency of the sprints also serves to keep us on track throughout the 3 months of our Orbital project, as we will always have new goals to work towards and a clear deadline to ensure we do not procrastinate on our work. We will also be accountable to each other. Our sprints can also be said to lead up to the bigger and more important Orbital Milestones at the end of each month.

4. We can quickly provide feedback to one another and bring up any issues about each other's work during our meetings.

5. This framework is simple and easy to understand, removing ambiguities in the development process. Therefore, we felt that this was a good framework to implement to increase our productivity throughout our Orbital journey.

Below is a diagram representing each sprint cycle, as well as how we implemented this framework in our application development:



1. Product Backlog - an overall list of all the tasks that need to be done for the entire project. In our case this would be the list of features we have planned for our whole application, as shown in our README. We make changes to our planned features as required throughout the course of these 3 months, for instance changing the plan for our game implementation due to feedback from past evaluations.

2. Sprint Backlog - the list of things to accomplish within the current sprint. We would discuss during each sprint meeting what the features we wanted to accomplish by the next sprint meeting were so that we could keep on track with our application development and complete it by the Orbital deadlines. *(Sprint Planning)*

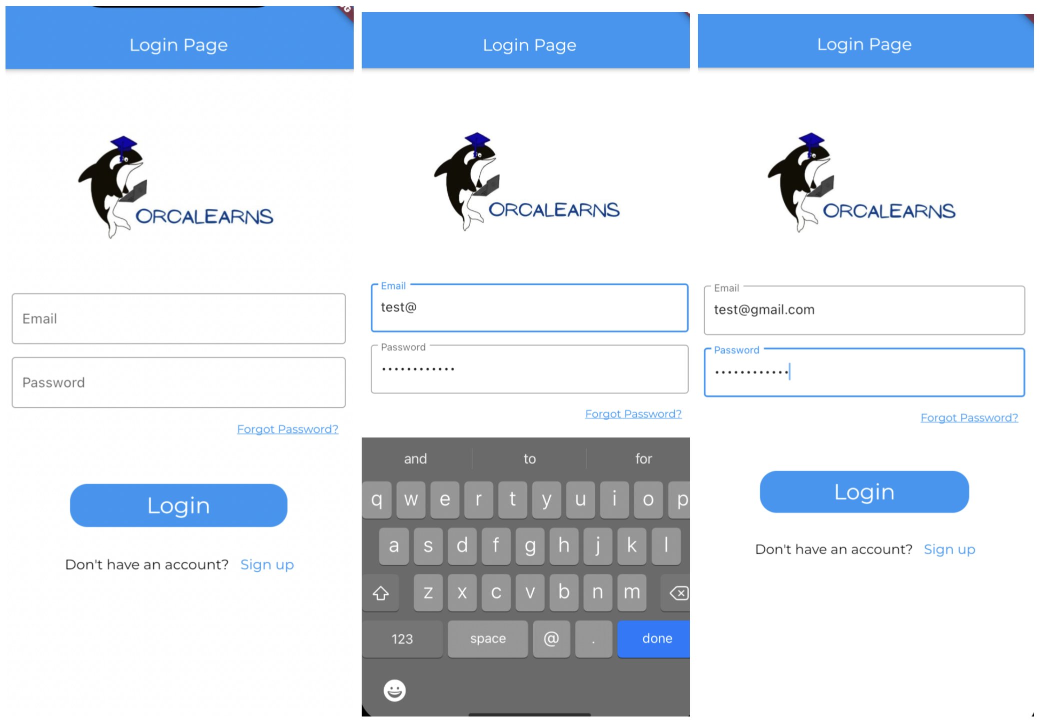
3. Increment - usable end-product from a sprint. During each sprint meeting, we would show each other what we had accomplished since our last meeting, and discuss if there were any issues present in our work as well provide suggestions on possible improvements. If we failed to produce the desired increment during the sprint period, we would have to justify why we were unable to do so, such as perhaps time constraints or struggles with the coding, and we would help each other if possible. We would also discuss how to improve our application overall. *(Sprint Review & Retro)*

4. Sprint - The actual period of time for us to individually work on our current sprint backlogs. Though sprints normally last two weeks in typical work environments, due to the time period of Orbital we decided to have week-long sprints instead. We also texted at least once a day to keep each other updated on our current programming activities or to ask for help. *(Daily Scrum)*

## Features

<u>Features done by Milestone 1:</u>

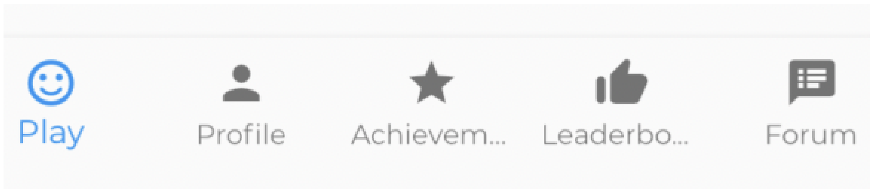
1. Login page



   a. Text boxes for users to input their email and password (TextField)
   b. Email address keyboard for email box, hidden text for password box (keyboardType and obscureText)
   c. Forgot Password button to help users reset their password if they forget (TextButton)

d. Login button that leads to the Home page (TextButton with boxDecoration)
e. Sign Up button that leads to the Register Page for new users (TextButton)

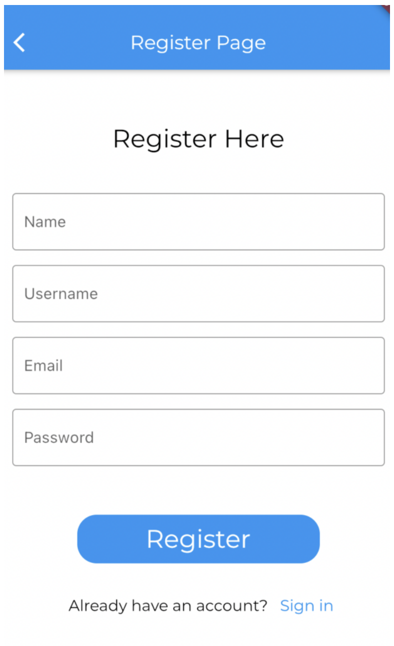
2. Home page with tabs

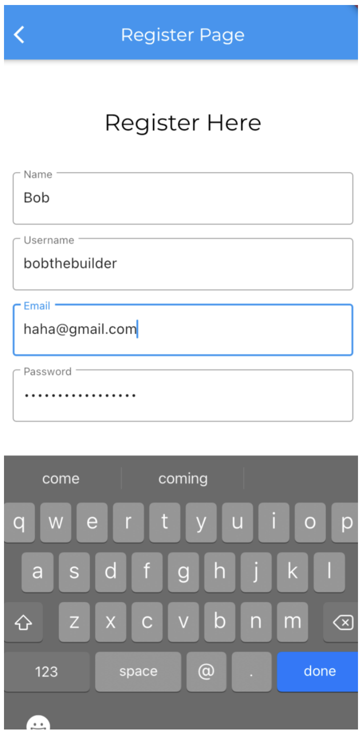


a. Tabs include *Play, Profile, Achievements, Leaderboard and Forum* (BottomNavigationBar)
b. Icons to better indicate the different tabs (Flutter Icons)
c. Clicking on the different tabs brings users to different features in our application, with the current active tab clearly highlighted in blue (onTap)

<u>Features done by Milestone 2:</u>

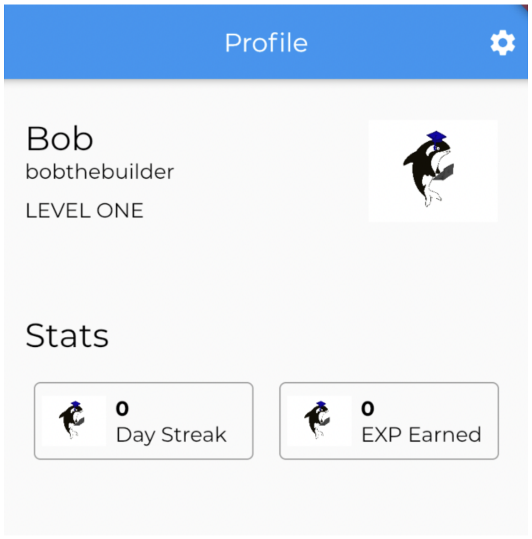
1. Register page





a. Text boxes for users to input their details (name, username, email, password) and create a new account (TextField)
b. Email address keyboard for email box, hidden text for password box (keyboardType and obscureText)
c. Successful registration of an account brings users to the Home page upon clicking on the Register button (TextButton with boxDecoration)
d. Linked page to Firebase to store new registered user accounts
e. Sign In button that leads to the Login Page for existing users (TextButton)

2. Profile

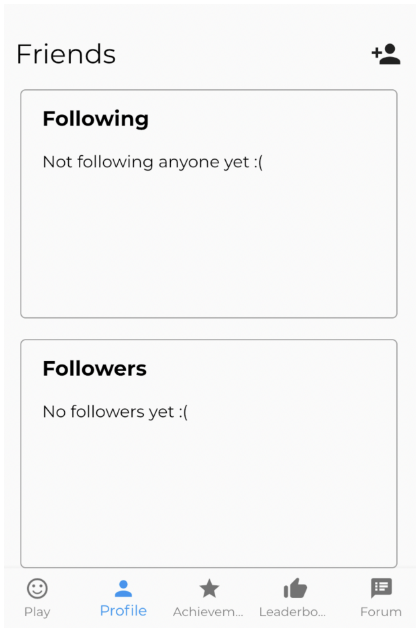

Profile

Bob
bobthebuilder
LEVEL ONE

Stats

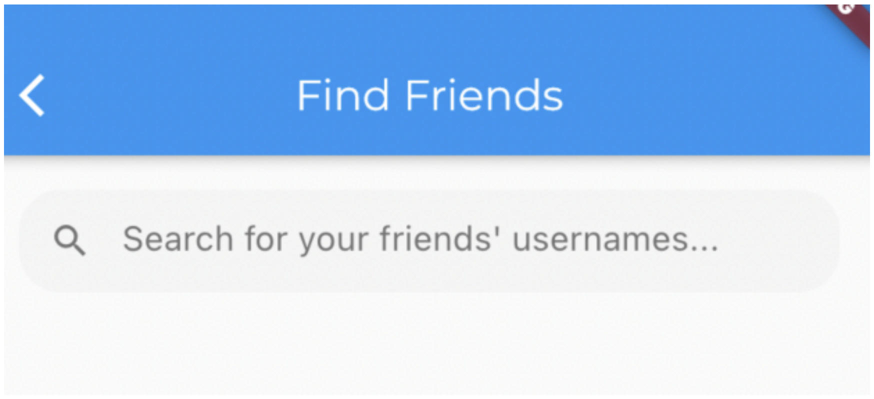
0
Day Streak

0
EXP Earned


Friends

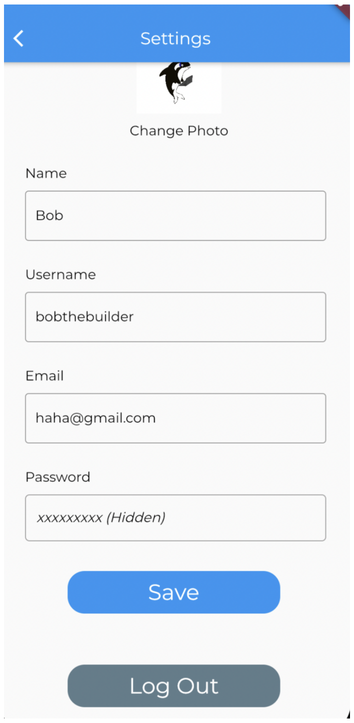
**Following**
Not following anyone yet :(

**Followers**
No followers yet :(


Find Friends

Search for your friends' usernames...

a. Display player's details (name, username, profile photo, level), stats (streak, total EXP earned) and friends with scroll view (ListView)
b. Add friends button (add friends icon) to bring users to Find Friends page (IconButton)
c. Find Friends page contains a search bar for users to search for friends to follow by their usernames

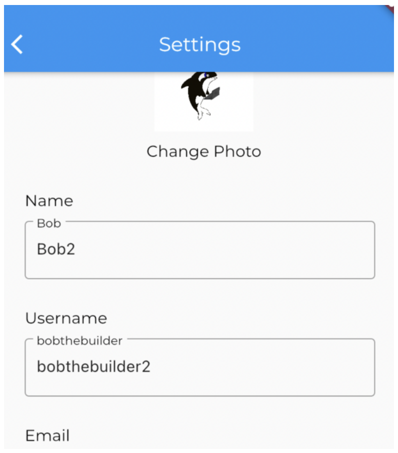


Name
Bob

Username
bobthebuilder

Email
haha@gmail.com

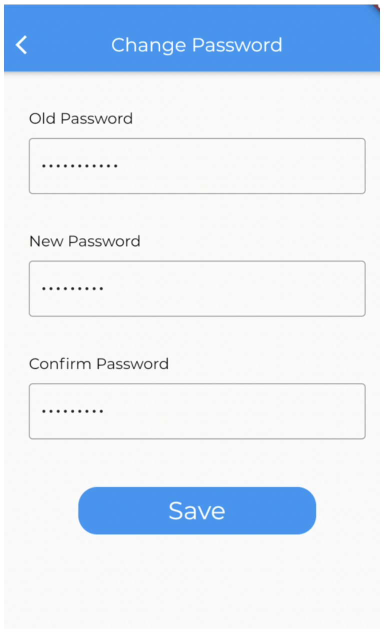
Password
*xxxxxxxxx (Hidden)*



Name
Bob
Bob2

Username
bobthebuilder
bobthebuilder2

Email



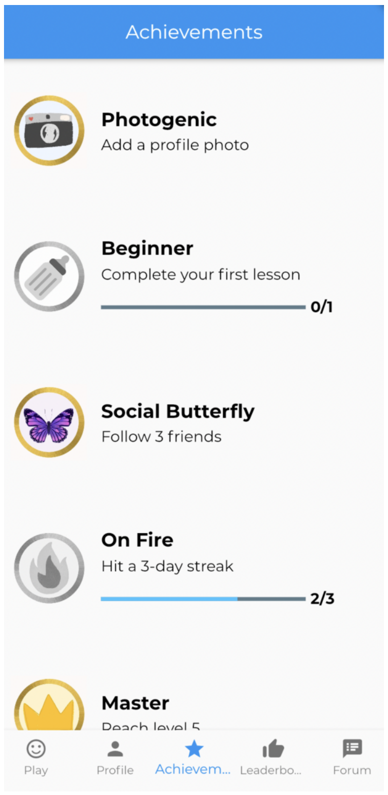
Old Password
···········

New Password
·········

Confirm Password
·········

d. Settings button (gear icon) leads to Settings page (IconButton)
   i. Displays their current details (name, username, email, password) in the text boxes with the password as hidden text (TextField, obscureText)
   ii. Allows players to change their name, username, or email by clicking and typing in the text box and clicking on the Save button (TextField, TextButton with boxDecoration)
   iii. Clicking on the password text box to change it leads to a special Change Password page (TextButton) where users can type in their new password in text boxes with hidden text (TextField, obscureText) and save the new password by clicking on the Save button (TextButton with boxDecoration)
   iv. Log Out button allows users to log out of their current account and return to the Login page (TextButton with BoxDecoration)

3. Achievements



Achievements

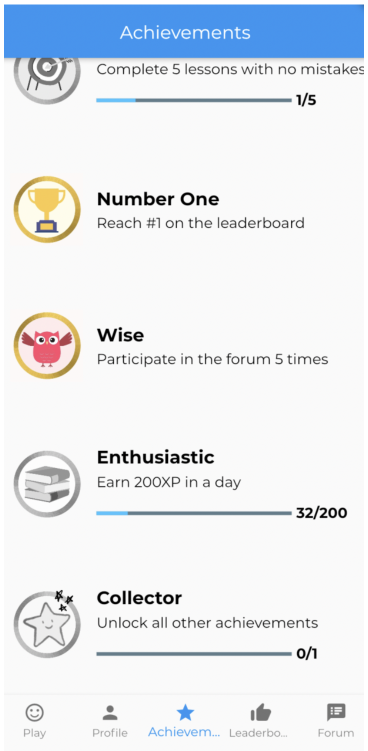
Photogenic
Add a profile photo

Beginner
Complete your first lesson
0/1

Social Butterfly
Follow 3 friends

On Fire
Hit a 3-day streak
2/3

Master
Reach level 5



Achievements

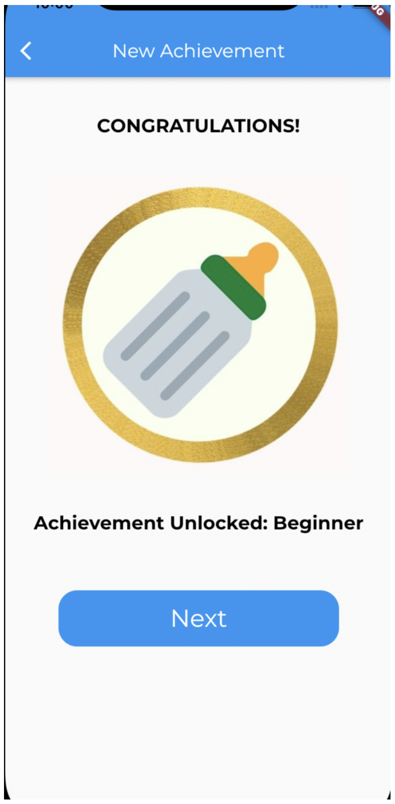
Complete 5 lessons with no mistakes
1/5

Number One
Reach #1 on the leaderboard

Wise
Participate in the forum 5 times

Enthusiastic
Earn 200XP in a day
32/200

Collector
Unlock all other achievements
0/1



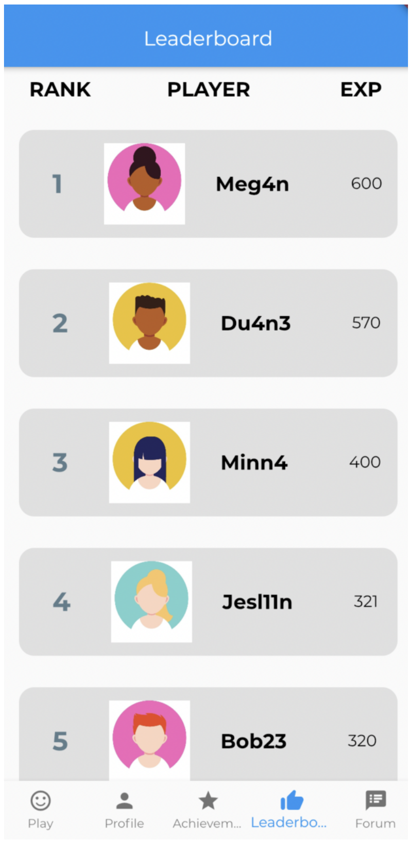
New Achievement

CONGRATULATIONS!

Achievement Unlocked: Beginner

a. All possible achievements to be attained by a player displayed with scroll view (ListView)
b. Uncompleted achievement: black and white icon, progress bar showing player’s progress towards attaining the achievement (LinearProgressIndicator)
c. Completed achievement: coloured icon, progress bar disappears (conditional statements for when the achievement is completed vs not completed yet)
d. Popup when a new achievement is attained (image shown is from when users complete level 1)

4. Leaderboard

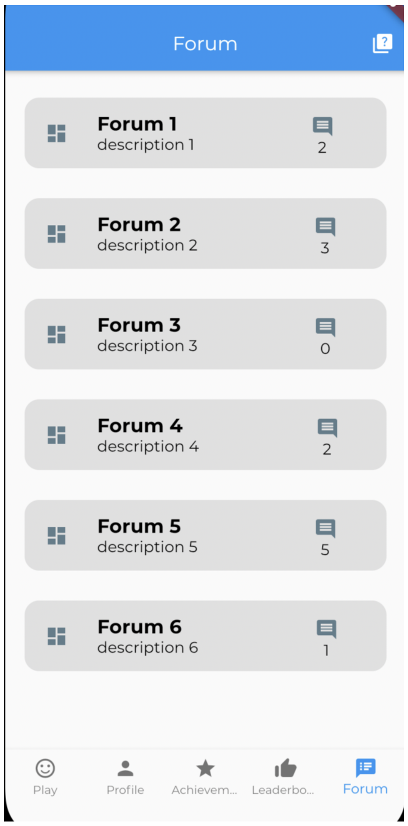


Leaderboard

| RANK | PLAYER | EXP |
|---|---|---|
| 1 | Meg4n | 600 |
| 2 | Du4n3 | 570 |
| 3 | Minn4 | 400 |
| 4 | Jesl1n | 321 |
| 5 | Bob23 | 320 |

a. Displays current top players (rank, profile photo, username, EXP) with scroll view, listed in order of EXP (ListView)

5. Forum



Forum 1
description 1
2

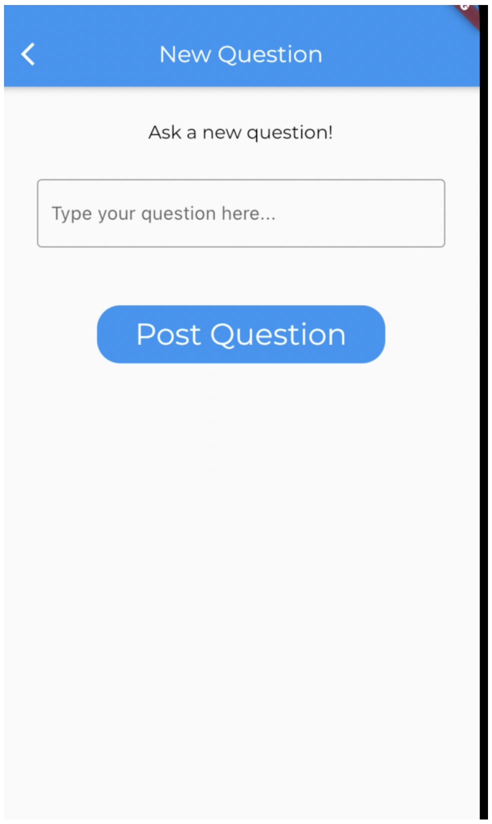
Forum 2
description 2
3

Forum 3
description 3
0

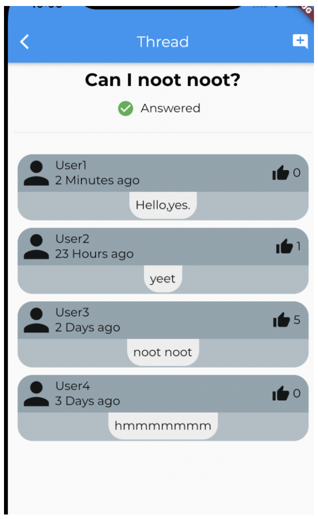
Forum 4
description 4
2

Forum 5
description 5
5

Forum 6
description 6
1



a. Displays all questions, clicking on one of them brings players to the question thread
b. Button on the top right to create a new thread by posting a question, takes player to New Question page



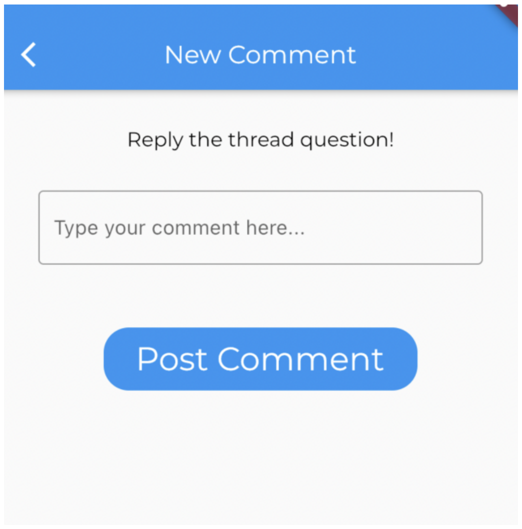
Can I noot noot?

User1
2 Minutes ago
0
Hello,yes.

User2
23 Hours ago
1
yeet

User3
2 Days ago
5
noot noot

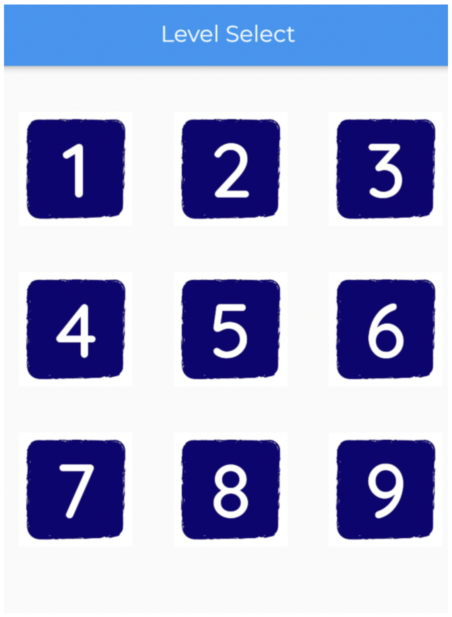
User4
3 Days ago
0
hmmmmmmmm



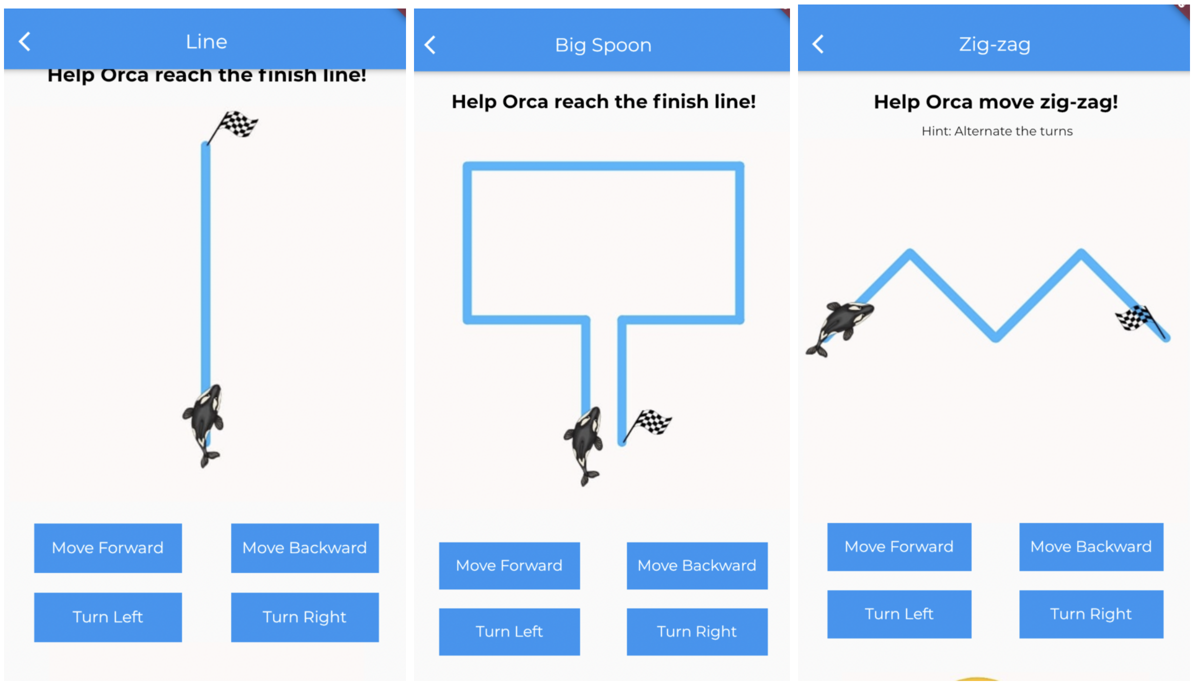
c. Question Thread
   i. Displays all comments with the number of votes/ likes
   ii. Button on the top right to post a new comment to the question thread, takes player to New Comment page

<u>Features done by Milestone 3:</u>

1. Proper game levels

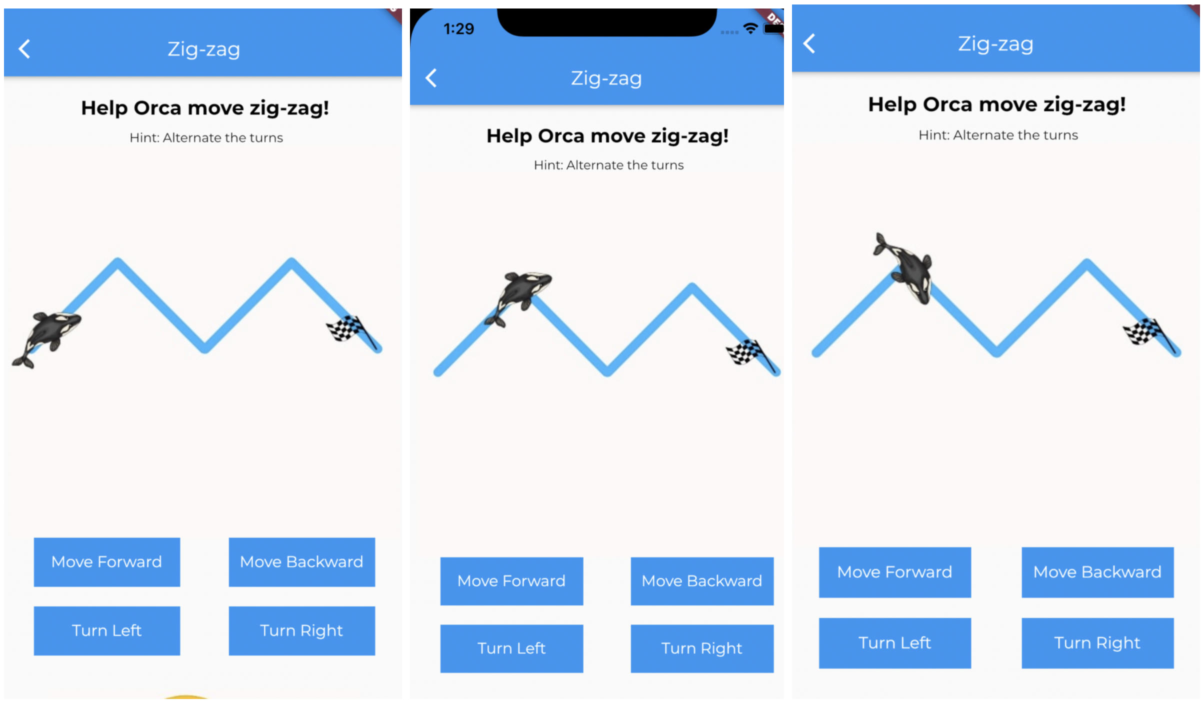


   a. Level select menu to allow users to navigate between the different levels easily by clicking on each level icon (FlatButton)

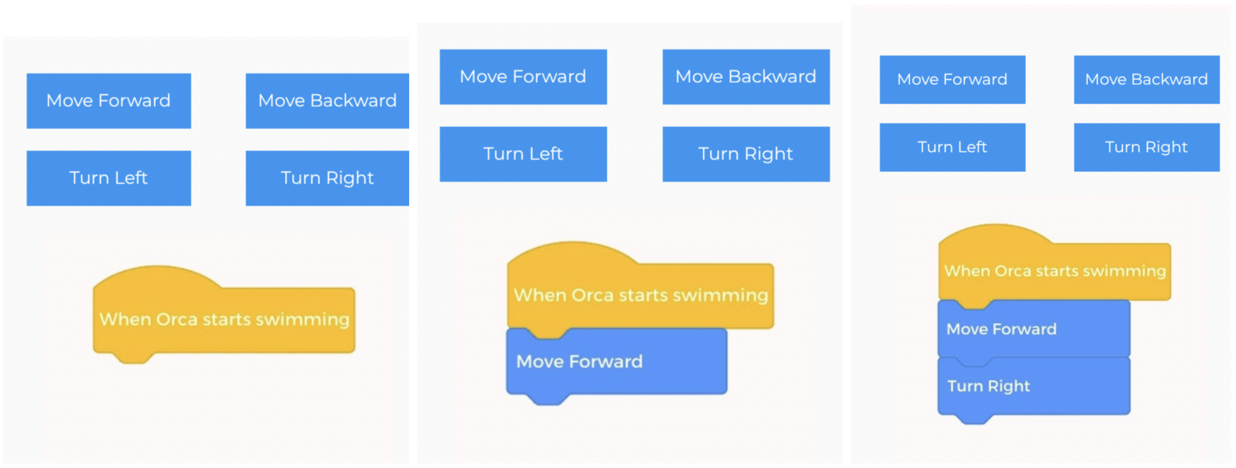


   b. Introductory game levels to begin teaching users programming basics. Each game level has the same goal of bringing the sprite from the initial point to the end point, as marked by the flag. Game levels gradually increase in difficulty (Images shown above are Levels 1, 7 and 9 in order from left to right)

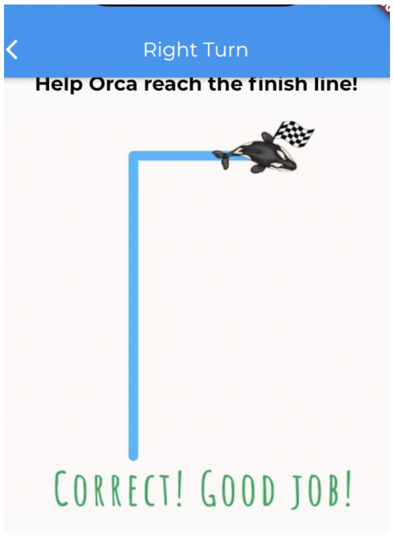
c. 4 buttons in each level (Move Forward, Turn Right, Turn Left, Move Backward) that allow users to navigate the sprite around to reach the finish line (TextButton), designed to appear more like coding blocks for users get to feel more like they are actually coding (boxDecoration)

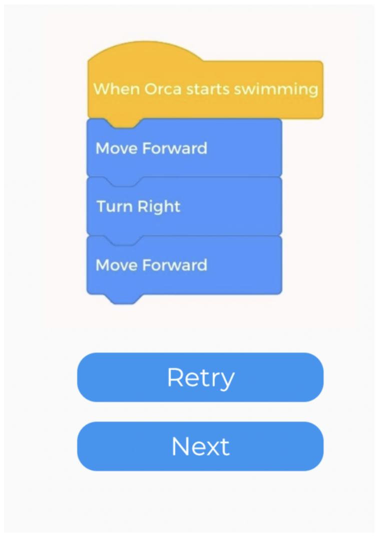


d. Sprite that moves in real time as users press the correct buttons so users can better visualise the effect of their code (constantly updating the value of a variable as users click on the different buttons and displaying the corresponding images at each variable value through conditional statements)

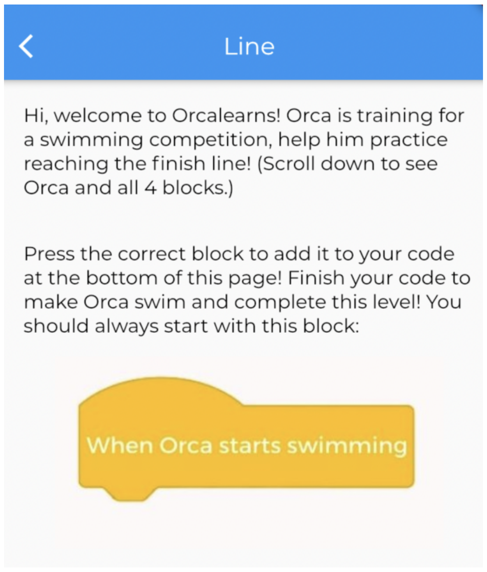


e. Blocks (similar to Scratch or mBlock) automatically built at the bottom of the screen as users click on the correct buttons, so they get to experience building real functioning programs while they complete the levels (using conditional statements to display the corresponding images depending on the value of the variable, similarly to the sprite movement)





f. Clear indication of when each level is completed so users are aware that they can move on to the next level, through the text "Correct! Good Job!" as well as Retry and Next buttons that only appear once the level is completed (conditional statements)
g. Retry button resets the level back to its initial state, while Next button brings users to the immediate next level (TextButtons with boxDecoration)



Hi, welcome to Orcalearns! Orca is training for a swimming competition, help him practice reaching the finish line! (Scroll down to see Orca and all 4 blocks.)

Press the correct block to add it to your code at the bottom of this page! Finish your code to make Orca swim and complete this level! You should always start with this block:

h. Actual teaching materials included at the start of certain levels to facilitate the learning of users as they play the game (Text)

2. Data-storage (Firebase)
   a. Prevent non-users from logging in (based on UID)
   b. Store and retrieve the profile details of each player to be displayed in the Profile page, and store any updates made through the Settings page
   c. Store and update each player's current achievement progress to be displayed on the Achievements page as the player does various tasks in the app
   d. Store forum questions, answers and likes

<u>Future developments:</u>

1. Increased game levels - more levels to further the teaching of programming. As of now, the first 9 levels implemented are mostly just introductory levels to get users familiarised with the logic and movement of sprites. Only moving and turning blocks have been introduced, as well as the starting block. Following the syllabus used to teach students in DuinoCode (where we work as part-time teachers), future levels would teach the following concepts in order:

    a. Turning in degrees - Currently all left and right turns are set to the default 90 degrees, but it would be good to teach students how to change how much their sprite turns by changing the number of degrees. It is understandable that as our target users are young children, some of them may not have yet learnt the concept of degrees in school. However, it is still worth teaching them that at least increasing and decreasing the number of degrees would make their sprite turn more or less respectively, as well as recognising important angles such as 45 degrees, 90 degrees and 180 degrees.

    b. Moving for a certain number of seconds / a certain distance - Currently moving forward or backward will move the sprite as far as we need it to move to reach the destination, but in the future we could add the element of time or distance to the Move Forward and Move Backward blocks so users could change the values on their own to control how far the sprite would move. This is more aligned to robotics in real life.

    c. Wait block - We could teach users how to use this block to make their sprite stay still for a certain number of seconds before resuming its actions.

    d. Repeat block - Instead of repeating the same code over and over again (for instance to make the sprite move in a square), users should instead learn how to use the repeat block to increase the efficiency and readability of their code. It will help save them a lot of time as well. Users should also learn to determine how many times exactly they need to repeat their code, or which parts of their code they should be repeating.

    e. Forever block - Based on our personal experience, even young children are able to grasp the concept of “forever” quite easily, so we just need to teach them that a forever block is essentially a repeat block that repeats their code infinitely as opposed to a certain number of times.

    f. If-then and If-then-else block - A very important element of actual programming, it would be very useful for users to learn the proper meaning and use of these two similar blocks. More levels would be needed to properly teach younger children about these blocks, as they tend to get quite confused between where to put their code when the “if” condition is satisfied or unsatisfied. They also find it difficult to differentiate when the If-then-else block is needed from when the If-then block is sufficient. These blocks have many potential applications as well, such as getting users to check if there is an obstacle blocking the path of their sprite and moving out of the way if this is the case.

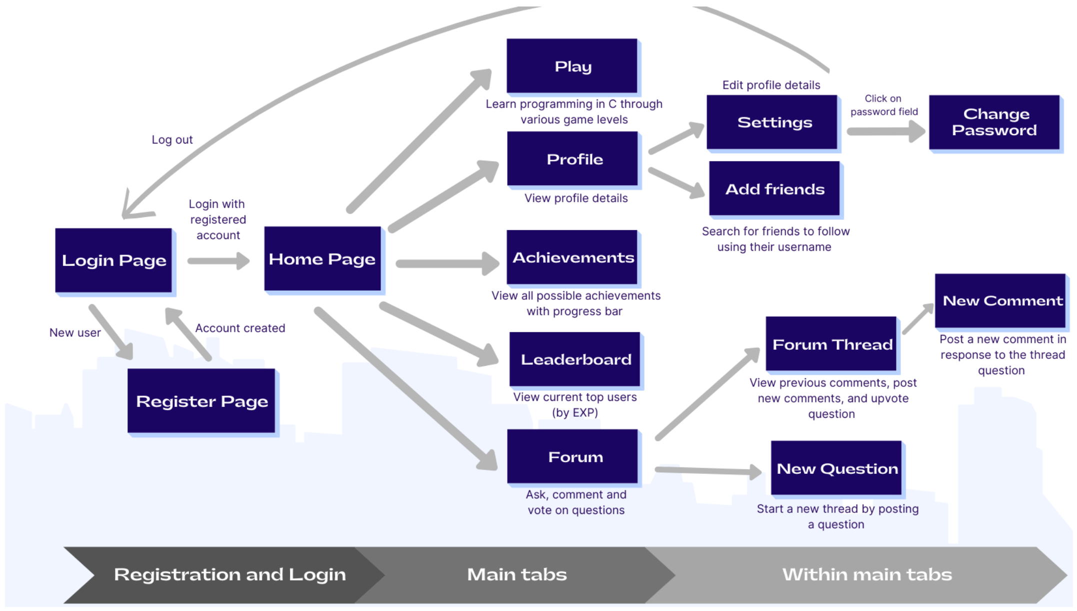
   g. Making their own variables - As users reach higher levels, they can also learn more advanced programming topics such as creating their own variables, setting their initial values, and updating the values.

   h. Making their own blocks (functions) - Even more advanced users can learn how to make their own functions in the form of blocks, and how to utilise these self-made functions in later parts of their code to further simplify their code and increase the efficiency of their programmes. As this is a rather high-level topic for young children to learn, we would have to ensure that we implement simple game levels to slowly introduce this concept to them.

   There are of course many other programming concepts that we could further add on for future levels, but these are our plans for future levels as of now.

2. Improved user profile page and friend settings - more information stored on the profile page that other users will be able to view. Increased interaction between users as they are able to track their friends' progress. This also allows for healthy competition to engage users and encourage their learning
   a. Be able to display achievements of each users on their profile page
   b. Be able to view friends' achievements and level progress
3. Improved login and register pages:
   a. Currently, logging in with registered username and email does not give rise to an error message, this will be fixed as soon as possible to ensure each user is unique.
   b. Currently, logging in with an unregistered account does not give rise to an error message, this will be fixed as soon as possible. We will still explore the possibility of guest accounts being available for use
4. Improved forum page:
   a. At the moment, the thumbs up and down features for comments and threads has not been enabled, so voting and sorting of comments by votes has not been enabled. This feature will be enabled to ensure that users are more able to view and sort and search for comments

## Program Flow


Log out
Login with registered account
Login Page
Home Page
New user
Account created
Register Page
Play
Learn programming in C through various game levels
Profile
View profile details
Achievements
View all possible achievements with progress bar
Leaderboard
View current top users (by EXP)
Forum
Ask, comment and vote on questions
Edit profile details
Settings
Click on password field
Change Password
Add friends
Search for friends to follow using their username
Forum Thread
View previous comments, post new comments, and upvote question
New Comment
Post a new comment in response to the thread question
New Question
Start a new thread by posting a question
Registration and Login
Main tabs
Within main tabs

## Testing

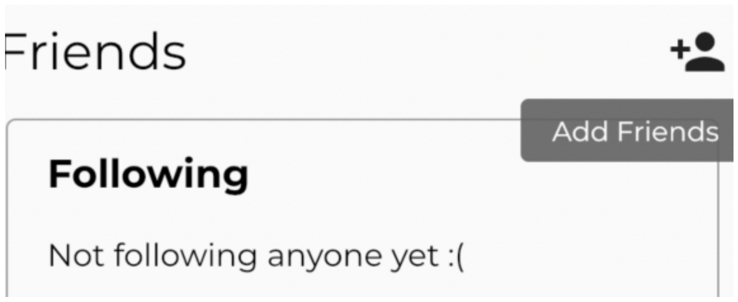
### Heuristics Testing

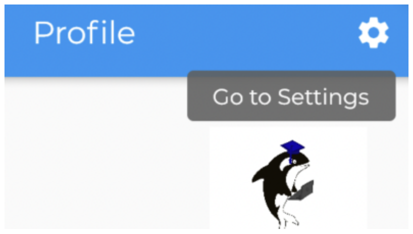
We carried out heuristics testing on some potential users (our peers, family members, as well as students) to judge the **usability** of our app for our target users and the **user experience**, based on the Nielsen Heuristics (Heuristics for User Interface Design) as developed by Jacob Nielsen in 1995:

1. Visibility of System Status (users should know the system status at all times and get feedback on interactions with it);
2. Match between system and the real world (the system should resemble the experiences that users already had);
3. User control and freedom (users should be able to reverse their action if done by mistake);
4. Consistency and standard (similar system elements should look similar)
5. Error prevention (minimize the likelihood of making mistakes);
6. Recognition rather than recall (users should be able to interact with the system without prior information or context;
7. Flexibility and efficiency of use (both new and experienced users should be able to efficiently use the system);
8. An aesthetic and minimalist design (declutter as much as possible, less is more);
9. Help users recognize, diagnose, and recover from errors (make error messages understandable, and suggest ways to fix an error);

10. Help and documentation (if a user has a hard time interacting with your app, make sure there's help that's easily accessible).

1. Regarding the System Status, users have feedbacked that there is no transition between levels, and no loading spinner icon. It would be possible to add a spinner icon in the future to showcase that the app is loading between screens. However, at the moment, it does not take any time to transition between screens, so it was not deemed a priority. It will be included in future developments.
2. The User Interface (UI) is simple enough to understand, with icons and words to describe where the icon brings you to. The icon is also intuitive for users that have tried the app. Some have said that icons for features such as the "add friends" feature might not be intuitive, as it is just a '+' icon next to the silhouette of a person. To combat this, if they long press the icon, the words "Add Friend" pops up, to inform them of what the button does.





3. Users are able to edit their profile however they wish, with custom usernames, profile pictures, and friends list. They are able to also remove friends. Therefore, there is a high level of customization for user profiles that they can make use of.

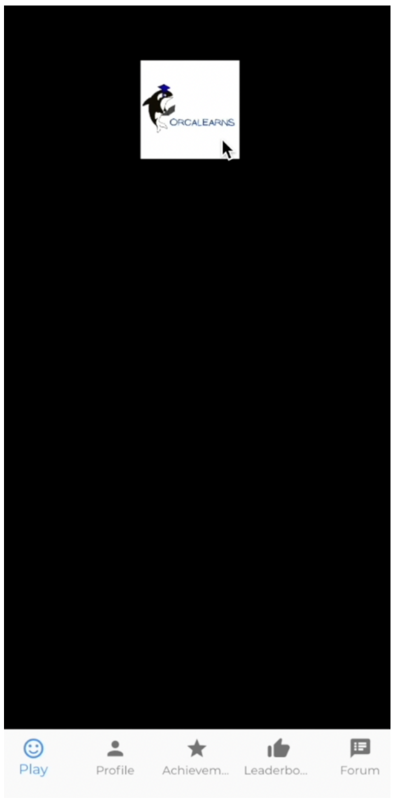
Users have feedback that they enjoy the fact that they can customise their profiles should they wish to do so.

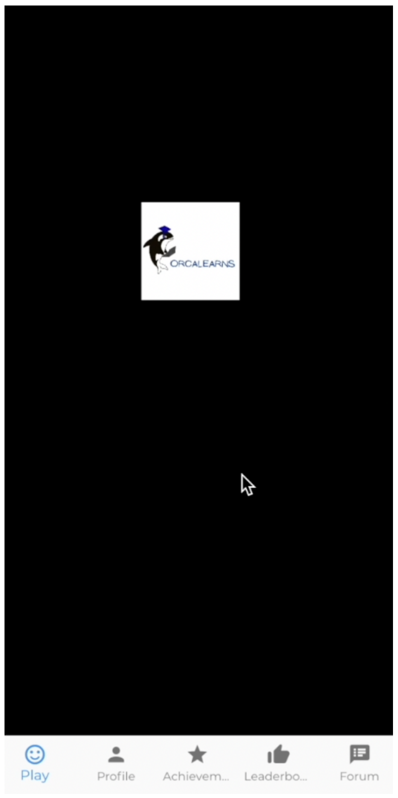
4. We have used a consistent colour scheme and font, as well as font style and size. As a result, we have not received any complaints from users regarding this matter, as there is nothing difficult to read or understand, with similar themes. However, some older users have feedback that some words are too small for them to read. We made the font slightly bigger, to the best of our abilities without hindering design. The issue has been more or less resolved as a result.
5. The user interface is simple enough to understand that we have received no feedback regarding error prevention. We believe that the design should be minimalistic to ensure that everyone understands the functions of each design.
6. Users have mentioned that everything was intuitive and easy to understand. Refer to points 2. and 5.
7. Our users were all new users who understood fundamentally what each tab in the main UI does, even without prompting. Therefore, we believe that users are able to properly interact with the system without prior information or context.
8. Users have told us that our design is easy to understand and easy on the eyes as well, there were no complaints regarding clutter and aesthetics.
9. The levels are simple enough to complete, as they are meant for young students and entry-level learners to learn basic programming methodology. As a result, the levels are relatively easy to complete, with the levels resetting each time there's a mistake. Therefore, it is easy for users to recognise exactly which step they failed to do correctly. Furthermore, no users have encountered an error message. With that being said, we do have a catcherror function in our code that displays the error message in the event that one occurs, so we believe this to be a non-issue.
10. At the moment, there is no tutorial, as we believe that the levels are already in the form of a tutorial, with each level being only slightly more difficult than the previous to aid in step-by-step learning. Users have not mentioned any troubles with using the application, and as a result there is no help section as of yet. However, when introducing new mechanics in levels in the future, we will inform users of how to make use of said mechanic, which we believe can aid in helping the users learn better.

## Testing with Potential Users

Using the same group of people we implemented our heuristics testing with, we also asked them to help us evaluate how **useful** our game was in accomplishing our aim of teaching young children programming as well as how **engaging** our game was, as the game is the important feature of our application so we felt that it required extra attention and testing.

This was the first iteration of our game (no input needed to go from the first image to the second):



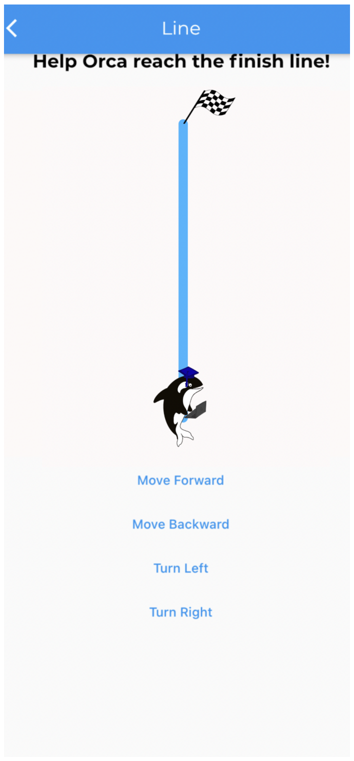


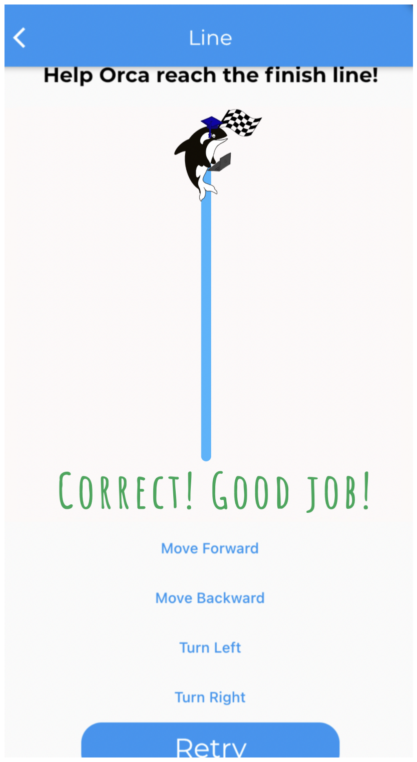
Our first game level initially had a sprite that automatically moved up and down the screen. Players could click on any part of the screen to stop the sprite from moving. Clicking on the screen again would make the sprite start moving once more. Our intention was to first familiarise players with simple movements such as up, down, left and right to begin their programming learning.

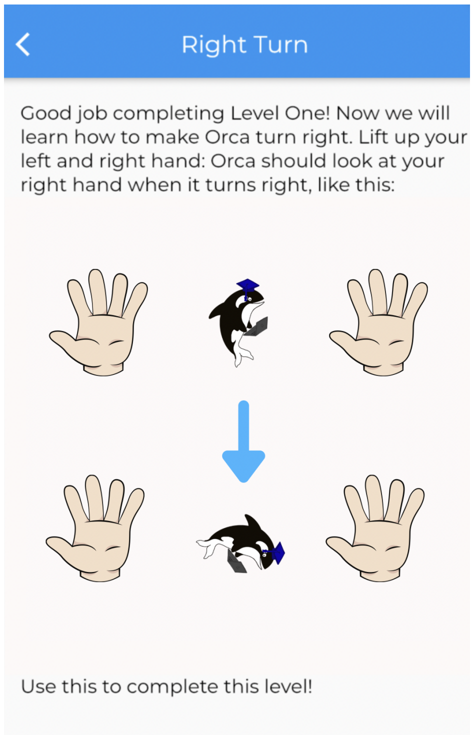
However, the feedback we received from many was that they could not see how this level would help to teach programming in any way, as it was not obvious what we were trying to show exactly through the moving sprite. It did not really feel like a game level either, as there was no clear objective other than just clicking on the screen and observing the movement of the sprite. It might not be engaging to young children using our app.

Taking this input into account, we decided to revamp our game levels entirely. We scrapped the idea of having automatically moving sprites and changed it so the sprite would only move when players clicked on the correct buttons. We also designed each level to have a goal of bringing the sprite from the starting point to the end point so there was a clear objective for players to accomplish, which would hopefully make our app more interesting for players.

Our second iteration of the game looked like this (the “Move Forward” button was pressed to get from the first image to the second, third image is the start of Level 2):







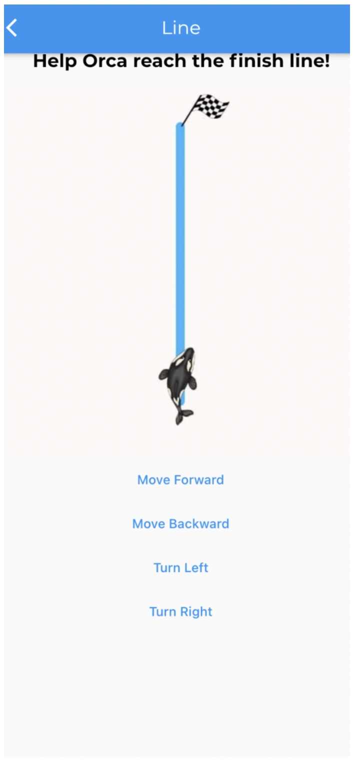
Good job completing Level One! Now we will learn how to make Orca turn right. Lift up your left and right hand: Orca should look at your right hand when it turns right, like this:

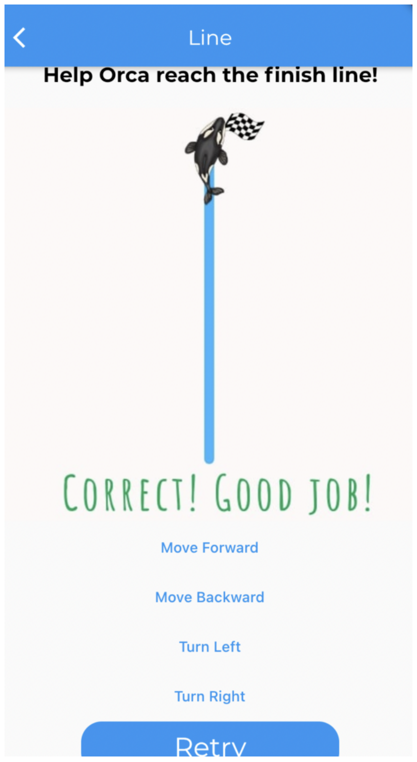
Use this to complete this level!

This version of the game received more positive reviews. With a clear objective to clear the level and an indication of when players had completed the level, it was said that this version felt more like an actual game. The short notes at the beginning of certain levels (such as in the third image) also helped to teach our players as they played the game.

However, we also received feedback that it was rather ambiguous which direction the sprite would move in. This is because the orca we used (from our logo) was side-facing to the right. Therefore, technically clicking on “Move Forward” should make the sprite move to the right of the screen as opposed to upwards as we intended. This would lead to undesired confusion for our players.

Therefore, we made the unfortunate decision to use another different front-facing orca as our sprite for the game levels instead of our Orcalearns logo orca, to avoid any confusion. This was our third game iteration (the “Move Forward” button was pressed to get from the first image to the second):

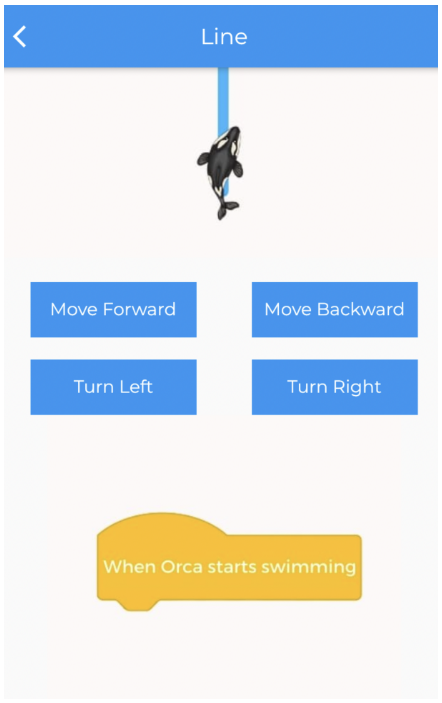


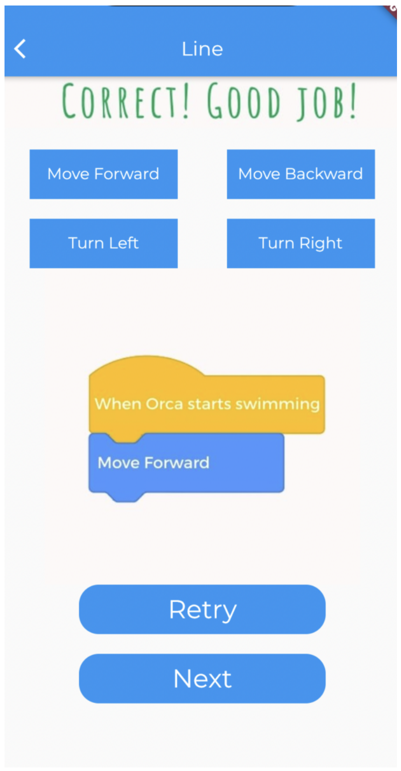


The feedback we received for this version was mostly that the game levels were good. The sprite on the screen helped players visualise the movements the sprite would take as they pressed the correct buttons, and helped to make the game more appealing to children. There was no ambiguity in the direction the sprite would move in with this orca.

However, some indicated that the link between our game and teaching programming was still unclear; it was not apparent how teaching players to navigate their sprite to the end point using the given buttons was explicitly programming. They stated that though our game was educational, it seemed to teach more logic as compared to actual coding. They suggested incorporating actual blocks into each level so players could at least see their code being built as if they were using online platforms such as Scratch or mBlock.

Following this advice, our final iteration of the game looked like this (the “Move Forward” button was pressed to get from the first image to the second):





In this version, the buttons themselves were made to look more like the blocks that players should add to their code to complete the level, and clicking on the correct block would automatically add the block to their code below. In this way, it would feel to the player as if they are clicking on the blocks to add it to their own code. At the end of each level, the player would have finished building their own code to bring the sprite from the start point to the end point, as shown in the second image above.

The common consensus was that this version made the link between our game and teaching programming basics more clear, as it was as if players were actually building their own code as they completed levels of the game. This was the final version of the game we decided to go with.

## Self/ Manual Testing

We also carried out self/ manual testing on our application to check for the **correctness** of our application, or in other words any problems with the implemented features. Below is the full testing documentation showing all the features we have tested along with whether or not they behaved as expected. If not, we included a description of the problem as well as how we fixed it.

*Full Testing Documentation:*

| Feature Tested | Passed? (Description of issue and solution if test failed) |
|---|---|
| LOGIN | |
| Users can type in their email and password in the text boxes | Yes |
| Clicking on the text box changes the phone keyboard to the email address keyboard | Yes |
| Text is censored when typing in the password text box | Yes |
| Clicking on “Sign up” brings users to the Signup page | Yes |
| Clicking on “Forgot Password?” allows users to reset their password through their email | Yes |
| Logging in with a registered account works when Login button is pressed | Yes |
| Logging in with an unregistered account gives rise to an error message when Login button is pressed | No. Lack of time |
| App logo is displayed | Yes |
| Proper navigation between Login page and Register page | No. Clicking “Sign up” on Login page and “Sign in”/Register button on Register page one after the other creates more and more Login and Register pages, adding more pages to the current route. Pressing the back button multiple times then brings users back through many alternating Login and Register pages<br><br>Solution: Navigator.pushReplacement changed to Navigator.pop for both buttons on Register page so the initial page of the app is the Login page and the Register page only allows navigation back to this initial page |
| No backward navigation from Login page | No. Back button appears when Login page is accessed through Log Out button in Settings page, and brings users back to the Settings page when it is clicked, therefore returning them to their previously signed-in account<br><br>Solution: automaticallyImplyLeading set to false for the Login page app bar so the appBar never displays any back button |

| REGISTER | |
|---|---|
| Users can type in their details (name, username, email and password) in the text boxes | Yes |
| Clicking on the text box changes the phone keyboard to the email address keyboard | Yes |
| Text is censored when typing in the password text box | Yes |
| Clicking on “Sign in” brings users to the Sign in page | Yes |
| Registering an account saves the new user details onto firebase when the Register button is pressed | Yes |
| Registering an account with an email address that is already registered in the database gives rise to an error message | No. Lack of time |
| Registering an account with a username that is already used by another user gives rise to an error message | No. Lack of time |
| Clicking on the back button returns user back to the initial Login page | Yes |
| GAME | |
| Clicking the level icon brings users to the corresponding level | Yes |
| Back button on every level leads back to level select menu | No. Back button led back to the previous level if the current level is accessed through the Next button in the previous level<br><br>Solution: Navigator.push changed to Navigator.pushReplacement for Next button in every level |
| Retry button resets level to initial state without creating the page again | Yes |

| | |
|---|---|
| Next button brings user to the next level | Yes |
| Retry and Next buttons only appear after a user has completed each level | Yes |
| Pressing the correct navigation buttons in each level updates the orca on the screen (moves in accordingly) | Yes |
| Pressing the wrong button resets the level back to its initial state without creating the page again | Yes |
| Next button brings user back to level select menu for the final level only | No. Next button brought user back to level select menu by creating a new level select menu page, not the original page on the home screen (meaning the bottom navigation bar was missing and users had to press the back button once to return to the real level select menu)<br><br>Solution: Navigator.pushReplacement changed to Navigator.pop for Next button of last level only |
| Level 1 works as expected | Yes |
| Level 2 works as expected | Yes |
| Level 3 works as expected | No. Though the level clearly indicates that the sprite should take right turns to move in a square (as per our intention and seen from the image on the screen), clicking “Turn Right” is registered as a mistake. Instead, clicking “Turn Left” is seen as correct. The sprite moves properly (taking right turns) and the blocks also appear correctly (with “Turn Right” blocks included) when we complete the level by replacing every instance of “Turn Right” with “Turn Left”<br><br>Solution: This problem suggested that the codes for the “Turn Right” and “Turn Left” buttons in our code were switched, so we just switched them back |
| Level 4 works as expected | Yes |
| Level 5 works as expected | Yes |
| Level 6 works as expected | Yes |
| Level 7 works as expected | Yes |
| Level 8 works as expected | Yes |

| | |
|---|---|
| Level 9 works as expected | No. The block code does not correspond to the current state of the Orca on the screen. For instance, when the level first starts the block “Move Forward” is already present even before the user has clicked on the “Move Forward” button<br><br>Solution: Our code was correct, but we had forgotten to save the first image that was supposed to be displayed, resulting in the level always displaying the next image instead of the current intended image. We added the missing image to our app Assets file accordingly |
| PROFILE | |
| User’s name, username, level, display picture displayed | Yes |
| User’s stats(streak days, total EXP earned) displayed | Yes |
| User’s list of followed users shown | Yes |
| User’s list of followers shown | Yes |
| Clicking on the add friends icon leads to Find Friends page | Yes |
| Find Friends page: search bar allows users to search for other users to follow using their usernames | Yes |
| Saving new password saves the updated password into the database and returns user to the previous screen (Settings) | No. User is returned to Settings screen by creating a new Settings page, adds more pages to current route so back button does not work properly (returns back to Password Change page)<br><br>Solution: Navigator.pushReplacement changed to Navigator.pop for Save button on Change Password page |
| Clicking on the gear icon on the Profile page brings users to Settings page | Yes |
| Saving changes to settings saves the updated settings into the database and returns user to the previous screen (Profile) | No. User is returned to Profile screen by creating a new Profile page, adds more pages to current route so back button does not work properly (returns back to Settings page)<br><br>Solution: Navigator.pushReplacement changed to Navigator.pop for Save button on Settings page |
| Clicking on the name, username or email boxes allows users to type in | Yes |

| | |
|---|---|
| their new details | |
| Clicking on the password box brings users to a special Change Password page | Yes |
| Log Out button is the only way users can return to the Login page once they are signed into their account, otherwise they are automatically logged in each time they launch the app | No. Back buttons at each tab on the Home page lead the user back to the Login page<br><br>Solution: Set automaticallyImplyLeading of the appBar to false for all 5 of the Home page tabs |
| ACHIEVEMENTS | |
| All achievements are displayed with their name, description and icon regardless of whether or not they have been completed | Yes |
| Uncompleted achievements have black and white icons | Yes |
| Progress bar showing the progress of the user towards completing each uncompleted achievement is present | Yes |
| Completed achievements have coloured icons | Yes |
| Progress bar disappears after an achievement is completed | Yes |
| Achievement progress is accurately updated as user completes various tasks in other parts of the app | Yes |
| LEADERBOARD | |
| Top 10 users by total EXP are displayed | Yes |
| Users are displayed with their rank, display picture, username and total EXP | Yes |
| Leaderboard is accurately updated | Yes |

| | |
|---|---|
| as users gain EXP | |
| FORUM | |
| All submitted questions displayed along with their respective number of comments: | Yes |
| Clicking on the new question icon leads to New Question page | Yes |
| Users can type in their question in a text box | Yes |
| Post Question button posts the new question onto the Forum page and brings user back to the Forum page | No. User is returned to the Forum page by creating a new Forum page instead of returning to the forum tab on the Home page (meaning the bottom navigation bar was missing), and the back button was no longer present so users were unable to navigate out of the new Forum page<br><br>Solution: Navigator.pushReplacement changed to Navigator.pop for Post Question button |
| Clicking on one of the questions posted on the forum brings users to the corresponding forum thread | Yes |
| Forum question displayed on forum thread | Yes |
| Whether or not question has been answered at least once shown on question thread | Yes |
| All responses to the thread question shown on question thread along with the responder display photos, usernames and time comment was posted | Yes |
| Number of likes/votes for each response shown on question thread | Yes |
| Clicking on the thumbs up icon adds a like/vote to the comment on question thread | No. Lack of time |
| Clicking on the thumbs up icon for | No. Lack of time |

| | |
|---|---|
| a second time removes the like/vote from the comment | |
| Clicking on the add comment icon brings users to New Comment page | Yes |
| Users can type in their comment in a text box | Yes |
| Post Comment button posts the new comment onto the current thread and brings user back to the current Thread page | No. User is returned to the Thread page by creating a new Thread page instead of returning to the previously accessed thread, adds more pages to current route so back button does not work properly (returns back to New Comment page)<br><br>Solution: Navigator.pushReplacement changed to Navigator.pop for Post Comment button |

Main Problems & Solutions

1. Navigation - the biggest problem we found throughout our app during our testing was the navigation between the different pages of the app. We had only used Navigator.push to move from one page to the next, which meant that each time we wanted to move from one page to the next, a brand new page would be created. This would add more and more pages to the current thread, making the back button behave unexpectedly (more details in the table above). This is consistent with what we reported in our Milestone 2 README.

   SOLUTION: We did some research and found other functions such as Navigator.pushReplacement or Navigator.pop that were more suited for our app, and made changes accordingly. Now the back button only allows users to navigate backwards to pages that make sense logically.

2. Engagingness of our game - We found that our initial implementation of the game was not appealing to our target users of children as it did not feel like a real game, so we had to brainstorm on how to improve our game so children would actually want to play it.

   SOLUTION: We decided that we had to add explicit goals to each level so players would actually feel like they were completing objectives in a game, as the main problem was that players did not really know what they were supposed to do in our game or what the point of our level was. We ended up remaking our game implementation completely. More details can be found in the Testing With Potential Users section.

3. Usefulness of our game in teaching programming - Through our testing, we found that our initial implementation of the game did not seem to teach any programming. Simply starting and stopping the automatic up and down movement of a sprite on the screen did not seem to be related to programming in any way.

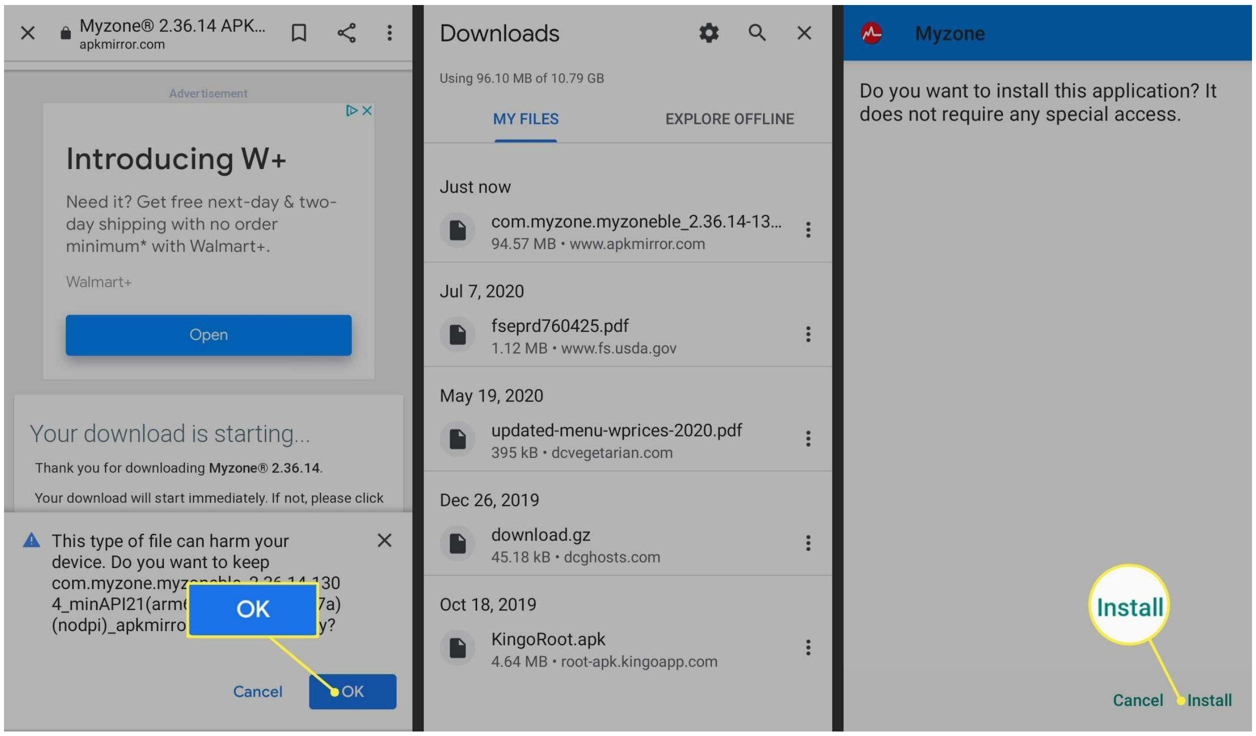
   SOLUTION: We went through many iterations of our game in our efforts to make our game more educational in terms of programming. At first we revamped the entire game so players could click on the correct buttons to move the sprite on the screen as required, instead of the sprite moving on its own. Then we incorporated blocks into our levels so players would feel more as if they were building actual code (such as with Scratch and mBlock) as they completed the levels. More details can be found in the Testing With Potential Users section.

## How to Install and Run Our App

*Click here to download our apk file!*

On Android Devices (recommended):

1. Download our apk file onto your phone using the link above.
2. If your phone does not automatically give you the option to open the file after downloading, you can access and open the apk file through the Downloads folder on your device.
3. Install the app using package installer (If the app requires any permissions, please allow them all).
4. Our app should appear together with all your other device apps. Guiding images (obtained from lifewire.com) are shown below.


Myzone® 2.36.14 APK...
apkmirror.com
Introducing W+
Open
Your download is starting...
This type of file can harm your device. Do you want to keep com.myzone.myzoneble_2.36.14-1304_minAPI21(arm64-v8a)(nodpi)_apkmirror.com.apk anyway?
OK
Cancel
Downloads
MY FILES
EXPLORE OFFLINE
com.myzone.myzoneble_2.36.14-13...
Myzone
Do you want to install this application? It does not require any special access.
Install
Cancel

<u>On Desktop</u>

1. Install and set up Android Studio onto your device.
2. Boot up your virtual device (an Android phone).
3. Download our apk file from the link above and drop the file directly into the *tools* folder in your SDK directory.
4. Type "adb install filename.apk" on the command prompt.
5. Our app should appear with all the other apps on the virtual device.

## Project Log

| S/N | Task | Date | Orbitee 1 Felicia Ivane Pranoto (hrs) | Orbitee 2 Aaron Lee Wei Qi (hrs) | Remarks |
|---|---|---|---|---|---|
| 1 | Liftoff Day 1 | 09/05/2021 | 2 | 2 | 1) First meeting with adviser<br>2) Discussion |

| | | | | | |
|---|---|---|---|---|---|
| 2 | Logo creation | 10/05/2021 | 5 | 5 | Drew the Orcalearns logo |
| 4 | Liftoff Day 2 | 12/05/2021 | 8 | 8 | 1) Work on poster submission<br>2) Work on video submission |
| 5 | Team meeting and initial planning | 13/05/2021 | 4 | 4 | 1) Planning on the overall project<br>2) Split tasks |
| 6 | Team meeting: Implementation details and initial project structure | 14/05/2021 | 5 | 5 | Discussed details of UI (such as appearance and features for users) |
| 7 | Self learning Flutter and Dart | 15/05/2021 - 25/05/2021 | 20 | 20 | Install and learn flutter online |
| 8 | Programming at home: Application UI | 26/05/2021 | 5 | 5 | Created Login page |
| 9 | Programming at home: Application UI | 27/05/2021 | 5 | 5 | Created Home page with bottom navigation bar |
| 10 | Programming at home: Application UI | 28/05/2021 | 2 | 2 | Linked Login page and Home page |
| 11 | Mission Control | 29/05/2021 | 2 | 2 | Attended the Flutter + Dart workshop in Mission Control #3 |
| 12 | Team meeting | 30/05/2021 | 5 | 5 | Worked on Milestone #1 |
| 13 | Team meeting | 05/06/2021 | 2 | 2 | Evaluated other teams |
| 14 | Team meeting | 06/06/2021 | 3 | 3 | Went through our peer evaluations, discussed improvements to make to our app and README for Milestone 2 |
| 15 | Programming at home: Application UI | 10/06/2021 | 8 | 0 | Created Achievements page |
| 16 | Programming at home: Application UI | 14/06/2021 | 10 | 0 | Created Profile page (including Settings page and Change Passwords page) |
| 17 | Programming at home: Application UI | 18/06/2021 | 6 | 0 | Created Forum |

| | | | | | |
|---|---|---|---|---|---|
| 18 | Programming at home: Application UI | 20/06/2021 | 1 | 0 | Created Register page |
| 19 | Programming at home: Data Storage | 21/06/2021 | 2 | 7 | Linked Login and Register pages to a database to store user accounts |
| 20 | Programming at home: Game | 12/06/2021 - 26/06/2021 | 0 | 20 | Created first few game levels |
| 21 | Programming at home: Application UI | 23/06/2021 | 4 | 0 | Created Leaderboard page |
| 22 | Team meeting | 28/06/2021 | 4 | 4 | Worked on Milestone 2 |
| 23 | Team meeting | 01/07/2021 | 2 | 2 | Evaluated other teams |
| 24 | Team meeting | 05/07/2021 | 3 | 3 | Went through our peer evaluations, discussed improvements to make to our app and README for Milestone 3 |
| 25 | Programming at home: New game levels (inc testing with potential users) | 07/07/2021 - 23/07/2021 | 30 | 0 | Reworked the game levels with feedback from potential users |
| 26 | Programming at home: Testing and bug fixing | 12/07/2021 - 17/07/2021 | 8 | 0 | Self testing and fixing application navigation |
| 27 | Programming at home: Achievements database | 08/07/2021 - 18/07/2021 | 0 | 20 | Include achievements and store in database for each user |
| 28 | Programming at home: Forum database | 19/07/2021 - 24/07/2021 | 0 | 10 | Attempt to include forum database for each user |
| 29 | Team Meetings | 22/07/2021 - 26/07/2021 | 14 | 14 | Worked on Milestone 3 |

| Total Hours | Orbittee 1 | Orbittee 2 |
|---|---|---|
| 202 | 161 | 149 |