BetaAlpha README

National University of Singapore

Orbital Project 2023

Done by:
Chan Hon Boon Timothy
Chew Zi Xuan

Proposed level of achievement: Apollo

Foreword by Developers:

A good day to all our readers. Welcome to BetaAlpha, an application which allows climbers to find out their climbing mistakes through video analysis and helps them improve within minutes!

Our application involves the user uploading a video of him/herself attempting the boulder problem and the application aims to help **analyse** the user's attempt at the problem. This will be based on breaking down the user's body positioning, centre of gravity as well as using vectors to model the users' direction of movement on the wall. As it is also common practice for climbers who successfully completed the boulder problem to upload their video of successful attempts on social media, the user can therefore **compare** his/her attempt with other successful climbers similar to his/her physical proportions to learn about ways to improve and do better for the next attempt by simply **running their videos through our app's built in analysis** as well.

Furthermore, sending a problem does not fully indicate an improvement in the user's bouldering ability as boulder problems can be tweaked slightly which will demand a completely different beta to send the problem. Thus our app also aims to **fundamentally improve the user's bouldering ability** by **keeping track of the user's mistakes and generating a training programme for the user**. The user can then **identify his weaknesses**, be it pulling strength, finger strength, technique or flexibility.

Motivation

Both of us are avid climbers who are constantly searching for ways to improve. One of them being, looking at online sources/videos for inspiration and guidance.

Rock climbing tests a climber's strength, endurance, agility and balance along with mental control. In rock climbing/bouldering, the main objective is to complete the problem (bouldering route) by manipulating our body from the starting tile (climbing hold) and to touch the end hold with both hands for 3 whole seconds. There are multiple beta(s) (solutions) to send (complete) the bouldering route.

It is common knowledge within the climbing community that every movement on the wall is complex and one must utilise/engage the correct muscles for a particular move, position their body at an appropriate angle to keep on the wall and many more technicalities that come with it.

When we first started out climbing, we noticed that we were desperate to improve but there was no proper guidance for us. Even if we were to go online for beta videos, we were still unclear on how to pinpoint our mistakes. As such, our climbing progress was stagnant for a period of time.

Furthermore, we realised that climbing is a relatively expensive sport and if one was to approach private coaches for guidance, this hobby of theirs might not be affordable.

With us being interested in both technology and climbing, we decided to find a way to incorporate them together.

User Stories

As a beginner who is starting out climbing alone and wishes to improve, I want to be able to get detailed guiding steps on how to do so.

As a general climber looking for advice on how to complete the route I have been working on for a while.

As a competitive climber training for bouldering competitions, I want to be able to learn about my bad habits and minimise them during competitions.

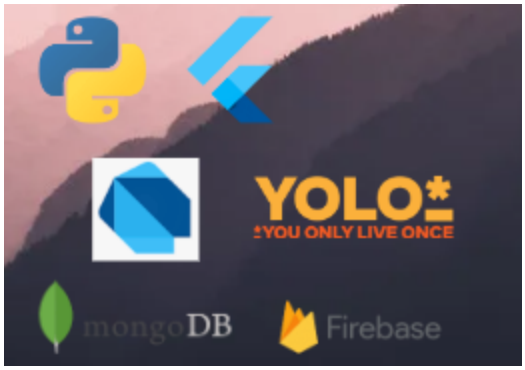
As a relatively shorter climber, I want to be able to learn how to properly generate power to reach bouldering holds that are far away.

As a relatively taller climber, I want to properly utilise my height and wingspan and pay attention to my centre of gravity.

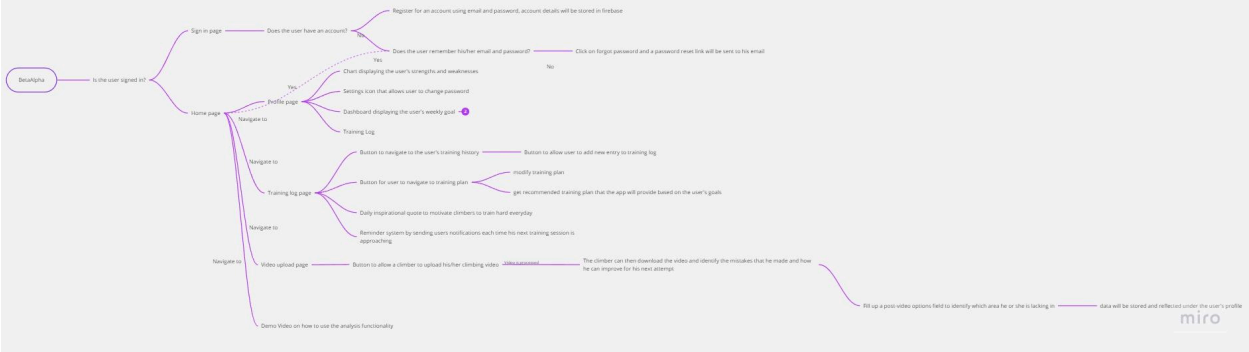
Tech Stack

To develop our mobile application, we have chosen to use Flutter, a cross application development framework of Dart, for our front-end. In order to authenticate users and allow them to create an account for a personalised experience, we chose to use Firebase as our backend.

Our application provides human pose estimation through the python library OpenCV, a computer vision library that is popular because of its simplicity and code readability.



Features/Application Framework/System Design



https://miro.com/app/board/o9J_l8XLDd4=/?share_link_id=313262816327

MongoDB: Store video file locations
Firebase: Store user's details
Backend: Flask
ML model: OpenPose CV
Frontend: Flutter
Deployment: Heroku

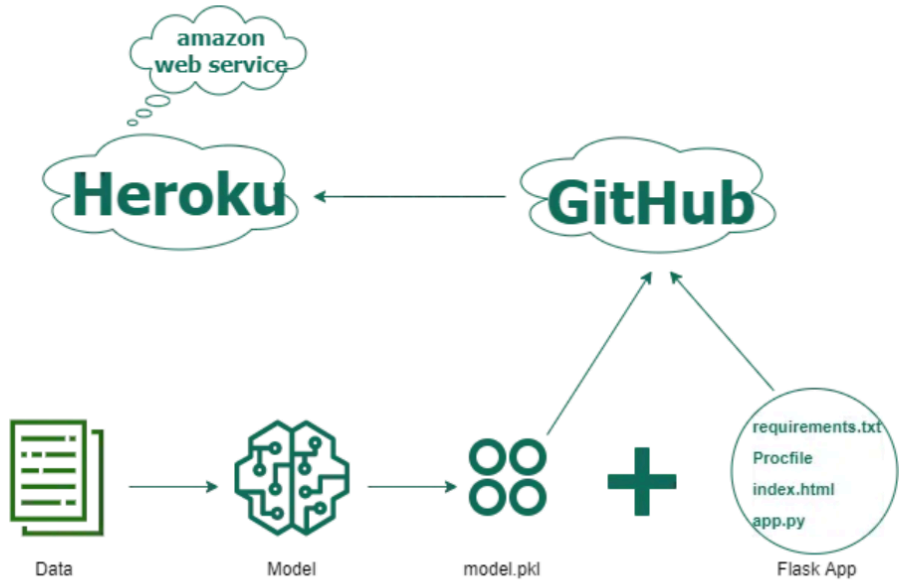
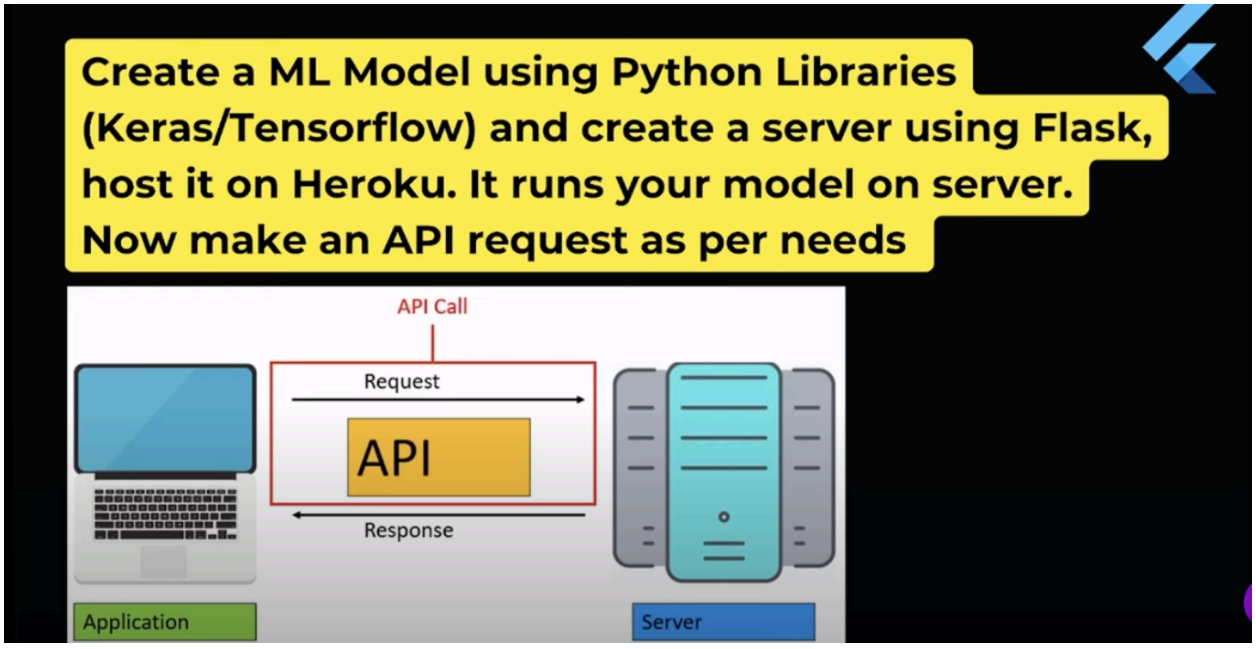
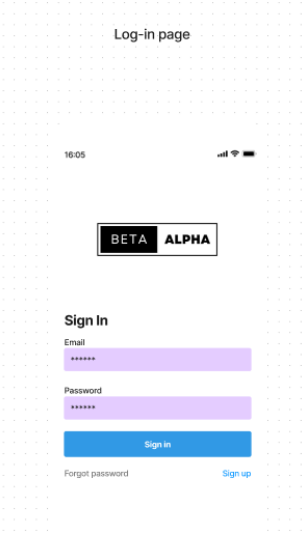
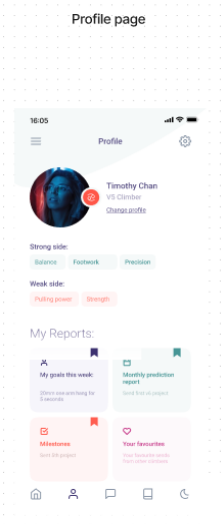


Image by Author

. . .

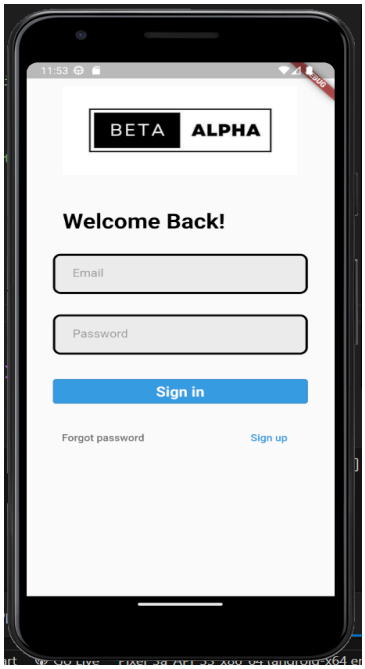


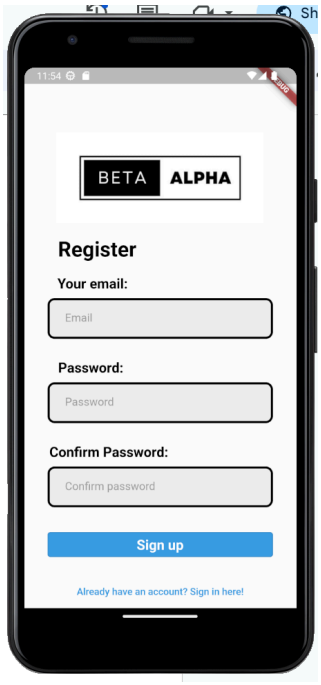


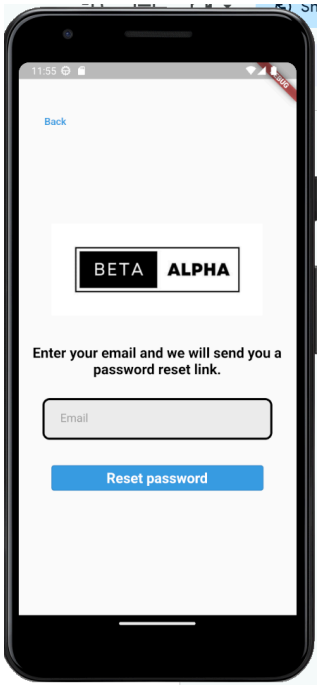


App Design and functionalities

Log in/Sign up Page:





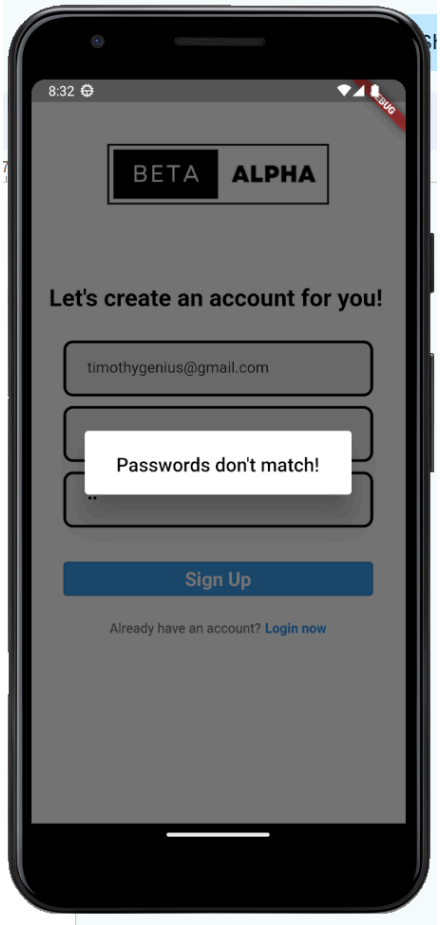


Sign In Page Sign up Page Reset password Page

Upon entering BetaAlpha for the first time, the user will arrive at the login page. Here, the user has 3 options:

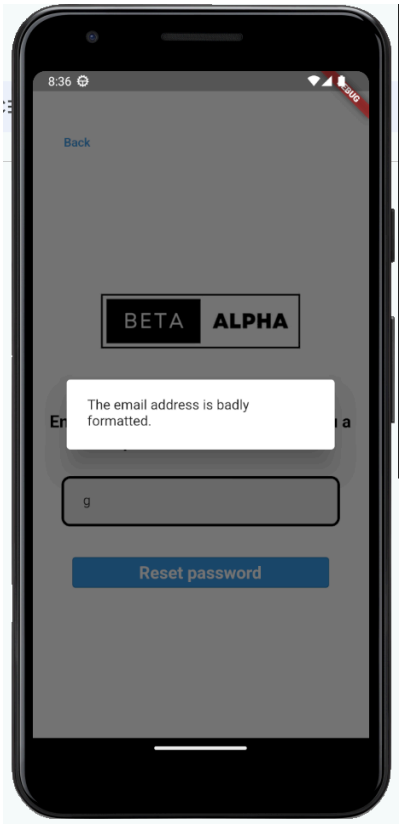
1) Forgot password: clicking on this button will bring the user to another page with a single textfield. After entering his email and pressing the Reset password button, the user will then receive an email with a password reset link to reset his password.
2) Sign up: If the user is new and has not created an account before, he will be brought to another page with 3 fields to enter. His email, a password that he wants, and a password confirm field to ensure that the user types in the password that he wants to use for his account.
3) Sign in: If the user has an existing account. He simply needs to input his email and the correct password and press sign in.

The following are some user authentication features that we included to ensure account security. We used Firebase by Google for user authentication.



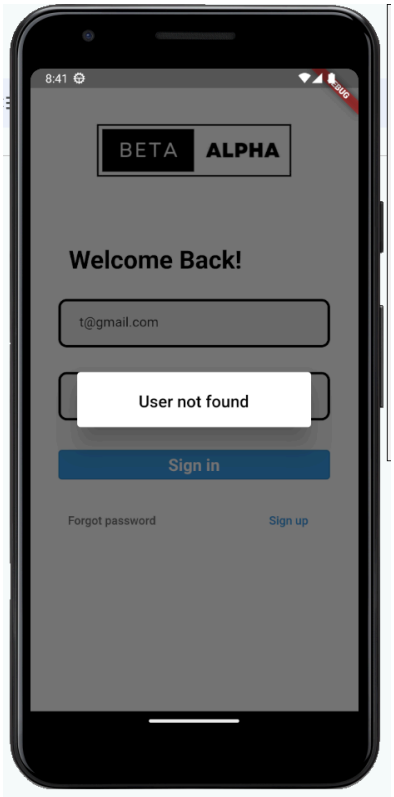
In the **Sign-up** page, if the user enters a different string in the password and confirm password fields and clicks on sign up, an exception would be thrown and the following message would be printed in the dialog box.

A check is done to see if the password and confirm password strings are exactly the same and whether the password fulfils our minimum length requirement.



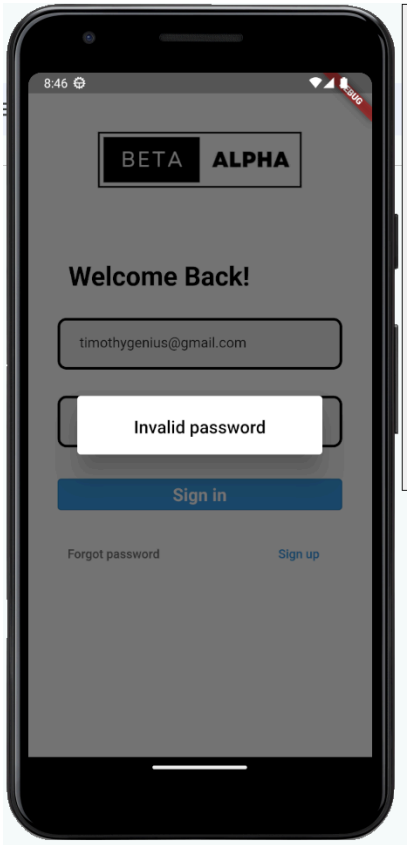
In the **reset password** page, our application conducts a check to see if the email that the users is in an email address' format.

If the user enters a string that is not in the correct format, an exception will be thrown and the following message will be printed in the dialog box.

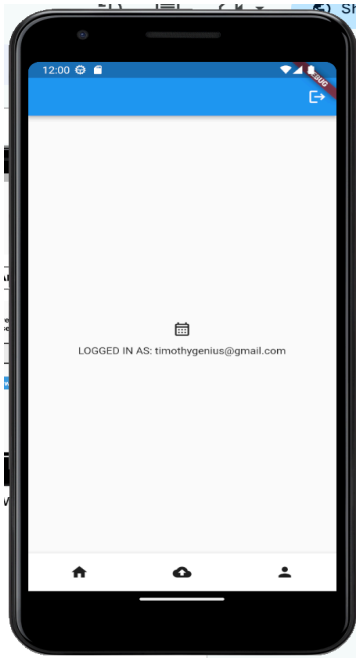


In BetaAlpha's sign in page, our application does a check in our database for an instance of the email that the user entered. Email addresses are unique.

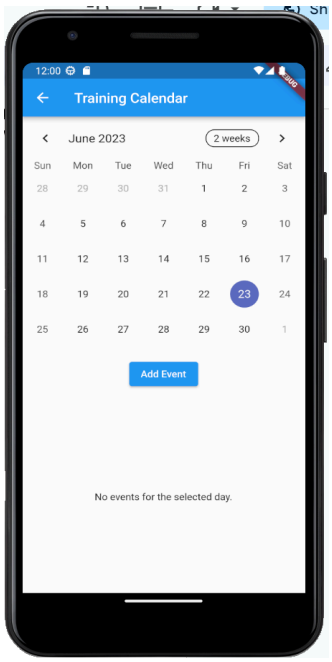
If there is no such user with the email address in our database, an exception is thrown and the following message is printed in the dialog box.



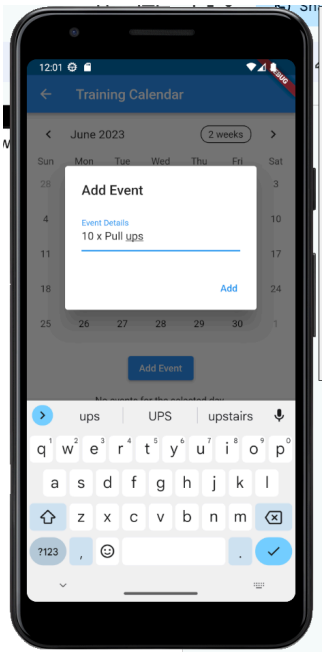
In the case where the user enters a email belonging to an account existing in our database but the password is incorrect, an “Invalid password” exception is thrown and the following message is printed in the dialog box.



Home Page

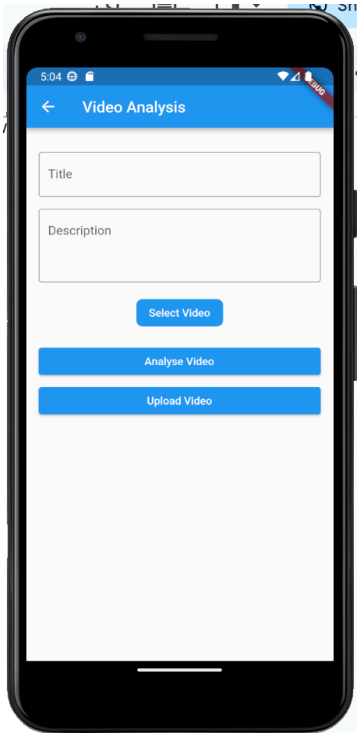


Calendar Page

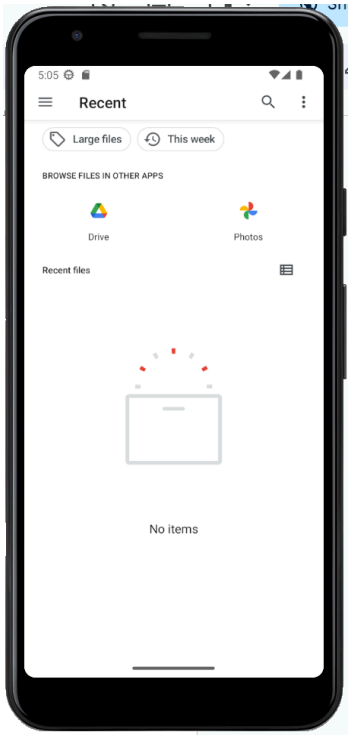


Add Event Widget

Video Upload Page:



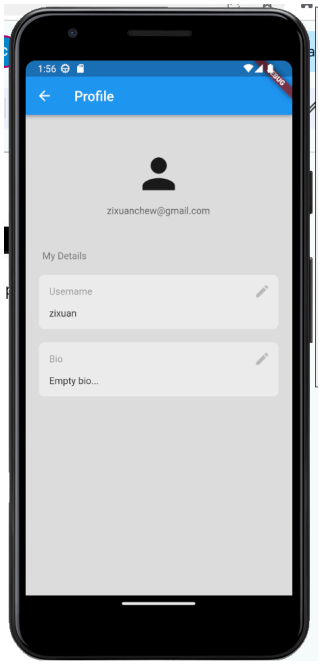
Video Upload Page



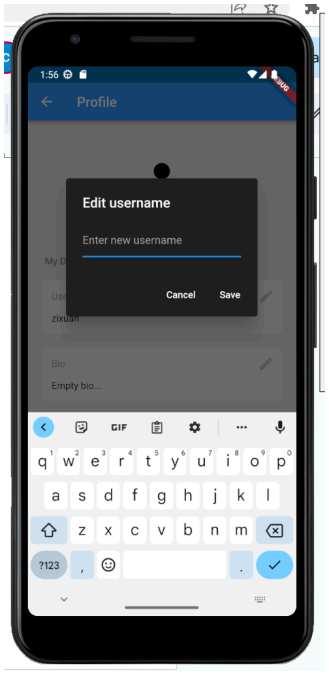
Upload from Drive or local gallery

Select video page

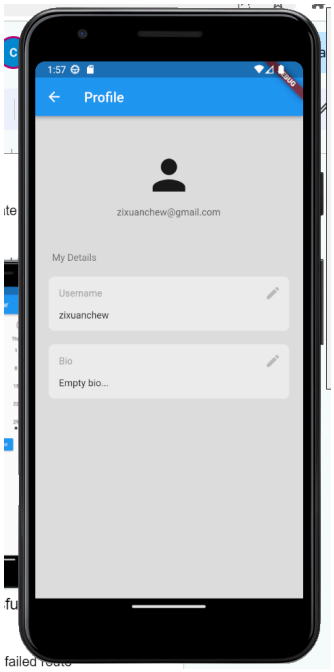
Profile Page:



Profile page



Edit username

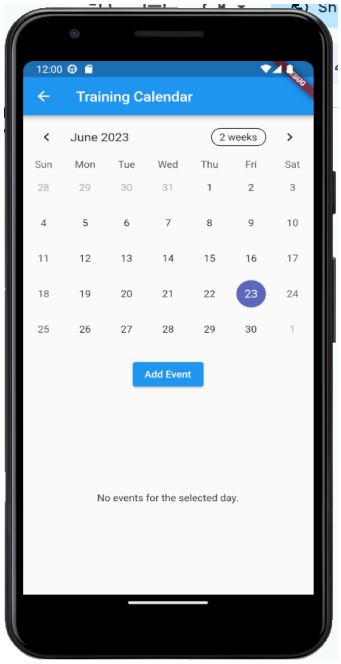


New username saved

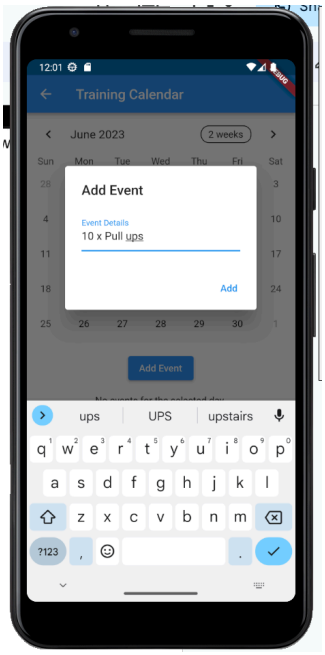
Features

Training Calendar: A calendar for the climber to plan and log training sessions to help the climber keep to his training schedule and ensure steady progress.

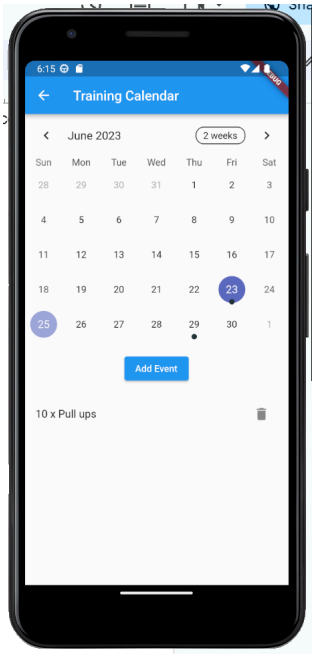
Uses Firestore as a database to keep each unique user's training plan which contains the date and training details.



Calendar Page

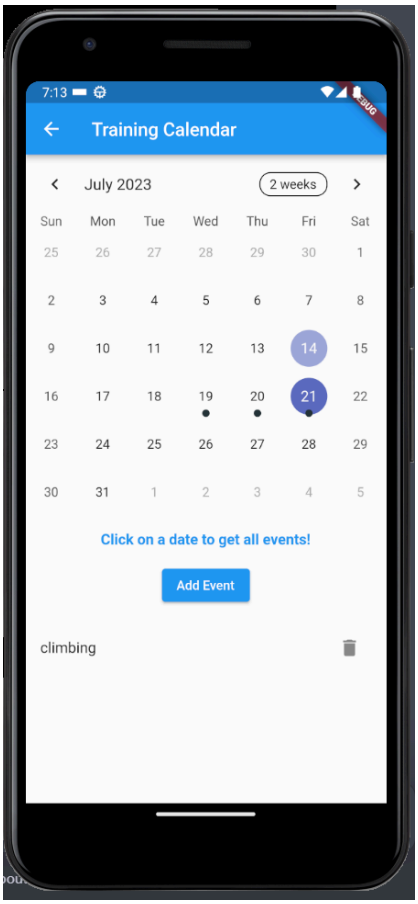


Add event Widget



Event successfully added

Notifications



Climbing video analysis:
A core feature that allows the climber to submit a video of a failed route attempt for analysis of his body positions and joint movements. Our pose estimation model is based on OpenPose by quanhua92. Here is his github: https://github.com/quanhua92/human-pose-estimation-opencv

One implementation that we are currently working on is deploying the pose estimation model on an online server and sending a video from our app to the server for analysis. We are currently exploring Flask and Heroku.

Multi-layered testing
User Testing

We tested each user action and devised the following tests as follows:

Log-In page

Test Type	Test	Pass/Fail
Widget	Check if log-in button brings user to Home Page	Pass
Widget	Check if forget password button brings user to password reset page	Pass
Widget	Check if Sign up button brings user to Register Account page	Pass

Test Type	Test	Pass/Fail
Developer	Check if user can enter log-in credentials and successfully log in	Pass
Developer	Check if user can reset password and password reset link is sent to user's email	Pass
Developer	Check if user can register for new account and account details is added to Firebase's authentication	Pass

Calendar page

Test Type	Test	Pass/Fail
Widget	Check if calendar button brings user to Calendar Page	Pass
Widget	Check if user can enter training programme details in add event widget	Pass
Widget	Check if add event can be clicked and functionality executed	Pass

Test Type	Test	Pass/Fail
Developer	Check if user can retrieve previously entered training programmes from Firestore	Pass
Developer	Check if user can add new training plan to Firestore	Pass
Developer	Check if user can delete training plan from calendar and Firestore accordingly	Pass

Video Upload page

Test Type	Test	Pass/Fail
Widget	Check if video upload button brings user to Video Upload Page	Pass
Widget	Check if select video button brings user to page that allows user to go to his gallery	Pass
Widget	Check if user can select a video for analysis	Pass

Test Type	Test	Pass/Fail
Developer	Check if select video button can connect the app to the user's local gallery	Pass
Developer	Check if user can reset password and password reset link is sent to user's email	Pass
Developer	Check if user can register for new account and account details is added to Firebase's authentication	Pass

Difficulties

Original plan: Initially, we wanted to use a TensorflowLite library in Flutter to handle the pose estimation aspect of our project and to store the model locally on a device. However, the library has been deprecated in the latest Flutter version and therefore we had to outsource our pose estimation model to a python library called OpenPose from OpenCV.

As we have to integrate a python script to our flutter app, we had difficulty establishing communication between our Flutter app and the Python Script, which has to be deployed on a server using Flask. Heroku was originally a free platform that we could use to deploy the Python script on a server and set up an API

endpoint that can receive requests and execute the desired Python function. However, there are no free tiers offered by Heroku and currently we are exploring other alternatives.

We will be using Flask to invoke our pose estimation model and return the resulting video when a request is made to the server's endpoint.

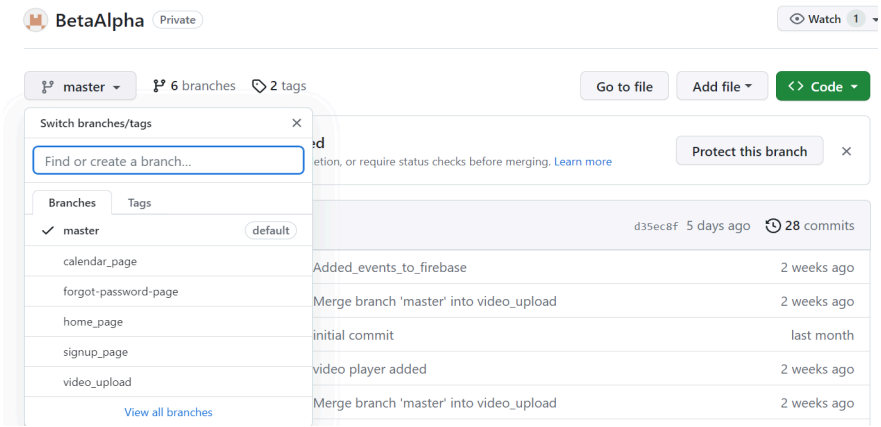
The key difficulties are choosing the right pose estimation model, how to use/deploy it and finding out how to incorporate it into our Flutter app.

There are also not much resources/guidance online regarding Flutter and its implementation with the various platforms. Even if there are, many of them are outdated due to the difference in Flutter version.

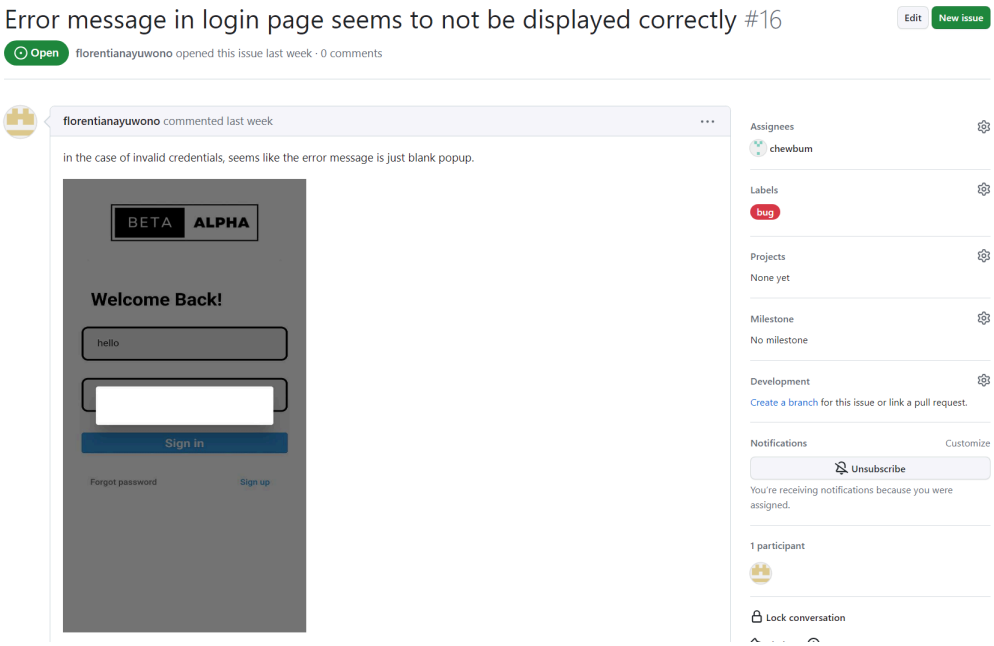
Version Control

We have decided to use Git on the terminal and GitHub for version control. We push commits frequently to our project's repository and we track issues using GitHub. In order to prevent merge conflicts from occurring, we delegated different tasks for each member to work on and we created different branches to work on our individual tasks before merging to the main branch. This helped us to reset to previous versions of our code.

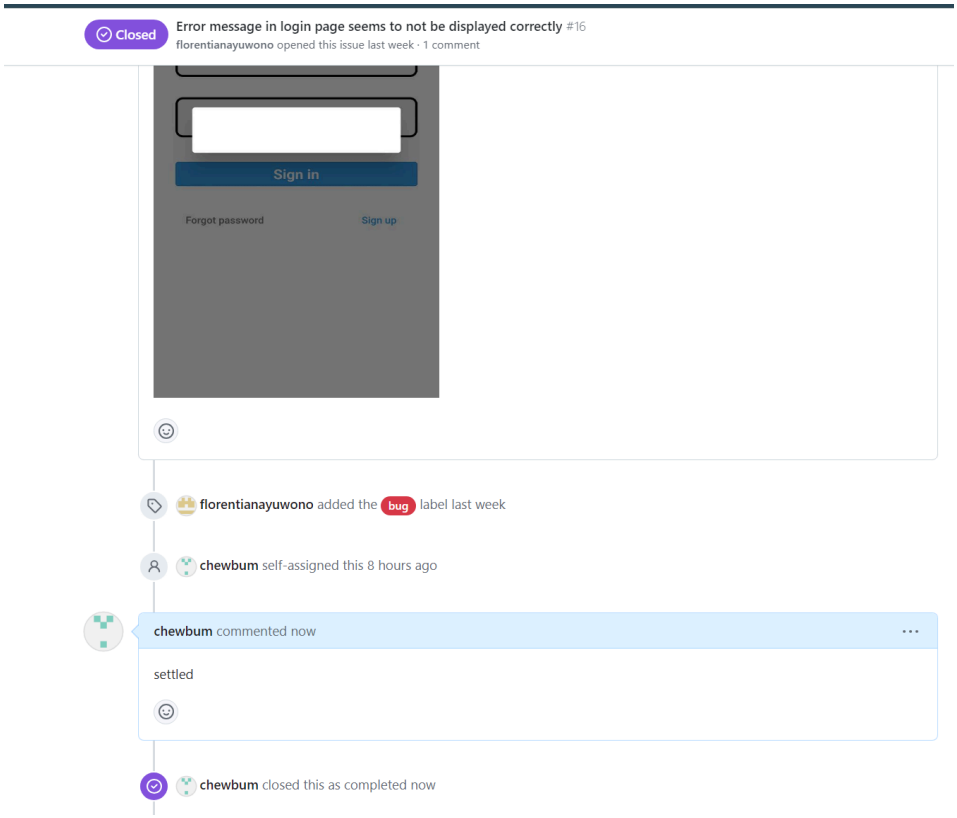
Below shows our different branches other than the main branch to show the splitting up of work and how we work on separate features at the same time:



Assigning of issues/bugs to each other:



Close issue upon solving bug:



Software Engineering Practices

Below are the software engineering practices that we followed:

Naming Practices	Variables and classes are named with nouns and Methods are named with verbs to differentiate between the two and make it closer to a real-world scenario. We followed the Camelcase convention to name our methods.
File Structuring	We structured our files into different folders according to the functionality they belong in so it is easier to find each class.
Code Structuring	Our code is structured logically such that class attributes, instance attributes, constructors, and methods are separated with empty lines. Indentation practices and good code styling practices were also followed. Our team placed emphasis on single responsibility principle where each class was designated to a singular job.
Team review	We do pair programming and during these sessions, we review each other's code and give advice to one another on how we can improve our code to make it more elegant.

Development Timeline

Below is the timeline that we will adhere to during the course of orbital:

1-7 May	• Create github repo • Start learning Dart and experimenting • Focus on frontend first	
8-21 May	• Create login page • Create home screen • Create profile screen • Create saved projects screen • Create video player page • Create loading screen	
22 May - 29 May	• Incorporate Firebase as our backend	
30 May - 6 June	• Experiment with pose estimation models	

	(Week 2)	
7 June - 14 June	• Finalise our video analysis feature	
15 June - 22 June	• Create training log function	
23 June - 30 June	• Create training plan function	
1 July - 8 July	• Start to incorporate the pose estimation model to our app by deploying it on heroku	
9 July - 16 July	• Deploy application to google play store	