**ActNow - COMMUNITY SERVICE CONNECTOR APP**

**CS19611 – Mobile Application Development Laboratory**

***Submitted by***

**SREENIDHI K (2116220701285)**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

# COMPUTER SCIENCE AND ENGINEERING



**RAJALAKSHMI ENGINEERING COLLEGE ANNA UNIVERSITY, CHENNAI**

# MAY 2025

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

**BONAFIDE CERTIFICATE**

Certified that this Project titled **“ActNow - Community Service Connector App”** is the bonafide work of **“ SREENIDHI K (2116220701285)”** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

# SIGNATURE

## Dr. Duraimurugan N., M.Tech., Ph.D.,

## SUPERVISOR

Professor

Department of Computer Science and Engineering,

Rajalakshmi Engineering College, Chennai-602 105.

Submitted to Project Viva-Voce Examination held on

**Internal Examiner External Examiner**

# ABSTRACT

In today's rapidly evolving society, a significant number of individuals express a desire to contribute to social causes but struggle to find opportunities that align with their skills, interests, and availability. Simultaneously, many non-profit organizations, NGOs, and local initiatives face challenges in reaching and mobilizing potential volunteers effectively. To address this gap, our project titled **"ActNow - Community Service Connector App"** presents a mobile-based solution aimed at seamlessly connecting volunteers with relevant service opportunities in their area.ActNow serves as a centralized platform where users can sign up either as Volunteers or Organizations. Volunteers can explore and register for community service events based on location, category, and skill preference, while organizations can post events and manage participant engagement. The application facilitates easy communication, provides real-time updates, and helps volunteers maintain a digital log of their contributions. The user-friendly interface ensures quick access to essential functionalities and encourages participation in social good.Developed using Android Studio with Kotlin and integrated with Firebase for data management, ActNow is lightweight, responsive, and designed for real-world deployment. The app aims to promote civic engagement, enhance the reach of charitable initiatives, and streamline the coordination between those who want to help and those in need of help. This project not only showcases our technical skills but also reflects our commitment to social responsibility.

# 

# 

# 

# ACKNOWLEDGMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavor to put forth this report. Our sincere thanks to our Chairman **Mr. S. MEGANATHAN, B.E, F.I.E.**, our Vice Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.,** and our respected Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN**, **Ph.D.,** for providing us with the requisite infrastructure and sincere endeavoring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. P. KUMAR, M.E., Ph.D.**, Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work. We convey our sincere and deepest gratitude to our Project Coordinator**, Mr. N. DURAI MURUGAN, M.Tech., Ph.D.,** Professor Department of Computer Science and Engineering for his useful tips during our review to build our project and for his valuable guidance throughout the course of the project.

**SREENIDHI K** **2116220701285**

**TABLE OF CONTENTS**

**CHAPTER NO. TITLE PAGE NO.**

|  |  |
| --- | --- |
| **ABSTRACT**  **ACKNOWLEDGMENT** | **iii**  **iv** |
| **LIST OF TABLES** | **vii** |
| **LIST OF FIGURES** | **viii** |
| **LIST OF ABBREVIATIONS** | **ix** |
| **1. INTRODUCTION** | **11** |
| 1.1 GENERAL | 11 |
| 1.2 OBJECTIVES | 11 |
| 1.3 EXISTING SYSTEM | 12 |
| **2. LITERATURE SURVEY** | **13** |
| **3. PROPOSED SYSTEM** | **16** |
| 3.1 GENERAL | 16 |
| 3.2 SYSTEM ARCHITECTURE DIAGRAM | 17 |
| 3.3 DEVELOPMENT ENVIRONMENT | 19 |
| 3.3.1 HARDWARE REQUIREMENTS | 19 |
| 3.3.2 SOFTWARE REQUIREMENTS | 19 |
| 3.4 DESIGN THE ENTIRE SYSTEM | 20 |
| 3.4.1 ACTIVITY DIAGRAM | 20 |
| 3.4.2 DATA FLOW DIAGRAM | 21 |
| 3.5 STATISTICAL ANALYSIS | 23 |
| **4. MODULE DESCRIPTION** | **25** |

* 1. [SYSTEM ARCHITECTURE25](#_TOC_250016)

4.1.1 [USER INTERFACE DESIGN 25](#_TOC_250015)

4.1.2 BACK END INFRASTRUCTURE [26](#_TOC_250014)

4.2 DATA COLLECTION & PREPROCESSING 26

4.2.1 DATASET & DATA LABELLING 26

* + 1. DATA PREPROCESSING 26

4.2.3 [FEATURE EXTRACTION 27](#_TOC_250009)

4.2.4 [CLASSIFICATION & MODEL SELECTION 27](#_TOC_250008)

* + 1. [PERFORMANCE EVALUATION 27](#_TOC_250007)
    2. [MODEL DEPLOYMENT 27](#_TOC_250006)

4.2.7 [CENTRALIZED SERVER & DATABASE 27](#_TOC_250005)

4.3 SYSTEM WORKFLOW 28

4.3.1 USER INTERACTION 28

4.3.2 ICD CODE PREDICTION 28

4.3.3 RESULT DISPLAY & REPORTING 28

4.3.4 CONTINUOUS MODEL IMPROVEMENT 28

1. IMPLEMENTATIONS AND RESULTS 29
   1. [IMPLEMENTATION 29](#_TOC_250004)
   2. [OUTPUT SCREENSHOTS 29](#_TOC_250003)
2. CONCLUSION AND FUTURE ENHANCEMENT 34
   1. [CONCLUSION 34](#_TOC_250002)
   2. [FUTURE ENHANCEMENT 34](#_TOC_250001)

[REFERENCES 36](#_TOC_250000)

## LIST OF TABLES

**TABLE NO TITLE PAGE NO**

3.1 HARDWARE REQUIREMENTS 19

3.2 SOFTWARE REQUIREMENTS 19

3.3 COMPARISON OF FEATURES 23

## LIST OF FIGURES

**FIGURE NO TITLE** **PAGE NO**

3.1 SYSTEM ARCHITECTURE 18

3.2 ACTIVITY DIAGRAM 20

3.3 DFD DIAGRAM 22

4.1 SEQUENCE DIAGRAM 25

5.1 DATASET FOR TRAINING 31

5.2 PERFORMANCE EVALUATION METRICS 31

5.3 CONFUSION MATRIX 32

5.4 BATCH TESTING 32

5.5 ICD CODE PREDICTION WEB INTERFACE 33

5.6 PREDICTION RESULT 33

## 

## LIST OF ABBREVIATIONS

**S. No ABBR Expansion**

1 AI Artificial Intelligence

2` API Application Programming Interface

1. ICD International Classification of Diseases
2. NLP Natural Language Processing
3. TF-IDF Term Frequency - Inverse Document Frequency
4. UI User Interface
5. LR Logistic Regression
6. DFD Data Flow Diagram
7. ML Machine Learning
8. RF Random Forest
9. SQL Structure Query Language
10. SVM Support Vector Machine

**CHAPTER 1**

**INTRODUCTION**

* 1. **GENERAL**

Volunteering is a vital component of social development and community growth. Despite the willingness of many individuals to contribute to social causes, there exists a lack of proper channels to connect them with organizations in need of assistance. This disconnect leads to underutilized human potential and inefficiencies in organizing community-based activities.**ActNow - Community Service Connector App** is designed to bridge this gap by providing a mobile platform that facilitates collaboration between volunteers and organizations. The app simplifies the process of discovering, joining, and managing social service events. Volunteers can sign up, browse events by interest or location, and register with a single click. Organizations, in turn, can post events, manage volunteer participation, and track event effectiveness.This mobile application aims to enhance civic engagement, ensure smoother operations for social organizations, and promote transparency and accountability in volunteer efforts. By digitizing the volunteer management process, ActNow aspires to become a valuable tool for social impact.

# OBJECTIVE

The primary objective of this project is to design and implement a mobile application that:

* Provides a centralized platform for volunteers and organizations.
* Allows volunteers to discover and join social events based on location, date, and interest.
* Enables organizations to post, edit, and manage volunteer events.
* Tracks volunteer participation and service hours.
* Encourages consistent civic engagement through timely notifications and an intuitive interface.

# 

# EXISTING SYSTEM

In the current landscape, volunteer coordination is often conducted through manual means such as WhatsApp groups, social media pages, and spreadsheets. These methods are disorganized, lack personalization, and are difficult to scale. Additionally, volunteers often miss opportunities due to lack of timely information, while organizations struggle to reach the right audience efficiently.Some existing platforms offer event listings or social cause networking, but they are either overly complex, not tailored for volunteer-based community service, or lack proper integration with mobile devices. This creates a need for a purpose-built app that focuses solely on connecting service-minded individuals with credible, real-time opportunities.

# CHAPTER 2 LITERATURE SURVEY

# A literature survey is essential in understanding the scope of existing technologies and methodologies that address the problem of volunteer management and social engagement. The research reveals that although several applications and platforms facilitate community interaction, very few are focused specifically on structured, localized volunteer coordination and event participation.

# 

# 2.1 Existing Applications

# FacebookGroups/WhatsApp Groups Many volunteer-based organizations currently use informal platforms such as Facebook groups or WhatsApp to manage events. These methods are manual, lack structure, and do not provide analytics, notifications, or participation tracking.

# GiveGab/VolunteerMatch These platforms offer web-based volunteering networks. However, they are primarily U.S.-based, often too complex for local NGOs, and not mobile-first. Moreover, they lack real-time interaction and event management features suited for smaller communities.

# GoogleForms+Sheets A commonly used alternative is collecting registrations through Google Forms and managing lists in Google Sheets. This requires manual follow-up and is not integrated with a notification or reminder system, leading to participant drop-off and inefficiencies.

# 2.2 Limitations of Existing Systems

# No personalized matching of volunteer interests and skills with events.

# Lack of real-time updates or reminders about event changes.

# Absence of a user-friendly mobile interface for on-the-go volunteering.

# No centralized database for tracking volunteer hours or generating participation records.

# Difficulty for NGOs to manage recurring volunteers or edit/update events dynamically.

# 2.3 Need for the Proposed System

# The proposed mobile application ActNow fills this gap by providing:

# A simplified, intuitive interface for both volunteers and organizers.

# Real-time event listing, registration, and notifications.

# Role-based login for separate functionalities.

# Local storage-based participation logs with the potential for future integration with Firebase for cloud syncing.

# An inclusive and community-focused digital platform to streamline social contribution.

**CHAPTER 3**

**PROPOSED SYSTEM**

* 1. **GENERAL**

The ActNow - Community Service Connector App is a mobile application designed to bridge the gap between volunteers and organizations. It simplifies the process of finding and managing social service activities through an intuitive, mobile-first interface. The proposed system aims to address the limitations of existing solutions by providing a centralized, easy-to-use, and real-time platform for both event organizers and volunteers.

# SYSTEM ARCHITECTURE DIAGRAM

# The application follows a modular, role-based structure:

# MainActivity: A login and role-selection screen where users enter their name, email, and select their role as either "Volunteer" or "Organization".

# VolunteerHomeActivity: Displays a list of community service events. Volunteers can browse, register, and view their registered events.

# OrganizationHomeActivity: Allows organizations to add, manage, and delete events.

# EventRepository: A central data store that holds shared events between volunteers and organizations.

# The application is developed using Kotlin and Jetpack Compose within Android Studio, following clean UI practices and structured navigation.

# KEY MODULES

 **Role-based Login**  
Users select whether they are a Volunteer or an Organization. Based on the role, they are redirected to the appropriate home screen.

**Event Creation and Management**  
Organizations can add events by entering the event name, description, and location. Events are dynamically stored and can be deleted if needed.

 **Volunteer Event Registration**  
Volunteers can view all events and register with a single click. Registered events are stored locally and can be reviewed via a dedicated "Registered Events" section.

 **UI & Styling Enhancements**  
The app includes colorful themes, rounded card corners, material components, scrollable views, and toast messages for interactivity.

# 3.3 ADVANTAGES OF PROPOSED SYSTEM

# Simplifies volunteer-event discovery.

# Offers a mobile-friendly, modern user interface.

# Local storage allows data retention during session.

# Encourages civic engagement through a rewarding user experience.

# Acts as a ready prototype for future Firebase or cloud integration.

# 

**Fig 3.1: System Architecture**

# DEVELOPMENTAL ENVIRONMENT

* + 1. **HARDWARE REQUIREMENTS**

The hardware specifications serve as the foundation for implementing and running the mobile application smoothly. A clear understanding of hardware requirements is crucial to ensure compatibility, responsiveness, and stability during both development and usage. These specifications help guide design decisions and ensure reliable performance on real Android devices and emulators.

**Table 3.1 Hardware Requirements**

|  |  |
| --- | --- |
| **COMPONENTS** | **SPECIFICATION** |
| PROCESSOR | Intel i5 or above (recommended) |
| RAM | 8 GB RAM OR Higher |
| HARD DISK | Minimum 4 GB free space |
| DISPLAY | 1280 x 720 resolution or higher |
| SMARTPHONE | Android 7.0 (API 24) and above |

# SOFTWARE REQUIREMENTS

# The software requirements define the essential components required for the design, development, testing, and deployment of the ActNow - Community Service Connector App. These specifications ensure that developers have the necessary tools and environments for efficient development. They also help in planning, cost estimation, task allocation, and version control throughout the development lifecycle.

**Table 3.2 Software Requirements**

| SOFTWARE COMPONENTS | DESCRIPTION |
| --- | --- |
| Operating System | Windows 10 / macOS / Linux |
| IDE | Android Studio (Electric Eel / later) |
| Programming Language | Kotlin (with Jetpack Compose) |
| Design Tool | XML for UI (Jetpack Compose UI Toolkit) |
| Emulator / Device | Android Emulator or physical Android phone |
| Database (Optional) | Firebase (can be added in future) |

# DESIGN OF THE ENTIRE SYSTEM

# 

# ACTIVITY DIAGRAM

The Activity Diagram models the logical flow of the application from launch to interaction. It shows how a user begins by opening the app, selects their role, and then proceeds to either volunteer-related or organization-related activities. This diagram helps visualize the user's journey and how different paths diverge based on role.

**Key Activities:**

* Launch the app.
* Select role: Volunteer or Organization.
* Volunteer:
  + View list of events.
  + Register for events.
* Organization:
  + Add new event.
  + Delete existing events.

The flow ends once the user completes their interaction or exits the app.

# 

**Fig 3.2: Activity Diagram**

# 3.4.2 DATA FLOW DIAGRAM

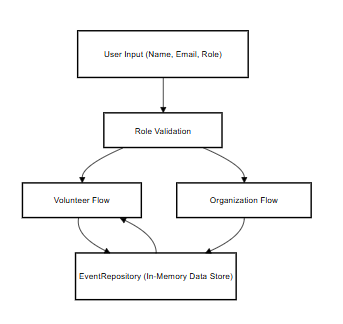
The Data Flow Diagram (DFD) describes how data moves within the system at a high level. It identifies the main processes involved in handling user input and shows the interaction with the data store (in this case, the EventRepository).

**Components:**

* **User Input**: The user provides their name, email, and selects their role.
* **Role Check**: Determines whether the user is a volunteer or organization.
* **Volunteer Process**:
  + Fetches and displays events.
  + Sends registration info to the data store.
* **Organization Process**:
  + Allows event creation or deletion.
  + Sends updates to the shared event store.
* **EventRepository**:
  + Acts as a shared in-memory storage that both user types interact with.

This flow ensures data consistency and coordination between volunteers and organizations, even without a backend database.

.



**Fig 3.3:Data Flow Diagram**

# CHAPTER 4 MODULE DESCRIPTION

The **ActNow** application is divided into well-defined modules, each responsible for specific functionality based on the user's role: **Volunteer** or **Organization**. These modules ensure a clean separation of concerns, ease of maintenance, and better user experience.

# SYSTEM ARCHITECTURE

# USER INTERFACE DESIGN

# The sequence diagram (Fig 4.1) depicts the process of predicting ICD codes. It starts with the user providing the input — a free-text description of clinical symptoms — through a web-based form.

# The input text is passed to the Flask backend server, where it undergoes text preprocessing operations such as cleaning, stopword removal, lemmatization, and TF-IDF feature extraction.

# PlantUML diagram

# Fig 4.1: SEQUENCE DIAGRAM

# BACK END INFRASTRUCTURE

# The backend infrastructure for the sequence diagram comprises a centralized database for managing datasets, storing raw and labeled clinical data for preprocessing, training, and testing. A machine learning framework such as Scikit-learn is used to implement and train models like Logistic Regression and Random Forest.

# The Flask framework with a WSGI server handles API requests and application logic, enabling seamless interaction between users and the backend for predictions and result delivery. The centralized server manages communication between the web interface and the prediction engine, ensuring secure and efficient processing.

# 4.2 DATA COLLECTION AND PREPROCESSING

## 4.2.1 Dataset and Data Labelling

## Labeled datasets are collected, including symptom descriptions paired with corresponding ICD codes. Accurate labeling differentiates between various clinical conditions and ensures effective model training.

**4.2.2. Data Preprocessing**

The raw dataset undergoes extensive preprocessing, which includes:

* Data Cleaning: Removal of inconsistent, redundant, or irrelevant data
* Missing Value Replacement: Imputation techniques to handle incomplete entries
* Text Normalization: Lowercasing, stopword removal, tokenization, and lemmatization to standardize text data.

**4.2.3 Feature Selection**

Advanced techniques are used to ensure relevant and optimized feature sets:

* **TF-IDF Vectorization**: Converts text into numerical feature vectors based on term frequency and inverse document frequency
* **Dimensionality Reduction**: Reducing data complexity while retaining critical features for accurate classification

**4.2.4 Classification and Model Selection**

Multiple models are evaluated for classification, such as:

* Logistic Regression: For multi-class text classification tasks
* Random Forest: For general-purpose and robust classification
* Model Selection: The best-performing model (Logistic Regression or Random Forest) is selected based on accuracy and adaptability in ICD code prediction

**4.2.5 Performance Evaluation and Optimization**

Model performance is assessed using metrics such as accuracy, confusion matrix, and classificationreport.The chosen model undergoes iterative tuning and optimization to maximize prediction accuracy and reduce misclassifications.

**4.2.6 Model Deployment**

The optimized model is deployed via a Flask-based system, enabling real-time ICD code predictions by processing live user inputs through the web application interface.

**4.2.7 Centralized Server and Database**

All data, including training results, predictions, and evaluations, is securely stored in acentralizeddatabase.The server handles communication between the web interface and the machine learning model, ensuring secure data processing.

**4.3 SYSTEM WORK FLOW**

**4.3.1 User Interaction:**

Users initiate the prediction process by submitting symptom descriptions through the webapplicationinterface.The system processes these inputs and prepares them for classification.

**4.3.2 ICD Code Prediction:**

Advanced machine learning techniques (Logistic Regression and Random Forest) are applied to identify patterns associated with various clinical conditions.  
The system analyzes the symptom description and predicts the most relevant ICD code.

**4.3.3 Result Display and Reporting:**

Once the ICD code is predicted, it is displayed to the user in real time via the web interface.The result provides healthcare professionals with accurate and standardized diagnostic codes for documentation purposes.

**4.3.4** **Continuous Learning and Improvement:**

The system can continuously update and retrain its machine learning models based on newsymptompatternsanduserFeedback. This ensures the model remains accurate and adaptable to evolving clinical terminologies and emerging healthcare needs.

# CHAPTER 5 IMPLEMENTATION AND RESULTS

# IMPLEMENTATION

# The project is developed and deployed using a robust technology stack, consisting of Python for backend processing, Flask as the web framework, and SQLite for simple and effective database management. The frontend is designed using HTML and CSS, ensuring a responsive and user-friendly interface.

# For ICD code prediction, the system leverages machine learning algorithms, including Logistic Regression and Random Forest, to analyze clinical symptom descriptions and predict the corresponding ICD codes accurately. The implementation involves setting up an intuitive web interface, allowing healthcare professionals to submit symptom descriptions for ICD code assignment.

# The backend server efficiently processes user inputs, performs text preprocessing (including cleaning, lemmatization, and TF-IDF vectorization), and passes the processed data to the trained machine learning model for prediction. The predicted ICD code is displayed to the user in real-time through the web interface.

# Additionally, a data management system is implemented, enabling storage and retrieval of symptom–ICD code pairs, model outputs, and evaluation metrics. The system is designed to be adaptable and scalable, allowing future updates with new clinical terms and improved prediction models. Continuous learning and model retraining ensure improved accuracy and reliability over time.

# OUTPUT SCREENSHOTS

# The project implementation is structured into modules and is demonstrated through a series of outputs:

# Fig 5.1 depicts the project’s overall system workflow, highlighting the seamless integration of machine learning for predictive analysis. It demonstrates the clear flow from data collection to real-time ICD code prediction through the web interface, ensuring usability and accessibility.

# Fig 5.2 showcases the machine learning model development and training process. It illustrates how clinical symptom descriptions are preprocessed, feature-extracted using TF-IDF, and fed into classifiers like Logistic Regression and Random Forest for model training and evaluation.

# Fig 5.3 compares the confusion matrices of multiple classifiers. It highlights the models' performance in predicting ICD codes, showing trends in accuracy and misclassifications. This visualization aids in selecting the best-performing algorithm for deployment.

# Fig 5.4 demonstrates the integration of the trained machine learning model within a Flask web application. The interface accepts user input of symptom descriptions and returns the predicted ICD code in real-time, ensuring effective deployment for practical healthcare use cases.

# Fig 5.5 illustrates the Flask web application’s user interface designed for ICD code prediction. The interface is simple and user-friendly, allowing healthcare professionals to input symptoms and receive ICD predictions quickly and efficiently.

# Fig 5.6 presents the prediction result page of the Flask web application. It displays the predicted ICD code along with a confidence score or classification outcome. The page includes a "Go Back" button for easy navigation, offering a seamless and smooth user experience.

# 

Fig 5.1 Dataset for Training

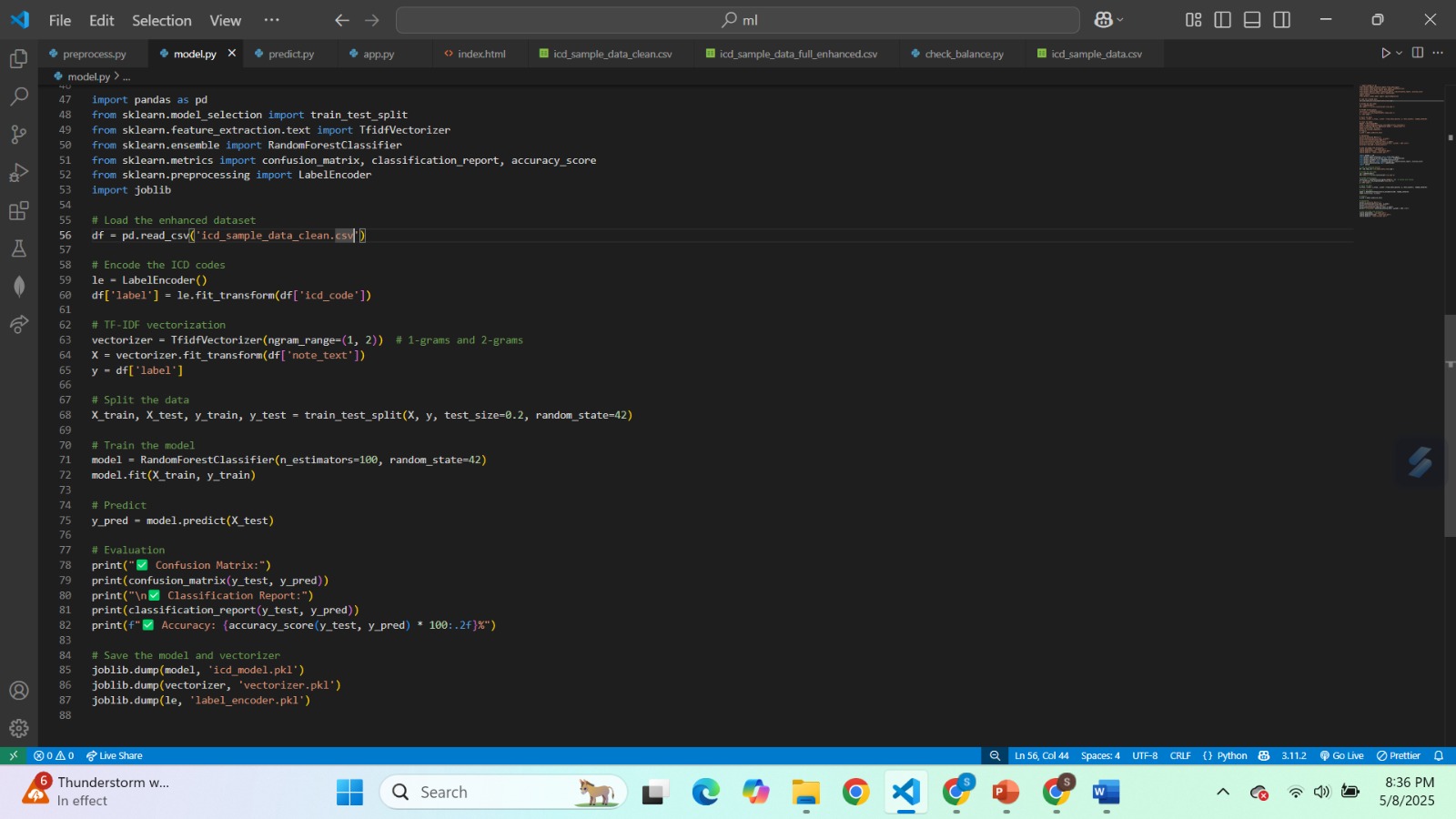


Fig 5.2 Performance Evaluation & Optimization

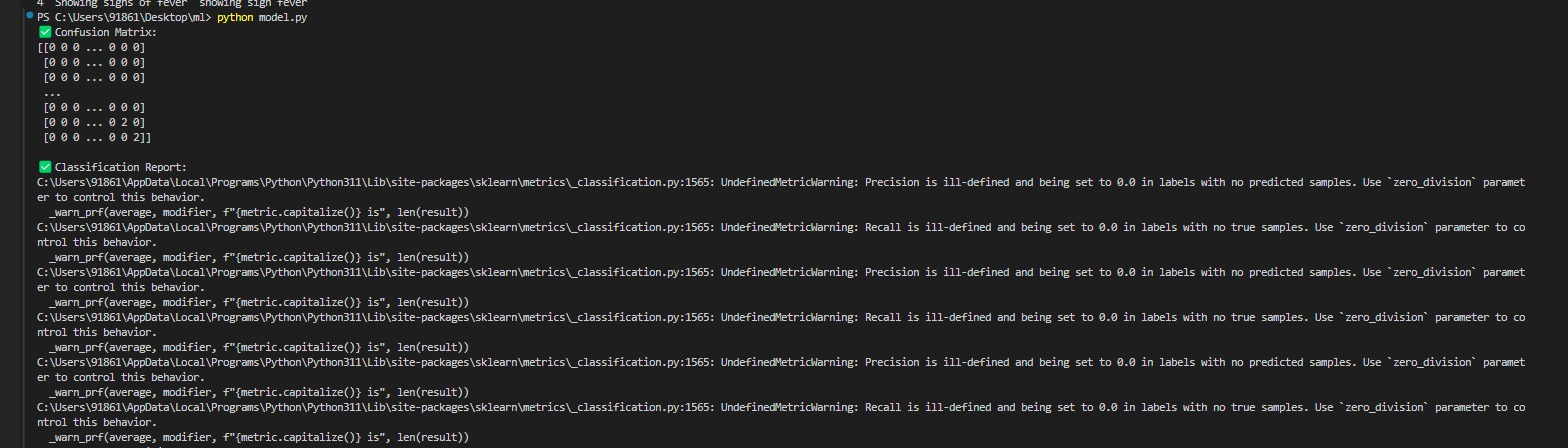


Fig 5.3 Confusion Matrix

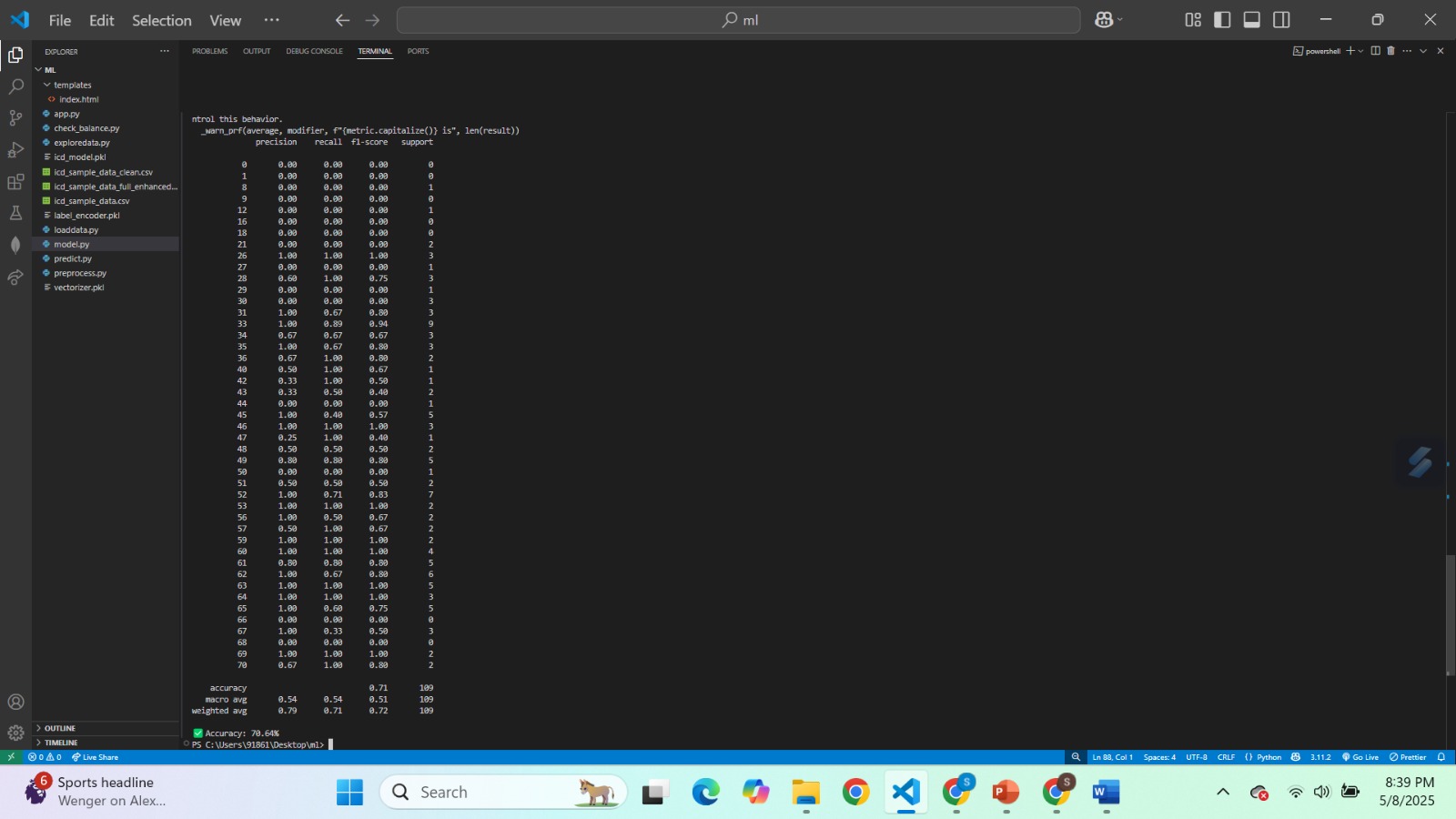


Fig 5.4 Batch Testing

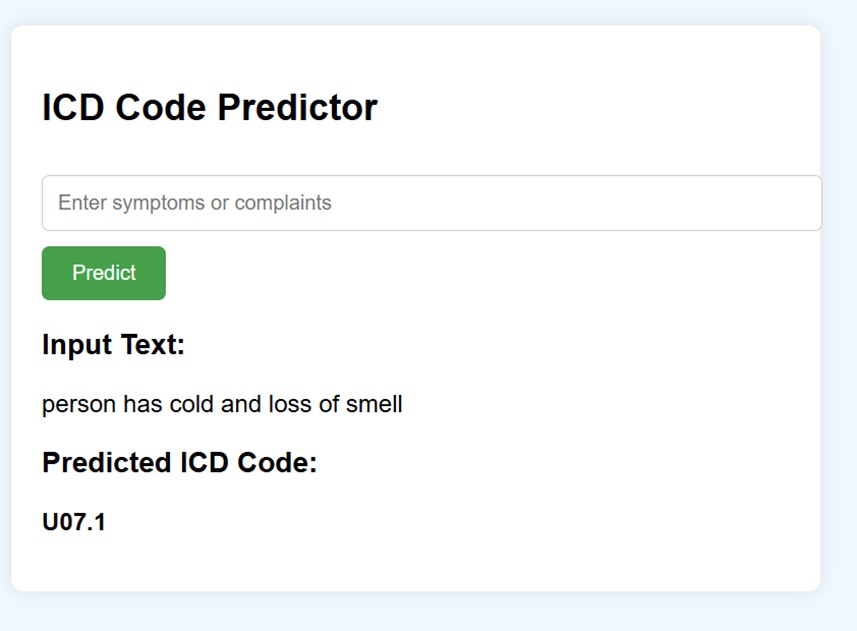
****

Fig 5.5 ICD Prediction from medical notes

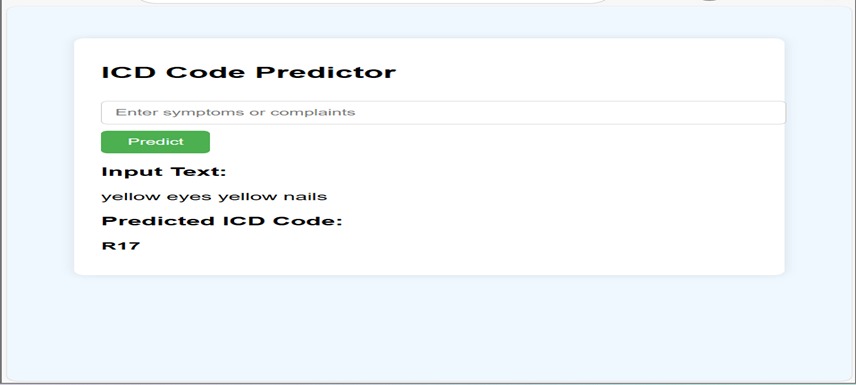


Fig 5.6 Prediction result

**CHAPTER 6**

**CONCLUSION AND FUTURE ENHANCEMENT**

# CONCLUSION

The proposed system leverages **machine learning** and **natural language processing (NLP)** technologies to create an innovative solution for automating **ICD code prediction** from clinical symptom descriptions, effectively addressing challenges in manual medical coding with enhanced accuracy, efficiency, and reliability.By analyzing a broad set of clinical text features, machine learning enables precise prediction of appropriate ICD codes, adapting to varying patterns and terminology in clinical narratives. The use of techniques such as **TF-IDF vectorization**, **Logistic Regression**, and **Random Forest** ensures robust feature extraction and accurate classification, supporting healthcare professionals in streamlining the coding process.The system’s user-friendly, **Flask-based web application** allows seamless deployment, providing healthcare providers with a real-time tool to assign ICD codes efficiently. By automating the coding process, the platform reduces manual workload, minimizes errors, and enhances the quality and consistency of healthcare documentation.

# FUTURE ENHANCEMENT

Future enhancements for this project could include integrating **deep learning models** such as **Convolutional Neural Networks (CNNs)** and **transformers like BERT** for more advanced and context-aware text analysis of complex clinical symptom descriptions. The use of **pre-trained biomedical language models** can further improve the system’s ability to understand domain-specific medical terminologies, thereby enhancing prediction accuracy.Incorporating **transfer learning** techniques can enable the system to adapt quickly to new clinical datasets and evolving medical terminologies without extensive retraining.