Dhanekula Institute Of Engineering And Technology

Affiliated to JNTUK, Kakinada & Approved By AICTE
Ganguru, Vijayawada - 521139

Summer Internship Report

"Building Apps Using LLM APIs" Internship Program

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

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

by

Maridu Naga Sai Srinivas

(228T1A4264)

CSE(AI & ML)

Dhanekula Institute Of Engineering And Technology

Affiliated to JNTUK, Kakinada & Approved By AICTE
Ganguru, Vijayawada - 521139

Department of Computer Science & Engineering

(Artificial Intelligence & Machine Learning)

BONAFIDE CERTIFICATE

This is to certify that the Summer Internship report for the program titled **"Building LLMs using APIs"** is a bonafide record of the work done by **Maridu Naga Sai Srinivas (228T1A4264)** during the period from **17th May 2025 to 10th July 2025**. The final project developed during this internship is titled **"Smart Tutor: An LLM-Powered Learning Application."** This work is submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering (Artificial Intelligence & Machine Learning) from Dhanekula Institute of Engineering and Technology, affiliated to Jawaharlal Nehru Technological University, Kakinada, during the academic year 2024-2025.

Dr. Ch Suresh Babu

Head of the Department

Department of CSE (AI & ML)

External Examiner

DECLARATION BY THE STUDENT

I, **Maridu Naga Sai Srinivas**, bearing Hall Ticket Number **228T1A4264**, hereby declare that this Summer Internship report, based on the **"Building LLMs using APIs"** program, has been prepared by me.

This report is a record of the original work carried out by me during the internship period and has not been submitted in part or full to any other university or institution for the award of any degree or diploma.

_____

Maridu Naga Sai Srinivas

(228T1A4264)

CSE(AI & ML)

ANDHRA PRADESH STATE COUNCIL OF HIGHER EDUCATION

(A Statutory Body of the Government of A.P.)

Edubot

Revolutionary Learning

ISO 9001:2015

Certificate of Completion

This Certificate is presented to

Maridu Naga Sai Srinivas

Who has successfully completed the **Building LLMs using APIs** Internship program held during period **17 May 2025 to 10 July 2025** through Rigorous Assignments, Projects, and Assessments.

Mr. Maridu Naga Sai Srinivas (228T1A4264) from **DIET, Andhra Pradesh** has proven their Proficiency, Technical skills, Problem-Solving Abilities, and Commitment to Excellence.

EDUBOT STATION PVT. LTD

1st Floor MK Aura, Road No 3, Mahanadu Road,
Vijayawada, AP 520008

www.edubottechnologies.com

Sasanka Paladugu
Founder & C.E.O

ABSTRACT

This report details the work and learning from an eight-week internship program, **"Building LLMs using APIs,"** which culminated in the development of a final project: the "Smart Tutor" application. This intensive program focused on building a full-stack web application that leverages Large Language Models (LLMs) to offer features like AI-powered explanations, practice problem generation, automated quizzes, and study note creation.

The "Smart Tutor" application is built using a modern tech stack. The backend is developed with Python and the Flask framework, utilizing a SQLite database managed via SQLAlchemy for data persistence. User authentication is securely handled using Flask-JWT-Extended. The frontend is a responsive single-page application built with React (TypeScript) and styled with Tailwind CSS, ensuring a seamless user experience across devices.

The core of the application lies in its integration with LLMs through the OpenRouter API, orchestrated by the LangChain framework. This allows the platform to process user queries and generate dynamic, high-quality educational content in real-time. Key features include an interactive Q&A module, a practice problem generator, a quiz creator, and a tool for generating and saving study notes.

This report details the project's entire lifecycle, from the initial learning of foundational technologies like Python, Flask, and React to the design, implementation, and deployment of the LLM-integrated features. The work done demonstrates a comprehensive understanding of full-stack development, API integration, and the practical application of AI in creating impactful educational tools. A detailed weekly log outlines the module-by-module progress and learnings acquired during the internship period.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Introduction to the Internship

This report documents the work completed during the eight-week summer internship program titled **"Building LLMS Powered Apps USING APIs,"** which took place from **May 17, 2025, to July 10, 2025**. This program, offered by Edubot Station Pvt. Ltd., was designed to provide practical, hands-on experience in full-stack web development and the integration of cutting-edge Artificial Intelligence technologies, specifically Large Language Models (LLMs).

The modern software landscape is increasingly dominated by intelligent applications that leverage AI to provide personalized and dynamic user experiences. This internship provided a structured learning path to build such an application from the ground up. The primary outcome of this intensive training is the development of a final project, "Smart Tutor," an AI-powered educational platform designed to act as a personal learning assistant for students.

1.2 Final Project Profile: Smart Tutor

Smart Tutor is the final project developed during this internship. It is an innovative AI-powered learning platform designed to revolutionize the way students learn by providing personalized, on-demand tutoring experiences. The platform addresses key challenges in modern education—such as the lack of personalized attention, the high cost of private tutoring, and the need for 24/7 learning support—by offering interactive learning modules and content tailored to each student's unique needs.

- **Mission:** To make high-quality education accessible to everyone by leveraging the power of AI to create an intuitive and effective learning environment that adapts to each student's pace and style.
- **Vision:** To empower learners by providing them with the tools and resources they need to understand complex concepts, test their knowledge, and achieve their academic goals, thereby fostering a culture of continuous learning and curiosity.
- **Services:** Smart Tutor provides a suite of AI-driven tools, including:
    - AI-generated, step-by-step explanations for complex questions.
    - Generation of practice problems with solutions.
    - Automated creation of subject-specific quizzes.
    - On-demand generation of comprehensive study notes.

The platform is designed to be a versatile tool for students across various academic levels and subjects, making it a valuable companion for both coursework and exam preparation.

1.3 Objectives of the Internship

The primary objectives set for this internship were comprehensive, aiming to bridge the gap

between academic knowledge and industry practice. The key goals were:

- To gain hands-on experience in full-stack web development using modern technologies, specifically the stack of Python, Flask, and React.
- To understand the fundamental concepts of Large Language Models (LLMs), including their architectures (Transformers, Attention Mechanisms) and capabilities in Natural Language Processing (NLP).
- To learn the practical aspects of integrating third-party APIs, particularly LLM APIs, into a web application to create intelligent and interactive features.
- To develop a fully functional, secure, and responsive web application from scratch, covering all stages from design and development to testing.
- To apply essential software engineering principles, including version control with Git, modular design, and secure coding practices.

1.4 Scope of the Work

The scope of this project involved the end-to-end development of the "Smart Tutor" platform. The key areas of work included:

- **Backend Development:** Creating a robust backend using the Flask micro-framework. This involved designing and developing a secure RESTful API to handle all application logic, including user authentication, database interactions, and communication with the AI service. A database schema was implemented using SQLAlchemy ORM to manage data persistence.
- **Frontend Development:** Building a dynamic and responsive user interface using React and TypeScript. This single-page application (SPA) was designed to provide a seamless and intuitive user experience, allowing users to interact with the AI features effortlessly on both desktop and mobile devices.
- **AI Integration:** Integrating the OpenRouter API using the LangChain framework. This was the core of the project, powering the intelligent features of the platform:
    - AI-Powered Explanations
    - Practice Problem Generation
    - Quiz Generation
    - Study Note Creation
- **User Features:** Implementing essential user-centric functionalities, including a secure user registration and login system, and a personal dashboard for saving and reviewing generated content like questions and notes.

1.5 Course Outcomes (COs) for the Internship

Upon successful completion of the internship, the student will be able to:

- **CO1:** Apply theoretical concepts of web development, database management, and AI to build a complete, functional full-stack application.

- **CO2:** Utilize modern software tools and frameworks (Flask, React, LangChain, Git) proficiently within a structured project development lifecycle.
- **CO3:** Function effectively in a project-based environment, demonstrating professionalism, ethical conduct, and efficient time management.
- **CO4:** Analyze a complex problem, design a viable technical solution that involves external API integration, and implement it from conception to completion.
- **CO5:** Communicate project architecture, implementation details, and outcomes effectively through comprehensive technical documentation and reporting.

1.6 Program Outcomes (POs) & Program Specific Outcomes (PSOs)

The skills and knowledge gained during this internship directly contribute to the attainment of the Program Outcomes (POs) and Program Specific Outcomes (PSOs) for the B.Tech in CSE (AI & ML).

Program Outcomes (POs):

- **PO1 (Engineering Knowledge):** Apply knowledge of mathematics, science, and engineering fundamentals.
- **PO2 (Problem Analysis):** Identify, formulate, and analyze complex engineering problems.
- **PO3 (Design/Development of Solutions):** Design solutions for complex engineering problems.
- **PO4 (Conduct Investigations):** Use research-based knowledge to conduct investigations of complex problems.
- **PO5 (Modern Tool Usage):** Create, select, and apply appropriate techniques, resources, and modern IT tools.
- **PO6 (The Engineer and Society):** Assess societal, health, safety, legal, and cultural issues.
- **PO7 (Environment and Sustainability):** Understand the impact of professional engineering solutions.
- **PO8 (Ethics):** Apply ethical principles and commit to professional ethics.
- **PO9 (Individual and Teamwork):** Function effectively as an individual and as a member or leader in diverse teams.
- **PO10 (Communication):** Communicate effectively on complex engineering activities.
- **PO11 (Project Management and Finance):** Demonstrate knowledge of engineering and management principles.
- **PO12 (Life-long Learning):** Engage in independent and life-long learning.

Program Specific Outcomes (PSOs) for CSE (AI & ML):

- **PSO1:** Apply principles of artificial intelligence, machine learning, and data science to design and develop intelligent systems.
- **PSO2:** Utilize modern tools and frameworks (like TensorFlow, Python, Scikit-learn,

LangChain) to build, train, and deploy robust AI and ML models.

- **PSO3:** Develop solutions for complex problems in specialized areas such as natural language processing and deep learning.

1.7 CO-PO/PSO Mapping

This table illustrates the correlation between the Internship Course Outcomes (COs) and the Program Outcomes (POs) and Program Specific Outcomes (PSOs).

Correlation Level: **1** - Low | **2** - Medium | **3** - High

COs	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**
CO1	3	2	2	-	-	-	1	-	-	-	-	2	3	2	2
CO2	2	2	2	1	3	-	-	-	2	-	-	3	2	3	2
CO3	-	-	-	-	-	1	-	3	3	2	2	1	-	-	-
CO4	2	3	3	2	2	1	1	-	2	-	2	2	3	2	3
CO5	-	-	-	-	-	-	-	2	2	3	1	1	-	-	-

CHAPTER 2: SYSTEM ANALYSIS AND DESIGN

This chapter outlines the key requirements, technologies, architecture, and database design of the **Smart Tutor** platform.

2.1 Requirements Analysis

Functional Requirements

- **User Management:** Register/login/logout with unique credentials; JWT-secured sessions.
- **AI Tutoring:** Users can input questions, get AI-generated step-by-step explanations, create practice problems, MCQs, and structured notes.
- **Content Management:** Save, view, and delete generated content via personal dashboard.

Non-Functional Requirements

- **Performance:** API responses <3s; AI responses depend on LLM with loading indicators.
- **Usability & Responsiveness:** Intuitive, feedback-driven, mobile-friendly UI.
- **Security:** Hashed passwords, JWT auth, environment-based secrets.
- **Scalability & Maintainability:** SQLAlchemy ORM for easy DB migration; modular, documented code.

2.2 Technology Stack

- **Backend:** *Flask (Python)* – lightweight, flexible REST API framework.
- **Frontend:** *React* + *TypeScript* – component-based, type-safe UI.
- **Database:** *SQLite (SQLAlchemy ORM)* – simple dev setup, migratable to PostgreSQL/MySQL.
- **AI Integration:** *LangChain* + *OpenRouter* – structured LLM interactions and model flexibility.

- **Styling:** *Tailwind CSS* – utility-first for fast, responsive UI design.

2.3 System Architecture

Three-tier client-server model:

1. **Frontend (React):** SPA handling UI, state, and API calls.
2. **Backend (Flask):** Authentication, CRUD logic, LLM communication via LangChain/OpenRouter.
3. **Data Layer:** SQLite DB for users/content; secure backend-only LLM API calls.
   (User → React → Flask API → Database/LLM → React → User)
   This decoupled design ensures independent development and scalability.

2.4 Database Design

Entities:

- **User:** id, username, email, password_hash, created_at
- **Question:** id, user_id, subject, question_text, answer_steps_list (JSON), answer_summary
- **Note:** id, user_id, subject, topic, heading, bullet_points (JSON)

Relationships: One-to-many (User → Questions, Notes).
JSON fields allow flexible storage for AI-generated, variable-length data.

CHAPTER 3: TECHNOLOGIES AND TOOLS STUDIED

The **Smart Tutor** platform was built using a modern full-stack ecosystem combining backend, frontend, and AI technologies to deliver an interactive and intelligent learning experience.

3.1 Backend Technologies

The backend manages logic, authentication, and data communication.
Python: Core backend language chosen for simplicity and rich AI/web libraries.
Flask: Lightweight Python framework for building RESTful APIs with flexibility and fast development.
SQLite: Serverless database ideal for development and testing.
SQLAlchemy: ORM providing secure, object-based database interaction and easy migration.
Flask-JWT-Extended: Handles authentication using JSON Web Tokens.
Flask-Migrate: Manages database schema updates through Alembic.
Flask-CORS: Enables safe cross-origin requests between frontend and backend.

3.2 Frontend Technologies

The frontend provides an interactive, responsive user interface.
React: Component-based JavaScript library for dynamic single-page applications.
TypeScript: Adds static typing for reliability and maintainability.
Tailwind CSS: Utility-first CSS framework for rapid, responsive design.
shadcn/ui: Prebuilt, accessible UI components for polished design.
Vite: Modern build tool ensuring fast development and optimized builds.
React Router DOM: Enables smooth, client-side routing without page reloads.

3.3 AI Integration

The AI layer powers tutoring and content generation.
LLMs (e.g., GPT, LLaMA): Generate explanations, questions, and notes.
OpenRouter API: Unified gateway to multiple LLMs, enabling model flexibility.
LangChain: Framework for building LLM workflows—prompt creation, chaining tasks, and formatting outputs into usable data.

3.4 Development & Deployment Tools

VS Code: Primary IDE for both frontend and backend.
Git & GitHub: Version control and collaboration.
Postman: API testing and debugging tool.
venv: Isolates Python dependencies.
npm: Manages frontend libraries and scripts.

CHAPTER 4: WEEKLY INTERNSHIP LOG

This chapter provides a detailed, week-by-week log of the activities and learnings undertaken during the 8-week internship period, from May 17, 2025, to July 10, 2025.

4.1 Week 1: Python Programming Basics (May 17 - May 23, 2025)

Day	Date	Task / Module	Learning / Activities Performed
1	17-May-2025	Introduction to Python	Studied Python syntax, data types, and operators; wrote simple calculation scripts.
2	18-May-2025	Control Structures	Practiced if-else, for/while loops through small programs (factorial, even-odd).
3	19-May-2025	Functions & Scope	Defined functions with arguments; understood local vs global scope; wrote modular code.
4	20-May-2025	File Handling	Learned file operations (`open`, `with`), modes ('r', 'w', 'a'); practiced reading/writing files.
5	21-May-2025	Error Handling	Implemented `try-except`; handled `ValueError`, `FileNotFoundError`; explored Python logging.
6	22-May-2025	Dev Environment Setup	Installed VS Code & Python, created virtual env, built a “Hello World” Flask app.
7	23-May-2025	Assignment 1	Completed Module-1 project: data validation & logging utility in Python.

4.2 Week 2: Web Development Essentials (May 24 - May 30, 2025)

Day	Date	Task / Module	Learning / Activities Performed
1	24-May-2025	HTML Basics	Studied semantic HTML5 tags (`<header>`, `<nav>`, `<main>`, `<footer>`); built basic webpage structure.
2	25-May-2025	CSS Basics	Learned selectors, properties (color, font, margin, padding), and box model; styled webpage layout.
3	26-May-2025	JavaScript Intro	Explored JS syntax, variables (`let`, `const`), and data types; ran basic scripts in browser console.
4	27-May-2025	DOM Manipulation	Modified HTML elements using `getElementById`, `querySelector`; added interactivity.
5	28-May-2025	Event Handling	Used `addEventListener` for clicks/forms; implemented a dark mode toggle feature.
6	29-May-2025	Responsive Design	Applied Tailwind CSS utility classes and grid system (`sm:`, `md:`, `lg:`) for responsive layouts.
7	30-May-2025	Assignment 2	Completed and submitted responsive landing page for **VMIS** using HTML, CSS, and JS.

4.3 Week 3: Flask Basics (May 31 - June 6, 2025)

Day	Date	Task / Module	Learning / Activities Performed
1	31-May-2025	Flask Setup & Routing	Set up Flask project structure; created routes using `@app.route` for multiple pages.
2	01-Jun-2025	Jinja2 Templating	Rendered dynamic HTML using Jinja2; passed variables from backend to templates.
3	02-Jun-2025	Template Inheritance	Built `base.html` and extended templates for consistent layout.
4	03-Jun-2025	Web Forms	Handled GET & POST requests; created HTML form to capture user input.
5	04-Jun-2025	Form Data & Validation	Retrieved form data via `request.form`; implemented basic server-side validation.
6	05-Jun-2025	Flash Messages	Used `flash()` to display one-time success/error messages post form submission.
7	06-Jun-2025	Assignment 3	Submitted Flask app for VMIS featuring dynamic pages and validated forms.

4.4 Week 4: Database Management (June 7 - June 13, 2025)

Day	Date	Task / Module	Learning / Activities Performed
1	07-Jun-2025	Relational Databases	Studied SQL concepts — tables, primary & foreign keys.
2	08-Jun-2025	Database Schema Design	Designed schema for Smart Tutor (User, Question, Note); created ER diagram.
3	09-Jun-2025	SQLAlchemy ORM	Integrated Flask-SQLAlchemy; defined models for database tables.
4	10-Jun-2025	Database Migration	Used Flask-Migrate for schema creation & version-controlled migrations.
5	11-Jun-2025	CRUD Operations	Implemented Create, Read, Update, Delete using SQLAlchemy queries.
6	12-Jun-2025	RESTful API Integration	Built Flask API endpoints to expose CRUD functionality.
7	13-Jun-2025	Assignment 4	Submitted database-integrated Flask app supporting full CRUD operations.

4.5 Week 5: Basics of Large Language Models (LLMs) (June 14 - June 20, 2025)

Day	Date	Task / Module	Learning / Activities Performed
1	14-Jun-2025	Introduction to LLMs	Studied evolution and capabilities of GPT, BERT, and modern LLMs.
2	15-Jun-2025	Transformer Architecture	Explored the Transformer model powering LLMs.
3	16-Jun-2025	Attention Mechanism	Learned self-attention and contextual understanding in text.
4	17-Jun-2025	Embeddings & NLP Concepts	Understood text vectorization and embedding representation.
5	18-Jun-2025	Hands-on with Hugging Face	Used `transformers` library to load GPT-2 and perform text generation.
6	19-Jun-2025	Prompt Engineering	Designed effective prompts for summarization and Q&A tasks.
7	20-Jun-2025	Assignment 5	Submitted report and scripts demonstrating LLM understanding.

4.6 Week 6: Working with LLM APIs (June 21 - June 27, 2025)

Day	Date	Task / Module	Learning / Activities Performed
1	21-Jun-2025	API Concepts & Postman	Studied REST API basics; tested public APIs using Postman.
2	22-Jun-2025	LLM API Setup	Registered on OpenRouter; stored API key securely via `.env`.
3	23-Jun-2025	API Interaction (Python)	Used `requests` library to send POST requests and parse JSON responses.
4	24-Jun-2025	Flask–LLM Integration	Built Flask route to accept input, call LLM API, and return responses.
5	25-Jun-2025	API Error Handling	Implemented handling for network, auth, and timeout errors.
6	26-Jun-2025	Rate Limit Management	Learned API rate limits; added exponential backoff for 429 errors.
7	27-Jun-2025	Assignment 6	Submitted Flask app with secure and resilient LLM API integration.

4.7 Week 7: Building LLM-Integrated Flask Applications (June 28 - July 4, 2025)

Day	Date	Task / Module	Learning / Activities Performed
1	28-Jun-2025	Designing Core AI Features	Planned API endpoints and data flow for Q&A, practice problems, quizzes, and notes.
2	29-Jun-2025	LangChain Intro	Learned to structure prompts, chain API calls, and parse LLM outputs.
3	30-Jun-2025	AI Backend Logic	Integrated LangChain into Flask to implement AI-powered features.
4	01-Jul-2025	React Frontend Setup	Initialized React project with Vite & TypeScript; created component structure and routing.
5	02-Jul-2025	Frontend–Backend Connection	Wrote React API service functions; handled asynchronous fetch calls to Flask backend.
6	03-Jul-2025	Dynamic UI Rendering	Built UI components for real-time display of AI responses without page reloads.
7	04-Jul-2025	Assignment 7	Submitted functional Smart Tutor prototype integrating Flask backend with React frontend.

4.8 Week 8: Final Project Integration and Documentation (July 5 - July 10, 2025)

Day	Date	Task / Module	Learning / Activities Performed
1	05-Jul-2025	Full-Stack Feature Finalization	Completed core features, including authentication and saving/deleting content.
2	06-Jul-2025	End-to-End Testing	Tested full application flow; fixed bugs in registration, AI features, and content handling.
3	07-Jul-2025	UI/UX Refinement	Enhanced interface using Tailwind CSS and shadcn/ui for intuitive and appealing design.
4	08-Jul-2025	Code Cleanup & Documentation	Refactored code, added comments, and updated README.md with instructions.
5	09-Jul-2025	Internship Report Writing	Compiled weekly logs, documentation, and project details into the final report.
6	10-Jul-2025	Final Project Demonstration	Delivered Smart Tutor demo showcasing all features and application workflow.

CHAPTER 5: WORK DONE / METHODOLOGY

The development of **Smart Tutor** followed a structured, module-based methodology, ensuring foundational skills were mastered before progressing to complex tasks and full integration.

5.1 Development Process Overview

The methodology was iterative and learning-driven, divided into four phases:

- **Phase 1: Fundamentals (Weeks 1–2)**
  Focused on Python and frontend web development (HTML, CSS, JavaScript). Practical assignments reinforced learning for later phases.

- **Phase 2: Backend Development (Weeks 3–4)**
  Built the server-side with Flask, including API development and database integration, forming the backbone of the application.

- **Phase 3: AI & API Integration (Weeks 5–6)**
  Introduced LLM concepts and practical integration via APIs (OpenRouter), bridging standard web apps to AI-powered features.

- **Phase 4: Full-Stack Integration & Finalization (Weeks 7–8)**
  Connected frontend to backend, integrated AI logic, polished UI/UX, tested, and documented the complete system.

5.2 Feature Implementation Example: AI Q&A Workflow

Step 1: Frontend – User Input (React)

- Form captures question and subject using `useState`.

- Async fetch POST request sends data to `/api/ask/` with JWT.

- React dynamically updates UI with AI response.

Step 2: Backend – API Endpoint & Validation (Flask)

- `/api/ask/` route handles POST requests.

- `@jwt_required()` ensures authentication.

- Validates JSON input, then calls AI utility (`get_ai_explanation`).

Step 3: AI Processing – Prompting & Parsing (LangChain)

- PromptTemplate structures question and subject for LLM.
- LangChain sends prompt to OpenRouter LLM (e.g., GPT or Gemini).
- Response parsed into structured JSON with `summary` and `steps`.

Step 4: Frontend – Display Result (React)

- Updates component state with structured answer.
- Renders summary and step-by-step explanation dynamically.

This workflow demonstrates seamless integration of **React frontend, Flask backend, and LLM AI processing**, applied similarly across all Smart Tutor features to deliver an intelligent and interactive learning experience.

CHAPTER 6: RESULTS AND DISCUSSION

6.1 Final Project Outcome

The Smart Tutor application, developed over the eight-week internship, is a fully functional AI-powered learning platform. It integrates a Python/Flask backend with a React/TypeScript frontend, communicating via a RESTful API. Core AI features are powered by LLMs through the OpenRouter API and orchestrated using LangChain.

The platform meets the internship objectives by providing a secure, responsive, and intelligent learning tool, offering interactive explanations, practice problems, quizzes, and study notes.

6.1.1 User Authentication

A secure registration and login system ensures private access to user data.

- **Registration Page:** Fields for "Username," "Email," and "Password" with client-side validation. A "Register" button submits the form.
- **Login Page:** Fields for "Email" and "Password," with dynamic error messages for invalid credentials. Successful login redirects to the dashboard.
- **Security:** Authentication tokens are managed with Flask-JWT-Extended to protect API endpoints.

6.1.2 AI-Powered Explanations (Q&A)

This core feature allows users to ask questions in any academic subject.

- **Interface:** Multi-line text area for the question, subject dropdown, and "Get Explanation" button. Loading spinner indicates processing.
- **Output:** Structured explanation with a summary and numbered or bulleted steps. Users can save questions via a "Save Question" button.

6.1.3 Practice Problem Generation

Users can generate practice problems to reinforce learning.

- **Interface:** Input fields for "Subject" and "Topic" with a "Generate Problems" button.
- **Output:** Numbered practice problems with solutions, displayed clearly for easy use.

6.1.4 Quiz Generation

The quiz feature tests user knowledge interactively.

- **Interface:** Similar to practice problems, requiring subject and topic input.
- **Output:** Multiple-choice questions displayed with clickable options for self-assessment.

6.1.5 Study Notes Generation

Users can generate concise study notes for quick revision.

- **Interface:** Inputs for subject and topic, with a "Generate Notes" button.
- **Output:** Well-structured notes with main heading and nested bullet points. A "Save Note" button allows storing content for future reference.

6.1.6 Saving and Viewing Content

All generated questions and notes can be saved to the user account for tracking learning progress.

- **Dashboard:** Displays "Saved Questions" and "Saved Notes" in card or list format.
- **Functionality:** Users can expand items to view full content and delete entries as needed.

CHAPTER 7: CONCLUSION

7.1 Overall Summary

The eight-week internship provided hands-on experience in building a modern, AI-powered web application. The **Smart Tutor** project successfully integrates a Flask backend, React frontend, and LLM API to create a secure, responsive, and feature-rich educational tool. All internship objectives were met, resulting in a well-architected prototype demonstrating practical AI applications in education.

The systematic, module-based methodology ensured mastery of each technology before advancing, culminating in a cohesive application. This project enhanced my technical skills and provided insights into the full software development lifecycle—from requirement analysis and design to implementation, testing, and documentation.

7.2 Learning Outcomes

Technical Skills:

- **Backend Development:** Proficiency in Python and Flask for RESTful APIs; database management using SQLAlchemy, schema design, migrations, and CRUD operations.
- **Frontend Development:** Built dynamic, responsive SPAs using React and TypeScript; practical experience with Vite and Tailwind CSS.
- **AI Integration:** Integrated LLM APIs via LangChain, including prompt engineering and structured output parsing.
- **Full-Stack Development:** Holistic understanding of web application lifecycle, covering backend, database, and frontend.
- **Security:** Implemented JWT authentication, password hashing, and secure API key management.

Soft Skills:

- **Problem-Solving:** Tackled challenges like API rate limits, asynchronous frontend operations, and flexible database design.
- **Project Management:** Learned to divide a large project into manageable phases and maintain timelines effectively.

- **Adaptability & Lifelong Learning:** Quickly adapted to new technologies, frameworks, and APIs, emphasizing continuous learning in AI and software development.

7.3 Challenges Faced and Solutions

- **API Rate Limiting:** Managed frequent OpenRouter API calls without exceeding limits by implementing server-side caching for repeated prompts.

- **Asynchronous UI Updates:** Ensured smooth frontend responses using React Query to manage loading states, errors, and data caching.

- **Storing Unstructured Data:** LLM responses were semi-structured; using JSON fields in the database allowed storage of nested, variable-length data.

- **Security Implementation:** Configured secure JWT authentication and stored sensitive credentials in environment variables, ensuring safe handling of API keys and tokens.

APPENDIX: APPLICATION SCREENSHOTS

This appendix contains the screenshots of the "Smart Tutor" application, illustrating the key features and user interface discussed in Chapter 6.

A.1 User Authentication

Figure A.1.1: User Registration Page *Description: A clean, centered form with fields for "Full Name," "Email," and "Password."*

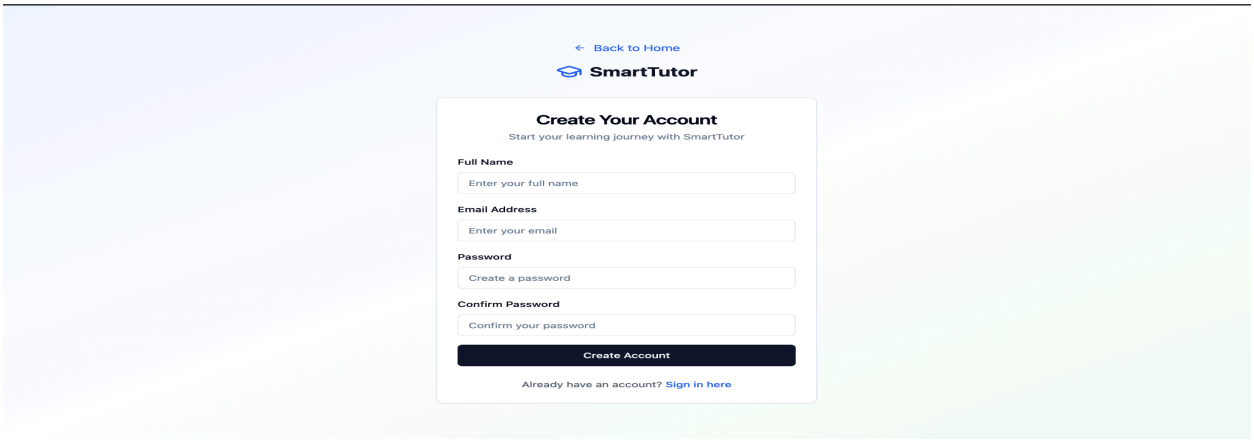
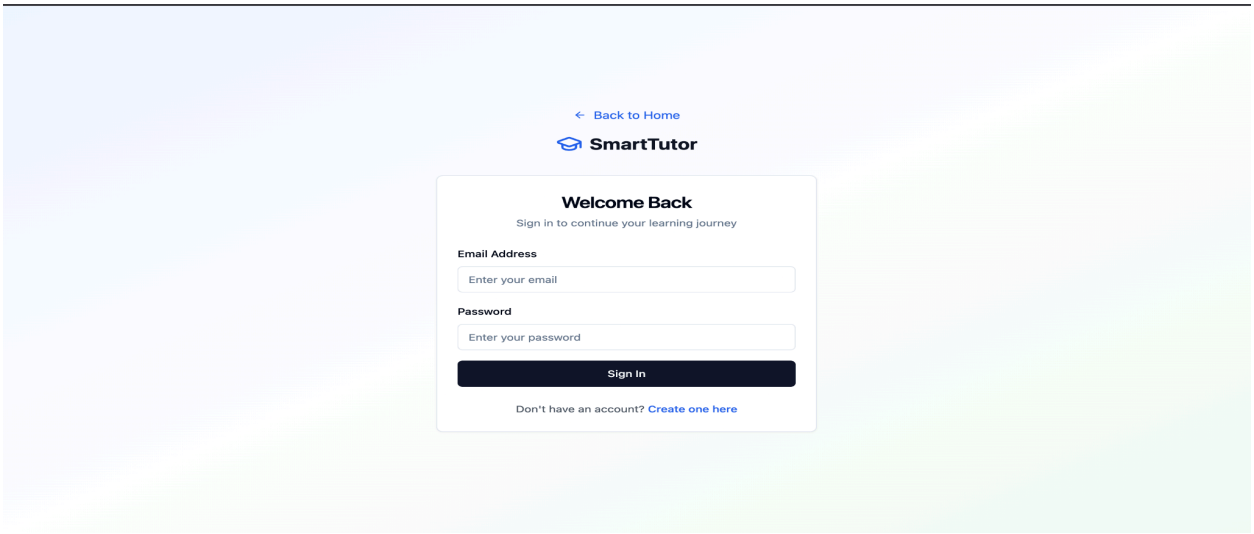


Figure A.1.2: User Login Page *Description: A form with fields for "Email" and "Password," and a "Login" button.*



A.2 AI-Powered Features

Figure A.2.1: AI Q&A Interface *Description: The "Ask a Question" page with a text area for the question and a dropdown for the subject.*

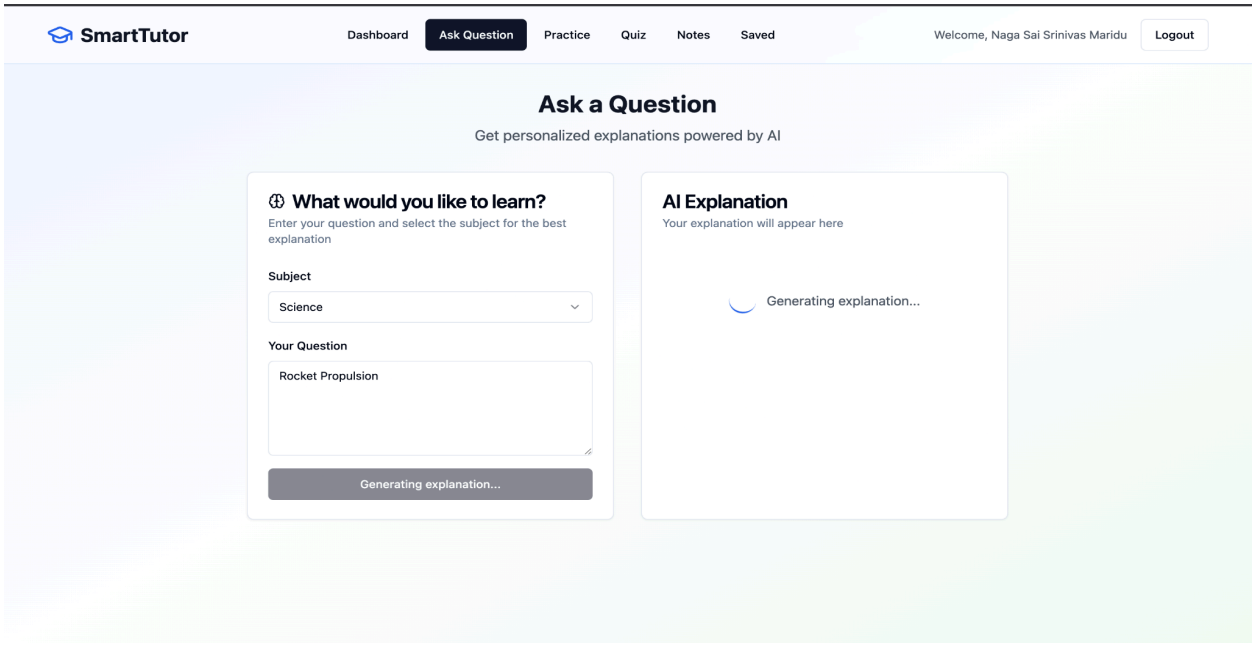


Figure A.2.2: Generated Explanation *Description: The dynamically displayed, step-by-step explanation generated by the AI.*

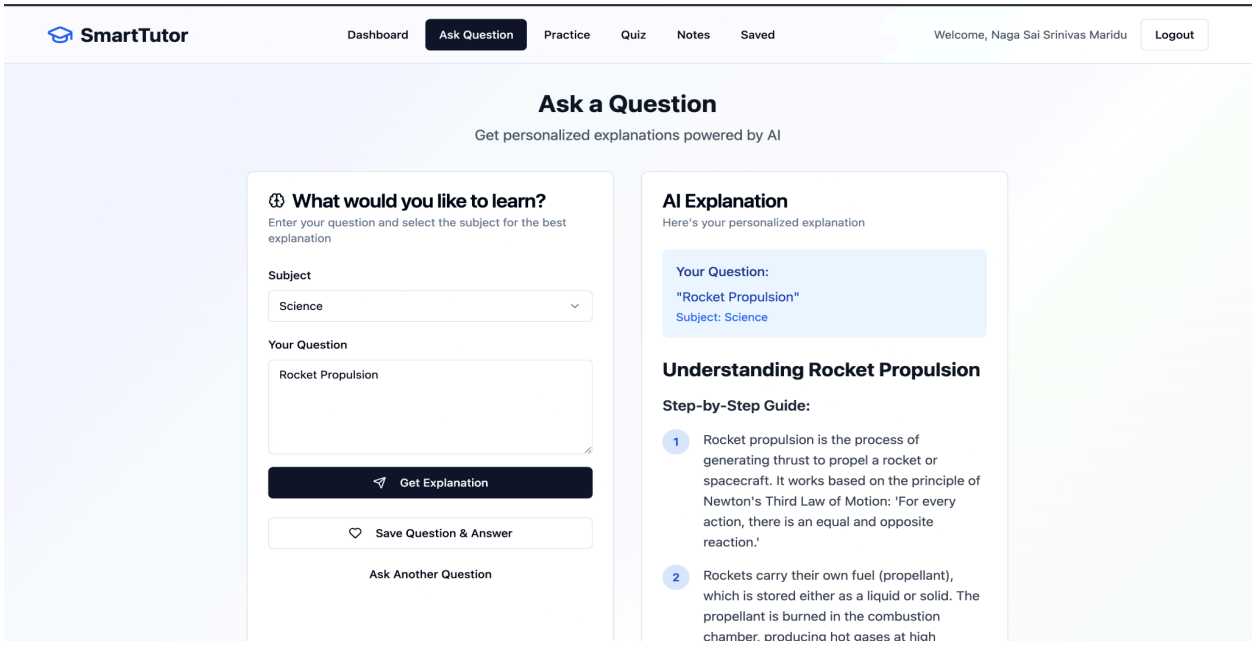


Figure A.2.3: Practice Problem Generation Interface *Description: The interface for generating practice problems, with fields for "Subject" and "Topic."*

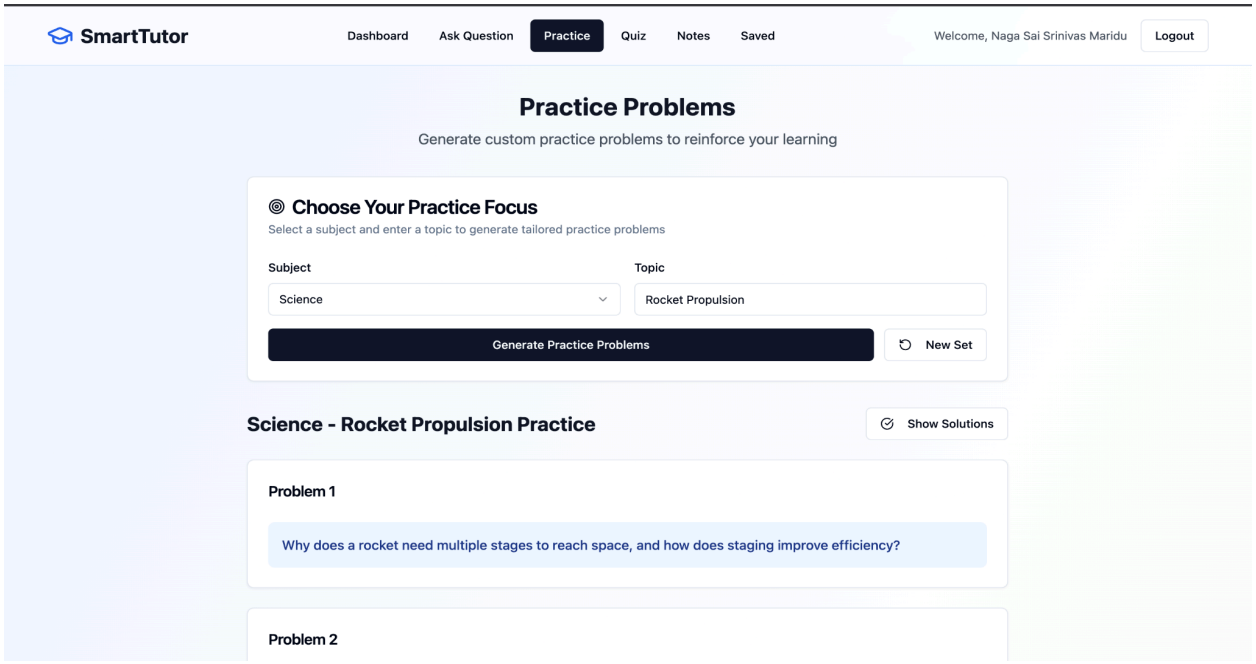


Figure A.2.4: Generated Quiz *Description: A multiple-choice quiz generated by the AI, with questions and selectable options.*

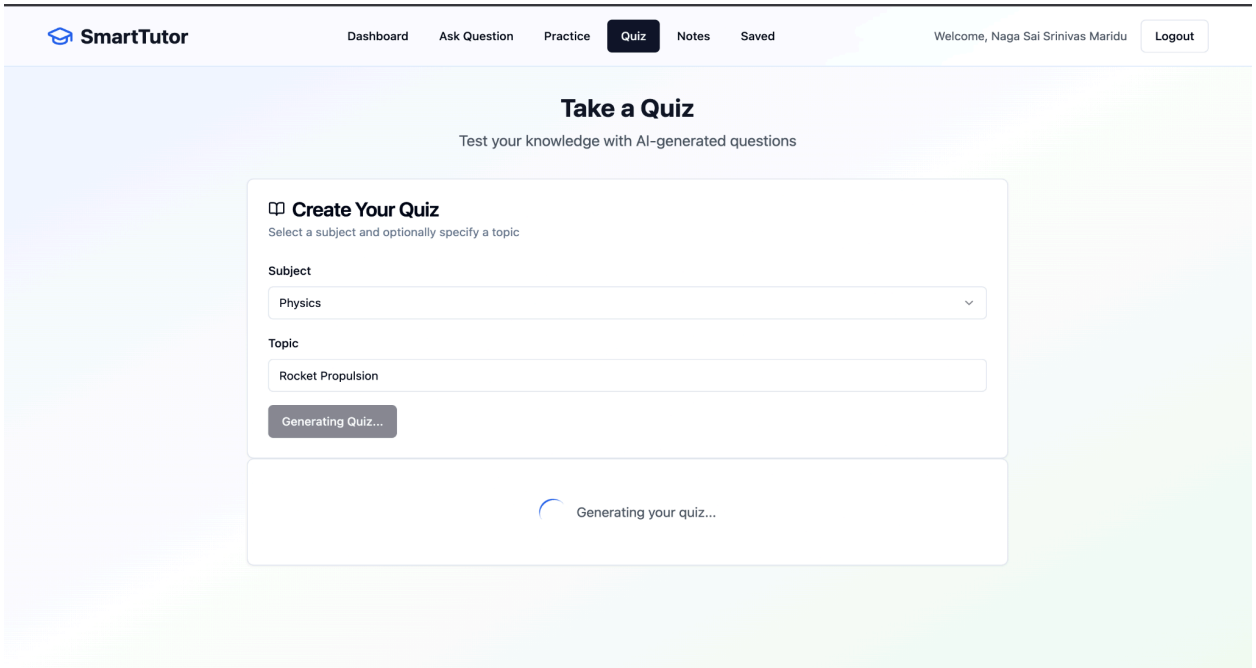
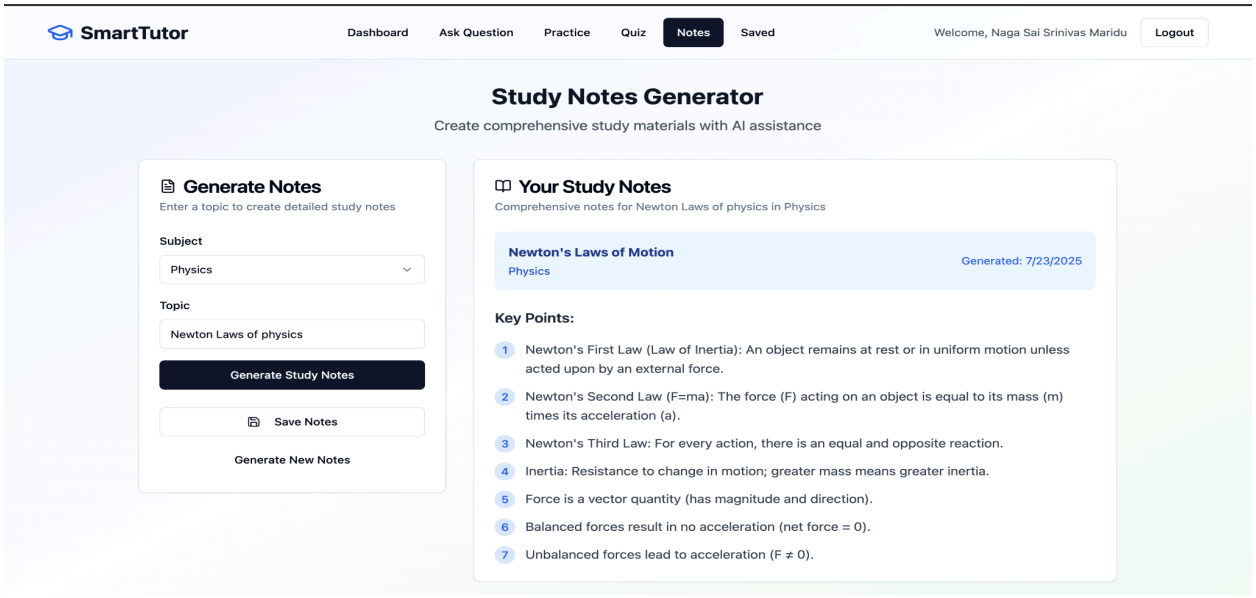
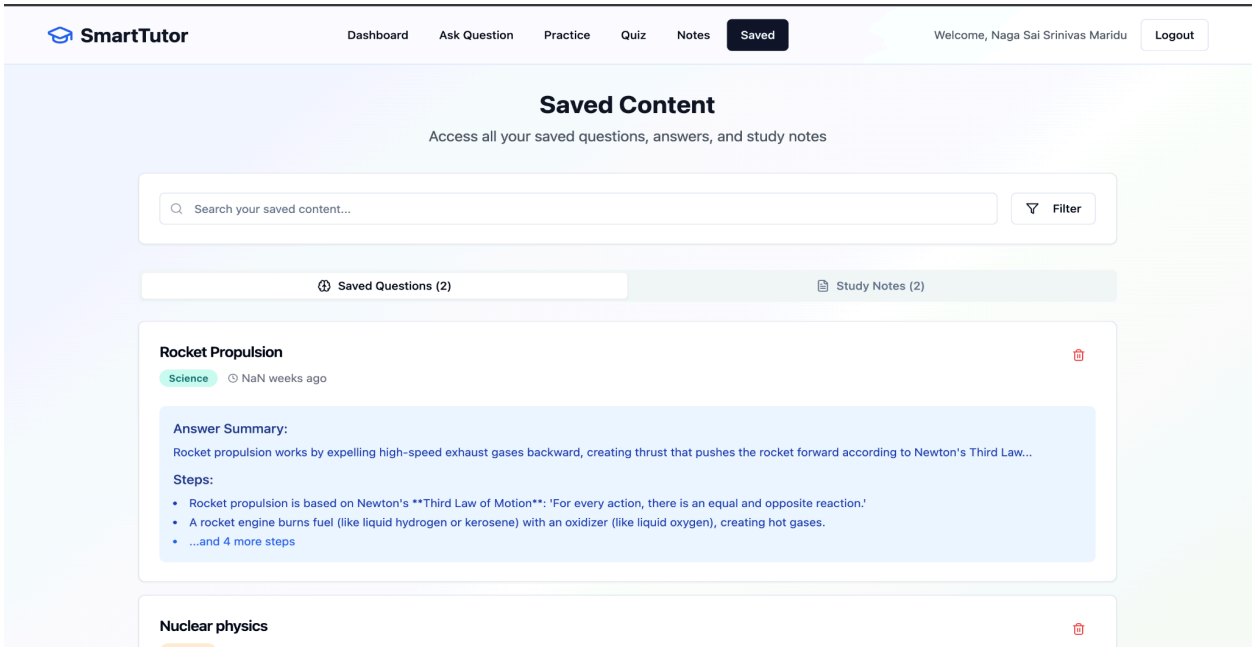


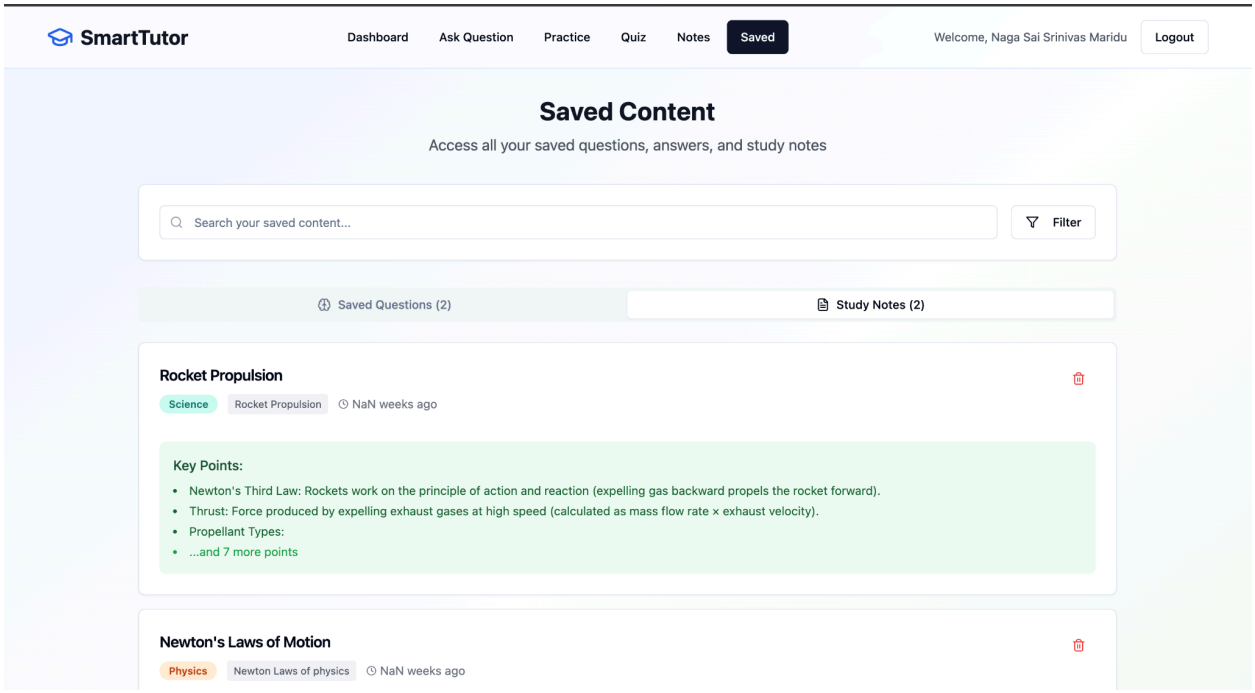
Figure A.2.5: Study Notes Generation Interface *Description: The interface for generating study notes, requiring a "Subject" and "Topic."*



A.3 User Dashboard

Figure A.3.1: Saved Content Dashboard *Description: The user's personal dashboard displaying cards for saved questions and notes, with options to view or delete them.*




SmartTutor
Dashboard
Ask Question
Practice
Quiz
Notes
Saved
Welcome, Naga Sai Srinivas Maridu
Logout
Saved Content
Access all your saved questions, answers, and study notes
Search your saved content...
Filter
Saved Questions (2)
Study Notes (2)
Rocket Propulsion
Science
Rocket Propulsion
NaN weeks ago
Key Points:
Newton's Third Law: Rockets work on the principle of action and reaction (expelling gas backward propels the rocket forward).
Thrust: Force produced by expelling exhaust gases at high speed (calculated as mass flow rate × exhaust velocity).
Propellant Types:
...and 7 more points
Newton's Laws of Motion
Physics
Newton Laws of physics
NaN weeks ago