**A**

**Project Report**

**On**

**Morse Code Translator**

**Submitted by**

**Kulkarni Soham Milind**

**Roll No.23334**

**MCA-I**

**SEM-I**

**Under Guidance of**

**Mrs. Dipali Patil**

**For the Academic Year 2023-24**

***Sinhgad Technical Education Society’s***

**Sinhgad Institute of Management**

**Vadgaon Bk Pune 411041**

**(Affiliated to SPPU Pune & Approved by AICTE New Delhi)**

### 

### Date:

**CERTIFICATE**

This is to certify that Mr. Kulkarni Soham Milind has successfully completed his/ her project work entitled **“Morse Code Translator”** in partial fulfillment of MCA – I SEM –I Mini Project for the year 2023-2024. He has worked under our guidance and direction.

### Mrs. Dipali Patil Dr. Chandrani Singh

**Project Guide Director, SIOM-MCA**

### Examiner 1 Examiner 2

**Date:**

**Place:** Pune

**DECLARATION**

I certify that the work contained in this report is original and has been done by me under the guidance of my guide.

* The work has not been submitted to any other Institute for any degree or diploma.
* I have followed the guidelines provided by the Institute in preparing the report.
* I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
* Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references.

### Name and Signature of Project Team Members:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Seat No.** | **Name of students** | **Signature of students** |
| **1** |  | **Kulkarni Soham Milind** |  |

**ACKNOWLEDGEMENT**

It is very difficult task to acknowledge all those who have been of tremendous help in this project. I would like to thank my respected guide **Mrs. Dipali Patil** for providing me necessary facilities to complete my project and also for their guidance and encouragement in completing my project successfully without which it wouldn’t be possible. I wish to convey my special thanks and immeasurable feelings of gratitude towards **Dr. Chandrani Singh, Director, SIOM-MCA.** I wish to convey my special thanks to all teaching and non-teaching staff members of **Sinhgad Institute of Management, Pune** for their support.

Thank You

Yours Sincerely,

Kulkarni Soham Milind.

# INDEX

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Chapter** | **Page No.** |
| 1 | **CHAPTER 1: INTRODUCTION** | 6-9 |
| 1.1 | Abstract | 6 |
| 1.2 | Existing System and Need for System | 7 |
| 1.3 | Scope of System | 8 |
| 1.4 | Operating Environment Hardware and Software | 8 |
| 1.5 | Brief Description of Technology Used | 9 |
| 2 | **CHAPTER 2: PROPOSED SYSTEM** | 10-11 |
| 2.1 | Feasibility Study | 10 |
| 2.2 | Objectives of the proposed system | 10 |
| 2.3 | Users of the system | 11 |
| 3 | **CHAPTER 3: ANALYSIS AND DESIGN** | 12-27 |
| 3.1 | Entity Relationship Diagram (ERD) | 12 |
| 3.2 | Class Diagram | 13 |
| 3.3 | Use Case Diagrams | 14 |
| 3.4 | Activity Diagram | 16 |
| 3.5 | Sequence Diagram | 17 |
| 3.6 | Component Diagram | 18 |
| 3.7 | Module and Hierarchy Diagram | 19 |
| 3.8 | Table Design | 20 |
| 3.9 | Data Dictionary | 21 |
| 3.10 | Sample Input and Output Screens | 22-27 |
| 4 | **CHAPTER 4: CODING Sample code** | 28-33 |
| 5 | **CHAPTER 5: LIMITATIONS OF SYSTEM** | 34 |
| 6 | **CHAPTER 6: PROPOSED ENHANCEMENTS** | 35 |
| 7 | **CHAPTER 7: CONCLUSION** | 36 |
| 8 | **CHAPTER 8: BIBLIOGRAPHY** | 37 |

Chapter 1

**Introduction**

**1.1 Abstract**

The Morse Code Translator is a mini project designed to provide users with the ability to translate text to Morse code and vice versa. Implemented in Java and utilizing a PostgreSQL database, the project offers user registration, login functionality, and a user-friendly dashboard. The frontend is designed using Java Swing, while the backend relies on PostgreSQL and JDBC drivers.

The primary objective of the Morse Code Translator project is to develop a versatile and user-friendly application that facilitates seamless translation between text and Morse code. This project aims to bridge the communication gap between traditional text-based messages and Morse code, offering users a tool to convert messages into Morse code and vice versa.

Functionalities Overview –

Translation Functionality: Enable users to input text and obtain its equivalent Morse code, as well as input Morse code and retrieve the corresponding text. This feature caters to both Morse code enthusiasts and those seeking an educational tool for Morse code understanding.

User Authentication: Implement a secure user authentication system to allow users to register, log in, and access personalized features. This ensures the privacy and security of user data.

Graphical User Interface (GUI): Design an intuitive and visually appealing GUI using Java Swing to enhance the user experience. The GUI should provide easy navigation, input fields, and interactive elements for a seamless interaction with the application.

Chart Display: Present a chart displaying Morse codes for letters, numbers, and symbols, offering users a quick reference guide. This feature supports user learning and engagement.

Database Integration: Utilize PostgreSQL for backend storage, maintaining user credentials securely and supporting data retrieval for logged-in users. The database integration ensures persistent storage of user information and preferences.

Translation History: Allow user to view his previous translations.

**1.2 Existing System Overview -**

Existing System:

The existing system for Morse code translation may consist of traditional methods where Morse code is manually deciphered or encoded by referring to printed reference charts.

It lacks user-friendliness and automation, making it time-consuming and prone to errors. Users need to have a deep understanding of Morse code patterns and symbols to use it effectively.

Challenges in the Existing System:

* Lack of Automation:

The manual nature of the existing system makes it cumbersome and slow, particularly for lengthy messages.

* Error-Prone:

Deciphering Morse code manually can lead to errors, especially for users who are not well-versed in Morse code.

* Limited Accessibility:

Printed Morse code charts are not easily accessible to everyone, limiting their use.

* Learning Curve:

Users need to invest time in learning Morse code and its symbols.

2.2 Updated System:

The updated system, the "Morse Code Translator" Java project, addresses the limitations of the existing system by providing a user-friendly and automated platform.

Key Features of the Updated System:

* User-Friendly Interface:

The updated system features a graphical user interface (GUI) that simplifies Morse code translation for users, eliminating the need for manual decoding.

* Text Input:

Users can input plain text messages directly into the application, making it accessible to a wide audience.

* Real-Time Translation:

The system translates text to Morse code and vice versa in real-time, offering immediate feedback.

* Error Handling:

It includes error handling mechanisms to detect and inform users of incorrect input.

Benefits of the Updated System:

* Improved Efficiency:

Automation reduces the time and effort required for Morse code translation.

* Error Reduction:

Automated translation minimizes the risk of errors in Morse code communication.

* Accessibility:

The user-friendly GUI makes Morse code accessible to a broader audience, including those without prior Morse code knowledge.

* Learning Aid:

The system can aid in learning Morse code, offering an interactive platform for practice.

* Modernization:

The project brings Morse code into the digital age, making it relevant in contemporary contexts.

**1.3 Scope of System -**

Scope:

The Morse Code Project in Java aims to create a versatile and user-friendly application for translating text messages into Morse code and vice versa. The project's scope encompasses various aspects, including:

* Morse Code Translation:

The primary focus is on accurately translating text to Morse code and Morse code back to text in real-time.

* User Interface (UI):

Developing an intuitive graphical user interface (GUI) to facilitate user interaction, input, and feedback.

* Error Handling:

Implementing error-checking mechanisms to validate input and provide meaningful error messages for incorrect input.

* Educational Value:

Offering a tool for learning and practicing Morse code, making it accessible to users regardless of their prior knowledge.

* Extensibility:
* Designing the project to be easily extensible, allowing for future enhancements and additional features.

**1.4 Operating Environment Hardware and Software:**

**Server-side requirement**

|  |  |
| --- | --- |
| **Software Requirement** | **Hardware Requirement** |
| Operating System: Windows 7 or above | Processor: Intel Core i3 or above |
| Front End: JavaSwing | RAM: 4GB or above |
| Back End: Java | HDD: 512 GB or above |
| Database: PostgreSQL |  |
| Web Browser: Chrome Mozilla Firefox |  |
|  |  |

**Client-side requirement**

|  |  |
| --- | --- |
| **Software Requirement** | **Hardware Requirement** |
| Operating System: Windows 7 or above | Processor: Intel Core i3 or above |
| Web Browser: Chrome, Mozilla Firefox | RAM: 2GB or above |
|  | HDD: 512 GB or above |

**1.5 Brief Description of Technologies Used -**

**1.5.1 Frontend Technologies**

* **Java Swing**: Utilized for designing the system's user interface (UI), providing a framework for creating interactive GUI components.

**1.5.2 Database**

* **PostgreSQL Database**: Employed as the backend database management system to store and manage user data, admin, translation history.
* **Connector & JDBC Drivers**: Used for establishing connectivity between the Java application and the PostgreSQL database, facilitating data retrieval, manipulation, and storage operations.

**1.5.3 System Architecture**

* **Java Programming Language**: Used as the primary language for developing the Course Companion system due to its platform independence and object-oriented nature.
* **Client-Server Interaction**: The system might follow a client-server architecture, where clients (instructors/students) interact with the server-side application, which manages and serves the requested data and functionalities.

**1.5.4 Considerations**

* **Scalability**: The choice of technologies and architecture may support scalability, allowing the system to accommodate increased users or future enhancements.
* **Cross-Platform Compatibility**: Java-based applications often offer cross-platform compatibility, enabling the Course Companion system to run on various operating systems without major modifications.

Chapter 2

**Proposed System**

**2.1 FEASIBILITY STUDY –**

**2.1.1. Technical Feasibility**

Technical feasibility assesses whether the proposed project can be successfully implemented from a technical perspective. For the Morse Code Project:

* Technical Expertise:

The project requires Java programming skills, knowledge of Java Database Connection. Ensuring that the development team possesses these skills is crucial for technical feasibility.

* Hardware and Software Requirements:

Ensuring that the specified hardware and software requirements for both the server-side and client-side are available and compatible with the project is essential.

**2.1.2 Economic Feasibility:**

Economic feasibility assesses whether the project is financially viable. Key considerations for the Morse Code Project include:

* Development Costs:

Estimate the costs associated with hardware, software, development tools, and human resources.

* Operational Costs:

Analyze ongoing operational expenses, including server hosting, maintenance, and potential support and updates after deployment.

**2.1.3 Operational Feasibility:**

Operational feasibility evaluates whether the proposed system can be effectively integrated into existing operations and whether it aligns with organizational goals. For project:

* User Adoption:

Assess whether potential users are willing to adopt and use the application.

* User Training:

Evaluate the ease of use of the application.

**2.2 OBJECTIVES OF THE PROPOSED SYSTEM –**

**Translation Functionality:**

Enable users to convert text to Morse code and vice versa. Provide a seamless and accurate translation mechanism.

**User Authentication:**

Implement a secure user registration and login system. Ensure data privacy and user-specific interactions.

**Graphical User Interface (GUI):**

Design an intuitive and visually appealing Java Swing GUI. Enhance user experience with easy navigation and interactive elements.

**Chart Display:**

Present a chart illustrating Morse codes for letters, numbers, and symbols. Serve as a quick reference guide for users.

**Database Integration:**

Utilize PostgreSQL for backend storage. Store and retrieve user credentials securely.

**Educational Tool:**

Serve as an educational resource for Morse code learning.Cater to Morse code enthusiasts and those seeking to understand Morse code.

**Dashboard Features:**

Provide users with a dashboard after login. Enable users to select translation options and access relevant features.

**The Morse Code Translator project aims to offer a comprehensive tool that combines functionality, security, user-friendliness, and educational value.**

**2.3 USERS OF THE SYSTEM –**

**1. Admin/Database Manager**:

Has overarching control over the system, managing database operations, system configurations, and user access privileges.

**2. Users:** The user functionalities include:

1. User Registration:

Users can create accounts by providing necessary information such as name username, and password.

2. User Login:

Secure user authentication enables registered users to log in to the application ensuring data privacy.

3. Dashboard Access:

Upon successful login, users are directed to a personalized dashboard displaying available options and features.

4. Translation Options:

Users can choose between translating text to Morse code or Morse code to text based on their communication needs.

5. Translation Input:

Input fields are provided for users to enter either text or Morse code, depending on their selected translation option.

6. Translation Output:

The translated output is displayed, allowing users to easily read the converted text or Morse code.

7. Chart Reference:

A chart displaying Morse codes for letters, numbers, and symbols serves as a reference guide for users during translation.

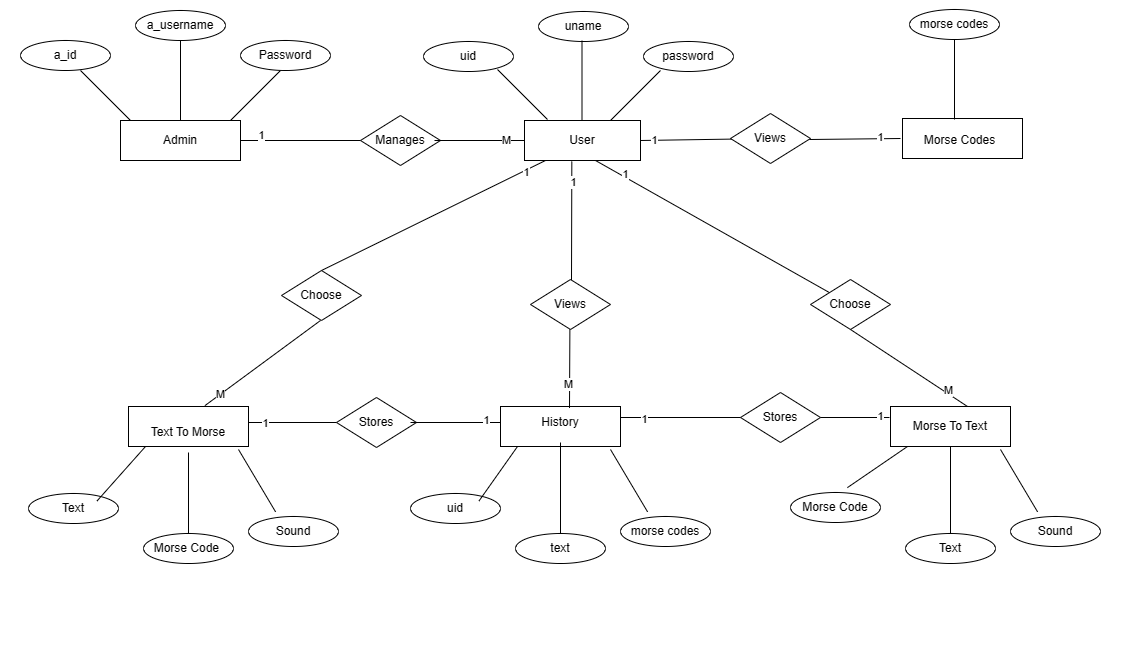
8. Logout Option:

Users have the option to log out, ensuring the security of their accounts and preventing unauthorized acces

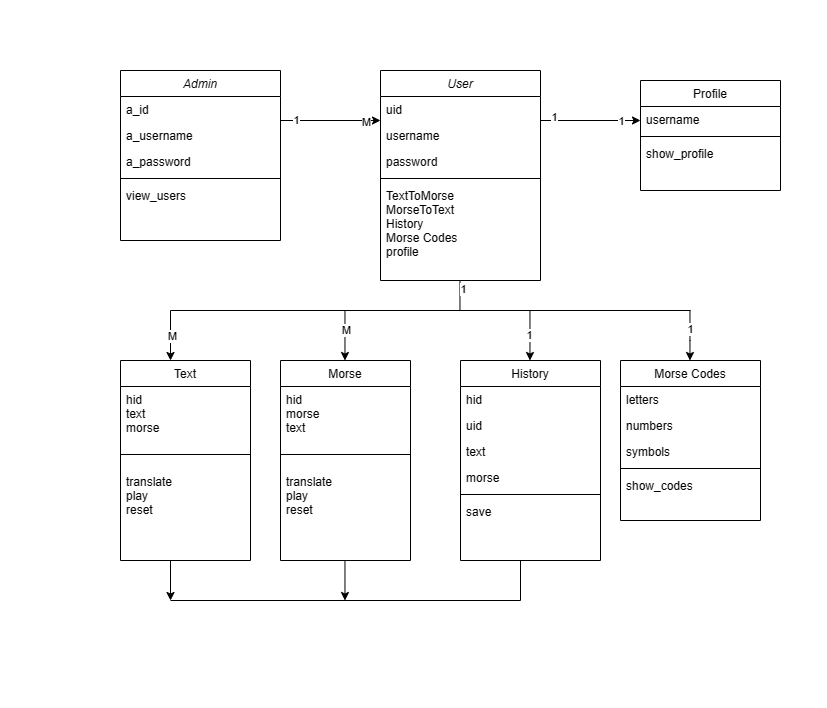
Chapter 3

**Analysis & Design**

**3.1 ENTITY RELATIONSHIP DIAGRAM**

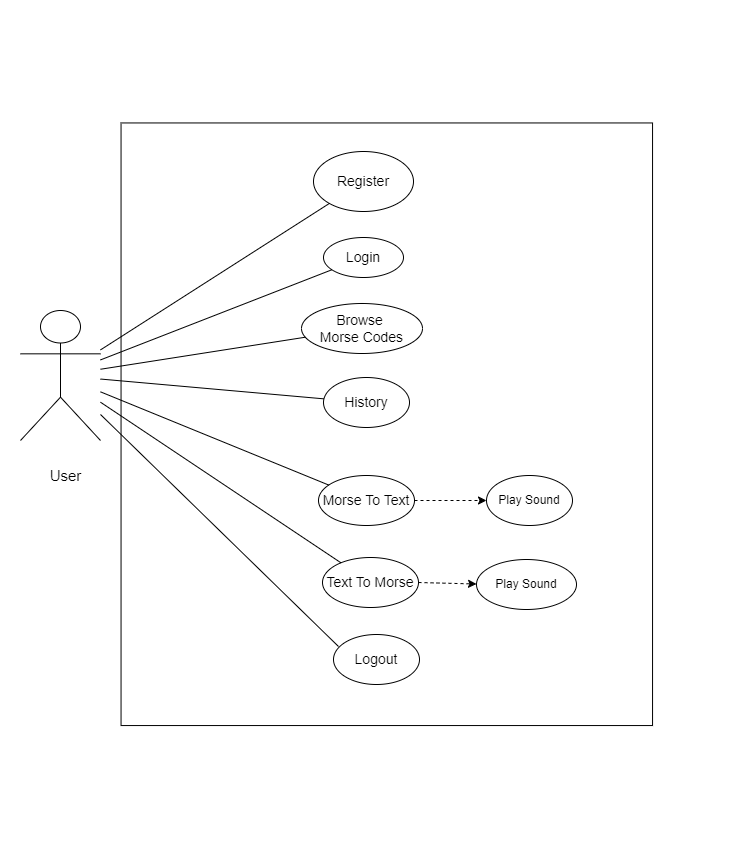


**3.2 CLASS DIAGRAM**



**3.3 USE CASE DIAGRAM**

**For User**

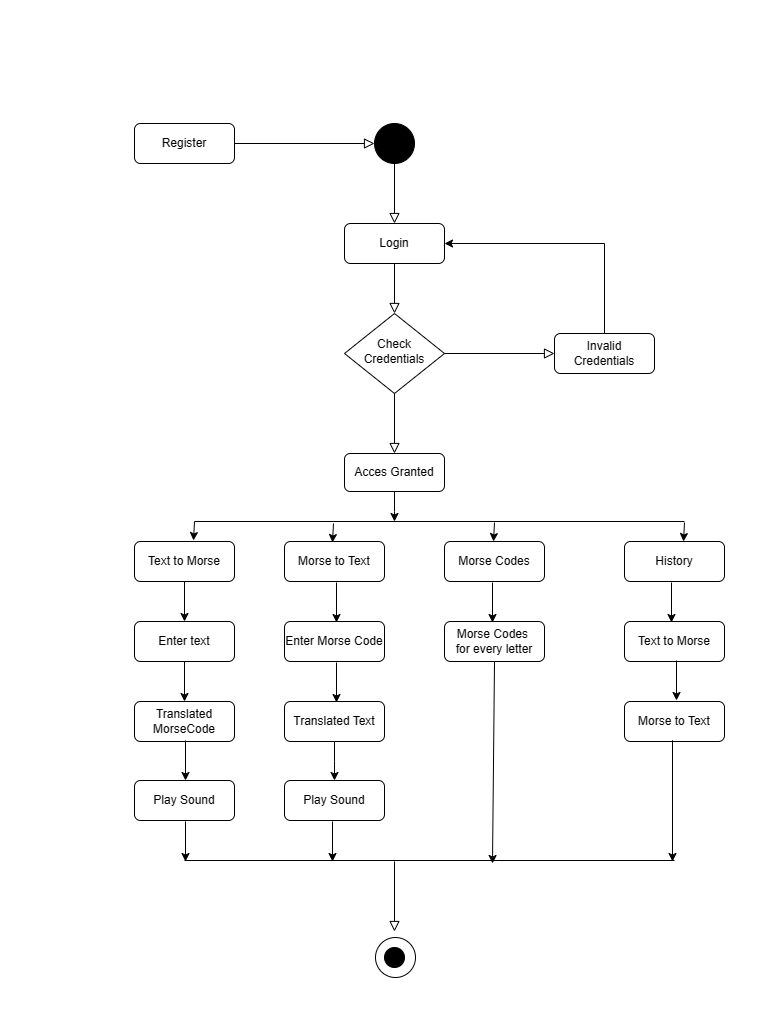


**For Admin**

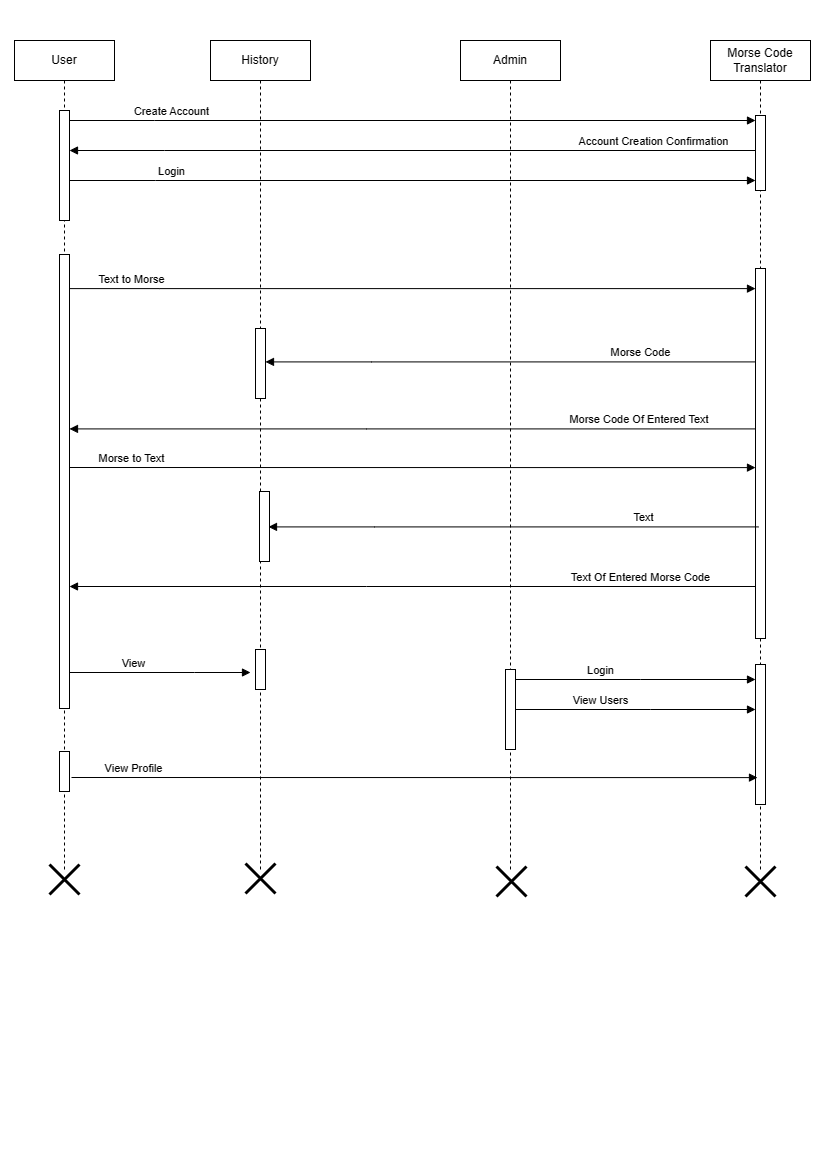
**A diagram of a person

Description automatically generated**

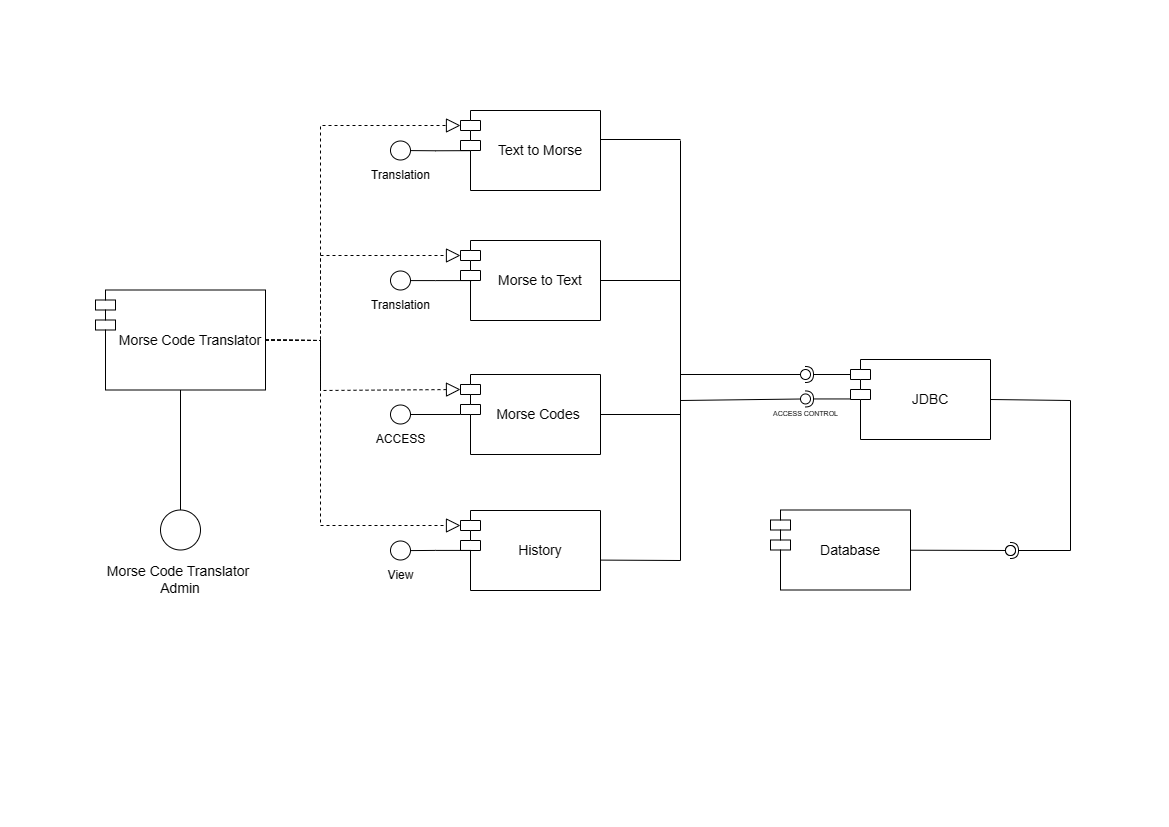
**3.4 ACTIVITY DIAGRAM**



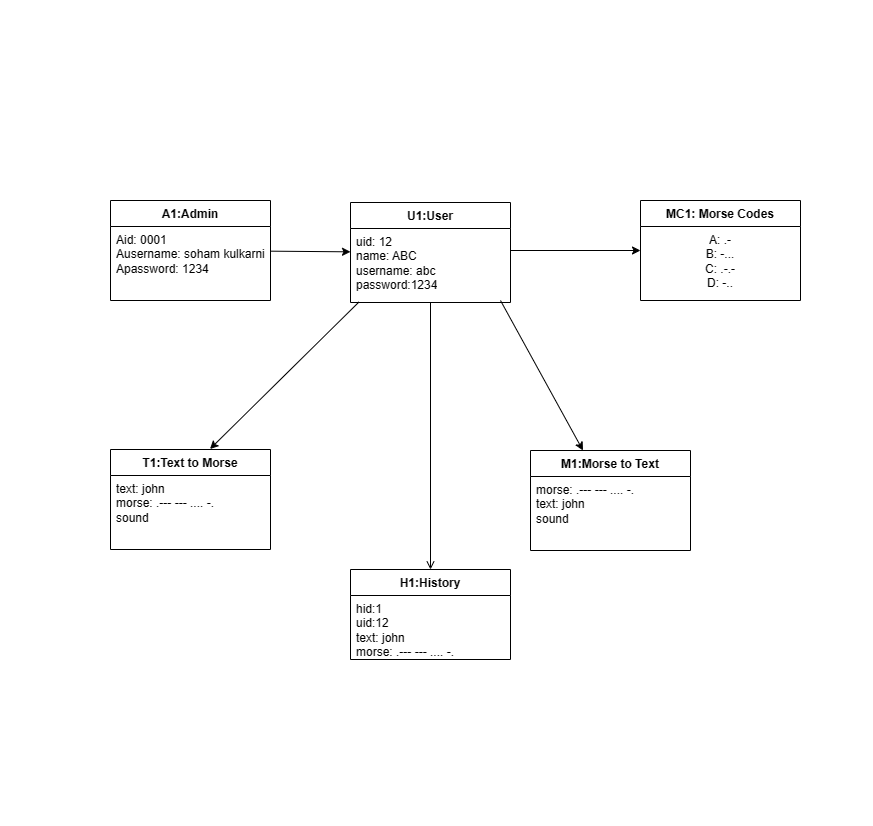
**3.5 SEQUENCE DIAGRAM**

****

**3.6 COMPONENT DIAGRAM**

****

**3.7 MODULE DIAGRAM**

****

**3.8 TABLE DESIGN –**

TABLE NAME – users

TABLE DESCRIPTION – Contains all the necessary information about User.

PIMARY KEY – uid

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr. No.** | **Field name** | **Field type** | **Field size** | **Constraints** | **Description** |
| 1 | uid | Int | 20 | Primary Key | student id |
| 2 | name | Varchar | 255 | Not Null | Name |
| 3 | username | Varchar | 255 | Not Null | Username |
| 4 | password | Varchar | 255 | Not Null | Password |

TABLE NAME – Text to Morse History

TABLE DESCRIPTION – Contains all the necessary information about Translation History of Text entered by the user to Morse Code.

PRIMARY KEY – id

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr. No.** | **Field name** | **Field type** | **Field size** | **Constraints** | **Description** |
| 1 | id | Int | 20 | Primary Key | Id for history |
| 2 | uid | Int | 20 | Foreign Key | user id |
| 3 | text | Varchar | 255 | Not Null | Text entered |
| 4 | morse | Varchar | 255 | Not Null | Morse Code given |

TABLE NAME – Morse Code to Text History

TABLE DESCRIPTION – Contains all the necessary information about Translation History of Morse Code entered by the user to Text.

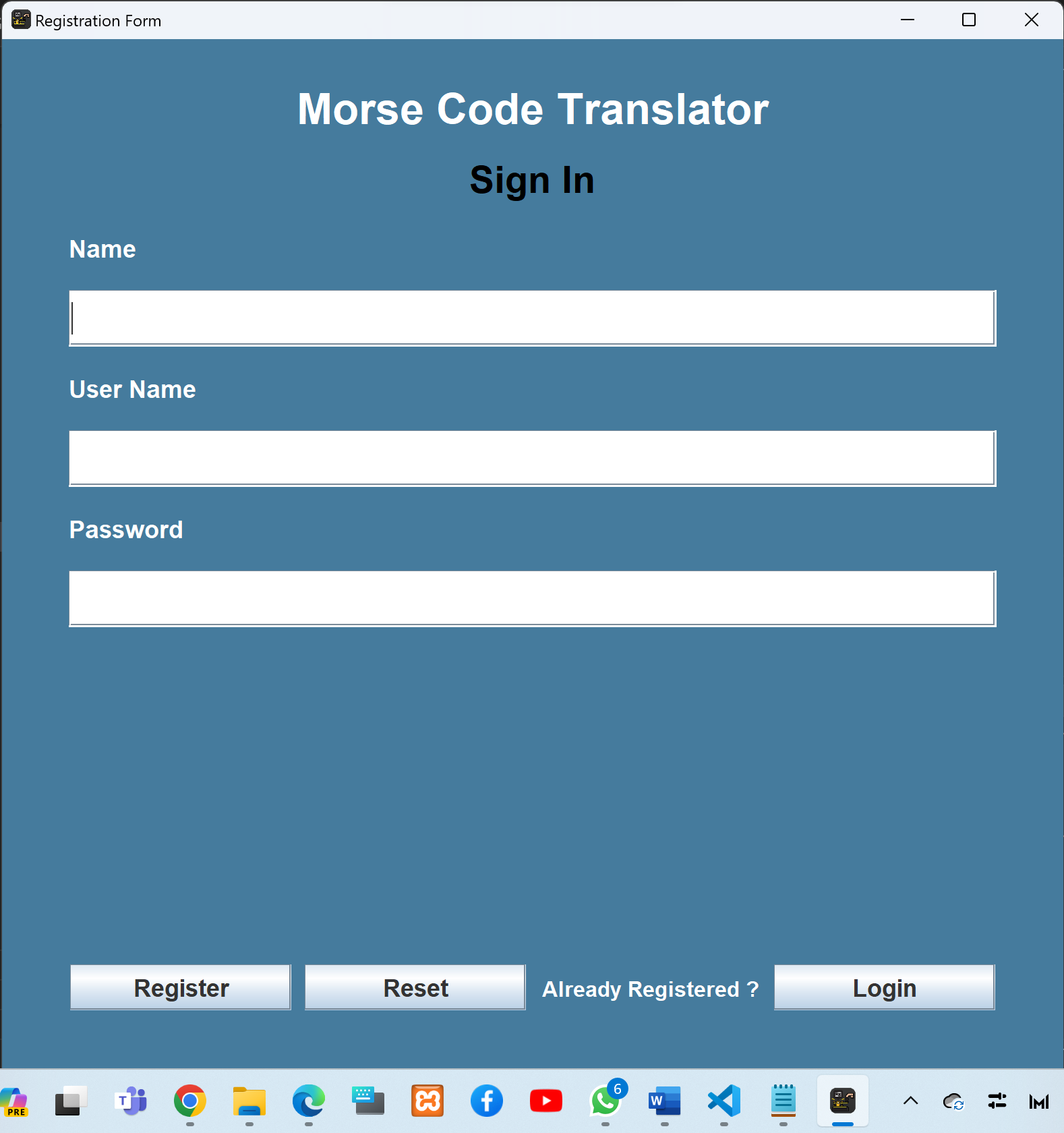
PRIMARY KEY – id

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr. No.** | **Field name** | **Field type** | **Field size** | **Constraints** | **Description** |
| 1 | id | Int | 20 | Primary Key | Id for history |
| 2 | uid | Int | 20 | Foreign Key | user id |
| 3 | morse | Varchar | 255 | Not Null | Morse Code entered |
| 4 | text | Varchar | 255 | Not Null | Text given |

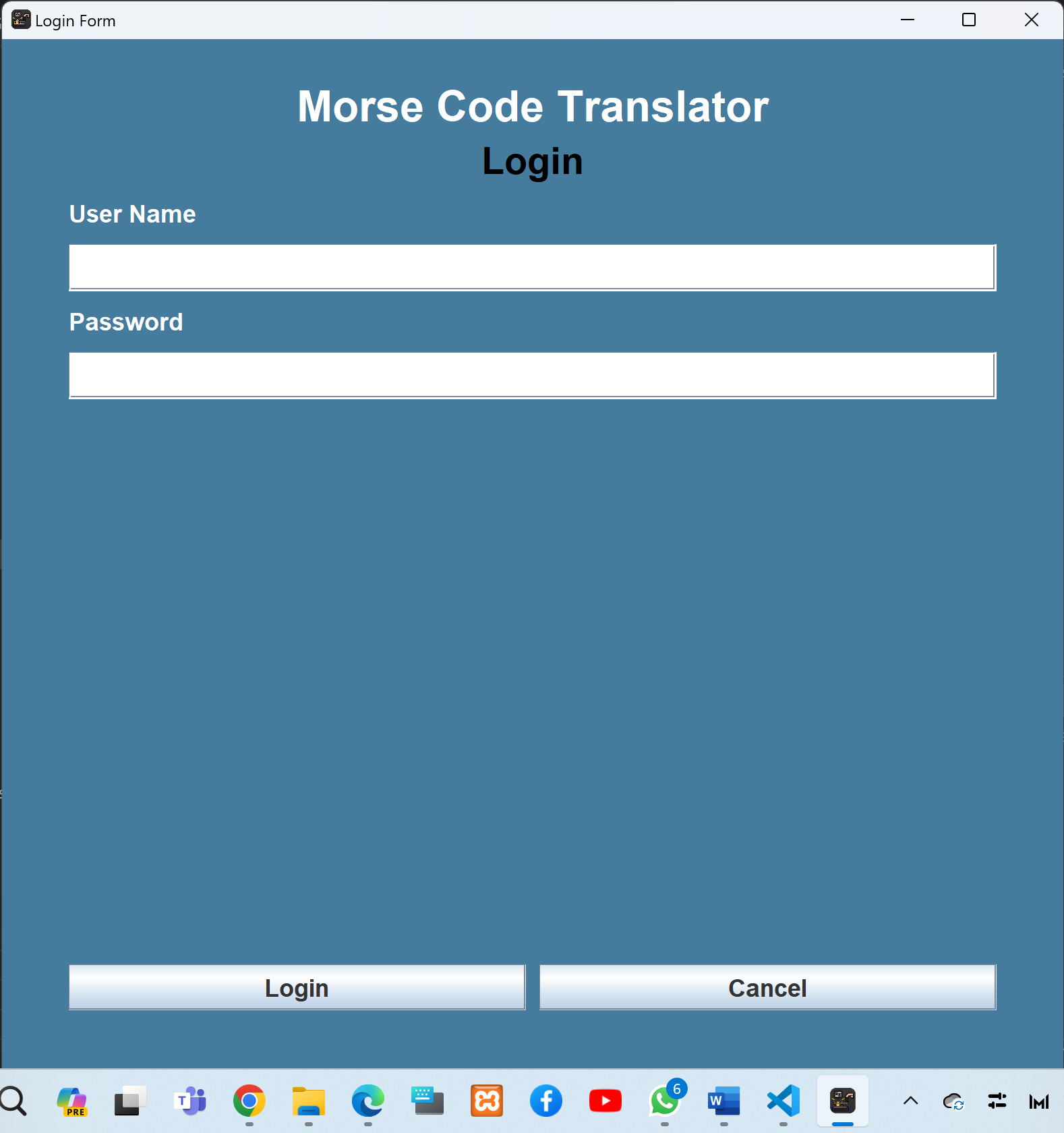
**3.9 DATA DICTIONARY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field name** | **Type** | **Constraints** | **Description** | **Table name** |
| User\_id | Int | Primary key | User id | history |
| History\_id | SERIAL | Required | History id | history |
| Text | String | Required | Entered text | history |
| Morse | String | Required | Morse Code Given | history |
| User\_id | SERIAL | Primary Key | User Id | users |
| Name | String | Required | Name | users |
| Username | String | Required | username | users |
| Password | String | Required | password | users |

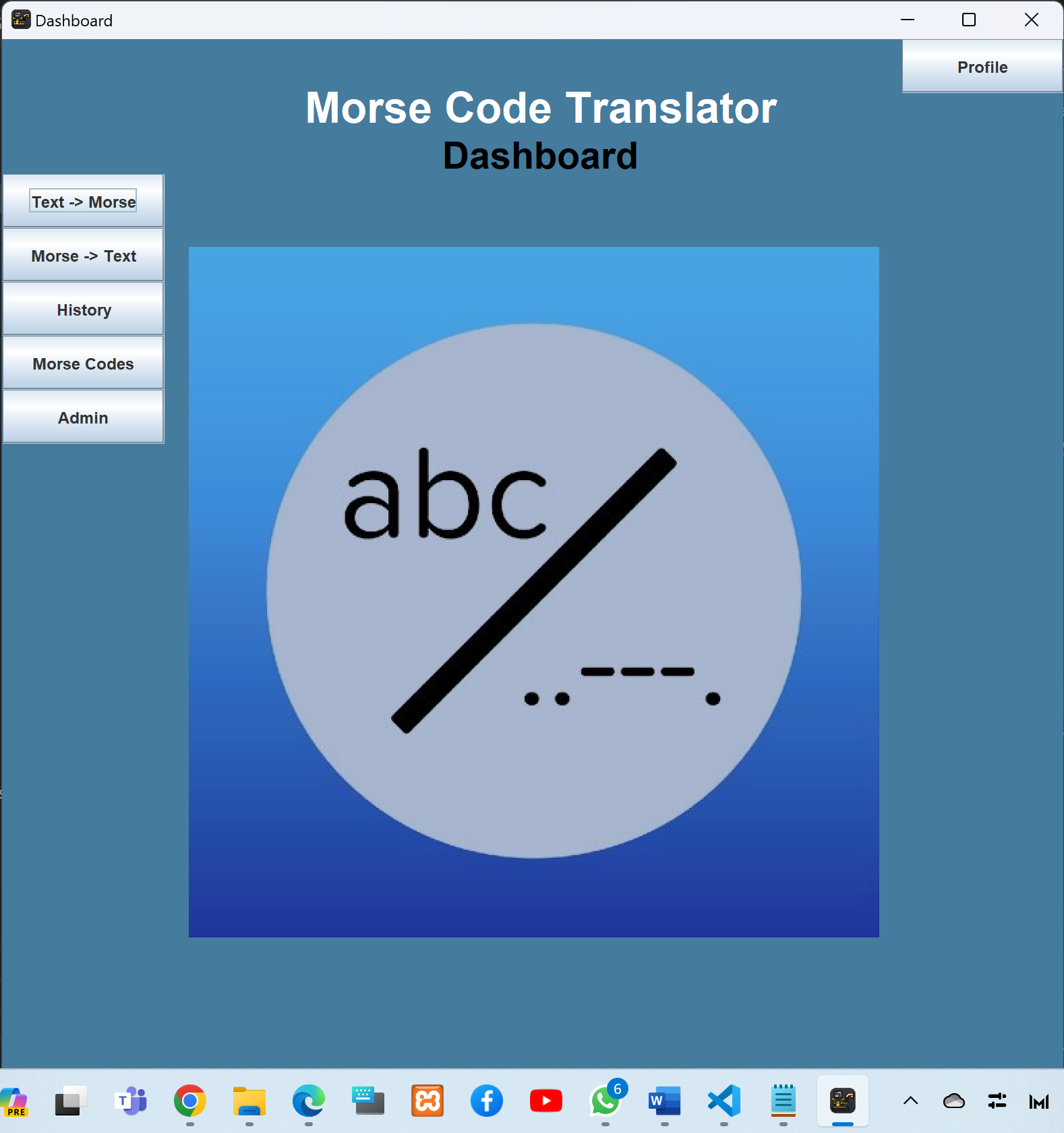
**3.10 SAMPLE SCREENS**

**Sign In**

**Login**

****

**Dashboard**



**Text-to-Morse**

A screenshot of a computer

Description automatically generated

**Morse-to-Text**

A screenshot of a computer

Description automatically generated

**Morse Codes**

A screenshot of a computer

Description automatically generated

Chapter 4

**Coding Sample Code**

**//Code for implementing Text to Morse Conversion**

import javax.sound.sampled.LineUnavailableException;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

public class TextToMorse extends JFrame implements KeyListener {

private TTMController morseCodeController;

private JTextArea textInputArea, morseCodeArea;

public TextToMorse(){

super("Morse Code Translator");

setSize(new Dimension(800, 900));

setResizable(true);

setLayout(null);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

getContentPane().setBackground(Color.decode("#457B9D"));

setLocationRelativeTo(null);

morseCodeController = new TTMController();

addGuiComponents();

}

private void addGuiComponents(){

// title label

ImageIcon icon = new ImageIcon("C:\\Users\\soham\\OneDrive\\Documents\\morse code project\\src\\images\\icon.png");

this.setIconImage(icon.getImage());

JLabel titleLabel = new JLabel("Morse Code Translator");

JLabel titleLabel1 = new JLabel("Text to Morse Code");

// changes the font size for the label and the font weight

titleLabel.setFont(new Font("Dialog", Font.BOLD, 32));

titleLabel1.setFont(new Font("Dialog", Font.BOLD, 28));

titleLabel.setForeground(Color.WHITE);

titleLabel1.setForeground(Color.BLACK);

titleLabel.setHorizontalAlignment(SwingConstants.CENTER);

titleLabel1.setHorizontalAlignment(SwingConstants.CENTER);

titleLabel.setBounds(0, 0, 800, 100);

titleLabel1.setBounds(50, 50, 700, 70);

JLabel textInputLabel = new JLabel("Text:");

textInputLabel.setFont(new Font("Dialog", Font.BOLD, 16));

textInputLabel.setForeground(Color.WHITE);

textInputLabel.setBounds(20, 100, 200, 30);

textInputArea = new JTextArea();

textInputArea.setFont(new Font("Dialog", Font.PLAIN, 18));

textInputArea.addKeyListener(this);

textInputArea.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

textInputArea.setLineWrap(true);

textInputArea.setWrapStyleWord(true);

JScrollPane textInputScroll = new JScrollPane(textInputArea);

textInputScroll.setBounds(20, 132, 744, 236);

JLabel morseCodeInputLabel = new JLabel("Morse Code:");

morseCodeInputLabel.setFont(new Font("Dialog", Font.PLAIN, 16));

morseCodeInputLabel.setForeground(Color.WHITE);

morseCodeInputLabel.setBounds(20, 390, 200, 30);

morseCodeArea = new JTextArea();

morseCodeArea.setFont(new Font("Dialog", Font.PLAIN, 18));

morseCodeArea.setEditable(false);

morseCodeArea.setLineWrap(true);

morseCodeArea.setWrapStyleWord(true);

morseCodeArea.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

JScrollPane morseCodeScroll = new JScrollPane(morseCodeArea);

morseCodeScroll.setBounds(20, 430, 744, 236);

JButton playSoundButton = new JButton("Play Sound");

playSoundButton.setBounds(270, 680, 100, 30);

playSoundButton.addActionListener(new ActionListener(){

@Override

public void actionPerformed(ActionEvent e){

playSoundButton.setEnabled(false);

Thread playMorseCodeThread = new Thread(new Runnable() {

@Override

public void run() {

try{

String[] morseCodeMessage = morseCodeArea.getText().split(" ");

morseCodeController.playSound(morseCodeMessage);

}catch(LineUnavailableException lineUnavailableException){

lineUnavailableException.printStackTrace();

}catch(InterruptedException interruptedException){

interruptedException.printStackTrace();

}finally{

playSoundButton.setEnabled(true);

}

}

});

playMorseCodeThread.start();

});

JButton btnreset = new JButton("Reset");

btnreset.setBounds(390, 680, 100, 30);

btnreset.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e) {

textInputArea.setText("");

morseCodeArea.setText("");

}

});

JButton backButton = new JButton("Back");

backButton.setBounds(20, 20, 80, 30);

backButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

SwingUtilities.invokeLater(() -> new DashBoards(""));

//dispose();

}

});

add(backButton);

add(titleLabel);

add(titleLabel1);

add(textInputLabel);

add(textInputScroll);

add(morseCodeInputLabel);

add(morseCodeScroll);

add(playSoundButton);

add(btnreset);

}

@Override

public void keyTyped(KeyEvent e) {

}

@Override

public void keyPressed(KeyEvent e) {

}

@Override

public void keyReleased(KeyEvent e) {

if(e.getKeyCode() != KeyEvent.VK\_SHIFT){

String inputText = textInputArea.getText();

morseCodeArea.setText(morseCodeController.translateToMorse(inputText));

}

}

}

Chapter 5

**Limitations Of The System**

1.User Interface Complexity:

Morse Code Translator's user interface design may be limited in terms of aesthetics and user experience, especially when compared to more sophisticated graphical user interfaces developed with advanced front-end technologies.

2.Scalability:

The project's architecture might face challenges when handling a large number of users or a significant volume of translation requests simultaneously. Scalability issues could arise in terms of database performance or server capacity.

3.Limited Error Handling:

The project may lack robust error handling mechanisms, leading to potential crashes or incorrect behaviour in response to unexpected inputs or situations.

4.Single-User Focus:

The project might be designed primarily for a single user environment. Collaborative features or simultaneous multi-user interactions might not be well-supported.

5.Limited Language Support:

The translator may focus solely on English or a specific set of languages, limiting its ability to handle translations for a broader range of languages.

6.Incomplete Feature Set:

Depending on the project scope, some desired features or functionalities (e.g., additional translation options, customization features) might be missing.

7.Documentation:

Limited or inadequate documentation may make it challenging for other developers or users to understand and extend the project.

Chapter 6

**PROPOSED ENHANCEMENTS**

Proposed enhancements for the Morse Code Translator Java project can include:

1.Multi-Language Support:

Expand the translator to support a broader range of languages, allowing users to translate text to and from Morse Code in languages beyond English.

2.Improved User Interface:

Enhance the graphical user interface (GUI) with modern design principles, user-friendly features, and responsive layouts to improve the overall user experience.

3.Customization Options: Provide users with customization options, such as adjusting the speed of Morse Code playback, choosing different sound effects, or selecting personalized themes for the application.

4.Collaborative Features:

Introduce collaborative features, enabling multiple users to work on translations simultaneously or share their translations with others.

5.Error Handling and Validation:

Strengthen error handling mechanisms to ensure robust handling of unexpected inputs and improve the overall stability of the application.

6.User Preferences:

Allow users to save their preferences, such as favourite translations, custom Morse Code mappings, or preferred settings, creating a more personalized experience.

7.Enhanced Chart:

Improve the chart functionality by providing additional information, such as historical Morse Code usage, interesting facts, or interactive elements for a more engaging learning experience.

8.Accessibility:

Ensure that the application is accessible to users with disabilities by implementing features such as screen reader compatibility, keyboard navigation, and adherence to accessibility standards.

Chapter 7

**CONCLUSION**

In conclusion, the Morse Code Translator project in Java successfully achieves its primary objective of translating text to Morse Code and vice versa. The project employs Java Swing for the frontend, providing a user-friendly interface, and utilizes PostgreSQL as the backend database with JDBC drivers for data management. The key functionalities include user registration, login, translation of text and Morse Code, Translation history and a chart displaying Morse codes for letters, numbers, and symbols.

With a focus on Java Swing for the GUI and PostgreSQL for database management, the project ensures a robust and interactive user experience. The multi-language support, offline mode, and customization options enhance the project's versatility.

While the project fulfils its primary objectives, there are opportunities for future enhancements. Proposed improvements include multi-language support, an improved user interface, offline functionality, customization options, collaborative features, error handling, internationalization, user preferences, an enhanced chart, accessibility, and security measures.

In summary, the Morse Code Translator project showcases the successful integration of Java and PostgreSQL, offering a functional and extensible application for translating text to and from Morse Code. The proposed enhancements aim to further elevate the project's capabilities, making it more user-centric, feature-rich, and adaptable to diverse user needs.

Chapter 8

**BIBLIOGRAPHY**

Reference Book:

Java: The Complete Reference by Herbert Schildt

Websites:

1.www.udemy.com

2.www.youtube.com

3. https://morsecode.world/international/translator.html