COLLEGE CODE : 9636

COLLEGE NAME : ARUNACHALA HITECH ENGINEERING COLLEGE DEPARTMENT STUDENT : COMPUTER SCIENCE AND ENGINEERING

NM-ID:3876EEA6EFEAE96918849B64CA22

ROLLNO : 963623104030

DATE : 22/09/2025

Completed the project named as Phase:03

TECHNOLOGY PROJECT NAME : NEWS FEED

APPLICATION

SUBMITTED BY,

NAME : SANTHOSH. A

MOBILE NO:9600634637

MVP Implementation

Project Setup

Setting up a project for a content feed application involves the following steps across different platforms: Android (Kotlin), iOS (Swift), and Web (React).

1**. Project Creation:**

- For Android, open Android Studio, select "Start a new Android Studio project," choose "Empty Activity," configure project details (name, package name, and Kotlin), and set the Minimum SDK (API 24 or 26). Click "Finish." - For Web, use terminal commands: `npx create-react-app my-news-feed-app` or `npm create vite@latest my-news-feed-app --template react`. Change to the directory with `cd my-news-feed-app`.

- For iOS, in Xcode, select "Create a new Xcode project," choose the "App" template, enter a Product Name, select Swift and interface type (SwiftUI recommended), and save.

2**. Adding Dependencies:**

- In Android, add dependencies in `build.gradle.kts`: Retrofit, GSON, Glide, and lifecycle components. Click "Sync Now." - In Web, install libraries like Axios and React Router using `npm install`.

- In iOS, use Swift Package Manager (SPM) or CocoaPods to add networking (Alamofire) and image loading (Kingfisher) dependencies.

3. **Permissions**:

- For Android, add `` in `AndroidManifest.xml`.

- iOS requires no explicit permission but may need "Privacy - App Transport Security Settings" for unsecured requests.

4. **API Key Setup:**

- Sign up for a news API to obtain a key and securely store it in configuration files rather than hardcoding it to protect it from version control exposure.

Core Features implementation

Implementing core features for a content feed application requires integrating UI, API, and data handling.

1. UI Setup: Start with the main activity/view to hold the feed, featuring a component for item display (e.g., RecyclerView for Android, UICollectionView for iOS, or a for web apps).

● Then, design the article card layout to be repeated for each article, including an ImageView for thumbnails, and TextViews for the title, source, and description.

2. API Integration and Data Fetching: Incorporate a networking library (e.g., Retrofit for Android, Axios for web).

Create an API service to define the necessary API endpoints, such as a GET request to /top-headlines. Make

asynchronous requests to fetch JSON responses with article data. Implement error handling for network issues or invalid API keys, displaying user-friendly messages if data retrieval fails.

3. Data Parsing and Binding: Utilize a JSON parsing library (e.g., Gson, Moshi) to convert the JSON string into a list of data model objects. For images, use libraries like Glide or Picasso to load images from URLs into the ImageView. Finally, bind the list of Article objects to the UI through an adapter, ensuring the article cards display correctly on the feed.

Data Storage (Local State / Database)

For a content feed application, selecting the appropriate data storage solution is crucial, depending on data nature and lifespan. Local state refers to in-memory storage for temporary or session-based data, ideal for scenarios like API responses and UI state management, using frameworks like React or simple variables in mobile development.

Pros include speed and lightweight implementation, while the primary con is data loss on app closure. Conversely, a local database offers persistent storage, retaining data even when the app or device restarts. It's suitable for offline access, user-specific data, and historical logs. Management typically involves using databases like SQLite, with libraries such as Room for Android and Core Data or Realm for iOS, facilitating interaction. The benefits of local databases include data persistence across sessions and enhanced offline capabilities, although they come with increased complexity in setup and management. Ultimately, local state is suited for temporary data, while local databases are essential for data requiring future access or offline use.

Testing Core Features

Unit Testing: focuses on testing individual components in isolation, such as data models or API methods. Key tests include verifying data models (like the Article class) for correct data handling, ensuring JSON parsing works without errors, and validating helper functions that format or transform data.

Integration Testing: assesses the interaction between different application parts, testing things like API services using a mock server. It verifies expected outcomes from API calls and ensures error handling is adequate. Additionally, it checks if the repository accurately fetches data and communicates with the ViewModel effectively.

UI Testing aims to confirm the user interface displays data correctly and responds to interactions. Tests include ensuring a correct article list is shown post API call, validating navigation upon article interaction, and managing loading and error states appropriately.

Following these testing strategies enhances application reliability and functionality.

Version Control(GitHub)

To use GitHub for version control, start by creating a repository and initializing it in your local project. First, create a repository on GitHub with a relevant name (like news-feed-app) and optionally add a README.md and .gitignore. If you added a README, clone the repository using `git clone [https://github.com/mbsmsm?tab=repositories]` . For an existing local project, initialize a Git repository with `git init` , add files using `git add .` , commit changes, link to the remote with `git remote add origin [repository_url]` , and

push with `git push -u origin main`. Use a standard workflow: modify code, check status with `git status`, stage changes with `git add .`, commit with `git commit -m "message"`, and push updates to GitHub with `git push origin main`. For new features, create a separate branch using `git checkout -b feature/name`, make changes, commit them, and push the branch to GitHub. Finally, create a Pull Request to merge the branch into the main branch after review, then merge the approved changes.

(https://github.com/santhosha234/Newsfeed-application.git)