# CodeAce

By Lidor Amraby, Konstantin Burykin
and Maya Abend

**CodeAce** is an innovative platform designed to connect individuals who have coding questions with experts who can provide solutions. Users can upload their coding questions to the app, and in return, they receive help from experienced coders through various communication channels such as phone, email, or direct messaging.

**Who Uses CodeAce?**

- **Students**: Whether they are in high school, college, or pursuing online courses, students often encounter challenging coding problems. CodeAce provides them with a reliable resource to get help quickly.
- **Professional Developers**: Even experienced developers sometimes need a second opinion or help with complex issues. CodeAce offers a community-driven support system.
- **Hobbyists**: Individuals who code as a hobby can find answers to their questions and improve their skills with the help of the community.
- **Educators**: Teachers and mentors can use CodeAce to support their students outside of class, providing additional guidance and resources.

**How does CodeAce work?**
1. **Upload Questions**: Users post their coding questions, providing as much detail as possible, including code snippets and error messages.
2. **Get Help**: Other users, including experienced coders and mentors, review the questions and offer solutions. They can reach out via phone, email, etc.

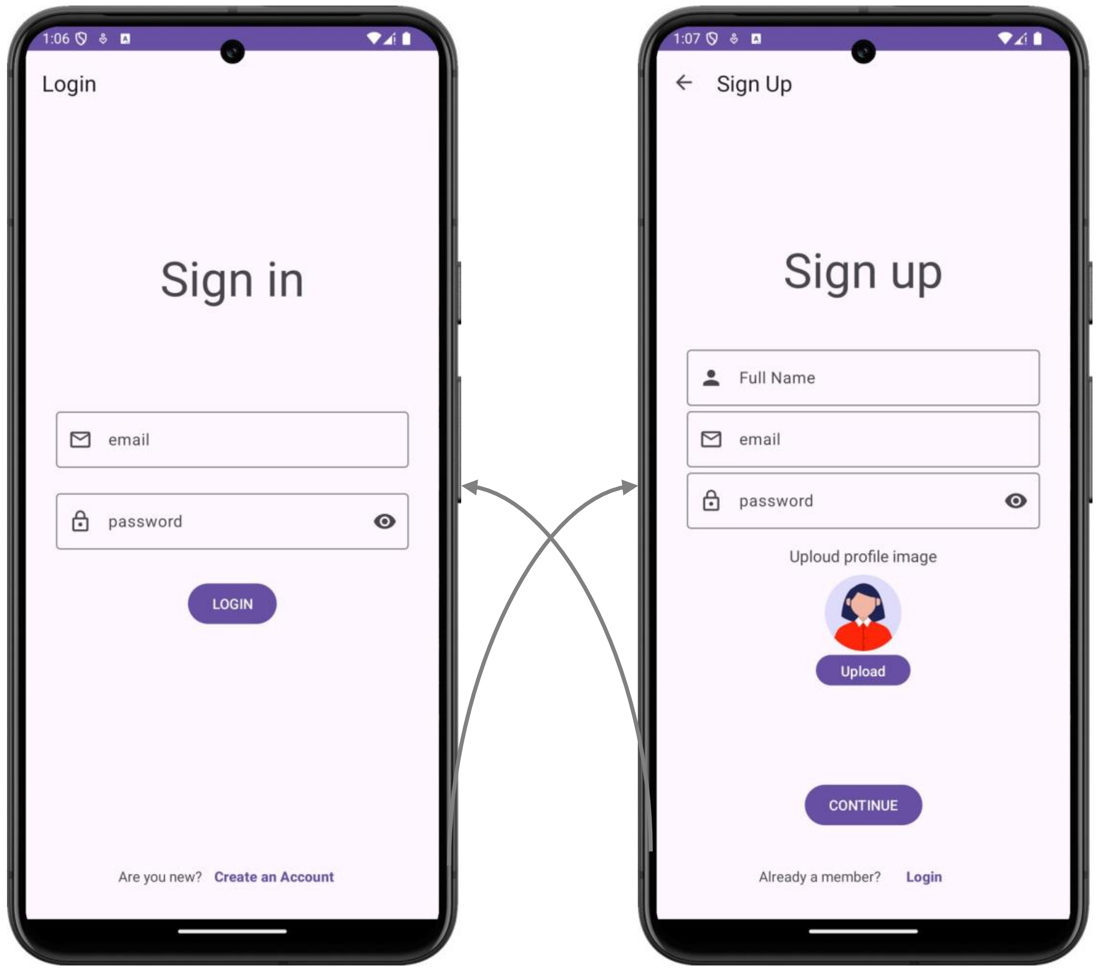
**Features**
- **User-Friendly Interface**: An intuitive design that makes it easy for users to post questions and navigate the platform.
- **Community Engagement**: Encourages a community of learners and experts who support each other.
- I**mpact on the World**

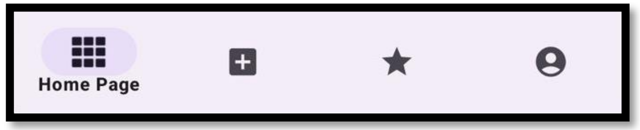
**CodeAce** contributes to making the world a smarter and nicer place in several ways:

- **Promotes Knowledge Sharing**: By connecting people who need help with those who can provide it, CodeAce fosters a culture of knowledge sharing and continuous learning.
- **Supports Lifelong Learning**: Users of all ages and skill levels can improve their coding abilities, promoting lifelong learning and skill development.
- **Encourages Collaboration**: The app builds a sense of community among coders, encouraging collaboration and mutual support.

## התחברות והרשמה

- Including the use of Firebase for user authentication
  Storing user data such as email and password in Firebase Authentication.
- Using Sign-Up (SignUpFragment.kt) and Login (LogInFragment.kt) functions, FirebaseUserModel provides functions for managing users.
- Files where Firebase is related to:
  FirebaseUserModel.kt
  UserRepository.kt
  UserViewModel.kt
  SignUpFragment.kt
  LogInFragment.kt

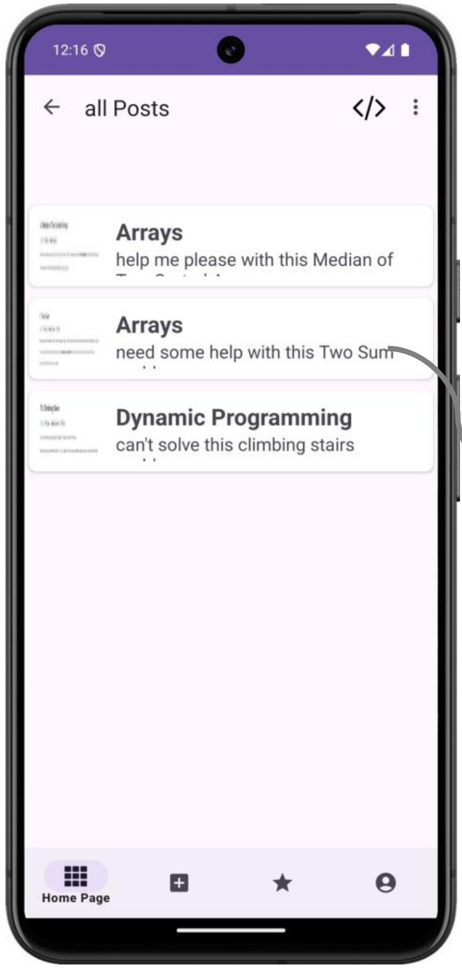




- the main screen is a list of posts Includes the option to add, edit, and delete posts
- Uses RecyclerView to display the list
- Checks user authentication before allowing edits

- Files where functionality is implemented:
    - MainActivity.kt
    - AllPostsFragment.kt
    - PostViewModel.kt
    - PostRepository.kt
    - FirebasePostModel.kt
    - PostsRecyclerAdapter.kt
    - FirebaseUserModel.kt
    - AddPostFragment.kt
    - EditDetailPostFragment.kt

## מסך הבית

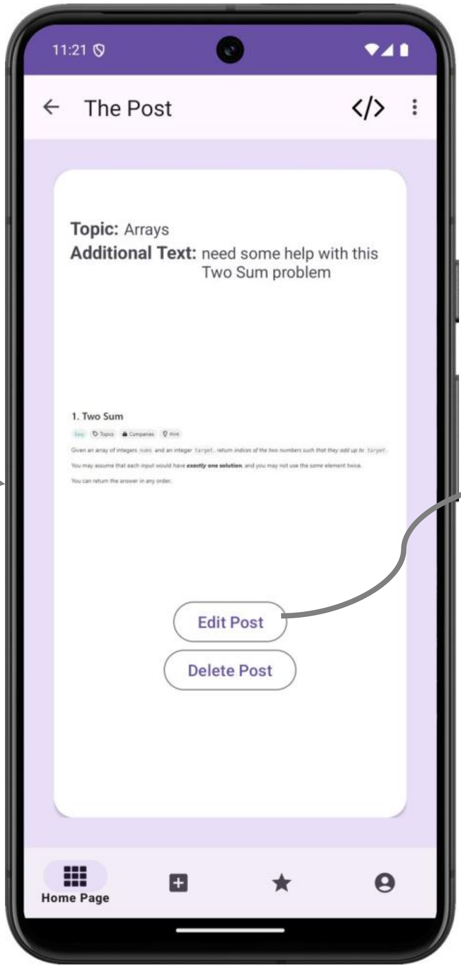


Arrays
help me please with this Median of

Arrays
need some help with this Two Sum

Dynamic Programming
can't solve this climbing stairs

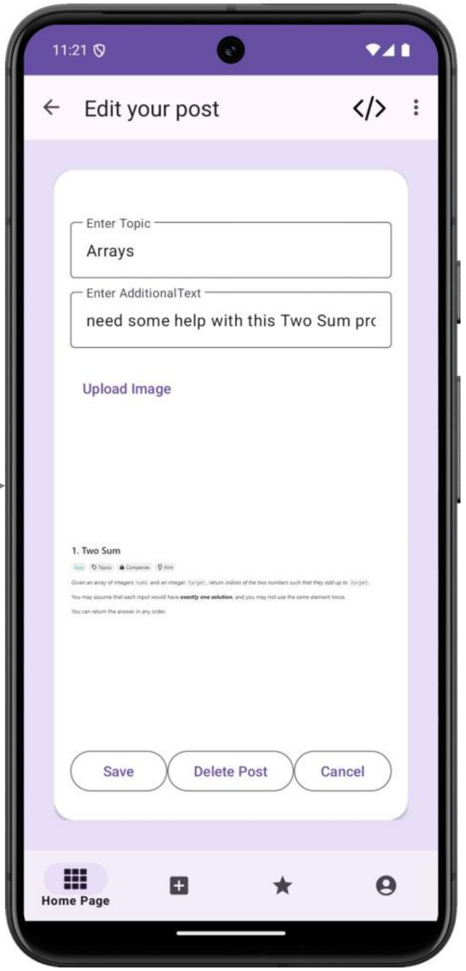
## תצוגת פוסט



**Topic:** Arrays
**Additional Text:** need some help with this Two Sum problem

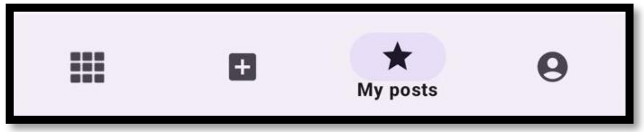
1. Two Sum

## עריכת פוסט



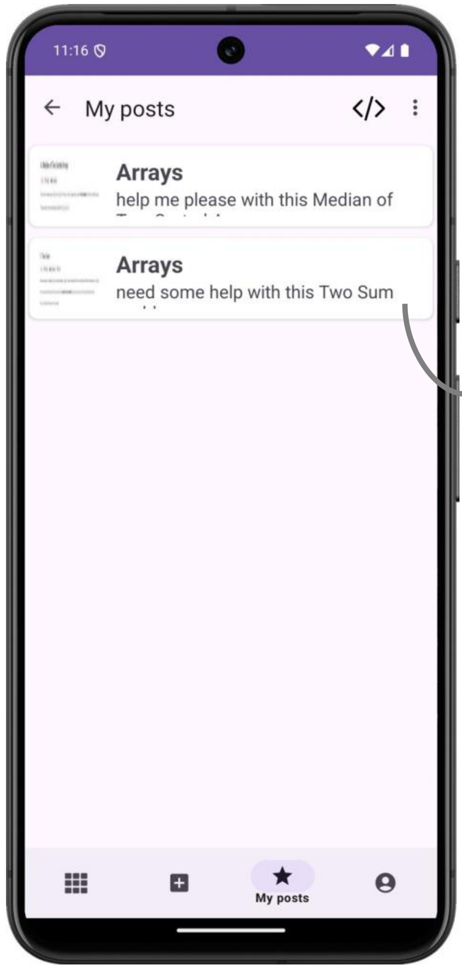
Enter Topic: Arrays
Enter AdditionalText: need some help with this Two Sum prc

1. Two Sum



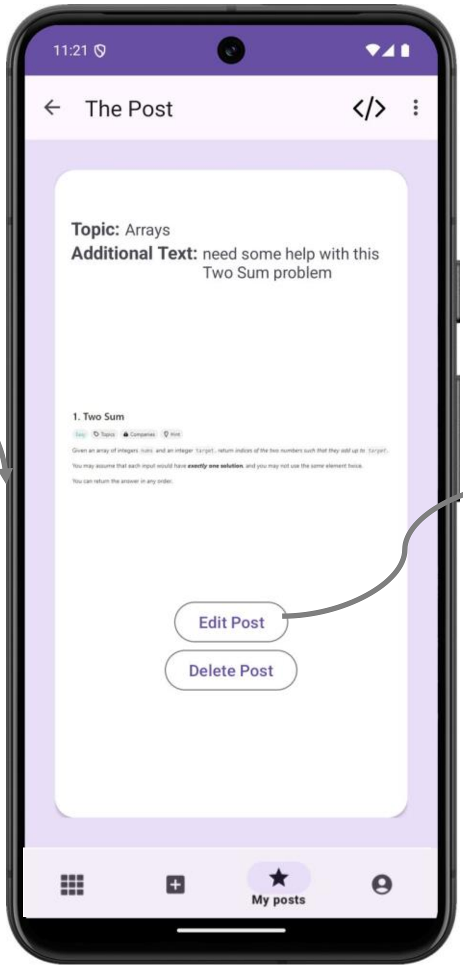
- Explanation of user authentication check:
- AddPostFragment.kt and EditDetailPostFragment.kt ensure that only authenticated users can add or edit posts.
- Before allowing a user to edit a post, the app checks if the current user is the creator of the post by comparing user emails.

## הפוסטים שלי



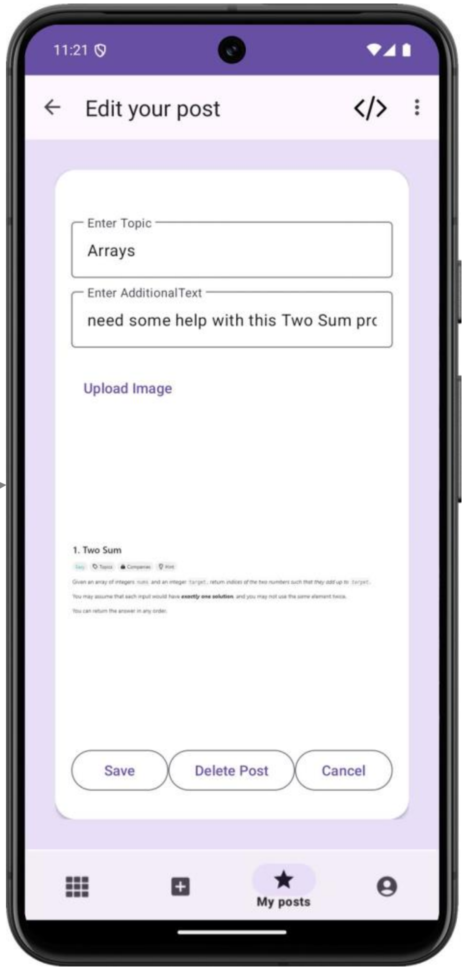
11:16
My posts
Arrays
help me please with this Median of
Arrays
need some help with this Two Sum
My posts

## תצוגת פוסט

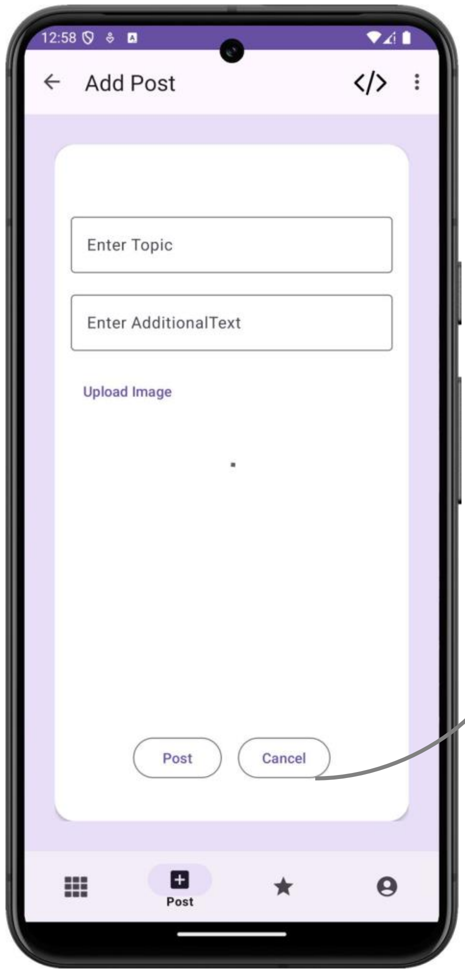


Topic: Arrays
Additional Text: need some help with this Two Sum problem

## עריכת פוסט



Enter Topic: Arrays
Enter AdditionalText: need some help with this Two Sum prc

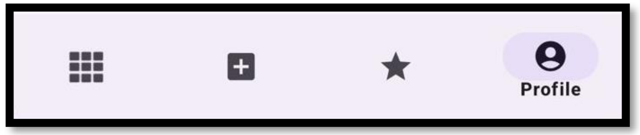


Back to previous fragment

# הוספת פוסט

- Editing an existing post
- Allows changing text and adding/replacing a photo
- Uses Firebase and Local Database (Room) for synchronization
- Check if the current user is the creator of the post before allowing edits

- Files where functionality is implemented:
  - EditDetailPostFragment.kt
  - PostViewModel.kt
  - PostRepository.kt
  - FirebasePostModel.kt
  - FirebaseUserModel.kt

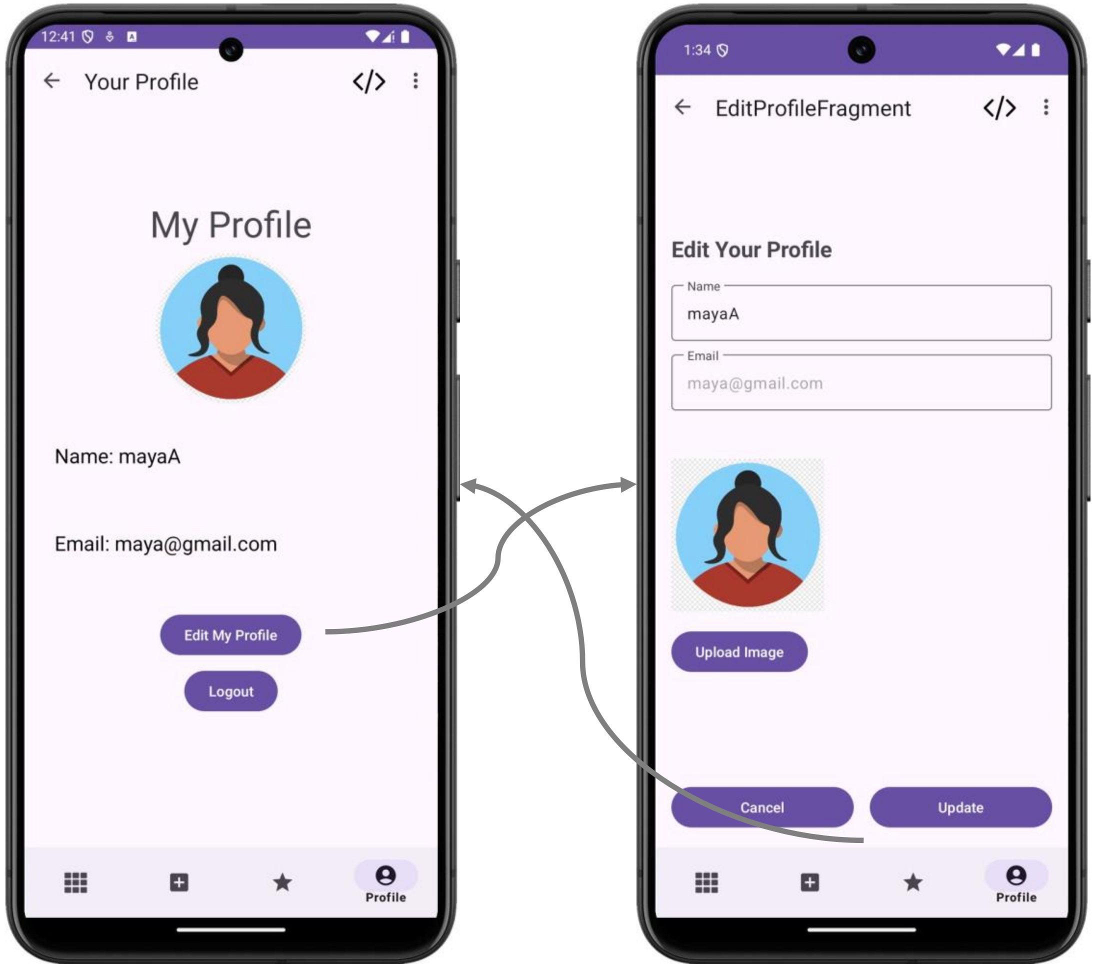


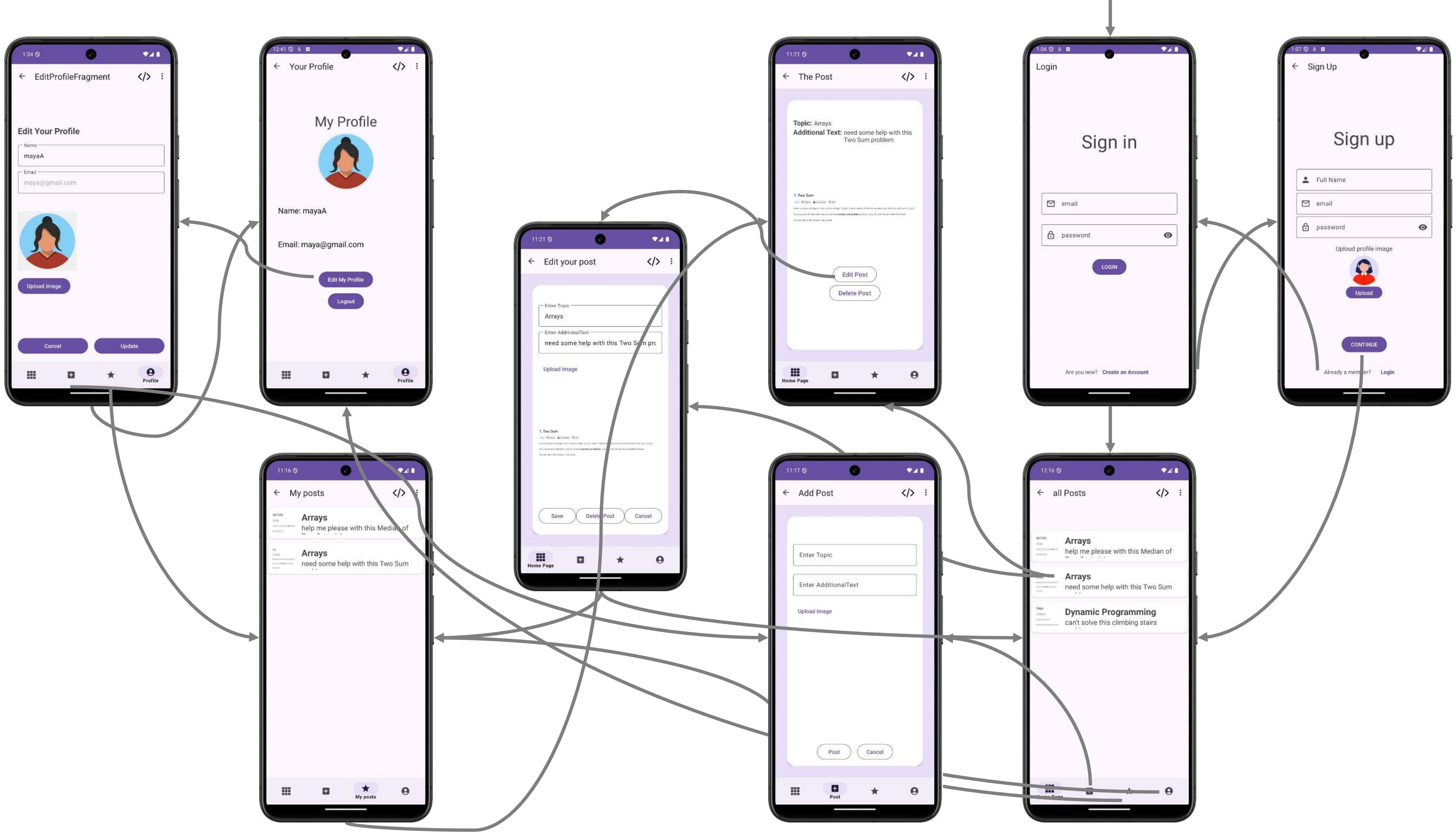
- Displaying the user's profile
- Shows user's personal details and profile picture
- Allows editing of profile information
- Synchronizes profile updates with Firebase Firestore
- Files where functionality is implemented:
  - ProfileFragment.kt
  - EditProfileFragment.kt
  - FirebaseUserModel.kt
  - UserRepository.kt

- Explanation of user authentication check:
- ProfileFragment.kt and EditProfileFragment.kt use through the UserViewModel, UserRep and FirebaseUserModel to ensure the user is authenticated.
- Fetches and updates user profile details only if the user is authenticated.

## הפרופיל שלי

## עריכת פרופיל




EditProfileFragment
Edit Your Profile
Name
mayaA
Email
maya@gmail.com
Upload Image
Cancel
Update
Profile
Your Profile
My Profile
Name: mayaA
Email: maya@gmail.com
Edit My Profile
Logout
Profile
Edit your post
Enter Topic
Arrays
Enter AdditionalText
need some help with this Two Sum pro
Upload Image
1. Two Sum
Save
Delete Post
Cancel
Home Page
The Post
Topic: Arrays
Additional Text: need some help with this Two Sum problem
1. Two Sum
Edit Post
Delete Post
Home Page
Login
Sign in
email
password
LOGIN
Are you new? Create an Account
Sign Up
Sign up
Full Name
email
password
Uploud profile image
Upload
CONTINUE
Already a member? Login
My posts
Arrays
help me please with this Median of
Arrays
need some help with this Two Sum
My posts
Add Post
Enter Topic
Enter AdditionalText
Upload Image
Post
Cancel
Post
all Posts
Arrays
help me please with this Median of
Arrays
need some help with this Two Sum
Dynamic Programming
can't solve this climbing stairs
Home Page

## In this project, we incorporated several key technologies and components to enhance functionality and performance. These include:

- RecyclerView
- MVVM Architecture
- LiveData
- Room Library
- Navigation Component
- Coroutines
- Google Mobile Services (GMS)
- Firebase

## Usage of RecyclerView:

- Displaying Lists of Posts:
- Main screen shows a list of posts using RecyclerView.
- Allows users to add, edit, and delete posts from the list.
- Implements a delete confirmation dialog.

- Files Where RecyclerView is Used:
- AllPostsFragment.kt:
- Displays all posts using RecyclerView.
- Handles item clicks to show post details and long clicks for additional actions.
- Uses PostsRecyclerAdapter to bind data to RecyclerView.

- PostsRecyclerAdapter.kt:
- Custom adapter for RecyclerView in AllPostsFragment.
- Binds post data to the views.
- Handles click events for each item in the list.

- MyPostsFragment.kt:
- Displays posts created by the current user using RecyclerView.
- Allows users to manage their posts (edit or delete).

## MVVM Architecture Implementation

**Purpose of MVVM Architecture:**

- **Separation of Concerns**: MVVM (Model-View-ViewModel) architecture separates the UI, business logic, and data access layers, making the codebase more modular, testable, and maintainable.
- **Reactive Programming**: MVVM leverages LiveData and ViewModel to ensure that the UI reacts to data changes automatically.

**Components of MVVM:**

1. **Model**:
    - Represents the data layer of the application.
    - Includes data models, database access through Room, and network access through Firebase.
2. **View**:
    - Represents the UI layer.
    - Includes Activities and Fragments that display data and handle user interactions.
3. **ViewModel**:
    - Acts as a mediator between the View and the Model.
    - Holds and manages UI-related data, handles business logic, and updates the View through LiveData.

## LiveData Implementation

**Purpose of LiveData:**

- **Reactive Programming**: LiveData is used to build reactive UIs by observing data changes and updating the UI accordingly.
- **Lifecycle Awareness**: LiveData is lifecycle-aware, meaning it respects the lifecycle of other app components, such as activities and fragments. This ensures that LiveData only updates observers that are in an active lifecycle state.

**Files Where LiveData is Used:**

1. **PostViewModel.kt**:
   - Manages and holds the data for posts.
   - Uses LiveData to expose data to the UI and ensure updates are handled reactively.

2. **UserViewModel.kt**:
   - Manages and holds the data for user profiles.
   - Uses LiveData to expose user data to the UI and ensure updates are handled reactively.

3. **AllPostsFragment.kt**:
   - Observes LiveData from PostViewModel to get the list of posts.
   - Updates the RecyclerView whenever the data changes.

4. **ProfileFragment.kt**:
   - Observes LiveData from UserViewModel to get user profile data.
   - Updates the UI elements whenever the user data changes.

## Room Library Implementation

**Purpose of Room Library:**

- **Local Database**: Room provides an abstraction layer over SQLite to allow fluent database access while harnessing the full power of SQLite.
- **Simplified Database Operations**: Room simplifies database operations with annotations and generates the necessary code at compile time.
- **Lifecycle Awareness**: Room works seamlessly with LiveData and ViewModel to ensure data updates respect the lifecycle of app components.

**Files Where Room Library is Used:**

1. **Post.kt**:
   - Defines the data entity for posts stored in the Room database.
   - Annotated with @Entity to indicate a table in the database.
2. **PostDao.kt**:
   - Data Access Object (DAO) for the Post entity.
   - Contains methods for accessing the database, such as inserting, updating, and deleting posts, as well as querying the list of posts.
3. **PostDatabase.kt**:
   - Defines the database configuration.
   - Annotated with @Database to represent the database holder and serves as the main access point for the underlying connection to the app's persisted data.
4. **PostRepository.kt**:
   - Manages data operations and acts as a mediator between the data sources (Room database and Firebase).
   - Ensures that the app works offline by storing data locally and synchronizing it with Firebase.

## Room Library Implementation

**5. User.kt**:
- Defines the data entity for users stored in the Room database.
- Annotated with @Entity to indicate a table in the database.

**6. UserDao.kt**:
- Data Access Object (DAO) for the User entity.
- Contains methods for accessing the database, such as inserting, updating, and deleting user profiles, as well as querying user data.

**7. UserDatabase.kt**:
- Defines the database configuration.
- Annotated with @Database to represent the database holder and serves as the main access point for the underlying connection to the app's persisted data.

**8. UserRepository.kt**:
- Manages data operations and acts as a mediator between the data sources (Room database and Firebase).
- Ensures that the app works offline by storing data locally and synchronizing it with Firebase.

## Navigation Component Implementation

**Purpose of Navigation Component:**

- **Simplified Navigation**: The Navigation Component simplifies the implementation of navigation between different fragments and activities.
- **Safe Args**: Provides type-safe arguments when navigating between destinations, reducing the risk of runtime errors.
- **Back Stack Management**: Manages the back stack automatically, ensuring a consistent and predictable navigation experience.

## Coroutines Implementation

**Purpose of Coroutines:**

- **Asynchronous Programming**: Coroutines provide a way to write asynchronous code that is easy to read and maintain.
- **Efficient Background Operations**: Coroutines are lightweight and can efficiently manage background operations without blocking the main thread.
- **Concurrency**: Coroutines handle multiple tasks concurrently, providing a responsive UI experience.

**Files Where Coroutines are Used:**

1. **PostRepository.kt**:
   - Handles data operations for posts in a background thread using coroutines.
   - Manages adding, updating, and deleting posts asynchronously.

2. **UserRepository.kt**:
   - Handles data operations for user profiles in a background thread using coroutines.
   - Manages updating user profiles asynchronously.

3. **PostViewModel.kt**:
   - Uses coroutines to launch data operations from the ViewModel scope.
   - Ensures that UI-related data operations are performed in the background without blocking the main thread.

4. **UserViewModel.kt**:
   - Uses coroutines to launch data operations from the ViewModel scope.
   - Ensures that user profile operations are performed in the background

## Google Mobile Services (GMS) Implementation

**Purpose of Google Mobile Services:**

- **Enhance User Experience**: GMS provides APIs that help improve app functionality and user experience.
- **Location Services**: Access location data for various functionalities such as geo-tagging posts or user locations.
- **Other APIs**: Includes APIs for authentication, cloud messaging, and more.

**Files Where GMS is Used:**

1. **PostRepository.kt**:
    - Utilizes GMS for real-time database synchronization and cloud storage.
2. **UserRepository.kt**:
    - Uses GMS for user authentication and profile management.
3. **FirebasePostModel.kt**:
    - Integrates Firebase for data storage and retrieval.
4. **FirebaseUserModel.kt**:
    - Manages user data in Firebase, including authentication.
5. **Location Services** (Hypothetical Example if Used):
    - Could be integrated in fragments or activities to access user location.

## Firebase Implementation

**Purpose of Firebase:**

- **Firestore Database**: Firebase is used to store and synchronize data, allowing seamless data updates across all clients.
- **User Authentication**: Firebase Authentication is used to manage user sign-up, login, and profile management.
- **Cloud Storage**: Firebase Cloud Storage is used to store and retrieve media files, such as user profile pictures and post images.

**Data Stored in Firebase:**

1. **Posts**:
   - Post details such as topic, additional text, image URL, publisher, and email.
   - Allows users to add, edit, and delete posts, with changes reflected in real-time.
2. **Users**:
   - User details such as name, email, and profile image URL.
   - Manages user authentication and profile updates.

**Files Where Firebase is Used:**

1. **FirebasePostModel.kt**:
   - Handles Firebase operations related to posts.
   - Functions for adding, updating, and deleting posts in Firebase.
2. **FirebaseUserModel.kt**:
   - Handles Firebase operations related to users.
   - Functions for adding, updating, and retrieving user profiles from Firebase.
3. **PostRepository.kt**:
   - Integrates Firebase operations for posts and manages data synchronization between Firebase and Room database.

## Firebase Implementation

4. **UserRepository.kt**:
   - Integrates Firebase operations for user profiles and manages data synchronization between Firebase and Room database.
5. **SignUpFragment.kt**:
   - Handles user registration using Firebase Authentication.
6. **LogInFragment.kt**:
   - Handles user login using Firebase Authentication.
7. **ProfileFragment.kt**:
   - Retrieves and updates user profile information from Firebase.
8. **AddPostFragment.kt**:
   - Adds new posts to Firebase.
9. **EditDetailPostFragment.kt**:
   - Updates existing posts in Firebase.