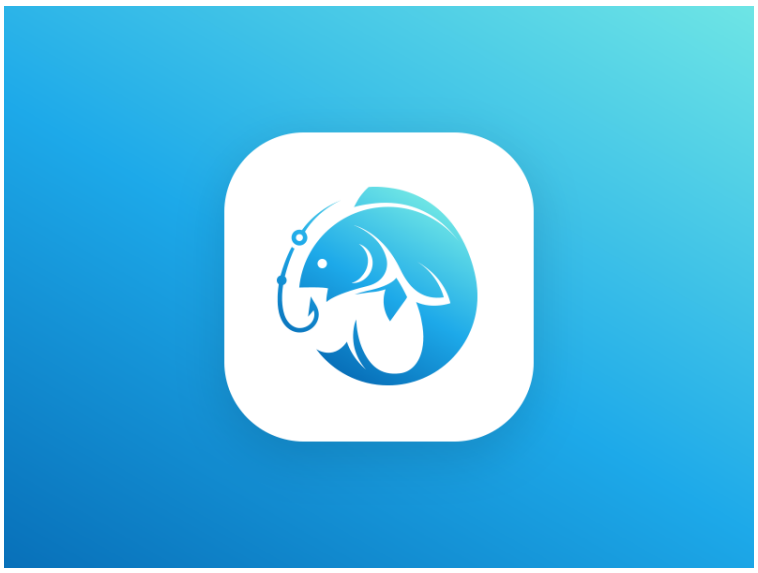
Department of Computer Science & Engineering
The University of Texas at Arlington

Detailed Design Specification
CSE 4317: Senior Design II
Spring 2023



Team FishQuest
Fish Quest

William Sigala
Othman Kamel
Kevin Phan
Brandon Stibich
Mohammed Ahmed Zakiuddin

Revision History

Revision	Date	Author(s)	Description
0.1	01.30.2023	BS	Document Creation
0.2	02.12.2023	WS, OK, KP, BS, MZ	Complete Draft
1.0	02.12.2023	WS	Section 1,2 & Sensor Layer
1.0	02.12.2023	BS	Database Layer
1.0	02.12.2023	OK	Data Processing Layer
1.0	02.12.2023	KP	Catch Logger UI, Social Feed UI, Map UI
1.0	02.12.2023	MZ	User Login/Registration UI, Missions UI
1.1	05.01.2023	WS, OK, KP, BS, MZ	Final Draft

CONTENTS

List of Figures

List of Tables

1 Introduction

FishQuest is a mobile iOS and android application that revolves around fishing and is meant to improve the fishing experience for all anglers. FishQuest aims to make fishing more enjoyable by making it more social oriented and also through rewards to create a higher incentive to fish. The main features of FishQuest include a catch logger, a social feed page that displays recently logged catches, a map feature to display the locations of all logged catches, and fishing related missions that provide rewards when completed.

2 System Overview

The FishQuest application consists of four main layers. The layers are the Sensors, User Interface, Data Processing, and Database layers. The diagram below represents the layers and how they interact with one another.

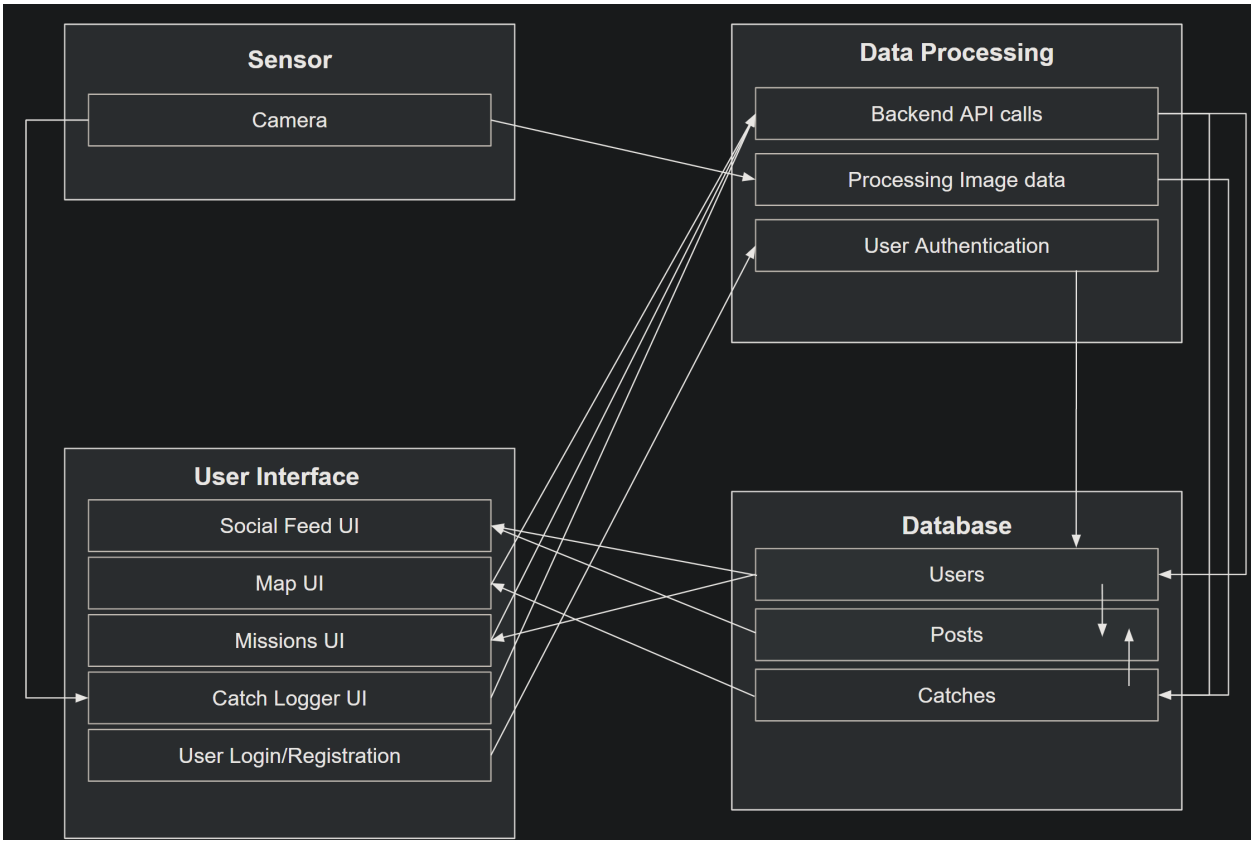


Figure 1: System architecture

The main purpose of the sensor layer is to enable image input for the components in our application that are dependent on images. Taking pictures will be done through the camera subsystem. The main app components that will receive images from the sensor layer are the catch logger and fish identifying machine learning model. The image is sent from the user interface to S3 cloud storage and also sent to the backend for inference. The data processing layer receives submitted input from the different components in the user interface, processes the data, and then stores the data into the database. The majority of the data will be processed and stored to the database by invoking a backend API. The data processing layer also processes credential data for unique users which will enable account registration

and log in operations. This information is securely stored with encryption. The fish image sent from the catch logger will be processed here with the fish classifier and return the prediction and any additional output to the user.

The user interface layer is the graphical side of the application that will allow users to navigate through the application. This layer is also responsible for handling user input from the different app components and then sending the data to the backend to be processed and stored. Aside from input, this layer also handles output by fetching data from the database and cloud storage to display it on the different pages in the app. The user interface layer will allow users to create a new account, log in to an existing account, log catches, interact with other users on the social feed page, view fishing missions, and access a map that displays all logged catches. The database layer is where all of FishQuest's data is stored. The database is divided into several partitions to group different types of data. The database contains partitions for users, catch posts, and logged catches. Image files will be stored seperately in S3 storage service.

3 Sensor Layer Subsystems

The sensor layer is a core component in the functionality of the app. It provides the image data by capturing the input from the camera of the user's phone. Upon the user's submission of a catch, the image data from the camera is transferred over to the data processing layer.

3.1 Layer Hardware

The device of the user will provide any necessary hardware.

3.2 Layer Operating System

This layer will require an Android or iOS device.

3.3 Layer Software Dependencies

This layer uses React Native for accessing the sensor data.

3.4 Camera

The camera from the user's device will capture image data when the camera view is open. The captured image data is then displayed to the user.

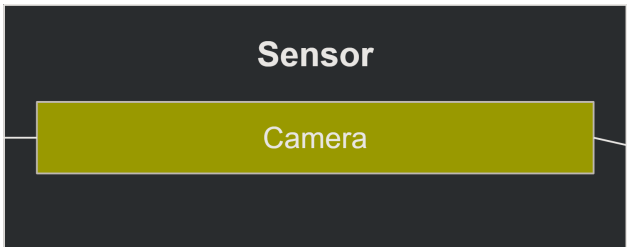


Figure 2: Camera Subsystem Diagram

3.5 Subsystem Hardware

The camera on the user's device.

3.6 Subsystem Operating System

This layer will require an Android or iOS device.

3.7 Subsystem Software Dependencies

React Native

3.8 Subsystem Programming Languages

Javascript

3.8.1 Subsystem Data Structures

The data structure used for this subsystem is the the base64 string representation of the image output from the camera.

3.8.2 Subsystem Data Processing

File compression and resizing will be performed on the image along with a HTTP POST request

4 User Interface Layer Subsystems

4.1 Layer Hardware

The hardware that allows users to interact with this layer is a smartphone.

4.2 Layer Operating System

This layer can run on both iOS and Android.

4.3 Layer Software Dependencies

This layer is dependent on a framework called React Native which is used to develop frontend components for mobile applications.

4.4 Social Feed User Interface Subsystem

The social feed UI provides users with an interface in the form of a traditional social media home page that displays recent fishing posts from other users. The social feed UI will also allow users to create posts of their own.

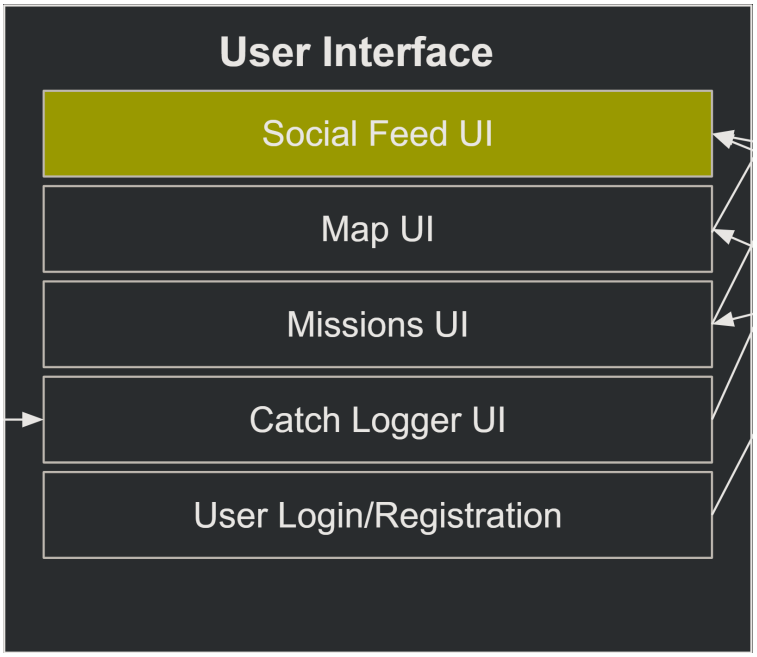


Figure 3: Social Feed UI Subsystems Diagram

4.4.1 Subsystem Hardware

A mobile device with internet connectivity is required to interact with the feed page.

4.4.2 Subsystem Operating System

This subsystem requires either Android or iOS to run.

4.4.3 Subsystem Software Dependencies

React Native framework is used to generate the UI components. NestJS and Express are required to store and fetch data to and from the database.

4.4.4 Subsystem Programming Languages

JavaScript is used for this subsystem.

4.4.5 Subsystem Data Structures

Data that is transferred to and from the social feed UI is structured in a JSON format.

4.4.6 Subsystem Data Processing

React Native provides a context API that will verify if the user is logged in and that their session is valid before displaying the social feed to the user.

4.5 Map User Interface Subsystem

The map UI subsystem provides users with a map that displays all previously logged catches from all users. The catches will e represented by markers at the location of the catch.

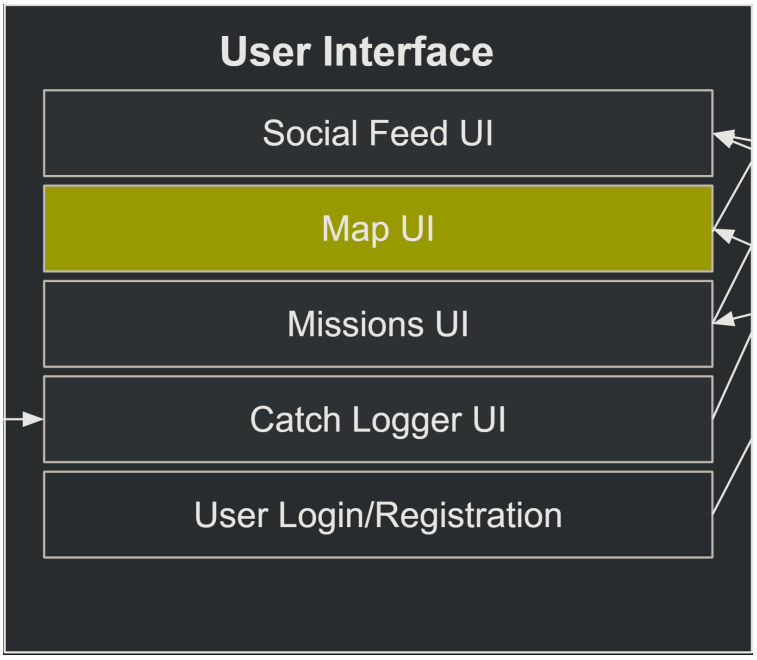


Figure 4: Map UI Subsystems Diagram

4.5.1 Subsystem Hardware

This subsystem will require a mobile device with internet connectivity and location services enabled.

4.5.2 Subsystem Operating System

This subsystem requires either Android or iOS to run.

4.5.3 Subsystem Software Dependencies

The map will require a React Native package called react-native-maps.

4.5.4 Subsystem Programming Languages

JavaScript

4.5.5 Subsystem Data Structures

The map will query location coordinates of catches from the database in a JSON format.

4.5.6 Subsystem Data Processing

Users must have their location services turned on and be logged in to view the map.

4.6 Missions User Interface Subsystem

The missions UI displays all of the active and completed fishing missions for each user.

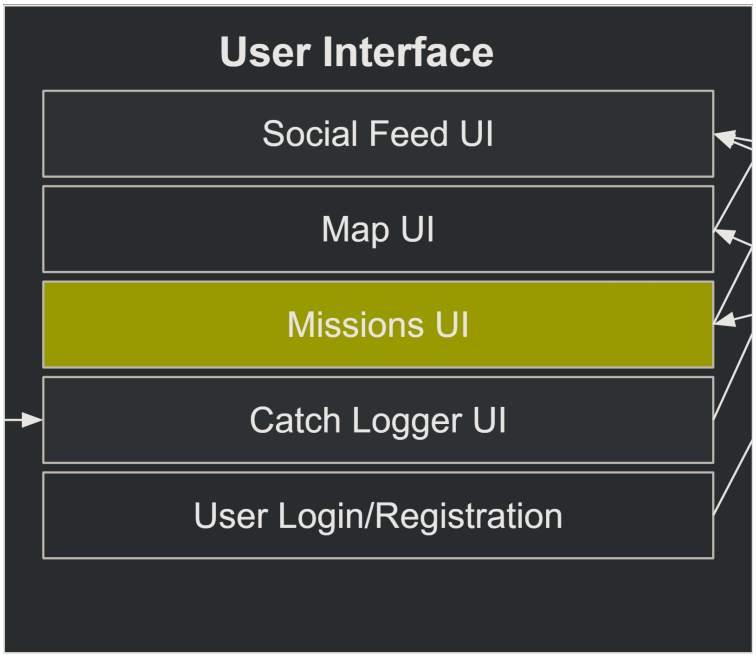


Figure 5: Missions UI Subsystems Diagram

4.6.1 Subsystem Hardware

A mobile device with internet connectivity is required.

4.6.2 Subsystem Operating System

This subsystem requires either Android or iOS to run.

4.6.3 Subsystem Software Dependencies

React Native is used to create the UI components of the missions page.

4.6.4 Subsystem Programming Languages

JavaScript

4.6.5 Subsystem Data Structures

The missions UI will receive data from the backend in JSON format.

4.6.6 Subsystem Data Processing

Refreshing the missions page will query new missions from the backend if they are available.

4.7 Catch Logger User Interface Subsystem

The catch logger UI allows users to log catches in addition to providing details about the catch such as an image, caption, weight, size, location, etc.

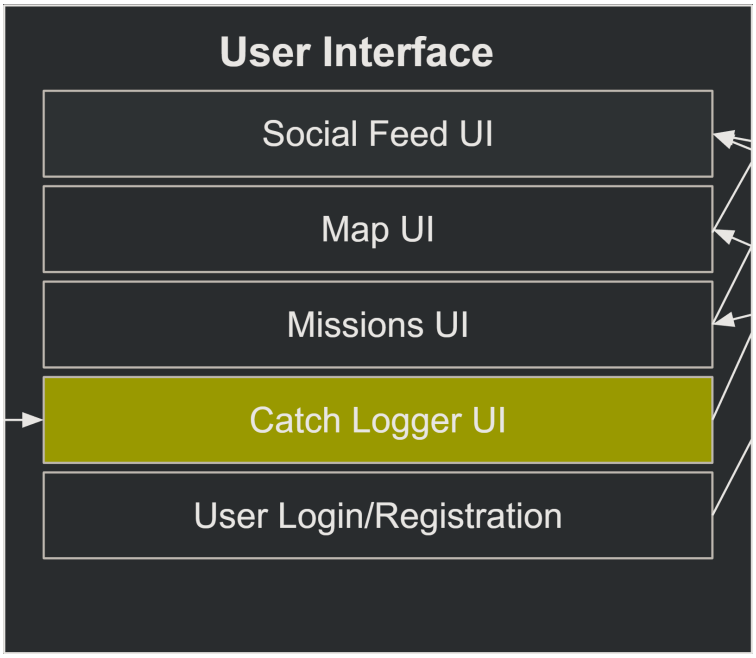


Figure 6: Catch Logger UI Subsystems Diagram

4.7.1 Subsystem Hardware

This subsystem will require a mobile device with internet connectivity, location services enabled, and a camera if the user chooses to take pictures of their catch.

4.7.2 Subsystem Operating System

This subsystem requires either Android or iOS to run.

4.7.3 Subsystem Software Dependencies

React Native is used to create the UI components of the catch logger.

4.7.4 Subsystem Programming Languages

JavaScript

4.7.5 Subsystem Data Structures

The catch logger UI will submit data to the database in JSON format.

4.7.6 Subsystem Data Processing

Users must have their location services turned on and be logged in to log catches.

4.8 Login/Registration User Interface Subsystem

This subsystem allows users to register for an account or to login to an existing account.

4.8.1 Subsystem Hardware

A mobile device with internet connectivity is required.

4.8.2 Subsystem Operating System

This subsystem requires either Android or iOS to run.

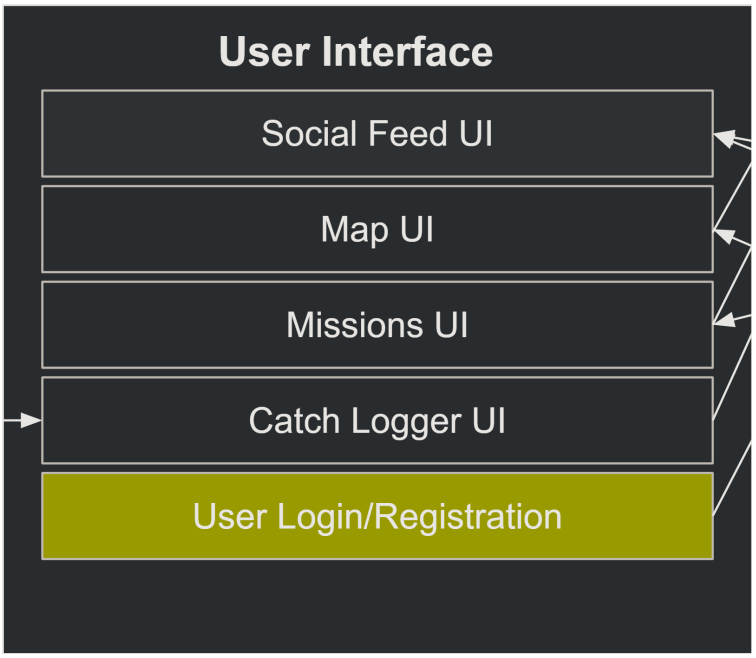


Figure 7: Login/Registration UI Subsystems Diagram

4.8.3 Subsystem Software Dependencies

React Native is used to create the UI components of the login and registration pages.

4.8.4 Subsystem Programming Languages

JavaScript

4.8.5 Subsystem Data Structures

When the user submits their registration information, the data will be structured in JSON format before being sent to the backend.

4.8.6 Subsystem Data Processing

React Native state will be used to determine if essential data inputs are empty before submitting registration information to the database.

5 Data Processing Subsystems

The Data Processing layer is responsible for handling the backend of the application. It receives input from the frontend layers which are the Sensor and User Interface layers and processes the data received before sending it to the database. Specifically, the data processing layer handles API calls, processes image data, and handles user authentication.

5.1 Layer Hardware

There will be no hardware involved as this is a purely software project.

5.2 Layer Operating System

IOS and Android devices.

5.3 Layer Software Dependencies

Software dependencies required include: Express, NestJS, NodeJS, TypeORM, and PostgreSQL.

5.4 Backend API Calls Subsystem

The purpose of this subsystem is to handle the requests made by the user and return back the appropriate response to the user. These requests include the basic functions: create, read, update, and delete (CRUD).

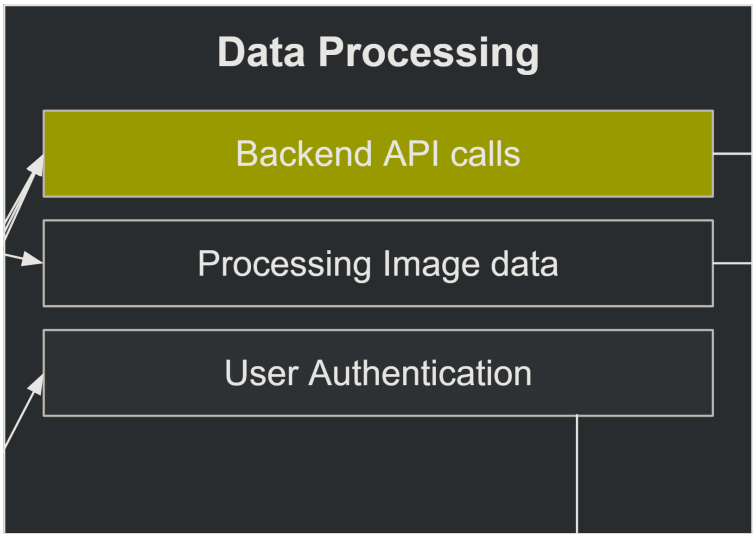


Figure 8: Backend API calls Subsystems Diagram

5.4.1 Subsystem Hardware

No hardware required.

5.4.2 Subsystem Operating System

No specific Operating System is needed.

5.4.3 Subsystem Software Dependencies

NestJS, NodeJS, TypeORM

5.4.4 Subsystem Programming Languages

TypeScript

5.4.5 Subsystem Data Structures

Some of the data processed will be stored in objects.

5.4.6 Subsystem Data Processing

HTTP methods such as POST, GET, PUT, and DELETE will be used to perform operations.

5.5 Processing Image Data Subsystem

The purpose of this subsystem is to process the image received from the camera and run the image detector to identify the type of fish as well as process the data entered by the user regarding the fish.

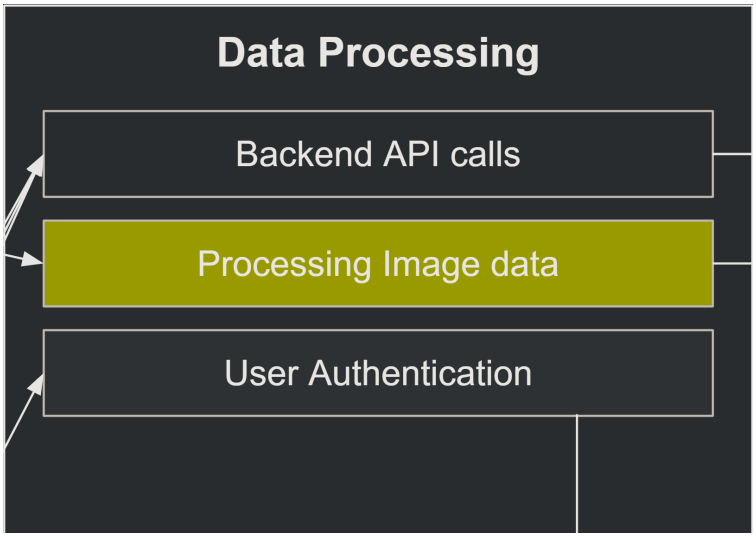


Figure 9: Processing Image Data Subsystems Diagram

5.5.1 Subsystem Hardware

No hardware required.

5.5.2 Subsystem Operating System

The Operating System of the camera that the image is taken from will be an Android or IOS device.

5.5.3 Subsystem Software Dependencies

React Native, NodeJS

5.5.4 Subsystem Programming Languages

TypeScript

5.5.5 Subsystem Data Structures

The classifier used for the image recognition will be stored in a key-value pair. The data entered by the user will be stored in a JSON format.

5.5.6 Subsystem Data Processing

The dataset is trained using the YOLOv8 model in PyTorch and ran with ONNX Runtime on the server.

5.6 User Authentication

The purpose of this subsystem is to verify the identity of the user in order to access their account.

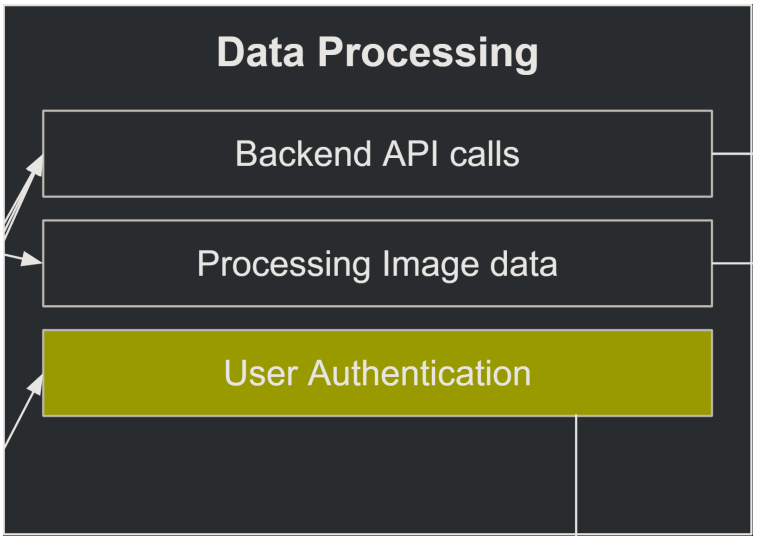


Figure 10: User Authentication Subsystems Diagram

5.6.1 Subsystem Hardware

No hardware required.

5.6.2 Subsystem Operating System

Android or IOS device

5.6.3 Subsystem Software Dependencies

React Native

5.6.4 Subsystem Programming Languages

TypeScript

5.6.5 Subsystem Data Structures

We will use async/await functions.

5.6.6 Subsystem Data Processing

We will use Basic Auth using the Axios HTTP client.

6 Database Layer Subsystems

The database layer is used to store, input and output data pertaining to all facets of the application. This layer will keep any information for later access be it used by a developer or a user.

6.1 Layer Hardware

There will be no hardware involved as this is a purely software project. Though AWS will be hosting the database, whatever AWS uses for their hardware can be mentioned here.

6.2 Layer Operating System

The OS of the db will be run on an aws server that will be running postgreSQL. The database will be connected to the rest of the app via typeORM.

6.3 Layer Software Dependencies

This layer requires typeORM and postgreSQL

6.4 Users Subsystem

The users subsystem describes the group of database tables that encapsulates all user data within the app. This includes the following tables: User, Achievements, and AchievementLabels.

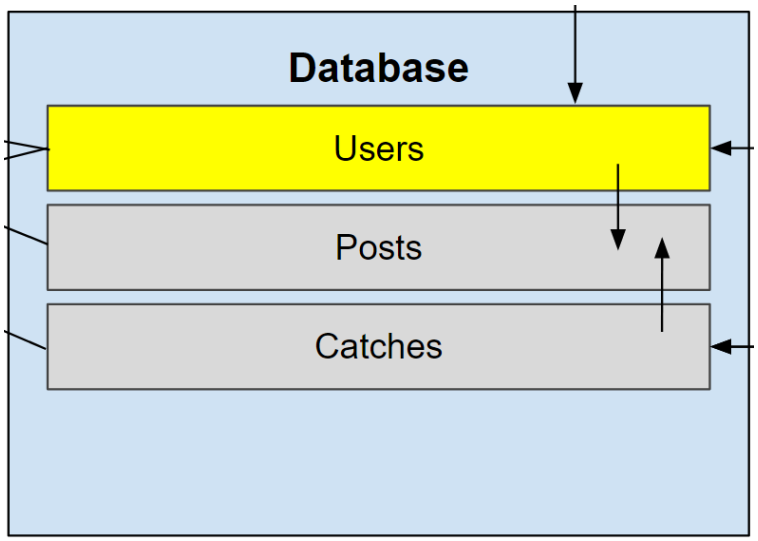


Figure 11: Users Subsystems Diagram

6.4.1 Subsystem Hardware

No hardware on our end, just whatever AWS uses for their server hosting.

6.4.2 Subsystem Operating System

This subsystem runs on postgreSQL

6.4.3 Subsystem Software Dependencies

This subsystem uses postgreSQL and typeORM

6.4.4 Subsystem Programming Languages

Typescript and SQL

6.4.5 Subsystem Data Structures

The data structures used for this subsystem is the tabular database structure of the SQL based postgreSQL.

6.4.6 Subsystem Data Processing

No extra data processing outside of whatever postgreSQL uses internally

6.5 Posts Subsystem

The posts subsystem describes the tables in the database pertaining to social media posts. This includes the tables: Post, Reaction, and Comment.

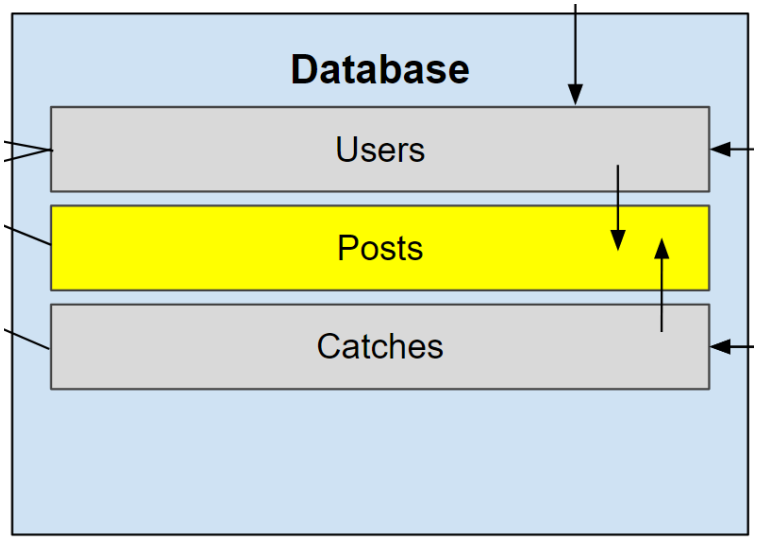


Figure 12: Post Subsystems Diagram

6.5.1 Subsystem Hardware

No hardware on our end, just whatever AWS uses for their server hosting.

6.5.2 Subsystem Operating System

This subsystem runs on postgreSQL

6.5.3 Subsystem Software Dependencies

This subsystem uses postgreSQL and typeORM

6.5.4 Subsystem Programming Languages

Typescript and SQL

6.5.5 Subsystem Data Structures

The data structures used for this subsystem is the tabular database structure of the SQL based postgreSQL.

6.5.6 Subsystem Data Processing

No extra data processing outside of whatever postgreSQL uses internally

6.6 Catches Subsystem

This subsystem pertains to all the tables pertaining to catches. In this particular instance there is only one table, 'Catch', however it is more independent and does not fall well into the other categories.

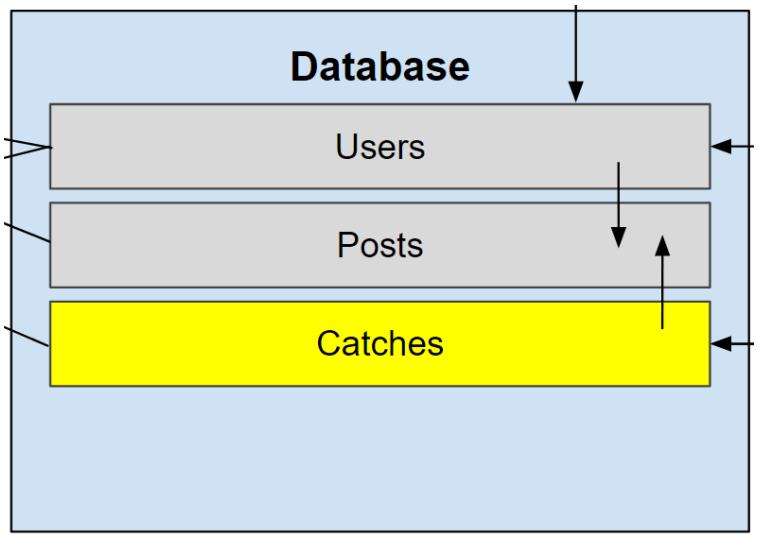


Figure 13: Catches Subsystem Diagram

6.6.1 Subsystem Hardware

No hardware on our end, just whatever AWS uses for their server hosting.

6.6.2 Subsystem Operating System

This subsystem runs on postgreSQL

6.6.3 Subsystem Software Dependencies

This subsystem uses postgreSQL and typeORM

6.6.4 Subsystem Programming Languages

Typescript and SQL

6.6.5 Subsystem Data Structures

The data structures used for this subsystem is the tabular database structure of the SQL based postgreSQL.

6.6.6 Subsystem Data Processing

No extra data processing outside of whatever postgreSQL uses internally

7 APPENDIX A

Include any additional documents (CAD design, circuit schematics, etc) as an appendix as necessary.

REFERENCES