ALPHABETICA

By

ID	NAME
21222203031	Aktaruzzaman
21222203037	Siam Hossain
21222203005	Md. Najmul Parves
21222203017	Md. Abdullah Al Mamun
21222203030	Md. Naim Hoshain Pranto

Submitted in partial fulfilment of the requirements **CSE 400: Software Development IV**
For the degree of Bachelor of Science in Computer Science and Engineering

SUPERVISED BY

Md. Mahbubur Rahman

Assistant Professor, Department of CSE

Bangladesh University of Business and Technology

Date of Submission: January 01, 2025

Abstract

This project, titled **"Alphabetika"**, is a mobile application designed to assist users in learning and recognizing different alphabets, numbers, and symbols in various languages. The application is built for the Android platform using Java and SQLite for local data storage. The primary aim of this project is to provide an interactive, user-friendly interface that allows learners to explore different character sets such as Bengali, Arabic, and English alphabets, as well as numbers. The app supports multiple educational modes, including interactive games and learning activities to enhance the user experience.

The application is divided into several modules, each focusing on different character sets: **Bengali Alphabets (Shoroborno)**, **Bengali Numbers (BenjonBorno)**, **English Alphabet**, **Arabic Alphabets**, **Arabic Numerals**, and others. Each section includes visual representations and pronunciations of the respective characters to facilitate better learning. Additionally, there is a game module integrated into the app to encourage users to learn in a fun and engaging manner.

The app features a secure user authentication system, allowing users to create accounts and log in to personalize their learning experience. The user data, such as login credentials, is stored securely using SQLite, and the app supports a session management system to remember the user's login status across app launches. An additional feature includes a sign-out option, which clears the session and redirects the user to the login screen.

The main technology stack used for this project includes **Java** for the backend logic, **SQLite** for local data storage, and **XML** for designing the user interface. This combination ensures the app runs efficiently on Android devices, providing a seamless learning experience.

In conclusion, **Alphabetika** provides an educational tool for users to learn multiple alphabets and numerals in different languages. The project aims to simplify language learning with an interactive and gamified experience, promoting better retention and understanding of each character set.

List of Figures

Table of Contents

CHAPTER 1
INTRODUCTION

1.1 Problem Specification

In the modern age, the need for interactive and engaging educational tools has become more pronounced, particularly for young learners. While traditional methods of learning alphabets, numbers, and languages have their place, there is a growing demand for interactive applications that make learning both enjoyable and effective. The **Alphabetica** app was developed to address this need. It is designed to provide an engaging platform for children and learners to learn the alphabet in various languages, understand different characters, and practice writing, all while playing educational games.

The problem that motivated this project is the lack of interactive, multilingual, and game-based learning tools that cater to children. Often, apps focus on one language or one type of character, which can make the learning process monotonous and less engaging. **Alphabetica** aims to combine various language alphabets, including Bengali, Arabic, and English, with engaging gameplay to facilitate effective learning.

1.2 Objectives

The primary objectives of this project are:

1. To develop an interactive and multilingual learning app that includes alphabet learning for multiple languages (Bengali, Arabic, and English).
2. To integrate educational games that make the learning process enjoyable for children.
3. To allow users to practice writing alphabets, learning pronunciation, and identifying characters.
4. To design a user-friendly interface with clear, visually appealing elements that attract children's attention and enhance the learning experience.
5. To enable easy navigation, allowing children to explore different sections of the app seamlessly.
6. To ensure that the app supports multiple devices and screen sizes for broader accessibility.

1.3 Scope

The **Alphabetica** app's scope is focused on creating a fun and interactive learning environment for children that helps them learn the alphabet in various languages, including Bengali, Arabic, and English. The app will:

1. Provide features for alphabet learning in different languages, such as visual recognition and pronunciation.
2. Include various interactive elements like quizzes, games, and writing exercises that promote active learning.

3. Incorporate educational activities for recognizing letters, understanding their forms, and practicing writing them in a fun way.
4. Include sections for both beginners and intermediate learners with progressively challenging levels.

The scope does not include the development of a full-scale educational system or curriculum, but rather focuses on a mobile application that targets basic alphabet learning and early literacy.

1.4 Organization of Project Report

The report is structured as follows:

- **Chapter 1: Introduction** - This chapter provides an overview of the problem specification, objectives, and scope of the **Alphabetica** app. It highlights the motivation behind the project and outlines what the app aims to achieve.
- **Chapter 2: Literature Review** - This chapter reviews existing educational apps, technologies used in language learning, and the impact of game-based learning on early childhood education. It provides insight into what tools and approaches have been used in the field and how the **Alphabetica** app differs.
- **Chapter 3: Methodology** - This chapter describes the technical approach and methods used to develop the app, including platform selection, design decisions, and the integration of different educational features such as games and quizzes.
- **Chapter 4: System Design and Implementation** - This chapter details the design process, including wireframes, UI/UX design, and the implementation of the app's core features, including language support, game mechanics, and interactive elements.
- **Chapter 5: Testing and Evaluation** - This chapter discusses the testing process of the app, user feedback, and how the app's features were evaluated for effectiveness in achieving the learning objectives.
- **Chapter 6: Conclusion and Future Work** - This chapter summarizes the results, the success of the project in meeting its objectives, and suggests future improvements and enhancements to the **Alphabetica** app.

CHAPTER 2
BACKGROUND

2.1. Existing System Analysis

Before developing **Alphabetica**, several systems existed to help users learn alphabets, whether in a manual or digital format. These systems range from traditional methods like books and classroom learning to various digital applications, such as educational apps and games. Below are some common existing systems, along with their advantages and limitations:

1. Traditional Manual Systems (Books, Flashcards, etc.)

- **Pros:**
  - **Easy accessibility**: Physical materials like books and flashcards are widely available and do not require any internet connection.
  - **Tactile learning**: Students can engage with the materials physically, enhancing their memorization and learning.
- **Cons:**
  - **Lack of interactivity**: There is minimal engagement with the learner, which can lead to passive learning.
  - **Limited reach**: These methods may not be as effective for tech-savvy younger generations who are more inclined to use smartphones or tablets.
  - **Fixed content**: Learners cannot track their progress or receive personalized feedback.

2. Existing Educational Apps (ABC Apps, Language Learning Apps)

- **Pros:**
  - **Interactive learning**: Many apps include games, quizzes, and exercises that make the learning process more engaging and fun.
  - **Progress tracking**: Users can track their learning progress and review past lessons to reinforce their knowledge.
  - **Personalization**: Some apps offer features that cater to individual learning speeds and styles, improving the user experience.
- **Cons:**
  - **Complex interfaces**: Some apps are not user-friendly, particularly for young learners who may struggle with navigation.
  - **Limited content variety**: Some apps focus solely on the alphabet, without offering additional lessons or gamified elements for deeper learning.
  - **Paid features**: Many educational apps include in-app purchases, which may limit access to full content unless the user subscribes.

Analysis of Existing Systems in Relation to Alphabetica:

Alphabetica aims to address some of the drawbacks of existing systems while enhancing the benefits. Unlike traditional manual systems, Alphabetica provides an interactive, digital platform

accessible through mobile devices, ensuring that young users can learn at their own pace with engaging content.

Compared to other educational apps, **Alphabetica** stands out by offering a more user-friendly interface designed specifically for children. It combines elements of gamification with rich learning content, ensuring a balance between fun and educational value. Furthermore, unlike many general learning platforms, **Alphabetica** focuses solely on alphabet and language learning in an interactive manner, providing tailored lessons that grow with the child's progress.

In conclusion, while existing systems do offer valuable learning opportunities, **Alphabetica** is designed to overcome their limitations, particularly in terms of interactivity, content personalization, and user experience. By doing so, it aims to become a more effective tool for young learners who are just starting to engage with the world of alphabets and language.

2.2. Supporting Literatures

The development of **Alphabetica** draws from a variety of theoretical concepts, methodologies, and technological tools that are central to modern mobile app development and educational technology. Below, we discuss the key knowledge areas, tools, and techniques used in the project:

1. Theoretical Foundations

- **Cognitive Learning Theory**: One of the foundational theories applied in the development of **Alphabetica** is the **Cognitive Learning Theory**, which emphasizes the role of active engagement and interaction in the learning process. This theory suggests that learners actively construct their own understanding based on their experiences. By incorporating interactive features, such as quizzes, games, and visual aids, **Alphabetica** aligns with the cognitive learning principles, making learning more engaging and effective for children.
- **Constructivism**: This theory, popularized by educational psychologists like Piaget and Vygotsky, focuses on how learners build on their prior knowledge. **Alphabetica** applies constructivist principles by offering personalized learning experiences. The app adapts to the learner's progress, reinforcing concepts and gradually introducing new ones, ensuring that the user's knowledge builds incrementally.
- **Gamification in Education**: The concept of **gamification**—using game-like elements in non-game contexts—has been widely studied and applied in educational settings. Gamification encourages engagement, motivation, and active participation. In **Alphabetica**, elements like points, badges, and interactive challenges are used to motivate children to learn the alphabet while having fun.

2. Mathematical Foundations

- **Algorithms for User Personalization**: To create a personalized learning experience, **Alphabetica** uses algorithms to assess a user's learning progress and adapt the content accordingly. These algorithms track the user's performance in quizzes and activities, adjusting the difficulty of the subsequent lessons to match the learner's ability. This method ensures that the app remains challenging yet manageable for the user.
- **Data Structures for Efficient Storage**: In terms of data storage, the app uses efficient data structures like arrays and hash maps to store user data, progress, and app settings. This

helps in quick retrieval of user-specific information, enhancing the app's performance and ensuring smooth user experience.

3. Methodological Approaches

- **Agile Development Methodology**: The **Agile Development** methodology was applied during the development of **Alphabetica**. Agile focuses on iterative development, collaboration, and flexibility. This approach allowed the development team to work in small, manageable sprints, deliver incremental improvements, and adapt to feedback from testers and potential users. By using Agile, we ensured that **Alphabetica** evolved continuously based on real-time user input and testing.
- **User-Centered Design (UCD)**: Given that **Alphabetica** is intended for children, **User-Centered Design** principles were essential in ensuring that the app is intuitive, visually appealing, and age-appropriate. UCD focuses on designing products by prioritizing the needs, behaviors, and preferences of the target audience. This approach was critical in creating an interface that is both engaging and easy for young users to navigate.

4. **Technological Knowledge and Tools**

- **Android Development (Java)**: The primary programming languages used to develop **Alphabetica** are **Java**, which are the most common languages for Android app development. **Java** is known for its reliability. These languages were chosen to ensure compatibility, scalability, and ease of development for mobile platforms.
- **Android Studio**: **Android Studio** was the main integrated development environment (IDE) used for the development of **Alphabetica**. Android Studio offers tools for UI design, debugging, testing, and performance monitoring, making it an ideal platform for creating a high-quality Android app.
- **SQLite**: For local storage and offline functionality, **SQLite** was implemented to store user data on the device. It allows for efficient management of small to medium-sized data sets and ensures that users can continue their learning even without internet connectivity.
- **XML for UI Layouts**: **XML** was used for designing the user interface of **Alphabetica**. XML allows for the separation of the UI design and logic, making it easier to manage and update the app's interface. This separation also improves code maintainability and scalability.
- **Multimedia Integration (Images, Audio, and Animations)**: Since **Alphabetica** is a learning app for children, integrating multimedia elements such as images, audio, and animations was crucial. These elements not only make the app more engaging but also help reinforce learning. For example, audio pronunciations of letters and visual animations of characters were used to enhance the alphabet learning process.

5. Why These Tools and Techniques?

- The chosen tools and methodologies were selected to create an interactive, engaging, and effective learning platform for children. The use of **Android Studio** and **Java** ensured that the app is accessible to a wide audience, given the popularity of Android devices. **Firebase** and **SQLite** were selected to manage user data efficiently, while **XML** and multimedia integration were crucial in creating a visually appealing and educational experience.

- The educational theories applied, such as **Cognitive Learning Theory** and **Constructivism**, align well with the app's goal of fostering active, personalized learning. By using **Gamification**, the app encourages children to stay motivated and engaged, making the learning process enjoyable.

In conclusion, the combination of these theoretical, mathematical, methodological, and technological principles and tools ensured that **Alphabetica** would be an effective, user-friendly educational app that meets the needs of its target audience—young learners eager to master the alphabet.

CHAPTER 3
SYSTEM ANALYSIS & DESIGN

3.1. Technology & Tools:

For the development of the Alphabetica app, a combination of tools, technologies, and software was utilized to build and run the app efficiently. Below is a brief overview of the main tools and technologies used in this project:

Software Requirements:

- **Operating System**: Ubuntu 24.04 – The project development was carried out using the Ubuntu 24.04 OS, which is a stable and developer-friendly Linux distribution.
- **Android Studio** (Version: 2024.1.1) – The primary IDE for developing the Android application. It provides various tools for app development, debugging, and testing.
- **Java** (Version: 20) – The programming language used for Android development, chosen for its reliability and compatibility with Android Studio.
- **JDK(OpenJDK):** Version: 20.0.2 (Corretto) – Java Development Kit for building the app.
- **SQLite** (Database) – For local data storage in the app, SQLite was used to manage user-related data, like login credentials and preferences.
- **XML** – For defining the layout of the app, XML was used to design the user interface in Android Studio.

Hardware Requirements:

- **RAM**: 16GB – The project development was carried out on a system with 16GB RAM to ensure smooth performance of Android Studio and other related tools.
- **Processor**: Intel Core i7 (10th Generation) – The system's powerful processor ensured fast execution of development tasks and a seamless development experience.

Other Tools:

- **GitHub** – For version control and collaborative work, GitHub was used to manage the project's source code and maintain its history.

3.2. Model & Diagram

3.2.1. Model (Agile Development Methodology)

For the development of **Alphabetica**, we adopted the **Agile Development Methodology**. This model emphasizes iterative progress, collaboration, and responsiveness to change, making it highly effective for software projects like Alphabetica, which required constant refinement and user feedback integration.

Why Agile?

- **Flexibility:** Agile allowed us to adapt to changing requirements and incorporate feedback efficiently.

- **User-Centric Development:** By involving users during every sprint, we ensured that the app met their expectations and usability standards.
- **Incremental Delivery:** Agile's sprint-based structure enabled us to deliver smaller functional modules, test them, and build upon them iteratively.
- **Team Collaboration:** Regular meetings (stand-ups) ensured the team stayed aligned on progress and goals.

Efficiency and Benefits for the Project:

- **Time Efficiency:** Allowed us to prioritize core features and deliver a functional MVP early in the development cycle.
- **Enhanced Quality:** Continuous testing at the end of each sprint helped identify and address issues promptly.
- **Scalability:** Agile supported easy incorporation of additional features like multilingual support, user session management, and game elements without disrupting the workflow.

Workflow:

1. **Requirement Gathering and Analysis:** Divided the project into smaller tasks such as the login system, language lessons, games, and dashboard.
2. **Planning Sprints:** Each sprint focused on specific features, such as implementing the database for user management, creating individual learning modules, and designing the UI.
3. **Design and Prototyping:** Developed low and high-fidelity designs for user feedback.
4. **Development:** Followed by coding each module in Java and Android Studio.
5. **Testing:** Regularly performed unit testing and UI/UX evaluations.
6. **Delivery:** Released incremental updates during each sprint.

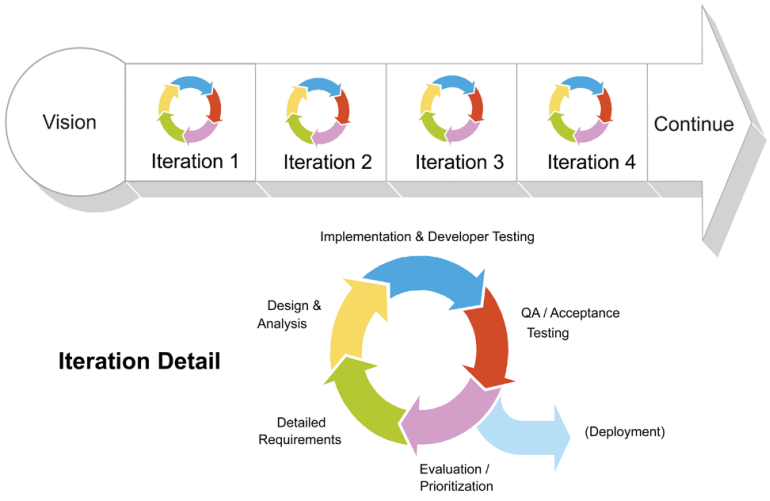

Figure 3.2: Agile Methodology

3.2.2. Use Case Diagram

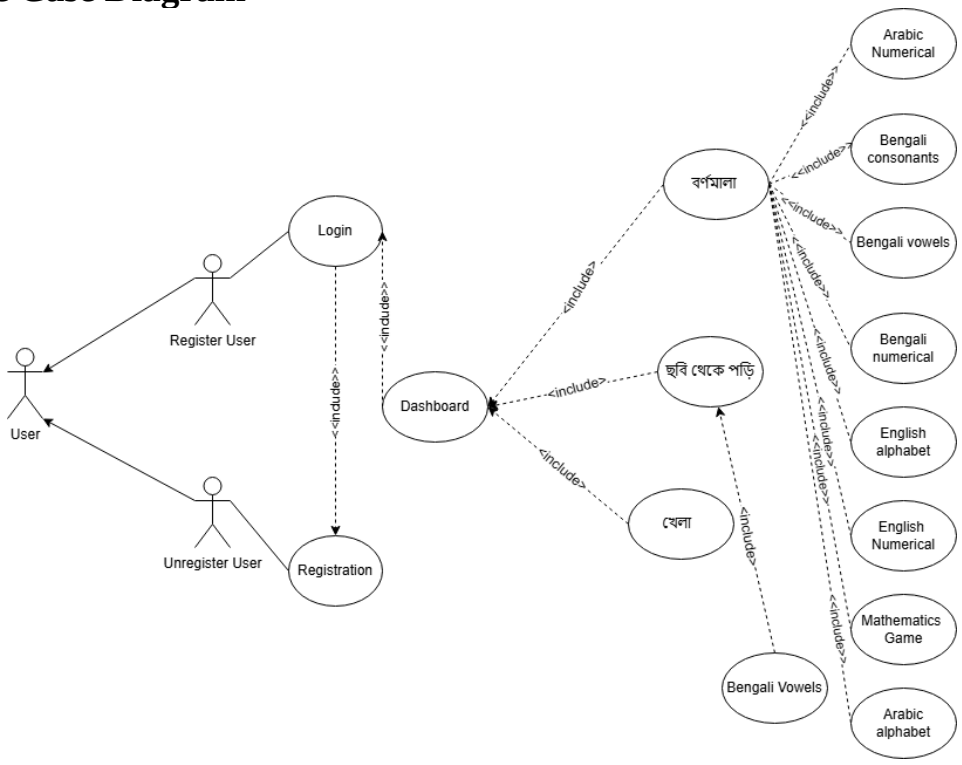

Figure 3.2: Use case Diagram

Actors:

1. User: Represents the individual using the Alphabetica app. This could be a student or anyone interested in learning alphabets and numerals in different languages.

Use Cases:

1. Login: Allows the user to log in with their credentials to access personalized features.
2. Registration: Enables new users to create an account in the app.
3. Dashboard: Serves as the central hub where users can navigate to different learning modules.

Learning Modules:

From the Dashboard, the user can access the following modules:

1. Bengali Vowels: Learn Bengali vowel sounds and characters.
2. Bengali Consonants: Learn Bengali consonant sounds and characters.
3. Bengali Numerical: Explore Bengali numerals.
4. English Alphabet: Learn the English alphabet.
5. English Numerical: Learn English numerals.
6. Arabic Alphabet: Explore Arabic letters.
7. Arabic Numerical: Learn Arabic numerals.
8. Mathematics Game: Engage in educational games focused on basic math concepts.

Relationships:

- The User interacts with the Login and Registration use cases initially.
- After logging in, the User is directed to the Dashboard, where they can choose any of the available modules.
- Each module represents a learning area tailored for specific alphabets, numerals, or games.

3.2.3. Level 0 DFD

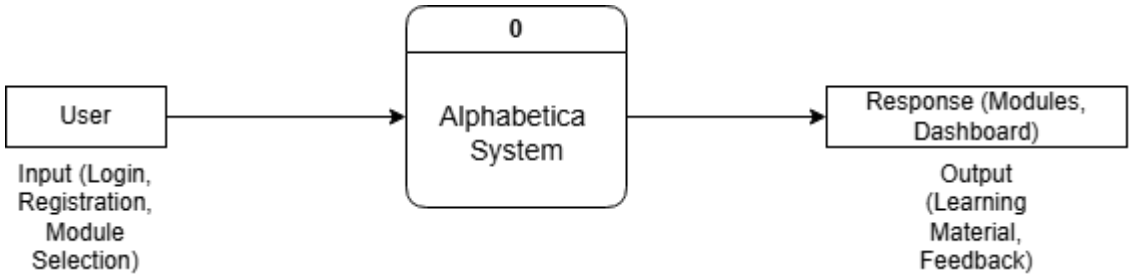

Figure 3.3: Level 0 DFD

1. **External Entity: User**
    - **Input:**
        - The user interacts with the system by providing:
            - Login or registration information.
            - Selection of learning modules (e.g., Bengali vowels, English alphabet, Math games).
    - **Output:**
        - The user receives responses from the system, such as:
            - Access to the dashboard.
            - Learning content for the selected module.
            - Feedback or results from activities (like math games).
2. **Process: Alphabetica System (labeled as 0)**
    - Represents the entire app as a single system.
    - Handles all interactions with the user, processes their inputs, and generates appropriate outputs.
    - Centralized point of data flow between the user and the system's internal functionality.
3. **Data Flow:**
    - **Input:** The user sends inputs like login details, module selections, or game interactions.
    - **Output:** The system responds with learning materials, dashboards, or results.
4. **Output Entity: Response (Modules, Dashboard)**
    - This represents the data or information sent back to the user. It includes:
        - Navigation to the dashboard.
        - Educational content, alphabets, or numerals.
        - Game results or feedback.

.

3.2.4. Level 1 DFD

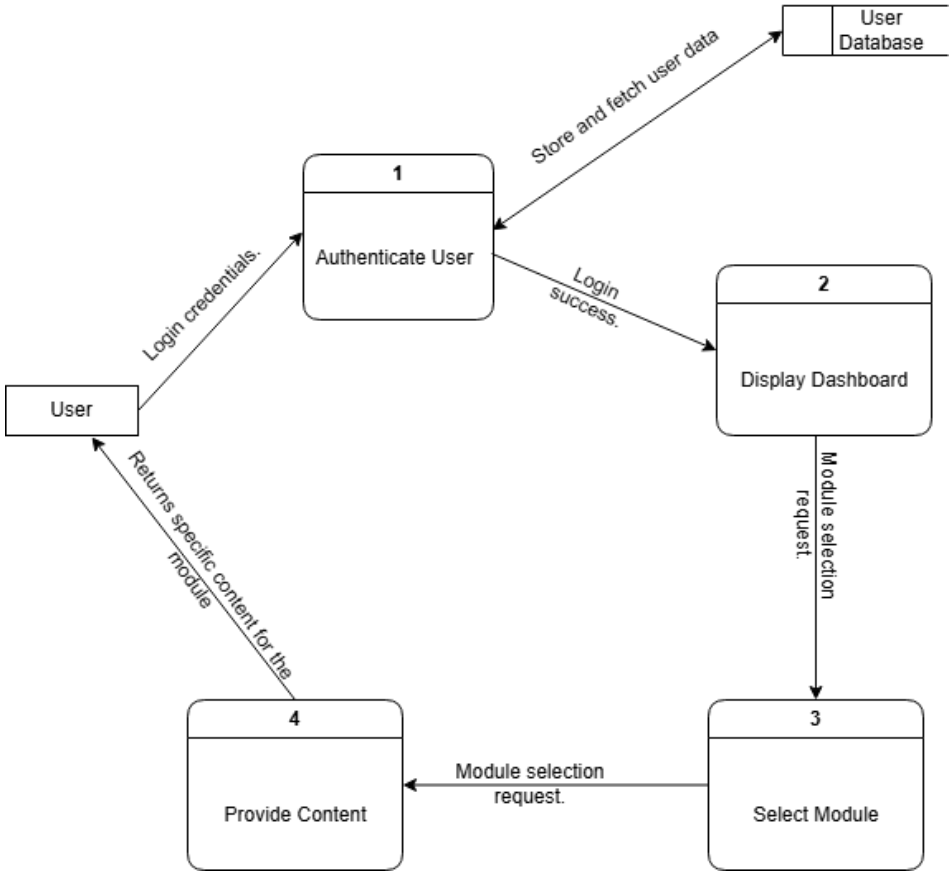

Figure 3.4: Level 1 DFD

The **Data Flow Diagram (DFD) Level-1** for the Alphabetica project represents how data flows between the external entities, processes, and data stores within the system. It highlights the core sub-processes and their interactions with data sources.

External Entity: User

The **User** interacts with the system by providing:

1. **Login Details** to authenticate and access the system.
2. **Module Selection** to begin the learning process.

Sub-Processes and Data Flow

1. **Authenticate User**:
    - **Inputs**: Login credentials (username and password) provided by the user.
    - **Outputs**: Authentication status (success or failure).
    - **Description**: Validates user credentials by querying the User Database and provides a result back to the user.
    - **Data Flow**:
        - User → Authenticate User → User Database → Authenticate User → User (Authentication status).

2. **Display Dashboard**:
    - **Inputs**: Successful authentication status.
    - **Outputs**: Dashboard options displayed to the user (e.g., list of modules or settings).
    - **Description**: Upon successful login, the system fetches dashboard options for the user.
    - **Data Flow**:
        - Authenticate User → Display Dashboard → User (Dashboard options).
3. **Select Module**:
    - **Inputs**: User-selected module.
    - **Outputs**: Module request sent to the "Provide Content" process.
    - **Description**: Captures the module selection input from the user and sends it for content retrieval.
    - **Data Flow**:
        - User → Select Module → Module Request.
4. **Provide Content**:
    - **Inputs**: Selected module request.
    - **Outputs**: Corresponding module content displayed to the user.
    - **Description**: Fetches the relevant content for the selected module from the Module Data Store and displays it to the user.
    - **Data Flow**:
        - Select Module → Provide Content → Module Data Store → Provide Content → User (Module content).
5. **Track Progress** (Optional):
    - **Inputs**: User activity (e.g., completed lessons or tasks).
    - **Outputs**: Updated progress saved in the User Database.
    - **Description**: Monitors user progress and stores it for future reference.
    - **Data Flow**:
        - User → Track Progress → User Database.

Data Stores

1. **User Database**:
    - Contains user credentials and progress data.
    - Interacts with the "Authenticate User" and "Track Progress" processes.
2. **Module Data Store**:
    - Stores content for all modules, user preferences, and related data.
    - Interacts with the "Provide Content" process to deliver relevant module data.

3.2.4. Database Schema

Column Name	Data Type	Constraints	Description
id	INTEGER	PRIMARY KEY, AUTOINCREMENT	Unique identifier for each user.
fullName	TEXT	NOT NULL	Stores the user’s full name
email	TEXT	NOT NULL	Stores the user’s email
userName	TEXT	UNIQUE, NOT NULL	Stores the user's unique username.
password	TEXT	NOT NULL	Stores the user's password.

Alphabetica app only requires a single table for handling both login and registration functionalities.

The **User Table** will handle:

1. **User Registration** (storing full name, email, username, password).
2. **User Login** (verifying with username and password).

3.2.5. Flowchart

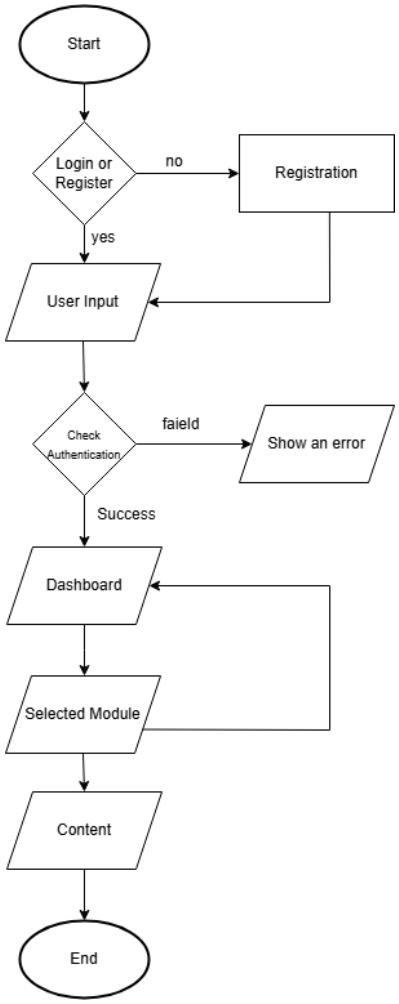

Figure 3.5: Flowchart

Flowchart for **Alphabetica** can help visualize the process flow of user interactions and backend processes like login, registration, and content retrieval.

1. **Start**
2. **Decision Point for Login or Registration**
    - Add a **diamond** shape for decision-making:
        - If the user wants to **Log In** or **Register**.
3. **Start with the User Input**
    - Use an **oval** shape for the "Start" process.
    - Add a **parallelogram** shape to represent the input of **Username** and **Password** during login or **Username, Password, and Confirm Password** during registration.
4. **Login Process**
    - If **Log In**, proceed to:
        - **Database Authentication**: Add a rectangle labeled "Authenticate User (Check Username & Password in Database)."
        - Add a decision:
            - If **success**, proceed to "Dashboard."
            - If **failure**, go to "Show Error Message."
5. **Registration Process**
    - If **Register**, proceed to:
        - Add a rectangle labeled "Validate Registration Details."
        - Add a decision:
            - If **Valid**, store user data into the **User Database.**
            - If **Invalid**, show "Registration Error Message."
6. **Content Selection**
    - Add a parallelogram to represent module selection by the user.
    - Add a rectangle labeled "Fetch Content Based on Selected Module."
    - Proceed to "Display Content to User."
7. **End**
    - Use an oval shape for the "End" of the process.

CHAPTER 4
IMPLEMENTATION

4.1. Interface/Front-End Design

The front-end design of Alphabetica was developed to provide an intuitive and user-friendly experience. The interface was built using **Android Studio**, leveraging XML for layout design. Key considerations in the front-end design were simplicity, ease of navigation, and responsiveness.

- **Login and Registration Forms**: The login and registration forms were designed with minimal fields to ensure simplicity and usability. The forms use standard UI components such as EditText for input fields and Button for actions.
- **Dashboard**: Upon successful login, the user is navigated to a clean and simple dashboard displaying available content modules. Each module is represented by a card or button for easy access.
- **Colors and Themes**: The application uses a consistent color palette to maintain a professional and user-friendly aesthetic.

4.2. Interface/Back-End Design

The back-end of Alphabetica was implemented using **Java**, focusing on functionality and seamless communication with the front-end.

- **Login and Registration Functionality**: User credentials are stored locally using **SharedPreferences**, allowing for lightweight and efficient data management without the need for external databases. The system verifies the entered credentials during login and provides feedback based on success or failure.
- **Data Validation**: Input fields such as username, password, and confirm password are validated to ensure correctness and prevent errors like empty fields or mismatched passwords during registration.
- **Content Display**: After successful login, the application dynamically fetches and displays content from predefined modules stored within the app.

4.3. Modules/Features

The following modules and features were implemented in Alphabetica:

1. **User Authentication**:
    - **Login**: Validates user credentials and grants access upon success.
    - **Registration**: Allows new users to create accounts by entering username, password, and confirm password.
    - **Error Feedback**: Provides error messages for incorrect login attempts or mismatched passwords.
2. **Dashboard**:
    - Displays available modules for content exploration.
    - Acts as the central hub for user navigation.
3. **Content Modules**:
    - Enables users to explore predefined educational or informational content.

4. **Local Data Storage**:
    - Uses **SharedPreferences** to store and manage user credentials.
    - Eliminates the need for external database connections, simplifying the architecture.
5. **Error Handling**:
    - Ensures smooth user experience by providing meaningful feedback during errors like invalid login or incomplete registration.

CHAPTER 5
USER MANUAL

5.1. System Requirements

5.1. Hardware Requirements

To run the **Alphabetica mobile app**, your device must meet the following minimum hardware specifications:

- **Processor**: Quad-core 1.4 GHz or higher.
- **RAM**: 2 GB (minimum), 4 GB (recommended).
- **Storage**: At least 500 MB of free disk space for the app and its data.
- **Display**: A screen resolution of 1280x720 pixels or higher.

5.2. Software Requirements

To ensure optimal performance, your device should meet the following software requirements:

- **Operating System**: Android 8.0 (Oreo) or higher.
- **Permissions**: The app requires access to the following features:
    - Storage: To save and load app-related files.
    - Internet: For accessing online features (if applicable).
    - Sound: For audio playback features.

5.2. User Interfaces

This section outlines the key interfaces of the project, describing the functionality of each panel, its flow, and behavior. Screenshots or diagrams should be included for better understanding, alongside credentials where applicable.

5.2.1. Login Panel

The **Login Panel** is the entry point for the application, where users input their credentials to access the app's features.

- **Inputs**:
    - **Username**: A unique identifier for the user.
    - **Password**: A secure key associated with the username.
    - **Confirm Password**: Ensures the user has entered the intended password during registration.
- **Process**:
    1. Use enters login credentials
    2. If validated: Navigates to the **Dashboard**.
    3. If invalid: Displays an error message.
- **Outputs**:
    - Success: Navigates to the **Dashboard**.
    - Failure: Displays a login error.

- **Features**:
    - User authentication and error handling.

5.2.2. Dashboard Panel

The **Dashboard Panel** acts as a navigation hub, allowing users to select a specific module, such as **Arabic Panel**, **English Panel**, or others.

- **Inputs**:
    - Module selection by the user (e.g., Arabic, English, etc.).
- **Process**:
    1. User clicks on a specific module.
    2. The app navigates to the selected module (e.g., Arabic, English).
- **Outputs**:
    - Success: Opens the corresponding panel/module.
- **Features**:
    - Dynamic navigation between modules.

For Example **English Panel** is a module, Focus on English alphabet.

- **Inputs**:
    - User clicks on an English alphabet or number.
- **Process**:
    - Plays an audio file for the selected alphabet/number.
    - Highlights the selected option.
- **Outputs**:
    - Audio playback of the selected option.
    - Visual feedback (highlight).

5.2.3. Back Navigation

The **Back Button** allows users to navigate back to the **Dashboard Panel** from any module.

- **Inputs**:
    - User clicks the back button.
- **Process**:
    1. Clears the current activity stack using flags (`FLAG_ACTIVITY_CLEAR_TOP` and `FLAG_ACTIVITY_NEW_TASK`).
    2. Navigates back to the **Dashboard Panel**.
- **Outputs**:
    - Smooth transition to the **Dashboard Panel**.

5.2.4. Persistent State Management

The application saves the state of the last selected item (in the **Arabic Panel** or other modules) using **SharedPreferences**.

- **Inputs**:
    - Selected item.
- **Process**:
    1. Save the selected item’s ID in SharedPreferences

2. Restore the selected item's state upon revisiting the module.

- **Outputs**:
  - Restored UI with previous user preferences.

5.2.5. Screenshot/Illustration:

Signup and Login page:

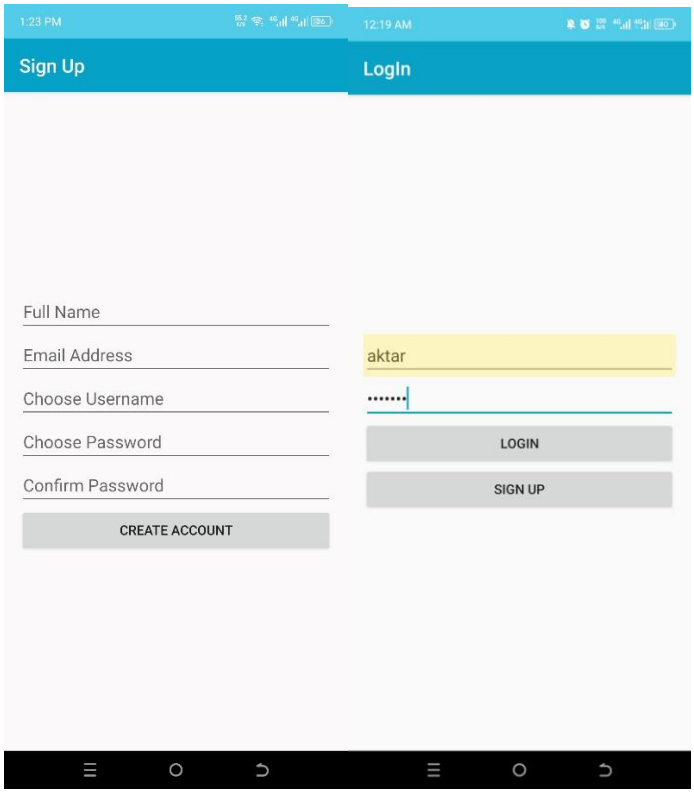

Figure 5.1: signup and login page

Dashboard and Profile:

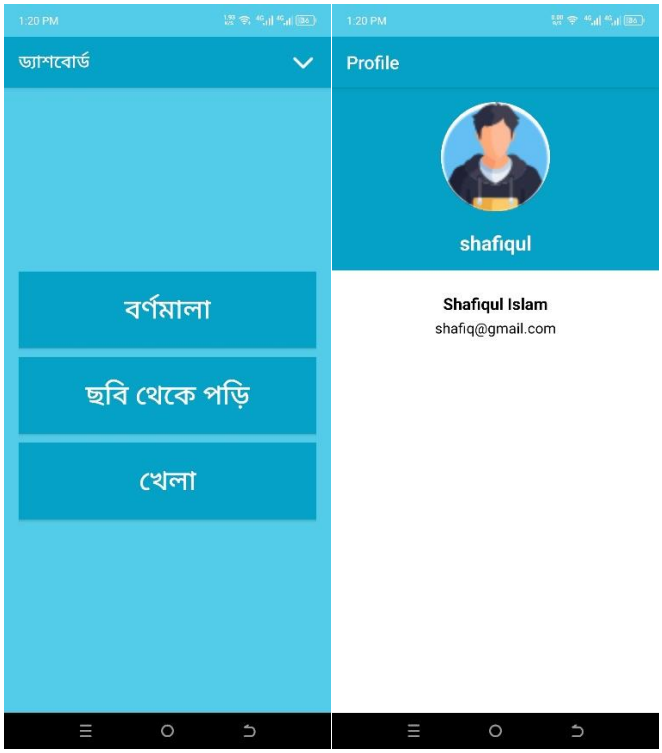

Figure 5.2: Dashboard and Profile

Arabic Alphabets before and after reopen App:

Figure 5.3: Arabic Alphabets page before and after reopen app

Read from Sentence page

Figure 5.4: Read From Sentence Page

Game and Game Score Page

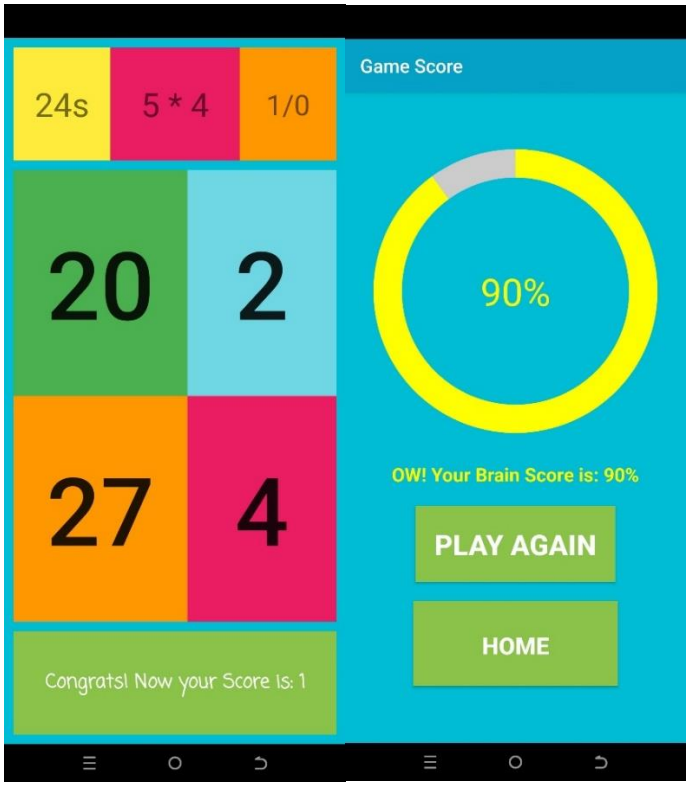

Figure 5.5: Game and game Score

CHAPTER 6
CONCLUSION

6.1. Conclusion

The project *Alphabetica* successfully achieves its goal of providing an interactive and educational application for users to learn alphabets in multiple languages. The implementation of persistent data storage through SharedPreferences, dynamic audio playback, and user-friendly navigation contributes to a seamless experience. Throughout this project, the team gained valuable insights into Android development, user interface design, and efficient resource management. This learning experience has been instrumental in understanding how to build scalable and interactive applications.

6.2. Limitations

1. The application is currently limited to a single user profile and does not support multiple users or accounts.
2. Audio files and layouts are hardcoded, making it less dynamic for future content updates.
3. The application lacks online synchronization, restricting user data to local storage.
4. There is limited accessibility support for differently-abled users, such as text-to-speech or larger font options.

6.3. Future Works

1. Implement support for multiple user profiles with personalized progress tracking.
2. Add more languages and customizable learning modes to expand the application's usability.
3. Integrate cloud storage for real-time data synchronization across devices.
4. Enhance accessibility features to make the app more inclusive.
5. Develop a gamified experience to engage users, such as quizzes or challenges for learning alphabets.
6. Add analytics to track user progress and improve the learning experience based on user behavior.

References

Here are the references used for the development and documentation of *Alphabetica*:

1. Android Developers. "SharedPreferences." *Android Documentation*, Google, developer.android.com/reference/android/content/SharedPreferences. Accessed 10 Dec. 2024.
2. Stack Overflow Community. "How to Save and Retrieve the Last Selected State in Android?" *Stack Overflow*, stackoverflow.com. Accessed 10 Dec. 2024.
3. JavaTpoint. "Android MediaPlayer Example." *JavaTpoint*, www.javatpoint.com/android-mediaplayer-example. Accessed 9 Dec. 2024.
4. Scribbr. "MLA Citation Generator." *Scribbr: Your MLA Citation Guide*, www.scribbr.com/citation/generator/. Accessed 12 Dec. 2024.
5. GeeksForGeeks. "How to Build a Multi-Language Android Application?" *GeeksForGeeks*, www.geeksforgeeks.org/how-to-build-a-multi-language-android-application/. Accessed 8 Dec. 2024.
6. Draw.io. *Diagrams for Process Flowcharts and DFDs*, diagrams.net. Accessed 7 Dec. 2024.
7. XML Layout Examples and Code Snippets. "Android UI Design Principles." *Medium: Mobile UI Design Tips*, medium.com. Accessed 6 Dec. 2024.

Appendix A

Alphabetica: Supplementary Information

A.1. Project Summary

Alphabetica is an educational Android application designed to help users learn alphabets from different languages interactively. The app allows users to listen to the pronunciation of letters by tapping on them and highlights the selected letter. It saves user preferences using SharedPreferences and retains the last selected state even after the app is reopened.

A.2. Technology Stack

Programming Language: Java

IDE: Android Studio
Database/Storage: SharedPreferences for local data storage
Media: MediaPlayer for audio playback
UI Design: XML for layouts, Material Design principles

A.3. Features and Functionalities

1. **Login and Registration**:
    - Secure user login and registration using SharedPreferences.
    - Fields: Username, Password, Confirm Password.
2. **Interactive Alphabet Modules**:
    - Multiple panels for different languages.
    - Each panel contains alphabets that play audio pronunciations when tapped.
3. **State Persistence**:
    - Highlights the last selected letter with a red color background.
    - Saves state using SharedPreferences so it is restored after reopening.
4. **Navigation**:
    - Back button to return to the home page.
    - Clears the activity stack for efficient navigation.

A.4. Key Implementation Details

- **Audio Map**: Maps TextView IDs to corresponding audio files for pronunciation.
- **Color Management**: Resets button colors dynamically to provide a clean and interactive user experience.
- **Back Button**: Uses Intent flags to navigate back to the home screen without restarting the stack.
- **MediaPlayer Management**: Releases MediaPlayer resources to avoid memory leaks.

A.5. Sample Code Snippets

*State Persistence Using **SharedPreferences**:*

```
SharedPreferences preferences = getSharedPreferences(PREFERENCES_NAME, MODE_PRIVATE);
SharedPreferences.Editor editor = preferences.edit();
editor.putInt(SELECTED_KEY, selectedId);
editor.apply();
```

*Audio Playback Using **MediaPlayer**:*

```
mediaPlayer = MediaPlayer.create(this, audioResId);
mediaPlayer.start();
mediaPlayer.setOnCompletionListener(mp -> {
    mp.release();
    mediaPlayer = null;
});
```

A.6. Testing Information

- Tested on Android 11 and above.
- Verified state persistence and audio playback across devices.
- Checked UI responsiveness and button functionality.

A.7. Acknowledgments

Special thanks to my supervisor, **Md. Mahbubur Rahman**, Assistant Professor, Department of CSE, Bangladesh University of Business and Technology, for his valuable guidance and support throughout the development of this project. I would also like to extend my gratitude to my peers and online communities for providing assistance, code examples, and troubleshooting tips, which greatly facilitated the progress of this project.