CS307 Team 18

WhiteBoard - Design Document

Chunao Liu
Anurag Shah
Jenna Zhang
Yierpan Abuduwaili
Michelle He
Jingyuan Yang

Index

Purpose **3**

- Functional Requirements 3
- Non-Functional Requirements 4

Design Outline **7**

- High Level Overview of the system 7
- Interactions Between Components (with Activity Diagram) 9

Design Issues **10**

- Functional Issues: 10
- Non Functional Issues: 11

Design Details **13**

- Class Design 13
- Descriptions of classes and interaction between classes 14
- Sequence Diagrams 16
- UI Mockup 20

Database Details and Structure **31**

Purpose

Modern Coding usually requires a laptop with an IDE (Integrated Development Environment) or a compiler, but there are many situations where someone must test a code sample without that code even in a textual format. For example, an interviewee preparing a coding interview will have to write their code on a piece of paper or on a whiteboard, and without a compiler, one can only debug by observing their written code or typing it out in full, a waste of time for all parties involved.

The purpose of this project is to develop a novel “IDE for Written Code” that can support typeform code or scanning hand-written code, compile the code and return the terminal output and stack trace back to the user. This application will take the form of a cross-platform mobile application, and will allow users to save their code images and organize into teams.

While numerous services provide IDEs/Text editors, Compilers, and OCR services, on a variety of platforms, there is no single service that integrates all of these. Further, OCR services that can detect the full ASCII character set are rare and inaccurate. So, we are confident our application will be well above current trends in providing an effective, packaged solution!

Functional Requirements

1. Authorization
   As a user,
   a. I would like to use the basic application functionality (processing a single image taken as a photo) without needing to create an account.
   b. I would like to register an account with a unique identifier (email address or phone number) for my Username
   c. I would like to be taken to the login screen if I am not logged in.
   d. I would like to have a “remember me” feature I can select when logging in, to stay logged in if I have not previously logged out.
   e. I would like to be able to reset my password if I forget it.

2. General Application Features
   As a user,
   a. I would like to have a navigation bar in all main pages to move around the application easily.

3. User Profile and Preferences
   As a user,
   a. I would like to have a profile page where I can enter and update my personal information.

b. I would like to have a profile picture and a banner that is visible to other users.
c. I would like to change the theme of my application between Light and Dark.
d. I would like to be able to deactivate and reactivate my account.

4. <u>Taking/Uploading Photos</u>

As a user,
a. I would like to take a photo with my camera.
b. I would like to retake the photo if needed.
c. I would like to upload a photograph from my library.
d. I would like to be able to manually type code instead of taking a picture.

5. <u>Viewing Processed Code</u>

As a user,
a. I would like to know if there is an error in my code or not.
b. I would like to know the code compilation result, if it compiles.
c. I would like to know where in the code my syntax errors are, by having that line underlined in the output image with a marker, and upon clicking that line display a popup with the error details.
d. I would like to manually specify the compilation language if I feel the backend has identified the wrong one.

6. <u>Saving Code</u>

As a user,
a. I would like to save the text output of my code in an editor application.
b. I would like to save the image output of my code, if it has errors, into my phone's library.
c. I would like to delete any code I have saved.

7. <u>Teams and Team Management</u>

As a user,
a. I would like to create a team.
b. I would like to accept invitations to a team.
c. I would like to be in multiple teams.
d. I would like to leave a team.
e. I would like to see my team's details, members, creator and managers.
f. I would like to access all shared code between the team.

As a team owner,
a. I would like to assign managers to a team I have created.
b. I would like to remove managers from a team I have created.

As a team manager or owner,
a. I would like to invite users to my team.
b. I would like to remove a member from the team
c. I would like to edit or delete my team members' code that they have contributed to this team.

Non-Functional Requirements

1. <u>General</u>
   As a developer,
   a. I would like the application (built on React Native) to work cross-platform on both Android and iOS mobile devices.
   b. I would like the application to have a single backend service.
   c. I would like to have the backend pipeline be multithreaded, with the knowledge that running the image through the HCR Model will be the pipeline bottleneck.
   d. I would like to use a Cache Manager to decrease backend latency.
   e. I would like the OCR Models to have at least 100 images for each ASCII character, or have a model pre-trained on a set of similar or larger size.

2. <u>Security</u>
   As a developer,
   a. I would like to use an authentication system or protocol (ex. OAuth2).
   b. I would like to send a password reset link to users to their email addresses.
   c. I would like to prompt users to re-login when their account has been inactive for 3 weeks.
   d. I would like to delete a deactivated account permanently if it is not reactivated in 90 days.
   e. I would like to restrict users to view only profiles of other user's in a team with them.
   f. I would like users in a team to have different roles assigned to them, so that they can access different features.

3. <u>Database</u>
   As a developer,
   a. I would like to store each account information in a specific SQL table. This would include their authentication information and profile information, using a unique identifier column which is distinct from their authentication information. This would also include the directory in the backend's file system where the user's saved images, associated code and results, and image manifest file, are stored.
   b. I would like to store each team's information in a specific SQL table. This would include the members and their various roles. This would also include the directory in the backend's file system where the teams' saved images, associated code and results, and image manifest file, are stored.
   c. I would like the mobile application to cache these images in the local application storage.

4. <u>Testing and Continuous Delivery</u>
   As a product manager,

a. I would like to have a continuous delivery pipeline implemented on the backend.
b. I would like to have separate unit tests for the frontend and backend portions of the application.

5. Main Pipeline: Image Preprocessing, OCR, Post Processing, and Compilation

As a developer,

a. I would like the preprocessing pipeline filter to contain a model that can detect if the code is typeform or handwritten in nature, and select the correct segmentation algorithm, preprocessing filters, and OCR model for either case.
b. I would like the selected preprocessing filter to segment characters from the image.
c. I would like the selected preprocessing filter to preprocess segmented characters into the format the model expects.
d. I would like the selected preprocessing filter to de-skew images.
e. I would like the selected OCR model to convert the image into characters.
f. I would like the post processing filter to detect what language the code is in.
g. I would like the post processing filter to convert the text into an output format suitable for the compiler, retaining whitespace data where it has syntactic or semantic significance.
h. I would like the post processing filter to return the output code text to the client.
i. I would like the compiler to attempt to compile the code, and send the compilation result as well as any errors to the client.

Design Outline

High Level Overview of the system

This project will be a mobile application. Client, Server, and Database are the three main components of the project.

1. Client
    - The client application will provide an interface to our users, allowing them to interact with the system.
    - The client will send requests to the server via REST API.
    - The client will receive responses from the server and act on those responses.
    - There can be more than 1 client.

2. Server
    - The server-side application will receive requests from clients and generate responses back to clients after those requests have been handled.
    - The server will send queries to the database based on the requests received to store, modify, retrieve, or delete data.

3. Database
    - The database receives queries from the server and responds with extracted data or results of the queries.
    - The database and associated file storage system will store all data from the application, such as user account info and image manifest files and textual representation of code that the users save

4. Model Preprocessing
    - The model preprocessing suite will take in an image as input from the Server, and will perform the necessary preprocessing operations on it before being sent onwards to the Character Recognition model.

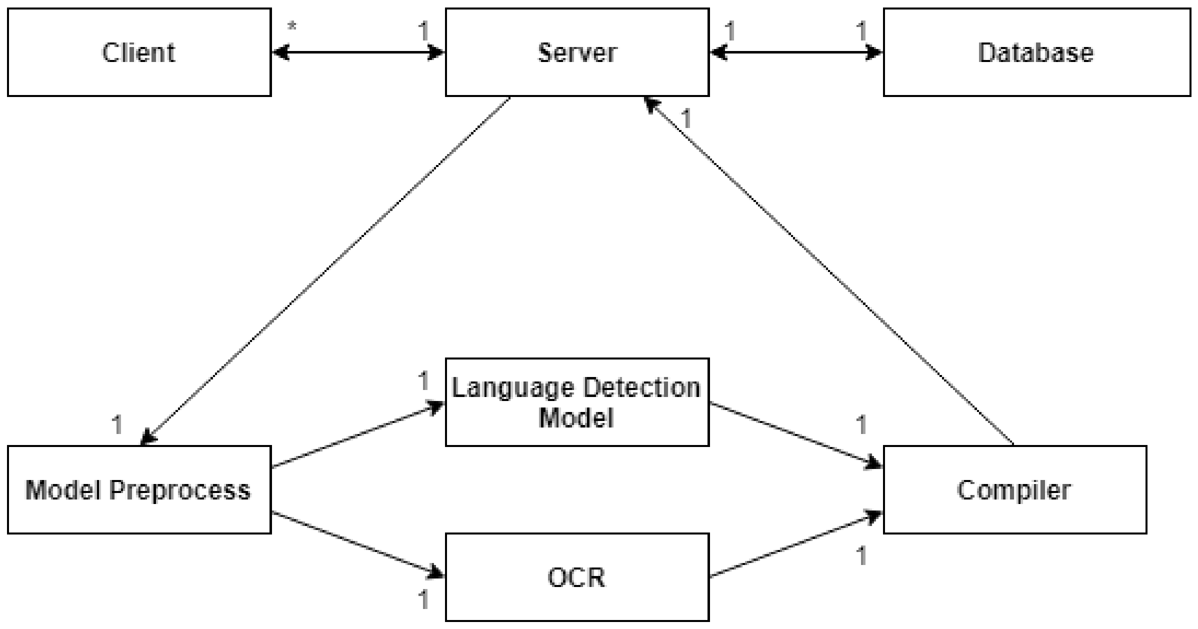
5. Language Detection Model
    - The Language Detection Model will take in the textual representation of the user's code from the image, and attempt to identify what programming language the code is written in. It will then forward this information to the Compiler step.

6. OCR
    - The OCR Model will take in an image and return the textual representation of that image, which it sends onwards to the Compiler.

7. Compiler
    - The compiler will take in the textual representation of the code, and the identified programming language, as its input.

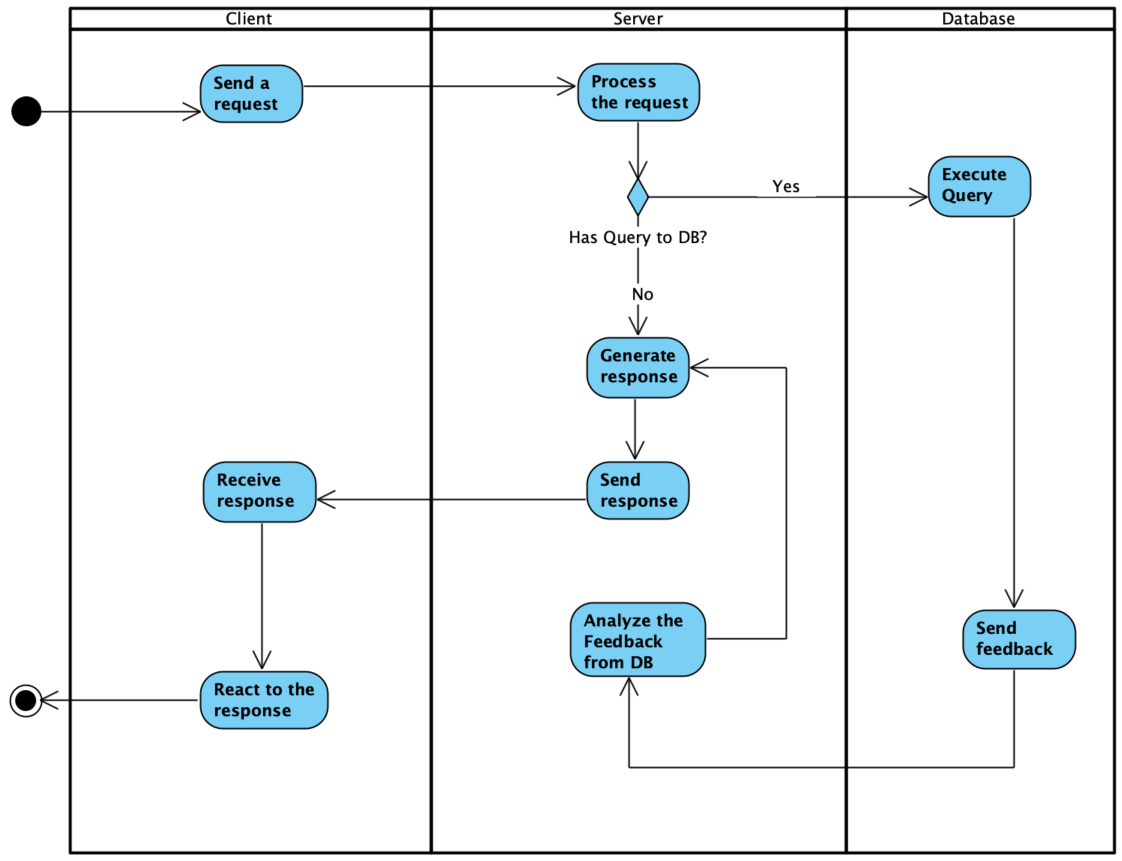
- It will respond to the server with the compilation result.

The application will be able to hold many Clients connected to the Server at the same time. When the Clients send requests to Server for accessing Database, the Server will queries the Database for the information requested or write the data to Database. When the Clients asks the Server to process the picture or typed code, the Server will send the request to the backend for Model Preprocessing, after being analyzed by OCR or the LDM, it will be sent to the Compiler to generate the syntax errors on the original file, the Compiler will send the result to the Server and the Server will send it back to the Clients or store the data to Database depending on Clients choice.



Interactions Between Components (with Activity Diagram)

The Android/IOS frontend client and the server will communicate via HTTP requests, and all requests will be made via the REST API. Data passed in those requests will be in a JSON format. The server will interact with the database using SQL, which will allow easy storage, retrieval, and modification of server-side data.



Design Issues

Functional Issues:

1. Do users need to login to use our service?
   - Option 1: Each user should have their unique account and password and the app will remember the returning user so that they do not have to enter password every time until they manually log out.
   - Option 2: The basic application functionality, ie. taking a photo and processing it, will not require the user to be logged in. Any other functionality, such as saving the images, accessing history, or team functionality, will require the user to be logged in, as in Option 1.

     Answer: Option 2
     We should allow users to use the application without logging in, as it may be something they use infrequently and do not require the advanced features for.

2. What information do we need to use to sign up an account?
   - Option 1: Users only need a username and password to sign up
   - Option 2: Users will need an username, email address, and a password, to sign up
   - Option 3: Users will need to use an external authentication system or Single Sign On system to sign up.

     Answer: Option 2.
     It is important for a user to have an email address associated with their account so that they can receive a password reset link. We have no reason to integrate with SSO Systems as they do not offer us any relevant advantages for our service.

3. How should the user input their code?
   - Option 1: Type code directly into a TextView.
   - Option 2: Take an image of a code sample.
   - Option 3: Take an image of a code sample or type code directly into a TextView

     Answer: Option 3
     Although Using OCR to process the image taken on-time is a key feature of our app, as an IDE it should allow the user to manually type the code and process it.

4. What should we show to the user after compilation?
   - Option 1: Return the standard output and stack trace
   - Option 2: Return the standard output, stack trace and the code in a textual form

- Option 3: Return the standard output, stack trace, the code in a textual form, and the original picture with errors underlined physically on the image (if any)

  Answer: Option 3
  To allow the user to see their compiled result with errors, and to allow the user to manually correct errors, the user should be able to access their original image submission. In order to fulfil the purpose of being an IDE, any errors should be underlined, with details forthcoming. Further, we need to send the textual representation of the code, so that any small errors can be fixed by the user with ease. If the compilation succeeds and the program runs without runtime errors, we do not need to return the stack trace, only the output. If the compilation fails, we need to return all of the above.

Non Functional Issues:

1. How are we going to host our backend services?
    - Option 1: Local Machine
    - Option 2: AWS

    Answer: Option 1
    We are first going to run it on a local machine owned by one of the team members. In case it cannot handle multiple users at the same time, we will upload the backend to the cloud AWS server.

2. What type of Database Management System are we going to deploy?
    - Option 1: Relational Database (SQL)
    - Option 2: Graph Database
    - Option 3: Document Database

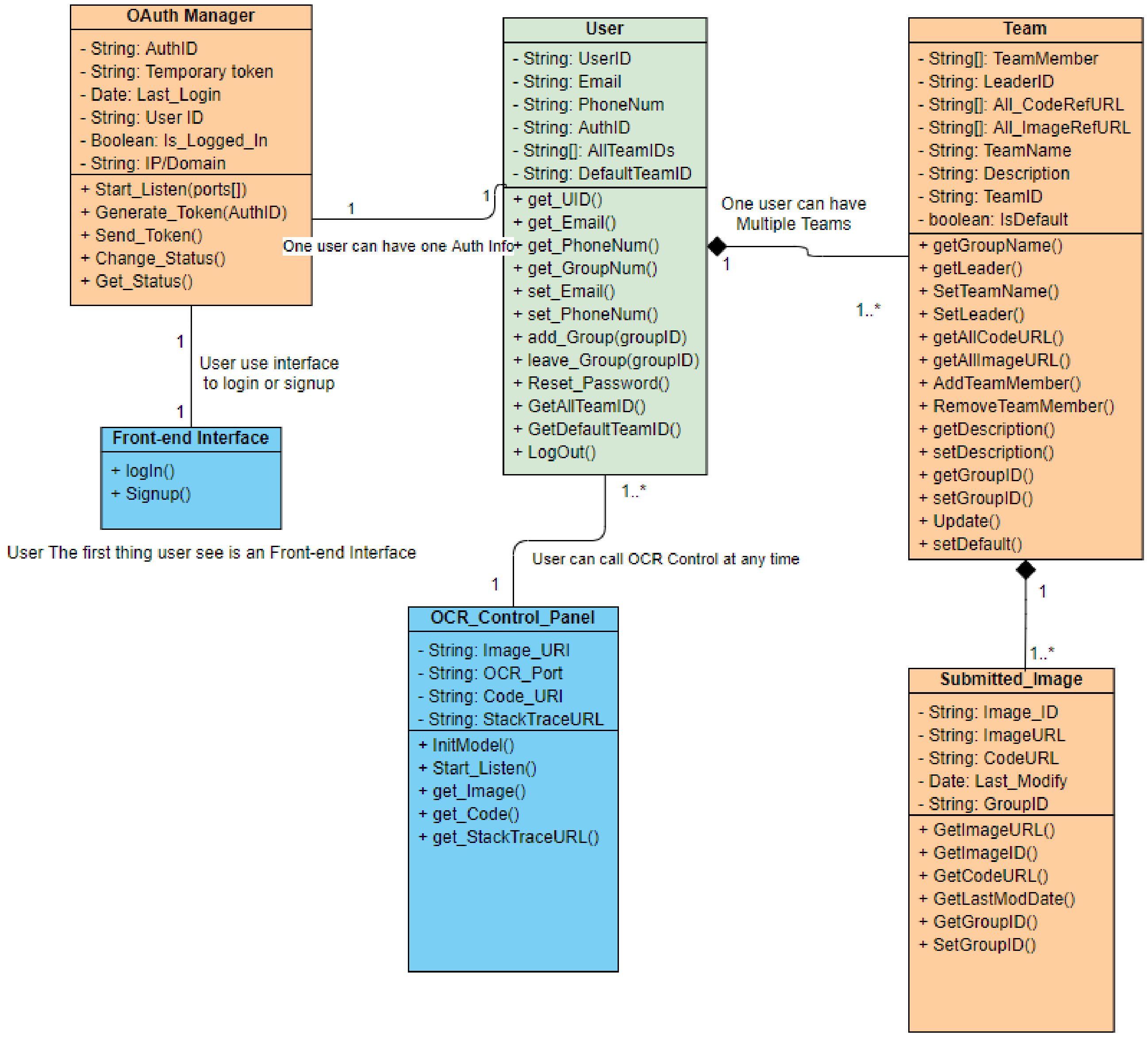
    Answer: Option 1
    Our database will include users' personal profile information, code information, authentication information, and team related information. A relational database is most suited to our needs, since each user's username (or uid) will appear in every table as either a primary key or a foreign key. This means all our tables are very connected with this specific unique identifier in this specific table, which is perfect for a Relational Database.

3. What back-end framework and language should we use?
    - Option 1: Spring (Java)
    - Option 2: Django (Python)
    - Option 3: Node.js (JavaScript)

Answer: Option 2
Python is by far the most convenient language to implement the Neural Networks needed. In order to make the data transfer between OCR's API and our backend easier to implement, we choose to use the back-end framework that uses the same language as our OCR's API, which is Django. With Django's extremely effective scalability, we can also easily reduce the difficulty to handle increasing requests from different users in the future.

Design Details

Class Design



Descriptions of classes and interaction between classes

- User
  - User refers to a registered user. People using our application without an account are visitors but not “Users” in this sense.
  - Each User will be associated with a unique user ID.
  - Each User will have a username, a password, and an associated email address.
  - Two accounts cannot be registered with the same email address (notably, the email address is still distinct from the unique identifier, as users should be allowed to change their address in their profile).
  - Users will be able to reset their password.
  - Users will have a list of all associated teams with them, including their default private team. Each group will be referred to an individual team class

- OAuth Manager:
  - Each user will be connected with his or her own OAuth Manager Interface.
  - Upon initialization, the OAuth manager will actively listen to a specific series of ports, which the front-end will use to send requests.
  - The user will send a request with its authentication information, including username and password, and the OAuth manager will verify it. If it is correct, we will create and return an Hashed token with the correct authority to the user and initialize the class so that the user stay logged in
  - If the login failed, the OAuth Manager will directly reject the request with an error message

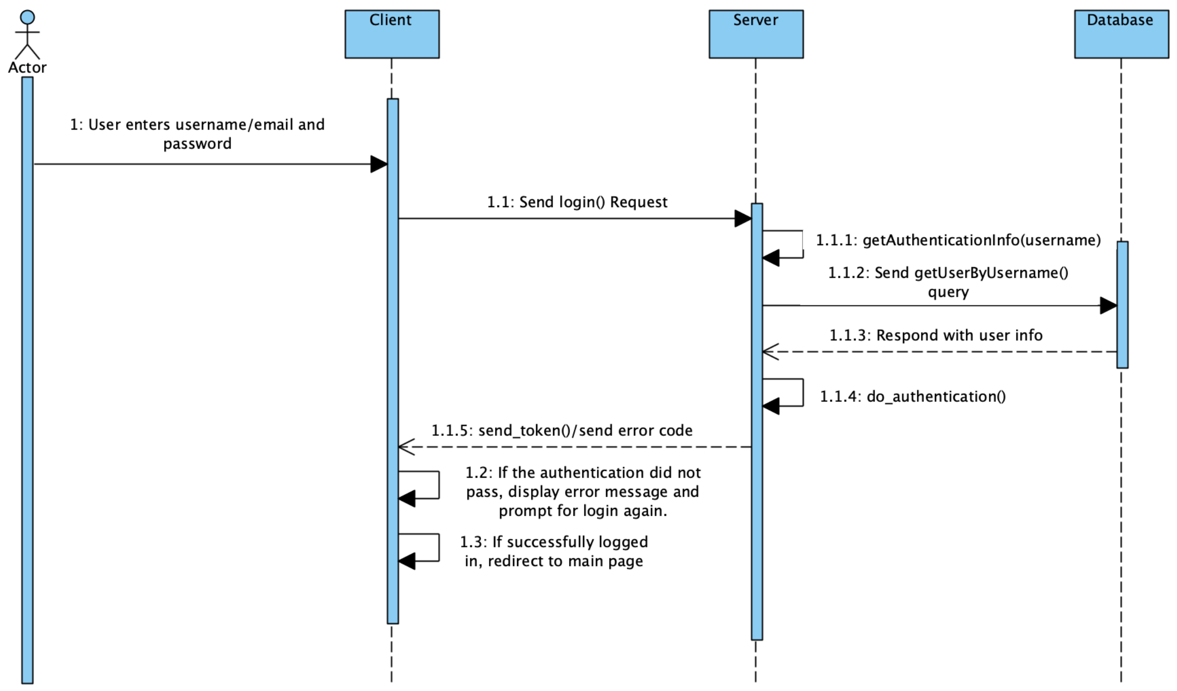
- Team:
  - Each team includes the leader’s userID
  - Each team includes a list of team members, including the leader
  - Each team has their associated code history, including both the code and any submitted pictures
  - Each team will have a name and description
  - Team leader has the authority to invite or remove any team member from the team
  - After new code is saved, users can use the update function to update their changes into the database

- Submitted_Image
  - Each Image and code will be stored in the form of URL, where it leads to the actual image and code in the database
  - Each image has the TeamID/GroupID of which team it belongs to
  - Each image stores the date of its last known modification

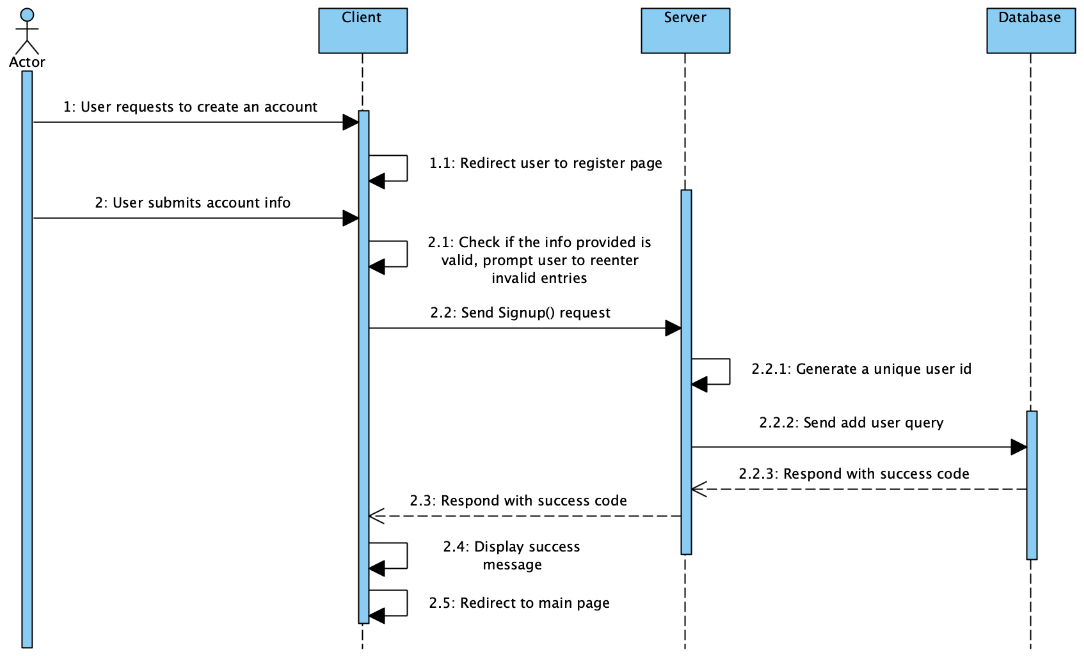
- OCR_Control_Panel
    - This is an interface used by the backend for easier development. It contains the URL of the Image needs to be processed by the OCR
    - Upon initialization, it will actively listen to a specific port for Image process request
    - Upon receiving a request, it will automatically run the OCR script to process the Image and fetch the result
    - It will send the proper output back to the user upon image process is complete
    - In this way, the back-end engineers can separate the work of initializing the scripts and handling the request, making it easier for service implementation and unit tests

Sequence Diagrams

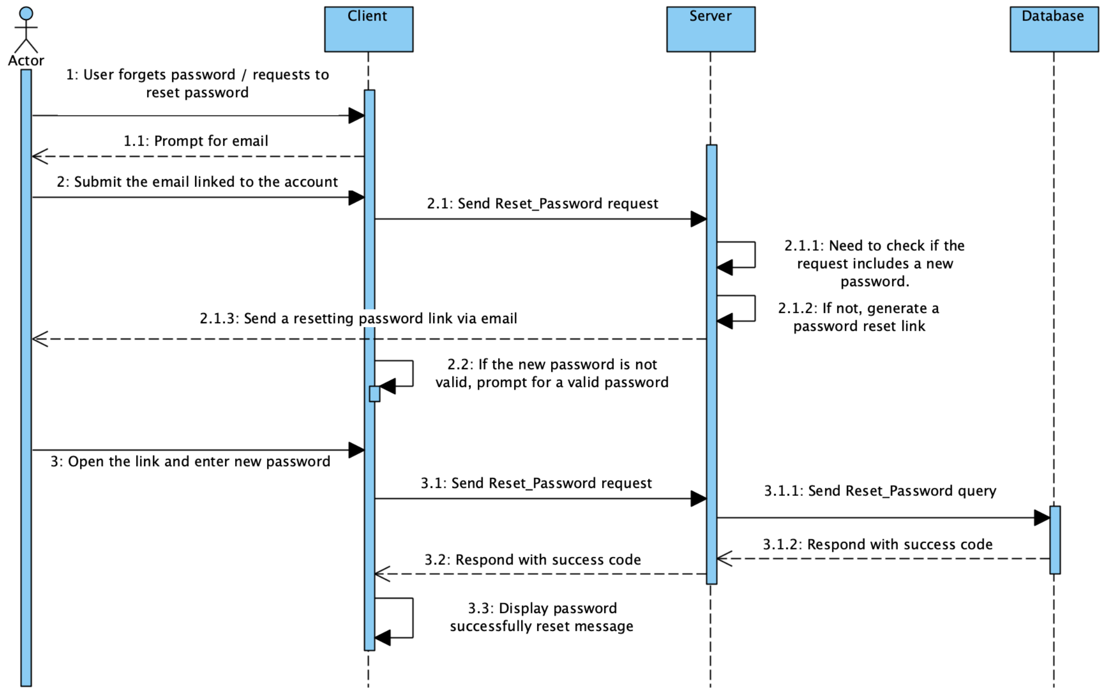
Sequence of events when users login


Actor
Client
Server
Database
1: User enters username/email and password
1.1: Send login() Request
1.1.1: getAuthenticationInfo(username)
1.1.2: Send getUserByUsername() query
1.1.3: Respond with user info
1.1.4: do_authentication()
1.1.5: send_token()/send error code
1.2: If the authentication did not pass, display error message and prompt for login again.
1.3: If successfully logged in, redirect to main page

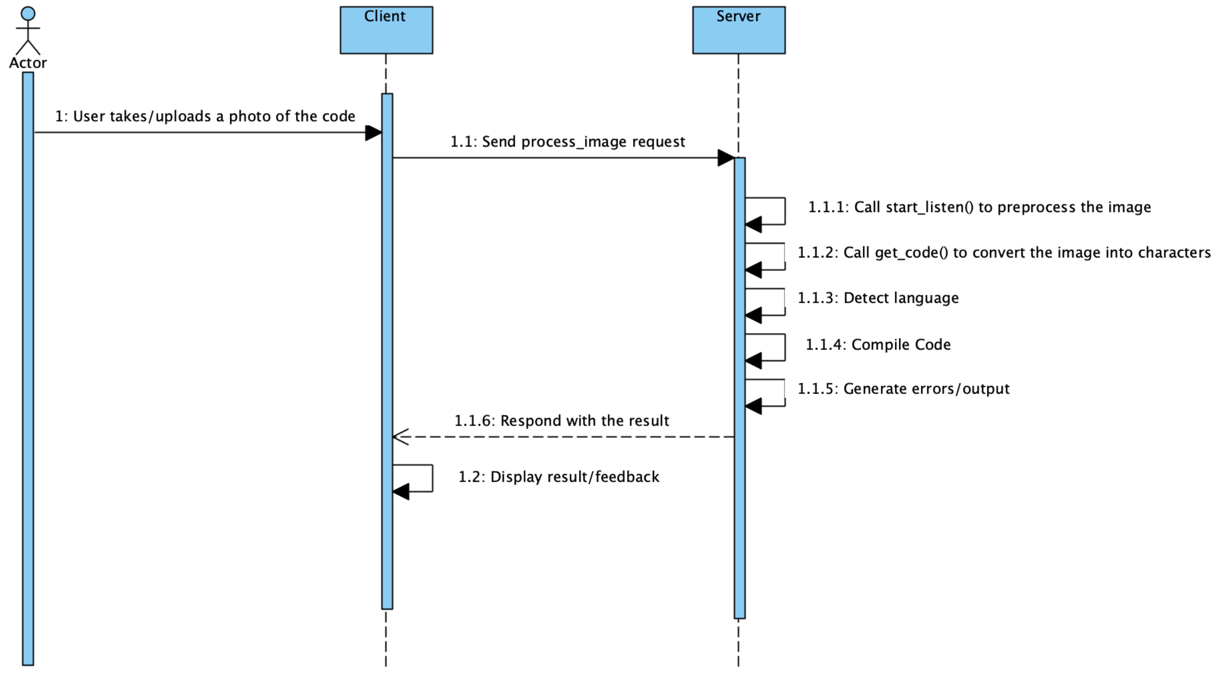
Sequence of events when a user creates a new account


Actor
Client
Server
Database
1: User requests to create an account
1.1: Redirect user to register page
2: User submits account info
2.1: Check if the info provided is valid, prompt user to reenter invalid entries
2.2: Send Signup() request
2.2.1: Generate a unique user id
2.2.2: Send add user query
2.2.3: Respond with success code
2.3: Respond with success code
2.4: Display success message
2.5: Redirect to main page

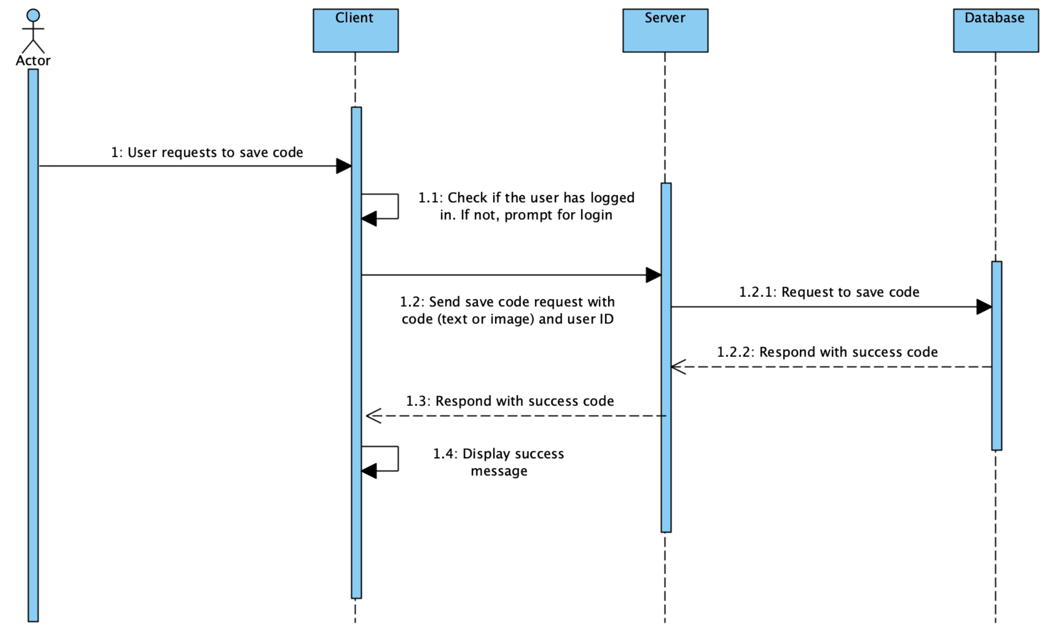
Sequence of events when a user forgets password or wants to reset password


Actor
Client
Server
Database
1: User forgets password / requests to reset password
1.1: Prompt for email
2: Submit the email linked to the account
2.1: Send Reset_Password request
2.1.1: Need to check if the request includes a new password.
2.1.2: If not, generate a password reset link
2.1.3: Send a resetting password link via email
2.2: If the new password is not valid, prompt for a valid password
3: Open the link and enter new password
3.1: Send Reset_Password request
3.1.1: Send Reset_Password query
3.1.2: Respond with success code
3.2: Respond with success code
3.3: Display password successfully reset message

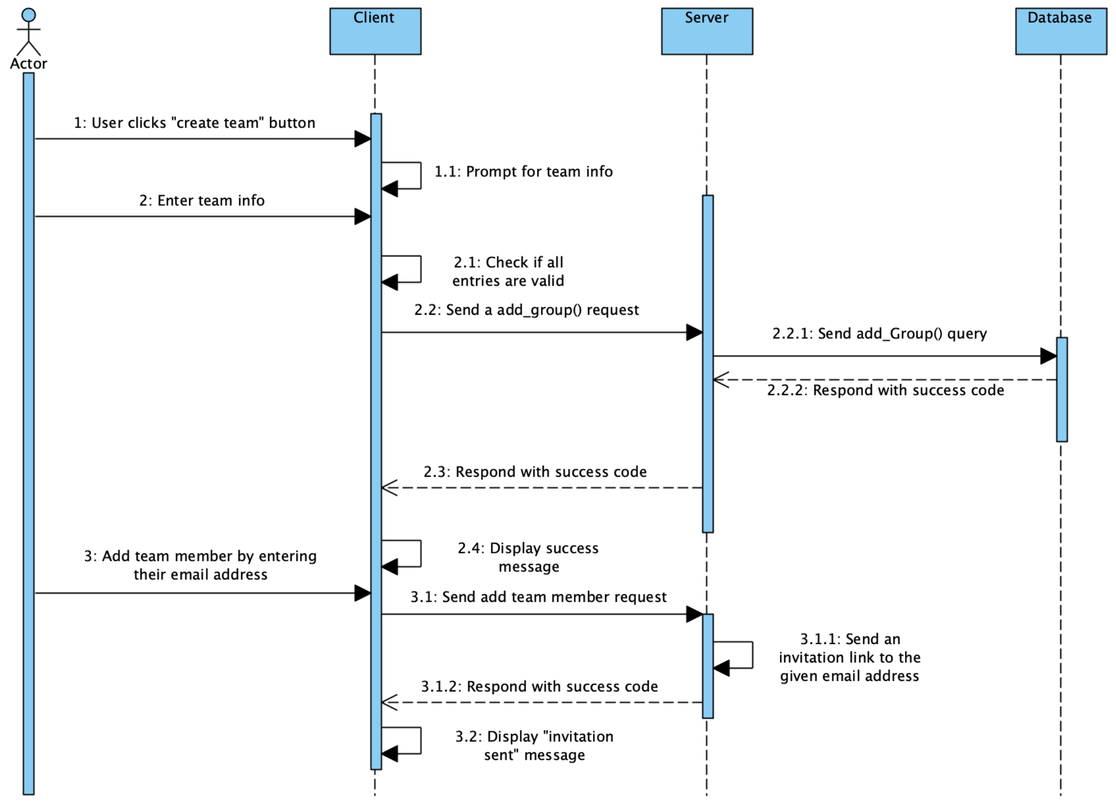
Sequence of events when a user takes/uploads a photo of code


Actor
Client
Server
1: User takes/uploads a photo of the code
1.1: Send process_image request
1.1.1: Call start_listen() to preprocess the image
1.1.2: Call get_code() to convert the image into characters
1.1.3: Detect language
1.1.4: Compile Code
1.1.5: Generate errors/output
1.1.6: Respond with the result
1.2: Display result/feedback

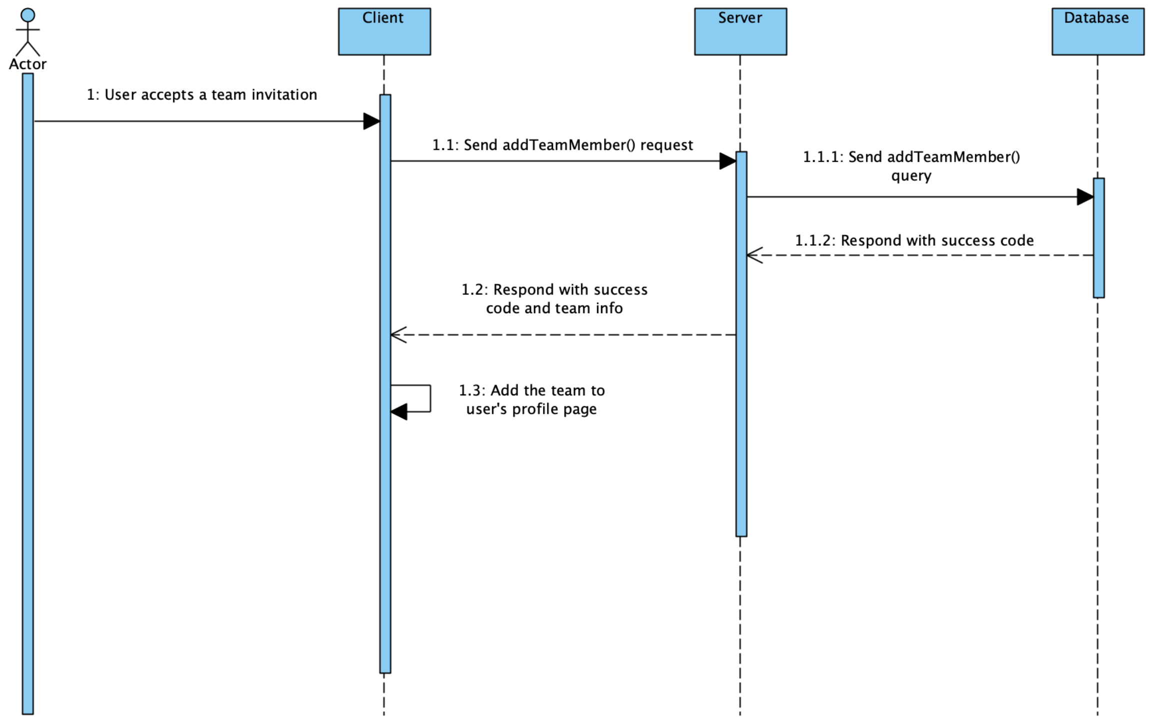
Sequence of events when a user saves code


Actor
Client
Server
Database
1: User requests to save code
1.1: Check if the user has logged in. If not, prompt for login
1.2: Send save code request with code (text or image) and user ID
1.2.1: Request to save code
1.2.2: Respond with success code
1.3: Respond with success code
1.4: Display success message

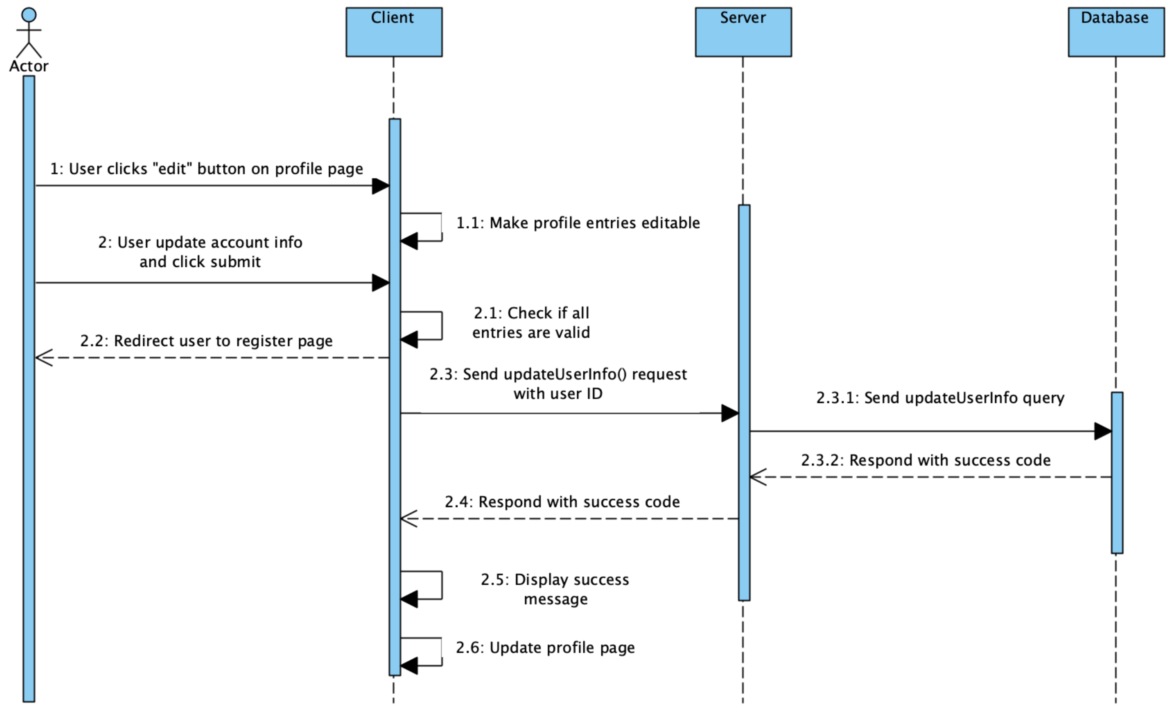
Sequence of events when a user creates a team and adds teammate


Actor
Client
Server
Database
1: User clicks "create team" button
1.1: Prompt for team info
2: Enter team info
2.1: Check if all entries are valid
2.2: Send a add_group() request
2.2.1: Send add_Group() query
2.2.2: Respond with success code
2.3: Respond with success code
2.4: Display success message
3: Add team member by entering their email address
3.1: Send add team member request
3.1.1: Send an invitation link to the given email address
3.1.2: Respond with success code
3.2: Display "invitation sent" message

Sequence of events when a user accepts an invitation to a team


Actor
Client
Server
Database
1: User accepts a team invitation
1.1: Send addTeamMember() request
1.1.1: Send addTeamMember() query
1.1.2: Respond with success code
1.2: Respond with success code and team info
1.3: Add the team to user's profile page

Sequence of events when a user changes their profile info


Actor
Client
Server
Database
1: User clicks "edit" button on profile page
1.1: Make profile entries editable
2: User update account info and click submit
2.1: Check if all entries are valid
2.2: Redirect user to register page
2.3: Send updateUserInfo() request with user ID
2.3.1: Send updateUserInfo query
2.3.2: Respond with success code
2.4: Respond with success code
2.5: Display success message
2.6: Update profile page

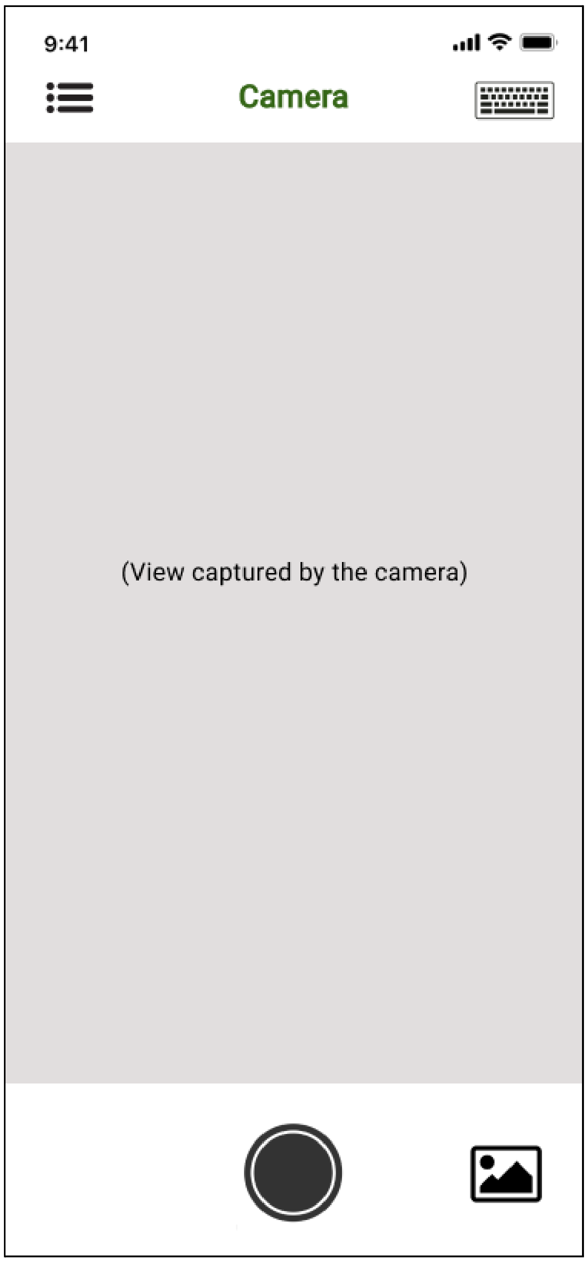
UI Mockup

Initial welcome page

When a user opens the app, this initial page with the logo of the application will be displayed.

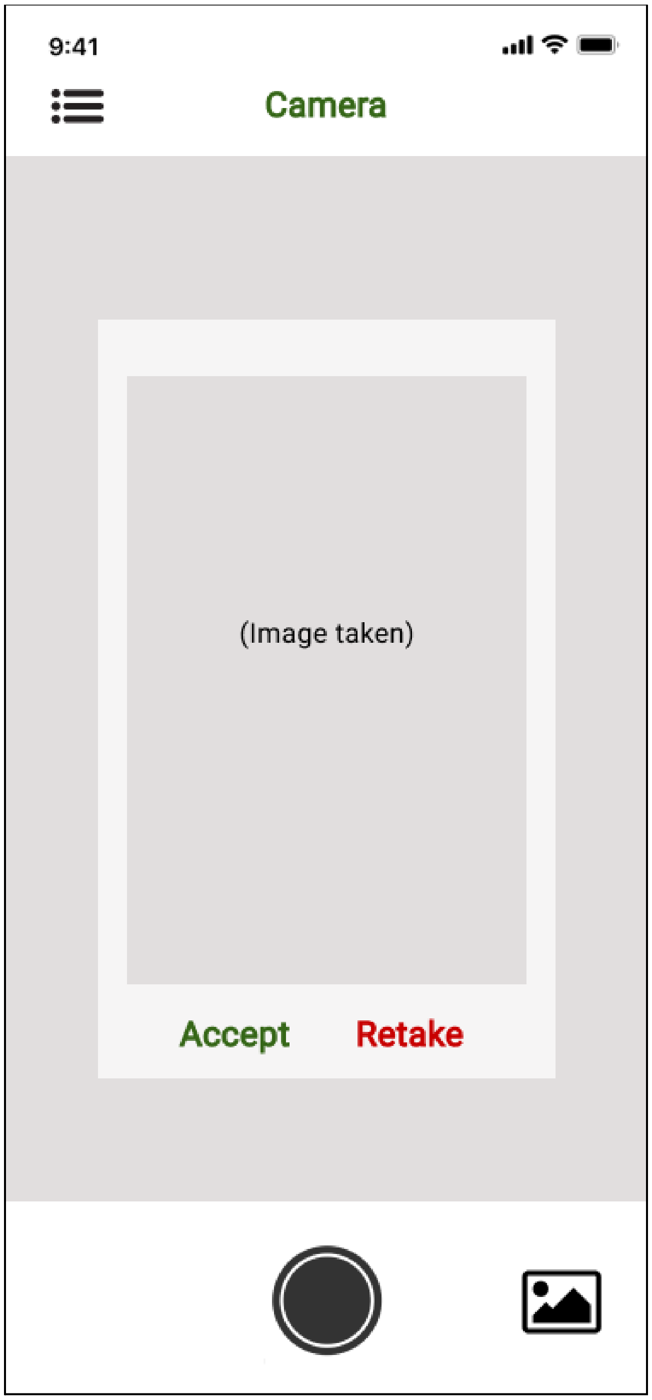
Camera page

The application then takes the user to the camera page, which is the root page of this app. Just like a normal camera app, the grey area would display the real time view captured by the camera. A user can either take a photo of the code or import a photo of the code. A user can also switch to the text editor mode that can type code directly through the keyboard icon on the upper right corner. The sidebar icon on the upper left allows users to access more features.



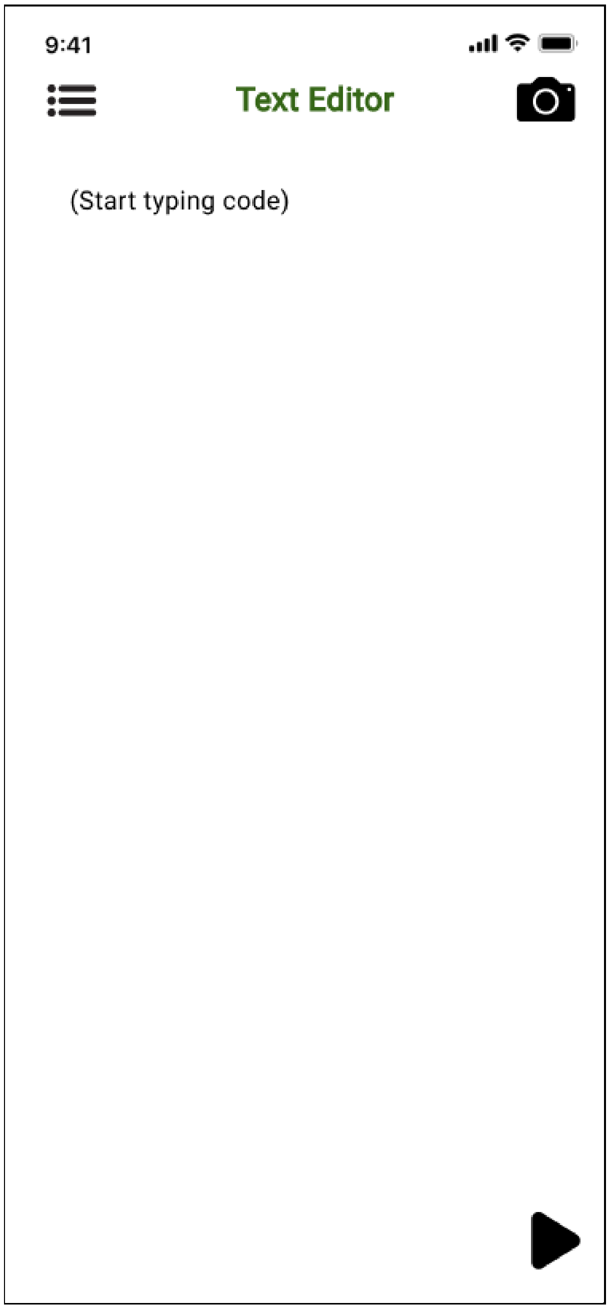
<u>Popup window after photo taken</u>

After a user takes a photo, a popup window will show up. There is the image taken, an "Accept" button, and a "Retake" button on the window. The user can look at the image taken and decide if they are satisfied with the image. If satisfied, press "Accept" and it will try to run the code. If not, press "Retake" and it will return to the camera page.



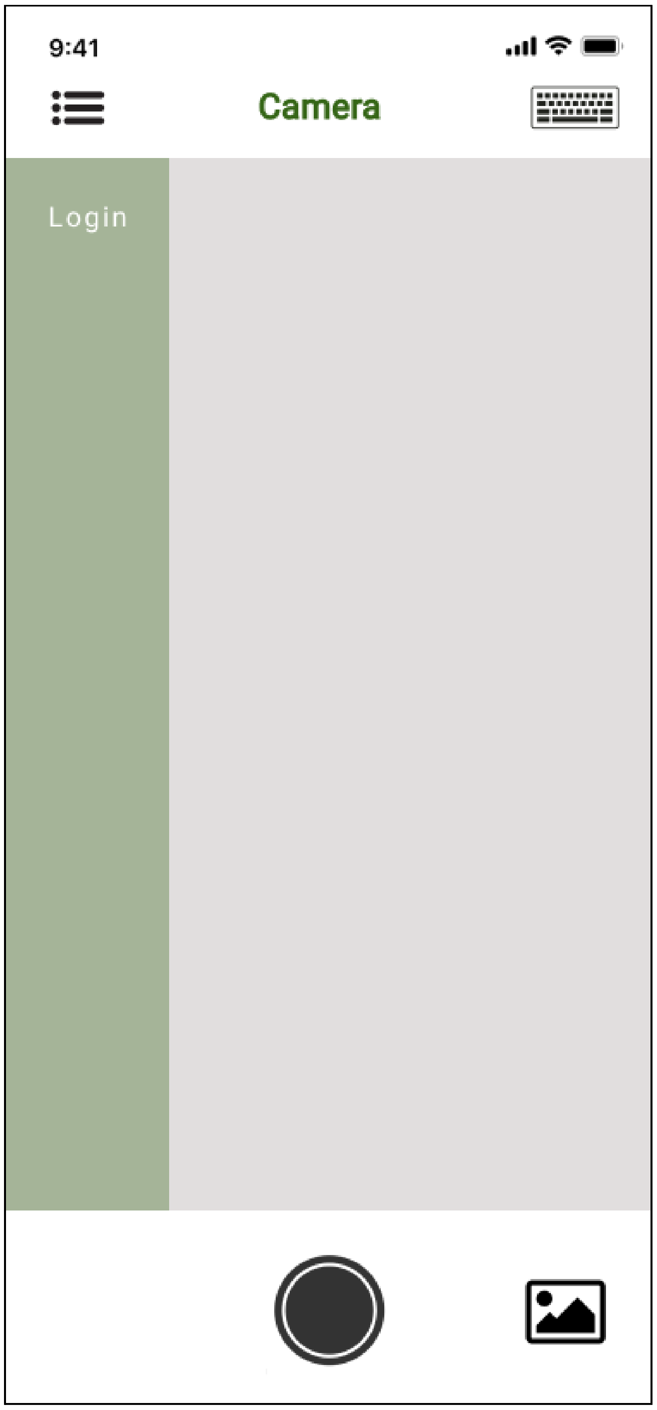
Text editor page

This page contains a text editor. A user can directly type code in this page and run it by pressing the play icon on the lower right corner. The user can switch back to the camera mode through the camera icon on the upper right corner. If an image has been processed and returned, this page will show the textual representation of the code relating to the same image.



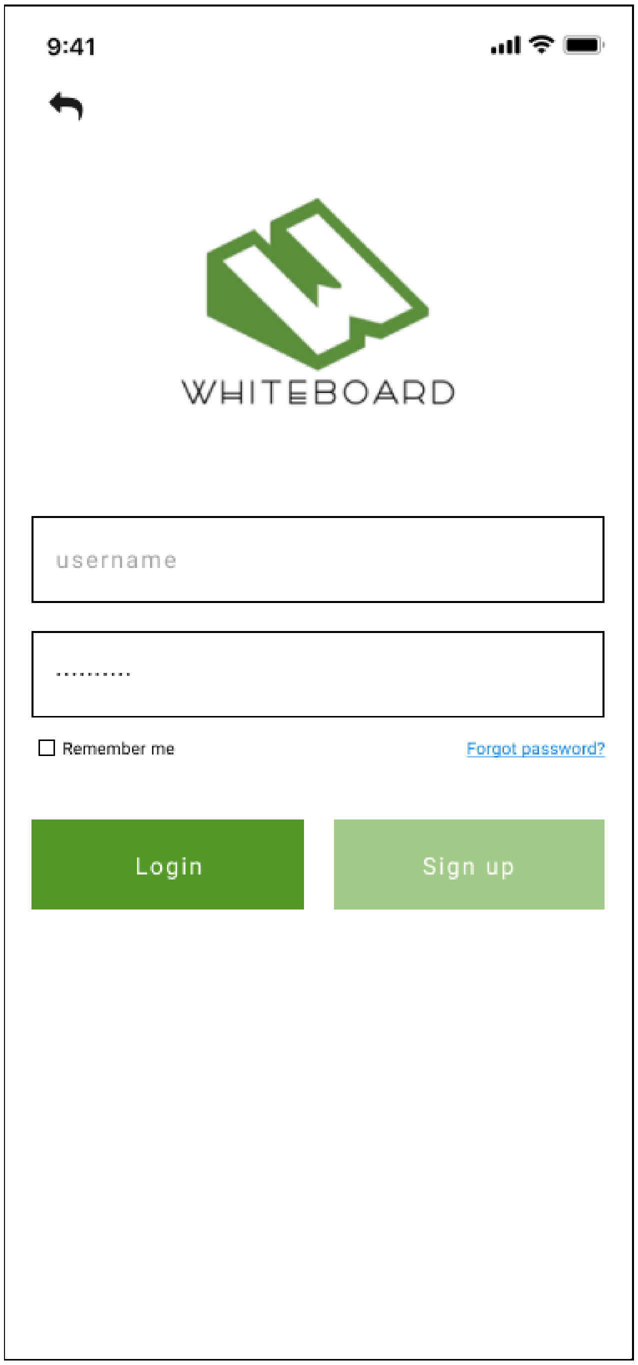
Camera page with sidebar when not logged in

If a user is not logged in, the sidebar will just show the “Login” button when the upper left icon is pressed. When the user presses the “Login” button, it will jump to the login page.



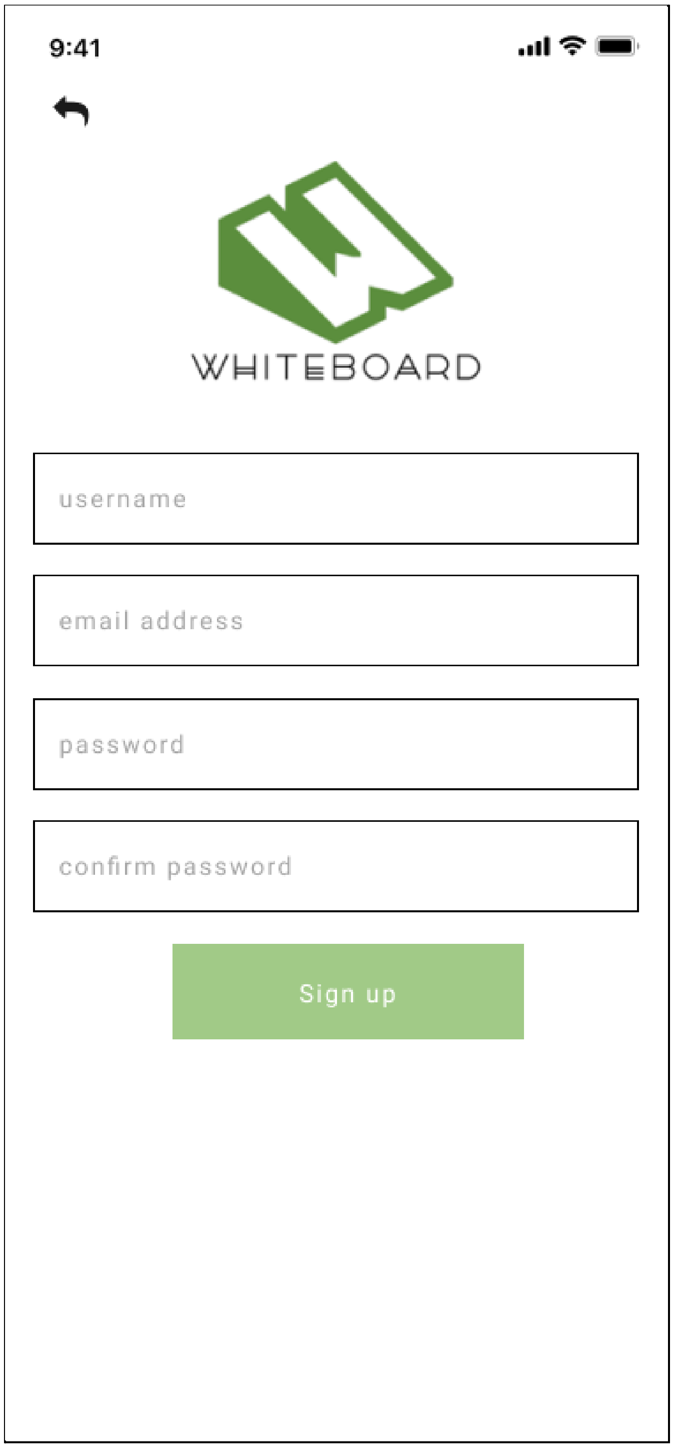
Login page

If a user has an account, they can enter their username and password to login into their account. The user can choose to stay logged in through checking the “Remember me” box. The user can also change their password by the “Forgot password” link. If a user does not have an account, they can create one by pressing the sign-up button; this will take them to the registration page.



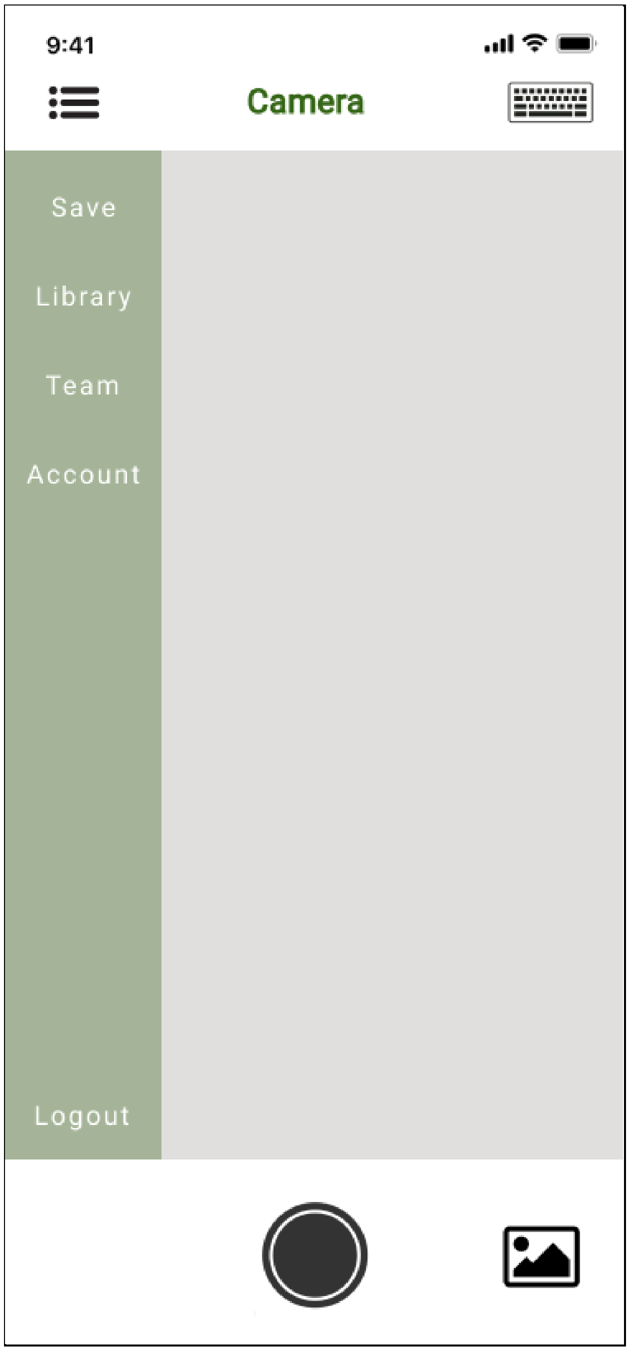
Registration page

A user will need to enter their username, email address, and password in order to register for an account. After registering, the user will be returned to the login page.



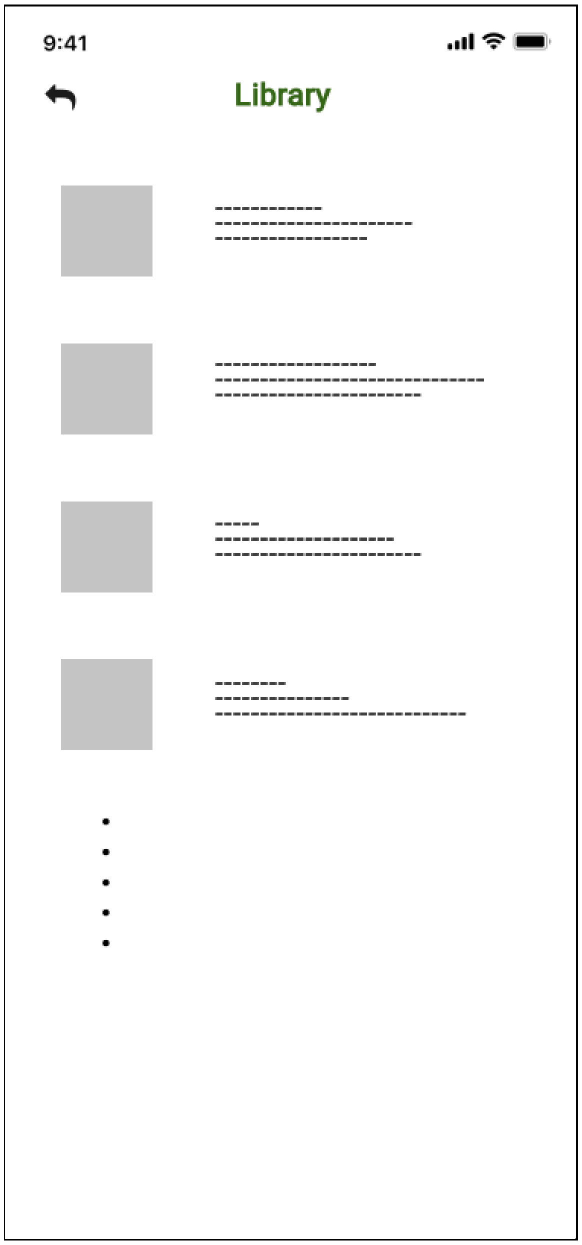
Camera page with sidebar when logged in

If a user is logged in, the sidebar will show additional features that the user can access when the upper left icon is pressed. The “Save” button will save the picture taken or the code typed to the library with the run results. The “Library”, “Team”, and “Account” buttons will jump to the library, team, and account page respectively. The “Logout” button simply logs the user out.



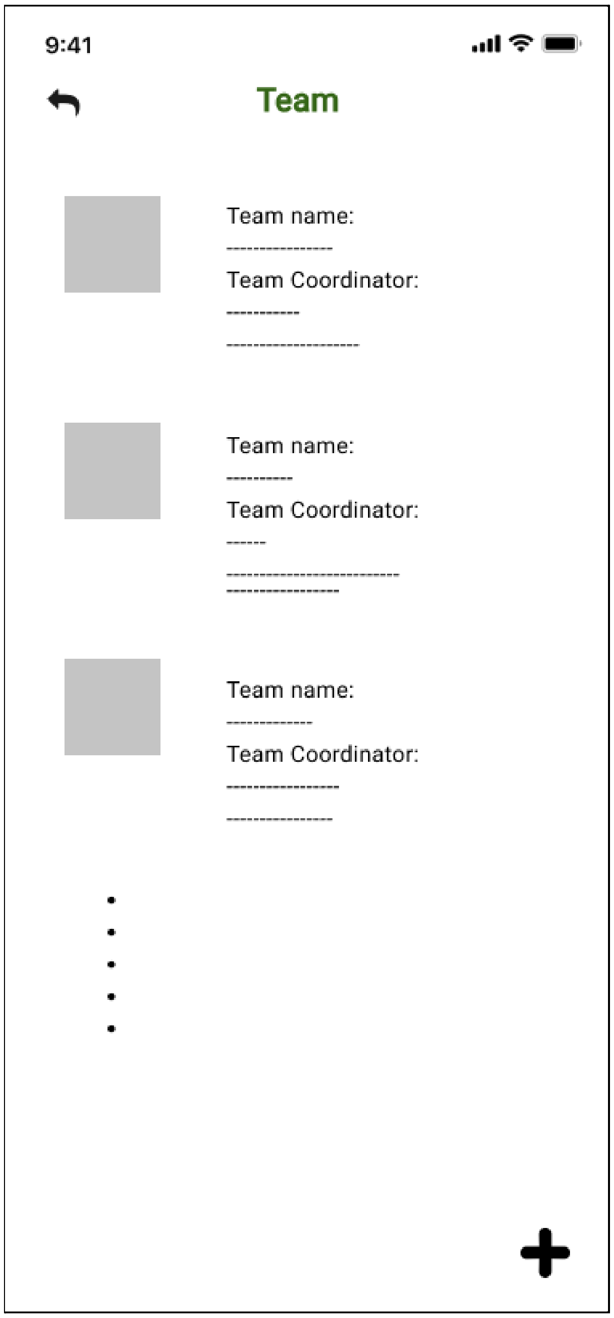
Library page

The library page displays a list of saved code of typen and picture forms. For each code, there is some basic information, such as name, date created, and short description of the code, displayed on the right of a small preview of the code segments. When the user click into a specific one, they can see the whole code and the details of the run results.



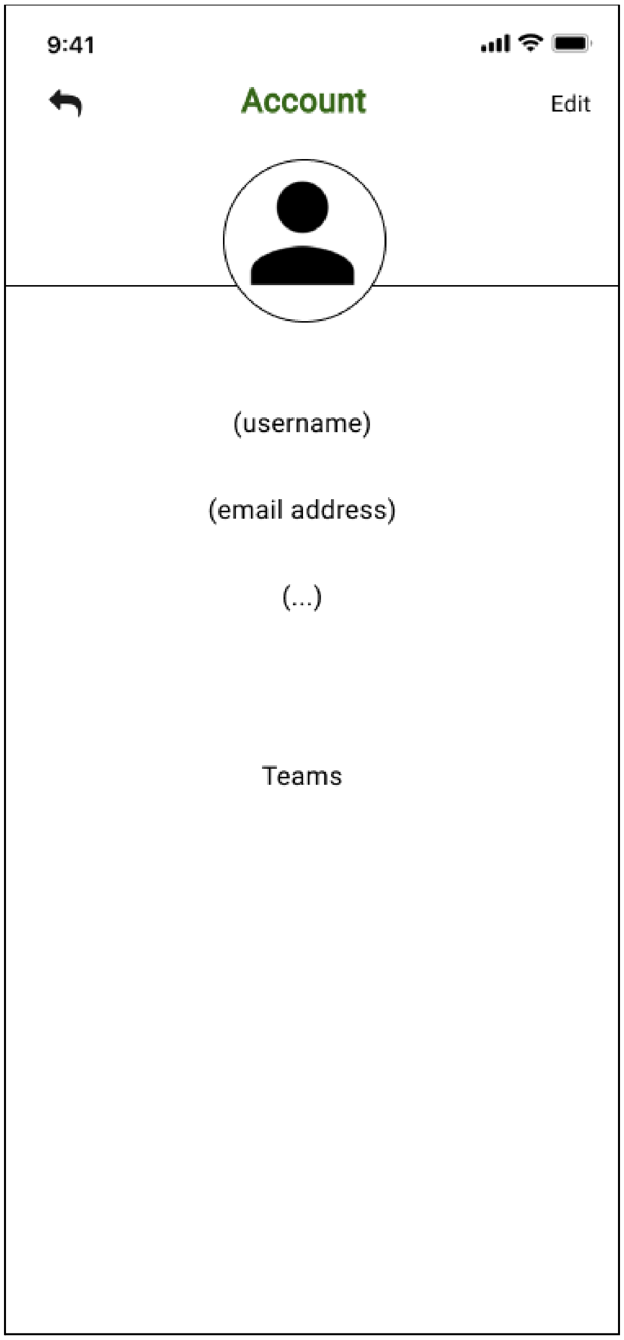
Team page

The team page shows a list of teams that a user belongs to. On the right of the team avatar, there will be the name of the team, the team coordinator(s), and some brief introduction of the team. When the user goes into a team, they can see the members, the contact information of the members and coordinators, and the code shared amongst the team. A user can create a team or accept an invitation through the plus button on the bottom right corner.



Account page

The account page displays the information of a user, like the username and the email address. The user can edit their information through the “Edit” button on the upper right corner. The teams that a user is in are displayed below the “Teams” column of the user, and the user can access the team page through there.



Database Details and Structure

The tables use MYSQL convention.

Users Table

Field	Datatype	Length	Description
uid	Unsigned Integer	20	Unique Identifier, Primary Key
username	VARCHAR	32	Account Username
email	VARCHAR	50	Email Address
phone_num	VARCHAR	64	Phone Number
password	VARCHAR	64	Encrypted Password
auth_id	VARCHAR	64	Authentication ID
default_team_id	VARCHAR	20	The default team ID
has_avatar	Boolean	1	If the user has a custom avatar
has_banner	Boolean	1	If the user has a custom banner
profile_data	String		A list of fields, one each for the different portions of a user's profile

OAuthManager Table

Field	Datatype	Length	Description
auth_id	VARCHAR	20	Primary Key
uid	VARCHAR	20	
temporary_token	VARCHAR	100	
last_login	DATETIME	null	The last login time of the user
is_logged_in	Boolean	1	
ip_address	VARCHAR	200	

User_Team Table
Each entry represents an association between a user and a team.

Field	Datatype	Length	Description
id	VARCHAR	20	Primary Key
user_id	VARCHAR	20	
team_id	VARCHAR	20	

Team Table
Each team has its own table.

Field	Datatype	LENGTH	Description
team_id	VARCHAR	20	Primary Key
leader_id	VARCHAR	20	The uid of the team leader
team_name	VARCHAR	64	
description	VARCHAR	64	The description of the team
is_default	Boolean	1	If this is a default group

Team_codeRefURL Table
Each entry represents an association between a team and a piece of code.

Field	Datatype	Length	Description
id	VARCHAR	20	Primary Key
team_id	VARCHAR	20	
code_ref_URL	VARCHAR	100	The reference url to a certain piece of code

Team_imageRefURL Table

Each entry represents an association between a team and an image.

Field	Datatype	Length	Description
id	VARCHAR	20	Primary Key
team_id	VARCHAR	20	
image_ref_URL	VARCHAR	100	The reference url to a certain image

Submitted_image Table

Field	Datatype	Length	Description
image_id	VARCHAR	20	Primary Key
image_url	VARCHAR	100	Image url
code_url			The url to the code extracted from the image
last_modify	DATETIME	null	
team_id	VARCHAR	20	The team this image belongs to