CPEN 321 Software Engineering

M6: Test Setup

UBC Explore

Group members:

Jane Shi	37998283
Xiaoqin Mei	71315980
Dylan Pither	64661267
Akshat Hari	28768299

Description:

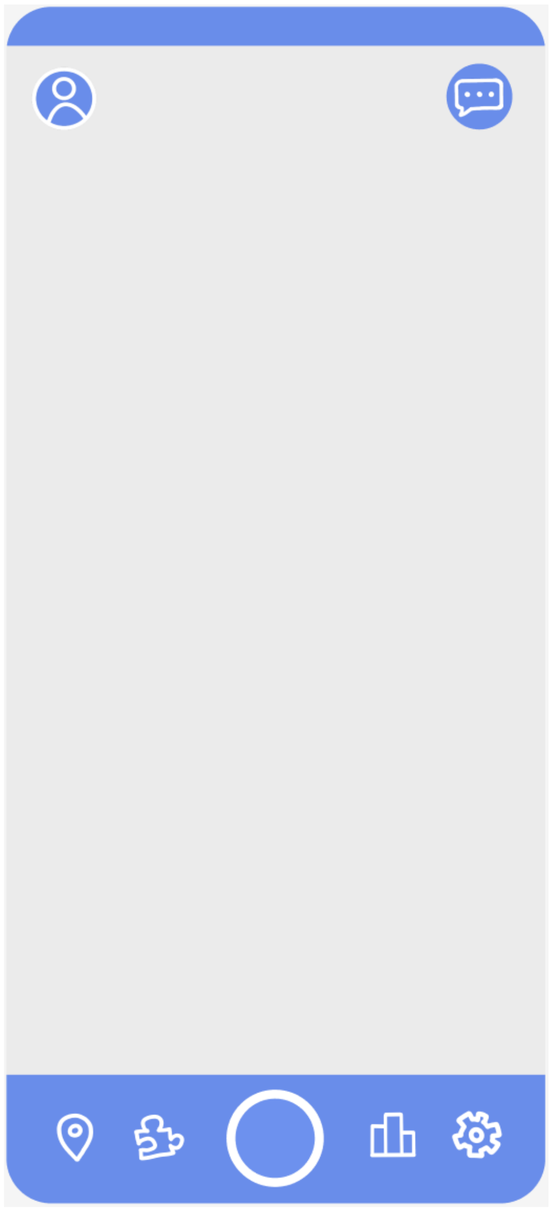
The UBC campus is large and it takes effort to explore and understand every part of it. Our app proposes to decrease the difficulty by showing the history and fun facts of UBC at the tip of your fingers. If a student or a tourist wants to know more about a part of the campus, they can select it as a destination and follow the directions on our app to get there. Once they arrive, they can direct their phone camera to the location and our app would give them a brief summary about it and direct them to related resources. To encourage people to explore, we will hide away AR cashes and custom Live2D creatures that can be found around the campus and collected for leaderboards and prizes.

- Google Maps API will be used to provide directions in our app.
- Leaderboards will be updated in real-time.

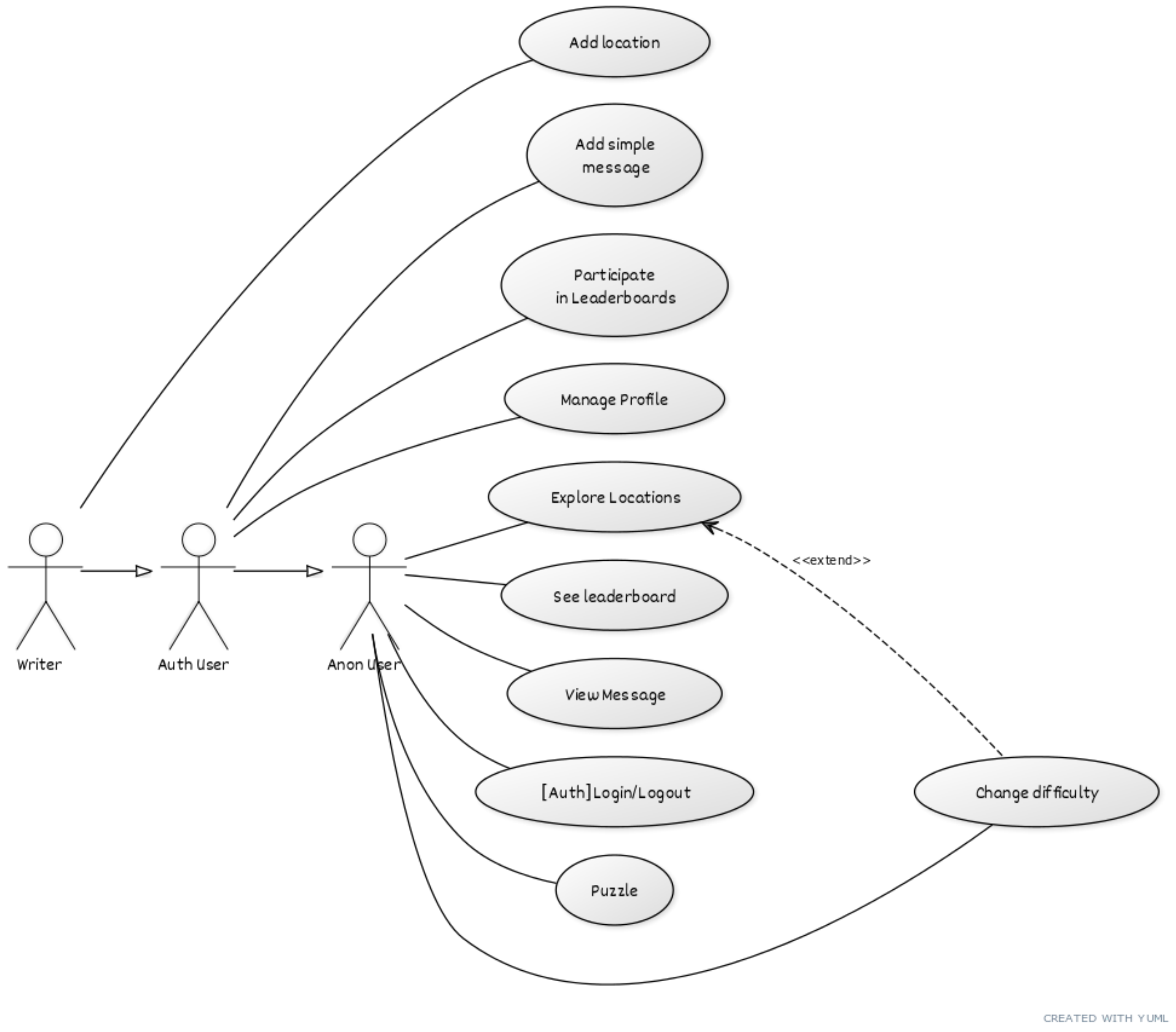
Interesting Features:

- Our app will have Artificial Reality items and custom Live2D creatures for the user to collect when they get to the location.
- Our app will also have Slick UI designs.

Main Screen Sketch:



Use Case Diagram:



Formal Use Case Specifications:

Title: Explore locations

Description: The user browses explorable locations and selects one they wish to explore. After following the provided directions to the location, information about the location is displayed.

Primary Actor: Anon User

Preconditions: None

Postconditions: The user is given information about their selected location

Main Success Scenario:

1. The user selects the location list button on the main page.
2. The app displays a list of explorable locations.
3. The user selects a location.
4. The app displays directions to the selected location using Google Maps API.
5. The user follows the directions to the location.
6. When the user arrives at the location the app displays information about the location.

Extensions:

7b. The Maps API fails

- 7b1. The application requests the user to try again.

Title: Puzzle

Description: This is an always-on feature. In the main screen with the camera, the user would find nooks and crannies to find either AR buttons or AR puzzle piece collectibles. After solving either of the puzzles they will get points in achievement as well as unlock secret locations from the location list.

Primary Actor: Anon User

Preconditions: None

Postconditions: The user solves the puzzle and gets achievements, a score and unlocks a secret location.

Main Success Scenario:

1. The user explores locations using their camera.
2. The user finds a set of AR interactable buttons
    2.1 The user presses buttons in sequence.
    2.2 The user gets achievements, a score and unlocks a secret location.
3. The user finds a puzzle piece
    3.1 The app stores the puzzle piece as user information.
    3.2 The user collects x puzzle pieces.
    3.3 The user solves the puzzle and gets achievements, a score and unlocks a secret location.

Extensions:

2a. The user does not press buttons in the correct order

- 2a1. The app displays a message saying “wrong order, please try again”.

3b. The user does not collect x puzzle pieces.

- 3b1. User puzzle piece collection will be stored in the user profile for later use.
- 3b2. Nothing happens

Title: Login/logout

Description: The user uses Google authentication to log into or out of the app.

Primary Actor: Anon User

Preconditions: To login, the user must be logged out of the app; to logout, the user must be logged into the app.

Postconditions: The user is logged into/out of the app.

Main Success Scenario:

1. The user selects the login button from the main page.
2. The app displays a dialog showing Google authentication.
3. The user can add their Google account to login.
4. The app shows the message “logged in successfully” and returns to the main page.

Extensions:

3a. Wrong credentials are entered.

- 3a1. The app displays an error message.

Title: Change difficulty

Description: The user can change their app’s difficulty by choosing between easy and medium.

Primary Actor: Anon User

Preconditions: None

Postconditions: The user has their difficulty set to the desired level.

Main Success Scenario:

1. The user selects the settings button on the main page.
2. The app displays the settings page.
3. The user selects the “change difficulty” button.
4. The user chooses between one of the two difficulty levels.
5. The user clicks “Ok”.
6. The app alerts the user of the new difficulty level that has been selected.

Extensions:

5a. The user clicks submit without clicking one of the two buttons.

- 5a1. The app requests the user to pick a difficulty level.

Title: See leaderboard

Description: The user views a leaderboard that displays authenticated users ranked by their achievements and collection score.

Primary Actor: Anon User

Preconditions: None

Postconditions: The leaderboard is presented.

Main Success Scenario:

1. The user selects the leaderboards button on the main page.
2. The app displays the global leaderboard to the user.
    - 2.1 If the user is an authenticated user they can select a friends tab on the leaderboards page.
    - 2.2 The app then displays a leaderboard where only the friends of the user are ranked.

Extensions:

2a.The app cannot retrieve the leaderboard.

- 2a1. The app displays a message stating that there was an error retrieving the leaderboard and to try again later.

2b. An anonymous user selects the friend's tab.

- 2b1. The app displays a message stating that you need to be logged in to access this feature. This message disappears after 5 seconds.

Title: View message

Description: The user can read the message left by another user at a certain location.

Primary Actor: Anon User

Preconditions: The user must be at a location where a user has left a message.

Postconditions: The message at the marked location is displayed to the user.

Main Success Scenario:

1. While walking around, the user can press on the get a message button to get notifications of any messages that are nearby.
2. The user clicks the button.
3. The app displays the message as a notification to the user.

Extensions:

2a. Internet connection error.

- 2a1. The app displays the error and asks the user to retry.

Title: Participate in leaderboards

Description: Authenticated users can share their achievements to compete with other users.

Primary Actor: Auth user

Preconditions: None.

Postconditions: The user’s number of achievements is collected and ranked with other users.

Main Success Scenario:

1. The user presses the leaderboard button on the main screen.
2. The app displays the leaderboards.
3. The user presses the “Participate in Leaderboards” button to join the ranks. Users that do not press this button will not show up on global or friend leaderboards.
4. The app collects the user’s achievement information and updates the leaderboard.

Extensions:

2a. The app cannot retrieve the leaderboard.

- 2a1. The app displays a message stating that there was an error retrieving the leaderboard and to try again later.

Title: Add a simple message

Description: The user is able to choose and add a simple 240-character message that can only be viewed by going to the same location.

Primary Actor: Authenticated User

Preconditions: None

Postconditions: A message is added to the location successfully.

Main Success Scenario:

1. The user clicks the “add message” button.
2. The app shows a form to add a message.
3. The user adds the message.
4. The user presses submit.
5. The app shows a message saying that the submission is successful.

Extensions:

1a. The user is an anonymous user.

- 1a1. The user gets an error message saying that they have to login to use this feature.
- 1a2. The user goes back to the main page.

4a. The user submits an empty message.

- 4a1. The app shows a warning message: “please add something before submitting.”
- 4a2. The user rectifies and submits.

5b. The app fails to submit.

- 5b1. Asks the user to wait a while and retry later.
- 5b2. The user waits and retries the submission.

Title: Add a location

Description: The user adds a location to the list of explorable locations.

Primary Actor: Writer

Preconditions: None

Postconditions: The provided location is added to the location list.

Main Success Scenario:

1. The user selects the add location button on the location list.
2. The app displays a form to add a location.
3. The user enters a location name, coordinates, a short description, and provides an image. Optionally they can add a creature they want to appear at the location.
4. The user clicks “Add location”.
5. The app alerts the user that the location has been successfully added.

Extensions:

4a. The user leaves any field empty

- 4a1. The app alerts that all fields must be filled out, displaying which fields are empty.

4b. The user provides invalid coordinates.

- 4b1. The system alerts the user that the provided coordinates are invalid and a range that they should be in.

4c. The user enters illegal characters in the location name or description.

- 4c1. The system alerts the user of the illegal characters that the fields contain.

4f. The user does not provide a creature.

- 4f1. The app alerts the user that a default creature will be used asking the user if this is ok.
- 4f2. If the user clicks no they return to the form where they can upload a creature.

5a. The app fails to add the location

- 5a1. The app tells the user to wait to try again.

Title: Manage profile

Description: The user will be able to edit their own profile, view a progression tab for their collectible and puzzle collection, and add authenticated users as friends.

Primary Actor: Authenticated User

Preconditions: None

Postconditions: The user successfully managed/viewed desired information on their profile.

Main Success Scenarios:

1. The user clicks on the profile button on the main page.
2. The app displays the user's collection of creatures and any puzzle pieces they have obtained. The completeness of their collection is also displayed.
3. The user can click the edit display name button on their profile page to change their display name.
    - 3.1 The app displays a text box.
    - 3.2 The user types in their desired name.
    - 3.3 The user clicks “confirm”.
    - 3.4 The app displays their profile with the updated name.
4. The user can click the friends tab on the profile page to view their friends list.
    - 4.1 The app displays the user's friends list.
    - 4.2 The user clicks the “+” button on the friends list page.
        - 4.2.1 The app displays a text box prompting the user to enter a display name.
        - 4.2.2 The user enters their friend's display name.
        - 4.2.3 The user clicks “Send request”.
        - 4.2.4 The app adds the request to the user's outgoing requests and to their friend’s incoming requests.
    - 4.3 The user clicks the “outgoing requests” tab on the friends list page.
        - 4.3.1 The app shows a list of display names corresponding to unanswered friend requests.

4.4 The user clicks the “incoming requests” tab on the friends list page.

 4.4.1 The app shows a list of display names corresponding to incoming friend requests from other users.

 4.4.2 The user can accept or decline these requests by clicking on the display name and choose accept or decline friend request.

4.5 The user clicks “X” on the friends list page.

 4.5.1 The app displays a text box prompting the user to enter a display name.

 4.5.2 The user enters their friend's display name.

 4.5.3 The user clicks “Remove”.

 4.5.4 The app removes the friend.

Extensions:

2a. There are no creatures or puzzle pieces collected.

- 2a1. The app displays an empty page with 0/x creatures/puzzle pieces found.

3a. Network error

- 3a1. The app displays a message indicating there was a network error when trying to update the display name.

3.3a. The desired name is taken.

- 3.3a1. The app displays a message stating the desired name has already been taken and to try a different name.

3.3b. The desired name is too long or too short.

- 3.3b1. The app displays a message stating the name must be 3-20 characters.

4a. Network error

- 4a1. The app displays a message indicating failure to add/remove friend

4.1a. There are no friends of the user.

- 4.1a1. The app displays that the user has no friends and shows a button that leads to the add friends screen.

4.2.3a. The entered user does not exist.

- 4.2.3a1. The app displays that there is no user named x and to try again.

4.3a. There are no outgoing requests.

- 4.3a1. The app displays text stating that there are no outgoing friend requests.

4.4a. There are no incoming requests.

- 4.4a1. The app displays text stating that there are no incoming friend requests.

Non-Functional Requirements:

[Usability] The user should not need more than 5 clicks to perform any action.

- This requirement is relevant because it ensures that the user has fast access to all of the functionality of the app which is important for the user experience.
- We plan to test this requirement by going through each of the use cases and make sure they all need no more than 5 clicks to perform.

[Performance] AR/relevant information should show up within 1 second after the user points their phone in a certain direction at a certain location.

- This requirement is relevant because showing AR/relevant information about the location is a major functionality of our app and we don't want there to be a significant delay when showing this information.
- We plan to test this requirement by going to each of the locations and measuring the time it takes for the AR/information to show up.

[Energy efficiency] The battery life should not drop by more than 1% after continuous usage of the app for 5 minutes.

- This requirement is relevant since our app is an outdoor-based app requiring our app to be constantly active, while requiring location and camera services at the same time, which both have high energy consumption.
- We plan to test the drainage using Battery Historian for virtual deployment and battery manager apps like AccuBattery for live deployment. We will test the app under heavy use (Constantly display AR image).

[User-friendly Interface] The UI should be simple and easy to use. The icons of the buttons should be able to suggest what they are used for.

- This requirement is relevant since our app's target users are students and tourists on UBC campus. The app's interface should be easy for them to understand.
- We plan to find several students who have never used the app before and introduce the general functionality of the app. After that, we will ask them to match the usages to each button.

Android Device:

Our group has three Android devices running Android versions 9,10 and 12 which can run the front-end app.

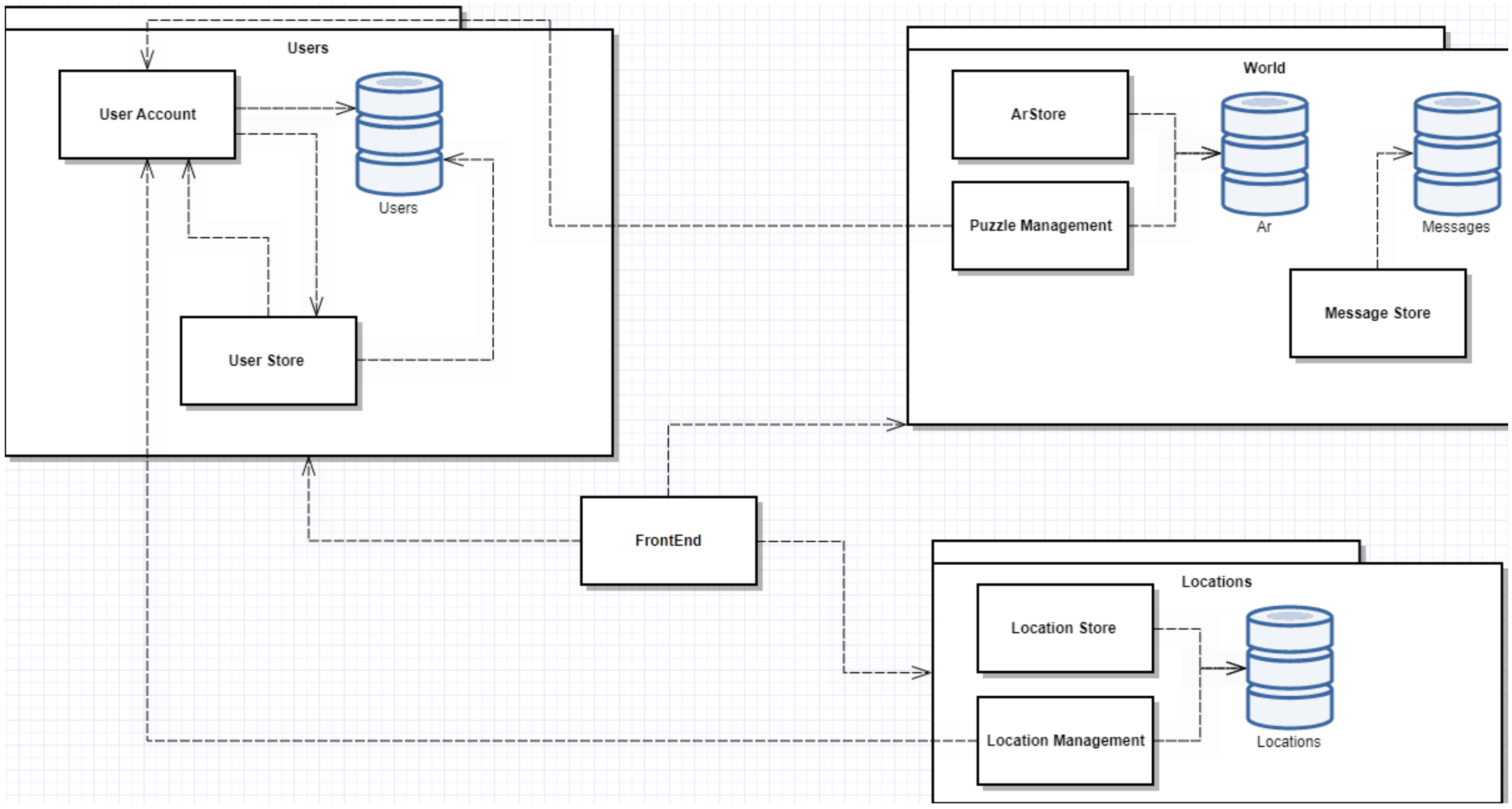
Main Modules and Sub-modules:

- Main modules:
    - Users
        - Purpose: Management of everything linked to user accounts including their collected creatures, puzzle pieces and friends. Tracks user preferences such as difficulty and if they are participating in leaderboards. Also manages global and friend leaderboards.
        - Sub-modules:
            - UserAccount
            - UserStore
    - World
        - Purpose: Management/Display of Artificial reality models and messages. Management of detailed location information as well as puzzles
        - Sub-modules:
            - ARStore
            - MessageStore
            - LocationStore

- Databases:
    - Users
        - Table to store general authenticated user information:
            - Display name, id, leaderboard participant, difficulty, score
        - Table to store friendships/requests:
            - Link two user ids with a status to represent a request or friendship
        - Table to store user collections:
            - Link an item id to a user id
        - Table to store user unlocked locations:
            - Link a location name to a user id
        - Table to store user achievements:
            - Link an achievement to a user id.
    - Messages
        - Messages sent by authenticated users into the world
        - Links one way into accounts
    - AR
        - Holds all AR components
            - Puzzles
            - Live2d models, motions and expressions
    - Locations
        - Holds information about locations
            - Location histories, fun facts, geolocation

- External components:
    - Google Authentication
        - Purpose: Get authentication for google service API
    - Google Maps
        - Purpose: Built-in Google Maps to set puzzle locations and guide users to the location

Module Dependency Diagram:



List of Interfaces:

- UserStore:
    - UserAccount findByName(String displayName)
        - Required when wanting to find another user to add as a friend or to check for a unique display name. Returns the account.
    - UserAccount createAccount(TokenInfo credentials)
        - Required if there is no existing user for the provided token. Returns the created account
    - UserAccount findById(String user_id)
        - Required to find UserAccounts from given ids.
    - UserAccount login(TokenInfo credentials)
        - Required after user validates their login token. Returns the account of the user (creating it if it does not exist).

- UserAccount login(String token)
    - Required when a user wants to retrieve their account after logging into their Google account and receiving a token. Returns their account provided the token is valid.
- UserAccount changeDifficulty(String user_id, String difficulty)
    - Required when a user wants to change their difficulty. Returns the updated account.
- UserAccount setDisplayName(String user_id, String name)
    - Required when a user wants to change their name. Returns the updated account.
- UserAccount addFriend(String user_id, String friendName)
    - Required when a user wants to send a friend request to another user. Returns the updated account, friendName is unique.
- UserAccount removeFriend(String user_id, String friendName)
    - Required when a user wants to remove another user from their friend's list. Returns the updated account, friendName is unique.
- UserAccount acceptRequest(String user_id, String friendName)
    - Required when a user wants to accept another user's friend request. Returns the updated account, friendName is unique.
- UserAccount denyRequest(String user_id, String friendName)
    - Required when a user wants to deny another user's friend request. Returns the updated account, friendName is unique.
- UserAccount participateInLeaderboard(String user_id)
    - Required when a user wants to participate in leaderboards (friend or global) which by default they are not. Returns the updated account.
- UserAccount unlockLocation(String user_id, LocationInfo location)
    - Required when a user unlocks a location from solving a puzzle. Returns the updated account.
- UserAccount unlockItem(String user_id, ARModel item)

        - Required when a user gets a new AR item (creature or puzzle piece). Returns the updated account.
    - UserAccount updateAchievements(String user_id, AchievementInfo achievement)
        - Required when a user gets a new achievement or makes progress on an achievement. Returns the updated account.
    - List getFriendsLeaderboard(String user_id)
        - Required when a user wants to view their friends leaderboard. Returns the list of only friends ordered by their score.
    - List getGlobalLeaderboard()
        - Required to retrieve the global leaderboard. Returns the top 100 account names ordered by their score.
- UserAccount:
    - UserAccount changeDifficulty(String difficulty)
        - Required to change the difficulty level of an account. Returns the updated account.
    - UserAccount setDisplayName(String name)
        - Required to change an account's display name. Returns the updated account.
    - List getFriends()
        - Required when retrieving a user’s friends list so it can be viewed in their profile or for use in the leaderboard.
    - UserAccount addFriend(String friendName)
        - Required for an account to send a request to another account. Returns the updated account, friendName is unique.
    - UserAccount removeFriend(String friendName)
        - Required for an account to remove a friend from its friends list. Returns the updated account, friendName is unique.
    - UserAccount acceptRequest(String friendName)
        - Required for an account to accept a friend request from another account. Returns the updated account, friendName is unique.

    - UserAccount denyRequest(String friendName)
        - Required for an account to deny a friend request from another account. Returns the updated account, friendName is unique.
    - UserAccount participateInLeaderboard()
        - Required for an account to participate in leaderboards (friend or global) by default accounts are not participating in leaderboards. Returns the updated account.
    - UserAccount unlockLocation(LocationInfo location)
        - Required for an account to add a location to its unlockedLocation list. Returns the updated account.
    - UserAccount unlockItem(ARModel item)
        - Required for an account to add AR item ids to its item list. Returns the updated account.
    - UserAccount updateAchievements(AchievementInfo achievement)
        - Required for an account to update its achievements list upon new acquiring new achievements or making progress. Returns the updated account.
    - List getFriendsLeaderboard()
        - Required for an account to retrieve the leaderboard of a users friends. Returns the list of friends ordered by their score.
- Users Database:
    - Int updateAchievements(AchievementInfo)
        - Required when a user gets to a new location and their achievements need to be updated
    - Int updateCollection(Collection)
        - Required when a user gets new AR collections to be updated in their profile.
    - Int updateUserLocation(Location,UserAccount)
        - Required when the user unlocks a location from solving puzzles
    - result createFriendRequest(String user_id, String friend_id)

        - Required when a user sends a friend request to another user. Returns an object describing the result of the database insertion.
    - result removeFriendship(String user_id, String friend_id)
        - Required when a user wants to remove another user from their friends list. Returns an object describing the result of the database deletion.
    - result acceptFriendRequest(String user_id, String friend_id)
        - Required when a user wants to accept a friend request, changing the status from pending to friends. Returns an object describing the result of the database update.
    - result removeFriendRequest(String user_id, String friend_id)
        - Required when a user wants to deny a friend request. Returns an object describing the result of the database deletion.
    - result updateDifficulty(String user_id, String difficulty)
        - Required when a user wants to change the difficulty. Returns an object describing the result of the database update.
    - result participateInLeaderboard(String user_id)
        - Required when a user requests to participate in leaderboards. Returns an object describing the result of the database update.
    - List getGlobalLeaderboard()
        - Required when a user wants to view the global leaderboard. Returns a list of up to 100 names ordered by their score.
    - List getFriendsLeaderboard(List friends)
        - Required when a user wants to view their friends leaderboard. Returns a list of only accounts in the provided list ordered by their score.
    - result updateDisplayName(String user_id, String name)
        - Required when a user wants to change their name. Returns an object describing the result of the database update.

- MessageStore:

    - Message getMessage(coordinates)
        - Required when a user gets to a certain coordinates, the app retrieves messages left at the location.
    - Int addMessage(Message)
        - Required when an authenticated user wishes to add a message at a certain location.
- ARStore:
    - ARModel getARModel(coordinates)
        - Required when a user gets to a certain coordinates, the app retrieves AR models at the location.
    - Int addARModel (ARModel)
        - Required when a writer wants to add an AR model to the app.
- LocationStore:
    - LocationInfo getLocationInfo(coordinates)
        - Required when a user gets to certain coordinates, the app retrieves location information.
    - Int addLocation(LocationInfo, ARModel)
        - Required when a writer wants to add a location to the app.
    - List getLocationList(UserAccount)
        - Required when a user wants to view the list of explorable locations
    - Void getDirections(String locationName)
        - Required when a user wants to get directions to a location, will either open google maps directions to the location or show an image depending on user difficulty level.
- Locations Database:
    - LocationInfo getLocationInfo(coordinates)
        - Required when a user gets to certain coordinates, the app retrieves location information.
    - Int addLocationInfo(LocationInfo)
        - Required when a writer wants to add a location to the app.
    - LocationInfo getLocation(String locationName)

        - Required when a user wants to get directions to a selected location.
    - List getLocations()
        - Returns the list of explorable locations
- AR Database:
    - ARModel getARModel(coordinates)
        - Returns the AR model at the location coordinates
    - Int addARModel(ARModel)
        - Required to add AR models to the database.
- Messages Database:
    - Message getMessage(coordinates)
        - Required when a user gets to a certain coordinates, the app retrieves messages left at the location.
    - Int addMessage(Message)
        - Required when an authenticated user wishes to add a message at a certain location.

DataTypes:

TokenInfo (https://developers.google.com/identity/sign-in/android/backend-auth){

String: iss,

String: sub (unique id),

String: azp,

String: aud,

String: iat,

String: exp,

String: email,

String: email_verified,

String: name,

String: picture,

String: given_name,

String: family_name,

String: locale

}

AchievementInfo{

String id,

String Type,

Int points,

String Image.Id,

}

ARModel (https://www.live2d.com/en/download/sample-data/) {

Int id,

Model data (cmo3)

Basic motions (can3)

Set of files for embedding (runtime folder)

- Model data (moc3)
- Motion data (motion3.json)
- Model settings file (model3.json)
- Physics settings file (physics3.json)

```
• Display auxiliary file (cdi3.json)
}
Coordinates{
int : longitude
Int: latitude
}
LocationInfo{
Int id,
Coordinates,
String location_name,
String: Fun Facts,
String: Related Links,
String: About,
String: Image,
UserAccount: creator,
String: public_access:
}
Message{
Int id,
Coordinates,
String: message,
UserAccount: creator,
}
UserAccount{
Collection{
List[AchievementInfo] achievements,
List[ARModel.id] items,
}
List[UserAccounts.displayName] friends,
List[LocationInfo.location_name] unlockedLocations,
String: difficulty,
Boolean: leaderboardParticipant,
String: displayName,
List[UserAccounts.displayName]: incomingRequests,
List[UserAccounts.displayName]: outgoingRequests,
int: score,
String: id
}
```

Sequence Diagrams:

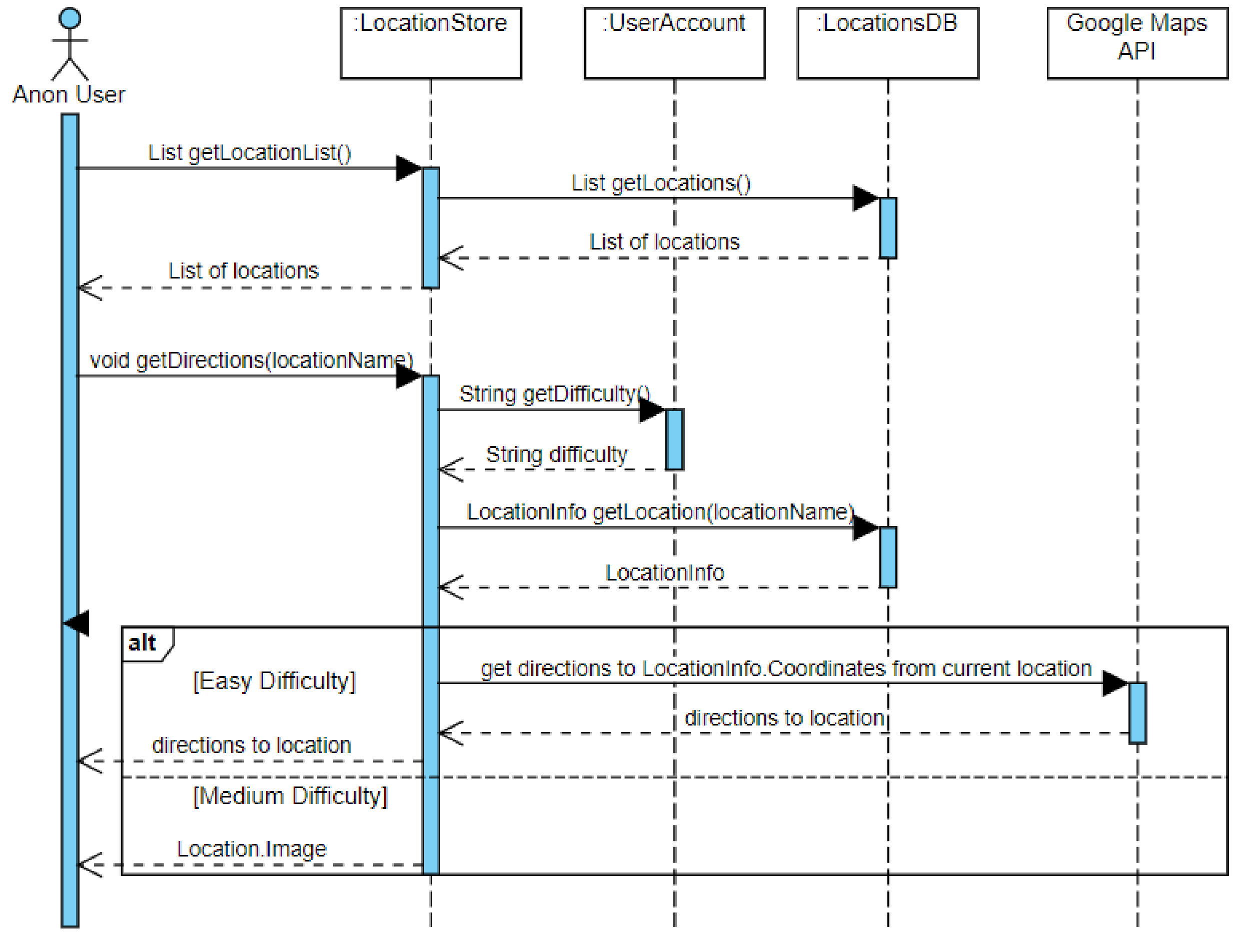
We assume the following meaning of return values in our sequence diagram:

1: Success

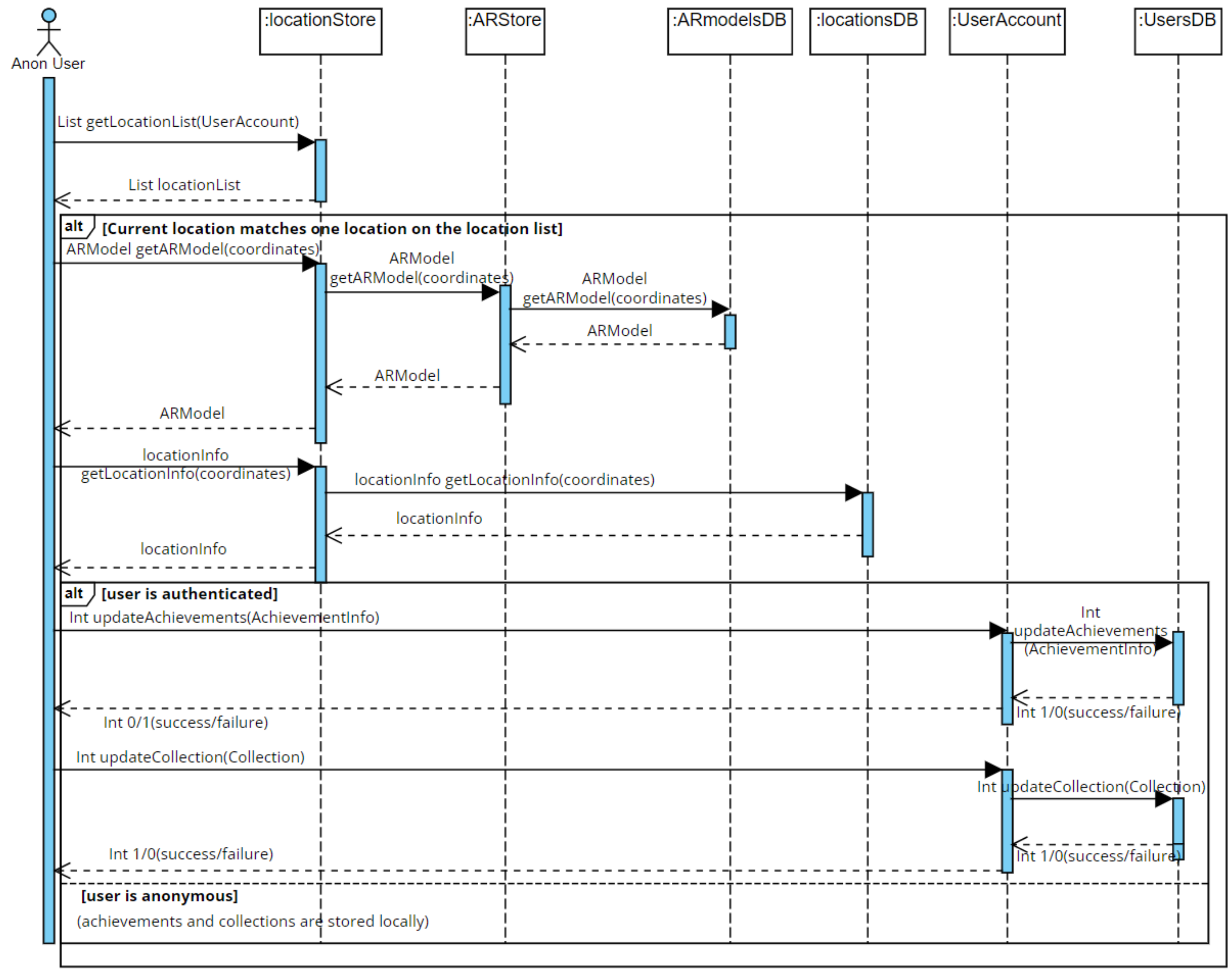
0: Failure

- **Explore locations**

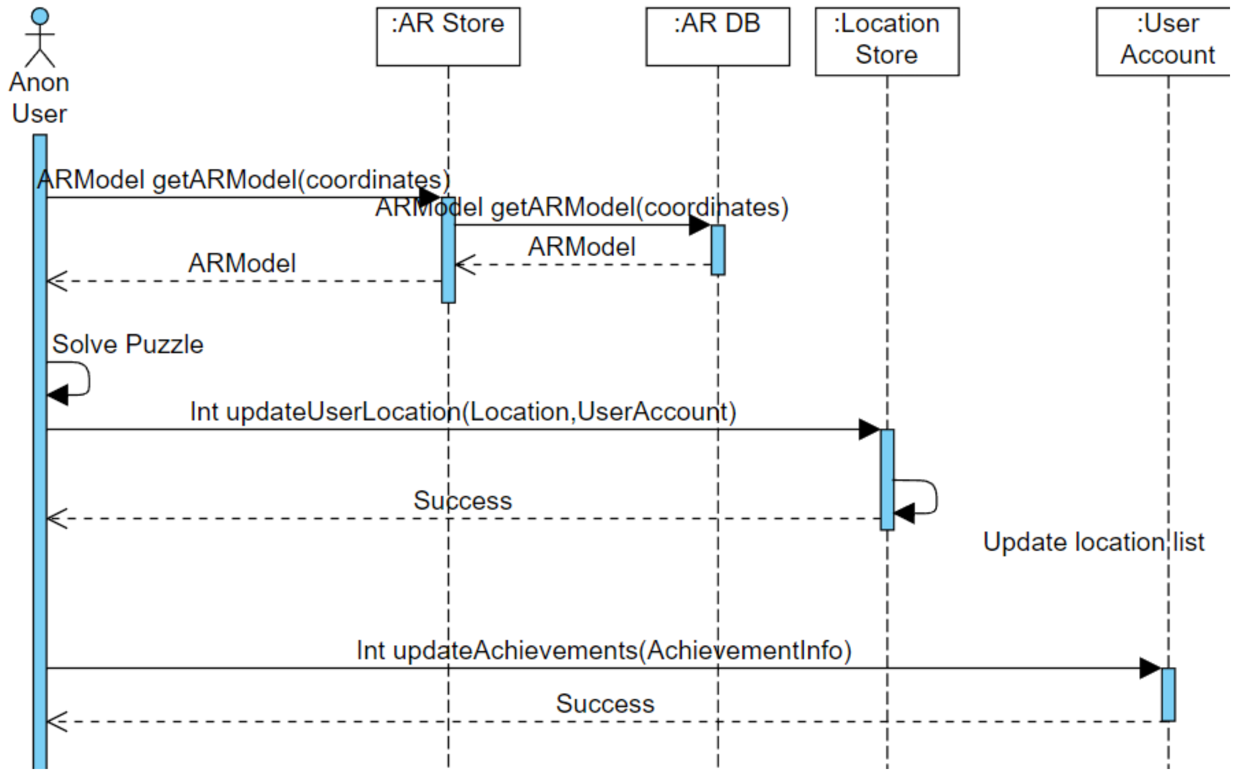
Getting directions to the location:



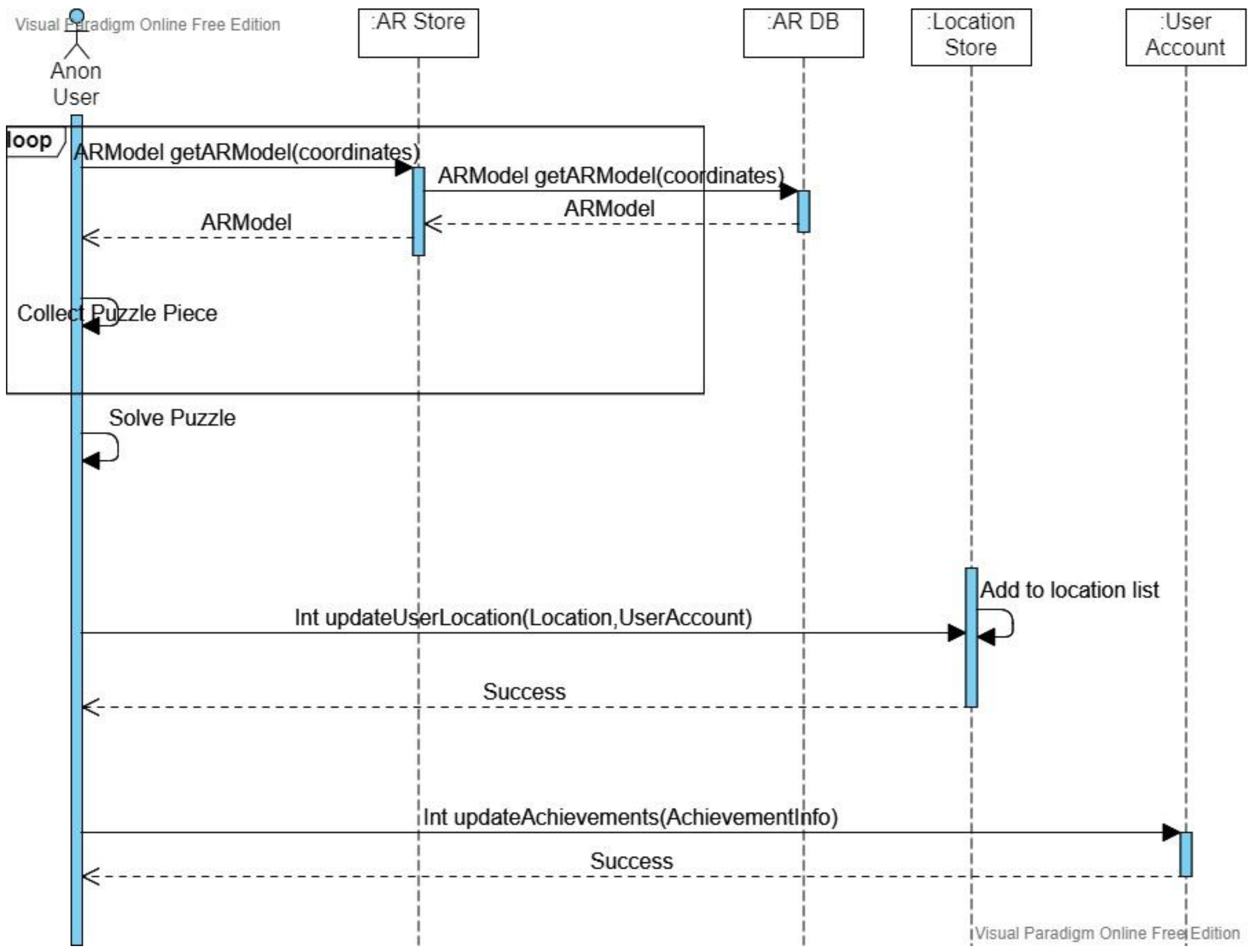
After reaching the location:


Anon User
:locationStore
:ARStore
:ARmodelsDB
:locationsDB
:UserAccount
:UsersDB
List getLocationList(UserAccount)
List locationList
alt
[Current location matches one location on the location list]
ARModel getARModel(coordinates)
ARModel getARModel(coordinates)
ARModel getARModel(coordinates)
ARModel
ARModel
ARModel
locationInfo getLocationInfo(coordinates)
locationInfo getLocationInfo(coordinates)
locationInfo
locationInfo
alt
[user is authenticated]
Int updateAchievements(AchievementInfo)
Int updateAchievements (AchievementInfo)
Int 1/0(success/failure)
Int 0/1(success/failure)
Int updateCollection(Collection)
Int updateCollection(Collection)
Int 1/0(success/failure)
Int 1/0(success/failure)
[user is anonymous]
(achievements and collections are stored locally)

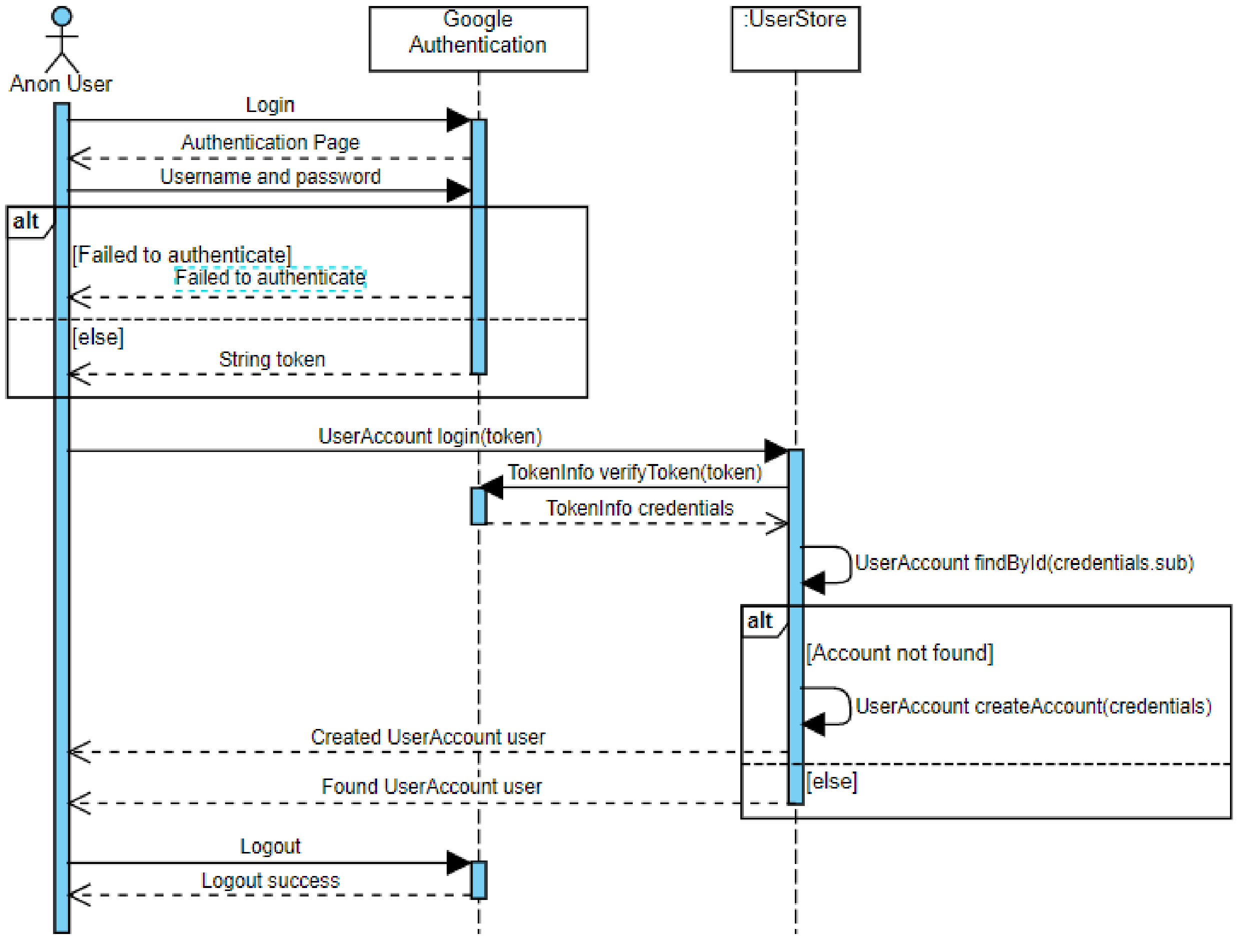
- **AR buttons**


Anon User
:AR Store
:AR DB
:Location Store
:User Account
ARModel getARModel(coordinates)
ARModel getARModel(coordinates)
ARModel
ARModel
Solve Puzzle
Int updateUserLocation(Location,UserAccount)
Success
Update location list
Int updateAchievements(AchievementInfo)
Success

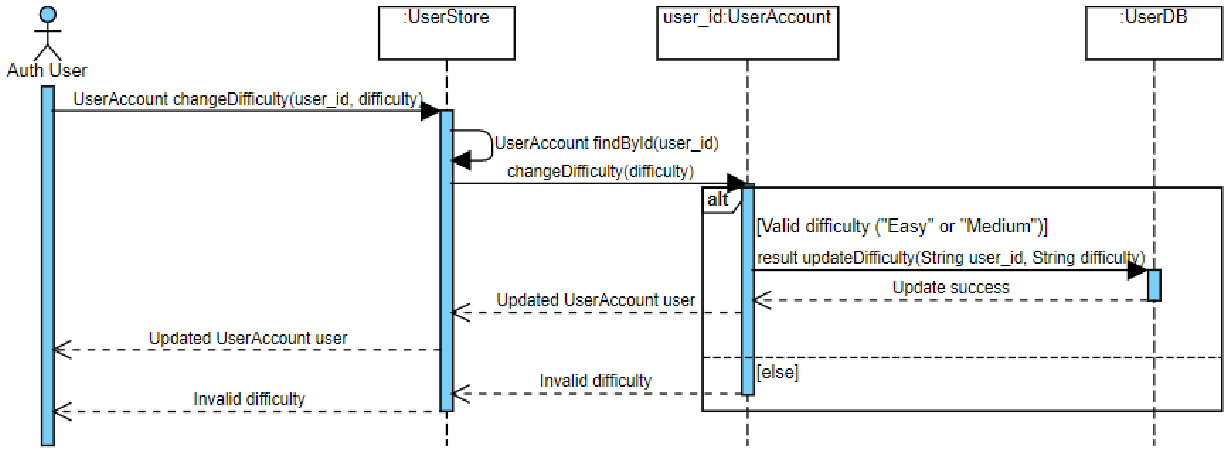
-   **AR puzzles**


Visual Paradigm Online Free Edition
Anon User
:AR Store
:AR DB
:Location Store
:User Account
loop
ARModel getARModel(coordinates)
ARModel getARModel(coordinates)
ARModel
ARModel
Collect Puzzle Piece
Solve Puzzle
Add to location list
Int updateUserLocation(Location,UserAccount)
Success
Int updateAchievements(AchievementInfo)
Success
Visual Paradigm Online Free Edition

- Login/Logout

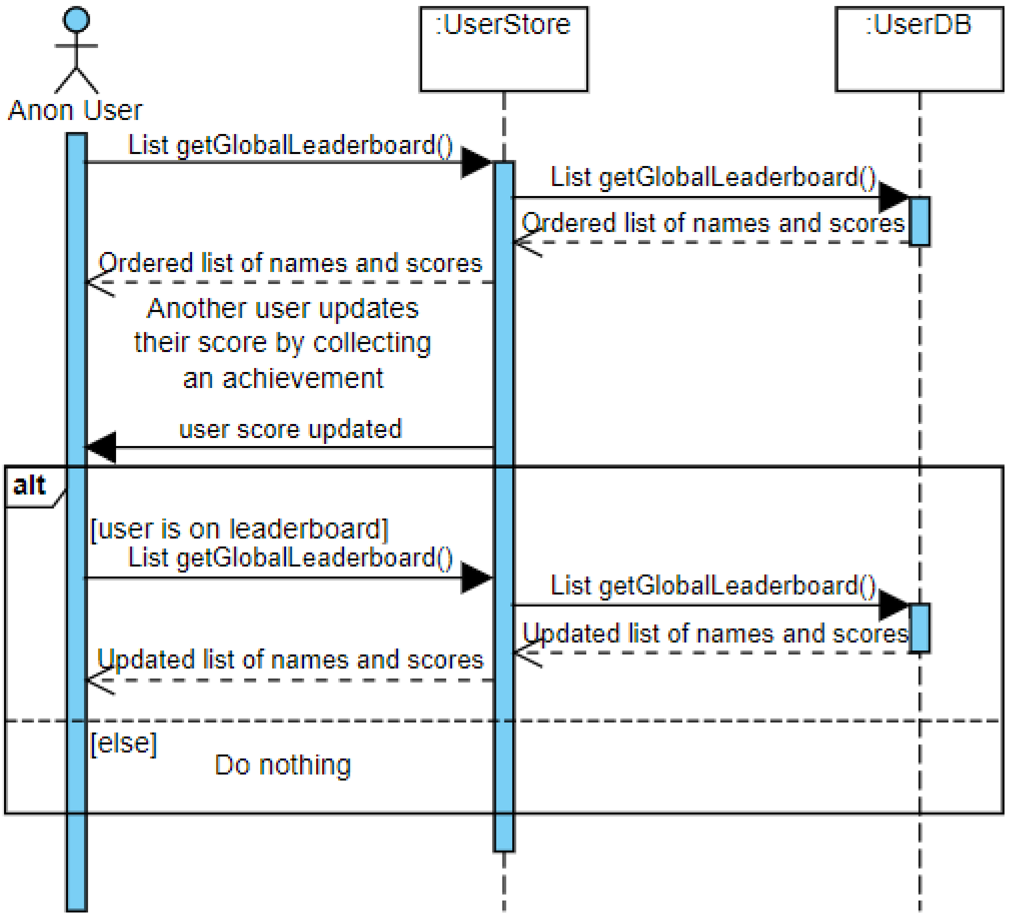


- Change difficulty

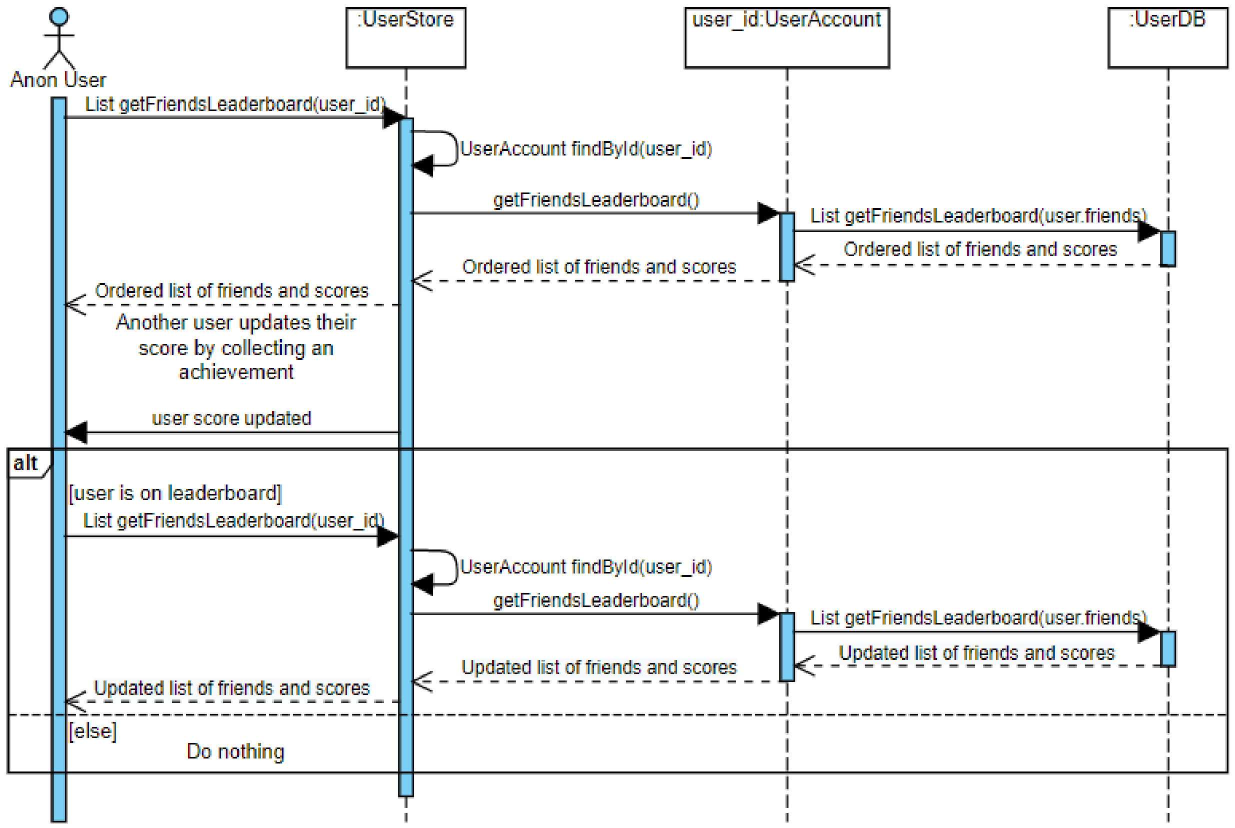

Auth User
:UserStore
user_id:UserAccount
:UserDB
UserAccount changeDifficulty(user_id, difficulty)
UserAccount findById(user_id)
changeDifficulty(difficulty)
alt
[Valid difficulty ("Easy" or "Medium")]
result updateDifficulty(String user_id, String difficulty)
Update success
Updated UserAccount user
Updated UserAccount user
[else]
Invalid difficulty
Invalid difficulty

- **See leaderboard**

Viewing the global leaderboard as Anon user:

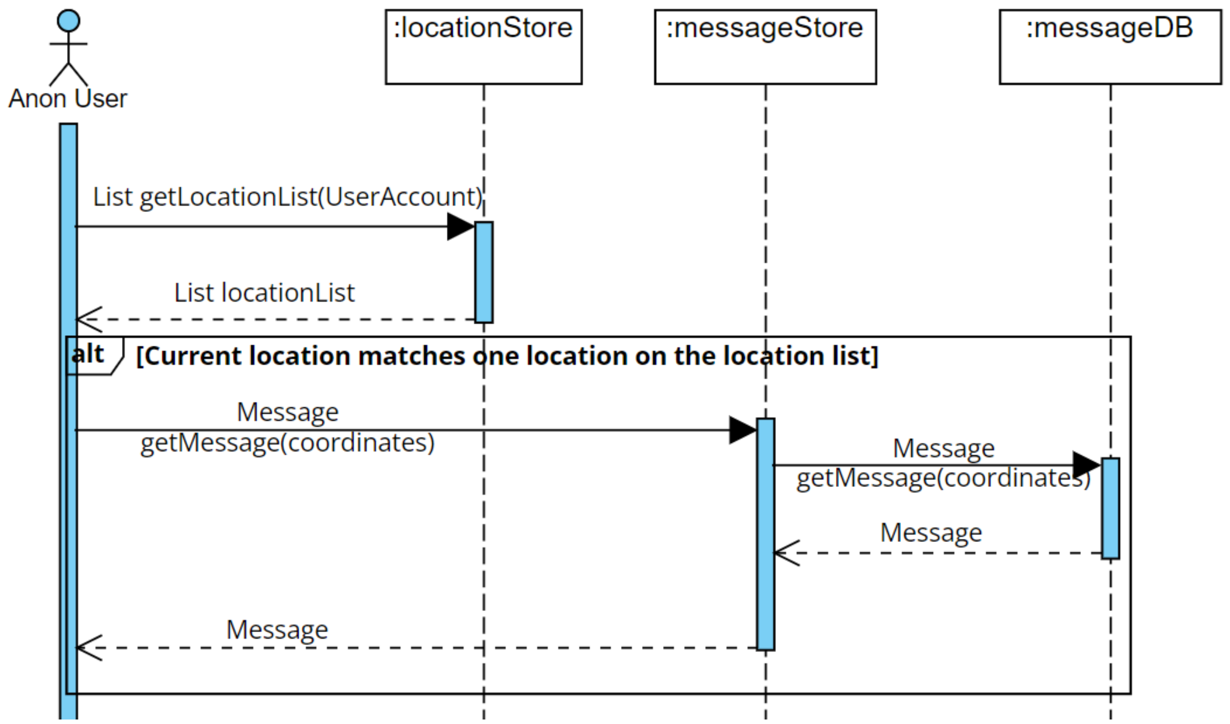

:UserStore
:UserDB
Anon User
List getGlobalLeaderboard()
List getGlobalLeaderboard()
Ordered list of names and scores
Ordered list of names and scores
Another user updates their score by collecting an achievement
user score updated
alt
[user is on leaderboard]
List getGlobalLeaderboard()
List getGlobalLeaderboard()
Updated list of names and scores
Updated list of names and scores
[else]
Do nothing

Viewing the friend leaderboard as Auth user:



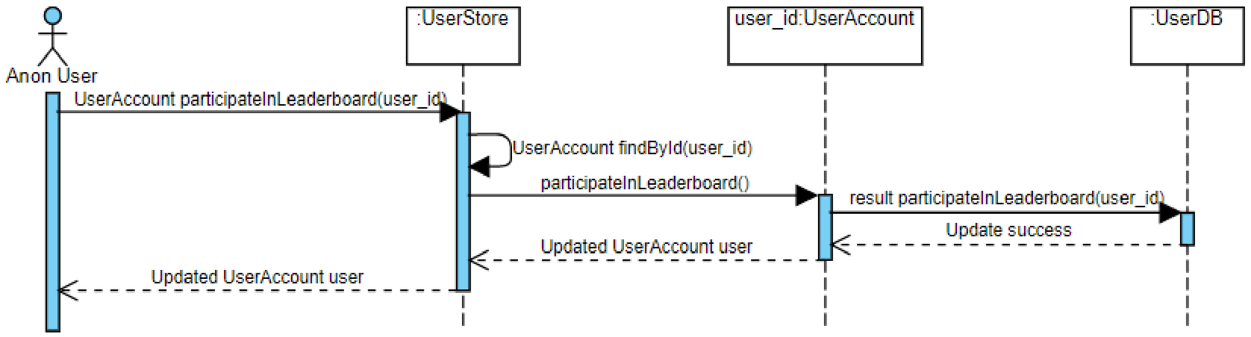
Anon User
:UserStore
user_id:UserAccount
:UserDB
List getFriendsLeaderboard(user_id)
UserAccount findById(user_id)
getFriendsLeaderboard()
List getFriendsLeaderboard(user.friends)
Ordered list of friends and scores
Ordered list of friends and scores
Ordered list of friends and scores
Another user updates their score by collecting an achievement
user score updated
alt
[user is on leaderboard]
List getFriendsLeaderboard(user_id)
UserAccount findById(user_id)
getFriendsLeaderboard()
List getFriendsLeaderboard(user.friends)
Updated list of friends and scores
Updated list of friends and scores
Updated list of friends and scores
[else]
Do nothing

- **View message**



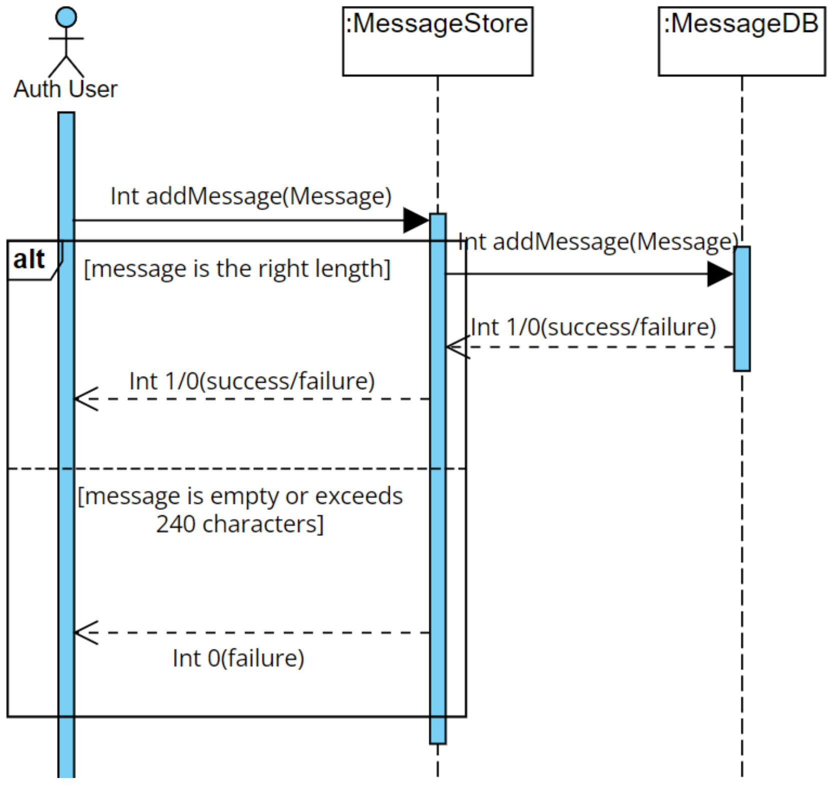
Anon User
:locationStore
:messageStore
:messageDB
List getLocationList(UserAccount)
List locationList
alt
[Current location matches one location on the location list]
Message getMessage(coordinates)
Message getMessage(coordinates)
Message
Message

- **Participate in leaderboards**

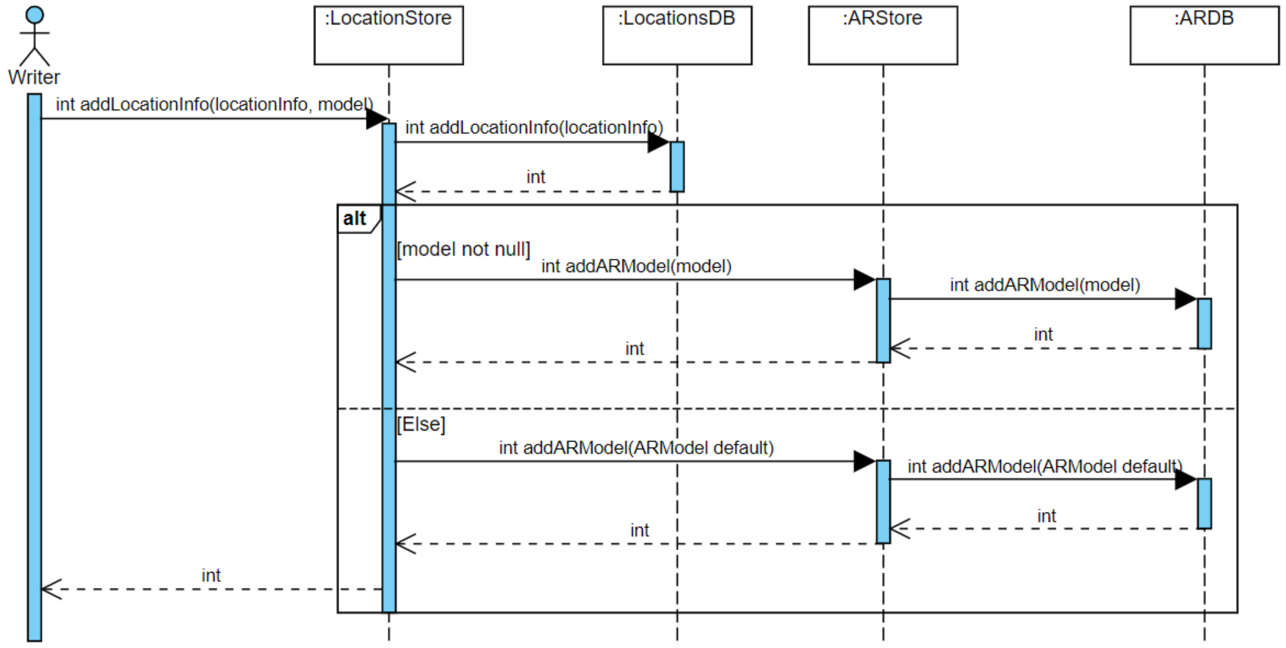


Anon User
:UserStore
user_id:UserAccount
:UserDB
UserAccount participateInLeaderboard(user_id)
UserAccount findById(user_id)
participateInLeaderboard()
result participateInLeaderboard(user_id)
Update success
Updated UserAccount user
Updated UserAccount user

- **Add a simple message**

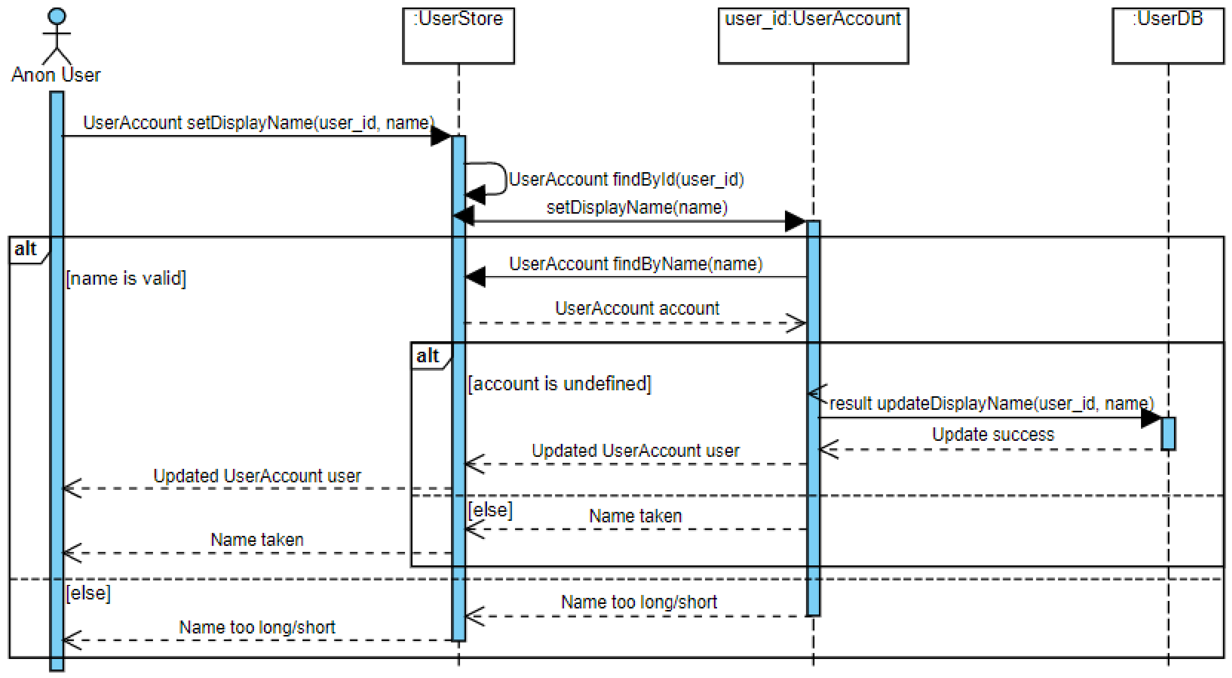


- **Add a location**

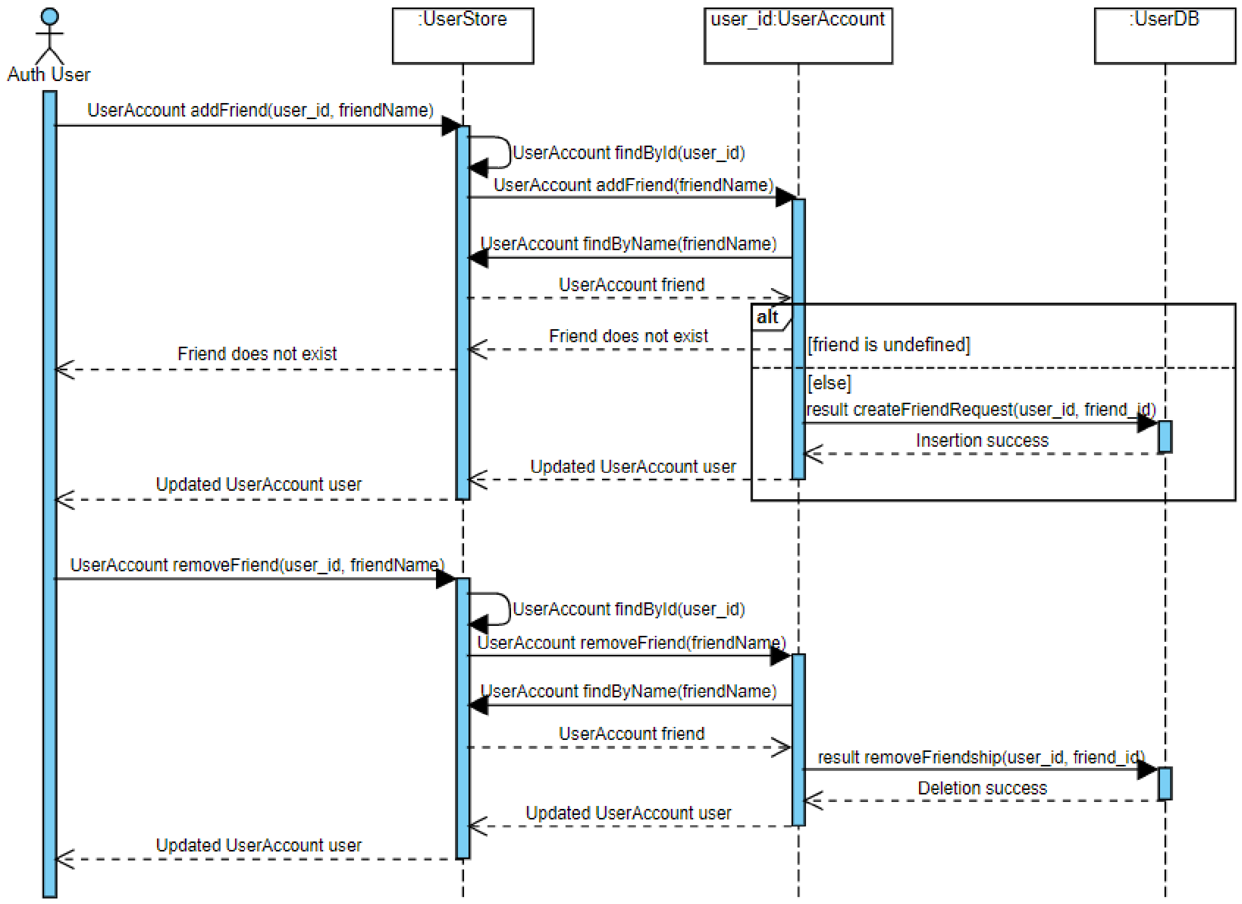

Writer
:LocationStore
:LocationsDB
:ARStore
:ARDB
int addLocationInfo(locationInfo, model)
int addLocationInfo(locationInfo)
int
alt
[model not null]
int addARModel(model)
int addARModel(model)
int
int
[Else]
int addARModel(ARModel default)
int addARModel(ARModel default)
int
int
int

- **Manage profile**

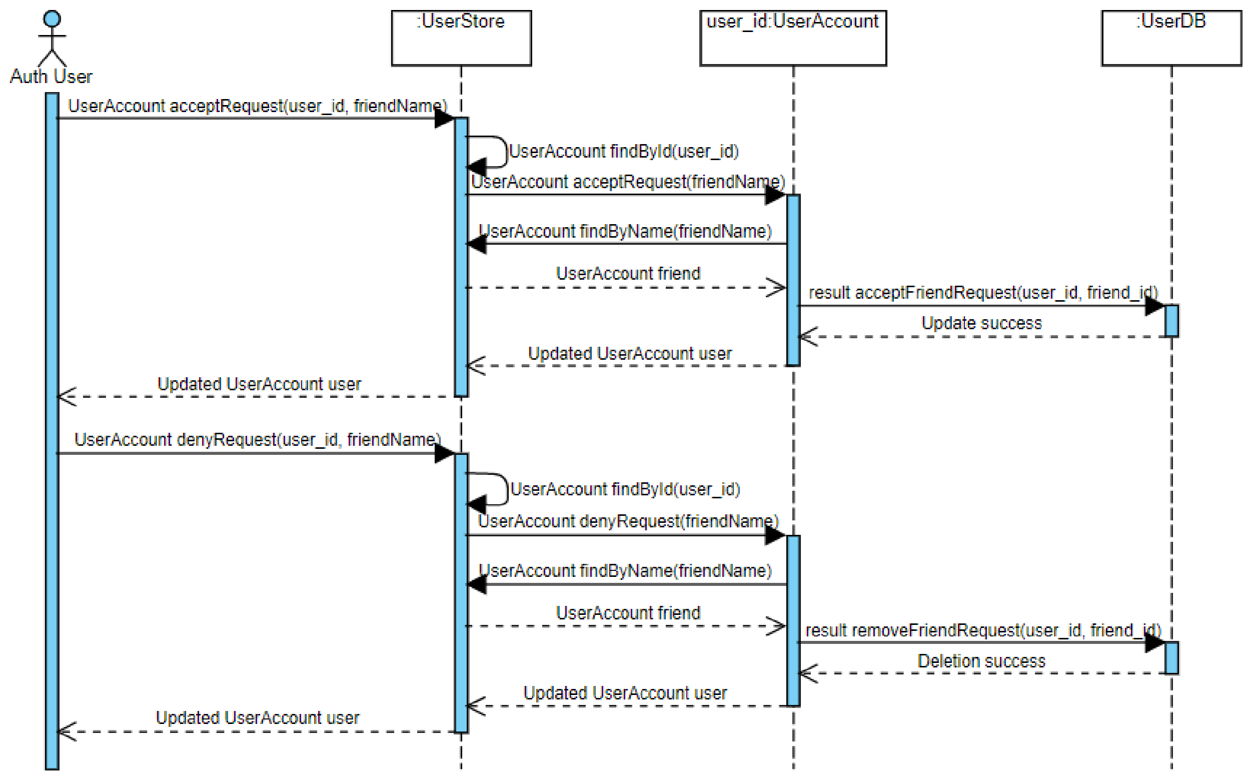
Changing name:



Sending requests and removing friends:


:UserStore
user_id:UserAccount
:UserDB
Auth User
UserAccount addFriend(user_id, friendName)
UserAccount findById(user_id)
UserAccount addFriend(friendName)
UserAccount findByName(friendName)
UserAccount friend
alt
Friend does not exist
[friend is undefined]
Friend does not exist
[else]
result createFriendRequest(user_id, friend_id)
Insertion success
Updated UserAccount user
Updated UserAccount user
UserAccount removeFriend(user_id, friendName)
UserAccount findById(user_id)
UserAccount removeFriend(friendName)
UserAccount findByName(friendName)
UserAccount friend
result removeFriendship(user_id, friend_id)
Deletion success
Updated UserAccount user
Updated UserAccount user

Managing incoming requests:



Auth User
:UserStore
user_id:UserAccount
:UserDB
UserAccount acceptRequest(user_id, friendName)
UserAccount findById(user_id)
UserAccount acceptRequest(friendName)
UserAccount findByName(friendName)
UserAccount friend
result acceptFriendRequest(user_id, friend_id)
Update success
Updated UserAccount user
Updated UserAccount user
UserAccount denyRequest(user_id, friendName)
UserAccount findById(user_id)
UserAccount denyRequest(friendName)
UserAccount findByName(friendName)
UserAccount friend
result removeFriendRequest(user_id, friend_id)
Deletion success
Updated UserAccount user
Updated UserAccount user

Realizing Non-Functional Requirements:

[Usability] The user should not need more than 5 clicks to perform any action.

- We will implement this requirement by making sure that features should only go to a minimum depth of 5.
- If it exceeds that we will
    - Reconsider the feature
    - Add a shortcut to the feature
- Each screen or button press will be automated into a list, our test will check if the list cascading is less than 5

[Performance] AR/relevant information should show up within 1 second after the user points their phone in a certain direction at a certain location.

- We plan to implement this feature making sure that all AR relevant features are pre-downloaded to the app for fast access
- We plan to test this requirement by simulating the camera using predefined pictures and locations and then measuring the time it takes for the AR/information to show up.

[Energy efficiency] The battery life should not drop by more than 1% after continuous usage of the app for 5 minutes.

- We plan to make sure that camera/internet only activates if our application detects that the phone is in a predestined location
- We plan to test the app before a feature is added to make sure it goes under our requirements
- We plan to test the drainage using Battery Historian for virtual deployment and battery manager apps like AccuBattery for live deployment. We will test the app under heavy use (Constantly display AR image).

[User-friendly Interface] The UI should be simple and easy to use. The icons of the buttons should be able to suggest what they are used for.

- All interactable elements should be at least 48 x 48 dp
- Neutral and Friendly Color Theming
- We will standardize the font usage to 2 font families
- Make sure that the minimum font size for our app is 12pt

- We plan to find several students who have never used the app before and introduce the general functionality of the app. After that, we will ask them to match the usages to each button.

Design For Beyond the Minimal Scope Functionality:

- AR animation:

  We plan to combine Live2d Cubism SDK for Unity and Google ARCore to create interactive AR puzzles for users. Live2d Cubism is an animation software that can create 2D models based on layers of images. Live2d animations occupy fewer resources compared to 3D models while being smoother than traditional frame-by-frame animations.

  Our team will design and build original live2d models. By changing model parameters, we can present various animations to users. We can also let users interact with the models by touching or dragging them. This will encourage users to challenge puzzles to collect these creatures as rewards, and so explore more places.

Frameworks:

Google ARCore

- In built functionality with Android devices
- Cheap
- Well documented with community support

Live2d Cubism

- Smoother and easier to make compared to frame-by-frame animation
- Occupies fewer resources than using 3D models
- Easy to control animations with automatically generated .json files

Unity

- Allows in-built integration between Cubism and ARCore
- Easy to use for modeling and implementation

MySQL vs MongoDb

- Our dataset is interlinked between user accounts, location information, AR model information, and Message information
- Our dataset is horizontally fixed, potential addition of columns are rare
- Accessing multi-table information is easier with MySQL

Node.js vs Python

- Required for Project
- Node.js is faster
- Has inbuilt functions for Multi-Threading (useful for servers)
- Scalable

Link to Git Repository:

https://github.com/jane-cz/CPEN321-final-project

Public IP of Server and APIs:

20.228.168.55

Users module:

- POST http://20.228.168.55/users/login
    - Request content: {“token”: ”value”}
        - Value is the token received after Google sign in
- PUT http://20.228.168.55/users/user_id/difficulty
    - user_id is a parameter
        - The id of the account to change the difficulty of
    - Request content: {“difficulty”: “value”}
        - Value is the difficulty the user wishes to change too
- PUT http://20.228.168.55/users/user_id/displayName
    - user_id is a parameter
        - The id of the account to change the name of
    - Request content: {“displayName”: “value”}
        - Value is the name that the user wishes to change their name too
- POST http://20.228.168.55/users/user_id/friends
    - user_id is a parameter
        - The id of the account that is sending a request
    - Request content: {“displayName”: “value”}
        - Value is the name the user wishes to send a friend request too
- DELETE http://20.228.168.55/users/user_id/friends/displayName
    - user_id is a parameter
        - The id of the account that is removing a friend
    - displayName is a parameter

        - Name of the friend the user wishes to delete off their friends list
- PUT http://20.228.168.55/users/user_id/requests
    - user_id is a parameter
        - The id of the account that is accepting a request
    - Request content: {“displayName”: “value”}
        - Value is the name the user wishes to accept a friend request from
- DELETE http://20.228.168.55/users/user_id/requests/displayName
    - user_id is a parameter
        - The id of the account that is deny a request
    - displayName is a parameter
        - Name of the friend the user wishes to deny a request from
- PUT http://20.228.168.55/users/user_id/participateInLeaderboard
    - user_id is a parameter
        - The id of the account that is going to participate in leaderboards
- POST http://20.228.168.55/users/user_id/locations
    - user_id is a parameter
        - The id of the account that is unlocking a location
    - Request content: {“location_name”: “value”}
        - Value is the name of the location the user has unlocked
- POST http://20.228.168.55/users/user_id/items
    - user_id is a parameter
        - The id of the account that its unlocking an item
    - Request content: {“id”: “value”}
        - Value is the id of the item the user has unlocked
- PUT http://20.228.168.55/users/user_id/achievements
    - user_id is a parameter
        - The id of the account that is updating achievements

        - Request content: {“id”: “value1”, “Type”: “value”, “points”: value, “image”: “value”}
            - The fields of an achievement (see AchievementInfo)
- GET http://20.228.168.55/users/user_id/leaderboard
    - user_id is a parameter
        - The id of the account to get the friends leaderboard of
- GET http://20.228.168.55/users/leaderboard

Locations module:

- GET http://20.228.168.55/locations/
- POST http://20.228.168.55/locations/
    - Request content: { "location_name": "value", "coordinate_latitude":value, "coordinate_longitude":value, "fun_facts":"value", "related_links":"value", "about":"value", "image_url":"value" }
- GET http://20.228.168.55/locations/location_name
    - location_name is a parameter
        - The name of the location to be retrieved
- PUT http://20.228.168.55/locations/location_name
    - location_name is a parameter
        - The name of the location to be updated
    - Request content: { "location_name": "value", "coordinate_latitude":value, "coordinate_longitude":value, "fun_facts":"value", "related_links":"value", "about":"value", "image_url":"value" }
- DELETE http://20.228.168.55/locations/location_name
    - location_name is a parameter
        - The name of the location to be deleted
- GET http://20.228.168.55/locations/user/user_account_id
    - user_account_id is a parameter
        - The id of the account to get unlocked locations of

World module:

- GET http://20.228.168.55/messages/
- POST http://20.228.168.55/messages/
  - Request content: { "coordinate_latitude":value, "coordinate_longitude":value, "message_text":"value", "user_account_id":value }
- GET http://20.228.168.55/messages/id
  - id is a parameter
    - The id of the message to be retrieved
- PUT http://20.228.168.55/messages/id
  - id is a parameter
    - The id of the message to be updated
  - Request content: { "coordinate_latitude":value, "coordinate_longitude":value, "message_text":"value", "user_account_id":value }
- DELETE http://20.228.168.55/messages/id
  - id is a parameter
    - The id of the message to be deleted

Plan for Implementing Complexity Idea:

The next step of our project is to integrate AR modules and puzzles into our application. This includes creating AR modules using unity and Live2dCubism. Also integrating those modules with google AR core. The management of the AR will be done on the backend while the camera integration with the models will be done on the frontend.

Changes in Project Scope:

Participate in LeaderBoard Use Case

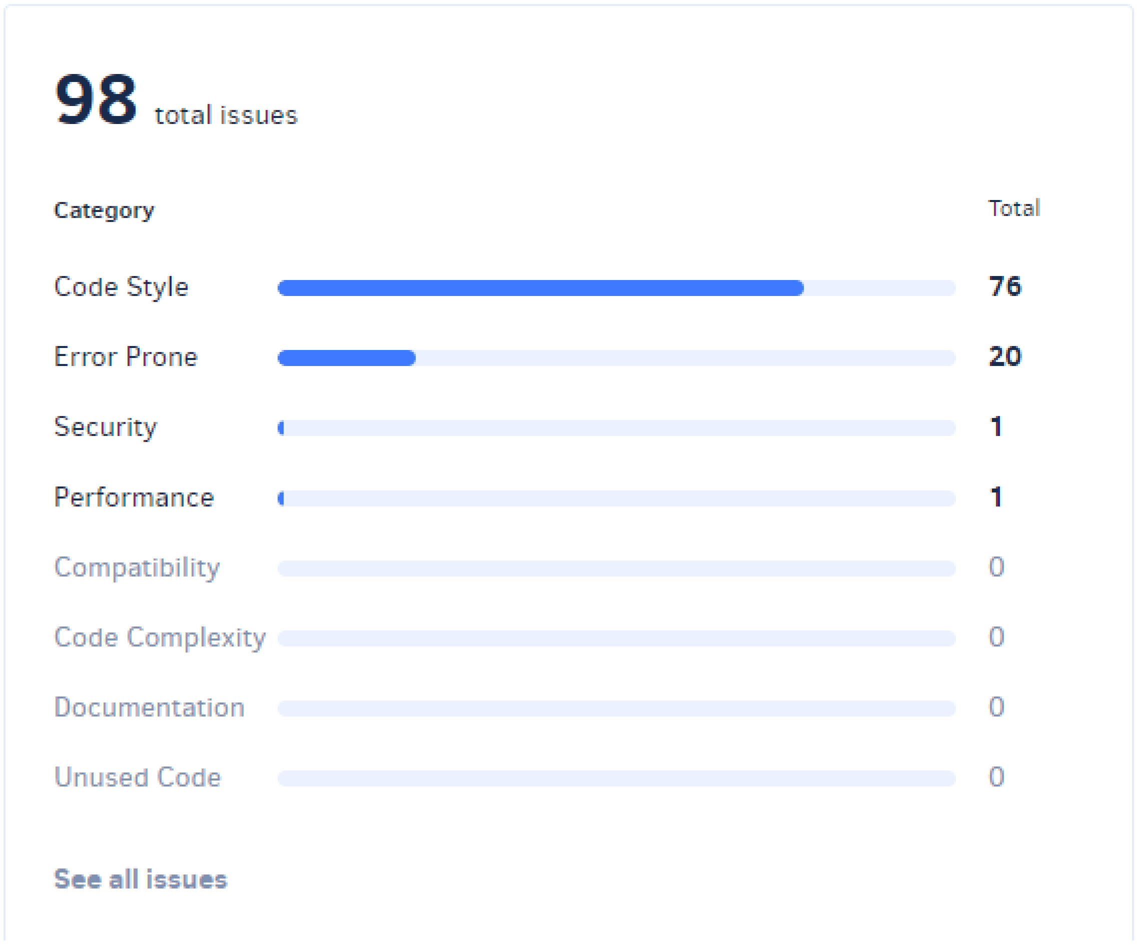
- Added Clarification that this use case also affects Friend Leaderboard.

Codacy Reports

Repository Dashboard Page: https://app.codacy.com/gh/CPEN321-ubcexplore/CPEN321-final-project/dashboard?branch=main

Commit SHA: 0fe8a312433264df4b6566b6298a141c71476d8e

Issues by Category



98 total issues

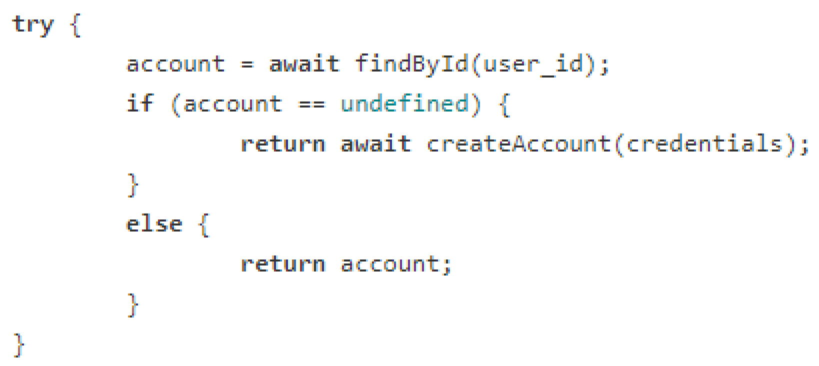
Category	Total
Code Style	76
Error Prone	20
Security	1
Performance	1
Compatibility	0
Code Complexity	0
Documentation	0
Unused Code	0

Issues by Pattern

- Unused Imports - 25

- import android.widget.TextView
- Field Declarations Should Be At Start Of Class - 7
  - private Socket mSocket
- Unused Local Variable - 5
  - MarkerOptions markerOptions = new MarkerOptions()
- Unused Private Field - 5
  - private int ID
- Unnecessary Constructor - 5
  - public CameraFragment()
- Use Equals To Compare Strings - 5
  - if (user_id == null || user_id == "")
- No unused vars - 5
  - const user_id = req.params.user_id
- Compare Objects With Equals - 5
  - if (user_id == null || user_id == "")
- No extra semi - 4
  - catch (err) { throw err; };
- One Declaration Per Line - 3
  - int lat = 0, lng = 0;
- Empty If Stmt - 3
  - PackageManager.PERMISSION_GRANTED){
- Inaccurate Numeric Literal - 3
  - "duration": 3100000000
- Non Static Initializer - 2
  - private Socket mSocket;
  - {
  - try {
- Security: Detect non literal fs filename - 1
  - (function(t){function e(e){for(var s,n,l=e[0],o=e[1],c=e[2],u=0,f=[];u<l.length;u++)n=l[u],Object.prototype.hasOwnProperty.call(i,n)&&i[n]&&f.push(i[n][0]),i[n]=0;for(s in

    - o)Object.prototype.hasOwnProperty.call(o,s)&&(t[s]=o[s]);d&&d(e);while(f.length)f.shift()();return r.push.apply(r,c||[]),a()}function a(){for(var t,e=0;e<r.length;e++){for(var a=r[e],s=!0,l=1;l<a.length;l++){var o=a[l];0!==i[o]&&(s=!1)}s&&(r.splice(e--,1),t=n(n.s=a[0]))}return t}var s={},i={app:0},r=[];function n(e){if(s[e])return s[e].ex
- Singular Field - 1
    - private ActivityLocationMapBinding binding;
- Class Naming Conventions - 1
    - public class recyclerAdapter extends RecyclerView.Adapter<recyclerAdapter.MyViewHolder> {
- ESLint8_n_no-path-concat - 1
    - res.sendFile(__dirname + "/index.html");
- No extra parens - 1
    - (function(t){function e(e){for(var s,n,l=e[0],o=e[1],c=e[2],u=0,f=[];u<l.length;u++)n=l[u],Object.prototype.hasOwnProperty.call(i,n)&&i[n]&&f.push(i[n][0]),i[n]=0;for(s in o)Object.prototype.hasOwnProperty.call(o,s)&&(t[s]=o[s]);d&&d(e);while(f.length)f.shift()();return r.push.apply(r,c||[]),a()}function a(){for(var t,e=0;e<r.length;e++){for(var a=r[e],s=!0,l=1;l<a.length;l++){var o=a[l];0!==i[o]&&(s=!1)}s&&(r.splice(e--,1),t=n(n.s=a[0]))}return t}var s={},i={app:0},r=[];function n(e){if(s[e])return s[e].ex
- Uncommented Empty Method Body - 1
    - public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
- Node: No extraneous require - 1
    - const { createProxyMiddleware } = require("http-proxy-middleware");
- No useless catch - 1

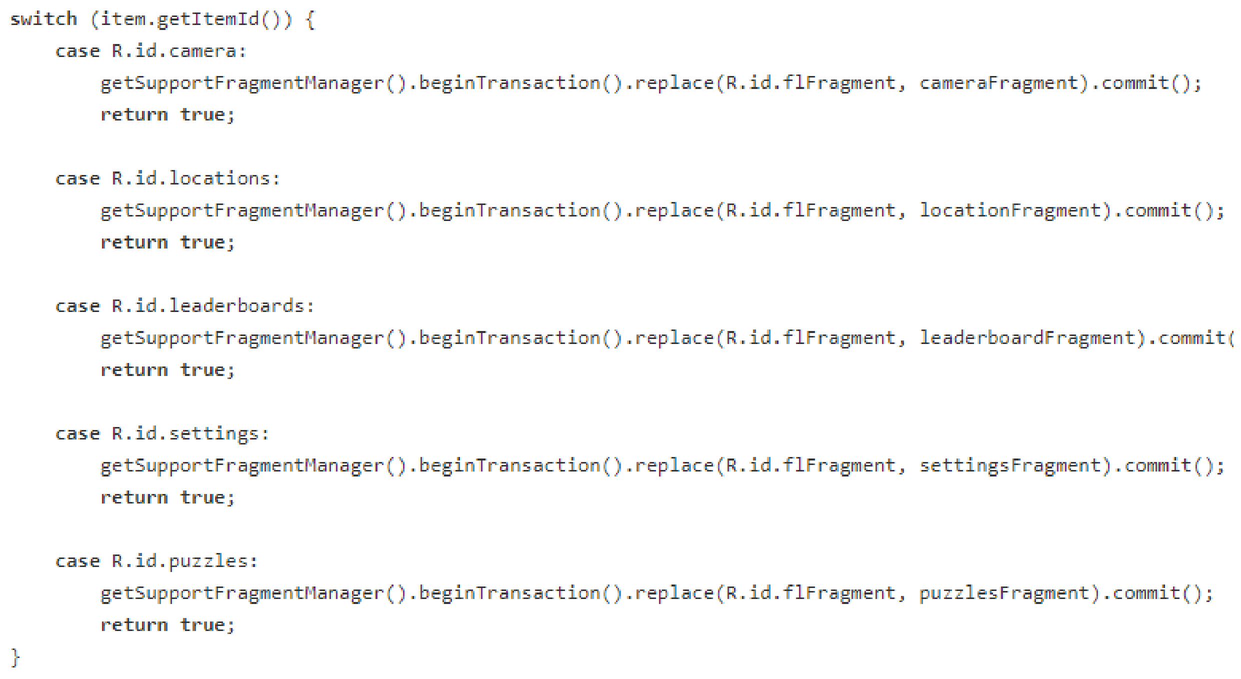
- 

```
try {
        account = await findById(user_id);
        if (account == undefined) {
                return await createAccount(credentials);
        }
        else {
                return account;
        }
}
```

- String To String - 1
    - String response;
    - response.toString().equals("[]");
- New cap - 1
    - (function(t){function e(e){for(var s,n,l=e[0],o=e[1],c=e[2],u=0,f=[];u<l.length;u++)n=l[u],Object.prototype.hasOwnProperty.call(i,n)&&i[n]&&f.push(i[n][0]),i[n]=0;for(s in o)Object.prototype.hasOwnProperty.call(o,s)&&(t[s]=o[s]);d&&d(e);while(f.length)f.shift()();return r.push.apply(r,c||[]),a()}function a(){for(var t,e=0;e<r.length;e++){for(var a=r[e],s=!0,l=1;l<a.length;l++){var o=a[l];0!==i[o]&&(s=!1)}s&&(r.splice(e--,1),t=n(n.s=a[0]))}return t}var s={},i={app:0},r=[];function n(e){if(s[e])return s[e].ex
- No sequences - 1
    - (function(t){function e(e){for(var s,n,l=e[0],o=e[1],c=e[2],u=0,f=[];u<l.length;u++)n=l[u],Object.prototype.hasOwnProperty.call(i,n)&&i[n]&&f.push(i[n][0]),i[n]=0;for(s in o)Object.prototype.hasOwnProperty.call(o,s)&&(t[s]=o[s]);d&&d(e);while(f.length)f.shift()();return r.push.apply(r,c||[]),a()}function a(){for(var t,e=0;e<r.length;e++){for(var a=r[e],s=!0,l=1;l<a.length;l++){var o=a[l];0!==i[o]&&(s=!1)}s&&(r.splice(e--,1),t=n(n.s=a[0]))}return t}var s={},i={app:0},r=[];function n(e){if(s[e])return s[e].ex
- No mixed operators - 1
    - (function(t){function e(e){for(var s,n,l=e[0],o=e[1],c=e[2],u=0,f=[];u<l.length;u++)n=l[u],Object.prototype.has

OwnProperty.call(i,n)&&i[n]&&f.push(i[n][0]),i[n]=0;for(s in o)Object.prototype.hasOwnProperty.call(o,s)&&(t[s]=o[s]);d&&d(e);while(f.length)f.shift()();return r.push.apply(r,c||[]),a()}function a(){for(var t,e=0;e<r.length;e++){for(var a=r[e],s=!0,l=1;l<a.length;l++){var o=a[l];0!==i[o]&&(s=!1)}s&&(r.splice(e--,1),t=n(n.s=a[0]))}return t}var s={},i={app:0},r=[];function n(e){if(s[e])return s[e].ex

- Switch Stmts Should Have Default - 1



```
switch (item.getItemId()) {
    case R.id.camera:
        getSupportFragmentManager().beginTransaction().replace(R.id.flFragment, cameraFragment).commit();
        return true;

    case R.id.locations:
        getSupportFragmentManager().beginTransaction().replace(R.id.flFragment, locationFragment).commit();
        return true;

    case R.id.leaderboards:
        getSupportFragmentManager().beginTransaction().replace(R.id.flFragment, leaderboardFragment).commit(
        return true;

    case R.id.settings:
        getSupportFragmentManager().beginTransaction().replace(R.id.flFragment, settingsFragment).commit();
        return true;

    case R.id.puzzles:
        getSupportFragmentManager().beginTransaction().replace(R.id.flFragment, puzzlesFragment).commit();
        return true;
}
```

- One var - 1
    - (function(t){function e(e){for(var s,n,l=e[0],o=e[1],c=e[2],u=0,f=[];u<l.length;u++)n=l[u],Object.prototype.hasOwnProperty.call(i,n)&&i[n]&&f.push(i[n][0]),i[n]=0;for(s in o)Object.prototype.hasOwnProperty.call(o,s)&&(t[s]=o[s]);d&&d(e);while(f.length)f.shift()();return r.push.apply(r,c||[]),a()}function a(){for(var t,e=0;e<r.length;e++){for(var a=r[e],s=!0,l=1;l<a.length;l++){var o=a[l];0!==i[o]&&(s=!1)}s&&(r.splice(e--,1),t=n(n.s=a[0]))}return t}var s={},i={app:0},r=[];function n(e){if(s[e])return s[e].ex
- Yoda - 1
    - (function(t){function e(e){for(var s,n,l=e[0],o=e[1],c=e[2],u=0,f=[];u<l.length;u++)n=l[u],Object.prototype.has

OwnProperty.call(i,n)&&i[n]&&f.push(i[n][0]),i[n]=0;for(s in o)Object.prototype.hasOwnProperty.call(o,s)&&(t[s]=o[s]);d&&d(e);while(f.length)f.shift()();return r.push.apply(r,c||[]),a()}function a(){for(var t,e=0;e<r.length;e++){for(var a=r[e],s=!0,l=1;l<a.length;l++){var o=a[l];0!==i[o]&&(s=!1)}s&&(r.splice(e--,1),t=n(n.s=a[0]))}return t}var s={},i={app:0},r=[];function n(e){if(s[e])return s[e].ex

- No return assign - 1
    - (function(t){function e(e){for(var s,n,l=e[0],o=e[1],c=e[2],u=0,f=[];u<l.length;u++)n=l[u],Object.prototype.hasOwnProperty.call(i,n)&&i[n]&&f.push(i[n][0]),i[n]=0;for(s in o)Object.prototype.hasOwnProperty.call(o,s)&&(t[s]=o[s]);d&&d(e);while(f.length)f.shift()();return r.push.apply(r,c||[]),a()}function a(){for(var t,e=0;e<r.length;e++){for(var a=r[e],s=!0,l=1;l<a.length;l++){var o=a[l];0!==i[o]&&(s=!1)}s&&(r.splice(e--,1),t=n(n.s=a[0]))}return t}var s={},i={app:0},r=[];function n(e){if(s[e])return s[e].ex
- Unnecessary Fully Qualified Name - 1
    - public void onLocationChanged(android.location.Location newLoc) {
- Assignment In Operand - 1
    - (function(t){function e(e){for(var s,n,l=e[0],o=e[1],c=e[2],u=0,f=[];u<l.length;u++)n=l[u],Object.prototype.hasOwnProperty.call(i,n)&&i[n]&&f.push(i[n][0]),i[n]=0;for(s in o)Object.prototype.hasOwnProperty.call(o,s)&&(t[s]=o[s]);d&&d(e);while(f.length)f.shift()();return r.push.apply(r,c||[]),a()}function a(){for(var t,e=0;e<r.length;e++){for(var a=r[e],s=!0,l=1;l<a.length;l++){var o=a[l];0!==i[o]&&(s=!1)}s&&(r.splice(e--,1),t=n(n.s=a[0]))}return t}var s={},i={app:0},r=[];function n(e){if(s[e])return s[e].ex
- No redeclare - 1
    - var message_text = req.query.message_text;
- No void - 1
    - (function(t){function e(e){for(var s,n,l=e[0],o=e[1],c=e[2],u=0,f=[];u<l.length;u++)n=l[u],Object.prototype.has

OwnProperty.call(i,n)&&i[n]&&f.push(i[n][0]),i[n]=0;for(s in o)Object.prototype.hasOwnProperty.call(o,s)&&(t[s]=o[s]);d&&d(e);while(f.length)f.shift()();return r.push.apply(r,c||[]),a()}function a(){for(var t,e=0;e<r.length;e++){for(var a=r[e],s=!0,l=1;l<a.length;l++){var o=a[l];0!==i[o]&&(s=!1)}s&&(r.splice(e--,1),t=n(n.s=a[0]))}return t}var s={},i={app:0},r=[];function n(e){if(s[e])return s[e].ex

- No unused expressions - 1
    - (function(t){function e(e){for(var s,n,l=e[0],o=e[1],c=e[2],u=0,f=[];u<l.length;u++)n=l[u],Object.prototype.hasOwnProperty.call(i,n)&&i[n]&&f.push(i[n][0]),i[n]=0;for(s in o)Object.prototype.hasOwnProperty.call(o,s)&&(t[s]=o[s]);d&&d(e);while(f.length)f.shift()();return r.push.apply(r,c||[]),a()}function a(){for(var t,e=0;e<r.length;e++){for(var a=r[e],s=!0,l=1;l<a.length;l++){var o=a[l];0!==i[o]&&(s=!1)}s&&(r.splice(e--,1),t=n(n.s=a[0]))}return t}var s={},i={app:0},r=[];function n(e){if(s[e])return s[e].ex

Commit SHA (after fixing issues): 13973c4bb894a21e5b7093b8b6b29ce6149004a2
0 issues left for all categories and all patterns.

Two Major Issues Found in Peer Group Code (Gruwup):
Backend - Duplicate Adventures:
When creating adventures it seems that multiple adventures of the same values (owner, title, location, etc.) can be created. While the creation returns a unique adventureId it seems unintuitive for events with the same information to exist.

Frontend - Should provide more feedback to the user:
It would be better if the user could be provided with more feedback when using the app. For instance, if the user fails to login, instead of getting a blank welcome page, the app can show a toast message (or other forms of message) informing the user of what went wrong, like internet connection issues or wrong credentials, so the user would know

what to do. The same is true for other parts of the app, like when the user fails to logout or the profile page fails to load, they should be provided with relevant details.

Two Major Issues Peer Group Found in Our Code:

Frontend / Avoid Repetition:

Server addresses and api key values are repeated heavily in files for frontend code. It is recommended to make a common file for it / include the strings in strings.xml to avoid repetition.

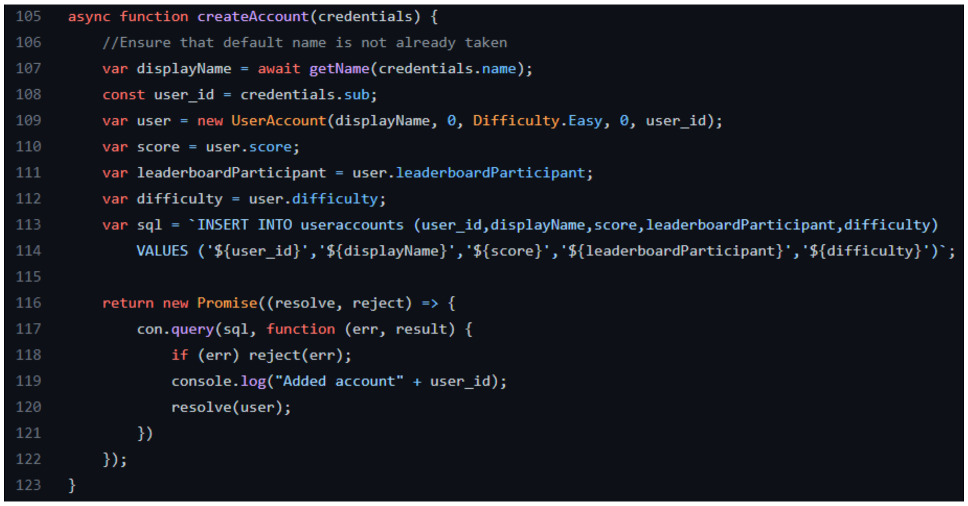
Backend file structure/Readability:

The controller, service, and database code are all in one single file for each major module, with no clear boundaries in between the code to indicate which ones belong to which layer.This makes it very hard to read and understand the structure. Also all sql queries are hard coded, making it very hard to debug. A better way would be having a folder/file to document all the string for fields of each data model, and importing them as variables to insert into sql query strings, this ensures consistency in the code and makes database calls easier to debug.

Fixing Identified Issues:

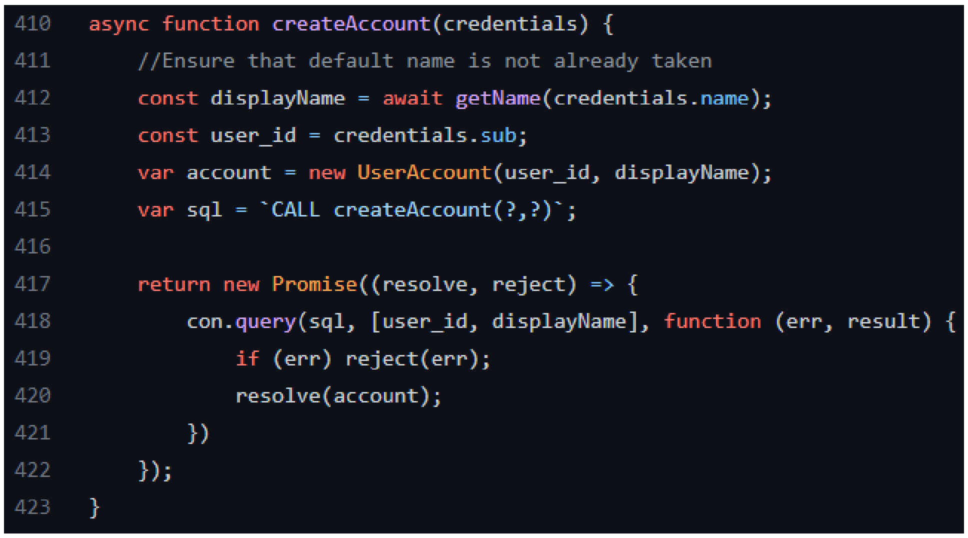
Backend: Created an sql procedures for the users database (found in backend/users/database/user_procedures.sql) which the user server now uses to interface with the database rather than hardcoding queries. Example of code before and after is provided below

Code before in users_server.js:



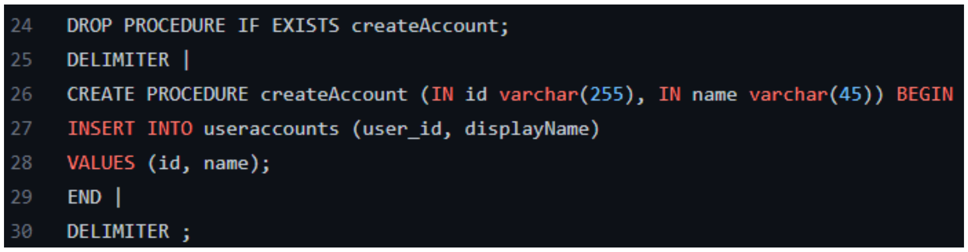
```
async function createAccount(credentials) {
    //Ensure that default name is not already taken
    var displayName = await getName(credentials.name);
    const user_id = credentials.sub;
    var user = new UserAccount(displayName, 0, Difficulty.Easy, 0, user_id);
    var score = user.score;
    var leaderboardParticipant = user.leaderboardParticipant;
    var difficulty = user.difficulty;
    var sql = `INSERT INTO useraccounts (user_id,displayName,score,leaderboardParticipant,difficulty)
        VALUES ('${user_id}','${displayName}','${score}','${leaderboardParticipant}','${difficulty}')`;

    return new Promise((resolve, reject) => {
        con.query(sql, function (err, result) {
            if (err) reject(err);
            console.log("Added account" + user_id);
            resolve(user);
        })
    });
}
```

Code after in users_server.js:



```
async function createAccount(credentials) {
    //Ensure that default name is not already taken
    const displayName = await getName(credentials.name);
    const user_id = credentials.sub;
    var account = new UserAccount(user_id, displayName);
    var sql = `CALL createAccount(?,?)`;

    return new Promise((resolve, reject) => {
        con.query(sql, [user_id, displayName], function (err, result) {
            if (err) reject(err);
            resolve(account);
        })
    });
}
```

Code after in user_procedures.sql

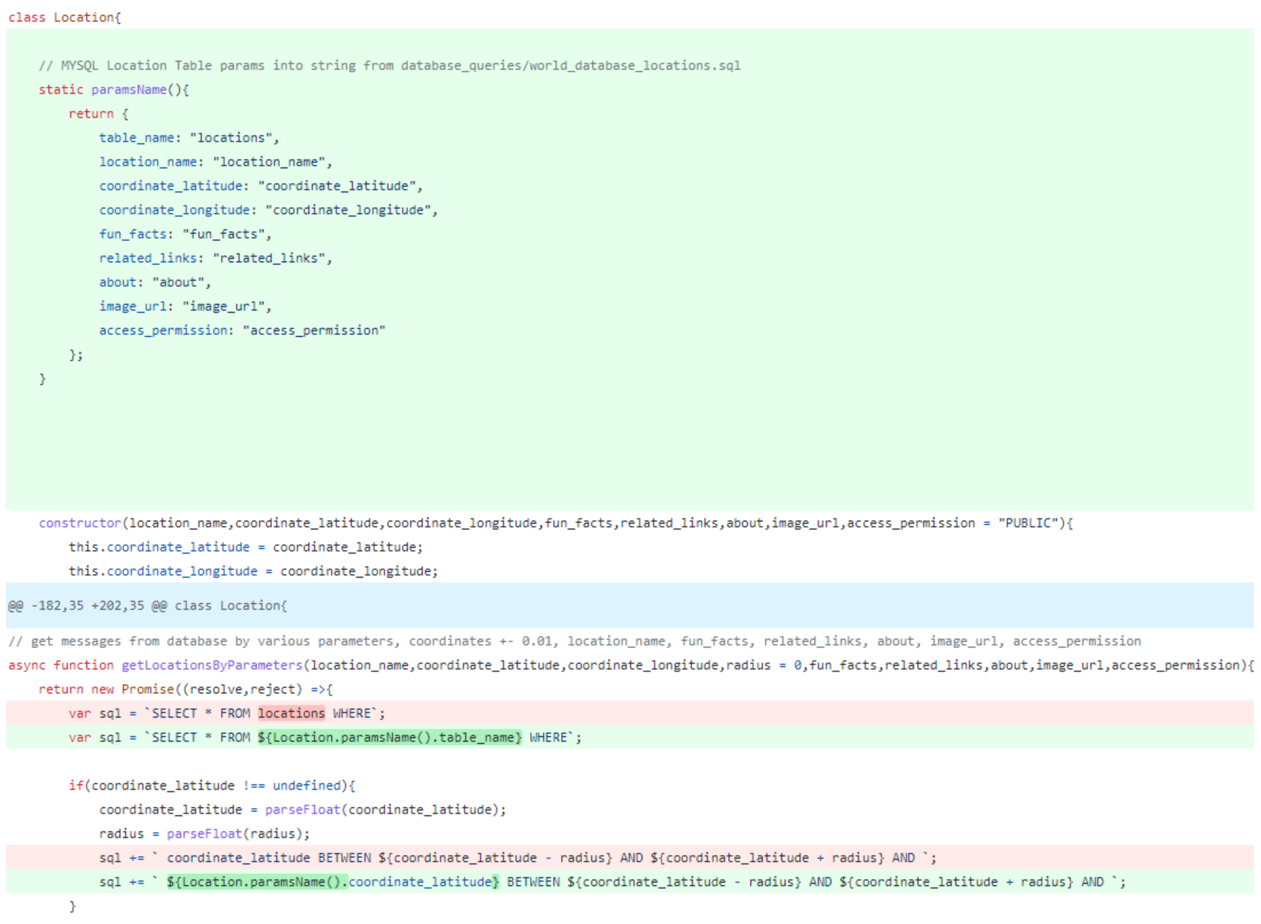


```
DROP PROCEDURE IF EXISTS createAccount;
DELIMITER |
CREATE PROCEDURE createAccount (IN id varchar(255), IN name varchar(45)) BEGIN
INSERT INTO useraccounts (user_id, displayName)
VALUES (id, name);
END |
DELIMITER ;
```

Backend: Adjusted structure of users_server and added comments to break up code for better distinction between layers. Added comments at the start and end of different parts (controller, userstore, useraccount). Moved lower level functionality like

UserAccount to the top of the file and higher level functionality like routing to the bottom of the file.
Note: After fixing this identified issue, codacy reported an unnecessary try block in the run function in users_server.js which we chose to ignore for similar reasons discussed in Piazza post @405.

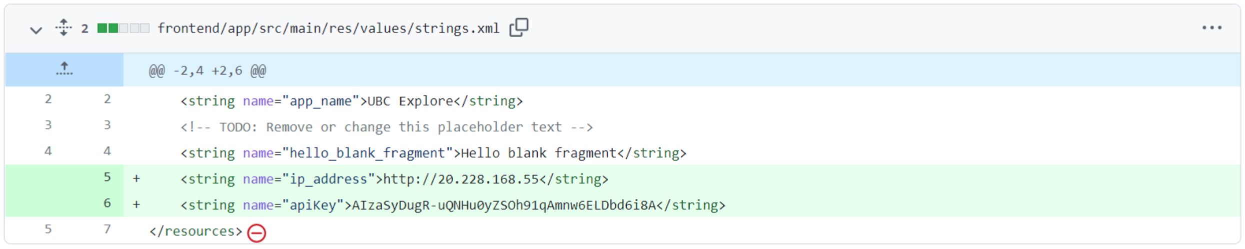
Backend: Made MySQL queries softcoded for locations and messages server. If a table column name is changed, only one variable must be changed to change all other MySQL queries.



```
class Location{

    // MYSQL Location Table params into string from database_queries/world_database_locations.sql
    static paramsName(){
        return {
            table_name: "locations",
            location_name: "location_name",
            coordinate_latitude: "coordinate_latitude",
            coordinate_longitude: "coordinate_longitude",
            fun_facts: "fun_facts",
            related_links: "related_links",
            about: "about",
            image_url: "image_url",
            access_permission: "access_permission"
        };
    }

    constructor(location_name,coordinate_latitude,coordinate_longitude,fun_facts,related_links,about,image_url,access_permission = "PUBLIC"){
        this.coordinate_latitude = coordinate_latitude;
        this.coordinate_longitude = coordinate_longitude;
@@ -182,35 +202,35 @@ class Location{
// get messages from database by various parameters, coordinates +- 0.01, location_name, fun_facts, related_links, about, image_url, access_permission
async function getLocationsByParameters(location_name,coordinate_latitude,coordinate_longitude,radius = 0,fun_facts,related_links,about,image_url,access_permission){
    return new Promise((resolve,reject) =>{
        var sql = `SELECT * FROM locations WHERE`;
        var sql = `SELECT * FROM ${Location.paramsName().table_name} WHERE`;

        if(coordinate_latitude !== undefined){
            coordinate_latitude = parseFloat(coordinate_latitude);
            radius = parseFloat(radius);
            sql += ` coordinate_latitude BETWEEN ${coordinate_latitude - radius} AND ${coordinate_latitude + radius} AND `;
            sql += ` ${Location.paramsName().coordinate_latitude} BETWEEN ${coordinate_latitude - radius} AND ${coordinate_latitude + radius} AND `;
        }
```

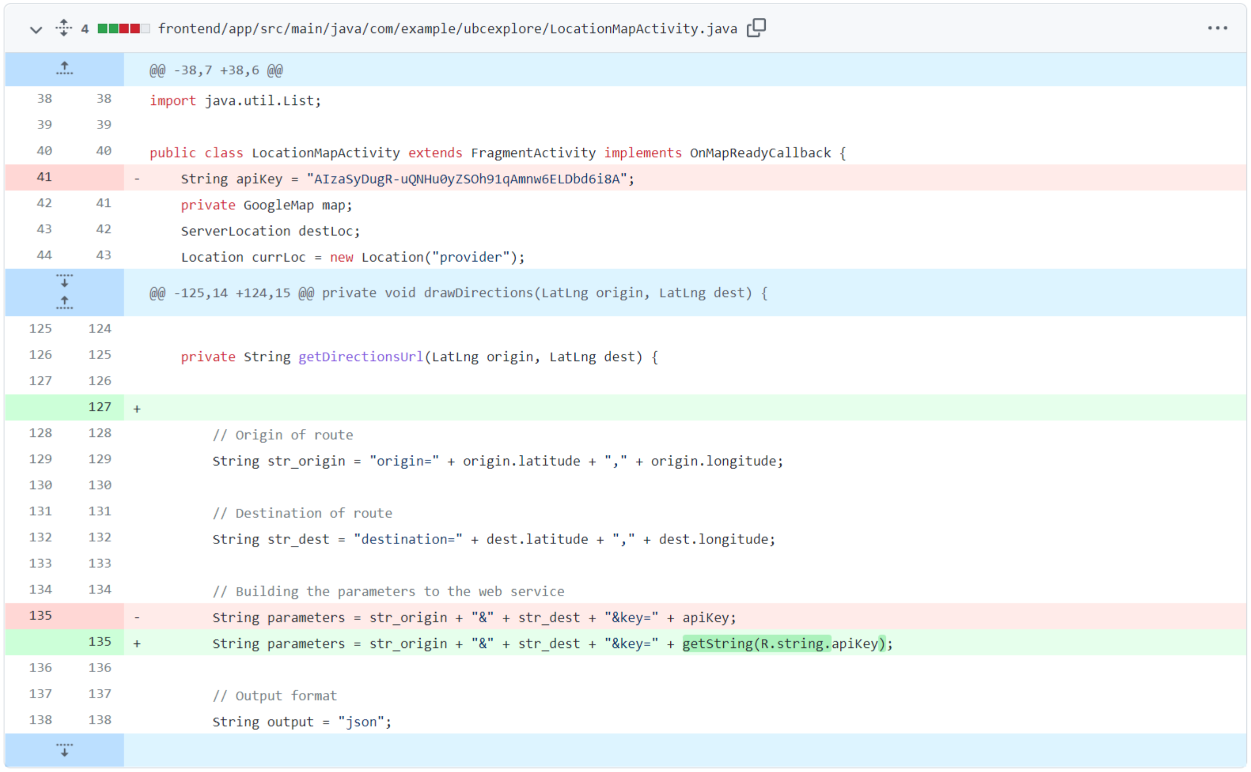
More location changes are in the commit a9778b388cf9b02c598c1db1873a8d574333cda4.
More message changes are in the commit f41c13ffa487240b6fc9eda627d88902ce4a63ad.

Frontend: moved the server address and api key value to strings.xml and updated corresponding parts in the files.



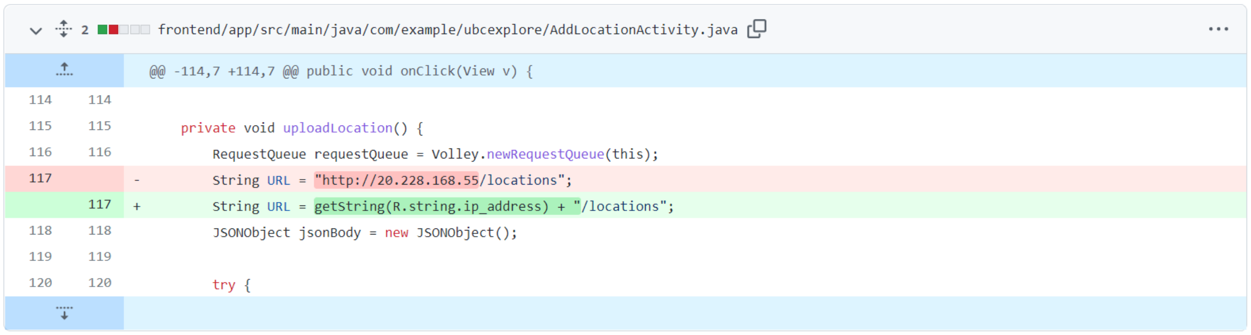
2 frontend/app/src/main/res/values/strings.xml

```diff
@@ -2,4 +2,6 @@
     <string name="app_name">UBC Explore</string>
     <!-- TODO: Remove or change this placeholder text -->
     <string name="hello_blank_fragment">Hello blank fragment</string>
+    <string name="ip_address">http://20.228.168.55</string>
+    <string name="apiKey">AIzaSyDugR-uQNHu0yZSOh91qAmnw6ELDbd6i8A</string>
 </resources>
```

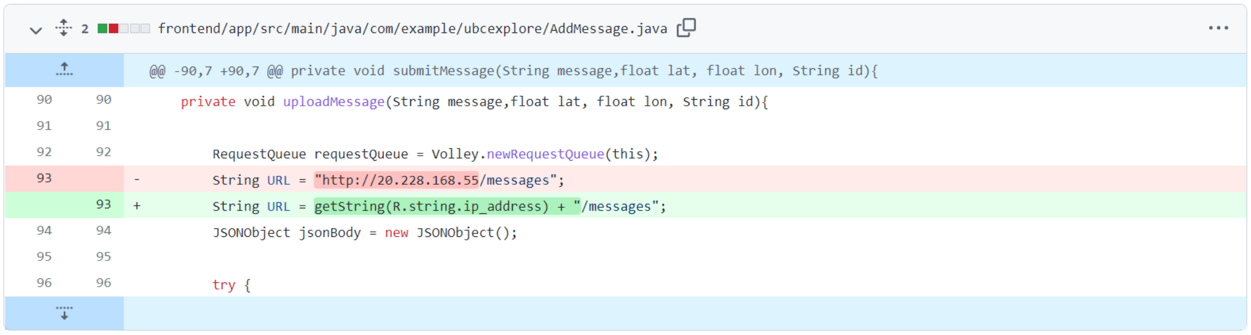


4 frontend/app/src/main/java/com/example/ubcexplore/LocationMapActivity.java

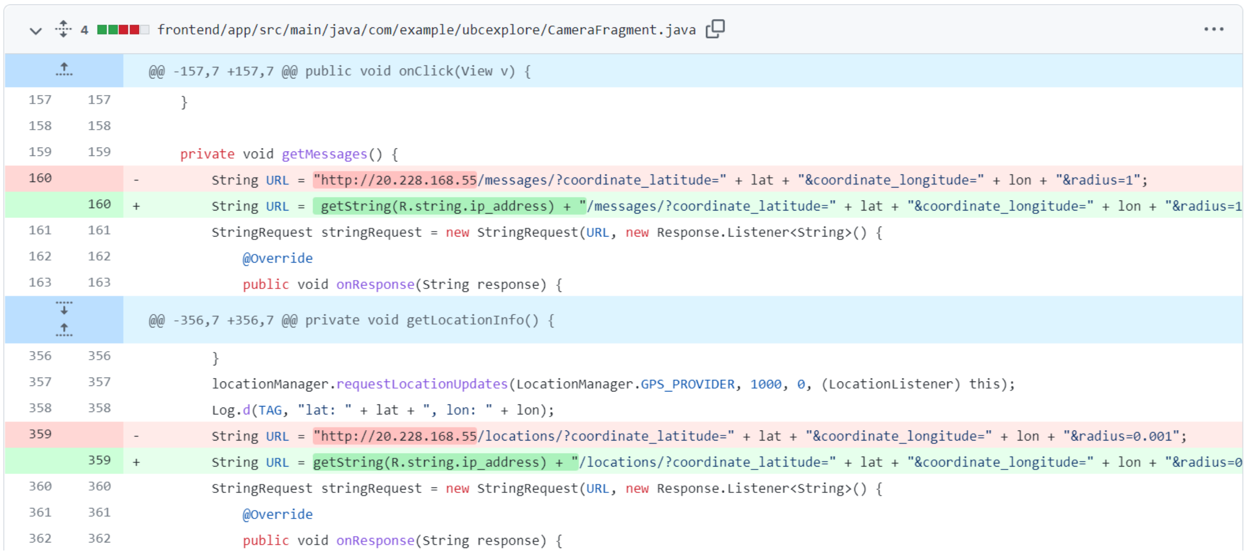
```diff
@@ -38,7 +38,6 @@
 import java.util.List;
 
 public class LocationMapActivity extends FragmentActivity implements OnMapReadyCallback {
-    String apiKey = "AIzaSyDugR-uQNHu0yZSOh91qAmnw6ELDbd6i8A";
     private GoogleMap map;
     ServerLocation destLoc;
     Location currLoc = new Location("provider");
@@ -125,14 +124,15 @@ private void drawDirections(LatLng origin, LatLng dest) {
 
     private String getDirectionsUrl(LatLng origin, LatLng dest) {
 
+
         // Origin of route
         String str_origin = "origin=" + origin.latitude + "," + origin.longitude;
 
         // Destination of route
         String str_dest = "destination=" + dest.latitude + "," + dest.longitude;
 
         // Building the parameters to the web service
-        String parameters = str_origin + "&" + str_dest + "&key=" + apiKey;
+        String parameters = str_origin + "&" + str_dest + "&key=" + getString(R.string.apiKey);
 
         // Output format
         String output = "json";
```



```
frontend/app/src/main/java/com/example/ubcexplore/AddLocationActivity.java
@@ -114,7 +114,7 @@ public void onClick(View v) {

    private void uploadLocation() {
        RequestQueue requestQueue = Volley.newRequestQueue(this);
-       String URL = "http://20.228.168.55/locations";
+       String URL = getString(R.string.ip_address) + "/locations";
        JSONObject jsonBody = new JSONObject();

        try {
```



```
frontend/app/src/main/java/com/example/ubcexplore/AddMessage.java
@@ -90,7 +90,7 @@ private void submitMessage(String message,float lat, float lon, String id){
    private void uploadMessage(String message,float lat, float lon, String id){

        RequestQueue requestQueue = Volley.newRequestQueue(this);
-       String URL = "http://20.228.168.55/messages";
+       String URL = getString(R.string.ip_address) + "/messages";
        JSONObject jsonBody = new JSONObject();

        try {
```



```
frontend/app/src/main/java/com/example/ubcexplore/CameraFragment.java
@@ -157,7 +157,7 @@ public void onClick(View v) {
    }

    private void getMessages() {
-       String URL = "http://20.228.168.55/messages/?coordinate_latitude=" + lat + "&coordinate_longitude=" + lon + "&radius=1";
+       String URL =  getString(R.string.ip_address) + "/messages/?coordinate_latitude=" + lat + "&coordinate_longitude=" + lon + "&radius=1
        StringRequest stringRequest = new StringRequest(URL, new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
@@ -356,7 +356,7 @@ private void getLocationInfo() {
        }
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1000, 0, (LocationListener) this);
        Log.d(TAG, "lat: " + lat + ", lon: " + lon);
-       String URL = "http://20.228.168.55/locations/?coordinate_latitude=" + lat + "&coordinate_longitude=" + lon + "&radius=0.001";
+       String URL = getString(R.string.ip_address) + "/locations/?coordinate_latitude=" + lat + "&coordinate_longitude=" + lon + "&radius=0
        StringRequest stringRequest = new StringRequest(URL, new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
```

Other file changes are similar and can be found in commit (fdf21c1eeed5adc749133ab719b5d29fe2fb63c4)

Member Contributions:

- Akshat: Fixed codacy-related issues and identified issues with locations, and messages module. Reviewed peer group's backend code.
- Jane: Fixed codacy related issues for frontend and fixed identified issues in frontend.

- Dylan: Fixed codacy related issues and identified issues with users module.
- Mei: Reviewed peer group’s frontend code.