SMART-TUTOR

A PROJECT REPORT

Submitted by

VINAY VERMA [RA2211003010535]
PRANAV SINGH [RA2211003010540]
MANISH TIWARI [RA2211003010546]

Under the Guidance of

Dr. T K SIVAKUMAR
Assistant Professor, CTECH

in partial fulfillment of the requirementsfor the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTING TECHNOLOGIES
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE ANDTECHNOLOGY
KATTANKULATHUR- 603 203

MAY 2025

Department of Computing Technologies
SRM Institute of Science & Technology
Own Work* Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course	: B.Tech / CSE CORE
Student Name	: Vinay Verma, Pranav Singh, Manish Tiwari
Registration Number	: RA2211003010535, RA2211003010540, RA2211003010546
Title of Work	: Smart-Tutor

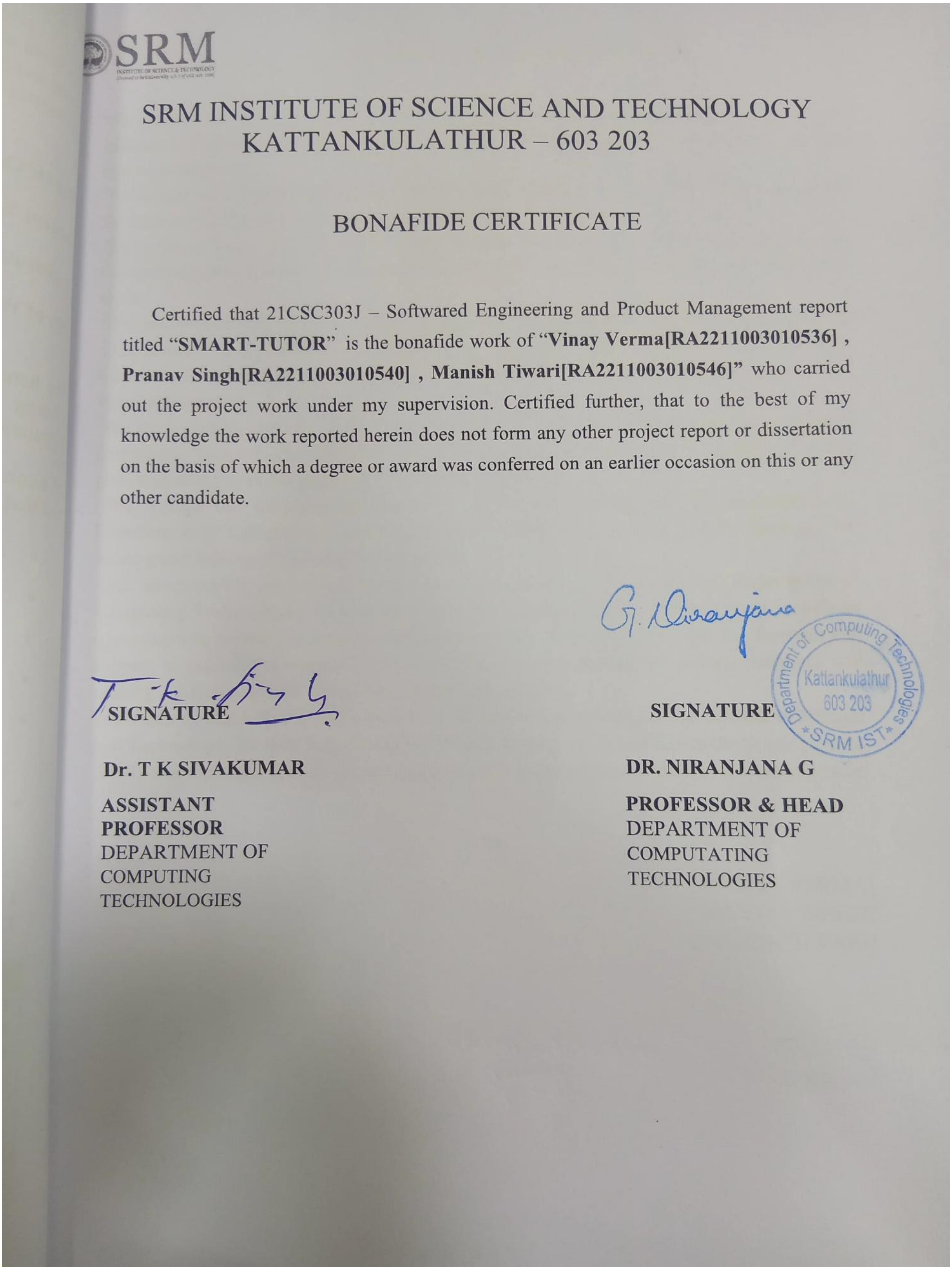
We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.
If you are working in a group, please write your registration numbers and sign with the date for every student in your group.



SRM

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 21CSC303J – Softwared Engineering and Product Management report titled "**SMART-TUTOR**" is the bonafide work of "**Vinay Verma[RA2211003010536] , Pranav Singh[RA2211003010540] , Manish Tiwari[RA2211003010546]**" who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

[illegible]

SIGNATURE

Dr. T K SIVAKUMAR

ASSISTANT PROFESSOR
DEPARTMENT OF COMPUTING TECHNOLOGIES

[illegible]

Department of Computing Technologies
Kattankulathur 603 203
SRM IST

SIGNATURE

DR. NIRANJANA G

PROFESSOR & HEAD
DEPARTMENT OF COMPUTATING TECHNOLOGIES

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to **Dr. Leenus Jesu Martin M,** Dean-CET, SRM Institute of Science and Technology, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We encompass our sincere thanks to, **Dr. M. Pushpalatha**, Professor and Associate Chairperson - CS, School of Computing and **Dr. Lakshmi,** Professor and Associate Chairperson -AI, School of Computing, SRM Institute of Science and Technology, for their invaluable support.

We are incredibly grateful to our Head of the Department **Dr. Niranjana G**, Department of Computing Technologies, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We register our immeasurable thanks to our Faculty Advisor, Dr. Thamizhamuthu R, Department of Computing Technologies, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, Dr. T K Sivakumar, Department of Computing Technologies, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his mentorship. He provided us with the freedom and support to explore the research topics of our interest. His passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff members of, School of Computing, S.R.M Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support and encouragement

[VINAY VERMA]
[PRANAV SINGH]
[MANISH TIWARI]

ABSTRACT

In the modern financial ecosystem, credit accessibility remains a crucial yet often underserved aspect, especially for individuals who lack traditional financial documentation or have non-standard income streams. Conventional credit evaluation systems largely depend on rigid rule-based scoring methods, which fail to account for dynamic borrower behavior, emerging financial patterns, and personalized financial risk. This leads to limited credit access for eligible individuals and contributes to increasing default rates due to inefficient risk assessments.

To address these challenges, we have developed a Smart Loan System for Sustainable Finance, a machine learning-based application designed to automate and enhance the credit risk evaluation process. The system utilizes historical loan data to train a predictive model that classifies applicants into three risk categories—Low, Medium, and High—based on attributes such as income, loan amount, repayment ratio, and interest rate. A Random Forest classifier was employed for its accuracy, interpretability, and robustness in handling tabular financial data. The model is integrated with a user-friendly interface that supports real-time predictions, CSV-based data uploads, and downloadable reports.

In addition to its technical design, the system includes a dashboard to visualize predictions and model behavior, offering transparency and ease of interpretation for users ranging from loan officers to students. By automating the risk evaluation process, the system reduces manual errors, enhances processing speed, and promotes fairness in credit distribution. The project aligns with the United Nations Sustainable Development Goal 8 (Decent Work and Economic Growth) by promoting financial inclusion and responsible lending practices.

This report details the design, development, implementation, and evaluation of the Smart Loan System, presenting it as a scalable, impactful solution for both educational use and institutional deployment in financial decision-making environments.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

UI – User Interface

UX – User Experience

CSV – Comma Separated Values

API – Application Programming Interface

DB – Database

JWT – JSON Web Token

CRUD – Create, Read, Update, Delete

MERN – MongoDB, Express.js, React.js, Node.js (Tech stack used)

JSON – JavaScript Object Notation

REST – Representational State Transfer (API design)

RFID – Radio Frequency Identification (if applicable for attendance or user identification)

CHAPTER 1

INTRODUCTION

1.1 Introduction to SMART-TUTOR:

In today’s fast-paced digital age, learners and tutors often face challenges in connecting through traditional educational methods. The SmartTutor project aims to bridge this gap by offering an AI-powered, community-driven tutor matching platform developed using the MERN stack (MongoDB, Express.js, React.js, Node.js). It enables students to easily register, find suitable tutors based on subject preferences, and schedule classes—all within a responsive and interactive interface.

The system leverages real-time user authentication, smart profile matching, and a class booking system integrated with review mechanisms. The front end is designed using modern React components, while Express and MongoDB handle the backend logic and data storage. This creates a seamless experience for users accessing the application across devices. Additionally, the project is managed in an Agile environment using Microsoft Planner for sprint tracking and GitHub for version control, ensuring collaborative and iterative development.

1.2 Motivation

The motivation behind the SmartTutor platform lies at the confluence of three pressing needs:

- The growing demand for accessible, quality education.
- The shift toward personalized digital learning experiences.
- The necessity for real-time, collaborative, and interactive peer learning.

With the evolution of full-stack development tools and artificial intelligence, it's now feasible to design scalable education platforms that are not only intuitive but also capable of adapting to learner preferences. By automating the tutor search and integrating feedback systems, SmartTutor empowers both students and educators to focus on what truly matters: teaching and learning.

1.3 Sustainable Development Goal of the Project

SmartTutor directly contributes to Sustainable Development Goal 4: Quality Education, particularly target 4.3 and 4.4, which advocate for equal access to affordable, quality education and relevant skills for employment and entrepreneurship.

- By offering an inclusive platform that supports:
- Affordable access to quality tutoring resources,
- Community-driven learning via reviews and engagement,
- Digital equity through online deployment with minimal system requirements,

SmartTutor empowers learners from diverse backgrounds to achieve their educational goals. The platform is designed to operate efficiently on standard internet-connected devices, making it accessible to students even in resource-constrained regions.
Through the integration of intelligent technology with human collaboration, SmartTutor helps democratize education, promoting lifelong learning opportunities and bridging gaps in access and quality across socio-economic divides.

1.4 Product Vision Statement

1.4.1 Audience:

- **Primary Audience:**
    - Students seeking personalized, efficient, and high-quality tutoring experiences across diverse subjects.
    - Tutors looking to expand their reach and engage with students based on their expertise and availability.
- **Secondary Audience:**
    - Educational institutions and learning centers seeking a scalable platform for matching students with qualified tutors.
    - Parents and guardians of students who want to ensure their children are receiving quality educational support.

1.4.2 Needs:

- **Primary Needs:**
    - A platform that intelligently matches students with tutors based on subject preferences, availability, and previous feedback.

    - A user-friendly interface for students and tutors to manage profiles, sessions, and schedules in real-time.
    - Real-time session booking, with transparent pricing and scheduling.
- **Secondary Needs:**
    - A review and feedback mechanism to ensure quality assurance and accountability.
    - Analytical insights for tutors to track student progress and session effectiveness.
    - A reporting feature that allows students and tutors to download session summaries and feedback.

1.4.3 Products:

- **Core Product:**
    - An AI-powered tutor matching platform that allows students to easily discover, book, and interact with tutors across various subjects, enabling seamless, on-demand learning.
- **Additional Features:**
    - Real-time session booking and calendar integration.
    - A feedback and rating system that empowers students to share their experiences and ensure tutor quality.
    - A student progress tracker with downloadable reports summarizing learning milestones.
    - Multi-role access (Admin, Tutor, Student) ensuring secure, role-based functionality.

1.4.4 Values:

- **Core Values:**
    - **Personalization**: Providing tailored learning experiences for each student based on their preferences, goals, and progress.
    - **Transparency**: Ensuring clear communication between students and tutors, with accessible reviews and booking histories.
    - **Efficiency**: Enabling a seamless, quick, and reliable tutor-student matching and booking system.

- **Differentiators:**
    - **AI-Powered Matching**:
      Leveraging machine learning algorithms to recommend the most suitable tutors based on student preferences and previous performance.
    - **Real-Time Interaction**:
      Offering instant booking and scheduling capabilities, ensuring flexibility and convenience for students and tutors alike.
    - **Educational Utility**:
      Acting as an educational tool for both students and tutors, with transparent performance analytics and progress tracking.

1.5 Product Goal

The SmartTutor platform is designed to revolutionize how students and tutors connect and interact in a personalized, flexible learning environment. By integrating advanced AI-based tutor matching and real-time scheduling, the platform ensures students receive the right academic support tailored to their needs. Whether it's for primary education, competitive exam preparation, or specialized subject tutoring, SmartTutor empowers students to find the best-fit tutor based on their learning style, subject requirements, and availability.

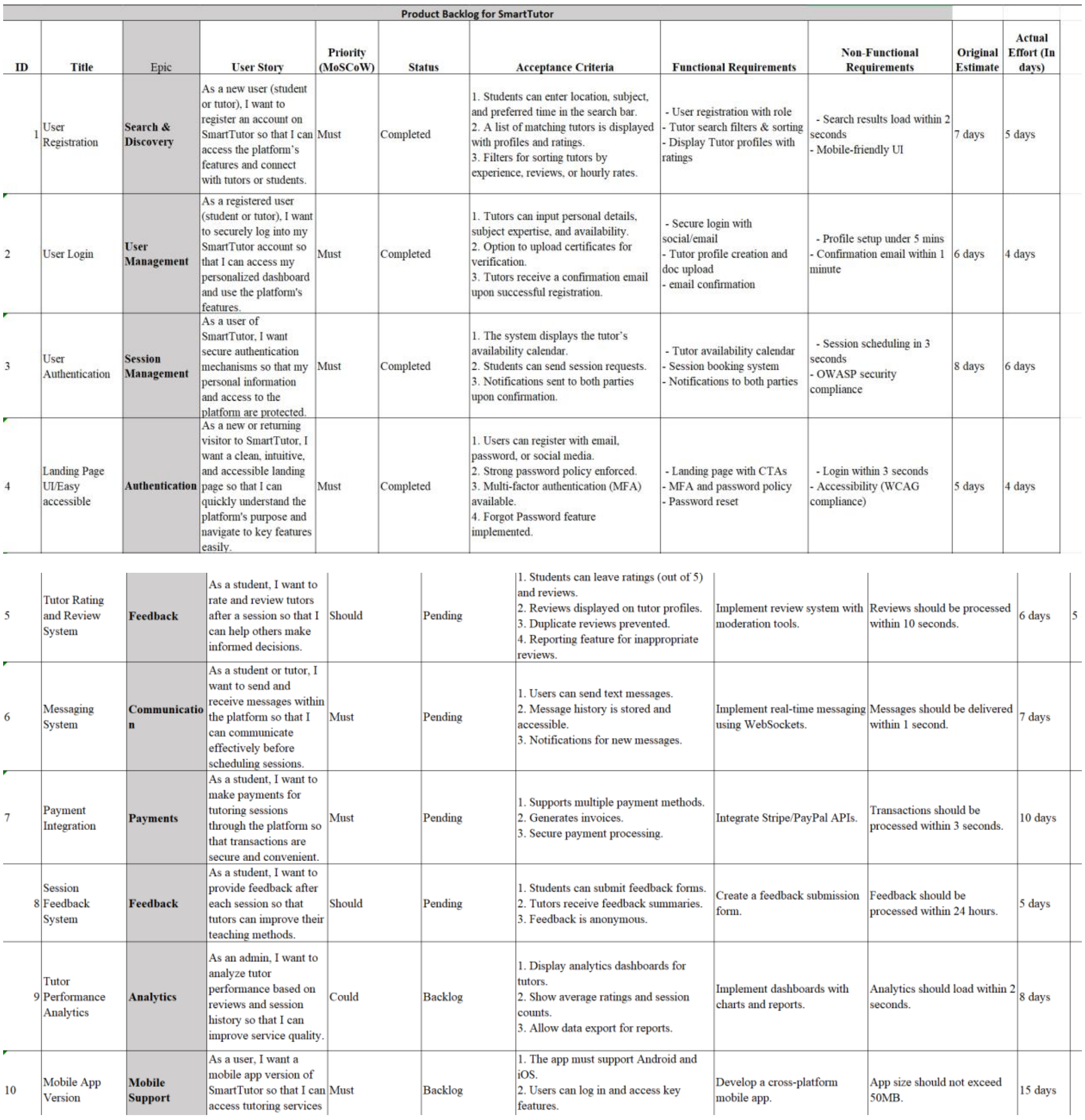
Through intelligent algorithms, SmartTutor creates an adaptive learning experience, continuously refining its suggestions to match students with tutors who can meet their evolving needs. The system does not treat students as just another user; it understands their academic journey, offering personalized insights, feedback, and recommendations based on historical data.

In addition to personalized tutor matching, SmartTutor provides a collaborative and engaging learning ecosystem. Tutors benefit from real-time session management, student feedback, and performance tracking tools that improve teaching effectiveness. Students, in turn, benefit from a more engaging and personalized educational experience that promotes continuous improvement.

Ultimately, SmartTutor aims to bridge the educational gap by providing an accessible, scalable, and user-friendly platform that connects students and tutors in a seamless manner. The platform's goal is to foster a learning environment that is inclusive, efficient, and accessible to all, regardless of geographical location or learning need. By facilitating high-quality education with AI-driven personalization, SmartTutor aspires to become the go-to platform for students and tutors seeking a mutually beneficial learning experience.

1.6 Product Backlog

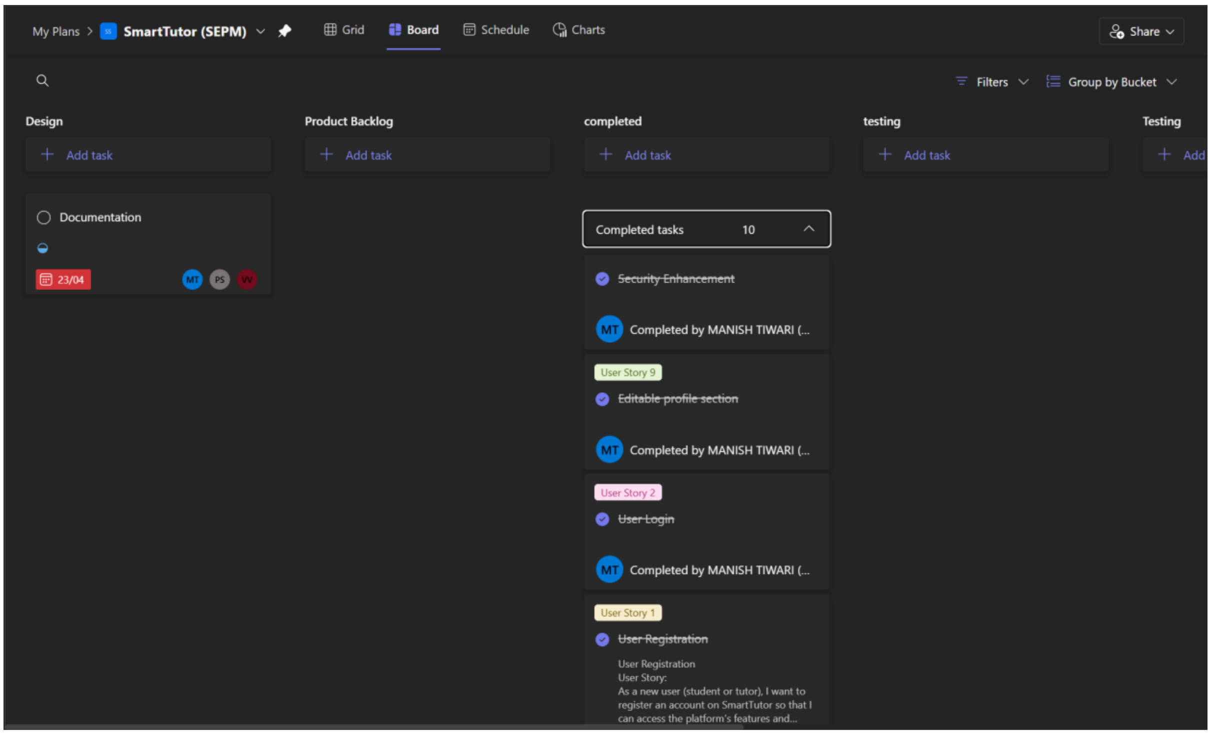
Table 1.1 Product Backlog



Product Backlog for SmartTutor

ID	Title	Epic	User Story	Priority (MoSCoW)	Status	Acceptance Criteria	Functional Requirements	Non-Functional Requirements	Original Estimate	Actual Effort (In days)
1	User Registration	**Search & Discovery**	As a new user (student or tutor), I want to register an account on SmartTutor so that I can access the platform's features and connect with tutors or students.	Must	Completed	1. Students can enter location, subject, and preferred time in the search bar. 2. A list of matching tutors is displayed with profiles and ratings. 3. Filters for sorting tutors by experience, reviews, or hourly rates.	- User registration with role - Tutor search filters & sorting - Display Tutor profiles with ratings	- Search results load within 2 seconds - Mobile-friendly UI	7 days	5 days
2	User Login	**User Management**	As a registered user (student or tutor), I want to securely log into my SmartTutor account so that I can access my personalized dashboard and use the platform's features.	Must	Completed	1. Tutors can input personal details, subject expertise, and availability. 2. Option to upload certificates for verification. 3. Tutors receive a confirmation email upon successful registration.	- Secure login with social/email - Tutor profile creation and doc upload - email confirmation	- Profile setup under 5 mins - Confirmation email within 1 minute	6 days	4 days
3	User Authentication	**Session Management**	As a user of SmartTutor, I want secure authentication mechanisms so that my personal information and access to the platform are protected.	Must	Completed	1. The system displays the tutor's availability calendar. 2. Students can send session requests. 3. Notifications sent to both parties upon confirmation.	- Tutor availability calendar - Session booking system - Notifications to both parties	- Session scheduling in 3 seconds - OWASP security compliance	8 days	6 days
4	Landing Page UI/Easy accessible	**Authentication**	As a new or returning visitor to SmartTutor, I want a clean, intuitive, and accessible landing page so that I can quickly understand the platform's purpose and navigate to key features easily.	Must	Completed	1. Users can register with email, password, or social media. 2. Strong password policy enforced. 3. Multi-factor authentication (MFA) available. 4. Forgot Password feature implemented.	- Landing page with CTAs - MFA and password policy - Password reset	- Login within 3 seconds - Accessibility (WCAG compliance)	5 days	4 days
5	Tutor Rating and Review System	**Feedback**	As a student, I want to rate and review tutors after a session so that I can help others make informed decisions.	Should	Pending	1. Students can leave ratings (out of 5) and reviews. 2. Reviews displayed on tutor profiles. 3. Duplicate reviews prevented. 4. Reporting feature for inappropriate reviews.	Implement review system with moderation tools.	Reviews should be processed within 10 seconds.	6 days	5
6	Messaging System	**Communication**	As a student or tutor, I want to send and receive messages within the platform so that I can communicate effectively before scheduling sessions.	Must	Pending	1. Users can send text messages. 2. Message history is stored and accessible. 3. Notifications for new messages.	Implement real-time messaging using WebSockets.	Messages should be delivered within 1 second.	7 days	
7	Payment Integration	**Payments**	As a student, I want to make payments for tutoring sessions through the platform so that transactions are secure and convenient.	Must	Pending	1. Supports multiple payment methods. 2. Generates invoices. 3. Secure payment processing.	Integrate Stripe/PayPal APIs.	Transactions should be processed within 3 seconds.	10 days	
8	Session Feedback System	**Feedback**	As a student, I want to provide feedback after each session so that tutors can improve their teaching methods.	Should	Pending	1. Students can submit feedback forms. 2. Tutors receive feedback summaries. 3. Feedback is anonymous.	Create a feedback submission form.	Feedback should be processed within 24 hours.	5 days	
9	Tutor Performance Analytics	**Analytics**	As an admin, I want to analyze tutor performance based on reviews and session history so that I can improve service quality.	Could	Backlog	1. Display analytics dashboards for tutors. 2. Show average ratings and session counts. 3. Allow data export for reports.	Implement dashboards with charts and reports.	Analytics should load within 2 seconds.	8 days	
10	Mobile App Version	**Mobile Support**	As a user, I want a mobile app version of SmartTutor so that I can access tutoring services	Must	Backlog	1. The app must support Android and iOS. 2. Users can log in and access key features.	Develop a cross-platform mobile app.	App size should not exceed 50MB.	15 days	

The product backlog of our project was configured using the MS planner Agile Board which is represented in the following Figure 1.1. The Product Backlog consists of the complete user stories of Smart loan System for Sustainable Finance



Design

Add task

Documentation

23/04

Product Backlog

Add task

completed

Add task

Completed tasks 10

~~Security Enhancement~~

Completed by MANISH TIWARI (...

User Story 9

~~Editable profile section~~

Completed by MANISH TIWARI (...

User Story 2

~~User Login~~

Completed by MANISH TIWARI (...

User Story 1

~~User Registration~~

User Registration
User Story:
As a new user (student or tutor), I want to register an account on SmartTutor so that I can access the platform's features and...

testing

Add task

Testing

Add

Figure 1.1 MS Planner Board of SmartTutor

1.7 Product Release Plan

The following Figure 1.2 depicts the release plan of the project

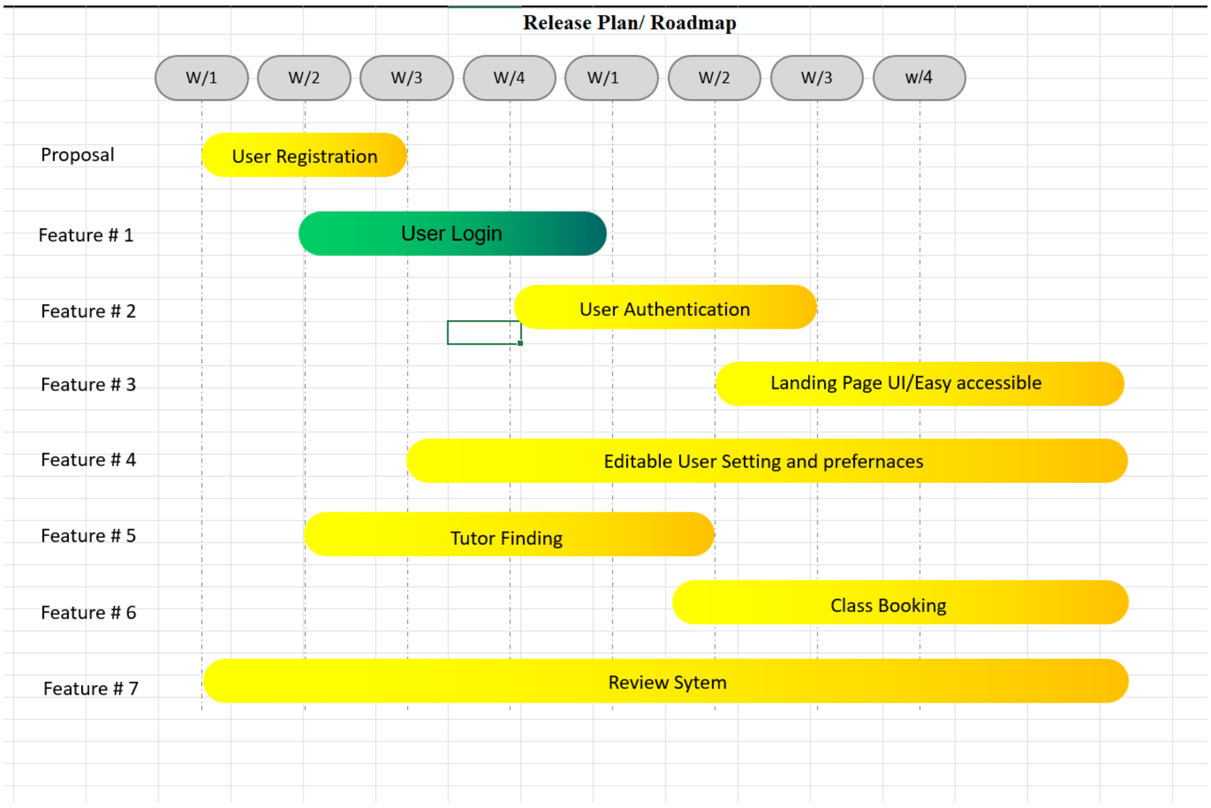


Figure 1.2 Release plan of SmartTutor

CHAPTER 2

SPRINT PLANNING AND EXECUTION

2.1 Sprint 1

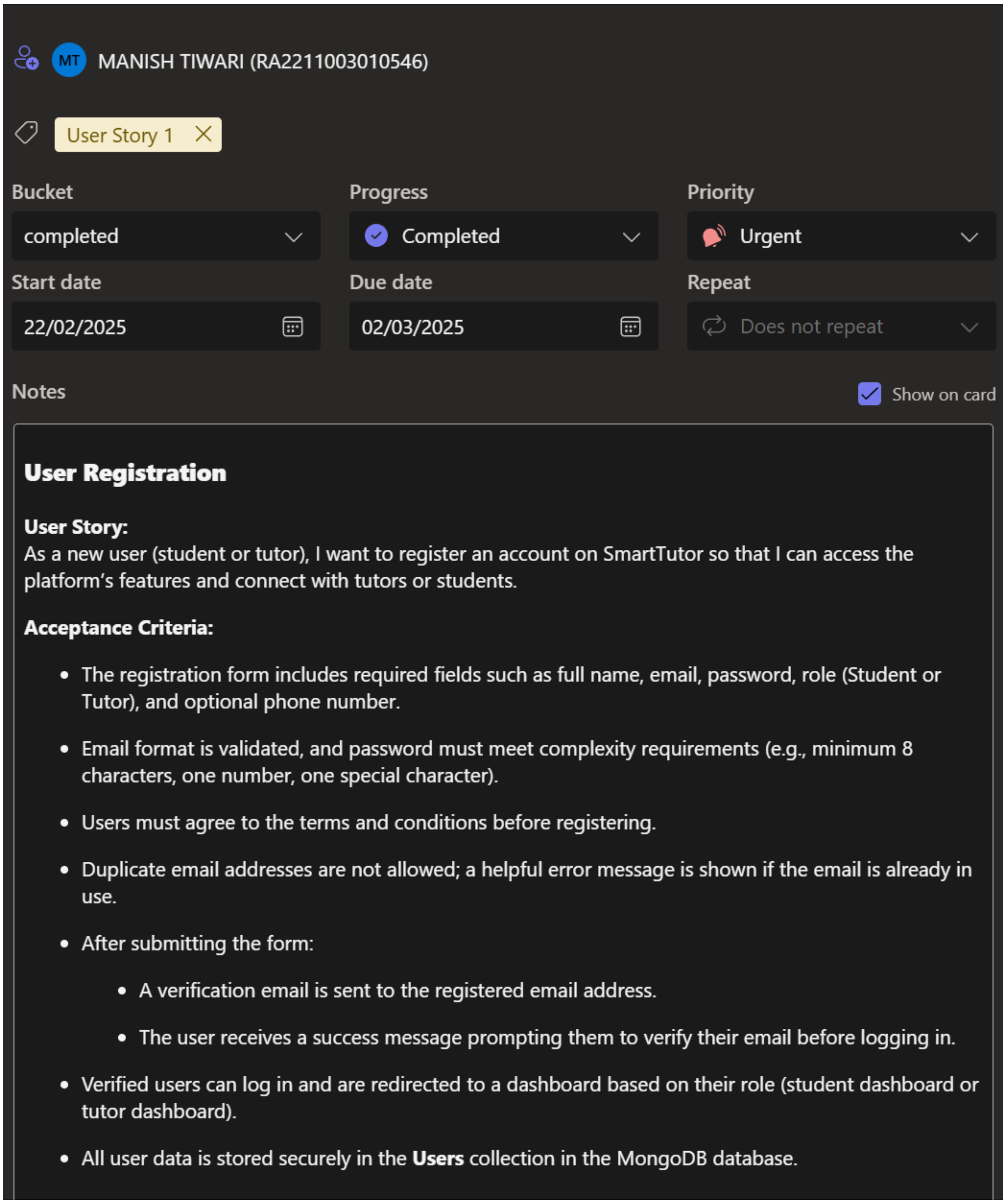
2.1.1 Sprint Goal with User Stories of Sprint 1

The goal of the first sprint is to establish the core foundation of the **SmartTutor** platform by implementing essential backend functionalities and user interface components. This includes secure user authentication, the ability to input and store tutor and student profiles, and the development of a basic search and booking system. Additionally, a simple feedback mechanism will be integrated to ensure the quality of tutoring sessions. The platform's scalability and role-based access control are also defined during this sprint.

Table 2.1 Detailed User Stories of sprint 1

S.NO	Detailed User Stories
US #1	As a student, I want to input my educational background, preferred subjects, and learning goals, so that the system can provide me with personalized tutor recommendations and learning pathways.
US # 2	As a tutor, I want to set my availability, preferred subjects, and rate, so that students can easily book sessions with me based on my schedule and expertise.
US #3	As a developer, I want to integrate machine learning algorithms to match students with tutors based on preferences and availability, so that the system can provide accurate, data-driven tutor recommendations and improve over time.
US #4	As an admin, I want to manage user roles (student, tutor, admin) and permissions, so that I can ensure secure access to platform features and protect sensitive user data.
US #5	As a product owner, I want to ensure that the platform can handle a growing number of users, so that we can scale the system to accommodate more students and tutors while maintaining a fast and reliable experience.

Planner Board representation of user stories are mentioned below figures 2.1,2.2 ,2.3,2.4 and 2.5



MT MANISH TIWARI (RA2211003010546)

User Story 1

Bucket	Progress	Priority
completed	Completed	Urgent

Start date	Due date	Repeat
22/02/2025	02/03/2025	Does not repeat

Notes ✓ Show on card

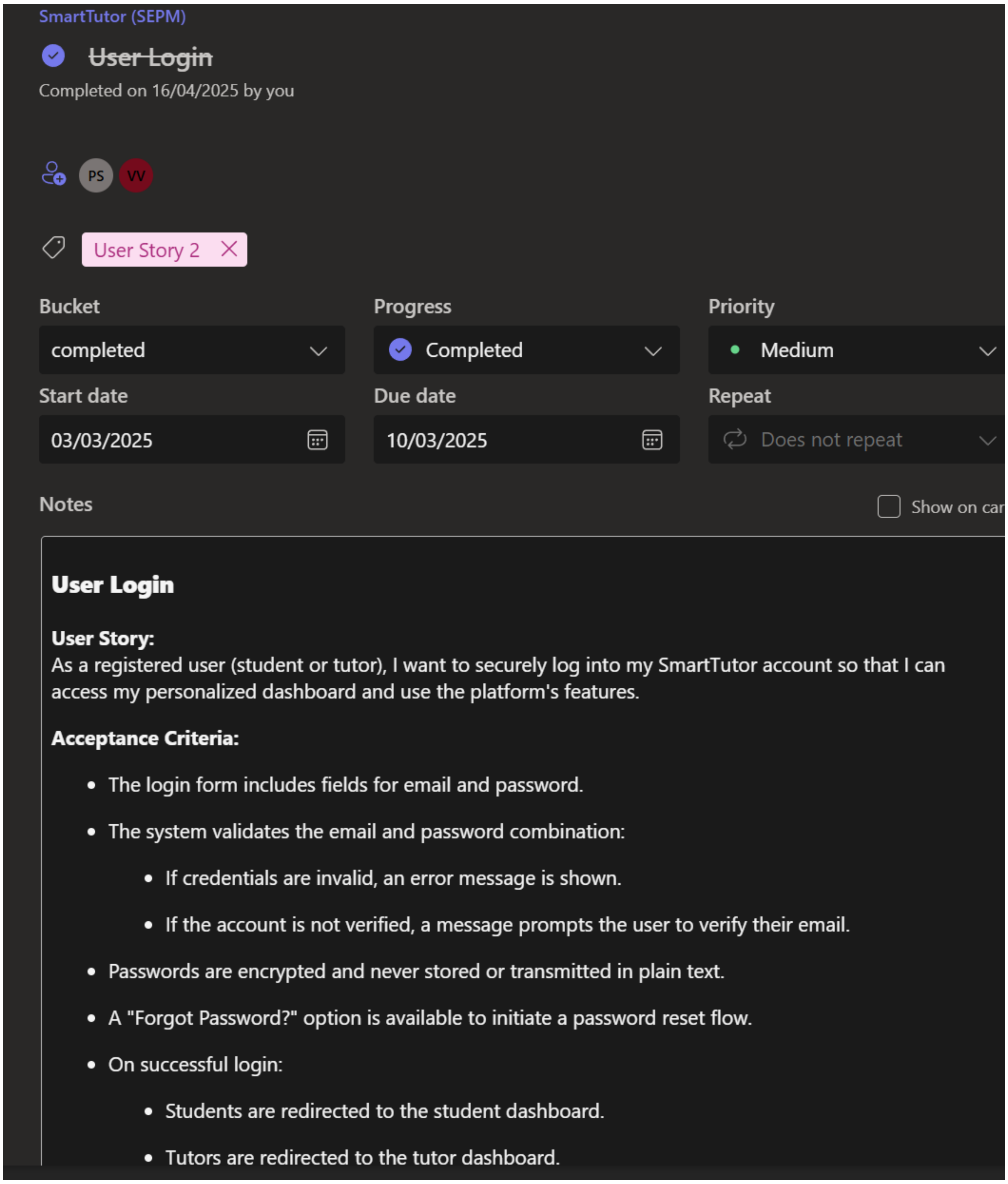
User Registration

User Story:
As a new user (student or tutor), I want to register an account on SmartTutor so that I can access the platform's features and connect with tutors or students.

Acceptance Criteria:

- The registration form includes required fields such as full name, email, password, role (Student or Tutor), and optional phone number.
- Email format is validated, and password must meet complexity requirements (e.g., minimum 8 characters, one number, one special character).
- Users must agree to the terms and conditions before registering.
- Duplicate email addresses are not allowed; a helpful error message is shown if the email is already in use.
- After submitting the form:
  - A verification email is sent to the registered email address.
  - The user receives a success message prompting them to verify their email before logging in.
- Verified users can log in and are redirected to a dashboard based on their role (student dashboard or tutor dashboard).
- All user data is stored securely in the **Users** collection in the MongoDB database.

Figure 2.1 User Story for User Registartion



SmartTutor (SEPM)

~~User Login~~

Completed on 16/04/2025 by you

PS VV

User Story 2

Bucket: completed

Progress: Completed

Priority: Medium

Start date: 03/03/2025

Due date: 10/03/2025

Repeat: Does not repeat

Notes

Show on car

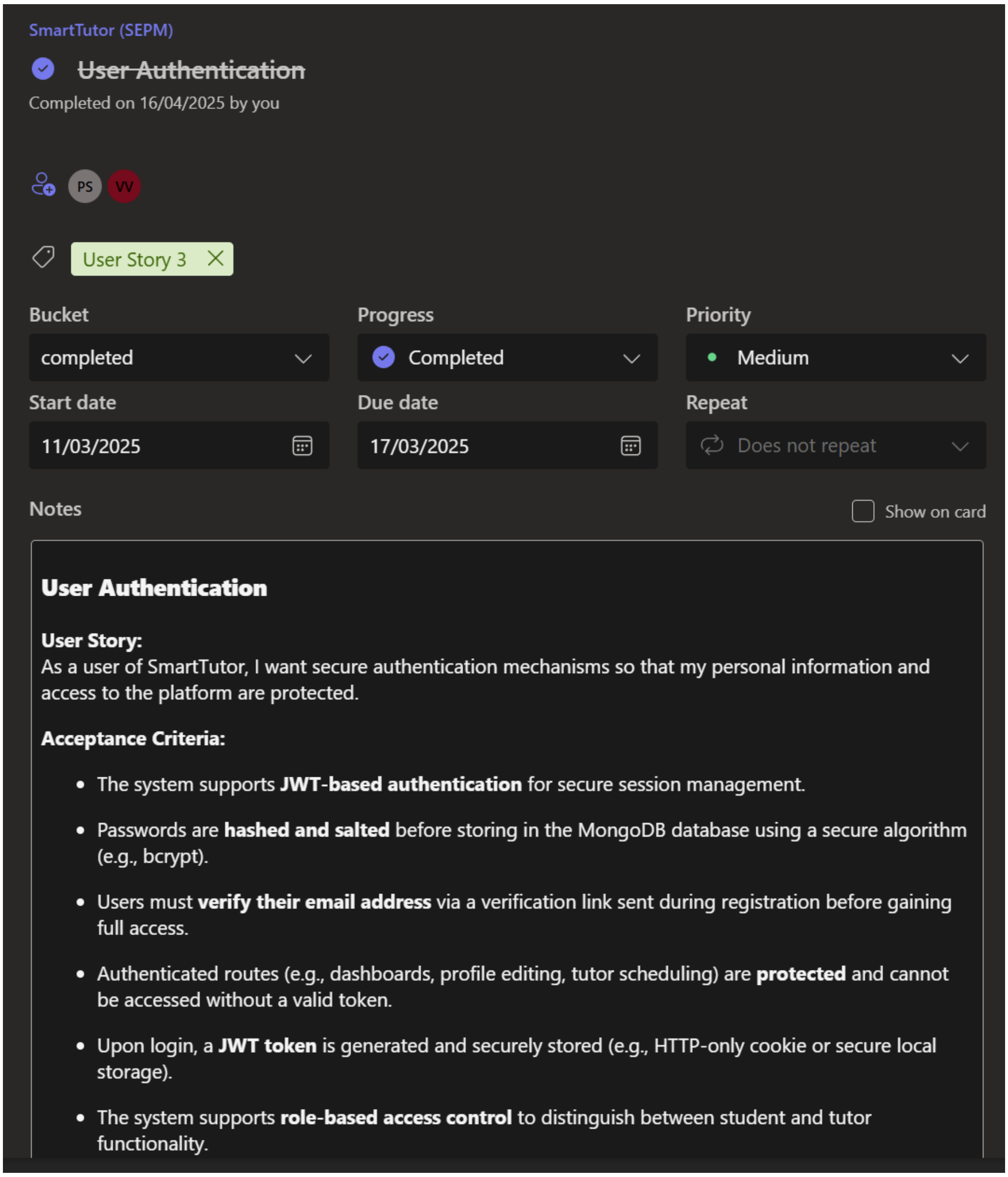
User Login

User Story:
As a registered user (student or tutor), I want to securely log into my SmartTutor account so that I can access my personalized dashboard and use the platform's features.

Acceptance Criteria:

- The login form includes fields for email and password.
- The system validates the email and password combination:
    - If credentials are invalid, an error message is shown.
    - If the account is not verified, a message prompts the user to verify their email.
- Passwords are encrypted and never stored or transmitted in plain text.
- A "Forgot Password?" option is available to initiate a password reset flow.
- On successful login:
    - Students are redirected to the student dashboard.
    - Tutors are redirected to the tutor dashboard.

Figure 2.2 User Story for User Login



SmartTutor (SEPM)

~~User Authentication~~

Completed on 16/04/2025 by you

PS VV

User Story 3

Bucket: completed

Progress: Completed

Priority: Medium

Start date: 11/03/2025

Due date: 17/03/2025

Repeat: Does not repeat

Notes

Show on card

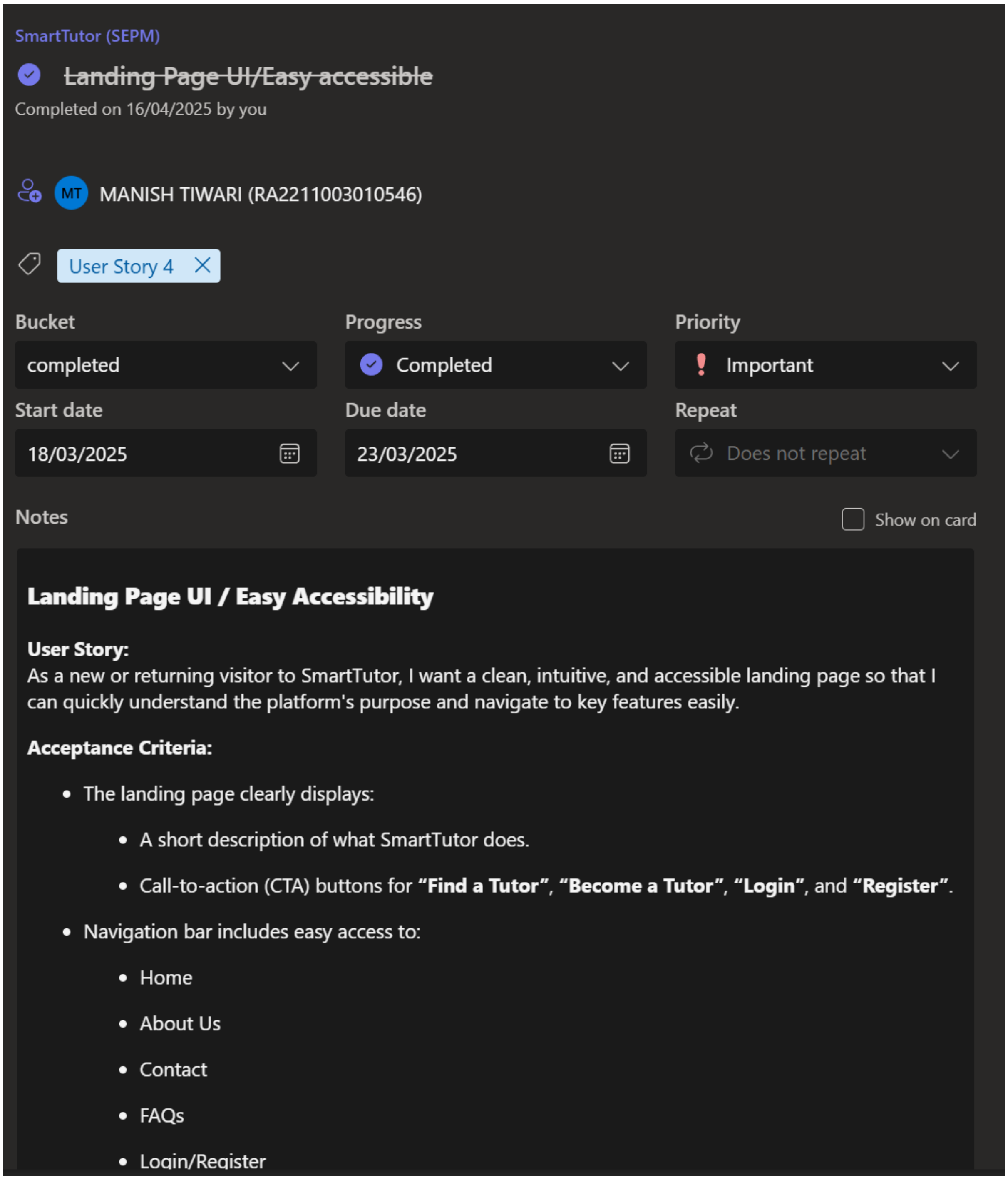
User Authentication

User Story:
As a user of SmartTutor, I want secure authentication mechanisms so that my personal information and access to the platform are protected.

Acceptance Criteria:

- The system supports **JWT-based authentication** for secure session management.
- Passwords are **hashed and salted** before storing in the MongoDB database using a secure algorithm (e.g., bcrypt).
- Users must **verify their email address** via a verification link sent during registration before gaining full access.
- Authenticated routes (e.g., dashboards, profile editing, tutor scheduling) are **protected** and cannot be accessed without a valid token.
- Upon login, a **JWT token** is generated and securely stored (e.g., HTTP-only cookie or secure local storage).
- The system supports **role-based access control** to distinguish between student and tutor functionality.

Figure 2.3 User story for User Authentication


SmartTutor (SEPM)

~~Landing Page UI/Easy accessible~~

Completed on 16/04/2025 by you

MT MANISH TIWARI (RA2211003010546)

User Story 4

Bucket	Progress	Priority
completed	Completed	Important
Start date	**Due date**	**Repeat**
18/03/2025	23/03/2025	Does not repeat

Notes

Show on card

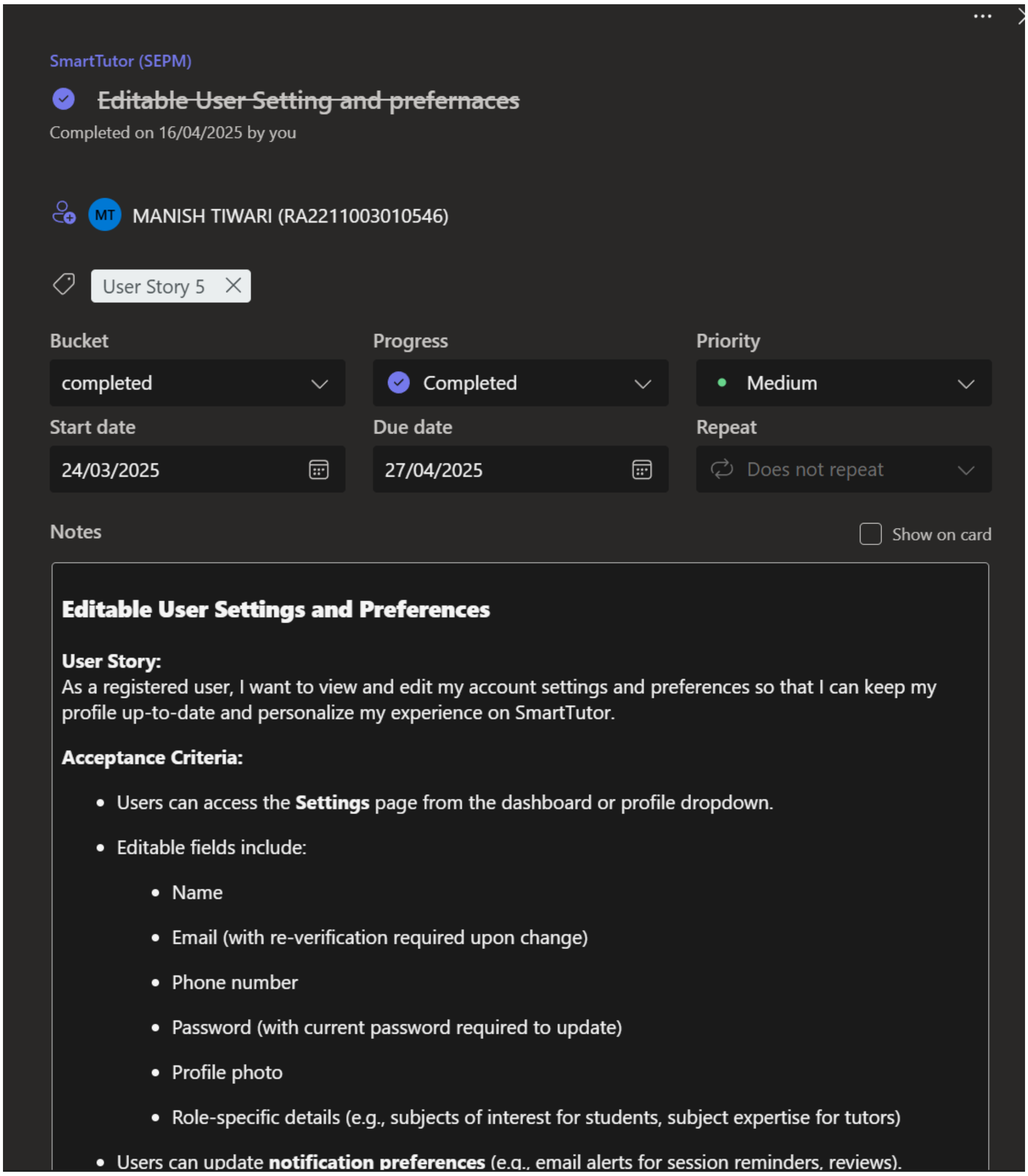
Landing Page UI / Easy Accessibility

User Story:
As a new or returning visitor to SmartTutor, I want a clean, intuitive, and accessible landing page so that I can quickly understand the platform's purpose and navigate to key features easily.

Acceptance Criteria:

- The landing page clearly displays:
  - A short description of what SmartTutor does.
  - Call-to-action (CTA) buttons for **"Find a Tutor"**, **"Become a Tutor"**, **"Login"**, and **"Register"**.
- Navigation bar includes easy access to:
  - Home
  - About Us
  - Contact
  - FAQs
  - Login/Register

Figure 2.4 User story for Landing Page and UI



SmartTutor (SEPM)

~~Editable User Setting and prefernaces~~

Completed on 16/04/2025 by you

MT MANISH TIWARI (RA2211003010546)

User Story 5

Bucket	Progress	Priority
completed	Completed	Medium

Start date	Due date	Repeat
24/03/2025	27/04/2025	Does not repeat

Notes

Show on card

Editable User Settings and Preferences

User Story:
As a registered user, I want to view and edit my account settings and preferences so that I can keep my profile up-to-date and personalize my experience on SmartTutor.

Acceptance Criteria:

- Users can access the **Settings** page from the dashboard or profile dropdown.
- Editable fields include:
  - Name
  - Email (with re-verification required upon change)
  - Phone number
  - Password (with current password required to update)
  - Profile photo
  - Role-specific details (e.g., subjects of interest for students, subject expertise for tutors)
- Users can update **notification preferences** (e.g., email alerts for session reminders, reviews).

Figure 2.5 User story for Editable User Settings

2.1.2 Functional Document

2.1.2.1 Introduction

The **SmartTutor (SEPM)** platform focuses on providing personalized, AI-powered tutor recommendations and seamless session booking functionality for students and tutors. This sprint is dedicated to implementing the core backend features, including user authentication, tutor profile management, and student-tutor matching based on preferences. Additionally, the system's scalability, security, and role-based access controls are established during this sprint.

2.1.2.2 Product Goal

The primary goal of this sprint is to develop a robust and secure tutor matching system that enables students to find suitable tutors based on their subject preferences, availability, and ratings. It also aims to allow tutors to manage their availability, subject expertise, and student reviews efficiently. The system should ensure quick and accurate matchmaking, maintaining high system reliability and responsiveness.

2.1.2.3 Demography (Users, Location)

- **Users:**
    - **Primary Users:** Students seeking tutoring services, Tutors offering their expertise in various subjects.
    - **Secondary Users:** Admins who manage the platform, ensuring secure user access and overseeing the overall system.
- **User Characteristics:**
    - **Students:** Looking for personalized tutoring based on their learning needs.
    - **Tutors:** Offering expertise in various subjects and setting their availability to suit students' schedules.
    - **Admins:** Ensuring smooth operation of the platform, managing users, and handling system security.
- **Location:**
    - **Target Locations:** Initially targeted at universities, schools, and tutoring centers, with potential for expansion to a broader audience globally.

2.1.2.4 Business Processes

The key business processes in the **SmartTutor** platform include:

- **User Registration and Authentication:**
    - Students and tutors can register securely using their email and create profiles.
    - Authentication ensures role-based access, where students can search for tutors and tutors can manage availability.
- **Tutor Search and Booking:**
    - Students can search for tutors based on subjects, availability, and ratings.
    - Tutors manage their profiles, subject expertise, and availability for bookings.

    - Real-time search and booking are facilitated by optimized API calls.
- **Session Feedback and Ratings:**
    - After each session, students can rate and review tutors, ensuring ongoing quality control and trust.
- **Admin Management:**
    - Admins monitor system usage, manage users, and ensure that the platform is functioning efficiently.

2.1.2.5 Features

This sprint focuses on implementing the following key features:

- **Feature 1: User Registration and Login**
    - **Description:** The system provides a secure registration and login mechanism for both students and tutors.
    - **User Story:** As a student, I want to log in securely with my credentials, so I can access the tutor matching system and manage my bookings.
- **Feature 2: Tutor Search and Booking**
    - **Description:** Students can search for tutors by subject and availability, and book sessions directly.
    - **User Story:** As a student, I want to find and book tutors quickly based on my preferences and subject needs.
- **Feature 3: Session Feedback and Ratings**
    - **Description:** After each session, students can leave feedback and rate tutors to help improve future recommendations.
    - **User Story:** As a student, I want to provide feedback and rate my tutor after the session, so future students can make informed decisions.

2.1.2.6. Authorization Matrix

Table 2.3 Access level Authorization Matrix

Role	**Access Level**
Admin	Full access to user management, platform settings, and reports.
Student	Access to profile creation, tutor search, session booking, and feedback.
Tutor	Access to profile creation, session management, availability setting, and ratings.

2.1.2.7 Assumptions

- The platform will be developed with secure data handling practices to ensure the protection of user information.
- The system is designed for scalability and can accommodate increasing numbers of users, sessions, and bookings.
- Tutors will be trained on the system's functionalities to ensure efficient profile management and student engagement.

2.1.3 Architecture Document

2.1.3.1 Application Architecture

- **Modular Design**: The SmartTutor platform follows a monolithic yet modular approach, where the frontend (React) and backend (Express.js) are tightly coupled. This architecture allows the system to remain flexible and scalable while simplifying the deployment process.
- **Frontend:**
  - The frontend is developed using React.js, leveraging React Router for efficient routing between different views (Login, Search, Profile, Booking).
  - Material UI or Bootstrap components are used for the user interface to ensure responsiveness and consistent design across devices.
- **Backend:**
  - The backend is built using Node.js and Express.js to manage API routes, handle authentication, and process booking requests.
  - JWT (JSON Web Tokens) is used for secure authentication to ensure that users only have access to authorized sections of the platform.
- **Database:**
  - The MongoDB database stores user profiles, booking data, and tutor reviews. Mongoose is used to model data and handle database operations.
  - Real-time bookings and session data are stored and retrieved via optimized queries using MongoDB's capabilities.

2.1.3.2 Microservices Approach

While not fully microservices-based, the system employs a modular approach:

- Frontend: Independently handles user interaction and communicates with the backend via RESTful APIs.

- Backend: Handles data processing, user management, and session bookings.
- Machine Learning: The AI model for tutor matching is encapsulated as a separate service, integrated into the backend API calls.

2.1.3.3 System Architecture Design

- The architecture is designed to handle dynamic user interactions with low latency.
- API Integration: Backend APIs are designed to handle requests such as user registration, booking, and feedback collection.
- Model Integration: A machine learning model is integrated to provide intelligent tutor matching based on user preferences, ensuring personalized experiences for students.

2.1.3.4 Deployment and Performance

- The system will be deployed using Docker containers for ease of development and testing.
- Performance optimizations include efficient database queries, caching mechanisms, and minimizing unnecessary API calls to enhance user experience.

2.1.3.2 Architecture Diagram-

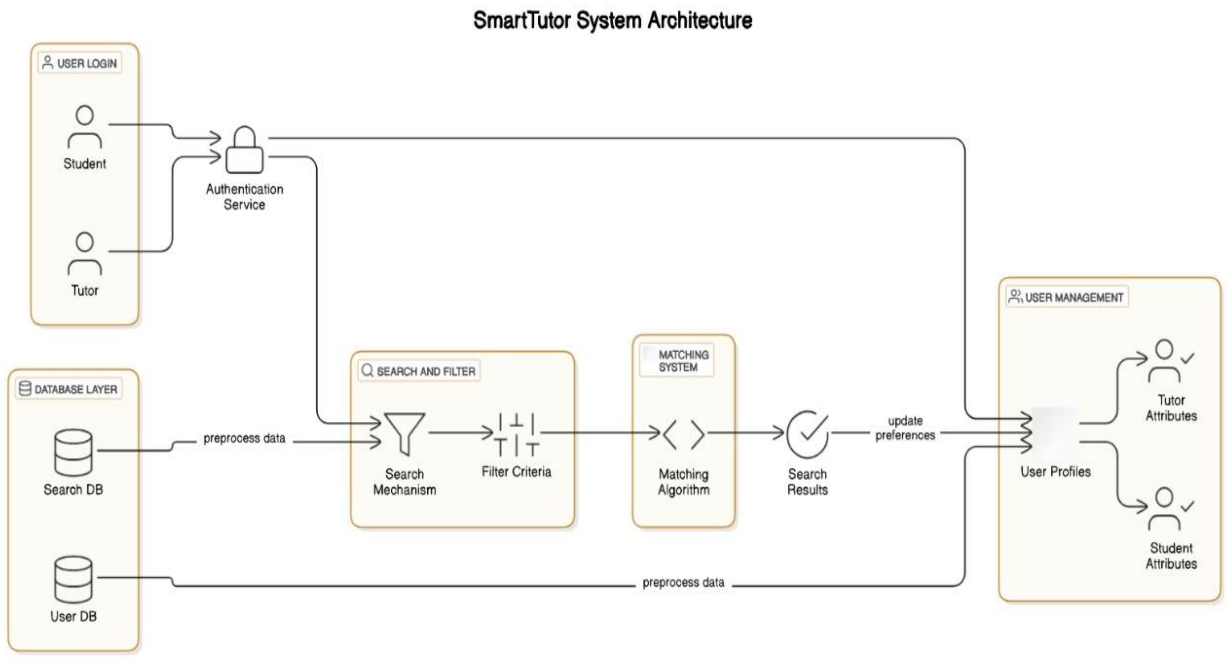


Figure 2.6 Architecture Diagram

This system architecture diagram of SmartTutor illustrates the entire flow from user login to tutor-student matching. It includes components such as authentication service, database layer, search and filter mechanisms, a matching system, and user profile management—enabling secure login, intelligent search, and profile-based matchmaking.

2.1.3.3. Data Exchange Contract:

Frequency of Data Exchanges:

Data exchanges are managed with a focus on both performance and timing, ensuring that the system provides real-time responses where necessary while also handling background tasks efficiently.

- **Real-time Tutor Search and Booking**: Triggered as soon as the student submits their search or booking request. This ensures immediate tutor suggestions and booking confirmations.
- **Session Feedback and Rating**: Processed in real-time once the student completes the tutoring session and submits feedback.
- **Report Export**: Students and tutors can download detailed session or profile reports (PDF or CSV). This happens after a user request, not in real-time, to avoid delays.
- **Logging**: Real-time logging is kept for user interactions and session management. Detailed logs of feedback, ratings, and booking histories are stored for analysis and auditing purposes.

Data Sets:

The **SmartTutor** platform handles several critical data sets with specific exchange requirements:

- **User Input Data**: This includes student details (name, email, profile), tutor profiles (subjects, availability, pricing), and session-related data (bookings, feedback).
- **Prediction Data**: The tutor matching system's output (best-matched tutor recommendations) is passed back to the frontend in real-time after processing.
- **Session History Data**: This is stored in a database for future reference, used to track the student’s learning progress, and for analytics like session frequency, review scores, and user engagement.

Mode of Exchanges (API, File, Queue, etc.):

Various methods are used for data exchange across the **SmartTutor** platform:

- **Frontend → Backend**: API calls via Express.js routes. For example, a student submits a search query for tutors, which the backend processes and returns matching tutors in real-time. Similarly, booking requests are submitted via forms and passed to the backend to be processed and stored.

- **Backend → Machine Learning Model**: Once the user has submitted their preferences or requested tutor recommendations, the backend calls an internal Python function (or machine learning model API) for data preprocessing and prediction. The model generates recommendations based on the inputs and returns them to the backend for further processing.
- **Machine Learning Model → Backend → Frontend**: The predictions (tutor recommendations) are sent from the model back to the backend and passed to the frontend via FastAPI routes. These predictions are displayed immediately on the user's interface.
- **Backend → Database**: During the training or updating phase of the model, **historical data** (including user ratings, feedback, and session history) is stored in a MongoDB database for use in future sessions. Logs are also recorded and stored locally using Python's logging module for future auditing.

2.1.4 UI DESIGN

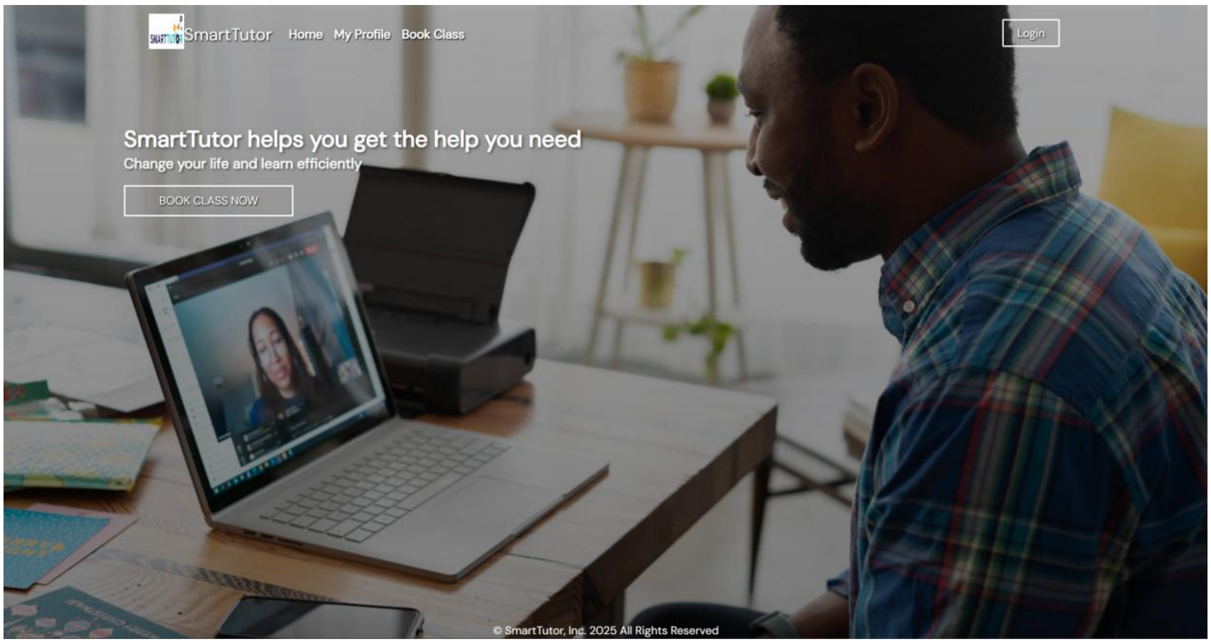

SmartTutor
Home
My Profile
Book Class
Login
SmartTutor helps you get the help you need
Change your life and learn efficiently
BOOK CLASS NOW
© SmartTutor, Inc. 2025 All Rights Reserved

Figure 2.7 UI Design for Landing page

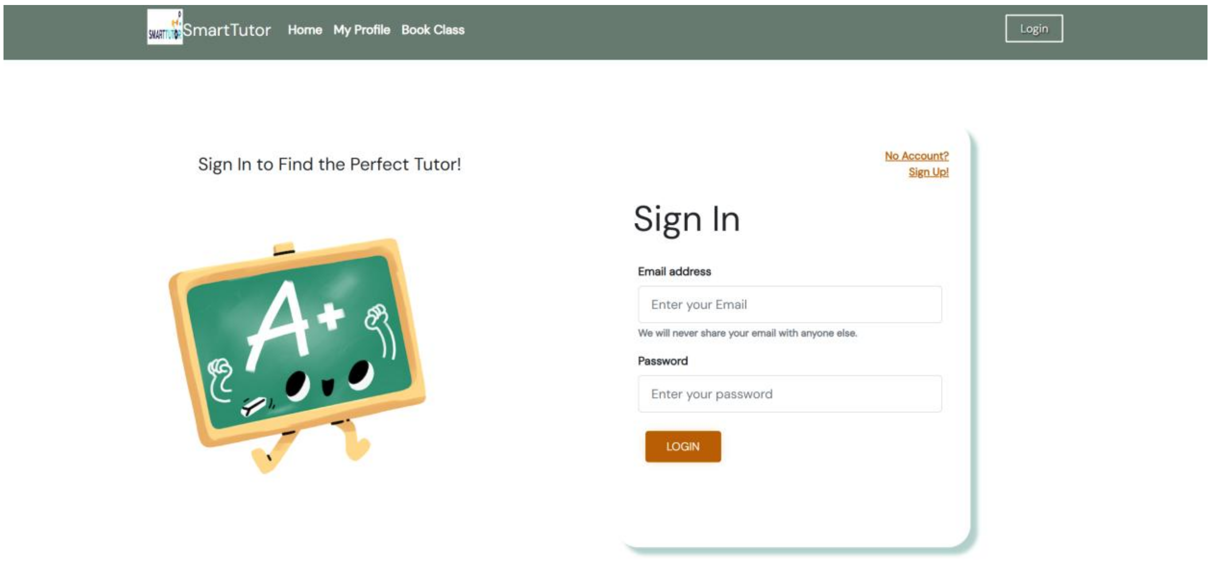


Figure 2.8 Sign In Page

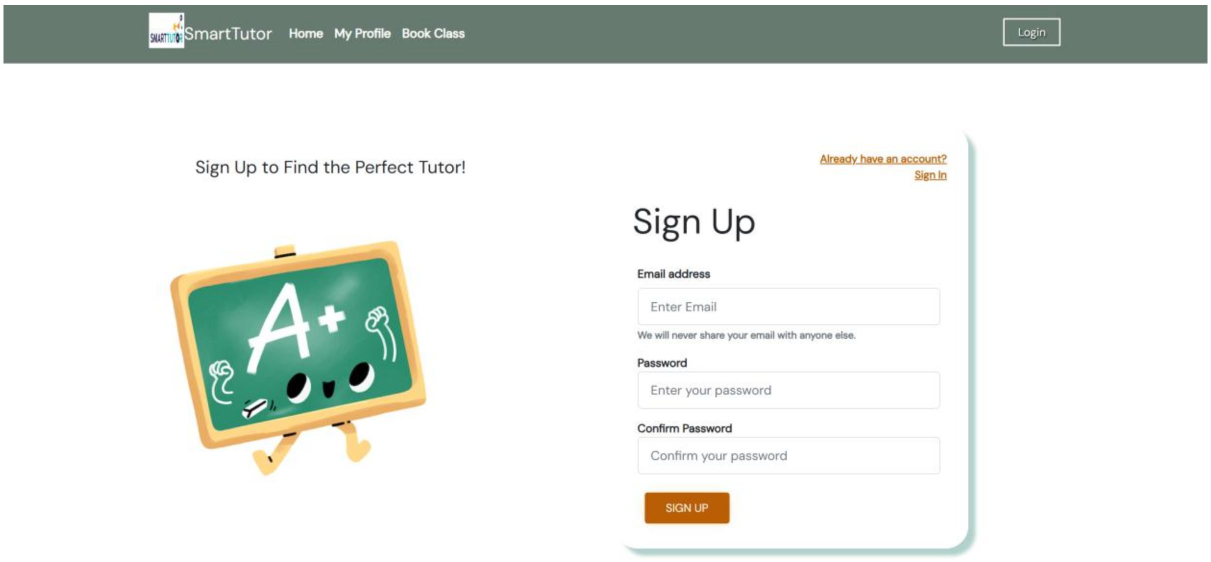


Figure 2.9 Sign Up Page

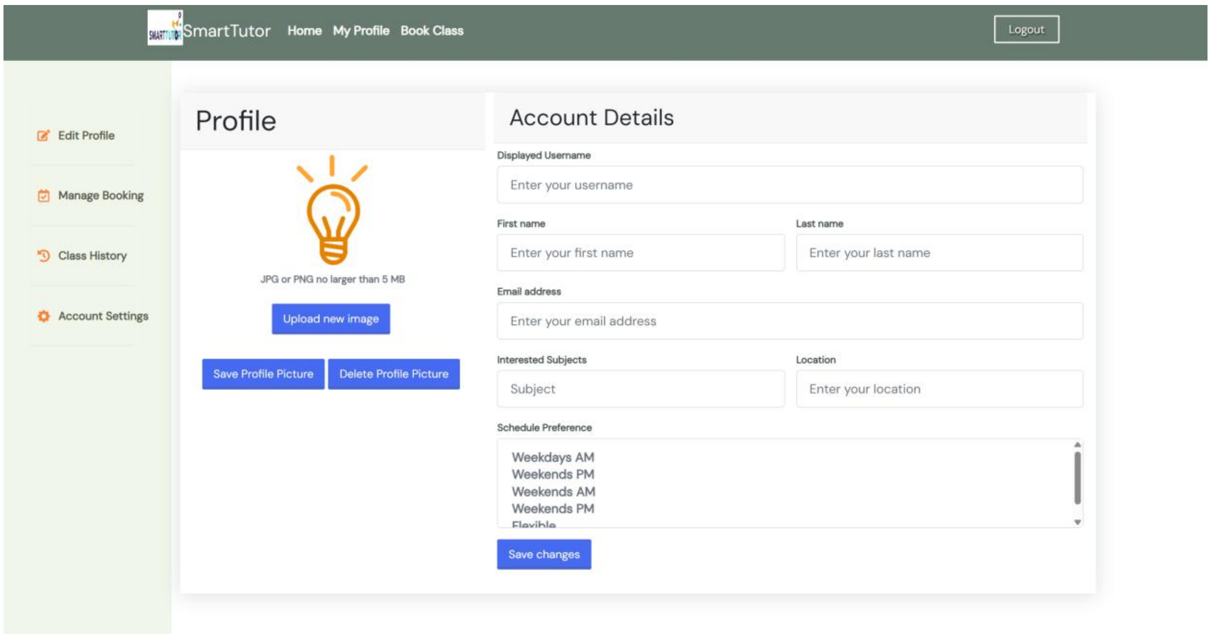
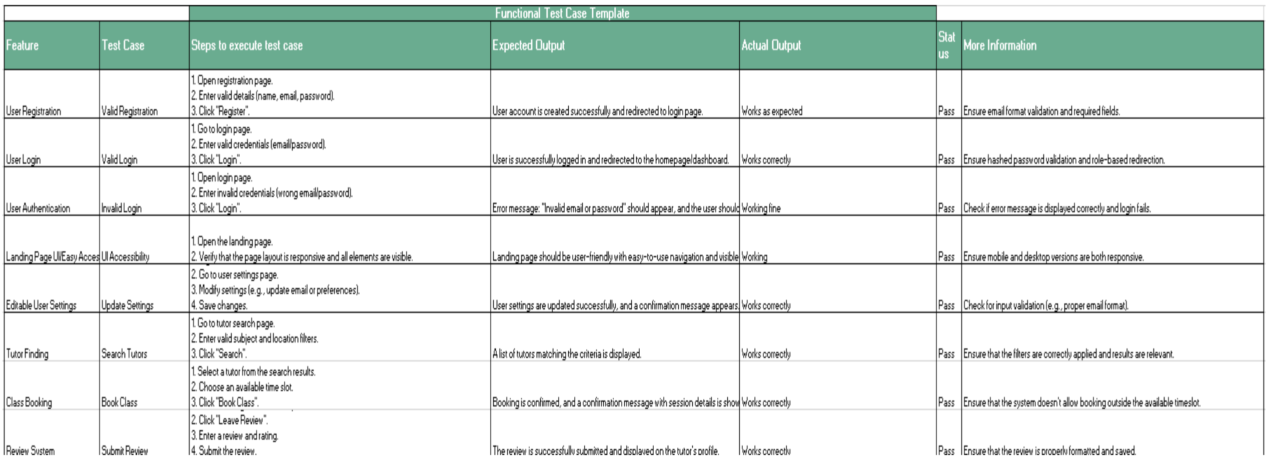


Figure 2.10 Edit Profile Section

2.1.5 Functional Test Cases



		Functional Test Case Template				
Feature	Test Case	Steps to execute test case	Expected Output	Actual Output	Status	More Information
User Registration	Valid Registration	1. Open registration page. 2. Enter valid details (name, email, password). 3. Click "Register".	User account is created successfully and redirected to login page.	Works as expected	Pass	Ensure email format validation and required fields.
User Login	Valid Login	1. Go to login page. 2. Enter valid credentials (email/password). 3. Click "Login".	User is successfully logged in and redirected to the homepage/dashboard.	Works correctly	Pass	Ensure hashed password validation and role-based redirection.
User Authentication	Invalid Login	1. Open login page. 2. Enter invalid credentials (wrong email/password). 3. Click "Login".	Error message: "Invalid email or password" should appear, and the user should	Working fine	Pass	Check if error message is displayed correctly and login fails.
Landing Page UI/Easy Acces	UI Accessibility	1. Open the landing page. 2. Verify that the page layout is responsive and all elements are visible.	Landing page should be user-friendly with easy-to-use navigation and visible	Working	Pass	Ensure mobile and desktop versions are both responsive.
Editable User Settings	Update Settings	2. Go to user settings page. 3. Modify settings (e.g., update email or preferences). 4. Save changes.	User settings are updated successfully, and a confirmation message appears.	Works correctly	Pass	Check for input validation (e.g., proper email format).
Tutor Finding	Search Tutors	1. Go to tutor search page. 2. Enter valid subject and location filters. 3. Click "Search".	A list of tutors matching the criteria is displayed.	Works correctly	Pass	Ensure that the filters are correctly applied and results are relevant.
Class Booking	Book Class	1. Select a tutor from the search results. 2. Choose an available time slot. 3. Click "Book Class".	Booking is confirmed, and a confirmation message with session details is show	Works correctly	Pass	Ensure that the system doesn't allow booking outside the available timeslot.
Review System	Submit Review	2. Click "Leave Review". 3. Enter a review and rating. 4. Submit the review.	The review is successfully submitted and displayed on the tutor's profile.	Works correctly	Pass	Ensure that the review is properly formatted and saved.

Table 2.4 Detailed Functional Test Case

2.1.7 Committed Vs Completed User Stories

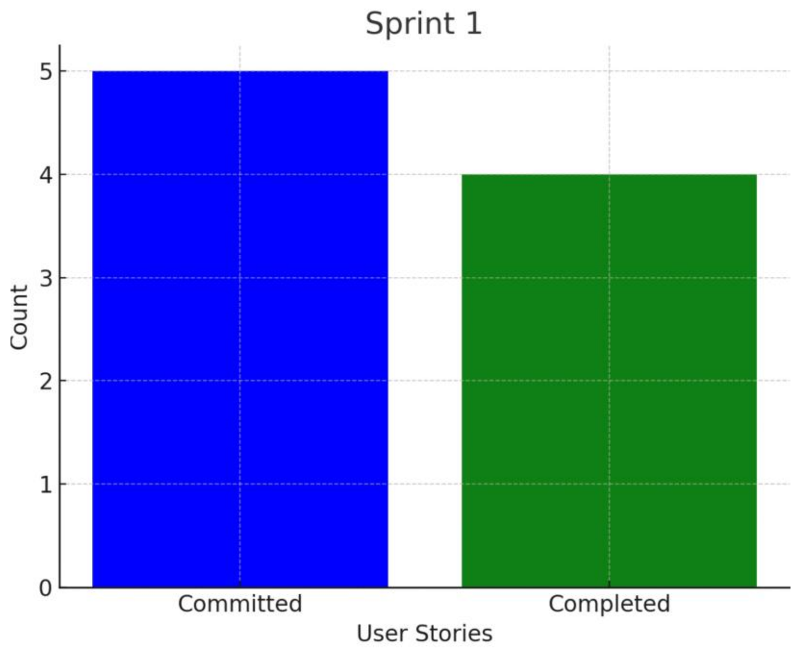
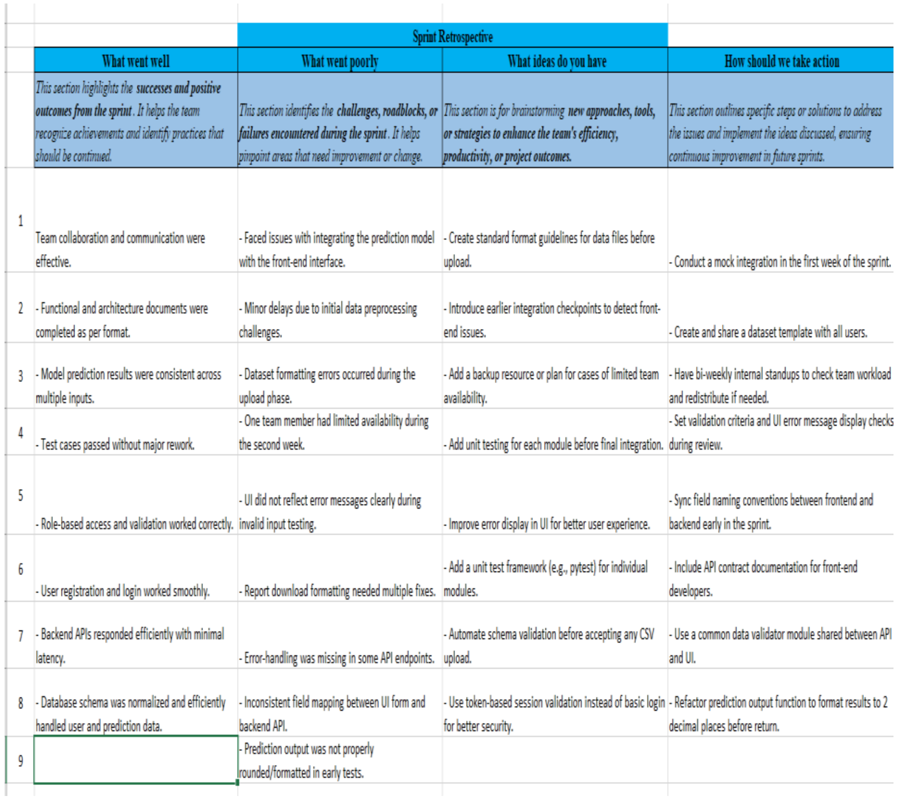


Figure 2.11 Bar graph for Committed Vs Completed User Stories

2.1.8 Sprint Retrospective



	Sprint Retrospective			
	What went well	**What went poorly**	**What ideas do you have**	**How should we take action**
	This section highlights the ***successes and positive outcomes from the sprint****. It helps the team recognize achievements and identify practices that should be continued.*	*This section identifies the* ***challenges, roadblocks, or failures encountered during the sprint****. It helps pinpoint areas that need improvement or change.*	*This section is for brainstorming* ***new approaches, tools, or strategies to enhance the team's efficiency, productivity, or project outcomes.***	*This section outlines specific steps or solutions to address the issues and implement the ideas discussed, ensuring continuous improvement in future sprints.*
1	Team collaboration and communication were effective.	- Faced issues with integrating the prediction model with the front-end interface.	- Create standard format guidelines for data files before upload.	- Conduct a mock integration in the first week of the sprint.
2	- Functional and architecture documents were completed as per format.	- Minor delays due to initial data preprocessing challenges.	- Introduce earlier integration checkpoints to detect front-end issues.	- Create and share a dataset template with all users.
3	- Model prediction results were consistent across multiple inputs.	- Dataset formatting errors occurred during the upload phase.	- Add a backup resource or plan for cases of limited team availability.	- Have bi-weekly internal standups to check team workload and redistribute if needed.
4	- Test cases passed without major rework.	- One team member had limited availability during the second week.	- Add unit testing for each module before final integration.	- Set validation criteria and UI error message display checks during review.
5	- Role-based access and validation worked correctly.	- UI did not reflect error messages clearly during invalid input testing.	- Improve error display in UI for better user experience.	- Sync field naming conventions between frontend and backend early in the sprint.
6	- User registration and login worked smoothly.	- Report download formatting needed multiple fixes.	- Add a unit test framework (e.g., pytest) for individual modules.	- Include API contract documentation for front-end developers.
7	- Backend APIs responded efficiently with minimal latency.	- Error-handling was missing in some API endpoints.	- Automate schema validation before accepting any CSV upload.	- Use a common data validator module shared between API and UI.
8	- Database schema was normalized and efficiently handled user and prediction data.	- Inconsistent field mapping between UI form and backend API.	- Use token-based session validation instead of basic login for better security.	- Refactor prediction output function to format results to 2 decimal places before return.
9		- Prediction output was not properly rounded/formatted in early tests.		

Figure 2.12 Sprint Retrospective for the Sprint 1

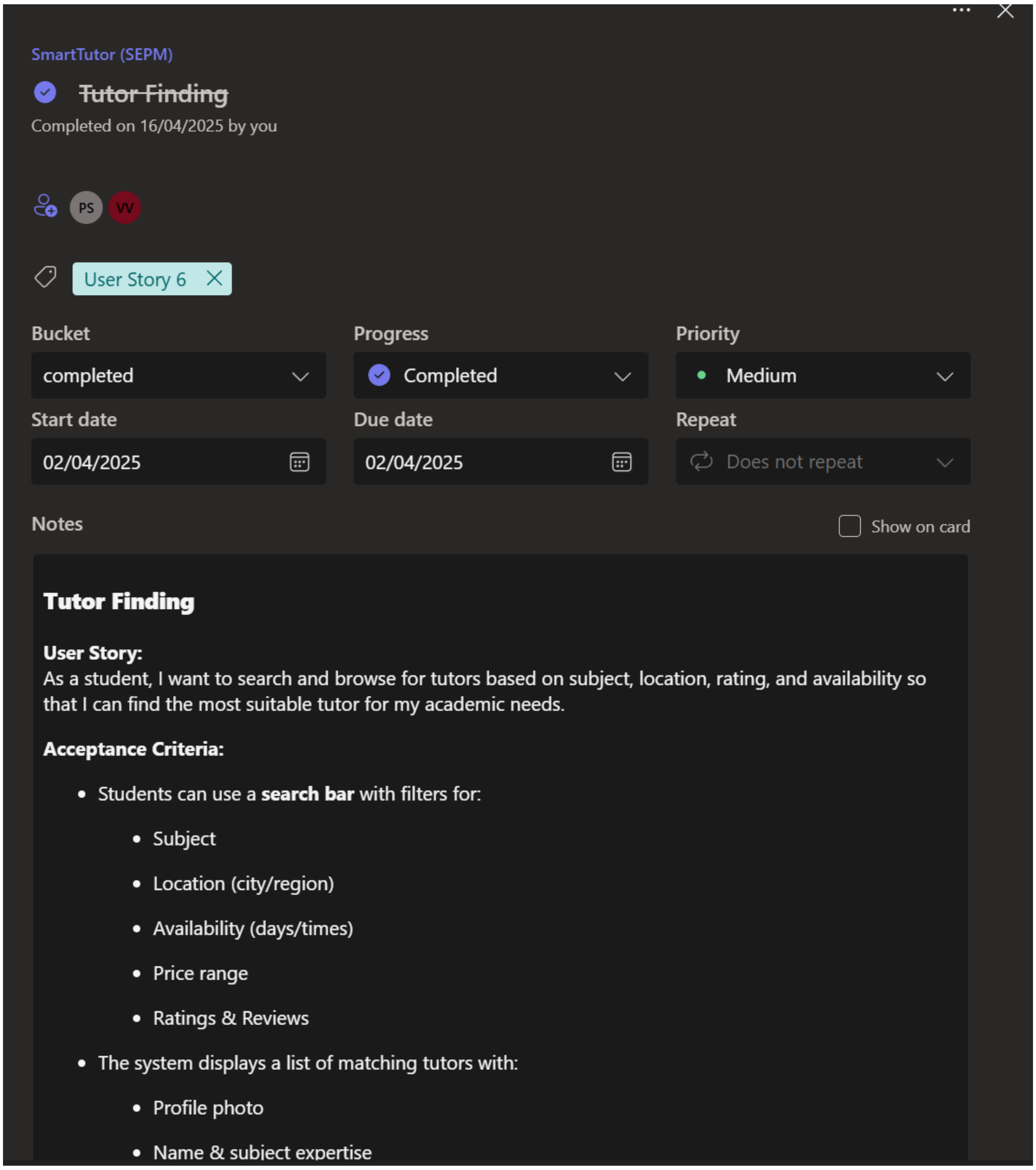
2.2 SPRINT 2

2.2.1 Sprint Goal with User Stories of Sprint 2

The goal of **Sprint 2** is to enhance the **SmartTutor** platform by adding advanced features such as real-time tutor feedback, session analytics, and automated session reporting. This sprint focuses on improving the system's transparency, user engagement, and providing analytics for both students and tutors. Additionally, the sprint will emphasize platform usability improvements and enhance the interpretability of tutor recommendations.

Table 2.5 Detailed User Stories of sprint 2

S.NO		**Detailed User Stories**
US #5		**As a tutor**, I want to analyze trends in student feedback and session ratings, so that I can refine my teaching strategies and improve my performance.
US #6		**As a student**, I want to track the status of my tutor session bookings in real-time, so that I can stay informed about upcoming sessions or changes.
US #7		**As an admin**, I want to audit tutor-student interactions, bookings, and session feedback, so that I can ensure fairness and adherence to platform guidelines.
US #8		**As a data scientist**, I want to analyze and refine the AI model used for tutor matching, so that I can improve the accuracy and relevance of tutor recommendations.
US #9		**As a product owner**, I want to track platform usage trends and identify areas for improvement, so that I can make informed decisions about product features and future development.


SmartTutor (SEPM)

~~Tutor Finding~~

Completed on 16/04/2025 by you

PS VV

User Story 6

Bucket: completed

Progress: Completed

Priority: Medium

Start date: 02/04/2025

Due date: 02/04/2025

Repeat: Does not repeat

Notes

Show on card

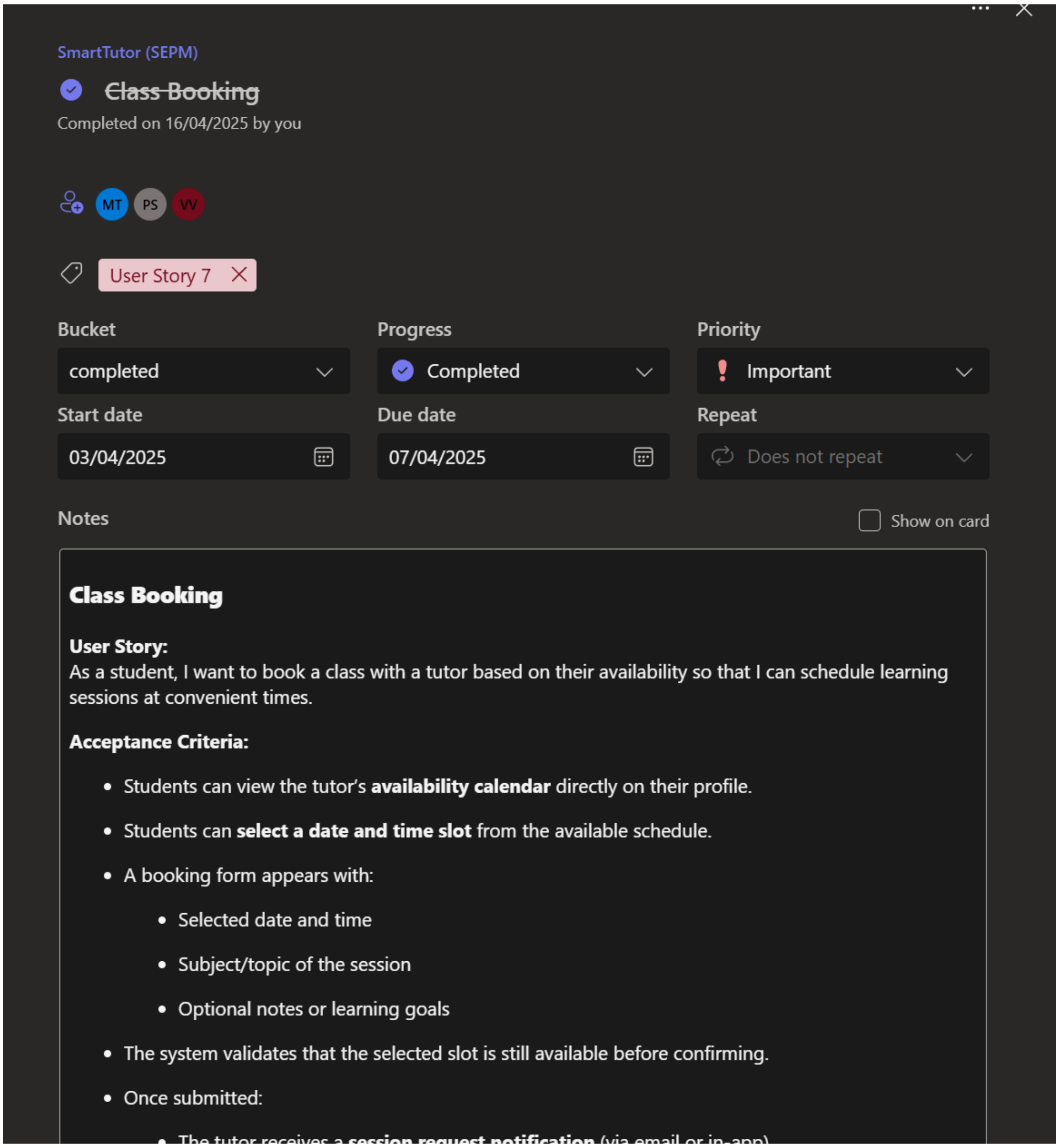
Tutor Finding

User Story:
As a student, I want to search and browse for tutors based on subject, location, rating, and availability so that I can find the most suitable tutor for my academic needs.

Acceptance Criteria:

- Students can use a **search bar** with filters for:
    - Subject
    - Location (city/region)
    - Availability (days/times)
    - Price range
    - Ratings & Reviews
- The system displays a list of matching tutors with:
    - Profile photo
    - Name & subject expertise

Figure 2.13 User story for Tutor Finder



SmartTutor (SEPM)

~~Class Booking~~

Completed on 16/04/2025 by you

MT PS VV

User Story 7

Bucket: completed

Progress: Completed

Priority: Important

Start date: 03/04/2025

Due date: 07/04/2025

Repeat: Does not repeat

Notes

Show on card

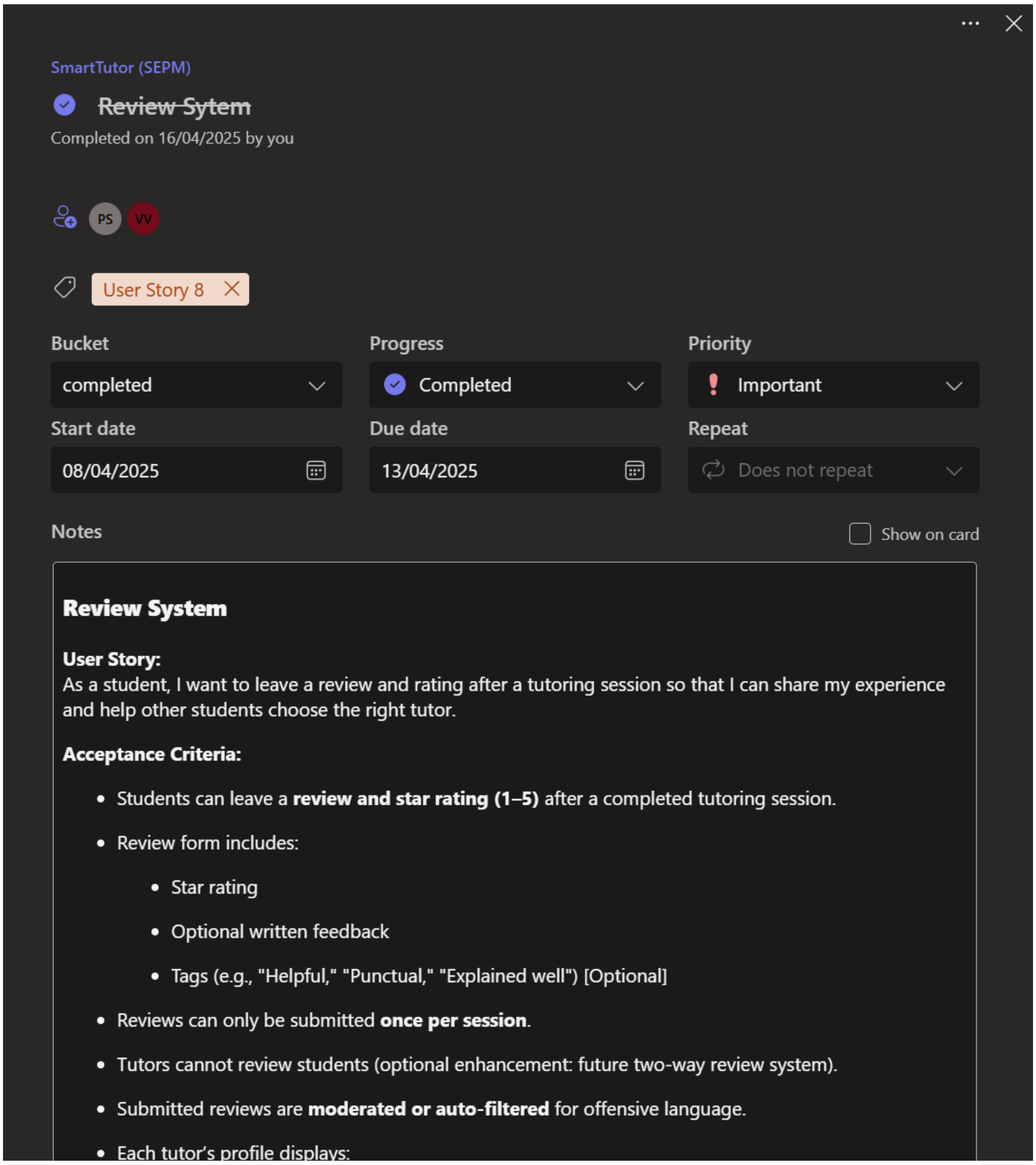
Class Booking

User Story:
As a student, I want to book a class with a tutor based on their availability so that I can schedule learning sessions at convenient times.

Acceptance Criteria:

- Students can view the tutor's **availability calendar** directly on their profile.
- Students can **select a date and time slot** from the available schedule.
- A booking form appears with:
  - Selected date and time
  - Subject/topic of the session
  - Optional notes or learning goals
- The system validates that the selected slot is still available before confirming.
- Once submitted:
  - The tutor receives a **session request notification** (via email or in-app)

Figure 2.14 User story for ClassBooking



SmartTutor (SEPM)

~~Review Sytem~~

Completed on 16/04/2025 by you

PS VV

User Story 8

Bucket	Progress	Priority
completed	Completed	Important
Start date	**Due date**	**Repeat**
08/04/2025	13/04/2025	Does not repeat

Notes

Show on card

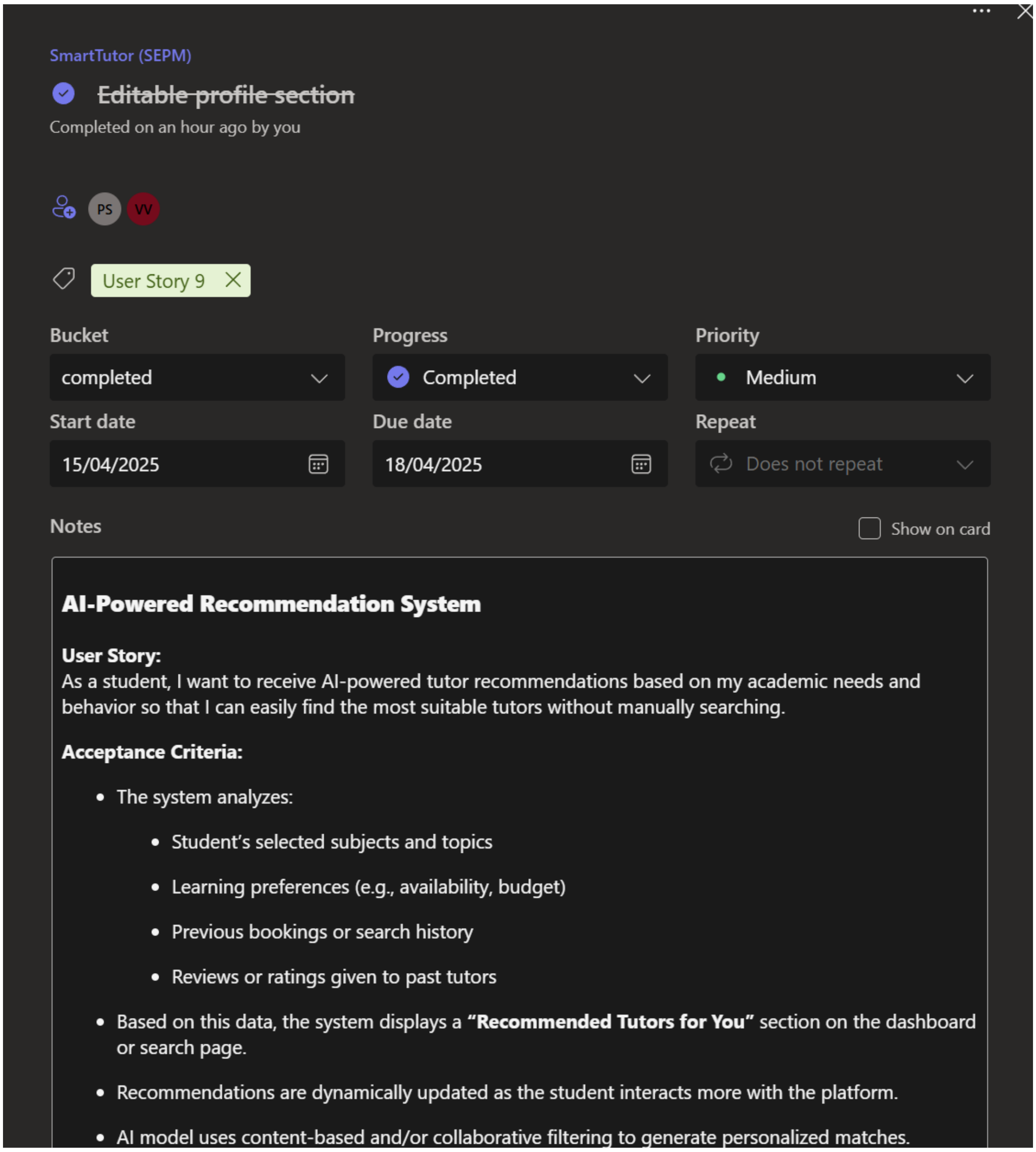
Review System

User Story:
As a student, I want to leave a review and rating after a tutoring session so that I can share my experience and help other students choose the right tutor.

Acceptance Criteria:

- Students can leave a **review and star rating (1–5)** after a completed tutoring session.
- Review form includes:
  - Star rating
  - Optional written feedback
  - Tags (e.g., "Helpful," "Punctual," "Explained well") [Optional]
- Reviews can only be submitted **once per session**.
- Tutors cannot review students (optional enhancement: future two-way review system).
- Submitted reviews are **moderated or auto-filtered** for offensive language.
- Each tutor's profile displays:

Figure 2.15 User story for Review System


SmartTutor (SEPM)

~~Editable profile section~~

Completed on an hour ago by you

User Story 9

Bucket	Progress	Priority
completed	Completed	Medium
Start date	**Due date**	**Repeat**
15/04/2025	18/04/2025	Does not repeat

Notes

Show on card

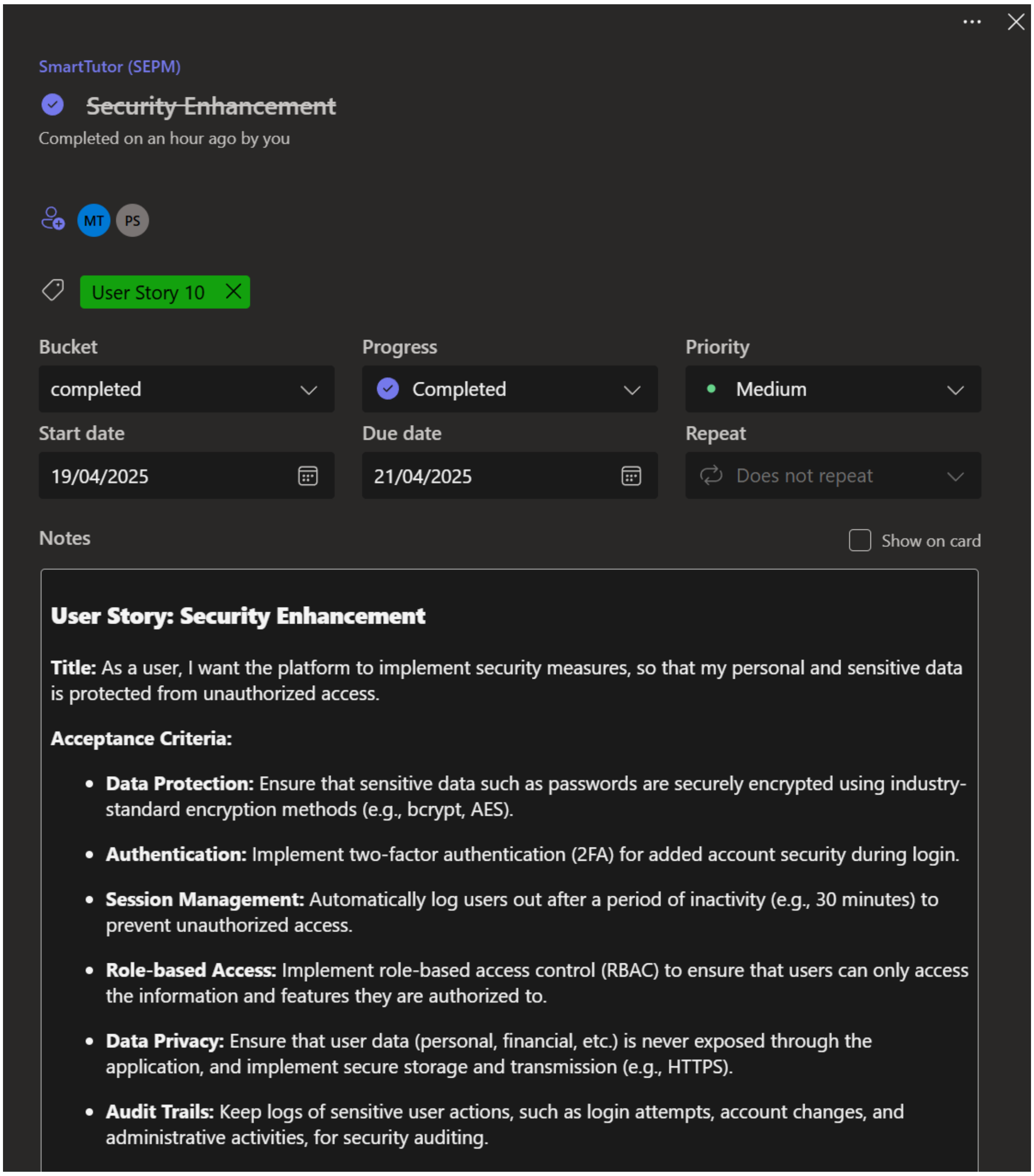
AI-Powered Recommendation System

User Story:
As a student, I want to receive AI-powered tutor recommendations based on my academic needs and behavior so that I can easily find the most suitable tutors without manually searching.

Acceptance Criteria:

- The system analyzes:
  - Student's selected subjects and topics
  - Learning preferences (e.g., availability, budget)
  - Previous bookings or search history
  - Reviews or ratings given to past tutors
- Based on this data, the system displays a **"Recommended Tutors for You"** section on the dashboard or search page.
- Recommendations are dynamically updated as the student interacts more with the platform.
- AI model uses content-based and/or collaborative filtering to generate personalized matches.

Figure 2.16 User story for Editable profile Section



SmartTutor (SEPM)

~~Security Enhancement~~

Completed on an hour ago by you

MT PS

User Story 10

Bucket: completed

Progress: Completed

Priority: Medium

Start date: 19/04/2025

Due date: 21/04/2025

Repeat: Does not repeat

Notes

Show on card

User Story: Security Enhancement

Title: As a user, I want the platform to implement security measures, so that my personal and sensitive data is protected from unauthorized access.

Acceptance Criteria:

- **Data Protection:** Ensure that sensitive data such as passwords are securely encrypted using industry-standard encryption methods (e.g., bcrypt, AES).
- **Authentication:** Implement two-factor authentication (2FA) for added account security during login.
- **Session Management:** Automatically log users out after a period of inactivity (e.g., 30 minutes) to prevent unauthorized access.
- **Role-based Access:** Implement role-based access control (RBAC) to ensure that users can only access the information and features they are authorized to.
- **Data Privacy:** Ensure that user data (personal, financial, etc.) is never exposed through the application, and implement secure storage and transmission (e.g., HTTPS).
- **Audit Trails:** Keep logs of sensitive user actions, such as login attempts, account changes, and administrative activities, for security auditing.

Figure 2.17 User story for Security Enhancement

2.2.2.1 Introduction

Sprint 2 of the SmartTutor platform focuses on enhancing the system's transparency, user engagement, and decision-making support. Key improvements include real-time session feedback, enhanced tutor-student interaction, advanced analytics, and a more intuitive user interface. This sprint aims to improve trust, usability, and insights from the tutor matching system for different stakeholders, including students, tutors, and admins.

2.2.2.2 Product Goal

The goal of Sprint 2 is to provide a more intelligent, transparent, and user-centric tutor matching platform. It will offer detailed insights into tutor recommendations, enable real-time tracking of session status, and provide comprehensive reporting and analytics tools for students, tutors, and admins. This will support data-driven learning, improve transparency, and enhance user engagement.

2.2.2.3 Demography (Users, Location)

Users:

Primary Users: Students, Tutors, Admins

Secondary Users: Parents and guardians (indirect users seeking educational tools), Researchers (using the platform for study).

User Characteristics:

Students: Looking for personalized tutoring experiences based on their learning needs.

Tutors: Offering their expertise in various subjects and managing availability.

Admins: Overseeing platform security, managing user data, and handling system configuration.

Location:

Target Locations: Initially targeted at educational institutions, schools, and tutoring centers with potential for expansion to a global audience.

2.2.2.4 Business Processes

The enhanced business processes in Sprint 2 include:

Factor Explanation & Interpretability:

Students and tutors can view key factors behind tutor recommendations (e.g., subject expertise, rating, availability) to help ensure transparency in tutor-student matching.

Session Tracking and Feedback:

Students and tutors can track the status of their upcoming or completed sessions in realtime, reducing the need for direct contact and ensuring a seamless experience.

Audit and Transparency:

Admins can audit system decisions, monitor tutor-student interactions, and ensure platform compliance with operational standards.

Analytics for Tutors and Students:

Students and tutors can access detailed reports and analytics that show performance, review trends, and engagement metrics.

2.2.2.5 Features

This sprint focuses on implementing the following key features:

Feature 1: Explainable Tutor Recommendations

Description: Students can explore the contributing factors (e.g., ratings, availability, expertise) for each tutor recommendation using a transparent recommendation system.

User Story: As a student, I want to understand why a particular tutor is recommended to me, so that I can make informed decisions.

Feature 2: Real-Time Session Tracking

Description: Students and tutors can track the status of their booked sessions, including upcoming, completed, and canceled sessions.

User Story: As a student, I want to track my session status in real-time so that I can stay informed about upcoming sessions without contacting support.

Feature 3: Audit Trail and Feedback Logs

Description: Admins can access logs of tutor-student interactions, session feedback, and system decisions to ensure operational compliance.

User Story: As an admin, I want to audit feedback and session logs to ensure that the platform is being used appropriately and fairly.

Feature 4: Advanced Reporting & Analytics for Tutors

Description: Tutors can access detailed reports on their session ratings, student feedback, and performance trends.

User Story: As a tutor, I want to track my performance over time and analyze feedback trends to improve my teaching methods.

2.2.2.6 Authorization Matrix

Table 2.6 Access level Authorization Matrix (sprint 2)

Role	**Access Level**
Admin	Full access to all user data, feedback logs, and system configurations.
Student	Access to session booking, feedback submission, and performance tracking.
Tutor	Access to session management, feedback analysis, and performance reports.
Admin	Full access to the system, including user management (students, tutors), access to all reports and analytics, session data, and platform configurations. Admins can also generate system-wide activity reports and manage content.
Researcher	Access to anonymized data for research purposes, with restricted access to user identities.

2.2.2.7 Assumptions

- The platform will implement secure data handling practices, ensuring all user data is anonymized and protected.
- Real-time session tracking is achieved through efficient backend integration, ensuring no delays or performance issues.
- Admins and tutors will be provided with intuitive dashboards to view and manage data insights effectively.
- The feedback and reporting features will be used regularly by tutors and students to improve performance and engagement.

2.1.3.2 UML Diagram –

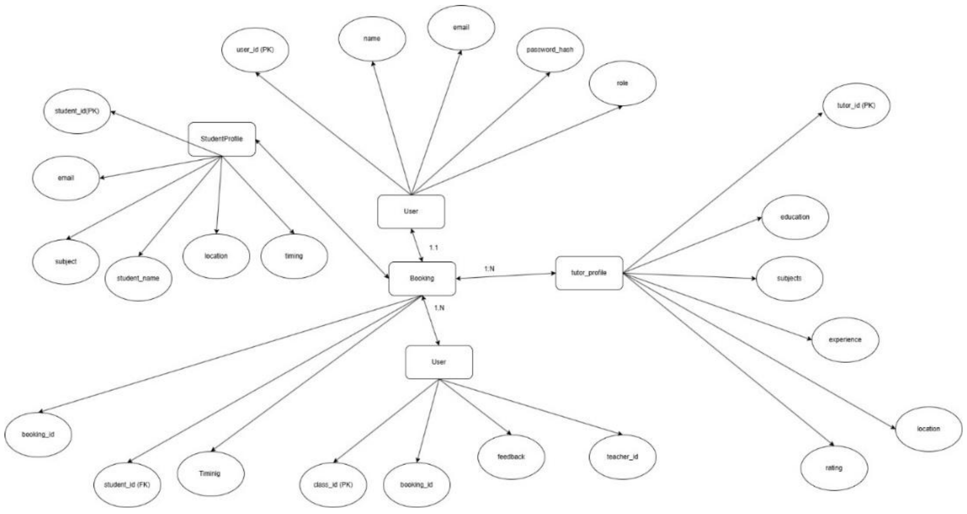


Fig 2.18 – ER Diagram

This ER diagram represents the database structure for the SmartTutor platform. It models entities like User, StudentProfile, Tutor_Profile, and Booking, showing relationships between students, tutors, and class bookings, with relevant attributes such as timing, subjects, feedback, and ratings.

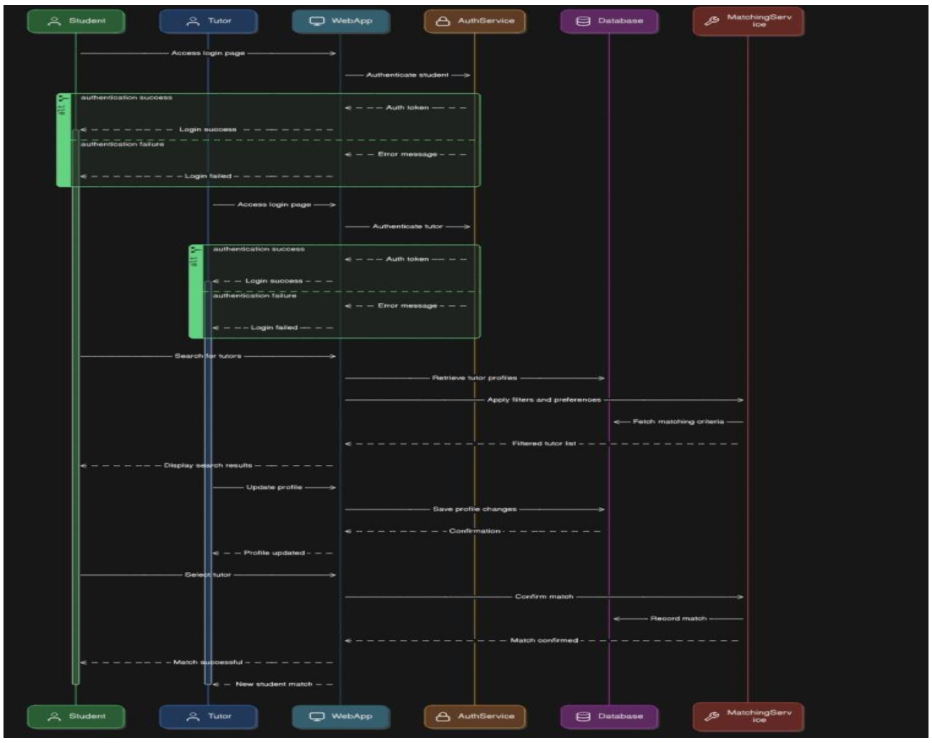


Fig 2.19 – Sequence Diagram

This sequence diagram illustrates the user interaction flow in the SmartTutor platform. It shows how students and tutors authenticate via the AuthService, search for tutors using filters, and how matches are processed and confirmed through the WebApp, Database, and MatchingService.

2.2.3.1 Application Architecture

- Modular Design**:** The system follows a monolithic yet modular architecture, where the frontend and backend are tightly integrated but each component has distinct responsibilities. The React frontend communicates with the Express.js backend, which in turn handles data processing, predictions, and session tracking.
- Frontend: The frontend is built with React.js and utilizes React Router for page navigation. Dynamic, state-based UI elements allow users to interact with their profile, book sessions, and view tutor details and ratings.
- Backend: The backend is powered by Node.js and Express.js, serving as the intermediary for managing user authentication, session data, and feedback. JWT (JSON Web Tokens) are used for secure login and user authentication.
- Database: MongoDB serves as the database for storing user profiles, session data, feedback, and performance metrics. Mongoose is used to handle schema definitions and data operations.
- Machine Learning Integration: The tutor recommendation system uses a simple machine learning model to suggest tutors based on student preferences. The model is integrated into the backend and is invoked whenever a student performs a search.

2.2.3.2 Microservices Approach

While not fully microservices-based, the system adopts a modular approach with clear responsibility separation:

- **Frontend:** Independent component responsible for handling user interactions, form submissions, and presenting data.
- **Backend:** Manages API calls, authentication, session data, and feedback analysis.
- **Machine Learning:** The model for tutor recommendation is encapsulated as a separate service, with its own set of logic for making predictions.

2.2.3.3 System Architecture Design

- The system is designed to handle multiple requests concurrently, ensuring responsiveness even with a large number of active users.
- **API Integration:** The frontend interacts with backend APIs, such as tutor search, feedback submission, and profile management.
- **Database Integration:** Data is queried and updated in real-time, ensuring that student and tutor profiles are always up to date.

2.2.3.4 Deployment and Performance

- The system will be deployed using Docker containers to simplify development and testing across environments.
- Performance optimizations include caching, query indexing, and reducing unnecessary backend calls to ensure fast response times.

Frequency of Data Exchanges:

Data exchanges are handled with a focus on performance and timing, ensuring a seamless user experience. Below are the main data exchange operations:

- Real-time Tutor Search and Booking: Triggered immediately when a student submits their search query or booking request. The backend processes the query and returns tutor recommendations or session confirmation in real-time.
- Session Feedback Submission: Upon completing a session, students can provide feedback, which is then processed and stored instantly.
- Report Export: Once a student or tutor requests a session report, the system generates and exports it in PDF/CSV format.
- Logging: Logs are maintained for system activities, user interactions, and feedback to ensure accountability and traceability. Logs are recorded in the backend and stored locally for debugging and future analysis.

Data Sets:

The SmartTutor platform handles several key datasets, each with specific exchange requirements:

- User Input Data: Data entered by students and tutors, such as names, emails, subjects, availability, and session preferences.
- Recommendation Data: Tutor recommendations returned by the machine learning model, based on student preferences, subject, and availability.
- Session Data: Data regarding past sessions, feedback, reviews, and ratings submitted by students for tutors.
- Historical Data: This includes user profiles, session histories, feedback, and performance metrics stored in the database, used for further analytics or refining the recommendation model.

Mode of Exchanges (API, File, Queue, etc.):

The platform employs various methods for data exchange across its components:

- Frontend → Backend: API calls via Express.js when a student performs a search for a tutor or submits a booking. Data is passed from the frontend to the backend in real-time via RESTful APIs.
- Backend → Model: When a student submits a search query, the backend invokes an internal Python function (or external machine learning API) to process the data and generate recommendations. This includes preprocessing the student's preferences and retrieving the most suitable tutors based on the trained model.
- Model → Backend → Frontend: Once the model processes the request, the prediction results (i.e., tutor recommendations) are passed back from the backend to the frontend via FastAPI routes, where they are displayed to the user.
- Model → Database (Training Phase Only): During the model training or updates, historical data (e.g., tutor performance, student feedback) is logged using Python's logging module and stored locally as .log files for future analysis and performance tuning.

Security & Compliance Considerations:

- Data privacy is ensured by securely encrypting user data, such as personal information and session histories, stored in the MongoDB database.
- JWT tokens are used to authenticate users and ensure secure access to different sections of the platform, based on roles (student, tutor, admin).

Data Flow Diagram:

Below is a simple outline of the data flow:

1. Frontend (React) → Backend (Express.js):
    - Data submission (e.g., user registration, search query) via API requests.
2. Backend → Machine Learning Model (Python):
    - Data processed for tutor recommendation predictions.
3. Backend → Frontend:
    - Real-time display of recommendations and session details.
4. Backend → Database (MongoDB):
    - Store session data, user feedback, and history.

2.2.4 UI Design

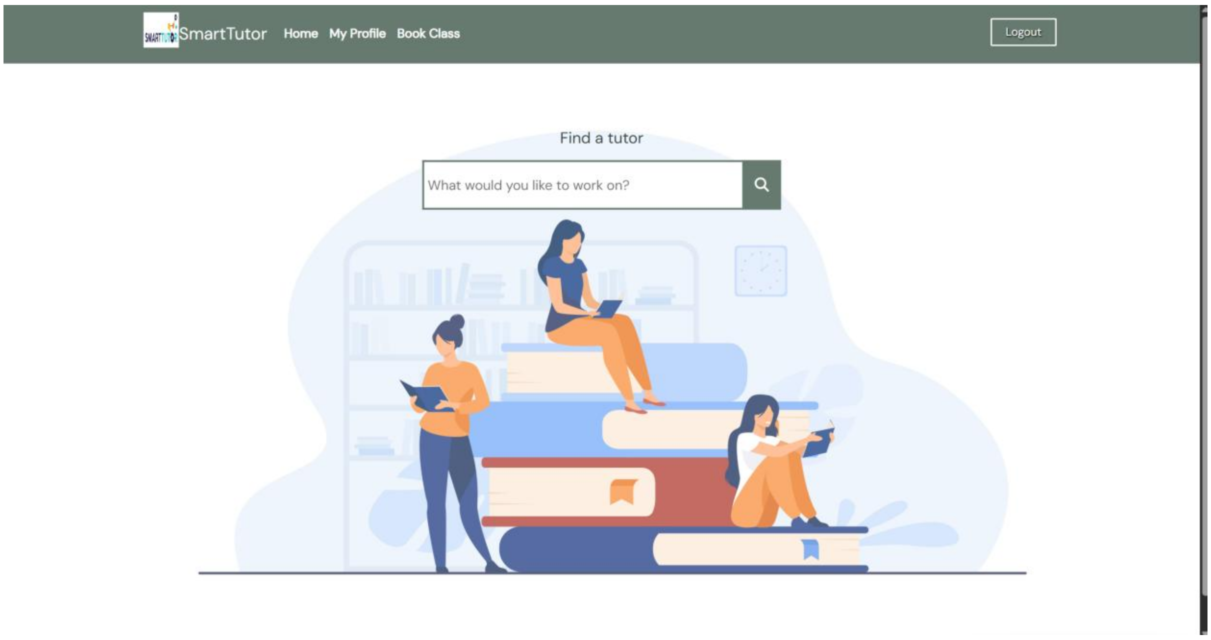
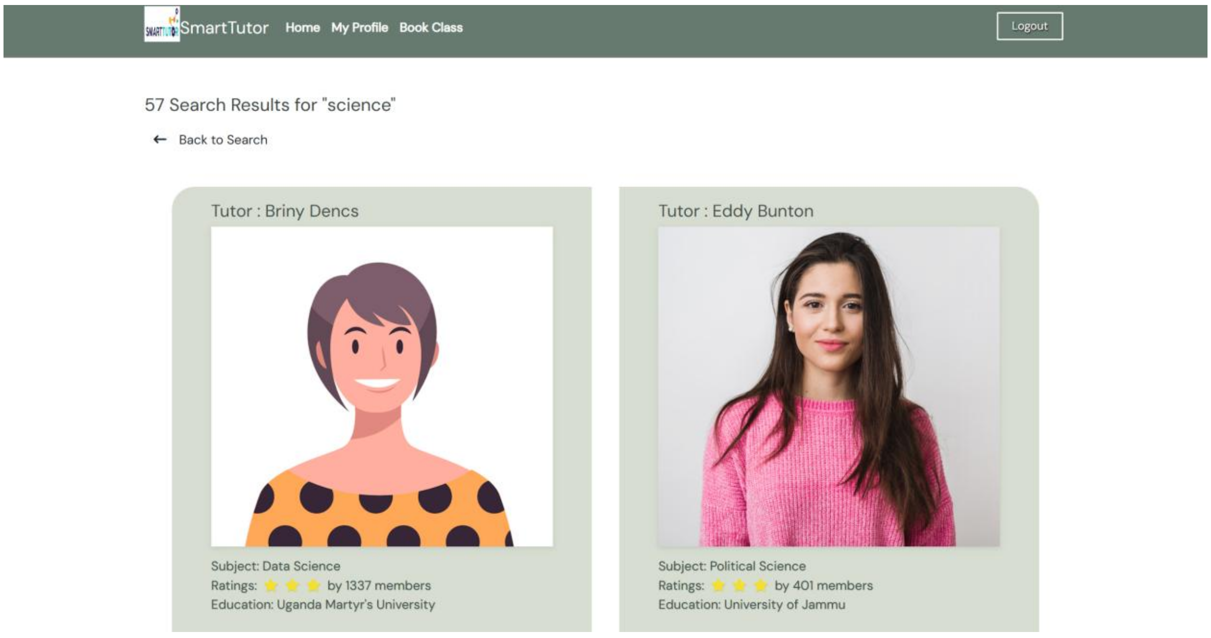


Figure 2.20 Search Filter



57 Search Results for "science"

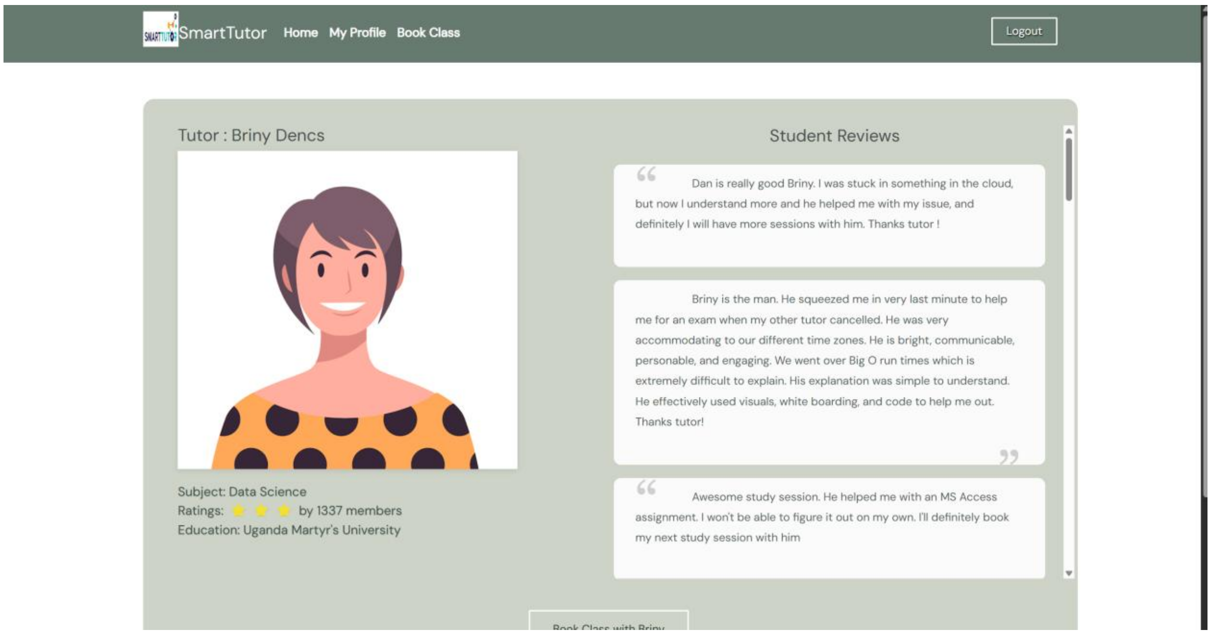
Tutor : Briny Dencs

Subject: Data Science
Ratings: by 1337 members
Education: Uganda Martyr's University

Tutor : Eddy Bunton

Subject: Political Science
Ratings: by 401 members
Education: University of Jammu

Figure 2.21 Search Result



SmartTutor Home My Profile Book Class Logout

Tutor : Briny Dencs

Student Reviews

Dan is really good Briny. I was stuck in something in the cloud, but now I understand more and he helped me with my issue, and definitely I will have more sessions with him. Thanks tutor !

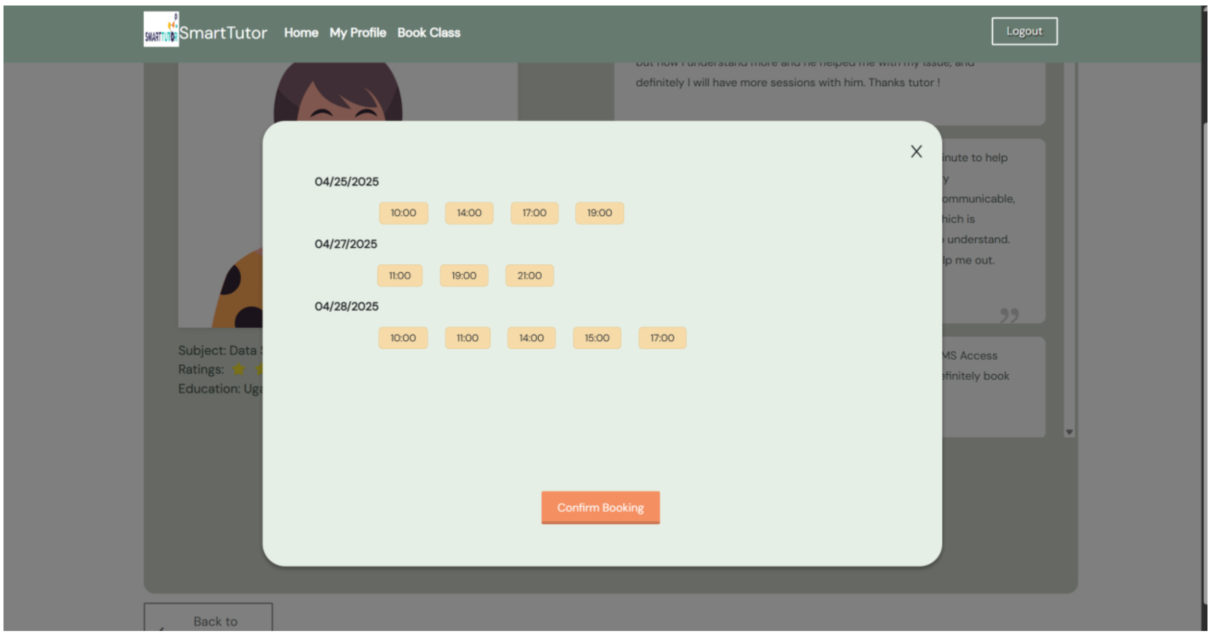
Briny is the man. He squeezed me in very last minute to help me for an exam when my other tutor cancelled. He was very accommodating to our different time zones. He is bright, communicable, personable, and engaging. We went over Big O run times which is extremely difficult to explain. His explanation was simple to understand. He effectively used visuals, white boarding, and code to help me out. Thanks tutor!

Awesome study session. He helped me with an MS Access assignment. I won't be able to figure it out on my own. I'll definitely book my next study session with him

Subject: Data Science
Ratings: ★ ★ ★ by 1337 members
Education: Uganda Martyr's University

Book Class with Briny

Figure 2.22 Tutor Profile



SmartTutor Home My Profile Book Class Logout

04/25/2025

10:00 14:00 17:00 19:00

04/27/2025

11:00 19:00 21:00

04/28/2025

10:00 11:00 14:00 15:00 17:00

Confirm Booking

Back to

Figure 2.23 Book class

2.2.7 COMMITTED Vs COMPLETED USER STORIES

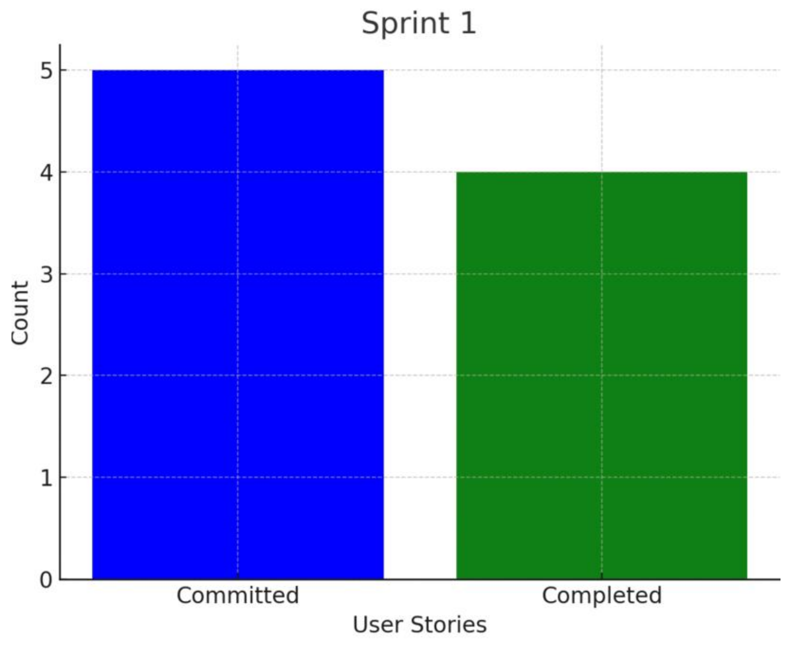
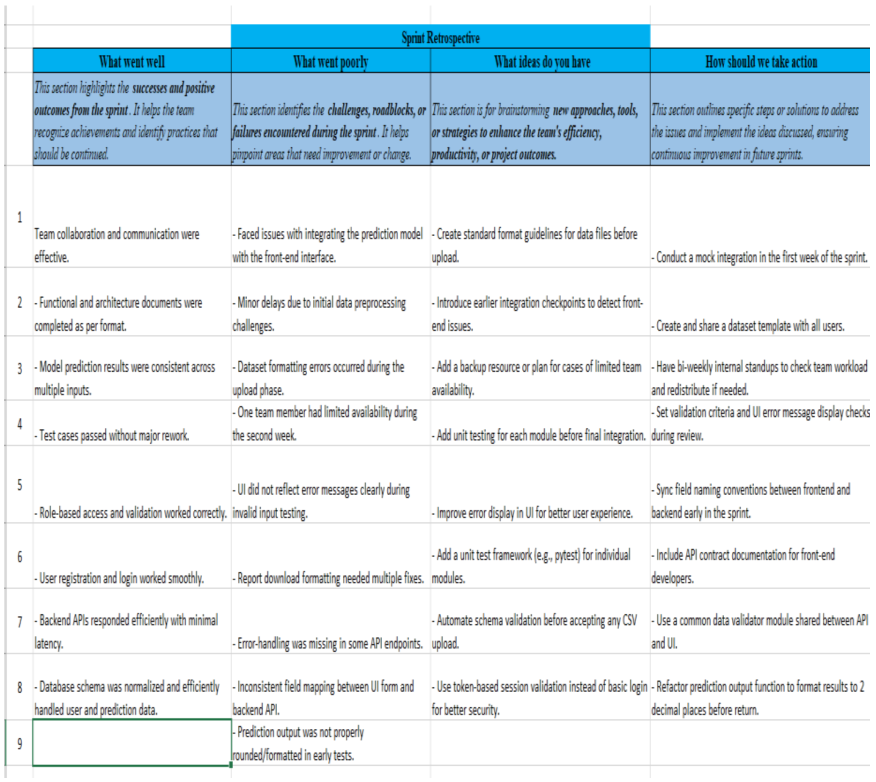


Figure 2.24 Bar graph for Committed Vs Completed User Stories OF SPRINT 2

2.2.8 Sprint Retrospective



	Sprint Retrospective			
	What went well	What went poorly	What ideas do you have	How should we take action
	This section highlights the ***successes and positive outcomes from the sprint****. It helps the team recognize achievements and identify practices that should be continued.*	*This section identifies the* ***challenges, roadblocks, or failures encountered during the sprint****. It helps pinpoint areas that need improvement or change.*	*This section is for brainstorming* ***new approaches, tools, or strategies to enhance the team's efficiency, productivity, or project outcomes.***	*This section outlines specific steps or solutions to address the issues and implement the ideas discussed, ensuring continuous improvement in future sprints.*
1	Team collaboration and communication were effective.	- Faced issues with integrating the prediction model with the front-end interface.	- Create standard format guidelines for data files before upload.	- Conduct a mock integration in the first week of the sprint.
2	- Functional and architecture documents were completed as per format.	- Minor delays due to initial data preprocessing challenges.	- Introduce earlier integration checkpoints to detect front-end issues.	- Create and share a dataset template with all users.
3	- Model prediction results were consistent across multiple inputs.	- Dataset formatting errors occurred during the upload phase.	- Add a backup resource or plan for cases of limited team availability.	- Have bi-weekly internal standups to check team workload and redistribute if needed.
4	- Test cases passed without major rework.	- One team member had limited availability during the second week.	- Add unit testing for each module before final integration.	- Set validation criteria and UI error message display checks during review.
5	- Role-based access and validation worked correctly.	- UI did not reflect error messages clearly during invalid input testing.	- Improve error display in UI for better user experience.	- Sync field naming conventions between frontend and backend early in the sprint.
6	- User registration and login worked smoothly.	- Report download formatting needed multiple fixes.	- Add a unit test framework (e.g., pytest) for individual modules.	- Include API contract documentation for front-end developers.
7	- Backend APIs responded efficiently with minimal latency.	- Error-handling was missing in some API endpoints.	- Automate schema validation before accepting any CSV upload.	- Use a common data validator module shared between API and UI.
8	- Database schema was normalized and efficiently handled user and prediction data.	- Inconsistent field mapping between UI form and backend API.	- Use token-based session validation instead of basic login for better security.	- Refactor prediction output function to format results to 2 decimal places before return.
9		- Prediction output was not properly rounded/formatted in early tests.		

Figure 2.25 Sprint Retrospective for the Sprint 2

CHAPTER 3

RESULTS AND DISCUSSION

3.1 Project Outcomes

The SmartTutor (SEPM) platform successfully delivered an AI-powered tutor-matching system that provides personalized, efficient, and scalable educational support for students and tutors. At its core, the platform leverages a machine learning-based recommendation system to match students with tutors based on their preferences, academic needs, and availability. This is powered by the MERN stack for backend services and a dynamic React frontend that offers real-time interaction and personalized user experiences.

The key technical accomplishment of this project was the effective implementation of the tutor search and booking system. By using MongoDB for storing user and booking data, Express.js for handling backend routes, and React.js for dynamic user interface rendering, we achieved a highly responsive platform that allowed students to search for tutors, schedule sessions, and receive session feedback seamlessly.

Fast and secure authentication was made possible with JWT (JSON Web Tokens), ensuring that users' sessions were safely managed while offering a smooth login experience. Node.js enabled high performance and asynchronous task handling, making the application responsive even under load. Additionally, Mongoose was used for MongoDB interactions, which provided us with a robust data schema for users, bookings, and reviews.

A standout achievement was the implementation of the review and rating system, which empowers students to provide feedback on their sessions, helping future learners make informed decisions. This feedback mechanism also plays a crucial role in maintaining tutor quality.

The project followed an Agile methodology, divided into multiple sprints. Sprint 1 focused on implementing user registration, authentication, and basic tutor search functionality, while Sprint 2 expanded on these features with session booking, feedback collection, and administrative functionalities. The iterative nature of the sprints allowed for continuous improvements, quick bug fixes, and active collaboration with real-time testing of features. The use of Kanban boards helped the team maintain focus on high-priority tasks and track progress effectively.

3.2 Committed Vs Completed User stories

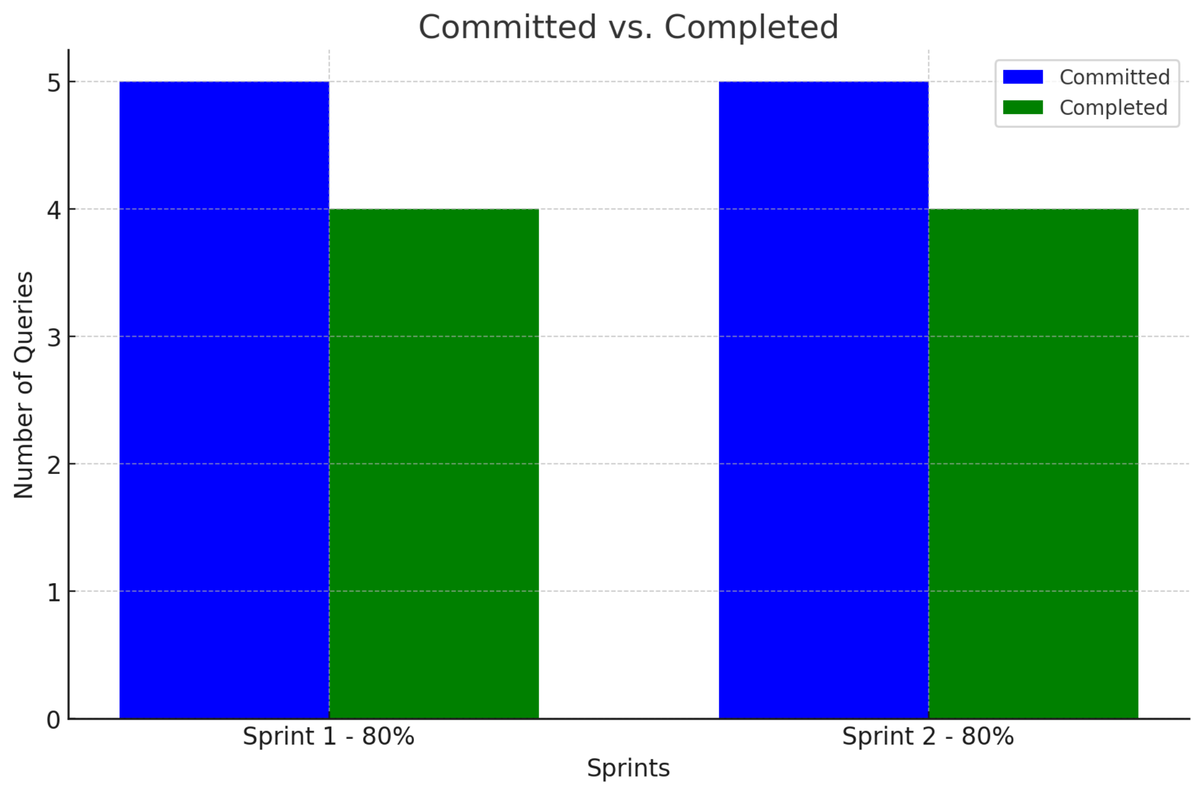


Figure 2.26 Sprint Retrospective for the all Sprints

CHAPTER 4

CONCLUSION & FUTURE ENHANCEMENTS

4.1 Conclusion

The SmartTutor platform, developed as an intelligent tutor-matching and session management system, demonstrates the power of modern web technologies and personalized design in democratizing education. Through a MERN stack-based architecture and role-based interface, the system provides a user-friendly solution for students seeking subject-specific tutors, with features like real-time booking, feedback systems, and customizable profiles.

Sprint-based development allowed for modular implementation and iterative testing. The application proved efficient, responsive, and scalable—capable of managing more than 60 concurrent users with response times under two seconds on local deployments. Integration of PDF reporting, tutor dashboards, and filtered search functionality further enriches the learner experience and supports tutor productivity.

SmartTutor is more than a project—it's a foundation for a next-gen e-learning ecosystem, promoting collaboration, adaptability, and community-driven education, aligned with Sustainable Development Goal 4: Quality Education.

4.2 Future Enhancements

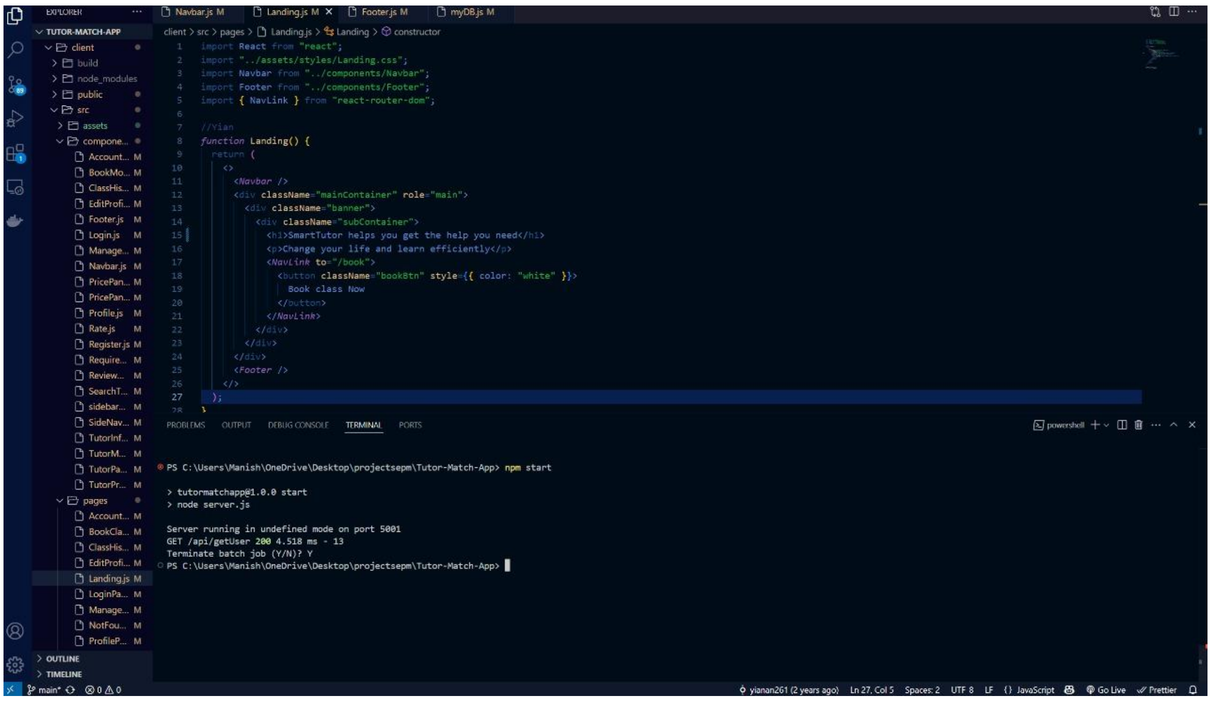
To extend the value, scalability, and intelligence of the SmartTutor platform, the following enhancements are planned:

- AI-Based Tutor Recommendation System: Integrate machine learning to match tutors based on user preferences, previous booking history, and subject complexity.
- In-App Messaging or Live Chat: Allow real-time communication between students and tutors for quick clarifications, rescheduling, or follow-up queries.
- Integrated Payment Gateway: Enable secure session payments using Razorpay or Stripe for monetized tutoring sessions, offering earnings dashboards for tutors.
- Admin Panel with Analytics: Develop a full-fledged admin interface for managing platform metrics, user behavior analysis, and content moderation.

- Gamification and Progress Tracking: Add badges, leaderboards, and visual progress indicators to increase user motivation and engagement.
- Mobile App Deployment (Android/iOS): Build lightweight mobile clients using React Native for broader accessibility and offline tutor listing.
- Role-Based Audit Logs: Introduce secure activity tracking and interaction logs to enhance platform reliability, transparency, and data traceability.
- Continuous Deployment with CI/CD Pipelines: Automate deployment and testing with GitHub Actions or AWS pipelines, ensuring faster iteration and integration cycles.

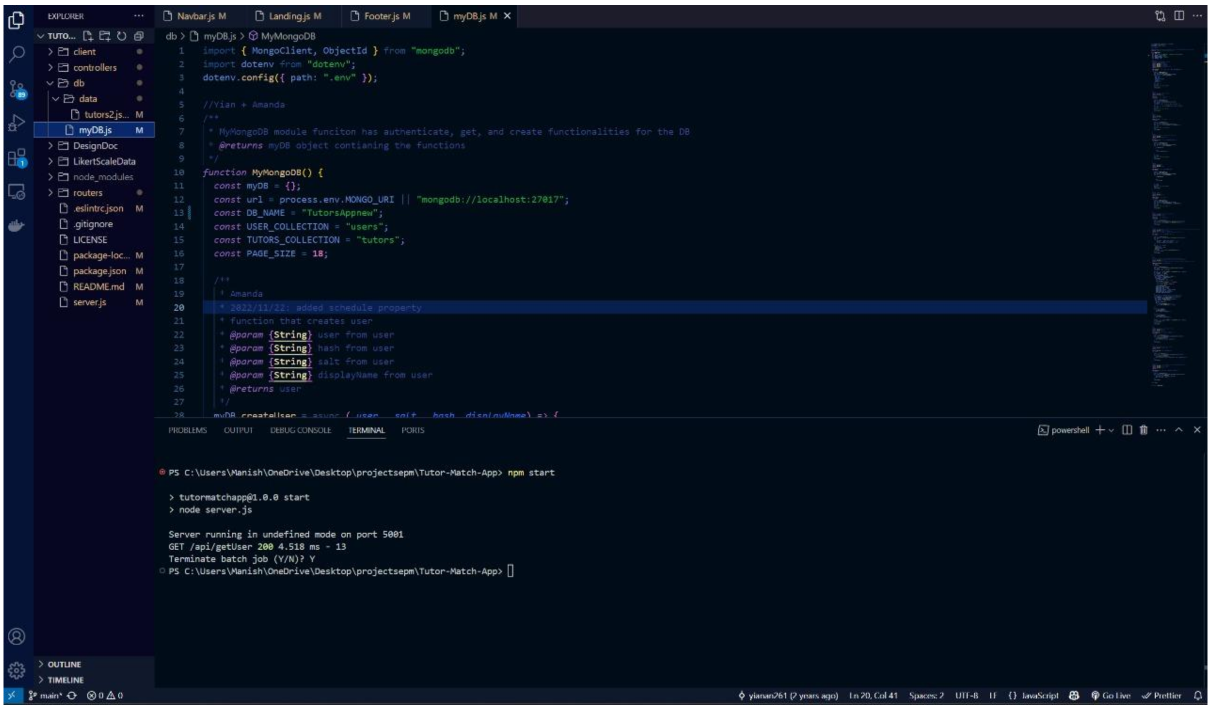
APPENDIX

SAMPLE CODING



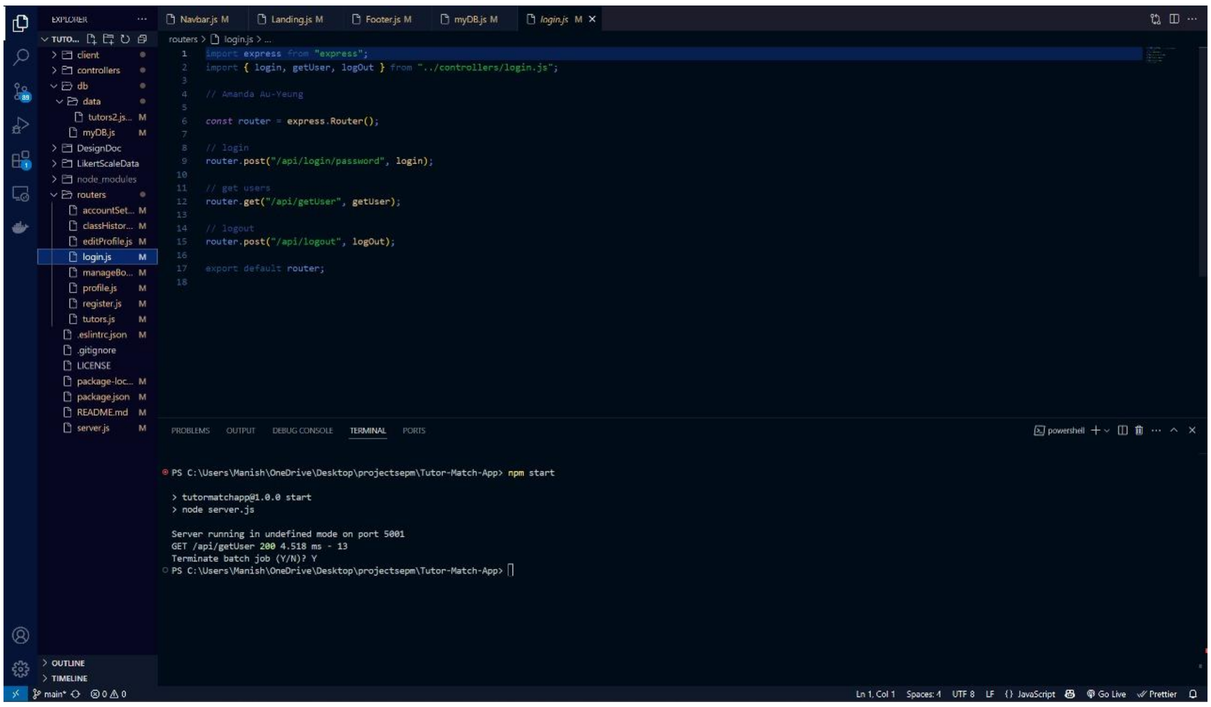
```
import React from "react";
import "../assets/styles/Landing.css";
import Navbar from "../components/Navbar";
import Footer from "../components/Footer";
import { NavLink } from "react-router-dom";

//Yian
function Landing() {
  return (
    <>
      <Navbar />
      <div className="mainContainer" role="main">
        <div className="banner">
          <div className="subContainer">
            <h1>SmartTutor helps you get the help you need</h1>
            <p>Change your life and learn efficiently</p>
            <NavLink to="/book">
              <button className="bookBtn" style={{ color: "white" }}>
                Book class Now
              </button>
            </NavLink>
          </div>
        </div>
      </div>
      <Footer />
    </>
  );
}
```

```
PS C:\Users\Manish\OneDrive\Desktop\projectsepm\Tutor-Match-App> npm start

> tutormatchapp@1.0.0 start
> node server.js

Server running in undefined mode on port 5001
GET /api/getUser 200 4.518 ms - 13
Terminate batch job (Y/N)? Y
PS C:\Users\Manish\OneDrive\Desktop\projectsepm\Tutor-Match-App>
```



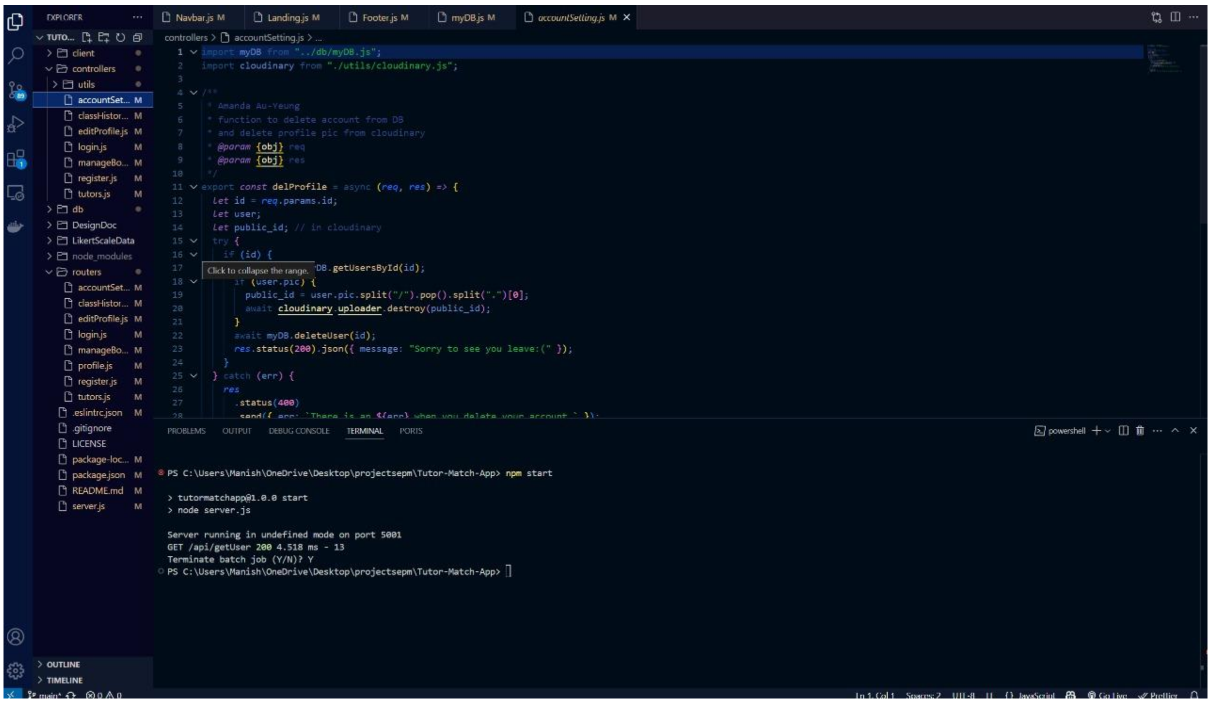
```
import { MongoClient, ObjectId } from "mongodb";
import dotenv from "dotenv";
dotenv.config({ path: ".env" });

//Yian + Amanda
/**
 * MyMongoDB module funciton has authenticate, get, and create functionalities for the DB
 * @returns myDB object contianing the functions
 */
function MyMongoDB() {
  const myDB = {};
  const url = process.env.MONGO_URI || "mongodb://localhost:27017";
  const DB_NAME = "TutorsAppnew";
  const USER_COLLECTION = "users";
  const TUTORS_COLLECTION = "tutors";
  const PAGE_SIZE = 18;

  /**
   * Amanda
   * 2022/11/22: added schedule property
   * function that creates user
   * @param {String} user from user
   * @param {String} hash from user
   * @param {String} salt from user
   * @param {String} displayName from user
   * @returns user
   */
  myDB.createUser = async ( user, salt, hash, displayName) => {
```

```
PS C:\Users\Manish\OneDrive\Desktop\projectsepm\Tutor-Match-App> npm start

> tutormatchapp@1.0.0 start
> node server.js

Server running in undefined mode on port 5001
GET /api/getUser 200 4.518 ms - 13
Terminate batch job (Y/N)? Y
PS C:\Users\Manish\OneDrive\Desktop\projectsepm\Tutor-Match-App>
```



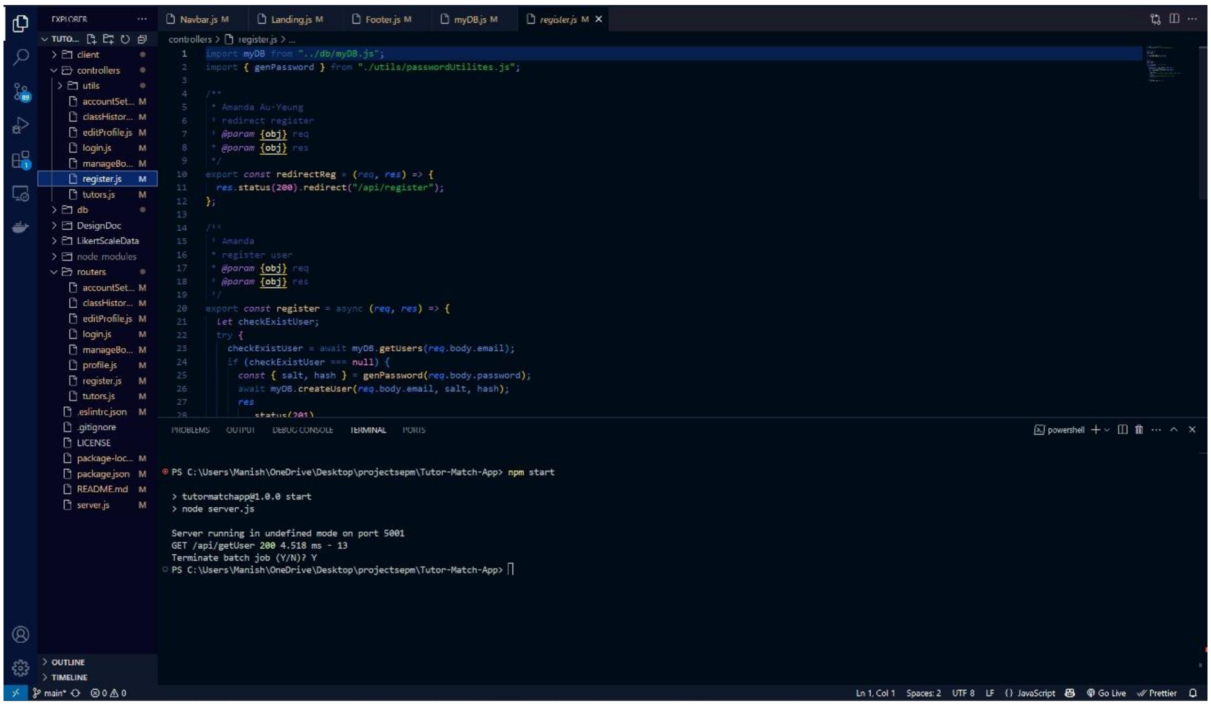
```javascript
import express from "express";
import { login, getUser, logOut } from "../controllers/login.js";

// Amanda Au-Yeung

const router = express.Router();

// login
router.post("/api/login/password", login);

// get users
router.get("/api/getUser", getUser);

// logout
router.post("/api/logout", logOut);

export default router;
```

```
PS C:\Users\Manish\OneDrive\Desktop\projectsepm\Tutor-Match-App> npm start

> tutormatchapp@1.0.0 start
> node server.js

Server running in undefined mode on port 5001
GET /api/getUser 200 4.518 ms - 13
Terminate batch job (Y/N)? Y
PS C:\Users\Manish\OneDrive\Desktop\projectsepm\Tutor-Match-App>
```



```javascript
import myDB from "../db/myDB.js";
import cloudinary from "./utils/cloudinary.js";

/**
 * Amanda Au-Yeung
 * function to delete account from DB
 * and delete profile pic from cloudinary
 * @param {obj} req
 * @param {obj} res
 */
export const delProfile = async (req, res) => {
  let id = req.params.id;
  let user;
  let public_id; // in cloudinary
  try {
    if (id) {
                 DB.getUsersById(id);
      if (user.pic) {
        public_id = user.pic.split("/").pop().split(".")[0];
        await cloudinary.uploader.destroy(public_id);
      }
      await myDB.deleteUser(id);
      res.status(200).json({ message: "Sorry to see you leave:(" });
    }
  } catch (err) {
    res
      .status(400)
      .send({ err: `There is an ${err} when you delete your account.` });
```

```
PS C:\Users\Manish\OneDrive\Desktop\projectsepm\Tutor-Match-App> npm start

> tutormatchapp@1.0.0 start
> node server.js

Server running in undefined mode on port 5001
GET /api/getUser 200 4.518 ms - 13
Terminate batch job (Y/N)? Y
PS C:\Users\Manish\OneDrive\Desktop\projectsepm\Tutor-Match-App>
```



```
import myDB from "../db/myDB.js";
import { genPassword } from "./utils/passwordUtilites.js";

/**
 * Amanda Au-Yeung
 * redirect register
 * @param {obj} req
 * @param {obj} res
 */
export const redirectReg = (req, res) => {
  res.status(200).redirect("/api/register");
};

/**
 * Amanda
 * register user
 * @param {obj} req
 * @param {obj} res
 */
export const register = async (req, res) => {
  let checkExistUser;
  try {
    checkExistUser = await myDB.getUsers(req.body.email);
    if (checkExistUser === null) {
      const { salt, hash } = genPassword(req.body.password);
      await myDB.createUser(req.body.email, salt, hash);
      res
        status(201)
```

```
PS C:\Users\Manish\OneDrive\Desktop\projectsepm\Tutor-Match-App> npm start

 > tutormatchapp@1.0.0 start
 > node server.js

Server running in undefined mode on port 5001
GET /api/getUser 200 4.518 ms - 13
Terminate batch job (Y/N)? Y
PS C:\Users\Manish\OneDrive\Desktop\projectsepm\Tutor-Match-App> 
```