Mobile Application Capstone Project

Illia Rohalskyi

Matriculation Number: k64968

Würzburg, Germany

Contents

Chapter 1

Unveiling the News: Introduction

1.1 Contextualizing the Digital News Realm

The ever-growing volume of information available online can be overwhelming, especially when it comes to staying informed on current events. Traditional news sources like newspapers and television broadcasts often have limitations in terms of accessibility, real-time updates, and personalization. This project addresses this need by developing a mobile news application.

This application aims to provide users with a convenient and user-friendly platform to access a variety of news articles from various sources. Unlike traditional media, mobile news apps offer several advantages:

- **Accessibility:** Users can access news articles anytime, anywhere, with just a few taps on their smartphones.
- **Real-time Updates:** Mobile news apps can deliver news and updates as they happen.
- **Personalization:** Many news apps allow users to customize their news feed based on their interests, ensuring they see the news stories that matter most to them.
- **Variety:** Mobile news apps can aggregate content from a wide range of sources, providing users with a diverse perspective on current events.

This project focuses on developing a mobile news application that leverages these advantages to create a user-friendly and informative experience for news consumers.

In conclusion, this project aims to contribute to the growing field of mobile news applications by offering users a convenient and personalized platform to stay informed on the latest news and events.

1.2 Personal Perspectives: The Drive Behind the App

As a Ukrainian citizen myself, the ongoing war has significantly impacted my daily life. Staying informed about current events has become an essential need for myself and countless others within Ukraine and around the world. Traditional news sources can often be limited in terms of accessibility, real-time updates, and the ability to curate content based on specific interests.

This personal experience motivated me to develop a mobile news application that addresses these limitations and empowers users to stay informed on critical developments during this challenging time.

In essence, this project reflects a desire to contribute to a more informed and empowered global citizenry, particularly in the face of such significant events.

Chapter 2

Architectural Blueprint: Building a Foundation for News Consumption

2.1 Functional Cornerstones - Engineering User Journeys

The developed mobile news application adheres to the following technical requirements:

- **Navigation:** The application provides a menu system for user navigation, implemented as either a drawer menu on the first screen.
- **News List:** The first screen features a scrollable list view for displaying news articles.
- **Data Loading:** The application utilizes infinite scrolling to load additional news articles in pages, enhancing user experience.
- **Data Source:** News data is retrieved from an API as a JSON file
- **Images:** Images associated with news articles are loaded from external URLs.
- **Custom List Tiles:** News articles are presented in custom list tiles, each containing text elements and potentially including images.
- **Detailed View:** A dedicated second screen displays detailed information about a selected news article.
- **User Input:** The application includes a search bar as a text input field

- **Theming:** The application offers a user-selectable theme, allowing users to switch between light and dark modes for a personalized experience.
- **BLoC Pattern:** The application leverages the BLoC (Business Logic Component) pattern for data management and communication between UI components.
- **Shared Preferences:** User preferences, such as the selected theme, are stored using the Android Shared Preferences mechanism.
- **Basic Design:** While in-depth design aspects were not covered in the course lectures, the application incorporates a basic design with colors and icons to enhance user experience.

These technical requirements ensure a well-structured, user-friendly, and informative mobile news application.

2.2 Beyond Functionality: Unveiling Non-Functional Nuances

In addition to the technical functionalities, the developed mobile news application prioritizes the following non-functional requirements:

- **Accessibility:** The application strives to be accessible to a broad user base. This includes a user-friendly interface that caters to users with varying levels of technical expertise.
- **Performance:** The application emphasizes efficient data retrieval and rendering to ensure a smooth and responsive user experience, especially on devices with varying processing power.
- **Usability:** Intuitive navigation, clear information presentation, and a well-organized layout are crucial for an application focused on news consumption. Users should be able to find and understand news articles quickly and effortlessly.
- **Real-time Updates:** Considering the fast-paced nature of current events, the application strives to deliver news updates in a timely manner, keeping users informed of the latest developments.
- **Personalization:** While a fully-fledged personalization system might be a future consideration, the application currently offers some basic user control over their news experience. Such as news language, category and theme selection.

These non-functional requirements ensure that the mobile news application is not just feature-rich but also user-centric, providing a valuable tool for staying informed in today's information age.

Chapter 3

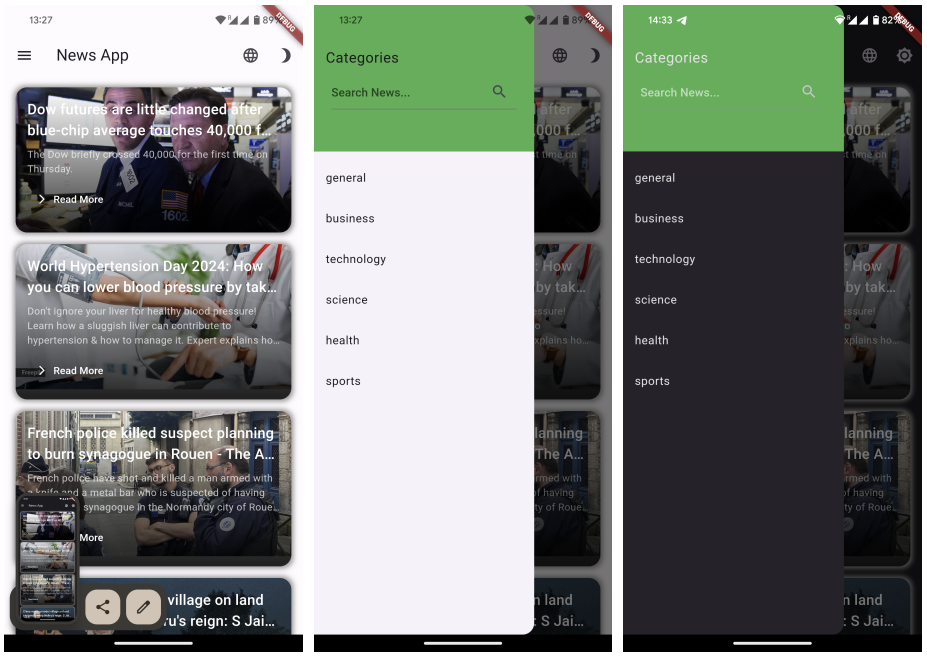
Insights into the App's Core

3.1 Navigating the Screen Flow: Enhancing User Experience

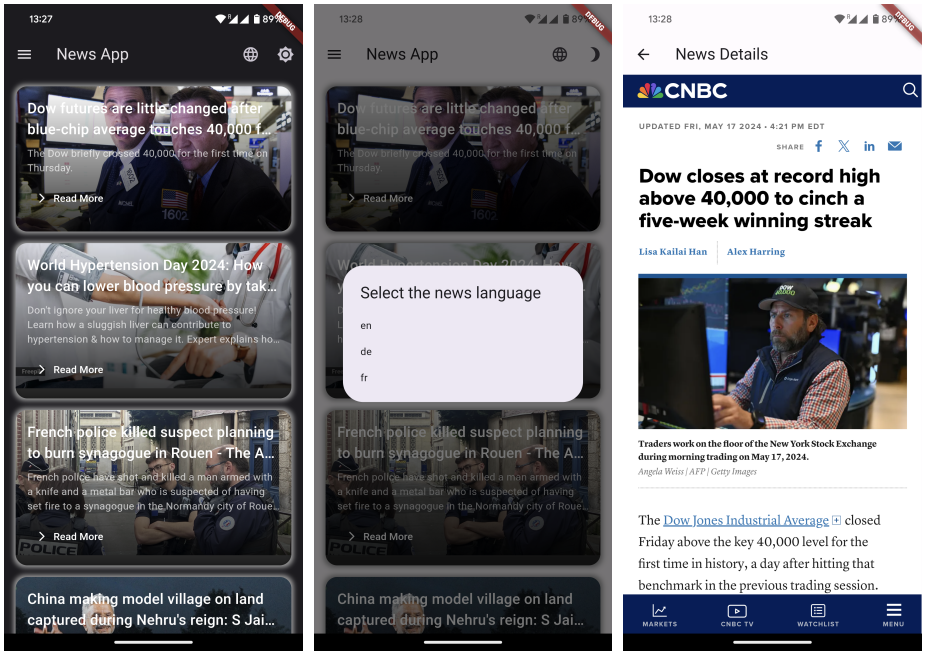
The screen flow is designed to be user-friendly and intuitive, facilitating easy navigation between different functionalities of the application. Below is a detailed description of the flow between various screens:

- **Main Screen:**
  - The main screen serves as the central hub of the application, providing access to various features.
  - A **drawer menu** is available, which can be accessed by swiping from the left edge of the screen or by tapping the menu icon.
  - Within the drawer, users can search for articles using the **search bar** or filter articles by specific **categories**. This allows for efficient content discovery and navigation.
  - A **light/dark mode icon** is located on the main screen, enabling users to toggle between light and dark themes. This feature enhances the user experience by allowing customization of the app's appearance based on personal preference or environmental lighting conditions.
  - A **language icon** is also present on the main screen. Clicking this icon opens a dialog box that allows users to select their preferred news language. The available language options are German, English, and French, catering to a diverse user base.
- **Webview Screen:**
  - When a user clicks on a news article, they are directed to the webview screen. This screen is designed to display the full content of the selected article within the app.

– The webview screen ensures that users can read articles seamlessly without having to leave the application, providing a more integrated and continuous reading experience.
– Navigation controls such as back, forward, and refresh buttons are available to enhance the user's control over the browsing experience.



(a) Main Screen (b) Drawer (c) Dark Drawer



(d) Dark Main Screen (e) Language dialog (f) Webview Screen

Figure 3.1: Screenshots of the application illustrating the screen flow.

3.2 Technological Foundations: Building Blocks of the App

We utilized the following technologies for developing the solution:

- **Flutter:** We chose Flutter as the primary framework for developing the mobile application due to its cross-platform nature and ability to create beautiful, native-like user interfaces.
- **HTTP Package:** We integrated the 'http' package to handle HTTP requests for fetching data from external APIs and services.
- **Webview Flutter:** The 'webview_flutter' package was utilized to embed web views within the application, allowing users to view web content seamlessly.
- **Shared Preferences:** We utilized the 'shared_preferences' package for storing and retrieving simple key-value pairs locally on the device, such as user preferences and settings.
- **RxDart:** RxDart was used for reactive programming, enabling efficient data streams management and handling asynchronous operations.
- **Flutter Bloc:** We implemented the BLoC (Business Logic Component) architecture pattern using the 'flutter_bloc' package for managing the application's state and business logic in a clean and modular way.
- **Lazy Load Scrollview:** The 'lazy_load_scrollview' package was used to implement lazy loading of content in scrollable views, improving performance by loading data only when needed.
- **Provider:** We utilized the 'provider' package for state management, enabling components to efficiently share and consume application state.
- **Cupertino Icons:** The 'cupertino_icons' package provided iOS-style icons for use within the application.

Additionally, we specified the project dependencies and configurations in the 'pubspec.yaml' file to manage package dependencies and project settings effectively.

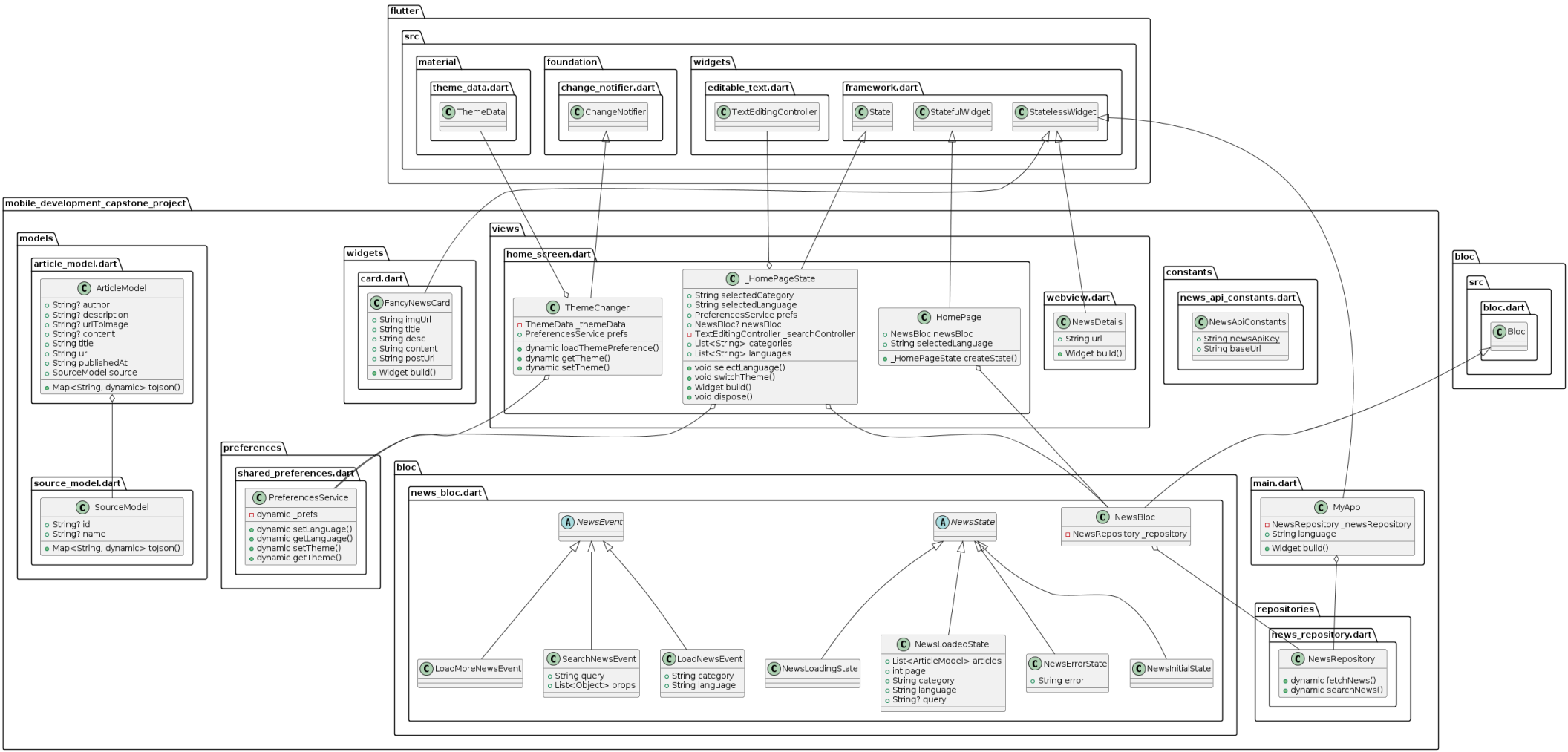
3.3 Exploring Key Components

The mobile development capstone project consists of several key components:

- **MyApp**: Main application widget responsible for building the user interface and managing the news repository.

- **ArticleModel**: Represents individual news articles with attributes such as author, description, image URL, content, and source.
- **SourceModel**: Represents the source of a news article, containing attributes like ID and name.
- **FancyNewsCard**: A custom widget designed to display news articles in an aesthetically pleasing format.
- **ThemeChanger**: Manages application themes and preferences, allowing users to switch between light and dark modes.
- **HomePage**: The main screen of the application, presenting a list of news articles along with options to select categories, languages, and themes.
- **_HomePageState**: Represents the state of the home screen widget, managing selected category, language, and search functionality.
- **NewsDetails**: Displays detailed content of a news article in a webview.
- **NewsBloc**: Handles state management for news-related operations such as loading news, searching news, and error handling.
- **NewsEvent**: Represents events dispatched to the NewsBloc to trigger specific actions.
- **NewsState**: Represents the state of news-related operations, including initial, loading, loaded, or error states.
- **NewsRepository**: Responsible for fetching and manipulating data related to news articles.
- **NewsApiConstants**: Contains constants such as the news API key and base URL.
- **PreferencesService**: Manages application preferences such as language and theme using shared preferences.

Here's the UML diagram illustrating the relationships between these components:


flutter
src
material
theme_data.dart
ThemeData
foundation
change_notifier.dart
ChangeNotifier
widgets
editable_text.dart
TextEditingController
framework.dart
State
StatefulWidget
StatelessWidget
mobile_development_capstone_project
models
article_model.dart
ArticleModel
String? author
String? description
String? urlToImage
String? content
String title
String url
String publishedAt
SourceModel source
Map<String, dynamic> toJson()
source_model.dart
SourceModel
String? id
String? name
Map<String, dynamic> toJson()
widgets
card.dart
FancyNewsCard
String imgUrl
String title
String desc
String content
String postUrl
Widget build()
views
home_screen.dart
ThemeChanger
ThemeData _themeData
PreferencesService prefs
dynamic loadThemePreference()
dynamic getTheme()
dynamic setTheme()
_HomePageState
String selectedCategory
String selectedLanguage
PreferencesService prefs
NewsBloc? newsBloc
TextEditingController _searchController
List<String> categories
List<String> languages
void selectLanguage()
void switchTheme()
Widget build()
void dispose()
HomePage
NewsBloc newsBloc
String selectedLanguage
_HomePageState createState()
webview.dart
NewsDetails
String url
Widget build()
constants
news_api_constants.dart
NewsApiConstants
String newsApiKey
String baseUrl
bloc
src
bloc.dart
Bloc
preferences
shared_preferences.dart
PreferencesService
dynamic _prefs
dynamic setLanguage()
dynamic getLanguage()
dynamic setTheme()
dynamic getTheme()
bloc
news_bloc.dart
NewsEvent
NewsState
NewsBloc
NewsRepository _repository
LoadMoreNewsEvent
SearchNewsEvent
String query
List<Object> props
LoadNewsEvent
String category
String language
NewsLoadingState
NewsLoadedState
List<ArticleModel> articles
int page
String category
String language
String? query
NewsErrorState
String error
NewsInitialState
main.dart
MyApp
NewsRepository _newsRepository
String language
Widget build()
repositories
news_repository.dart
NewsRepository
dynamic fetchNews()
dynamic searchNews()

3.4 Persisting Data Across Sessions

Data persistence in the mobile development capstone project involves storing news articles and user preferences using appropriate methods. Here's how it works:

- **News Articles:** News articles are typically fetched from an external source, such as a news API, and stored locally within the application. Depending on the requirements, articles may be stored in a database, local files, or in-memory cache for quick access. The chosen method should optimize performance and ensure reliable access to articles even when offline.
- **User Preferences:** User preferences, such as selected language, theme settings, and favorite categories, are stored locally to provide a personalized experience for users. This data can be stored using various techniques, such as shared preferences, SQLite databases, or local storage files. The chosen approach should prioritize security, data integrity, and efficient retrieval of user preferences.

3.5 Demystifying API Alchemy

The integration of the News API in the mobile development capstone project plays a crucial role in fetching news articles from external sources. Here's how it was implemented:

- **Selection of News API:** The project utilized the News API to access a wide range of news articles from various sources worldwide. The choice of the News API was based on factors such as reliability, coverage, ease of use, and available features.
- **API Key Authentication:** To access the News API, the project obtained an API key by registering with the provider. This API key was then included in API requests to authenticate and authorize access to the news data.
- **API Requests and Responses:** The project implemented HTTP requests to interact with the News API endpoints. These requests were made using libraries such as 'http' in Flutter, allowing the application to send queries and receive responses containing news articles in JSON format.
- **Data Parsing and Display:** Upon receiving news articles from the API, the project parsed the JSON responses to extract relevant information, such as article titles, descriptions, authors, publication dates, and article URLs. This data was then displayed to users within the application's interface.

Chapter 4

The Evolving News Landscape: A Look Ahead

4.1 Insights and Reflections

The mobile development capstone project aimed to create a versatile news application that provides users with access to the latest news articles from various sources. The app's key functionalities include:

- Seamless navigation through news categories and articles.
- Personalized experience with language selection and theme customization.
- Efficient fetching and display of news articles using the News API.
- Intuitive user interface with features such as search functionality and category selection.

Overall, the app successfully fulfills its purpose of delivering relevant and up-to-date news content to users in a convenient and accessible manner.

4.2 Charting the Course Ahead

While the current version of the app meets the basic requirements and functionalities, there is still room for improvement and additional features. Some potential areas for future work include:

- **Enhanced Personalization:** Implement more advanced user preference settings, such as personalized recommendations based on reading history and user behavior.
- **Improved Offline Support:** Enhance offline support by implementing caching mechanisms to store previously fetched articles and enable users to access them without an internet connection.

- **Social Integration:** Integrate social media sharing functionalities to allow users to share interesting articles with their friends and followers on various platforms.
- **Accessibility Features:** Implement accessibility features such as voice-assisted navigation and text-to-speech functionality to make the app more inclusive and user-friendly.
- **Performance Optimization:** Continuously optimize the app's performance by minimizing loading times, reducing resource consumption, and addressing any potential bottlenecks.

These future enhancements would further elevate the user experience and ensure that the app remains competitive in the dynamic landscape of news applications.