# Text File Translator

CSE 400: Software Development Project IV

SUBMITTED BY


| Name | ID | Intake |
|---|---|---|
| Al Jahan Akter Barsha | 21225103094 | 49 |
| Tarikul Islam Maruf | 21225103098 | 49 |
| Soumya Balsrum Ritchil | 21225103224 | 49 |
| Shah Md Hasibur Rahaman hasib | 21225103355 | 49 |
| Ariful Haque Rohan | 21225103510 | 49 |

Supervised By:

G. M Waliullah
Assistant Professor,
Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BANGLADESH UNIVERSITY OF BUSINESS AND TECHNOLOGY (BUBT)


16, MAY, 2025

# Abstract

This project details the development of a "Text File Translator," an application built using Flutter for the frontend and leveraging Java for any native Android integrations. It allows users to directly upload text files (.txt) from their mobile devices and translate their contents into a selected target language using the Google Translate API. The application features a user-friendly interface with support for dark mode to enhance user experience. Additionally, it maintains a local history of the last five translations, allowing users to quickly access recent work. The app is designed to be intuitive, efficient, and provide a seamless experience for on-the-go text file translation.

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Problem Specification/Statement

In an increasingly globalized world, the need for quick and accessible document translation is paramount. Many existing mobile translation tools require users to copy and paste text, which is inefficient for entire files. Furthermore, user experience aspects like readability in different lighting conditions (addressed by dark mode) and quick access to recent work are often overlooked. This project aims to develop a mobile application using Flutter and Java that allows users to directly upload text files (.txt), translate their content using the Google Translate API, view translations within the app, toggle dark mode, and access a history of their last five translations.

## 1.2 Objectives

The primary objectives of this project are:

- To develop a cross-platform mobile application using Flutter for translating text files.
- To enable users to upload `.txt` files directly from their mobile device storage.
- To integrate the Google Translate API for accurate language translation.
- To provide a user-selectable target language for translation.
- To implement a dark mode feature for enhanced user comfort and accessibility.
- To store and display a history of the last five translation activities locally on the device.
- To ensure a responsive and intuitive user interface for a seamless user experience.
- To allow users to view the translated text within the application.

## 1.3 Flow of the Project

The user journey within the Mobile Text File Translator app is as follows:

1. **Launch App:** The user opens the application.
2. **File Selection:** The user navigates to the file upload section and selects a `.txt` file from their device.
3. **Language Selection:** The user chooses the desired target language for translation from a provided list.
4. **Translation Process:** The app extracts text from the uploaded file. This text is then sent to the Google Translate API via an HTTP request, including the API key for authentication.
5. **Display Translation:** The Google Translate API returns the translated text. The app displays this translated text to the user.
6. **History Update:** Details of the translation (e.g., original filename (or snippet), target language, timestamp, translated text snippet) are saved locally, maintaining a list of the last five translations. If the history already contains five entries, the oldest one is removed.
7. **Dark Mode Toggle:** The user can toggle dark mode at any time through the app settings for a different visual theme.
8. **View History:** The user can access a dedicated screen to view their last five translations and potentially re-view the translated content.

## 1.4 Organization Of Project Report

This project report is organized into the following chapters:

- **Chapter 1: Introduction** - Outlines the project's problem statement, objectives, operational flow, and report structure.
- **Chapter 2: Background** - Discusses existing systems and relevant literature.
- **Chapter 3: System Analysis & Design** - Details the technologies, development model, and system diagrams.
- **Chapter 4: Implementation** - Describes the front-end design, application logic, and key modules.

- **Chapter 5: User Manual** - Provides system requirements and guidance on using the application's interfaces.
- **Chapter 6: Conclusion** - Summarizes the project, its limitations, and potential future enhancements.
- **References** - Lists the sources consulted during the project.

# Chapter 2

# Background

## 2.1 Existing System Analysis

Current language translation solutions vary widely, but often have limitations when it comes to mobile file-based translation:

- **Online Web Translators (e.g., Google Translate Web):** While powerful, they often require copy-pasting text from files, which is cumbersome on mobile devices for entire documents. Direct file upload is sometimes supported but might not be optimized for mobile workflows.
- **Generic Mobile Translation Apps:** Many apps focus on translating typed or spoken phrases. Few offer robust direct text file translation, and features like persistent translation history or UI themes like dark mode are not consistently available.
- **Dedicated Document Translator Apps:** Some apps specialize in document translation but may be paid, limited in file types, or lack the simplicity desired for quick `.txt` file translations.
- **API-based Solutions:** Developers can use translation APIs, but these require building a custom application wrapper, which is what this project aims to achieve for a specific mobile use case.

This project fills a gap by providing a dedicated mobile solution for `.txt` file translation with essential usability features like dark mode and accessible history.

## 2.2 Supporting Literatures

- **Flutter Documentation (`https://flutter.dev`):** Essential for understanding the Flutter framework, widget system, state management, and plugin ecosystem used for building the cross-platform mobile application.

- **Dart Language Documentation (https://dart.dev):** Provides the foundation for Flutter development, covering syntax, libraries, and asynchronous programming crucial for API calls.
- **Google Cloud Translation API Documentation:** Details the API endpoints, authentication methods (API Key), supported languages, and usage quotas for the translation service.
- **Material Design Guidelines (https://material.io):** Influences the UI/UX design of the application, including principles for dark theme implementation and component styling.
- **Effective Mobile App Design Principles:** General literature on creating user-friendly and intuitive mobile applications, focusing on navigation, readability, and performance.
- **SQLite Documentation (https://sqlite.org) / Shared Preferences (Flutter):** Relevant for understanding local data storage mechanisms on mobile devices, which will be used for storing translation history.

# Chapter 3

# System Analysis & Design

## 3.1 Technology & Tools

- **Frontend Framework:** Flutter (using Dart language)
- **Native Integration (if needed):** Java (for Android-specific functionalities via Flutter plugins or platform channels)
- **Translation Service:** Google Translate API (accessed via HTTP requests with an API Key)
- **Local Storage:** SQLite (via `sqflite` plugin) or SharedPreferences for storing translation history and user preferences (like dark mode state).
- **IDE:** Android Studio, VS Code
- **Version Control:** Git, GitHub
- **Design & Prototyping:** Figma, Lucidchart (for diagrams)
- **API Testing:** Postman (optional, for testing Google Translate API calls independently)

## 3.2 Model & Diagram

### 3.2.1 Development Model

The project will adopt the **Agile methodology**. This iterative approach allows for flexibility in development, frequent testing, and incremental feature delivery. Sprints will focus on specific functionalities like file handling, API integration, UI development for different screens, history management, and dark mode implementation. Regular feedback will be incorporated to refine the application.

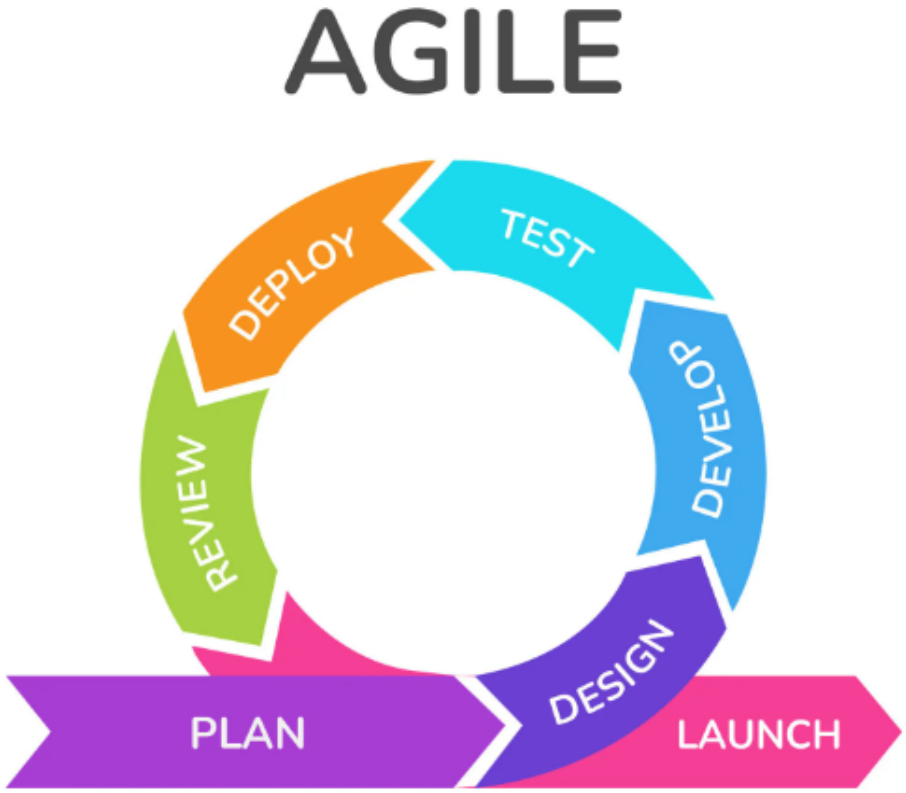


Figure 3.1: Agile Development Model.

### 3.2.2 Use Case Diagram

The Use Case Diagram illustrates the interactions between the user and the system's main functionalities. Key actors and use cases include:

Figure 3.2: Use Case Diagram for Mobile Text File Translator.

- **Actor:** User
- **Use Cases:**
  - Upload Text File
  - Select Target Language
  - Translate Text
  - View Translated Text
  - Toggle Dark Mode
  - View Translation History

### 3.2.3 High-Level System Architecture Diagram

This diagram illustrates the main components and data flow within the application. The archi-

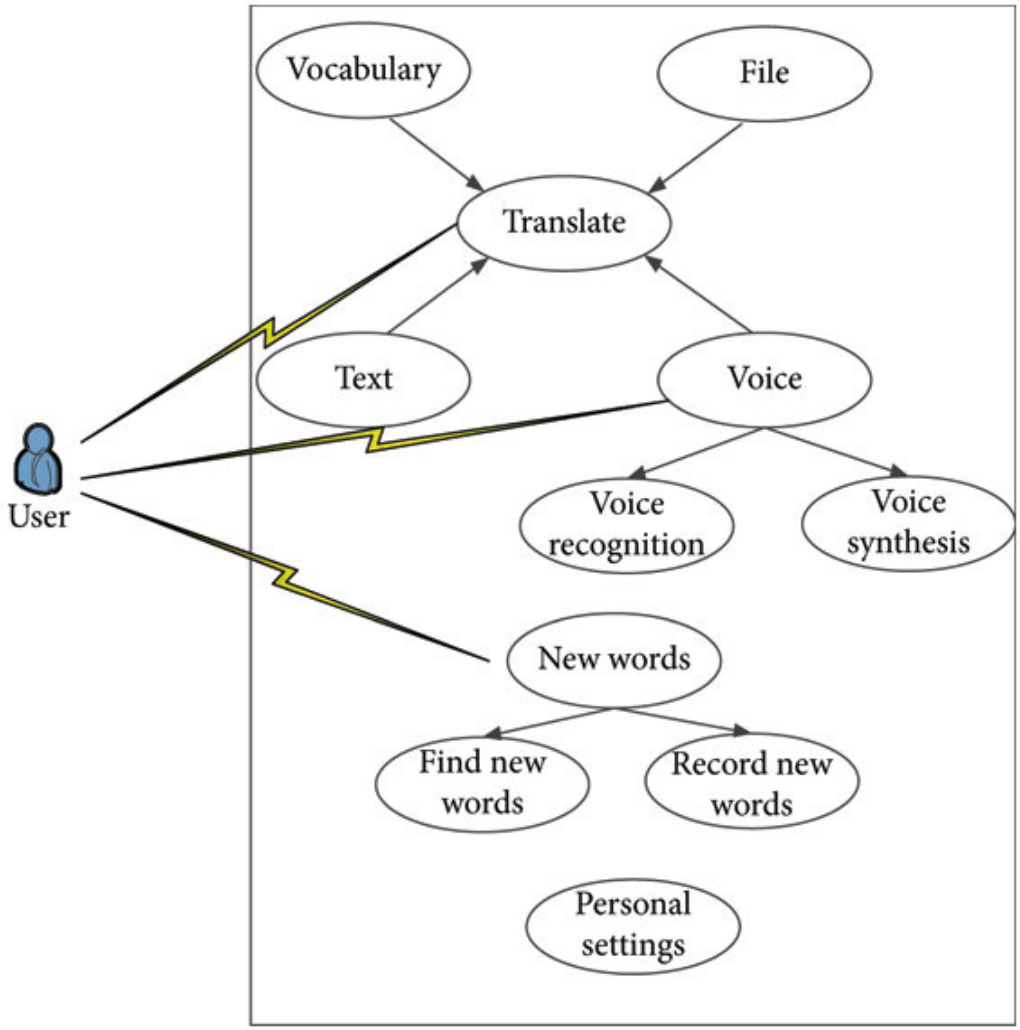


Figure 3.3: High-Level System Architecture Diagram.

tecture consists of: UI Layer (Flutter Widgets), Business Logic Layer (Dart for file handling, state management, history), Local Storage (SQLite/SharedPreferences), API Integration Layer (Dart HTTP to Google Translate API), and the external Google Translate API.

# Chapter 4

# Implementation

## 4.1 Interface Design/Front-End

The front-end is developed using Flutter, ensuring a consistent look and feel across Android (and potentially iOS). Key UI elements include:

- **Main Translation Screen:**
  - Button to "Upload .txt File".
  - Dropdown/Selector for "Target Language".
  - Display area for original text (optional or snippet).
  - Display area for translated text.
  - "Translate" button.
- **Translation History Screen:**
  - A scrollable list displaying the last 5 translations (e.g., showing a snippet of original/translated text and target language).
  - Option to tap on a history item to view the full translated text again.
- **Settings Screen/Area:**
  - A toggle switch for "Dark Mode".
- **Overall Design:**
  - Clean, intuitive navigation (e.g., using a BottomNavigationBar or Drawer).
  - Responsive layout adapting to different screen sizes.
  - Clear visual feedback for user actions (e.g., loading indicators during translation).
  - Implementation of both light and dark themes, selectable by the user.

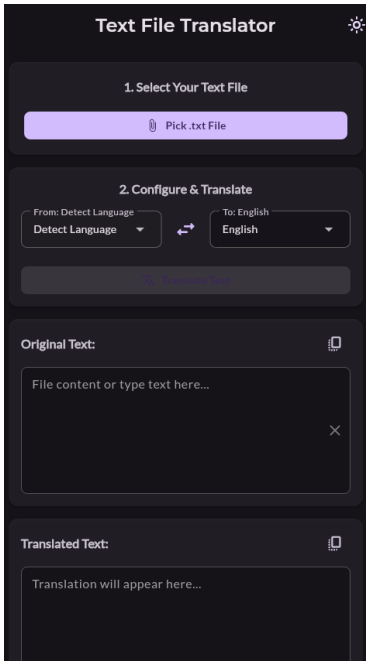


Figure 4.1: App Interface Mockups (Conceptual).

## 4.2 Application Logic & API Integration

- **File Handling:** Utilize Flutter plugins like `file_picker` to allow users to select `.txt` files. Read the content into a string.
- **API Integration (Google Translate API):** Use Dart's `http` package for POST requests to the Google Translate API endpoint. Include source text, target language code, and API Key. Parse JSON response. Implement error handling.
- **State Management:** Employ a Flutter state management solution (e.g., Provider, Riverpod, BLoC) for UI updates, dark mode state, and current translation.
- **History Management:** Use `sqflite` (SQLite) or `shared_preferences` to store the last 5 translations (original identifier, target language, translated text snippet, timestamp). Manage the 5-item limit.
- **Dark Mode Implementation:** Define light and dark themes. Allow user toggle. Persist preference using `shared_preferences`.
- **Java Integration (if necessary):** Use platform channels for specific Android functionalities if Flutter plugins are insufficient.

## 4.3 Modules/Features

The application is structured around the following core modules/features:

1. **File Input Module:** Handles Browse and loading of `.txt` files.
2. **Language Selection Module:** UI for choosing the target language.
3. **Translation Core Module:** Manages communication with Google Translate API, processes text, handles responses.
4. **UI/UX Module:** Renders screens, manages themes, ensures responsive design.
5. **History Module:** Manages local storage for the last 5 translations and displays history.
6. **Error Handling Module:** Provides user-friendly messages for errors.

# Chapter 5

# User Manual

## 5.1 System Requirement

### 5.1.1 Hardware Requirement (Typical Android Device)

- **Processor:** Dual-core 1.2 GHz or higher.
- **RAM:** 2 GB minimum (3 GB recommended).
- **Storage:** At least 50 MB of free internal storage.
- **Display:** Screen resolution of 720x1280 pixels or higher.
- **Internet:** Active internet connection required.

### 5.1.2 Software Requirement

- **Operating System:** Android 6.0 (Marshmallow) or newer.
- **Permissions:** Storage access, Internet access.
- **(Implicit) Google Play Services:** May be required.

## 5.2 User Interfaces

### 5.2.1 Main Translation Screen

- **Functionality:** Primary screen for translations.
- **Elements:** "Upload File" Button, "Select Target Language" Dropdown, "Translate" Button, Original Text View (optional), Translated Text View, Navigation to History/Settings.

### 5.2.2 Translation History Screen

- **Functionality:** View recent translation activity.
- **Elements:** Scrollable list of last five translations (original snippet/filename, target language, translated snippet, timestamp optional). Tapping an item might reload full translated text.

### 5.2.3 Settings (Dark Mode)

- **Functionality:** Customize app appearance.
- **Elements:** "Dark Mode" Toggle Switch. Preference saved.

# Chapter 6

# Conclusion

## 6.1 Conclusion

The "Mobile Text File Translator" project successfully delivers a user-friendly mobile application for translating `.txt` files. By leveraging the Flutter framework, the app provides a responsive interface and the potential for cross-platform deployment. The integration of the Google Translate API ensures access to high-quality translations across a wide range of languages. Key features, including direct file upload, a selectable dark mode for user comfort, and a locally stored history of the last five translations, address common pain points in existing mobile translation solutions. This project demonstrates the feasibility of creating a practical and efficient tool for users needing quick document translations on their mobile devices, enhancing productivity and accessibility.

## 6.2 Limitations

- **File Format:** Currently only supports `.txt` files.
- **Online Only:** Translation functionality is entirely dependent on an active internet connection.
- **API Dependency & Costs:** Relies on the Google Translate API (subject to terms, quotas, potential costs).
- **Limited History:** Translation history is limited to the last five entries and stored locally.
- **Formatting Preservation:** Only plain text content is processed.
- **Error Propagation:** Complex API error messages might not always be user-friendly.
- **Single File Translation:** No batch processing of multiple files.

## 6.3 Future Works

- **Expanded File Format Support:** Add support for `.docx`, `.pdf`.
- **OCR Integration:** Allow translation of text from images.
- **Offline Translation Mode:** Integrate on-device translation for limited languages.
- **Cloud Sync for History:** Optionally sync translation history across devices.
- **Expanded History & Management:** Offer larger history and search/management features.
- **In-app Text Editing:** Allow minor edits to original or translated text.
- **Automatic Source Language Detection.**
- **iOS Version:** Fully test and release an iOS version.
- **Batch Translation:** Functionality to translate multiple files.

# Bibliography


[1] Google Cloud Translation API Documentation. https://cloud.google.com/translate/docs *Provides guidelines for using the Google Translate API, including authentication, language support, and request formats.*

[2] Flutter Official Documentation. https://flutter.dev/docs *Comprehensive resource for Flutter framework, widgets, and development practices.*

[3] Dart Language Documentation. https://dart.dev/guides *Official documentation for the Dart programming language used in Flutter.*

[4] Material Design Guidelines. https://material.io/design *Guidelines for UI/UX design, including principles for dark themes.*

[5] sqflite (Flutter Plugin) Documentation. https://pub.dev/packages/sqflite *Documentation for the SQLite plugin used for local database storage in Flutter.*

[6] shared_preferences (Flutter Plugin) Documentation. https://pub.dev/packages/shared_preferences *Documentation for storing simple key-value data locally.*

[7] file_picker (Flutter Plugin) Documentation. https://pub.dev/packages/file_picker *Documentation for the plugin enabling file selection from device storage.*

[8] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is All You Need. In *Advances in Neural Information Processing Systems (NeurIPS). Introduces the Transformer model, foundational for many modern translation systems like Google Translate.*