NEW SUMMIT COLLEGE

Affiliated To

Tribhuvan University

Institute of Science and Technology

A Final Year Project Report On

" MeetMyTutor "

Submitted to

Department of Computer Science and Information Technology

New Summit College

In partial fulfillment of the requirement of the Bachelor Degree in

Computer Science and Information Technology

Under The Supervision of **Indra Chaudhary**

Submitted By:

Kamal Rai [28512/078]

Miraj Tamang [28520/078]

Saraswoti Khadka [28537/078]

June, 2025

Acknowledgement

We would like to take this opportunity to express our sincere gratitude to all those who have supported and guided us throughout the ongoing development of our project, MeetMyTutor. Our heartfelt thanks go to our respected supervisor, **Mr. Indra Chaudhary**, for his continuous guidance, encouragement, and insightful feedback. His expertise and support have been invaluable in shaping the direction of this project, and we are truly grateful for the time and effort he continues to invest in our work.

We are also thankful **to Mr. Chok Raj Dawadi**, Principal of New Summit College, for providing us with the opportunity and necessary resources to work on this project in a positive and motivating environment. We would like to extend our appreciation to the faculty members and staff of the CSIT Department at New Summit College for their encouragement and support throughout the project's development phase. Finally, we are grateful to our friends and peers for their feedback, cooperation, and constant motivation, which have helped us move forward with confidence and determination.

Yours Sincerely,
Kamal Rai (28512/078)
Miraj Tamang (28520/078)
Saraswoti Khadka (28537/078)

Abstract

MeetMyTutor is an intelligent tutor management system designed to help students find the nearest available tutors efficiently. The system uses Dijkstra's algorithm to calculate the shortest distance between students and tutors based on longitude and latitude, ensuring accurate and location-based recommendations. Students can register, search for tutors by subject, level, time and price, and request sessions directly through the platform. Tutors can manage session requests by accepting, rejecting, or rescheduling them as needed.

An integrated admin panel manages users, verifies tutor credentials, and oversees system operations. By combining location intelligence with digital management, MeetMyTutor simplifies the process of finding local tutors and enhances accessibility to personalized learning.

Keywords: *MeetMyTutor, Tutor Management, In-Person Tutoring, Session Booking, Price Negotiation*

Table of Contents

Acknowledgement i
Abstract ii
List of Figures v
List of Tables vi
List of Abbreviations vii
Chapter 1: Introduction 1
1.1 Introduction 1
1.2 Problem Statement 1
1.3 Objectives 1
1.4 Scope and Limitations 1
1.4.1 Scope: 1
1.4.2 Limitations: 2
1.5 Methodology 2
1.6 Report Organizations 3
Chapter 2: Background Study and Literature Review 4
2.1 Background Study 4
2.2 Literature Review 4
Chapter 3: System Analysis 6
3.1 Requirement Analysis 6
3.1.1 Functional Requirements 6
3.1.2 Non-Functional Requirements 7
3.2 Feasibility Study 8
3.2.1 Technical Feasibility 8
3.2.2 Operational Feasibility 8
3.2.3 Economic Feasibility 9
3.2.4 Schedule Feasibility 9
Chapter 4: System Design 10
4.1 System Design 10
4.1.1 System Architecture 10
4.1.2 Class Diagram 11
4.1.3 Sequence Diagram 12
4.1.4 Activity Diagram 13

4.2 Descriptions of Algorithm 13
Chapter 5: Implementation and Testing 15
5.1 Implementation 15
5.1.1 Tools Used 15
5.1.2 Implementation Details of Modules 15
5.2 Testing 16
5.2.1 Unit Testing 16
Chapter 6: Conclusion 20
6.1 Conclusion 20
6.2 Future Recommendations 20
References 21

List of Figures

Figure 1.1: Iterative Waterfall Model 2

Figure 3.1: Use Case Diagram 7

Figure 4.1: System Architecture 10

Figure 4.2: Class Diagram..... 11

Figure 4.3: Sequence Diagram..... 12

Figure 4.4: Activity Diagram 13

List of Tables

Table 1: Gantt Chart 9

Table 2: Unit Testing 16

List of Abbreviations

AI: Artificial Intelligence

API: Application Programming Interface

CSS: Cascading Style Sheet

Django: Python-based Web Framework

HTML: Hyper Text Markup Language

IDE: Integrated Development Environment

JS: JavaScript

PostgreSQL: Object-Relational Database Management System

Python: Programming Language

Chapter 1: Introduction

1.1 Introduction

MeetMyTutor is a proposed centralized platform that bridges the gap between students seeking academic support and tutors offering in-person services. Unlike online tutoring systems, this platform focuses on facilitating offline, face-to-face sessions, while digitizing all processes leading up to the physical class. The system will allow students to browse tutor profiles, select based on location, subject, availability, and price, and request sessions easily. Tutors, in turn, will manage their profiles, respond to session requests, and handle scheduling. The platform will also include session extension logic, and a review mechanism to maintain accountability and quality service.

1.2 Problem Statement

Students and parents struggle to find qualified tutors in their immediate locality. This forces them to waste time on lengthy searches or settle for distant tutors, leading to increased travel time and cost. Most existing tutor management platforms operate with fixed pricing models that restrict students from choosing tutors aligned with their budgets and specific learning needs. This inflexibility limits opportunities for negotiation and personalized tutoring arrangements. Additionally, the management of tutoring sessions including scheduling, and feedback is often manual and uncoordinated. This leads to confusion, miscommunication, and reduced accountability, ultimately affecting the quality and reliability of tutoring services.

1.3 Objectives

- To develop a location-based matching system for discovering the nearest available tutors
- To implement a structured negotiation and session management system by creating a formal workflow for session requests, price negotiation, and interactions.

1.4 Scope and Limitations

1.4.1 Scope:

- Allows students to search, compare, and request tutors based on subject, level, location, time, and price.
- Enables tutors to manage their profiles, session requests and pricing.

- Supports negotiation on session time and pricing between student and tutor.
- Real-Time Communication can be done through integrated chat system (using WebSockets) accessible only after a session is accepted.
- Ability for students to rate and provide feedback for tutors after a session is completed.

1.4.2 Limitations:

- Only supports in-person sessions (no online video-based learning).
- Requires both tutor and student to have internet access for system usage.

1.5 Methodology

Iterative Waterfall Model

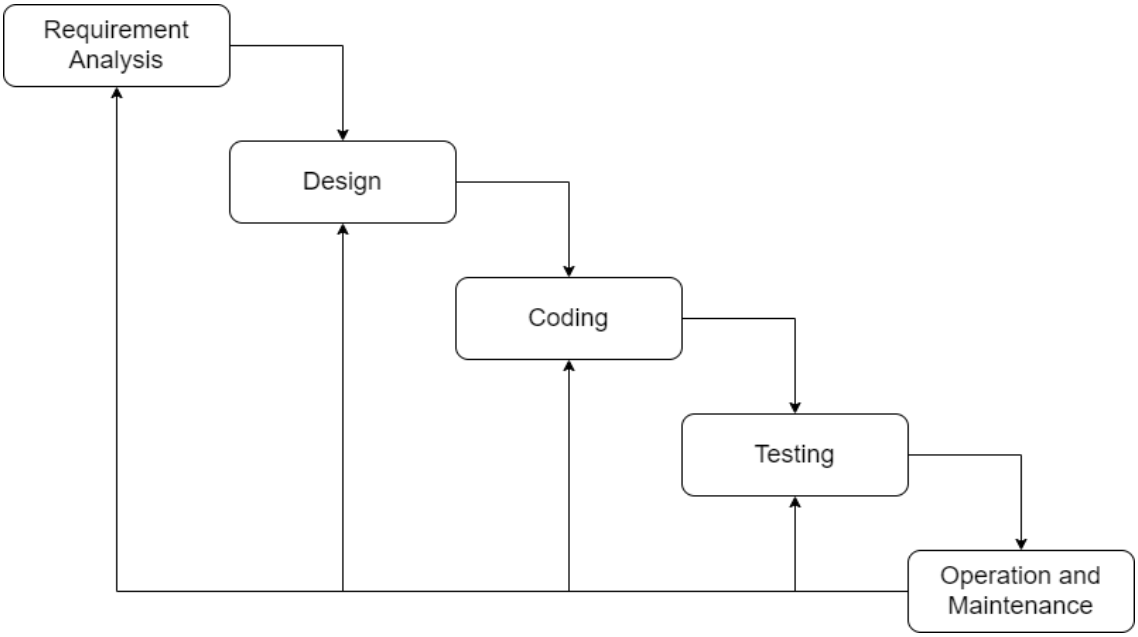

Figure 1.1: Iterative Waterfall Model

1. **Requirement Analysis**: This phase involves gathering and documenting the software requirements from stakeholders focusing on understanding what the system needs to do and sets the foundation for all subsequent phases.
2. **Design**: In this phase, the system's architecture is defined based on the requirements. It involves creating detailed design specifications that guide the coding process, including data models, interfaces, and system components.
3. **Coding**: The actual implementation of the software occurs in this phase. Developers

write the code according to the design specifications, translating the design into a functional software application.

4. **Testing**: This phase involves verifying that the software works as intended by executing various tests. It aims to identify and fix bugs, ensuring the software meets the specified requirements.
5. **Operation & Maintenance**: After deployment, this phase ensures the software operates smoothly in the live environment. It includes ongoing maintenance tasks such as fixing bugs, updating the software, and enhancing its functionality based on user feedback.

1.6 Report Organizations

This report is divided into six chapters to explain the project clearly:

Chapter 1: Introduction

Gives a brief overview of the system, what problem it solves, its goals, scope, and how the report is structured.

Chapter 2: Background Study and Literature Review

Covers the basic concepts behind the project and reviews similar systems or related works that helped shape this one.

Chapter 3: System Analysis

Focuses on what the system needs, its feasibility, and how it was analysed using diagrams and models.

Chapter 4: System Design

Explains how the system was designed from database and interface design to algorithms and diagrams used in planning.

Chapter 5: Implementation and Testing

Talks about the tools used, how the system was built, tested, and how it performed during testing.

Chapter 6: Conclusion and Future Recommendations

Summarizes what was achieved, lessons learned, and suggestions for improving or extending the system in the future.

Chapter 2: Background Study and Literature Review

2.1 Background Study

In today's fast-paced and technology-driven world, students often face challenges in finding qualified tutors who are available nearby and suitable for their learning needs. Traditional methods of searching for tutors through word of mouth or physical advertisements are inefficient, time-consuming, and limited by location. With the rise of web-based applications, there is an increasing demand for platforms that can digitally connect students with the right tutors in real time. MeetMyTutor is developed to address this gap by providing a centralized, location-based tutoring system. The system integrates geolocation technology and Dijkstra's algorithm to help students find the nearest tutors efficiently. It also allows tutors to manage their schedules, pricing, and availability while enabling students to book sessions, communicate directly, and give feedback. By digitalizing the private tutoring process, MeetMyTutor aims to make education more accessible, transparent, and personalized for every learner.

2.2 Literature Review

Prosikhya is an educational platform that connects students with private tutors for personalized, one-on-one in-person learning. The system operates through a simple three-step process: students first search for tutors based on subject, grade level, location, and preferences, using tutor profiles that showcase qualifications, teaching experience, and areas of expertise. Once a suitable tutor is selected, students can book a tuition session by choosing a convenient time slot, with a guided booking process that ensures ease of use. After booking, the platform enables direct communication between students and tutors to discuss learning needs, clarify expectations, and finalize payment arrangements [1].

Prosikhya's strengths lie in its simplicity, detailed tutor profiles, direct interaction, and focus on offline learning. However, compared to MeetMyTutor, it lacks certain advanced features such as price and time negotiation, attendance tracking, session validation, and centralized admin management for subjects, reviews, and user activity. Additionally, its payment process is manual and not integrated into the system, limiting automation and efficiency.

TutorBird is a tutor management platform designed for tutoring centers and freelance tutors. It offers features like student management, calendar scheduling, attendance tracking, invoicing, payments, multi-tutor support, a student portal, and a website builder. It focuses on streamlining admin tasks and business operations [2]. Unlike MeetMyTutor, TutorBird does not support direct price/time negotiation between students and tutors or attendance validation by both parties. It is better suited for established tutoring businesses, while MeetMyTutor is focused on helping students find and manage in-person private tutors more flexibly and transparently.

Private Tutor Finder System is a web-based platform aimed at connecting parents and tutors for personalized, nearby home tuition. It includes three main roles: Admin, Parents, and Tutors. Admin manages tutor registrations, e-books, and parent records. Tutors are registered by admin and receive credentials via email. Parents can search, filter, and request demo classes with tutors. After the demo, they can book sessions, rate tutors, and access e-books. Tutors can manage demo requests and bookings upon logging in with credentials.

The Private Tutor Finder system relies more on admin-managed tutor registration and lacks features like direct negotiation, payment integration, and session-based accountability [3].

TeacherOn is a global platform that connects students and tutors for both online and offline classes. It supports a wide range of subjects, including academics, languages, music, and competitive exams. Tutors can create profiles, and students can directly contact them using a coin system. It's widely used in Nepal and offers secure payment options, helping tutors earn from both local and international students. Teachers create a complete profile to access and contact students—some leads are free while others require coins, which help minimize spam and ensure serious inquiries. Students can also reach out to tutors at no cost. Both parties negotiate their own terms and payment methods, but TeacherOn provides an optional secure payment system where students deposit fees, tutors deliver the service, and payment is released only after student satisfaction. This protects both sides from fraud. Teachers can receive payments via Bank (in select countries), PayPal, or TeacherOn Coins, with a 15% service fee (or 10% if paid in coins). Students get 150 free coins initially to contact up to three tutors, making it easy to start learning [4].

Chapter 3: System Analysis

3.1 Requirement Analysis

3.1.1 Functional Requirements

A Functional Requirements (FR) is a description of the service that the software must offer. It describes a software system or its component. A function is nothing but inputs to the software system, its behavior, and outputs. The functional requirements are as follows:

- **User Registration and Login:**
  The system allows both students and tutors to create accounts by providing necessary details such as name, email, contact number, and role (student or tutor). After registration, users can securely log in using their credentials. Password encryption and session management are implemented to ensure account safety and data privacy.
- **Tutor Profile Management:**
  Tutors can create and update their profiles with relevant information including subjects taught, educational qualifications, teaching experience, location, available time slots, and fee structure. These profiles help students evaluate and select tutors based on their learning needs. Tutors can also upload profile pictures and certifications for added credibility.
- **Search and Filter Tutors:**
  Students can search for tutors based on multiple criteria such as subject, grade level, location (nearest), availability, and price. The system provides advanced filters to narrow down search results, making it easier for students to find the most suitable tutors quickly. The filtered results display brief tutor details and ratings.
- **Sessions Request and Negotiation:**
  Once a student selects a tutor, they can book a session by choosing a preferred date and time from the tutor's available schedule. The system guides users through a step-by-step booking process, confirming session details before finalization. After booking, both the student and tutor receive confirmation and session reminders.
- **Session Management**
  Tutors and students can manage ongoing sessions, view session history, and track scheduled sessions.

- **Interactive Chat System:**

  Real-time messaging between students and tutors is available after session acceptance.

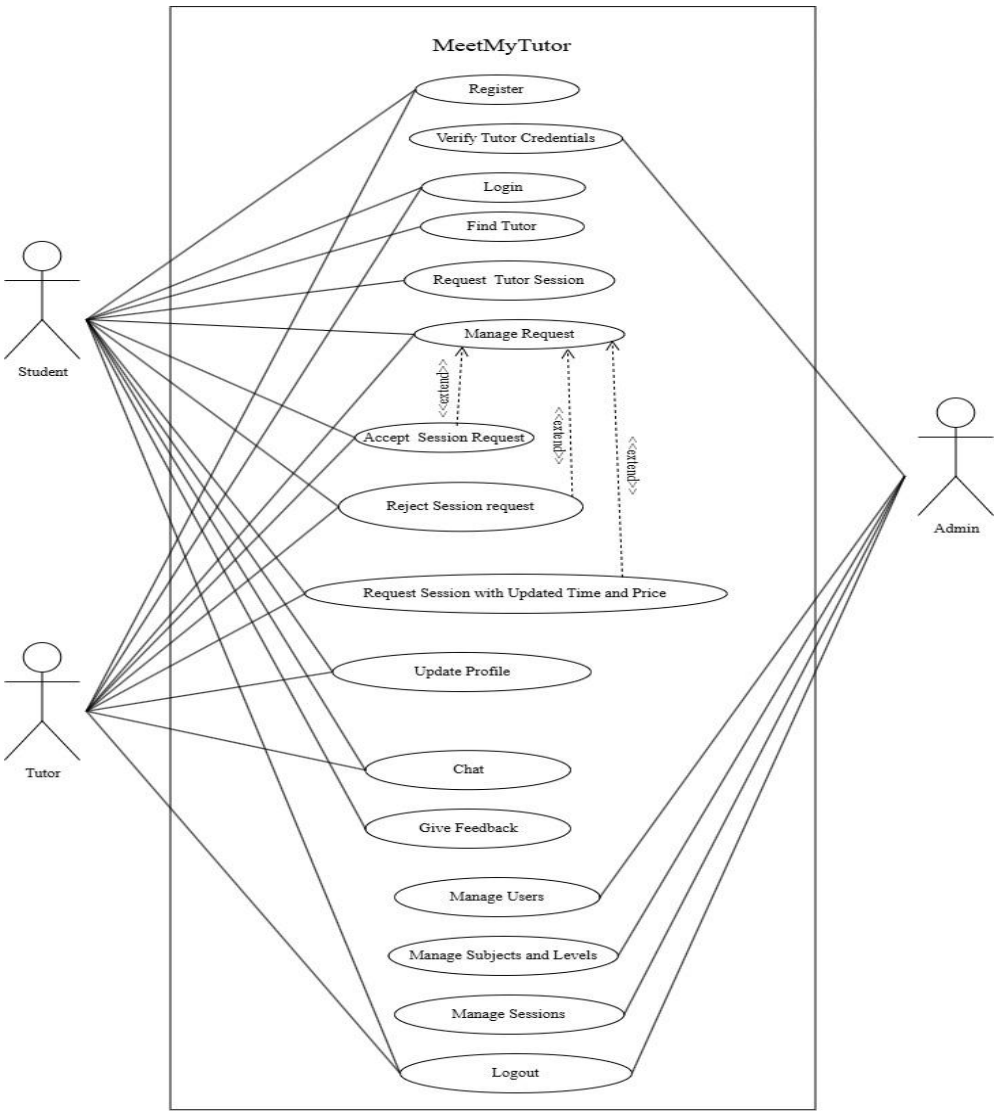

Figure 3.1: Use Case Diagram

3.1.2 Non-Functional Requirements

A Non-Functional Requirements (NFR) are a set of specifications that describe the systems Operation capabilities and constraints and attempt to improve its functionality. They describe qualities like performance, reliability, security, usability, and scalability, ensuring the system meets user expectations and operates efficiently under various conditions. Following are non-functional requirements of our system:

- **Performance:**

  The system should be able to handle multiple simultaneous users, including students and tutors, without noticeable delays in loading pages or fetching data. Searching for nearby tutors should return results in under 3 seconds.

- **Scalability:**

  The system should support growth in the number of users, tutors, sessions, and queries without affecting performance, allowing future expansion across cities or regions.

- **Security:**

  User credentials, session data, and payment information must be securely stored and transmitted using encryption protocols. JWT-based authentication and secure database access should prevent unauthorized access.

3.2 Feasibility Study

3.2.1 Technical Feasibility

All the tools and technologies required for MeetMyTutor are widely accessible, well-supported, and suitable for building a scalable and interactive tutoring platform. The system leverages Django for the backend and modern frontend technologies to ensure real-time data handling, efficient user management, and location-based computations. The required tools are cost-effective and compatible with standard development environments.

i. Programming Language used: Python (backend), JavaScript (frontend)

ii. Development Tool used: Visual Studio Code

iii. Frameworks and Technologies used:

- Django for server-side logic, session management, and secure authentication
- HTML, CSS, Tailwind CSS, JavaScript for responsive and interactive user interfaces
- WebSockets for real-time messaging and notifications
- SQLite (development) / PostgreSQL (production) for storing user, session, and location data

3.2.2 Operational Feasibility

The system simplifies the process of finding and managing private tutoring sessions. Students can easily locate nearby tutors based on location and subject, while tutors can manage session

requests and schedules efficiently. The platform is user-friendly, accessible across devices, and supports real-time communication, ensuring smooth daily operations for both students and tutors.

3.2.3 Economic Feasibility

The system is cost-effective to develop and maintain, requiring only standard computing hardware and open-source software frameworks. While there are costs associated with development, hosting, and maintenance, the benefits such as efficient tutor-student matching, time savings, and improved learning accessibility making the system financially viable and valuable in the long term.

3.2.4 Schedule Feasibility

Table 1: Gantt Chart

Months (in week)	April			May				June				July				August			
Tasks	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Requirement Analysis	■	■	■																
Design				■	■														
Coding						■	■	■	■	■	■	■	■						
Testing														■	■	■			
Operation and Maintenance																	■	■	■
Documentation	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■

Chapter 4: System Design

4.1 System Design

System design is the process of planning and defining the structure, components, modules, interfaces, and data for a system to satisfy specified requirements. Generally, this chapter deals with the module, database design, user interface design and problem design. For MeetMyTutor, this chapter focuses on designing the overall system architecture, database, modules, user interfaces, and algorithms to ensure smooth and efficient operation.

4.1.1 System Architecture

System Architecture is the high-level structure of a software system that defines the components, interactions, technologies and design patterns.

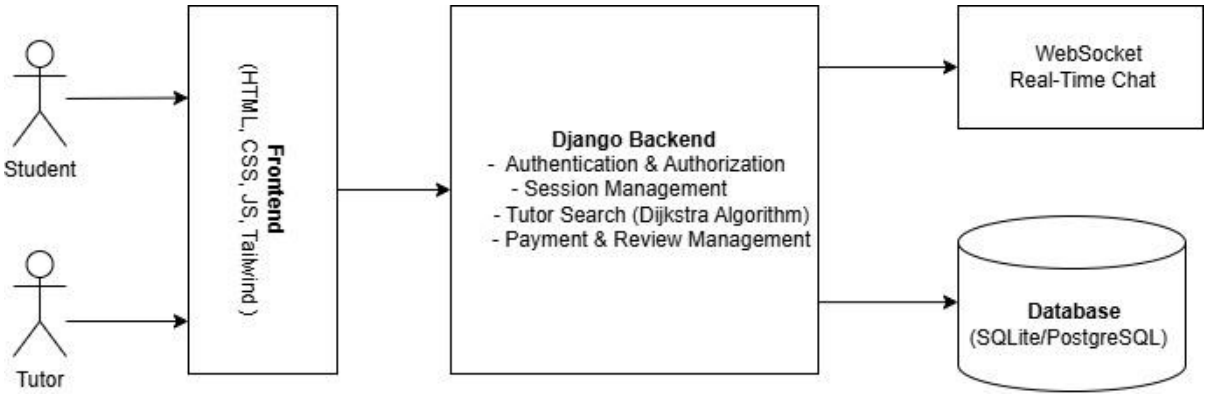

Figure 4.1: System Architecture

4.1.2 Class Diagram

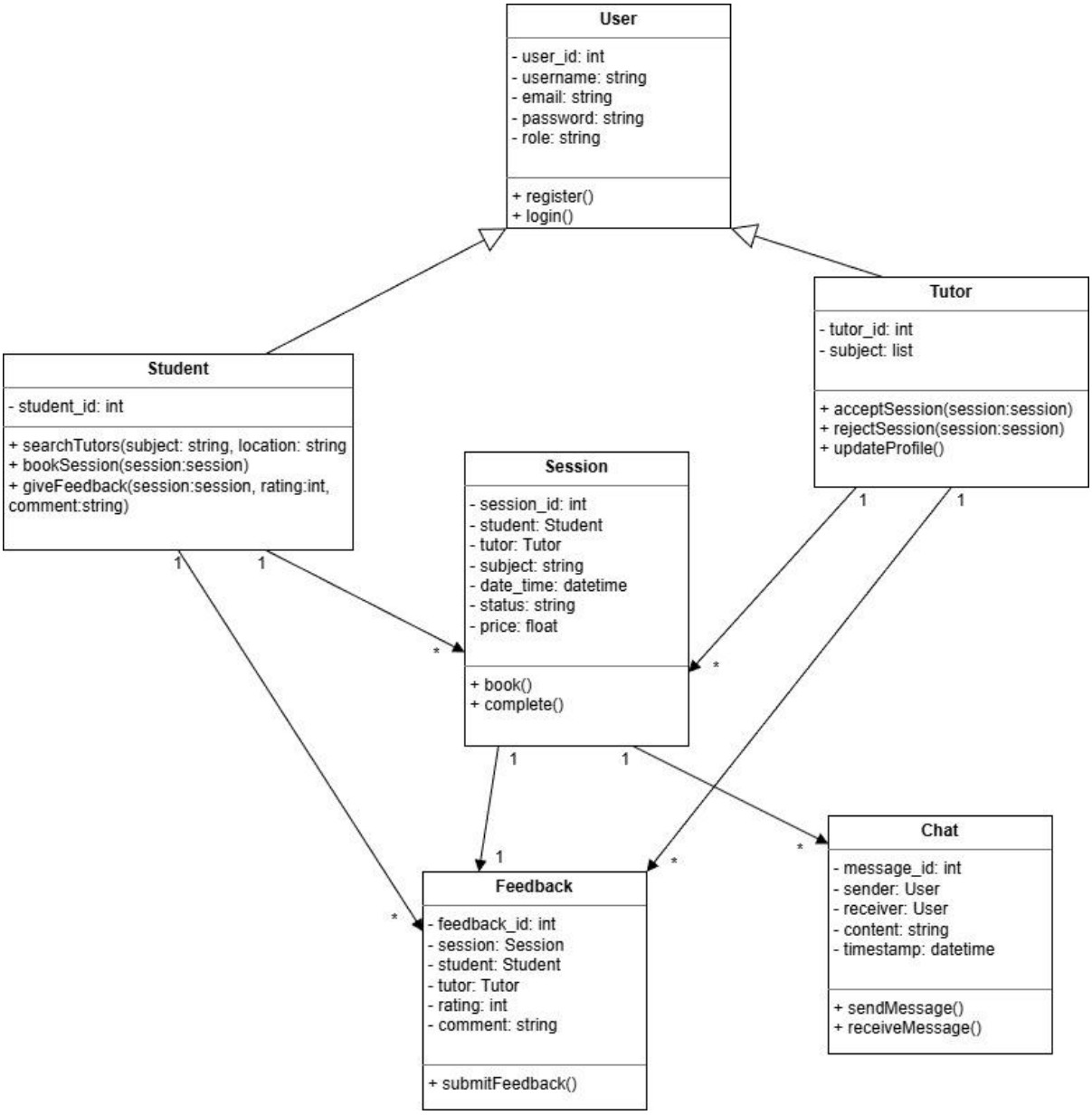

Figure 4.2: Class Diagram

4.1.3 Sequence Diagram

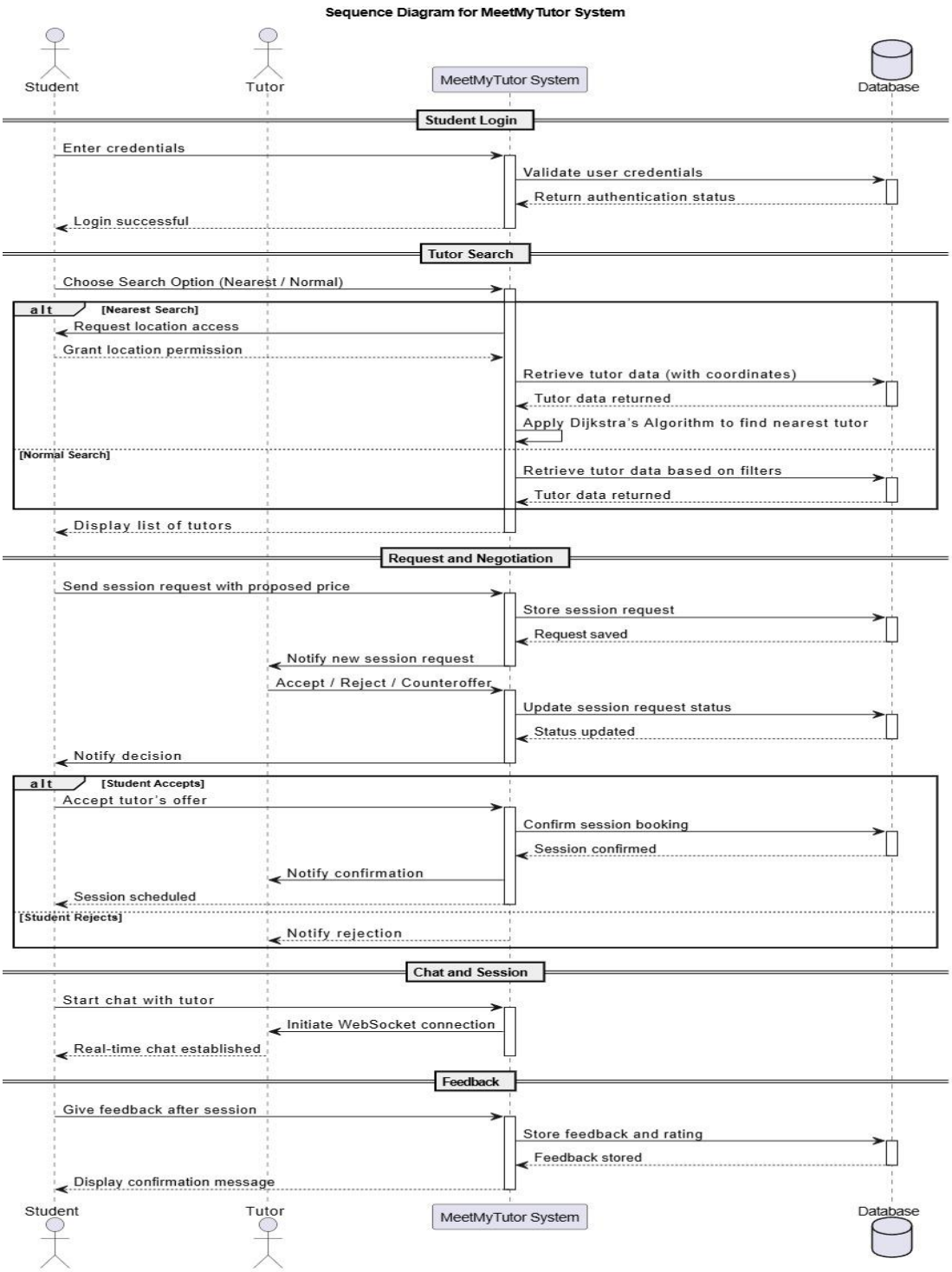

Figure 4.3: Sequence Diagram

4.1.4 Activity Diagram

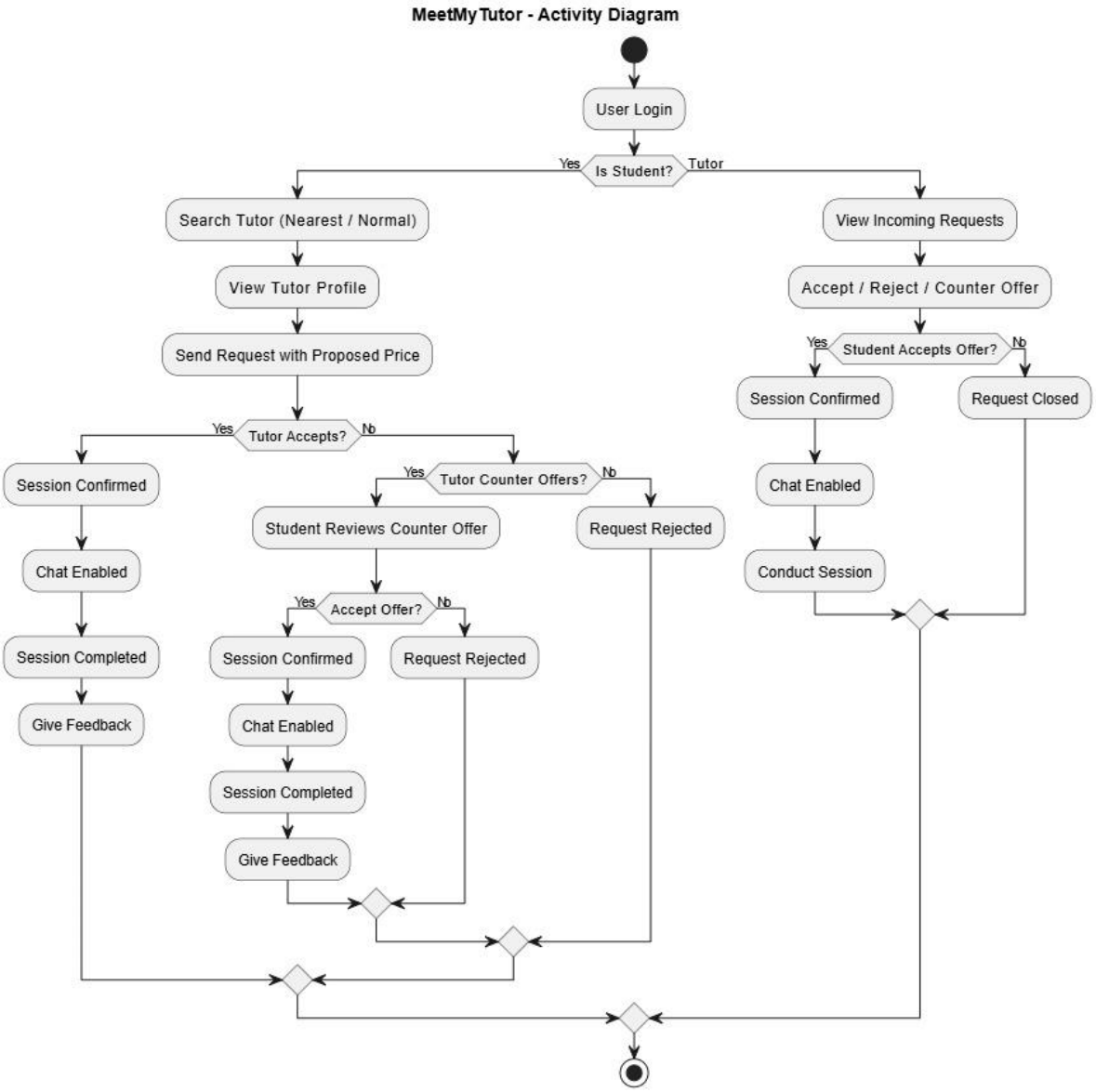

Figure 4.4: Activity Diagram

4.2 Descriptions of Algorithm

Dijkstra's Algorithm: An algorithm used to find the shortest path from a starting point to all other points in a weighted graph. It calculates the minimum distance to each node and helps determine the closest or fastest route.

Purpose:

To find the nearest tutor to a student based on location, considering location permissions.

Input:

- student_location (latitude, longitude)
- tutor_list (list of tutors with latitude, longitude, and location permission)
- graph (weighted graph of locations)

Output:

- nearest_tutor
- minimum_distance

Steps:

1. Request student location permission. If denied, stop execution.
2. Filter tutors who have granted location permission.
3. Map student_location and tutor locations to nearest graph nodes.
4. Initialize Dijkstra's algorithm:
    a. distance[start_node] = 0
    b. distance[all other nodes] = ∞
    c. Initialize priority queue with start_node
5. While the priority queue is not empty:
    a. Pop node with smallest distance
    b. For each neighbor:
        i. Calculate new_distance = current_distance + edge_weight
        ii. If new_distance < distance[neighbor]:
            - distance[neighbor] = new_distance
            - Add neighbor to priority queue
6. Find the tutor node with minimum distance.
7. Return nearest_tutor and minimum_distance.

Chapter 5: Implementation and Testing

5.1 Implementation

5.1.1 Tools Used

i. **Django:** A high-level Python web framework used for backend development. It provides an organized structure for building secure, scalable, and maintainable web applications, handling server-side logic, database interactions, and routing.

ii. **SQLite:** A lightweight, file-based relational database used during development to store user profiles, sessions, messages, feedback, and other application data.

iii. **HTML5:** The standard markup language used to structure the frontend pages and render dynamic content through Django templates.

iv. **CSS / Tailwind CSS:** Tailwind CSS is a utility-first framework used to style the frontend efficiently. It helps build responsive, modern, and consistent UI components.

v. **JavaScript:** Used to implement interactive frontend features, such as chat functionality, session requests, and real-time updates.

vi. **WebSockets:** Enables real-time communication between students and tutors for instant messaging and live session updates.

vii. **Python:** The primary programming language used to develop backend logic, process requests, and manage database operations in Django.

viii. **Psycopg2:** A PostgreSQL database adapter for Python (optional if using PostgreSQL) to handle database connections and queries.

ix. **Git & GitHub:** Version control and repository hosting tools used to track changes, collaborate, and manage the project source code.

x. **JWT (JSON Web Token):** Used for secure authentication and session management, ensuring that only authorized users can access protected resources.

5.1.2 Implementation Details of Modules

i. **Authentication and Authorization Module:** This module manages user registration and login securely using Django's built-in authentication system. It verifies user credentials and controls access based on whether the user is a student or a tutor.

ii. **User Management Module:** This module allows tutors and students to update their personal information, subjects, and location details. The stored location data helps in finding nearby tutors when needed.

iii. **Tutor Discovery Module:** This module helps students find the nearest tutors based on their geographical location. It uses Dijkstra's Algorithm to calculate the shortest distance between the student and tutors using latitude and longitude values.

iv. **Session Management Module:** This module handles the process of creating, accepting, and completing tutoring sessions. It ensures smooth communication between students and tutors throughout the session lifecycle.

v. **Messaging Module:** This module provides real-time chatting between tutors and students using WebSockets. It allows users to exchange messages instantly and stay connected during sessions.

vi. **Feedback and Rating Module:** This module enables students to give feedback and rate tutors after completing a session. The collected ratings help improve tutor visibility and maintain quality across the platform.

5.2 Testing

5.2.1 Unit Testing

Table 2: Unit Testing

S.N.	Test Case ID	Test Description	Steps Executed	Expected Result	Actual Result	Pass/ Fail
1	UT-001	Register new Tutor/Student with valid credentials	Go to the registration page, enter valid name, email, password, and user type (Tutor/Student	Account should be created successfully and redirected to the login page.	Account created successfully and redirected to login page.	Pass

			), then click Register.			
2	UT-002	Login with valid credentials	Enter correct email and password, then click Login.	User should be logged in and redirected to the dashboard.	User logged in successfully and redirected to dashboard.	Pass
3	UT-003	User login with invalid credentials	Enter incorrect email or password.	Error message should appear indicating invalid credentials.	Error message displayed correctly.	Pass
4	UT-004	Tutor registration with location access	Register as a tutor and allow location permission when prompted.	Tutor account should be created with stored location data.	Tutor account created and location stored successfully.	Pass
5	UT-005	Student search for nearest tutors	Log in as a student and click “Find Tutors Near Me.”	System should display nearest tutors based on student’s location using Dijkstra’s algorithm.	Nearest tutors displayed correctly based on location.	Pass

6	UT-006	Tutor listing by subject	Search tutors by subject or category.	Tutors teaching that subject should be listed.	Tutors correctly listed by subject.	Pass
7	UT-007	Tutor profile update	Log in as tutor, go to profile settings, and update name, subjects, or availability.	Updated information should be saved and reflected immediately.	Profile updated successfully and changes visible.	Pass
8	UT-008	Student sends session request	Log in as student and request a session with a tutor.	Session request should be sent and notification delivered to tutor.	Request sent and tutor notified successfully.	Pass
9	UT-009	Tutor accepts session request.	Log in as tutor and accept the session request.	Session status should change to “Accepted.”	Session status updated to “Accepted.”	Pass
10	UT-010	Real-time chat between tutor and student	Open chat window and send a message.	Message should be delivered instantly using WebSocket.	Real-time chat working correctly.	Pass

11	UT-011	End a completed tutoring session	Tutor marks session as completed.	Session status should update to "Completed."	Session marked as completed successfully.	Pass
12	UT-012	Rate and review a tutor	Student rates tutor after session completion.	Rating and feedback should be stored and displayed on tutor profile.	Rating and feedback saved and displayed.	Pass
13	UT-013	Logout functionality	Click Logout from dashboard.	User should be logged out and redirected to login page.	User logged out successfully.	Pass
14	UT-014	Access restricted pages without login	Try accessing dashboard or chat page without logging in.	User should be redirected to login page.	Redirected to login page correctly.	Pass

Chapter 6: Conclusion

6.1 Conclusion

The MeetMyTutor system successfully bridges the gap between students seeking guidance and tutors offering educational services. It provides a user-friendly and efficient platform where students can easily find tutors based on location, subject, and rating, while tutors can manage their profiles, schedule sessions, and communicate directly with learners. The integration of algorithms such as Dijkstra's shortest path enhances tutor discovery based on proximity, ensuring relevance and convenience. Overall, the system achieves its objective of simplifying the tutoring process through a secure, responsive, and interactive web application.

6.2 Future Recommendations

i. **AI-Based Tutor Matching:** Incorporate an AI recommendation system that automatically matches students with the most suitable tutors based on learning style, ratings, and past sessions.

ii. **Video Conferencing Integration:** Add a built-in video call feature for conducting online classes directly within the platform.

iii. **Mobile Application Development:** Develop a mobile app version for both Android and iOS to improve accessibility and user engagement.

iv. **Advanced Analytics Dashboard:** Introduce analytics for tutors and admins to track performance, engagement, and earnings.

v. **Multi-Language Support:** Include local language options to make the platform more inclusive for users from diverse backgrounds.

References

[1] "Prosikshya," [Online]. Available: https://prosikshya.com/about-prosikshya/. [Accessed 14 May 2025].

[2] "TutorBird," [Online]. Available: https://www.tutorbird.com/. [Accessed 14 May 2025].

[3] "Online Private Tutors Finder System," Nevon Projects, [Online]. Available: https://nevonprojects.com/online-private-tutors-finder-system/. [Accessed 14 May 2025].

[4] "TeacherOn," [Online]. Available: https://www.teacheron.com/?utm_medium=company_profile&utm_source=trustpilot&utm_campaign=domain_click. [Accessed 14 May 2025].