CSE 535: Mobile Computing

Tabs — A News Reading App

Team

Ananya Garg - 2022068	(ananya22068@iiitd.ac.in)
Ishika Gupta - 2022222	(ishika22222@iiitd.ac.in)
Swarnima Prasad - 2022525	(swarnima22525@iiitd.ac.in)

Introduction

Purpose

Tabs is a news application designed to provide users with access to news articles from various genres, fetched from online sources using hyperlinks or APIs. The app ensures a seamless reading experience by implementing caching for offline access, network connectivity handling, and error management. Tabs also enhances accessibility with features such as **Talkback (text-to-audio conversion)**, **energy-efficient screen dimming**,

Users

Tabs is intended for a diverse audience, including general readers seeking convenient access to news, professionals staying informed, and students researching current events. It will also cater to users who prefer consuming news in their native language and those in areas with limited internet connectivity.

Functional Requirements

Use Cases

Use Case	**Description**	**Precondition**	**Status**
Search	Search for news relevant to keywords	None	Completed
Voice search[ADDED]	Voice input for search	None	Completed
Saved button	To go the saved news items	None	Completed
News Card	To click on a news item and view its content	None	Completed
Talkback	To read aloud the news text	News card opened	Completed
Save button	To save a news item	News card opened	Completed
Filtering Tags	To filter the news item by selecting a specific tag	None	Completed
ThemeToggle (Light/Dark Mode)[ADDED]	Enables users to switch between light and dark themes for comfort and energy efficiency.	None	Completed

Use Case Details

1. Search[DONE]

- **Actor**: User
- **Description**: Allows the user to search the news using keywords.
- **Preconditions**: None.
- **Postconditions**: Displays the search results matching the keywords.
- **Event Flow**:
    1. The user taps on the search bar on the home page.
    2. The user enters keywords and submits the search.
    3. The system fetches matching news articles for display.

2. Voice Search[ADDED]

- **Actor:** User
- **Description:** Enables users to search for news articles using voice input via the device's microphone sensor.
- **Precondition:** None.
- **Postcondition:** Articles are filtered based on the recognized voice query.
- **Event Flow:**
    1. User taps the microphone icon in the toolbar.
    2. System checks for and requests microphone permission if not already granted.
    3. Upon permission, system activates the speech recognizer and listens.
    4. User speaks the search query.
    5. System transcribes the speech, updates the search bar, and filters articles accordingly.

3. Saved Button[DONE]

- **Actor**: User
- **Description**: Allow users to view saved news.
- **Preconditions**: None.
- **Postconditions**: Displays a list of saved news.
- **Event Flow**:
    1. The user taps on the Saved button.
    2. The system retrieves and displays saved news articles.

4. News Card[DONE]

- **Actor**: User
- **Description**: Allows the user to view an overarching full content of the news article.
- **Preconditions**: None.
- **Postconditions**: Displays full news content.
- **Event Flow**:
    1. User taps onto a news card.
    2. System fetches the news article in full view on the screen.

5. Talkback[DONE]

- **Actor**: User
- **Description**: Content of an open news article is read aloud.
- **Preconditions**: The news card should be open.

- **Postconditions**: The content of the news is read out loud.
- **Event Flow**:
    1. The user opens a news card.
    2. User activates talkback.
    3. The system reads aloud the article.

6. Save Button[DONE]

- **Actor**: User.
- **Description**: Allows the user to save a news article for later reference.
- **Precondition**: News card opened.
- **Postcondition**: The article is saved in the user's collection.
- **Event Flow**:
    1. User taps the save button on a news card.
    2. System saves the article and provides confirmation.

7. Filtering Tags[DONE]

- **Actor**: User.
- **Description**: Allows for filtering news by particular tags or categories.
- **Preconditions**: None.
- **Postcondition**: Displays news articles that match the tag selected.
- **Event Flow**:
    1. User selects a filter tag.
    2. The system filters the pertinent news articles and displays them.

8. Theme Toggle (Light/Dark Mode) [ADDED]

- **Actor:** User.
- **Description:** Allows the user to switch between light and dark themes to enhance visual comfort and reduce battery consumption.
- **Precondition:** App is running on a device that supports theme switching.
- **Postcondition:** The app's appearance changes based on the selected mode.
- **Event Flow:**
    1. User taps the theme toggle button on the toolbar or Saved News Fragment.
    2. System switches the app theme between light and dark mode.
    3. The new theme is applied across all UI components instantly.
    4. The selection is saved for future sessions.

Non Functional Requirements

Below are the non-functional requirements that we have deemed necessary for the application for now. Any future improvements might be added as per demands.

1. Performance Requirements[DONE]

- Under normal network conditions, loading of the home page takes 2 to 3 seconds.
- For simple queries, search results are returned to the user between 3-4 seconds.

3. Usability Requirements[DONE]

- The app supports speech recognition for voice search.
- TalkBack support is provided for users with visual impairments.
- Every feature is reachable within three taps or less.

4. Technical Stack Requirements[DONE]

- Frontend: XML + Kotlin is used for now but in future Jetpack Compose could also be used.
- Backend: Retrofit for APIs, RoomDatabase for data storage.

5. Compatibility Requirements[DONE]

- App supports Android 8 (API 26) onwards.
- Functions in emulators and physical devices alike.

6. Reliability Requirements[DONE]

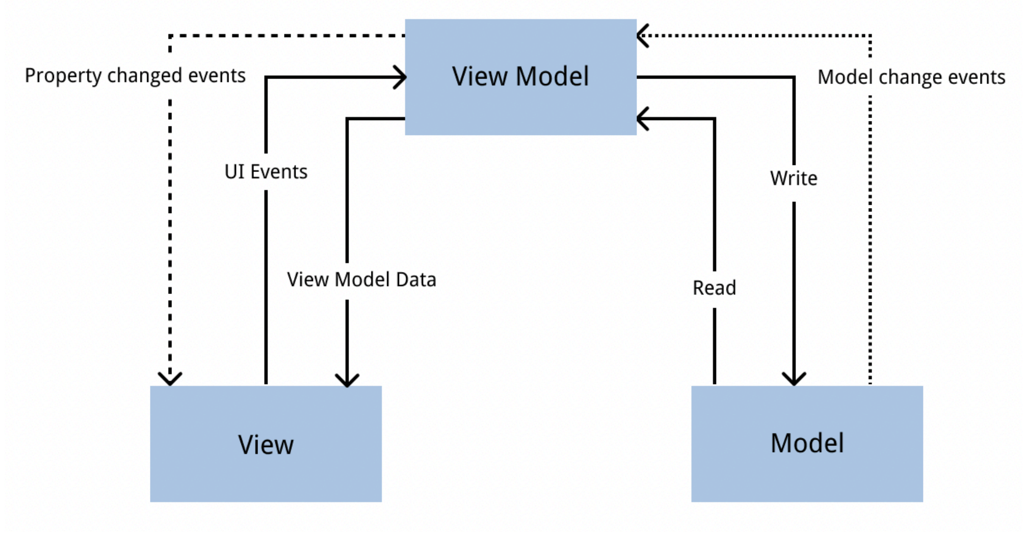
- 99.9% uptime for core features.

9. Maintainability and Extensibility[DONE]

- Frontend MVVM architecture setup.
- Highly modular coded structure for easy addition of features.

Architecture

We have followed the **MVVM (Model-View-ViewModel)** architecture.



1. Model (Data Layer)

- Responsible for fetching and storing data.
- Fetches news articles from APIs or websites.
- Caches data for offline access.
- Handles network errors and exceptions.

Components:

- **Repository**: Manages data sources (API, database, cache).
- **Retrofit/Volley**: To fetch news articles.
- **Room Database/SQLite**: For local storage (offline access).
- **WorkManager**: For background tasks like news updates.

2. View (UI Layer)

- Displays news articles to users.
- Sends user interactions to the ViewModel.

Components:

- **Activities/Fragments**: UI screens for displaying news.
- **RecyclerView**: For listing news articles.
- **Navigation Component**: Handles screen transitions.

3. ViewModel (Logic Layer)

- Acts as a **bridge** between the View and Model.
- Stores UI-related data.
- Handles business logic (e.g., fetching data from API and passing it to UI).
- Prevents **memory leaks** by surviving configuration changes.

Components:

- **LiveData**: Observes and updates UI.
- **Coroutines**: For handling asynchronous tasks.