



WEATHER APP DOCUMENTATION

Youssef M Taha
Cellula Technologies




Table of Contents

<i>Introduction.....</i>	<i>2</i>
Overview	2
Development methodology	2
<i>Project description.....</i>	<i>3</i>
Project overview	3
Scope and requirements	3
Product backlog	4
Sprints	7

Introduction

Overview

The aim of this project is to build an application that fetches weather according to the location of the user and then checks weather it is suitable for training or not using a local artificial intelligence model hosted using python Flask.

Development methodology

The application was built using the Agile methodology dividing tasks into sprints to enhance the features incrementing process and testing. The project was built depending on clean architecture principles and using the model- view model -view (MVVM) principles to ensure an optimized application that is easy to update and fix bugs that arise during the testing.

Project description

Project overview

The system is built for users with secure authentication to login and signup using firebase and after the user signs in, the weather data for the week is fetched according to the location using a weather api. The weather data are then translated from the JSON format and then filtered to send the required data for the ai model to reply with the response.

The ai model requires data to be in the form of a list of Booleans if the weather is raining, sunny, hot, mild and humid or not. The ai model then responds by a Boolean if the weather is suitable or not.

Scope and requirements

- **Sign in:**
 - The user enters their data (email and password)
 - If the data is valid, the user is directed to the home screen.
 - Suitable prompts in case of errors.
- **Sign up:**
 - The user enters their data (email and password)
 - The user then is redirected to sign in
 - Suitable prompts in case of errors.
- **Reset password:**
 - The user enters their email.
 - A reset password email is sent by firebase to reset the password
 - The user is then prompted to sign in.
- **Home screen:**
 - The main screen of the application.
 - Once the user is in the home screen, a call to the weather api is made.
 - If the data is fetched a call to the ai model is made.
 - Suitable error handling and prompts in case of errors.

Product backlog

- Sign in

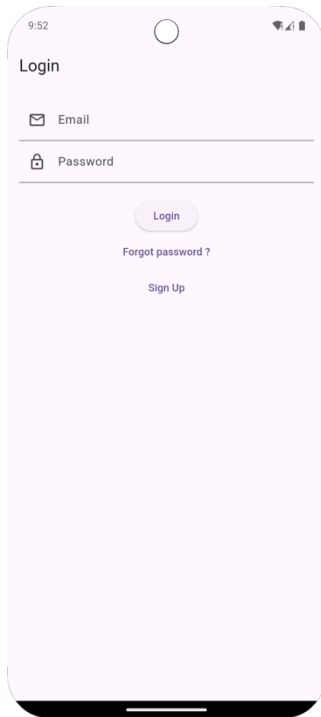


Figure 1: The sign in screen

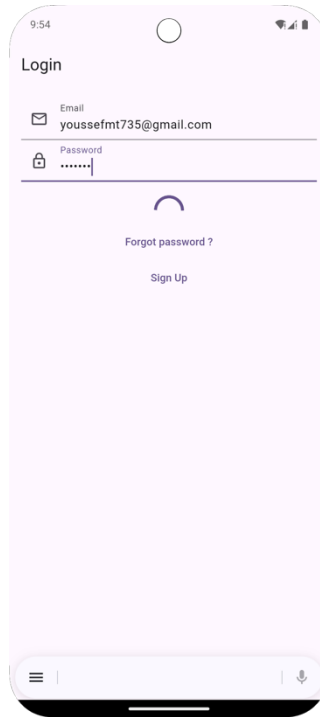


Figure 2 : Loading and processing the data

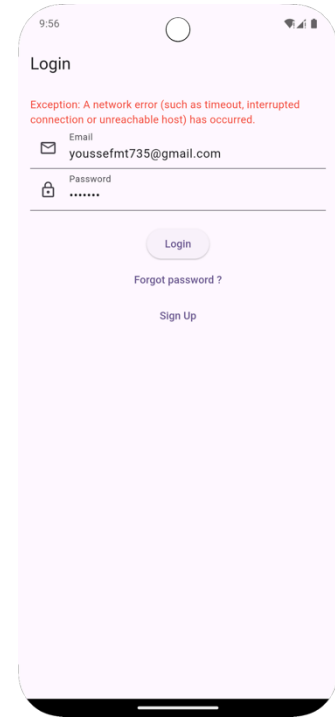


Figure 3 : Network error handling

- **Reset password/ sign up:**



Figure 3: reset password screen



Figure 4: sending the reset password email

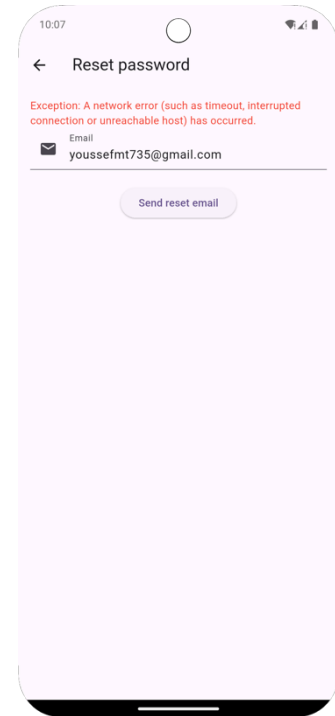


Figure 5: Error handling

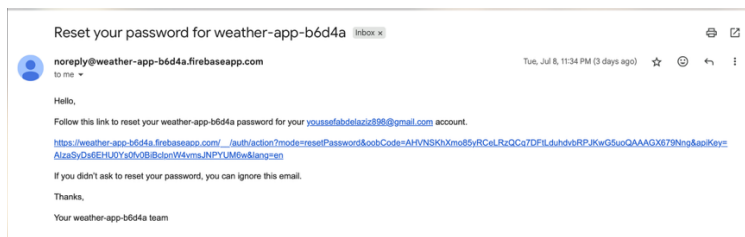


Figure 6: an old screenshot of the reset email.

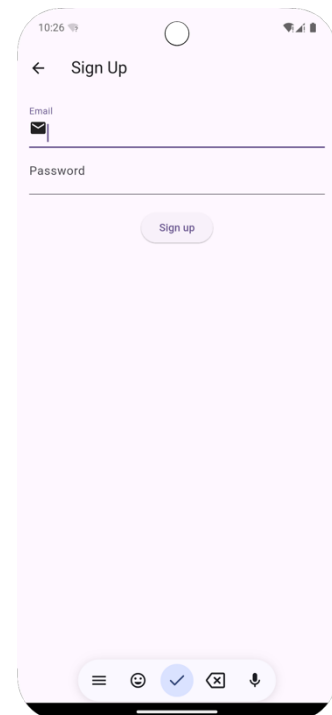


Figure 7: The sign up screen

- **Home screen:**
 - The city name is shown in the head
 - A calendar of the upcoming week is shown
 - Weather info is shown with the AI response

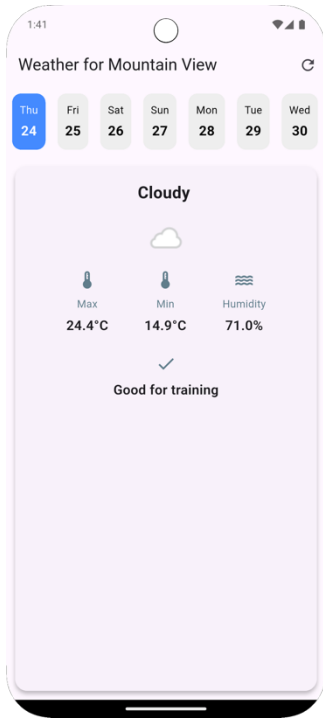


Figure 8: the home screen with suitable training conditions

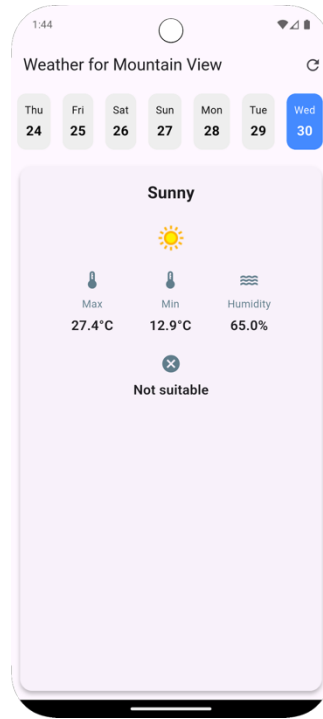


Figure 9: the home screen with a non suitable conditions

Sprints

The project was split into three main sprints:

- **Authentication:**
 - Setting up the firebase.
 - A simple UI for the sign up/ sign in/ reset password screens.
 - Testing the authentication process.
 - Directing to a dummy home screen.
- **Weather API:**
 - Setting up the location request.
 - Fetching the current location.
 - Sending the location to the API.
 - Receiving the response and translating the JSON response.
 - Updating the home screen to show the basic weather conditions.
- **AI integration:**
 - Setting up the python environment.
 - Locally hosting the server.
 - Modifying the weather viewmodel to create the features list required by the model.
 - Sending the features to the model.
 - Checking the response.
 - Modifying the home screen to show the model's response.