FINAL PROJECT REPORT ON
"LANGUAGE LEARNER"

Submitted in partial fulfillment of requirement for the award of the degree of

BACHELOR IN COMPUTER APPLICATIONS

DEPARTMENT OF COMPUTER APPLICATIONS.

SESSION: 2024-2025

SUBMITTED BY:

EKTA SINGH (22201020026)
AARTI AGRAHARI (22201020001)
STUTI AGRAWAL (22201020074)
AISHLY KESARWANI (22201020012)

SUPERVISED BY:

MR. BHOOPENDRA SINGH
ASSISTANT PROFESSOR
DEPARTMENT OF
COMPUTER APPLICATIONS

UNITED UNIVERSITY,

RAWATPUR, PRAYAGRAJ, UTTAR PRADESH-211012

Certificate

This is to certify that the project report entitled **"LANGUAGE LEARNER"** submitted to United University, in partial fulfillment of the requirements for the award of the degree of **Bachelor of Computer Application (BCA)**, is an original work carried out by **Ms. Ekta Singh, Aarti Agrahari, Stuti Agrawal,** and **Aishly Kesarwani** under the supervision of **Mr. Bhoopendra Singh**. The content embodied in this project is genuine work carried out by us and has not been submitted whether to this University or to any other University/Institute for the fulfillment of the requirement of any course of study.

Date:

Signature

Ekta Singh (22201020026)

Aarti Agrahari (22201020001)

Stuti Agrawal (22201020074)

Aishly Kesarwani (22201020012)

Verified by the Supervisor

Acknowledgement

This Major Project is the result of contribution of many minds. I would like to acknowledge and thank my project guide **Mr. Bhoopendra Singh** for his valuable support and guidance. He guided me through the process from conception and till the completion of this project.I would also like to thank my **HoD Dr. Prashant Shukla** and **Dean Dr. Chetan Vyas** and all my faculties. I thank to lab staff members and other non-teaching members.
I am very thankful for the open-handed support extended by many people. While no list would be complete, it is my pleasure to acknowledge the assistance of my friends who provided encouragement, knowledge, and constructive suggestions.

Signature

Ekta Singh (22201020026)

Aarti Agrahari (22201020001)

Stuti Agrawal (22201020074)

Aishly Kesarwani (22201020012)

Table of Contents

1. Introduction 6
2. Objective 7
   2.1. Purpose: 7
   2.2. Scope: 7
3. System Analysis 8
   3.1. Functional Requirements 8
      3.1.1. Language Learning Features 8
      3.1.2. User Interface (UI) & Interactivity 8
      3.1.3. Data Handling & File Management 8
      3.1.4. Performance & Optimization 8
   3.2. Non-Functional Requirements 9
      3.2.1. Performance 9
      3.2.2. Usability 9
      3.2.3. Scalability 9
      3.2.4. Security 9
   3.3. User Requirements 9
   3.4. System Constraints 10
      3.4.1. Hardware Limitations 10
      3.4.2. Software Constraints 10
      3.4.3. Storage Limitations 10
      3.4.4. Internet Dependency 10
      3.4.5. File Format Support 10
      3.4.6. Input Size Limitation 10
      3.4.7. Browser Compatibility 10
      3.4.8. Language Support Limitation 10
4. Feasibility Study 11
   4.1. Technical Feasibility 11
   4.2. Economic Feasibility 11
   4.3. Operational Feasibility 12
   4.4. Legal Feasibility 12
   4.5. Scheduling Feasibility 12
5. Hardware and Software Requirements 13
   5.1. Hardware Requirements 13
      5.1.1. Minimum Requirements 13

5.1.2. Recommended Requirements 13
5.2. Software Requirements 13
5.2.1. Development Tools 13
5.3. Deployment Requirements 14
6. System Design 15
6.1. System Architecture 15
6.2. Modules and Components 16
6.3. User Interface Design 17
6.4. Security and Sessions 17
7. Activity Diagram 18
8. Data Flow Diagram 19
9. Use Case Diagram 21
10. Implementation And Coding 22
10.1. Project Organization and Structure 22
10.2. Core Implementations 23
10.2.1. User Input Handling 23
10.2.2. Speech and Text Processing 23
10.2.3. Database and User Management 23
10.2.4. User Interface (UI) 23
10.2.5. Download and Library System 24
11. Progress Screenshots 25
12. Future Scope 29
13. References 30

1. Introduction

In an increasingly interconnected world, the ability to communicate effectively across languages has become a critical skill. Whether for academic purposes, professional growth, travel, or personal development, language proficiency enhances one's ability to engage with diverse communities.

However, mastering a language involves not just vocabulary acquisition but also grammatical accuracy, correct pronunciation, contextual understanding, and effective usage. Recognizing these needs, the Language Learner project was conceived to offer a comprehensive, AI-driven solution to support users in improving their language skills.

Language Learner is an innovative web application that combines modern Artificial Intelligence (AI) techniques, Natural Language Processing (NLP), and Speech Recognition technologies to create an interactive and user-friendly platform for language improvement. The application provides a wide range of functionalities designed to assist users at different stages of their language learning journey.

It offers grammar and spell checking for both typed and spoken text, real-time pronunciation evaluation through speech-to-text conversion, multi-language translation, synonyms and antonyms lookup, dictionary search, and the ability to download and manage corrected or translated text files through a personalized library feature.

The backend of the application is developed using Python with the Flask web framework, ensuring a lightweight and efficient server-side operation.

The frontend is designed with HTML, CSS, and Bootstrap, offering users an attractive, responsive, and intuitive interface. A MySQL database is used for securely storing user credentials and managing session activities, while various APIs and libraries such as TextBlob, LanguageTool, Google Speech Recognition, and Google Translator are integrated to provide advanced language processing capabilities.

The Language Learner platform enables users to interact in multiple ways:

- They can type text directly into the interface for correction.
- They can upload text files to detect and correct errors.
- They can translate text or speech into multiple international languages like Hindi, French, Spanish, German, Japanese, Chinese, and English.
- They can find synonyms and antonyms of words to build a richer vocabulary.
- They can search dictionary definitions to better understand word usage and meaning.
- They can download their corrected or translated work and manage it through a built-in library.
- Security and privacy are critical considerations in the project. Passwords are encrypted using bcrypt hashing ensures that user data remains protected throughout their interaction with the system.

2. Objective

The objective of the Language Learner project is to design and develop a comprehensive, AI-powered web application that supports users in improving their English language proficiency through a combination of spell checking**,** grammar correction**,** speech recognition, and text translation services.

This project seeks to bridge the gap between traditional language learning methods and modern technological advancements by integrating Natural Language Processing (NLP)**,** Machine Learning (ML)**,** and speech processing techniques into an interactive and user-friendly platform. The application is intended to assist not only students but also professionals and language enthusiasts in enhancing their written and spoken English through real-time feedback and corrective suggestions.

2.1. Purpose:

The purpose of Language Learner is to empower users with an AI-driven platform that improves their language skills through real-time grammar checking, spelling correction, translations, vocabulary building, and speech-based learning.
Inspired by modern educational tools, the project integrates cutting-edge NLP models**,** speech-to-text technologies, and user-friendly design to offer an engaging and accessible learning environment.

2.2. Scope:

The scope of Language Learner includes:

- **Spell and Grammar Correction**:
  To accurately identify spelling mistakes and grammatical errors in user-input text, uploaded files, or voice input, and provide intelligent correction suggestions using advanced NLP models such as TextBlob and LanguageTool.

- **Voice-to-Text Functionality**:
  To implement a robust speech recognition system that captures spoken language using microphones and converts it into written text using APIs like Google Speech Recognition, enabling pronunciation learning and spoken language practice.

- **Multilingual Translation**:
  To offer real-time translation of text from English into multiple languages (Hindi, French, German, Spanish, Japanese, Chinese, etc.) and vice versa, leveraging APIs like Googletrans, thus facilitating bilingual or multilingual learning.

3. System Analysis

3.1. Functional Requirements

These define what the **Language Learner** application must accomplish:

3.1.1. Language Learning Features

- Users can input text manually or upload files (.txt, .docx) for grammar checking, translation, and vocabulary assistance.
- The system detects and highlights grammar errors, spelling mistakes, punctuation issues, and sentence structure problems.
- Real-time suggestions are provided for corrections, vocabulary enhancement, and translation.
- The system offers synonyms and antonyms for selected words to improve writing quality.
- Corrected and translated text can be downloaded or saved in the user's File Library.

3.1.2. User Interface (UI) & Interactivity

- A simple and intuitive UI should guide users to input text, view suggestions, and download corrected text.
- The UI should dynamically update based on the user's actions (input text, upload files, correct text).

3.1.3. Data Handling & File Management

- The application should allow users to upload text files for grammar checking.
- Corrected versions should be downloadable in different formats (e.g., .txt, .docx).
- Users should be able to view their corrected files in the "File Library" section.

3.1.4. Performance & Optimization

- The application should run smoothly on devices that meet the minimum hardware requirements.
- Speech recognition and language processing should provide real-time responses.

3.2.Non-Functional Requirements

These define the system's quality attributes:

3.2.1. Performance

- The application should process grammar checks and spell checks within 1–2 seconds for normal text sizes.

3.2.2. Usability

- The application should be beginner-friendly, requiring no technical training.
- Navigation should be simple, with all major features easily accessible on the interface.

3.2.3. Scalability

- The system should allow future updates, such as new languages and additional learning modules.

3.2.4. Security

- Uploaded text and corrected files should be handled securely.
- User data and files should not be lost due to crashes or network failures.

3.3. User Requirements

1. The application should be easy to use, even for non-technical users.
2. Grammar and spelling errors should be accurately detected and corrected.
3. Users should be able to upload text files and get corrections easily.
4. Users should have a visual confirmation (like a message or indicator) once the grammar check is complete.
5. The application should allow users to download the corrected text or save it in their library for later access.

3.4. System Constraints

3.4.1. Hardware Limitations

The application must be optimized for lower-end devices with limited processing power.

3.4.2. Software Constraints

The project is built using Python, so only compatible frameworks and libraries can be used.

3.4.3. Storage Limitations

Learning progress data should be stored efficiently to minimize storage usage.

3.4.4. Internet Dependency

The application requires a stable internet connection to access grammar checking APIs, translation services, and vocabulary databases.

3.4.5. File Format Support

Only .txt and .docx file formats are supported for upload and correction. Other file types (e.g., .pdf, .odt) are not currently supported.

3.4.6. Input Size Limitation

The system can handle text inputs up to a certain limit (e.g., 10,000 words or 5 MB file size). Very large files may not be processed efficiently.

3.4.7. Browser Compatibility

The application is optimized for modern browsers (Google Chrome, Mozilla Firefox, Microsoft Edge). It may not work properly on outdated browsers.

3.4.8. Language Support Limitation

Translation and grammar checking are available only for supported languages (e.g., English, Hindi, Spanish). Unsupported languages will not have full functionality.

4. Feasibility Study

4.1. Technical Feasibility

This examines whether the required technologies and tools are available for development. **Feasible** – The Grammar Checking and Language Assistance application is developed using appropriate technologies.

- **Development Tools:** Python, Flask, HTML, CSS, JavaScript.
- **Programming Language:** Python (widely used for backend development and text processing tasks).
- **Database Management:** MySQL (for user authentication, user files, session management).
- **User Interaction:**
  - Text input for grammar checking and vocabulary enhancements.
  - File upload (.txt, .docx) for grammar checking.
  - Voice-to-text conversion for quick text input (optional).
  - Translation of text into multiple languages.
  - Synonyms and antonyms lookup using Datamuse API.
  - Dictionary definitions lookup.
- **Performance Considerations:**
  - Optimized for desktops, laptops, and mobile browsers.
  - Supports real-time grammar correction and translation with minimal delay.

Challenges:

- Ensuring real-time performance while integrating grammar checking, translation, and vocabulary APIs.
- Handling large text files efficiently without server lag.

4.2. Economic Feasibility

This assesses whether the project is cost-effective.
Feasible – Since the project is developed for academic purposes, development costs are minimal.

- **Development Tools:**
  - Open-source frameworks and libraries (Flask, MySQL, free APIs like Datamuse and Google Translator).
- **Potential Costs:**
  - Future hosting costs if deployed publicly.
  - Costs for scaling storage or using premium APIs if needed later.

Challenges:

- If the system expands to handle many users, hosting, storage, and premium API costs may arise.

4.3. Operational Feasibility

This determines whether the system will function effectively and be user-friendly. **Feasible** – The system is simple to use and performs its functions efficiently.

- Clean and intuitive interface for grammar checking, translation, and vocabulary lookup.
- Simple upload and download functionality for corrected files.
- Dashboard to access all features easily.
- File Library to manage saved files.

Challenges:

- Ensuring smooth user experience when switching between grammar checking, translation, and synonyms/antonyms search.

4.4. Legal Feasibility

This verifies compliance with software licensing and intellectual property laws. **Feasible** – The project uses open-source tools and publicly available APIs.

- Usage of open-source libraries and APIs ensures license compliance.
- No copyrighted or unauthorized material is used.

Challenges:

- If the project is commercialized in the future, licensing for third-party APIs (e.g., Google Translation, Datamuse API) must be reviewed.

4.5. Scheduling Feasibility

This assesses whether the project can be completed within the academic timeline. **Feasible** – The project follows a structured plan for timely delivery.

- **Development (Coding & Implementation):** In Progress.
- **Testing and Refinement:** Planned after completing main features.
- **Final Deployment:** Scheduled to meet academic project deadlines.

Challenges:

- Balancing project development with academic workload and ensuring enough time for testing.

5. Hardware and Software Requirements

5.1. Hardware Requirements

The hardware configuration required for smooth running of the Grammar and Spell Checker Web Application is:

5.1.1. Minimum Requirements

- **Processor:** Dual-core CPU, 1.6 GHz or higher
- **RAM:** 2 GB
- **Storage:** 500 MB free disk space
- **Operating System:** Windows 7 (64-bit) and above.
- **Peripherals:** Keyboard, Microphone (only for voice input feature)

5.1.2. Recommended Requirements

- **Processor:** Quad-core CPU, 2.0 GHz or higher
- **RAM:** 4 GB or higher
- **Storage:** 1 GB free disk space
- **Internet Connection:** Stable internet connection for grammar checking, translation, dictionary, and synonyms services
- **Operating System:** Windows 10 (64-bit) and above.
- **Peripherals:** Good quality microphone (for better voice-to-text performance)

5.2. Software Requirements

The application is developed and deployed using the following tools and technologies:

5.2.1. Development Tools

- **Programming Language:** Python
- **Framework:** Flask (for backend web development)
- **Database:** MySQL (for user data and file management)
- **Libraries and APIs:**
    - language_tool_python (Grammar Checking)
    - TextBlob (Spelling Correction)
    - Googletrans (Translation)
    - Datamuse API (Synonyms and Antonyms)
    - SpeechRecognition (Optional voice input)
- **Frontend Technologies:** HTML5, CSS3, JavaScript

5.3. Deployment Requirements

Supported Platforms: Windows, macOS (through modern browsers)

Browser Support: Google Chrome, Mozilla Firefox, Microsoft Edge (HTML5 and JavaScript compatible)

Server Requirements: Python environment with Flask support.

6. System Design

The system is designed as a modular, user-friendly web application that offers grammar correction, translation, vocabulary improvement, and file management in an efficient and scalable way.

The design follows the Client-Server Architecture and is divided into three main layers:

6.1. System Architecture

Client (Frontend)

- Built with HTML, CSS, Bootstrap, and JavaScript.
- Allows users to:
    - Input or upload text
    - View corrected grammar and spelling
    - Use translator, dictionary, and synonym/antonym tools
    - Download corrected files
    - Navigate through dashboard, profile, and file library

Server (Backend)

- Built using Python and Flask framework
- Handles routes, sessions, authentication, file processing, and API integrations
- Integrates external services for:
    - Grammar/spell checking (language_tool_python, TextBlob)
    - Translation (googletrans)
    - Synonyms/antonyms (Datamuse API)
    - Dictionary lookup
    - Voice input (SpeechRecognition)
- Stores user credentials and file information using MySQL

Database Layer

- MySQL database is used to:
    - Store user account details (signup/login)
    - Track downloaded and saved files
    - Enable session management and secure access

6.2. Modules and Components

User Authentication

- Users can sign up, log in, and manage their profile
- Session-based access control using Flask sessions

Grammar & Spell Checker

- Uses TextBlob for spelling corrections
- Uses language_tool_python for grammar rule checking
- Highlights errors and provides suggestions

Voice-to-Text Input

- Uses Google SpeechRecognition API
- Converts spoken input into text for grammar correction

Translation Module

- Uses googletrans to translate user input into supported languages (e.g., Hindi, French, Spanish)

Synonym and Antonym Lookup

- Queries Datamuse API
- Displays related words and opposites to improve vocabulary

Dictionary Lookup

- Integrates a public API to show definitions for user-entered words

File Handling

- Upload .txt files for grammar checking
- Save and download corrected files
- View/download/delete files in the “Library” section

6.3. User Interface Design

- Bootstrap-based responsive design
- Key screens:
    - Sign In / Sign Up
    - Dashboard with tools (Spell Checker, Translator, Dictionary, Synonyms/Antonyms)
    - File Library
    - Profile/About page

6.4. Security and Sessions

- Passwords are hashed using Flask-Bcrypt
- Sessions are securely managed using Flask's session management
- Routes are protected using login decorators (@login_required)

7. Activity Diagram

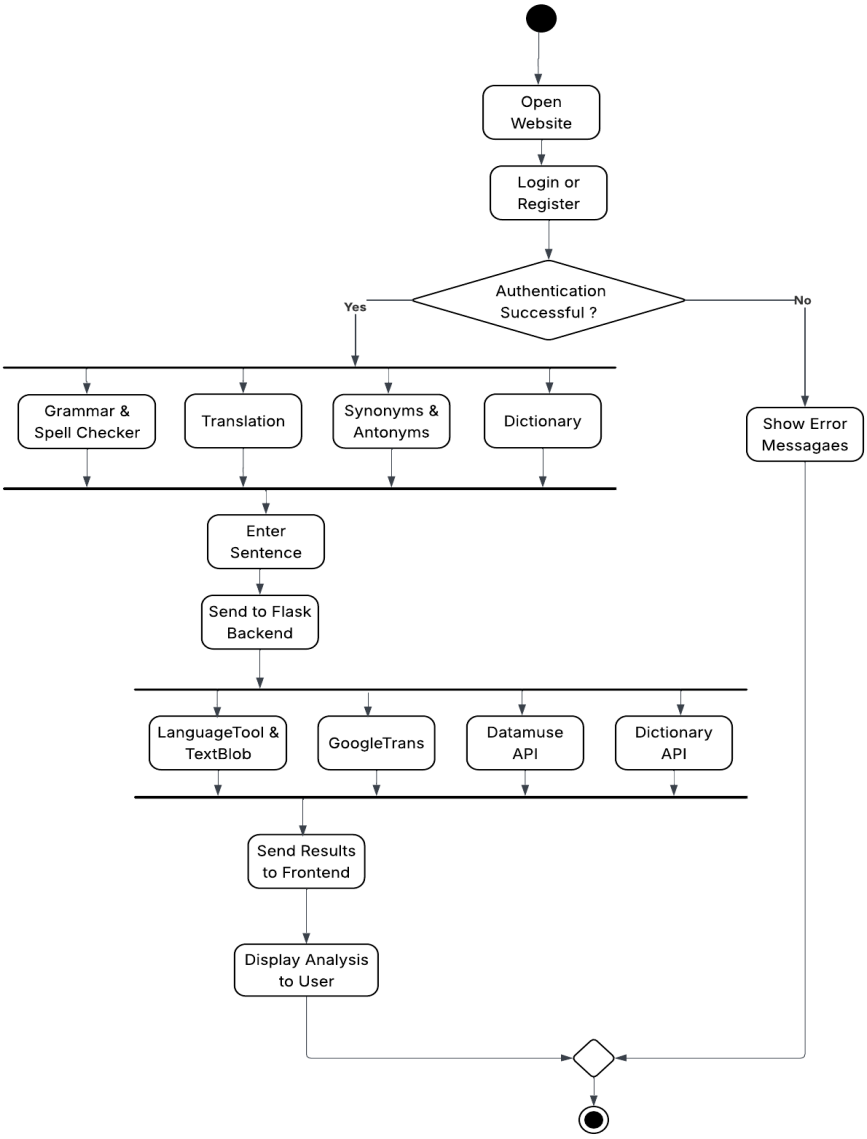

Figure 1: Activity Diagram

8. Data Flow Diagram

LEVEL 0 DFD

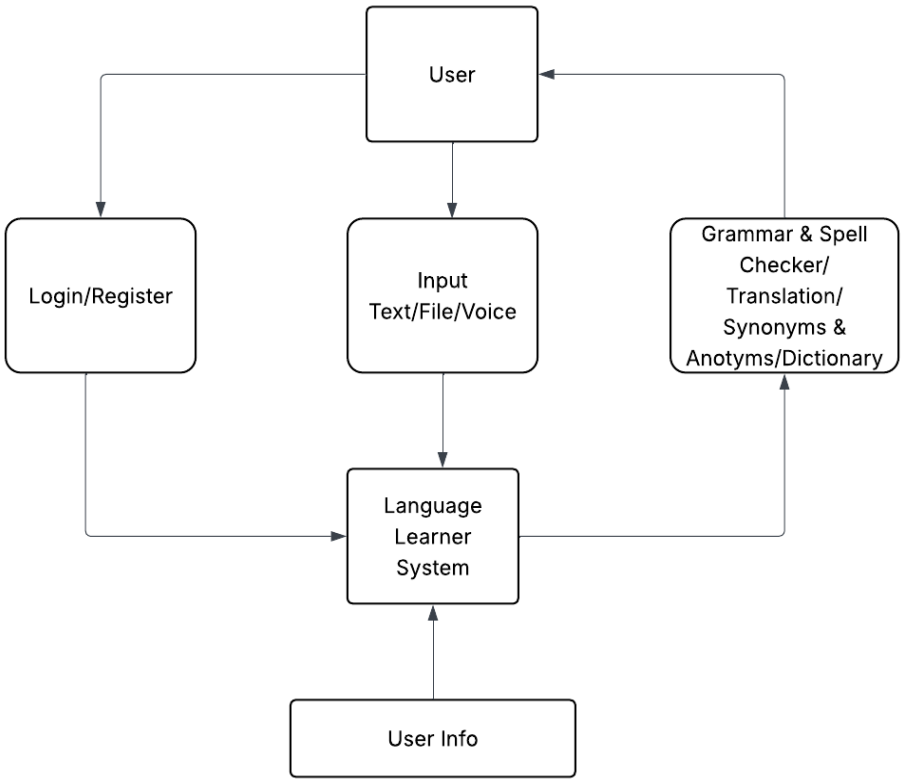

Figure 2: DFD Level 0

LEVEL 1 DFD

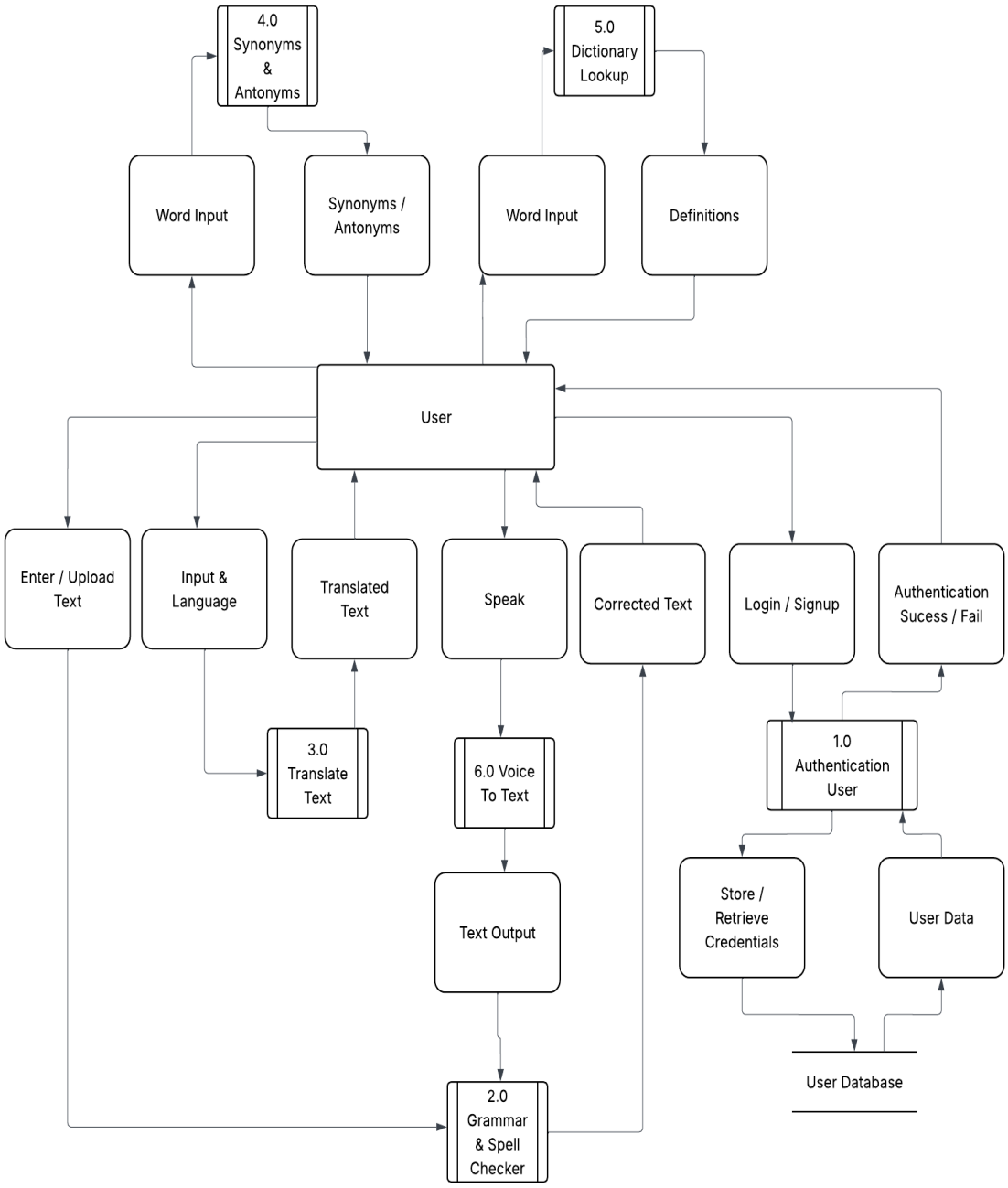

Figure 3: DFD Level 1

9. Use Case Diagram

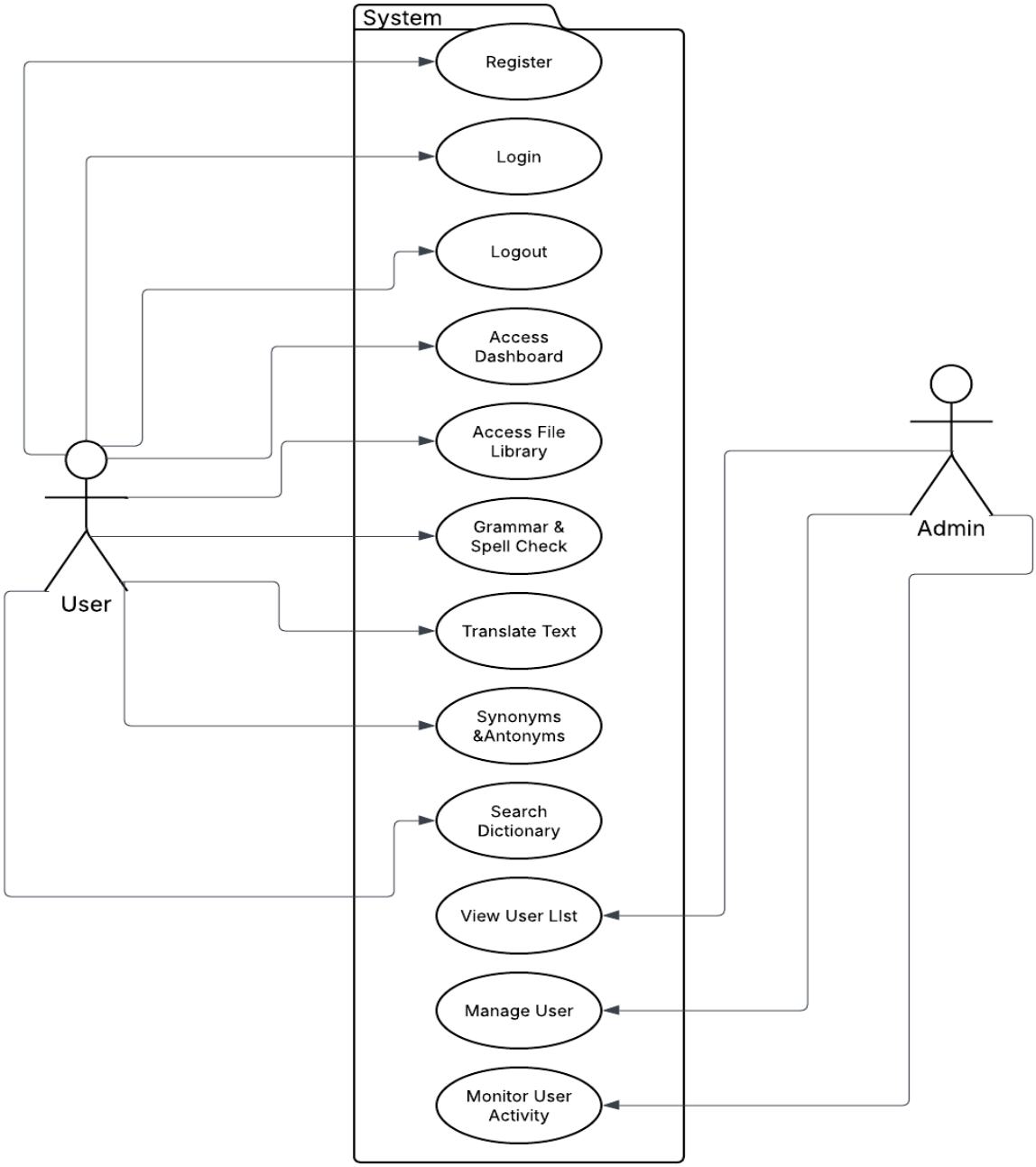

Figure 4: Use Case Diagram

10. Implementation And Coding

The implementation phase of the Language Learner application focuses on translating the design into a functional system using Python and relevant libraries. This phase includes setting up core functionalities, integrating speech processing, and ensuring a smooth user experience.

10.1. Project Organization and Structure

The Language Learner project follows a modular structure to ensure maintainability and efficient development.

Folder Structure

The project is organized into distinct folders for different components:

```
Language_Learner/
├── .vscode/
│   └── launch.json
├── __pycache__/
├── downloads/
├── templates/
│   ├── index.html
│   ├── dashboard.html
│   ├── dictionary.html
│   ├── library.html
│   ├── Profile.html
│   ├── signin.html
│   ├── signup.html
│   ├── synoanto.html
│   └── translator.html
├── venv
├── app.py
└── model.py
```

Each folder ensures that different components remain well-structured, supporting smooth integration and future updates.

10.2. Core Implementations

10.2.1. User Input Handling

- **Text Input:**
  Users can manually type text for spell checking**,** grammar correction, and translation.
- **File Upload:**
  Users can upload text files (.txt) for bulk grammar checking and vocabulary enhancement.
- **Voice Input:**
  Users can speak through a microphone. The application converts voice to text using Google Speech-to-Text API, which is then corrected for spelling and grammar.

10.2.2. Speech and Text Processing

- **Speech Recognition:**
  Converts spoken words into editable text using the SpeechRecognition library.
- **Grammar and Spell Checking:**
  - LanguageTool API identifies grammatical errors, punctuation mistakes, and spelling issues.
  - TextBlob is used for correcting spelling errors at the word level.
- **Vocabulary Enhancement:**
  - Datamuse API fetches synonyms and antonyms to improve vocabulary.
  - Dictionary API is used to look up word definitions for better understanding.
- **Translation:**
  Text is translated into multiple languages using Google Translator API to promote multilingual communication.

10.2.3. Database and User Management

- **MySQL Database:**
  - Stores user login and registration data securely.
  - Manages user sessions and access control.

10.2.4. User Interface (UI)

- **Responsive Web Interface:**
  Designed using HTML5, CSS3, and Bootstrap 5 for clean, responsive layouts across devices.
- **Dashboard:**
  Provides easy navigation to all features — Grammar Checker, Translator, Synonyms/Antonyms, Dictionary, and Library.
- **Result Display:**
  Shows corrected text, translation results, grammar error highlights, and vocabulary suggestions interactively.
- **File Management:**
  Users can download corrected or translated files and manage their saved files through the Library module**.**

10.2.5. Download and Library System

- **Download Corrected Content:**
  After grammar correction or translation, users can download the output as a text file.
- **Library View:**
  A section where users can see all their previously downloaded files and have options to download again or delete files.

11. Progress Screenshots

Signup Page:

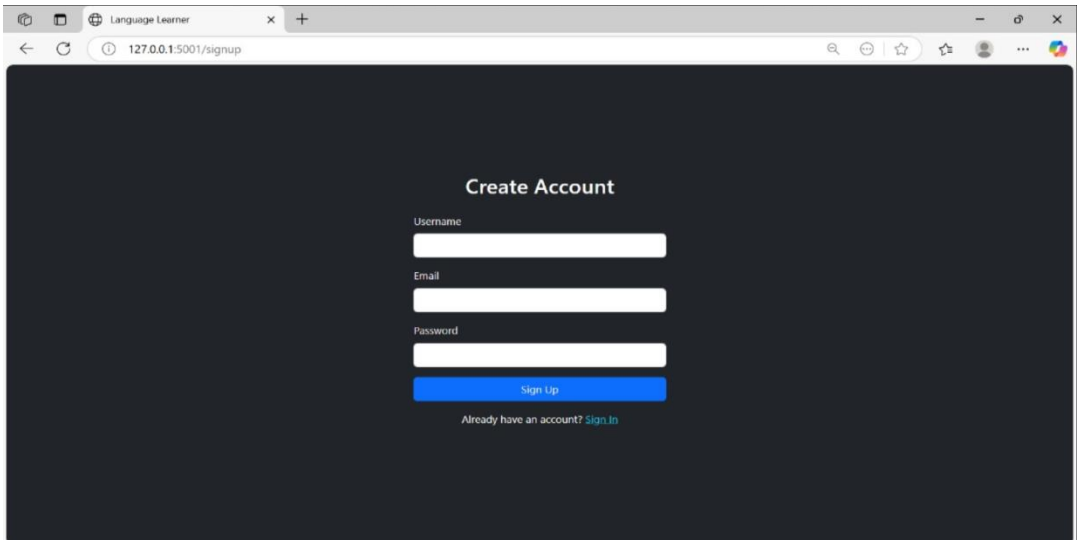

Figure 5: Signup Page

Login Page:

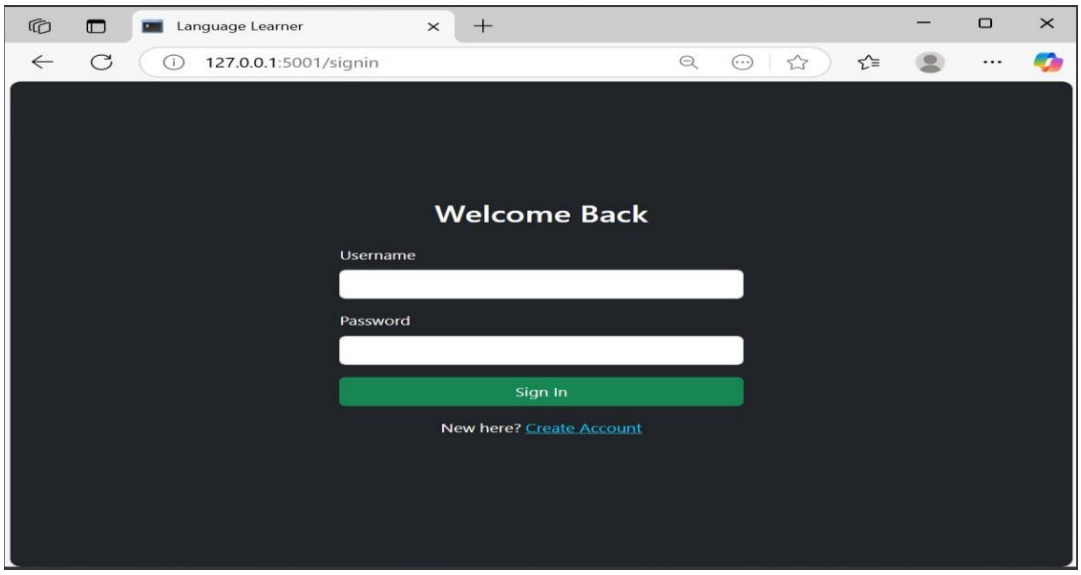

Figure 6: Login Page

Dashboard Page:

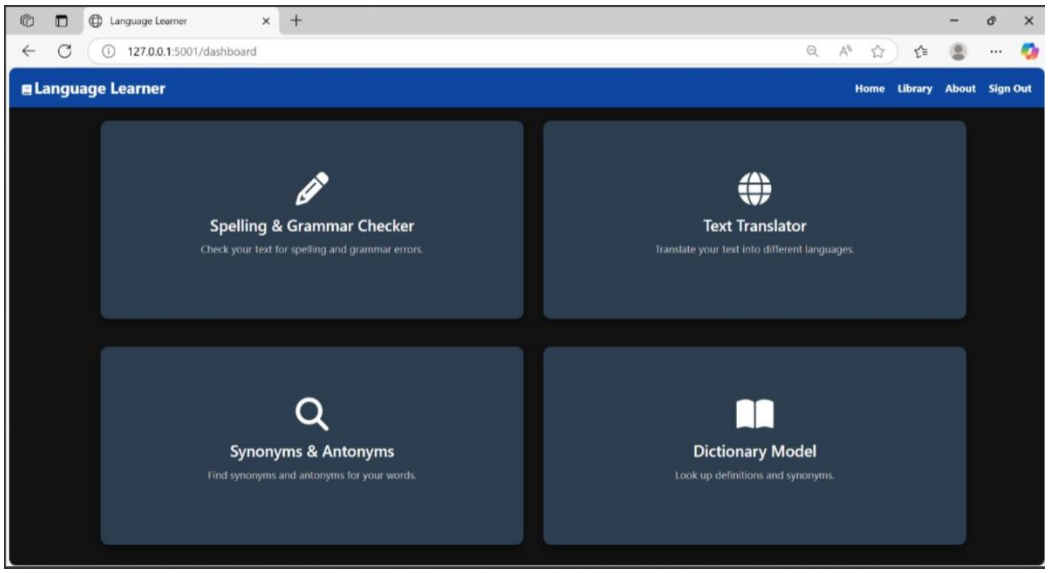

Figure 7: Dashboard Page

Grammar and Spell Checker Page

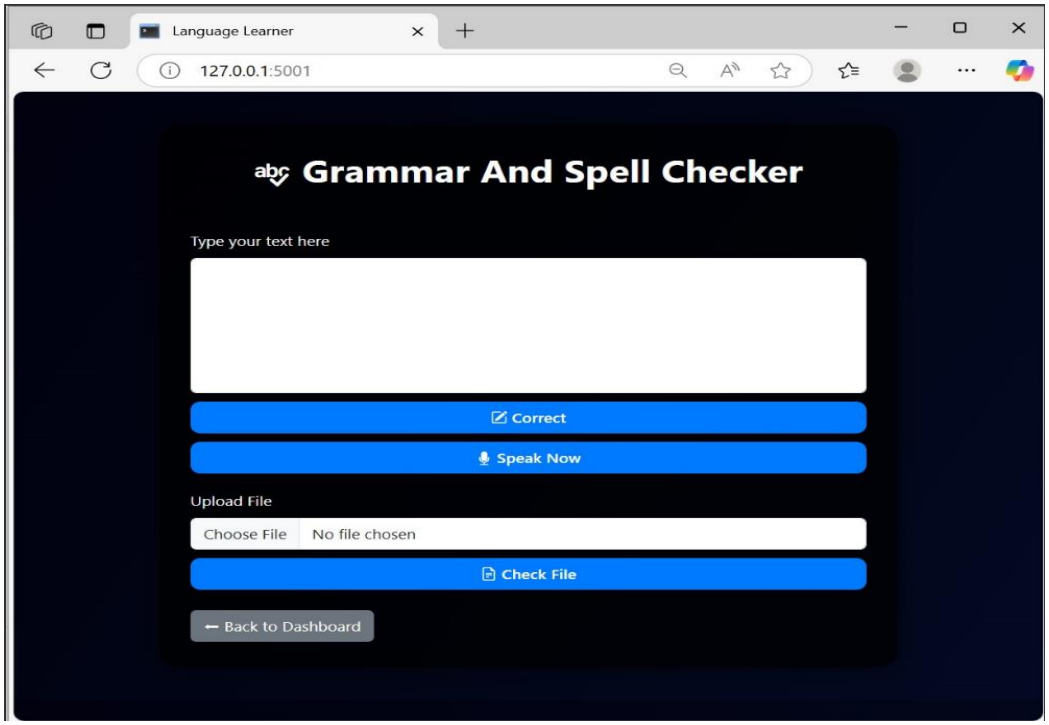

Figure 8: Grammar and Spell Checker Page

Translate Page:

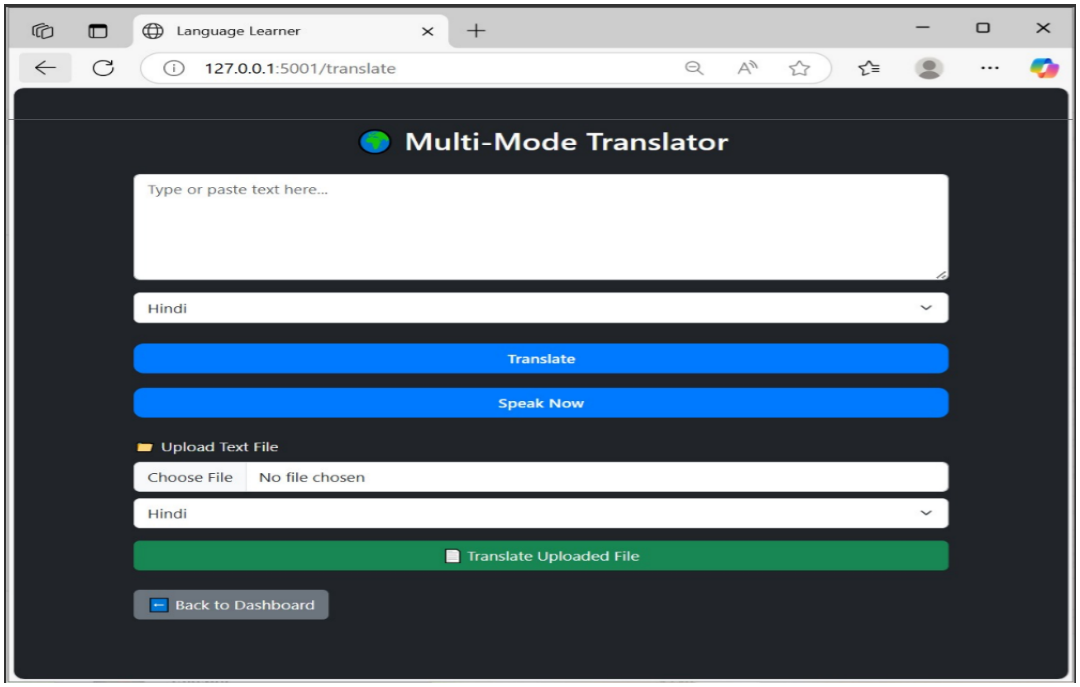

Figure 9: Translate Page

Synonyms & Antonyms Page:

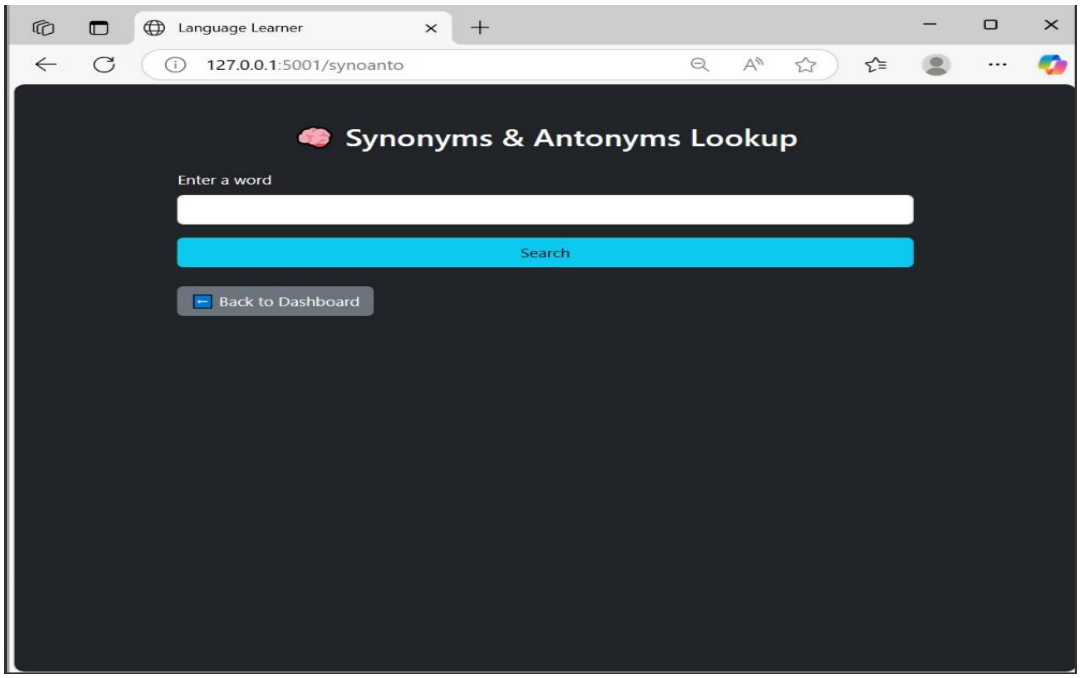

Figure 10: Synonyms & Antonyms Page

Dictionary Page:

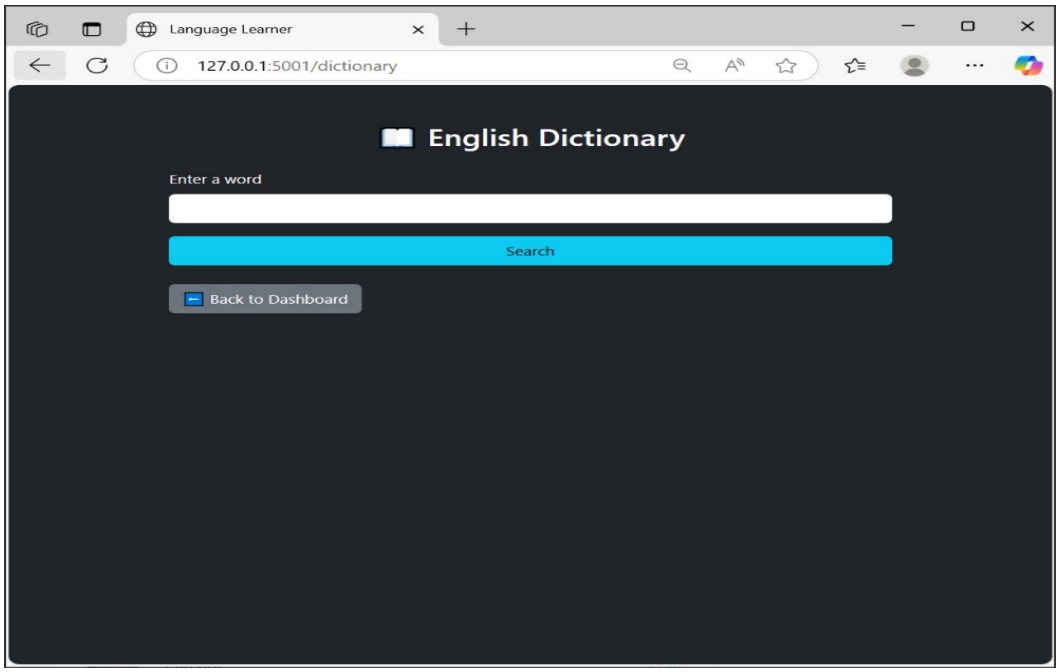

Figure 11: Dictionary Page

Profile Page:

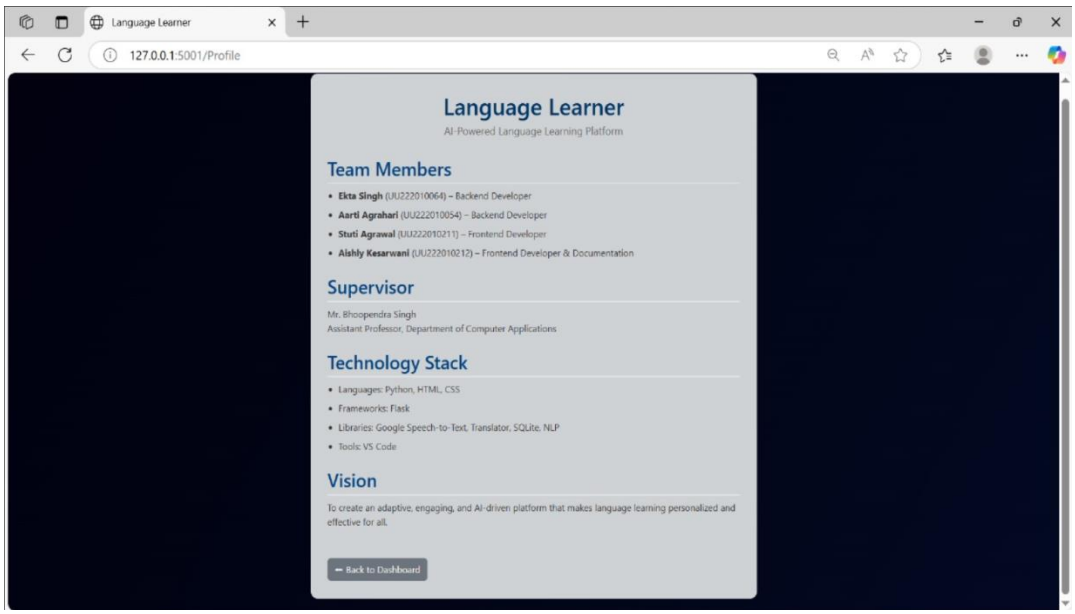

Figure 12: Profile Page

12. Future Scope

The Language Learner has strong potential for future improvements and enhancements. Possible extensions include:

- Support for more file types such as .docx, .pdf, and batch file processing.
- Mobile app development for Android and iOS to improve accessibility.
- User progress tracking, including grammar correction history and vocabulary improvement.
- AI-based smart suggestions for better sentence structure and writing tone.
- Offline mode for basic grammar checking without an internet connection.
- Pronunciation feedback for enhancing spoken English.
- Integration with platforms like Google Docs or browser extensions for real-time correction.

These enhancements will make the system more powerful, educational, and user-friendly for a wide range of users.

13. References

- LanguageTool API. (n.d.). LanguageTool – Open Source Grammar Checker. Retrieved from https://languagetool.org

- Googletrans Library. (n.d.). Google Translate API for Python. Retrieved from https://py-googletrans.readthedocs.io

- Datamuse API. (n.d.). Datamuse API – Word-finding query engine. Retrieved from https://www.datamuse.com/api

- SpeechRecognition Library. (n.d.). SpeechRecognition 3.8.1 Documentation. Retrieved from https://pypi.org/project/SpeechRecognition

- TextBlob. (n.d.). Simplified Text Processing. Retrieved from https://textblob.readthedocs.io
- Flask Documentation. (n.d.). Flask – The Python Microframework. Retrieved from https://flask.palletsprojects.com

- MySQL. (n.d.). MySQL Database. Retrieved from https://www.mysql.com

- Bootstrap. (n.d.). Bootstrap 5 Documentation. Retrieved from https://getbootstrap.com/docs/5.0/getting-started/introduction