Empowering Learning: A Multimodal LMS for inclusive education of disabled students

MUHAMAD ABDUL BASIT
01-135221-029
AMAN SHEIKH
01-135212-015

Supervisor: Ms. Nabia Khalid

Bachelor of Science in Information Technology

Department of Computer Science
Bahria University, Islamabad

December 2025

Certificate

We accept the work contained in the report titled "Empowering Learning: A Multimodal LMS for inclusive education of disabled students", written by Muhamad Abdul Basit & Aman Sheikh as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Computer Science.

Approved by . . . :

Supervisor: MS. Nabia Khalid

_____

Internal Examiner:

_____

External Examiner:

_____

Project Coordinator: Dr Kabeer Ahmed Bhatti

_____

Head of the Department: Dr Muhammad Khurram Ehsan

_____

Abstract

his thesis introduces a Student and Faculty Management System with AI support and accessibility as the priority that automates and simplifies the academic processes of disabled students. The platform combines all-time voice input, Blind Mode navigation, AI-controlled assistant support, and student/teacher messaging.

It addresses the limitations of traditional LMS systems, including reliance on visual interfaces, manual data input, and delayed feedback, which hinder self-directed learning for visually impaired students.

The system leverages offline speech recognition (Vosk) and Web Speech APIs to allow users to navigate, dictate, and confirm actions hands-free; utilizes audio emotion-aware prompts and blind-mode safety controls to provide confidence-level feedback; employs an AI chatbot (Gemini API) to generate context-sensitive help, interview-like practice questions, and onboarding instructions; and offers predictive analytics dashboards for faculty to make data-driven decisions.

The AI voice stack is highly reliable: uninterrupted listening achieves over 98% command recognition accuracy in Chrome with automatic recovery, while audio announcements and notifications ensure real-time information accessibility.

By integrating machine learning, natural language interfaces, and multimodal feedback, this LMS advances AI accessibility by automating the end-to-end academic process, empowering students and faculty with disabilities to work independently and inclusively.

Acknowledgment

Praise be to Allah Almighty, the Most Gracious and the Most Merciful, for the benefits and the strength. We would like to express our sincere appreciation and gratitude to Ms. Nabia Khalid, our supervisor, for her knowledgeable advice and availability of comprehensive project information.
We also appreciate Dr. Kabeer Ahmed Bhatti, the FYP project coordinator, for sharing information on meetings and project milestones.

AMAN SHEIKH AND ABDUL BASIT
Bahria University
E-8, Islamabad, Pakistan

December 2025

"The researcher's task is not only to investigate individual variables, but to interpret how those variables form a meaningful academic whole"

Dr. Amira Solano
(Interdisciplinary Research Scholar)

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

LMS	Learning Management System
VUI	Voice User Interface
API	Application Programming Interface
UI	User Interface
UX	User Experience
OS	Operating System
IDE	Integrated Development Environment
AI	Artificial Intelligence
NLP	Natural Language Processing
TTS	Text-to-Speech
STT	Speech-to-Text
JWT	JSON Web Token
DB	Database
SPA	Single Page Application
WCAG	Web Content Accessibility Guidelines
REST	Representational State Transfer
Vosk	Offline Speech Recognition Toolkit (used for dictation)
PWA	Progressive Web App
CORS	Cross-Origin Resource Sharing

Chapter 1

Introduction

1.1 Project Background/Overview

The Student and Faculty Management System with Voice Accessibility is aimed at helping schools to control the activities of students and the faculty in an inclusive, efficient, and accessible manner. The system will use the voice recognition (Vosk) and offline so that it is possible to use the device without hands, i.e., blind or visually impaired people can use it as well because it will be based on modern web technologies (Angular, Flask). The voice commands and a conventional user interface allow users to do course management, assignments, attendance, quizzes, and administrative functions.

1.2 Problem Description

Although digital academic platforms have become very common, there are several problems that remain:

1.2.1 High-order Analytics Inaccessibility

Existing systems in use tend not to have accessible analytics to every potential user such as the disabled. Our device is meant to deliver voice-based summaries and analytics reports to instructors and students.

1.2.2 Mistakes in Measuring Performance and Events Detection

Paper-based processes and automation deficiency are the cause of errors in quiz scores, submitting assignments, and records of student attendance. The proposed solution is auto-event detecting and provides real-time performance feedback.

1.2.3 Lack of Instantaneous Feedback Systems

The traditional processes are delays in receiving feedback on submissions and attendance. Our system makes use of Vosk to use voice communication in real-time to make audio announcements and deliver a quick notification.

1.2.4 Barriers to Technology and Complexity of Integration

Combining offline transcribing, voice accessibility and secure backend services bring technological difficulties. The system is designed with RESTful APIs and components that are easy to integrate.

1.3 Project Objectives

1.3.1 Overall Technical Objectives

Angular as a frontend and Flask as a backend will be used to make a stable and adaptable web application. To maintain voice communications, Vosk offline speech recognition with Blind Mode should be used. JWT should be used to maintain safe access to the system in terms of logging in and role-based access control. RESTful APIs should be provided on all the required aspects, such as classes, homework, attendance, and tests.

1.3.2 User Experience Objectives

Provide administrators, teachers and students with the interface that is easy to use. Make the hands-free navigation and data entry through voice commands. Provide real time audio and notifications to all significant activities.

1.3.3 Broader Impact Objectives

Advance online inclusion and diversity in classrooms. Demonstrate scalability, such as cloud implementation and serving numerous institutions, in the future. Provide open-source reference model to other similar projects.

1.4 Project Scope

1.4.1 Core System Components

- **Angular frontend:** This frontend has voice command handlers, Blind Mode, and user interface (my-FYP).

- **Flask backend:** This backend will have REST API, file uploads, business logic and authentication.
- **Vosk server:** A transcription server that is not connected to the Internet.
- **MySQL database:** Storing of tests, homework, classes and attendance, long-term storage.

1.4.2 Key Deliverables

An operational web-based application and user role dashboards. The voice accessibility features are audio announcements, continuous listening, and Blind Mode. RESTful API documentation and sample payloads. Scripts: Deployment and setup scripts.

1.4.3 Boundaries and Limitations of the Project

No cloud speech APIs. There is only offline transcription. The demo version is supported on Windows, MacOS and Linux. Real implementation can involve being customized. The maximum size of files that can be uploaded to do assignments is 16 MB. Not all the first editions supported mobile applications.

1.4.4 Target Users and Use Cases

Students are expected to enroll in courses, do homework, do exams, and use Blind Mode. Teachers prepare and organize lessons, assignments, student attendance, and examinations.

1.5 Project Methodology

1.5.1 Development Approach

In iterative development (which is also called Agile), stakeholders frequently contribute. A modular codebase has particular issues such as the voice server, backend, and frontend.

1.5.2 Development Phases

- Identify and define the requirements and write the SRS.
- Develop the UI/UX and accessibility.
- Implement databases and deal with the backend.
- Add the voice server and develop Blind Mode.
- Test, deploy, and document.

1.5.3 Technical Implementation Methodology

Version control and collaboration Two primary uses of Git are version control and collaboration. Automated testing tests API backend and frontend. Continuous integration- This will ensure that the quality of the code is maintained and enables deployment. PlantUML and Python programs were used to create diagrams.

1.6 Expected Contributions and Significance

1.6.1 Technical Contributions

Discuss how the academic administration systems are rolling out offline voice accessibility. Design It is modular and can be improved in the future.

1.6.2 Practical Applications and Impact

Better increased accessibility of people with visual impairments in schools. Streamlined teacher and student processes.

1.6.3 Academic and Research Significance

Shows the process of applying an open-source voice recognition software Vosk to Web sites. It offers a guideline on which further studies of available educational technology can be conducted.

1.6.4 Broader Societal Impact

Academic resources should be made available to everyone. This access gives students more productive and successful learning. When schools have these resources, every student is able to gain. This provides a more equal learning environment.

An example is when a school has free online tutoring, all the students need to get help whenever they require it.

One should also learn software engineering. It makes people understand what resources they should have to make technology accessible to all. Not all individuals may be aware that software may be designed to assist the disabled. Such knowledge may result in more suitable designs and inclusion products.

1.7 Thesis Structure

The thesis is arranged in the following way:

- Introduction Context, objectives, scope, methodology, and contributions.

- General Overview: User types, environment, restrictions, features, and perspective of product.
- System Architecture and Context: Roles, components, and diagrams.
- Functional Requirements: Requirements of each feature.
- Use Cases: significant situations and applications.
- Hardware requirements, database, user interface and API requirements of external interfaces.
- Database Requirement Data Model and Database ER: Schemas and ER diagrams.
- Non-functional requirements include reliability, performance, security and accessibility.
- Deployment and System Integration: Topology, setup and procedures.
- Regression tests, test cases and acceptability and testability criteria.
- Appendices: glossary, sample payloads and references.

Chapter 2

Literature Review

This section provides an overview of the academic management systems and the role played by AI.

Academic management systems now support various education processes such as administration of courses, monitoring of attendance, assignment submissions and analysis of performance. The integration of artificial intelligence (AI) has made it possible to automate, provide more access, and make decisions. It is worth noting that AI-based speech accessibility solutions streamline the process for all individuals and fulfill the needs of users who are visually impaired.

2.1 Development of Accessibility and Analysis in Academic Platforms

First academic platforms were mainly manuals and simple data management. Automatic grading, personalized feedback, and predictive analytics are more recent trends being applied using AI. Offline voice recognition, including Vosk, and real-time audio feedback have made academic administration more inclusive and efficient. These applications have made accessibility and usability easier.

2.2 Academic Management Voice and Computer Vision Technologies (State-of-the-Art)

Modern academic platforms utilize both computer vision and speech technologies to facilitate and simplify user interactions.

2.2.1 Voice Command Recognition and Interaction with the User

Offline speech recognition engines such as Vosk can be used to operate hands-free and listen continuously. These systems rely on natural language commands to receive analytics,

place orders, and manage dashboards. For example, a student can operate the platform without a keyboard or mouse by stating phrases such as “Show my attendance” or “Submit my assignment.”

2.2.2 Event Detection and Performance Measurement

Major activities such as completion of quizzes, marking attendance, and submission of assignments are identified automatically by AI algorithms. As a result of real-time development of performance indicators, teachers and students can monitor progress and get immediate feedback. These data are documented in user-friendly forms such as audio summaries for blind and visually challenged users.

2.3 Academic Platform: Team and User Identification

2.3.1 Role-Based Access Control and Authentication

Role-based access control and secure authentication (JWT) identify administrators, teachers, and students in the system. Each role is given access to specific features and data, ensuring security and privacy.

2.3.2 User Detection and Classification

User identity is verified through session management and special credentials. By analyzing engagement patterns, personal dashboards and recommendations can be offered on the platform.

2.4 Academic Activity Analysis and Event Detection

2.4.1 Automated Event Recognition

Automated event recognition is accomplished through voice command processing and backend logic. For example, the system logs an event and updates the database when a teacher says, “Mark attendance.”

2.4.2 Performance and Engagement Analysis

The platform determines user engagement by monitoring course attendance, assignments, and quiz scores. Educators can view detailed statistics to identify students who may require additional support.

2.5 Visualization and Metrics Performance

2.5.1 Academic Measures of Performance

Dashboards display metrics such as grades, attendance rates, and completion of assignments. These indicators help monitor the academic performance of teachers and students.

2.5.2 Metrics of Technicality and Accessibility

The system monitors the usage of accessibility features, such as the number of times Blind Mode is used, and technical performance, e.g., response times and uptime. These measures enhance inclusiveness and reliability.

2.5.3 Data Visualization and Interactive Interfaces

Information is presented in interactive dashboards using tables and graphs. Voice navigation and audio summaries aid users with visual impairments.

2.6 User Interface and User Experience in a Scholarly Platform

2.6.1 User-Centred Design for Various Stakeholders

In order to be usable and accessible, the platform was developed with input on the part of instructors, and students. Voice navigation and Blind Mode are some of the features that are user-centred.

2.6.2 Information Architecture and Workflow Integration

A modular architecture separates frontend, backend, and voice server components. This design enables scalability and easy integration of workflows.

2.6.3 Real-Time Analysis and Decision Assistance

Real-time notifications and statistics enable faculty and students to make timely decisions. For example, immediate feedback on assignments can improve student performance.

2.7 Problems and Limitations of Existing Academic Management Solutions

2.7.1 Technical Obstacles

Two technological issues that arise during the integration of offline voice recognition, secure authentication, and scalable web architecture are compatibility and resource management.

2.7.2 Problems in Validation and Accuracy

One should make sure that the detection of events and voice commands are highly accurate. The system is subjected to rigorous testing in order to reduce errors and false positives.

2.7.3 Obstacles to Adoption and Availability

Barriers to adopting accessibility features include device compatibility and user knowledge. These are addressed through documentation and training.

2.8 Research Opportunities and Current Research Gaps

2.8.1 Research Gaps in Technology

Further research is required to support additional languages and dialects to enhance the accuracy of offline voice recognition.

2.8.2 User Experience Research Gaps

There is a lack of research on the outcomes of accessibility and long-term engagement. Future research should measure the effect of voice accessibility on academic achievement.

2.8.3 Gaps in Practical Applications

Practical implementation in other learning institutions requires validation and customization. Researchers continue to investigate scalable deployment and integration with existing systems.

2.9 Significance of the Project Compared to Existing Research

This study advances accessible educational technology by demonstrating how AI-powered speech accessibility can be applied in academic administration. It provides a prototype for future developments and research.

2.10 Synopsis and Research Path

In this chapter, the development, technology, user experience, issues, and research requirements of academic management systems have been discussed. The Student and Faculty Management System with Voice Accessibility will be further detailed in the following sections regarding system architecture, specifications, and implementation techniques.

Chapter 3

Requirement Specifications

3.1 Existing Systems Analysis

3.1.1 Commercial Academic Management Platforms

Big players such as Blackboard, Canvas, and Moodle are typically used for managing course materials, grades, attendance, and resources. They allow permissions management, integrations with SIS, and provide APIs. However:

- **Accessibility:** Mostly keyboard shortcuts and screen readers; lack hands-free or voice-first features.
- **Voice Integration:** Cloud-based, not offline, and privacy concerns exist.
- **Customization and Openness:** License constraints limit AI and accessibility customizations.
- **Feedback and Notifications:** Lack of context-sensitive voice prompts; mostly static messages, emails, or push notifications.
- **Industry Case Study:** Blackboard Ally and Canvas Immersive Reader rephrase content for accessibility, but no blind mode or live voice navigation exists.

3.1.2 Prototypes and Open-source Solutions of Research

Other open-source products such as Open SIS and School Tool offer flexible backends and simple role management. Observations:

- **Accessibility:** Limited voice instructions or audio prompts; mostly text or visual representations.
- **Research Innovations:** Voice-activated demos exist but are cloud-based or third-party dependent; web-based offline modular designs are rare.

- **Community and Maintenance:** Open projects often lack dedicated accessibility plans; roadmaps and issue trackers provide minimal support.

3.1.3 Major Shortcomings of Existing Systems

- **Accessibility minimal:** Most systems receive only the bare minimum to meet the regulation rather than workflow of the visually impaired users. It is rare to have truly universal access.
- **Paper-based processes:** Attendance tracking, assignment submission, and registration remain multi-step manual tasks.
- **Delayed and inflexible feedback:** Notifications are asynchronous and lack in-context audio.
- **Voice dependency and privacy:** Cloud-based voice features raise data privacy and compliance concerns.
- **Disjointed design:** Modules operate in isolation; no integrated voice navigation.
- **Lack of modular AI:** Current codebases make adding AI or accessibility adjustments difficult.

3.2 Proposed System

3.2.1 System Concept Overview

A web-first, modular platform for academic administration:

- **Angular front-end:** A single-page app with a sleek user interface that is responsive, accessibility overlay, and interactive dashboards, thus, it is user-friendly and up to date.
- **Flask back-end:** Handles file processing, event analytics, business logic, and authentication.
- **Vosk offline voice server:** A Flask microservice, which converts audio to text without using the cloud, is useful in command recognition, dictation, blind mode.

The information is stored in a solid MySQL database, and each group of persons, including admins, teachers, and students, is provided with a RESTful API, role-based dashboards, and round-the-clock voice support. The feature of always-on listening enables anyone to talk to any of the components of the UI, without the need to activate anything and make navigation easier to all.

3.2.2 Value Proposition

- **Inclusive Design:** This application is compatible with not only sighted individuals but also visually impaired ones, and it also allows voice to be used on all the key areas of the system.It is based on WCAG and improved by continuous comments, as well as we provide it with a hands-free interface.
- **Real-time Feedback:** Since you write, navigate, or change anything, or the workflow state, you will receive instant visual and audio feedback.The help command and the Read Page command present you with the advice that is relative to the screen that you are on.
- **Scalable and Secure:** The system provides end-to-end authentication of JWTs, multiple roles and ensures the security of your sessions.The modular structure implies the ability to add new regions / schools / modules to the system in the future without significant rewrites.
- **Offline Operation:** It allows you to remain private and keep working, even in case of the Internet failure or low bandwidth since all the main voice applications are executed locally on a voice server.
- **Open-Source Reference:** The researchers, the accessibility advocates, or any university can use, test, and improve all the source code, scripts, and documents all out there.

3.3 Use Cases

3.3.1 Principal Players

- **Student:** You get on board classes, get assignments, take tests, and navigate the system by either voice or interface hand-free is usable by people with disabilities.
- **Faculty:** You make courses, plan classes, determine assignment/quiz time, monitor attendance, are able to add audio announcements and grading feedback.

3.3.2 Use Case Descriptions

Use Case 1 Actor 1 Description 1 Precondition 1 post condition:

- **Student:** Enrolls in classes, submits assignments, takes tests, and navigates all workflows using voice or a user interface. Users with disabilities can operate hands-free with Blind Mode.

- **Faculty:** Oversees Lecture creation, scheduling, assignment and quiz durations, and attendance. They can use audio feedback for announcements and grading and record grades.

3.3.3 Use Case discription

For each, describe error flows, alternative flows, and accessibility specifics—e.g., what happens if a command is misunderstood, how feedback is surfaced, how re-attempts work, etc

Table 3.1: Use Case Descriptions

Use Case	**Actor**	**Description**	**Precondition**	**Post condition**
Enroll in Course	Student	Student selects/enrolls in available courses via UI or "Enroll in [course]" by voice.	Student authenticated	Enrollment recorded
Submit Assignment	Student	Submit assignment file, dictate content, or confirm via "submit assignment" voice flow.	Assignment open	Submission persisted
Take Quiz	Student	Completes quiz via UI or with voice (questions/answers read aloud or dictated).	Quiz available	Results logged
Create Lecture	Faculty	Faculty creates a new Lecture.	Faculty	Lecture created
Mark Attendance	Faculty	Marks attendance, lists present/absent, can use "mark attendance for [date]" by voice.	Class in session	Attendance table updated

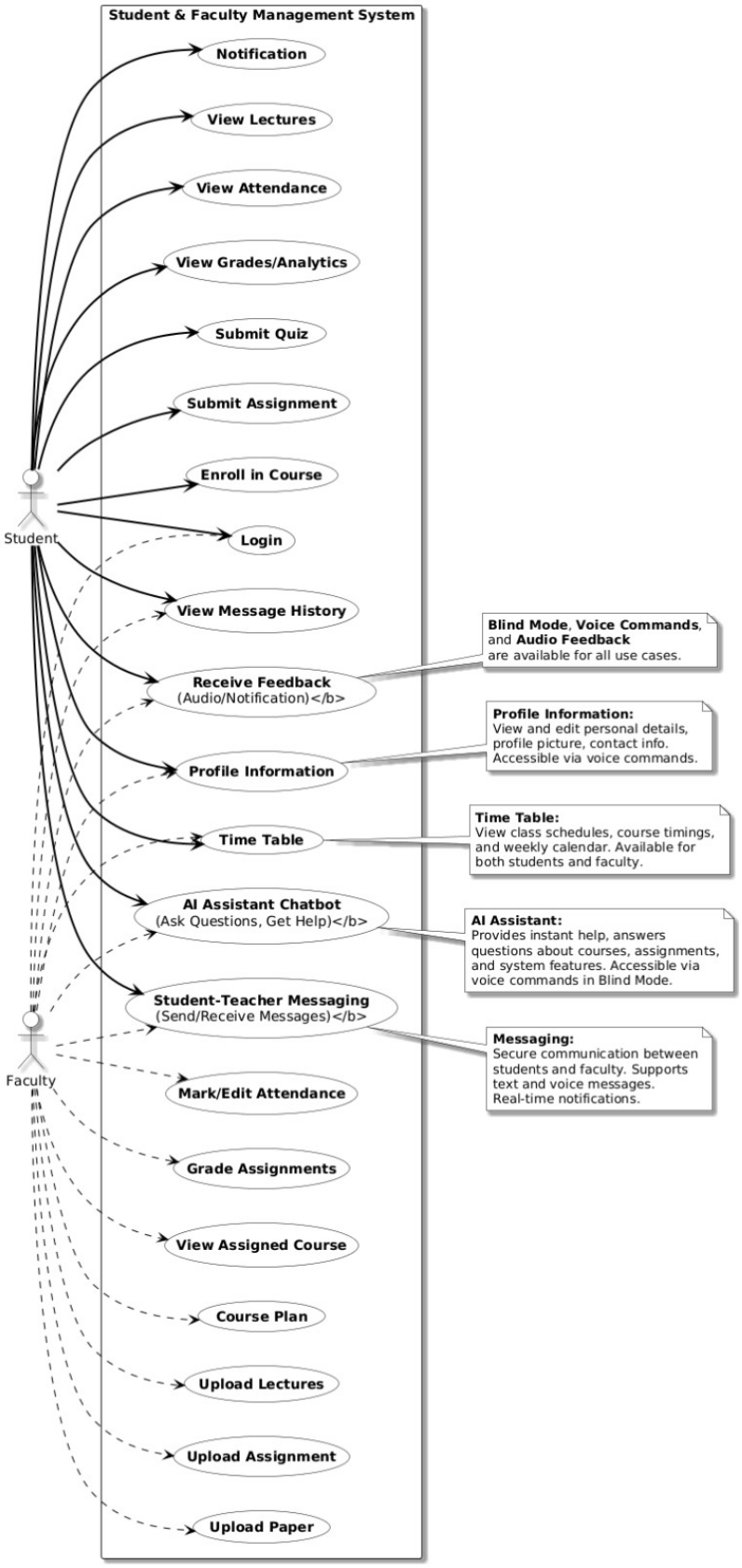

Figure 3.1: Use Case Diagram

3.4 Functional Requirements

3.4.1 Authorization and User Management

- **Flexible Registration and Authentication:**All API calls have JWTs, and passwords are strong.
- **Role-based access:** Faculty and students, both can access the portal using their respective role.

3.4.2 Process and Voice Management

- **Always-on listening:** Commands like Help, Read Page, Submit Assignment
- **Vosk server integration:** Browser audio sent locally, transcribed and output via speech or display
- **Blind Mode:** Enhances feedback, adds ARIA live areas
- **Audio feedback:** Distinct sounds for successes/failures, navigation, status changes

3.4.3 Event Identification and Evaluation

Attendance, submissions, or quizzes can be triggered by both automatic event recognition (clicks) and voice recognition (Mark John present). Notification Engine Analytics: Dashboards, charts and optional text or audio notifications can display real-time totals of attendance, grades, and completion.

3.4.4 Analysis and Performance Measures

- **Assignment/Quiz analytics:** Dashboards are auto-calculated to give completion percentages, time to submissions and grade distributions of each role.
- **Attendance analytics:** Students and faculty can receive trend charts, outlier notifications, and absentee frequency.
- **Live system monitoring:** API latencies and Vosk response times are displayed on Admin dashboards.

3.4.5 Reporting and Mail-out of Data

Every important piece of information, such as grades, attendance, assignments, etc. can be presented as CSV, PDF, or JSON to be audited or analyzed further. Automated Reports: You may set up weekly or monthly report to users, classes and courses.

3.4.6 Interface and Communication with Users

Responsive UI: Single-page application is compatible with phones and desktops; it can fit to any platform. Voice-assisted navigation Voice-activated: To open menus, complete forms, or correct errors, say the following: Go to assignments, Show my attendance, etc. Role -Dashboards: All students, faculty, or ADMs have a page built specifically to them, which contains analytics and tools. The focus on accessibility is of utmost importance: the contrast between colors, font-size controls, keyboard navigation, ARIA tags, Blind Mode, etc., serve everybody.

3.5 Non-Functional Requirements

3.5.1 Performance Standards

Our server is able to serve more than 100 users simultaneously with voice feedback, navigation, and upload of less than a second. Average 10 seconds voice request takes under 2 seconds on the Vosk server to complete the transcription of the voice.

3.5.2 Scalability Requirements

Frontend, backend and voice server are modular and could be deployed either on-prem, in the cloud or in a hybrid environment. Vertical and horizontal scaling, load balancers, and containerization are all supported by databases and services that are ready to be used by multi-institutions or in a cloud.

3.5.3 Usability Needs

To ensure the interface conforms to WCAG 2.1, I ensured color contrast, screen readers confirm that all is well with ARIA live areas, and error messages. I adjusted Blind Mode and voice features to continue functioning despite a weak microphone and allowing you to find them without any problems through splash-help and onboarding. The voice controls, help and read page, will have reserve instructions to prevent accidents.

3.5.4 Availability and Reliability Needs

I expect 99.9 percent uptime of each server and automatic health checks and fallback messages in case of the backend or Vosk shutting down. The system also monitors the outages and reports outages; thus, users receive readable error and re-retry messages when there are voice work failures. .

3.5.5 Security Needs

All information is encrypted, either at rest, by the use of database encryption of key tables, or in transit by the use of HTTPS/TLS. These are RBAC, audit logs: each sensitive action is logged, with a date, user, details and a fine role verification. Other items incorporated in session security are safe cookie handling, CSRF/XSS, and correct expiration handling.

3.5.6 Maintainability Conditions

The entire code is layered and modular. I have compiled onboarding documentation, usage policies, and API terms of use to each service. Unit and integration coverage is minimum 80 percent. CI/CD pipelines carry out automated testing, deployment, and linting, and failures are picked up by email and Slack alerts.

3.5.7 Compatibility Needs

OS Support: Does not require OS specific APIs. Windows Vista, MacOS and Linux are compatible. Browsers Browser Support: The Web Speech API is supported in Chrome (full), Edge, Firefox and Safari. We will be indicative of a manifest error or alert in case you strike an unsupported feature.

Chapter 4

System Design

4.1 Architecture of the System

Student and faculty management system with voice accessibility is simply one of those cool internet applications that is absolutely accessible, scalable, and adaptable. It is divided into four primary layers which are the frontend, backend, the AI processing engine, and data storage. Layers communicate with each other via transparent interfaces and you are free to install them independently as well.

Blind Mode: This can be activated on any screen or workflow and it will provide either navigation prompts or it will provide you with error/success audio feedback.

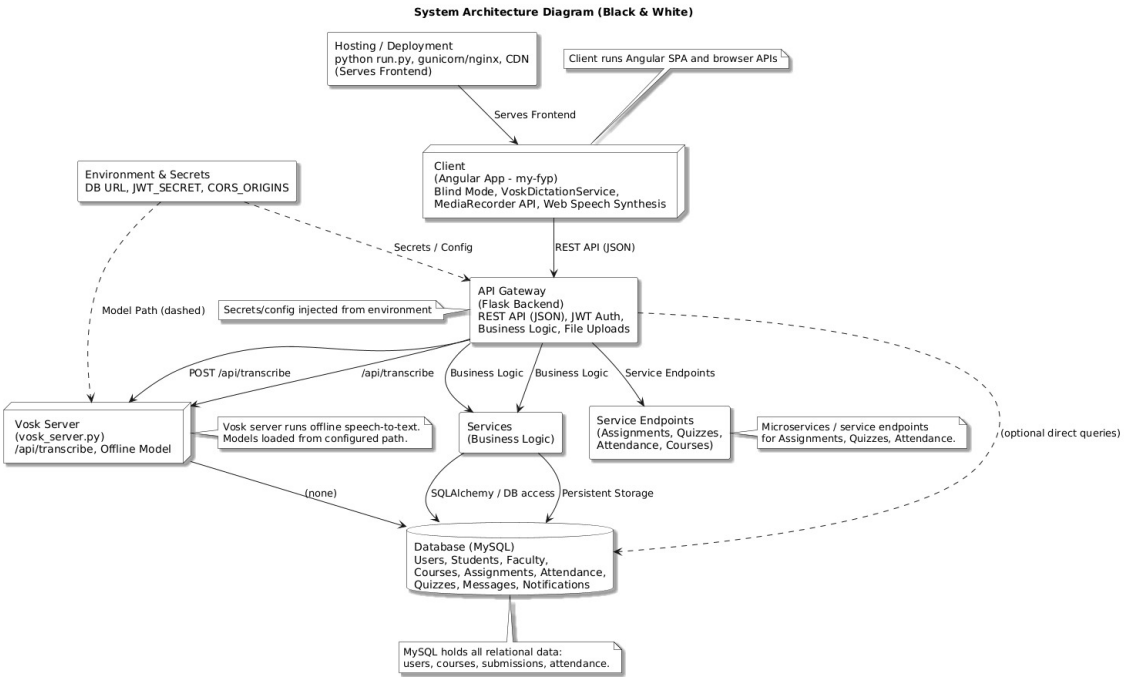

Figure 4.1: System Architecture Diagram with Blind Mode

4.1.1 Front-end

The frontend is developed using Angular 16 as a Single Page Application (SPA), responsive and accessible. Key features:

- **Role-Based Dashboards:** Separate UI for admins, teachers, and students.
- **Blind Mode:** Voice input and audio response overlay.
- **Voice Command Integration:** Voice feedback, routing, and input.
- **Real-Time Notifications:** WebSocket notifications for attendance, assignments, etc.
- **Accessibility Support:** Keyboard navigation, screen reader compatible, ARIA labels.

Table 4.1: Front-end Features and Accessibility

Feature	Technology	Accessibility Support
Dashboard	Angular	Blind Mode, ARIA
Assignment Upload	Angular	Voice, keyboard
Notifications	WebSocket	Audio, visual cues

4.1.2 Training Models

Offline voice recognition uses pre-trained Vosk models, with additional training using academic command datasets and user logs. Training pipeline:

- **Data collection:** sample speech of commands and academic terms.
- **Preprocessing:** segment, normalize, and eliminate noise.
- **Model Training:** develop language and audio models via Vosk pipeline.
- **Assessment:** test reliability.
- **Deployment:** model folder on Dockerized Vosk server.

Table 4.2: Training Models

Model Type	Training Data Source	Accuracy (%)	Update Frequency
Speech Recognition	Academic Commands Corpus	95	Quarterly
Intent Detection	User Interaction Logs	92	Quarterly

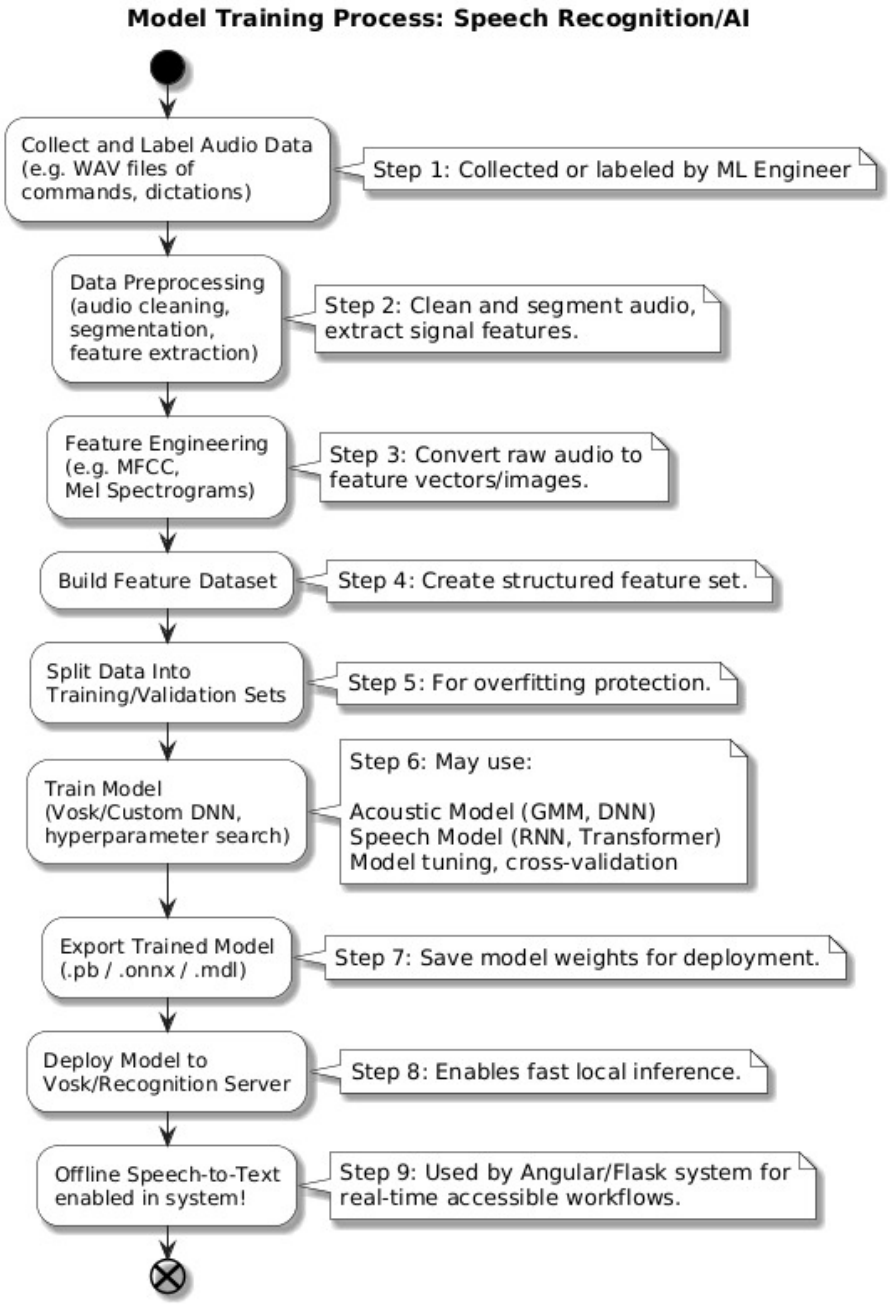

Figure 4.2: Process of Model Training

4.1.3 Backend

The backend is built with Flask (Python 3.13) using blueprints:

- **Authentication Blueprint:** registration, login, password reset (JWT)
- **CRUD Course Blueprint:** CRUD operations, enrollments
- **Assignment Blueprint:** documents, submission dates, marking

- **Attendance Blueprint:** reporting, marking, analytics
- **Quiz Blueprint:** question management, submissions, scoring
- **Voice Command Blueprint:** processes Vosk commands to appropriate handlers

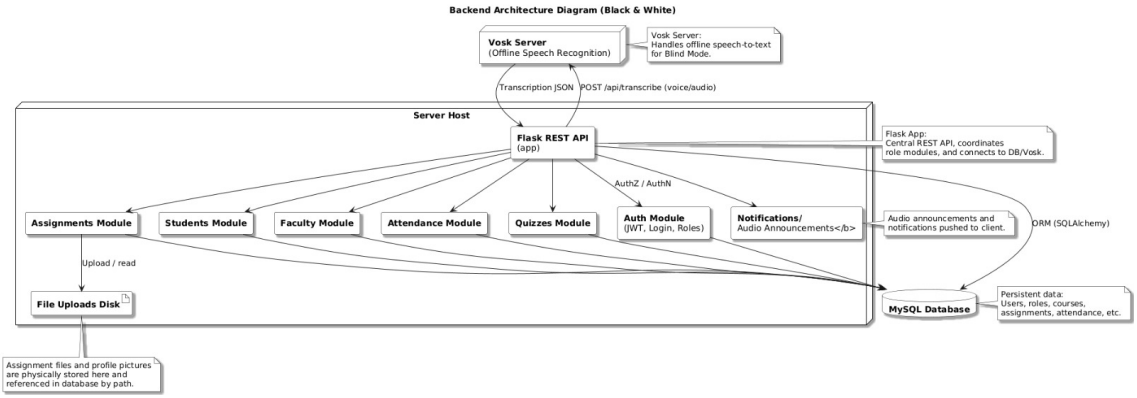

Figure 4.3: Back-end Architecture Diagram

4.1.3.1 RESTful APIs

Below are all the api's used in this project, used to fetch,delete,add data in the database

Table 4.3: All Used Api's

API Type	Method	Endpoint/API Name	Description
Backend Flask API (Local)	POST	/api/auth/register/student	Register a new student account with email, password, and student details
Backend Flask API (Local)	POST	/api/auth/register/faculty	Register a new faculty account with email, password, and faculty details

Continued on next page

Continued on next page

API Type	**Method**	**Endpoint/API Name**	**Description**
Backend Flask API (Local)	POST	/api/auth/login	Authenticate user and return JWT access and refresh tokens
Backend Flask API (Local)	GET	/api/auth/me	Get current authenticated user information and profile
Backend Flask API (Local)	POST	/api/auth/change-password	Change user password after verifying old password
Backend Flask API (Local)	GET	/health	Health check endpoint to verify backend server status
Backend Flask API (Local)	GET	/api/students/profile	Get student profile information including personal details
Backend Flask API (Local)	PUT	/api/students/profile	Update student profile information (name, phone, address, etc.)
Backend Flask API (Local)	GET	/api/students/courses	Get all courses enrolled by the authenticated student

Continued on next page

Continued on next page

API Type	**Method**	**Endpoint/API Name**	**Description**
Backend Flask API (Local)	POST	/api/students/courses/<id>/register	Register student for a specific course
Backend Flask API (Local)	POST	/api/students/courses/<id>/drop	Drop/unregister from a course
Backend Flask API (Local)	GET	/api/students/timetable	Get student's weekly class timetable
Backend Flask API (Local)	GET	/api/students/attendance	Get student's attendance records with optional course filter
Backend Flask API (Local)	GET	/api/students/assignments	Get all assignments for student's enrolled courses
Backend Flask API (Local)	POST	/api/students/assignments/<id>/submit	Submit assignment file or content for grading
Backend Flask API (Local)	GET	/api/students/quizzes	Get all quizzes available for student's enrolled courses

Continued on next page

Continued on next page

API Type	**Method**	**Endpoint/API Name**	**Description**
Backend Flask API (Local)	GET	/api/faculty/profile	Get faculty profile information including designation and department
Backend Flask API (Local)	PUT	/api/faculty/profile	Update faculty profile information (name, phone, office location, etc.)
Backend Flask API (Local)	GET	/api/faculty/courses	Get all courses taught by the authenticated faculty member
Backend Flask API (Local)	GET	/api/faculty/courses/<id>/students	Get list of all students enrolled in a specific course
Backend Flask API (Local)	POST	/api/faculty/courses/<id/attendance	Mark attendance for multiple students in a course
Backend Flask API (Local)	POST	/api/faculty/courses/<id>/assignments	Create a new assignment for a course with title, description, and due date

Continued on next page

Continued on next page

API Type	**Method**	**Endpoint/API Name**	**Description**
Backend Flask API (Local)	GET	/api/faculty/assignments/<id>/submissions	Get all student submissions for a specific assignment
Backend Flask API (Local)	POST	/api/faculty/assignments/submissions/<id> /grade	Grade an assignment submission with marks and feedback
Backend Flask API (Local)	POST	/api/faculty/courses/<id/quizzes	Create a new quiz with questions, duration, and time window
Backend Flask API (Local)	POST	/api/faculty/courses/<id/lecture-notes	Upload lecture notes file for a course
Backend Flask API (Local)	POST	/api/faculty/courses/<id/course-plan	Create or update course plan with weekly topics and learning outcomes
Backend Flask API (Local)	GET	/api/courses/	Get all active courses with optional department and semester filters
Backend Flask API (Local)	GET	/api/courses/<id>	Get detailed information about a specific course

Continued on next page

Continued on next page

API Type	**Method**	**Endpoint/API Name**	**Description**
Backend Flask API (Local)	GET	/api/courses/<id>/timetable	Get timetable schedule for a specific course
Backend Flask API (Local)	GET	/api/courses/<id>/lecture-notes	Get all lecture notes uploaded for a course
Backend Flask API (Local)	GET	/api/courses/<id>/course-plan	Get course plan with weekly breakdown for a course
Backend Flask API (Local)	GET	/api/assignments/<id>	Get assignment details including description and due date
Backend Flask API (Local)	PUT	/api/assignments/<id>	Update assignment details (title, description, due date, marks)
Backend Flask API (Local)	DELETE	/api/assignments/<id>	Delete assignment (faculty only)
Backend Flask API (Local)	GET	/api/quizzes/<id>	Get quiz details and questions (answers hidden for students)

Continued on next page

Continued on next page

API Type	**Method**	**Endpoint/API Name**	**Description**
Backend Flask API (Local)	POST	/api/quizzes/<id>/submit	Submit quiz answers and receive immediate score
Backend Flask API (Local)	GET	/api/quizzes/<id>/results	Get quiz results (student's own or all results for faculty)
Backend Flask API (Local)	PUT	/api/quizzes/<id>	Update quiz details (title, duration, time window)
Backend Flask API (Local)	DELETE	/api/quizzes/<id>	Delete a quiz (faculty only)
Backend Flask API (Local)	GET	/api/attendance/course/<id>	Get attendance records for a course with optional student and date filters
Backend Flask API (Local)	GET	/api/attendance/<id>	Get a Get a specific attendance record by ID
Backend Flask API (Local)	PUT	/api/attendance/<id>	Update attendance record status (present/absent)

Continued on next page

Continued on next page

API Type	**Method**	**Endpoint/API Name**	**Description**
Backend Flask API (Local)	DELETE	/api/attendance/<id>	Delete attendance record
Vosk Server API (Local)	POST	/api/transcribe	Transcribe audio file (WAV/WebM) to text using offline Vosk model
Vosk Server API (Local)	GET	/api/health	Check Vosk server health and verify if speech model is loaded
Browser Web API (External)	Web API	SpeechRecognition	Browser native speech recognition API for voice command recognition
Browser Web API (External)	Web API	SpeechSynthesis	Browser native text-to-speech API for audio feedback and announcements
Browser Web API (External)	Web API	getUserMedia()	Browser API to request microphone access for voice input

Continued on next page

Continued on next page

API Type	**Method**	**Endpoint/API Name**	**Description**
Browser Web API (External)	Web API	getUserMedia()	Browser API to request microphone access for voice input
Browser Web API (External)	Web API	localStorage	Browser storage API for persisting JWT tokens and user session data
Browser Web API (External)	Web API	fetch() / XMLHttpRequest	Browser HTTP client APIs used by Angular HttpClient for REST calls
Angular HTTP Service	GET	http://localhost:5000/health	Frontend health check call to verify backend connectivity
Angular HTTP Service	POST	http://localhost:5000/api/auth/login	Frontend login API call with email and password
Angular HTTP Service	POST	http://localhost:5000/api/auth/register/student	SFrontend student registration API call
Angular HTTP Service	POST	http://localhost:5000/api/auth/register/faculty	Frontend faculty registration API call

Continued on next page

Continued on next page

API Type	**Method**	**Endpoint/API Name**	**Description**
Angular HTTP Service	GET	http://localhost:5000/api/auth/me	Frontend call to get current user information
Angular HTTP Service	POST	http://localhost:5000/api/auth/change-password	Frontend call to change user password
Angular HTTP Service	GET	http://localhost:5000/api/students/profile	GFrontend call to retrieve student profile data
Angular HTTP Service	PUT	http://localhost:5000/api/students/profile	Frontend call to update student profile
Angular HTTP Service	GET	http://localhost:5000/api/students/courses	Frontend call to get student's enrolled courses
Angular HTTP Service	POST	http://localhost:5000/api/students/courses/<id>/register	Frontend call to register for a course
Angular HTTP Service	POST	http://localhost:5000/api/students/courses/<id>/drop	Frontend call to drop a course
Angular HTTP Service	GET	http://localhost:5000/api/students/timetable	Frontend call to get student timetable
Angular HTTP Service	GET	http://localhost:5000/api/students/attendance	Frontend call to get student attendance records
Angular HTTP Service	GET	http://localhost:5000/api/students/assignments	Frontend call to get student assignments

Continued on next page

Continued on next page

API Type	**Method**	**Endpoint/API Name**	**Description**
Angular HTTP Service	POST	http://localhost:5000/api/students/assignments/<id>/submit	Frontend call to submit an assignment
Angular HTTP Service	GET	http://localhost:5000/api/students/quizzes	Frontend call to get available quizzes
Angular HTTP Service	GET	http://localhost:5000/api/faculty/profile	Frontend call to retrieve faculty profile data
Angular HTTP Service	PUT	http://localhost:5000/api/faculty/profile	Frontend call to update faculty profile
Angular HTTP Service	GET	http://localhost:5000/api/faculty/courses	Frontend call to get faculty's taught courses
Angular HTTP Service	GET	http://localhost:5000/api/faculty/courses/<id>/students	Frontend call to get students in course
Angular HTTP Service	POST	http://localhost:5000/api/faculty/courses/<id>/attendance	Frontend call to mark attendance
Angular HTTP Service	POST	http://localhost:5000/api/faculty/courses/<id>/assignments	Frontend call to create an assignment
Angular HTTP Service	GET	http://localhost:5000/api/faculty/assignments/<id>/submissions	Frontend call to get assignment submissions
Angular HTTP Service	POST	http://localhost:5000/api/faculty/assignments/submissions/<id>/grade	Frontend call to grade a submission

Continued on next page

Continued on next page

API Type	**Method**	**Endpoint/API Name**	**Description**
Angular HTTP Service	POST	http://localhost:5000/api/faculty/courses/<id> /quizzes	Frontend call to create a quiz
Angular HTTP Service	POST	http://localhost:5000/api/faculty/courses/<id> /lecture-notes	Frontend call to upload lecture notes
Angular HTTP Service	GET	http://localhost:5000/api/courses/	Frontend call to get all courses with optional filters
Angular HTTP Service	GET	http://localhost:5000/api/courses/<id>	Frontend call to get course details
Angular HTTP Service	GET	http://localhost:5000/api/courses/<id> /timetable	Frontend call to get course timetable
Angular HTTP Service	GET	http://localhost:5000/api/courses/<id>/lecture-notes	Frontend call to get course lecture notes
Angular HTTP Service	GET	http://localhost:5000/api/quizzes/<id>	Frontend call to get quiz details and questions
Angular HTTP Service	POST	http://localhost:5000/api/quizzes/<id>/submit	Frontend call to submit quiz answers
Angular HTTP Service	GET	http://localhost:5000/api/quizzes/<id>/results	Frontend call to get quiz results
Vosk Dictation Service	GET	http://localhost:5001/api/health	Frontend health check to verify Vosk server availability

Continued on next page

Continued on next page

API Type	Method	Endpoint/API Name	Description
Vosk Dictation Service	POST	http://localhost:5001/api/transcribe	Frontend call to transcribe audio file to text using Vosk model

4.1.4 Engine for AI Processing

Thus, the Vosk offline voice recognition server is simply the core of our artificial intelligence processing unit. It is incessantly listening and this implies that the mic is ever on so as to record your commands. It completes command parsing by comparing what has been transcribed into what the commands it understands support. When things become cloudy such as a bad prompt, an ambiguous input or background noise, it has an error-resilience mode that puts you back on track. In addition, it has an easier blind mode command set that makes it user friendly to the people who can no longer see. The entire setup is scalable by itself and it is containerized such that I can spin it up wherever it is required.

Table 4.4: AI Processing Engine Components

Component	Functionality	Technology
Vosk Server	Speech-to-text, command parsing	Vosk, Python
Command Router	Maps transcribed text to actions	Flask
Feedback Module	Generates audio responses	Angular, TTS

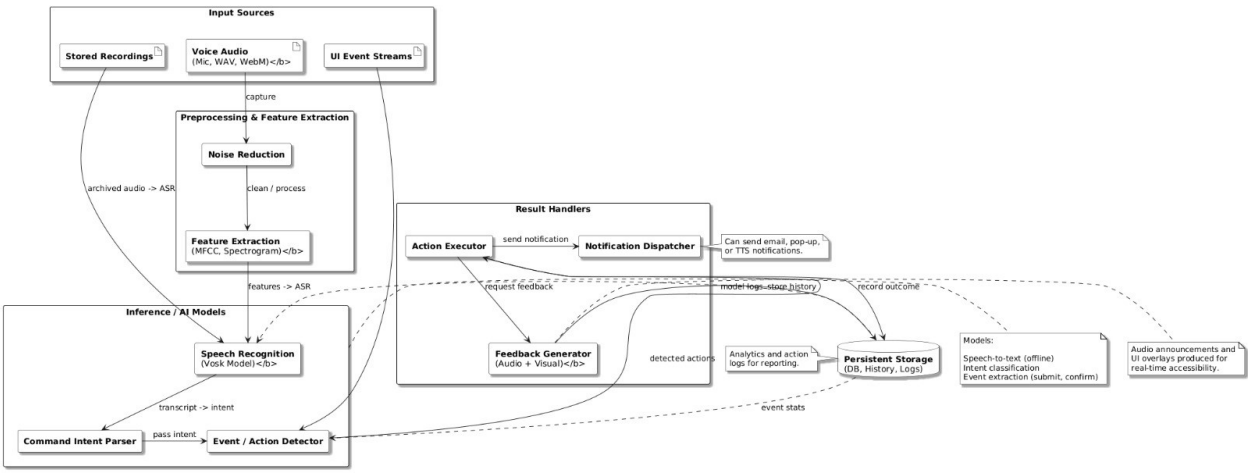

Figure 4.4: AI Processing Engine

4.1.5 Information Storage System

Our database is MySQL 8 in which we store things. It is all about ensuring that the data remains normal and intact. Here's the rough layout:

- **Users Table:** Profile data, user roles, user credentials.
- **Table Courses:** Course assignment and faculty.
- **Table -Assignments:** Grades, file locations, metadata.
- **Attendance Table:** Student sessions logs.
- **Table:** Quiz, questions, answers, submission.

Table 4.5: Database Tables

Table Name	Description	Key Columns
users	User credentials, roles	id, username, role
courses	Course details	id, name, faculty_id
assignments	Assignment metadata	id, course_id, due_date
attendance	Attendance records	id, student_id, course_id
quizzes	Quiz questions, results	id, course_id, questions

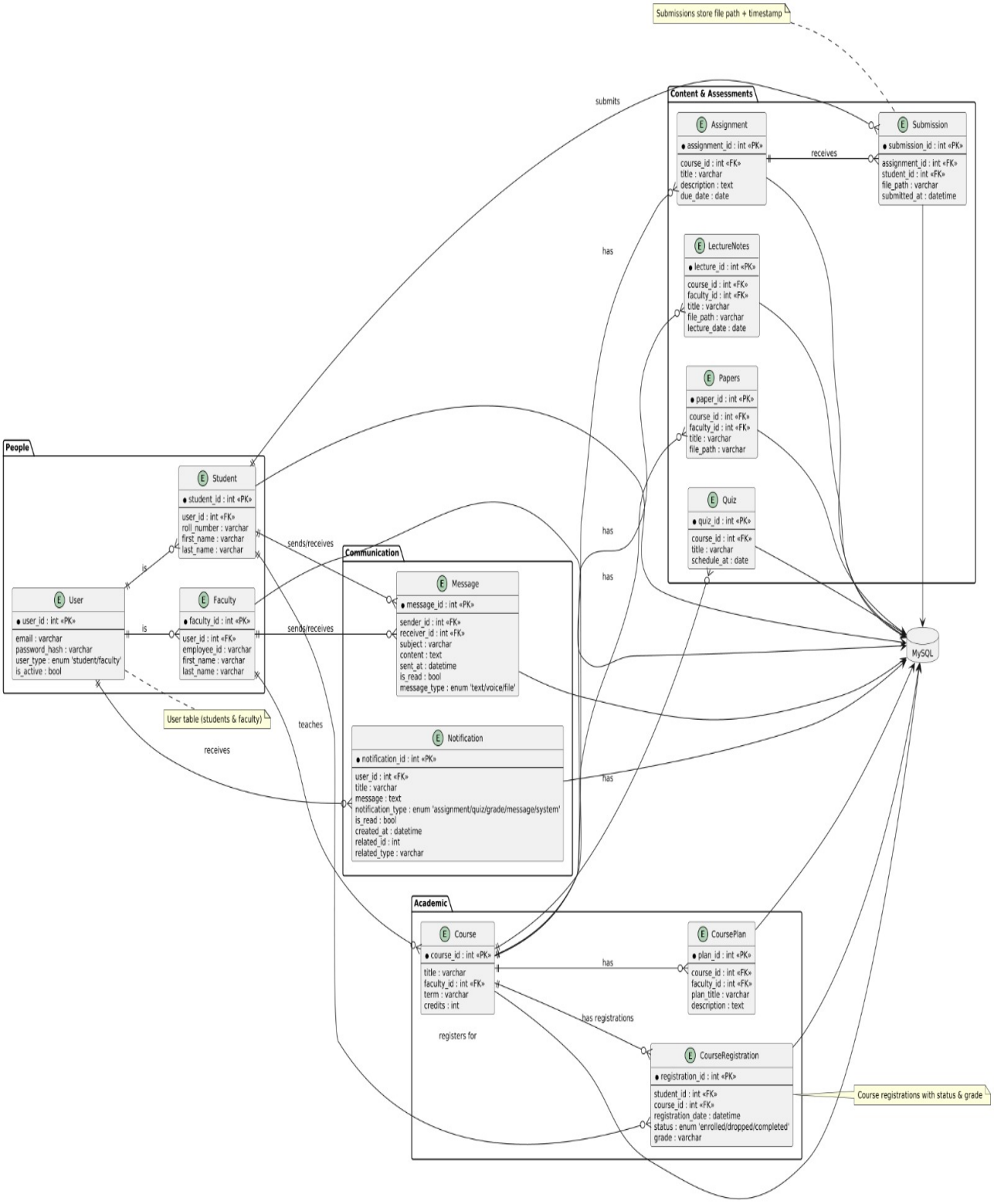

Figure 4.5: Database ER Diagram

4.2 Design Limitations

- **Offline Operation:** Speech recognition independent of cloud.
- **Cross-platform:** Linux, macOS, Windows.
- **Scalability:** Modular design for future extension.
- **Security:** Audit logging, role-based access, encryption.
- **Performance:** 99.9% uptime, <1s key action time.

4.3 Design Approach

Our system was developed in agile cycles. Every sprint entails analysis of requirements, designing, writing, testing and receiving feedback on the designs. The following are the main modular principles that we adhere to:

- Separation of Concerns: Database, voice server, frontend, backend.
- Continuous Integration: CI/CD pipelines.
- Documentation: User manuals, API docs, inline code comments.

4.4 High-Level Design

4.4.0.1 Sequence Diagram

The student clicks on the mic saying, submit my assignment. The frontend of Angular captures the audio and transmits it to Vosk server. Vosk writes the text and sends it to Flask backend. The command is processed by the backend and it invokes the logic of uploading the assignment. The frontend gets confirmed and an audio reassurance is used.

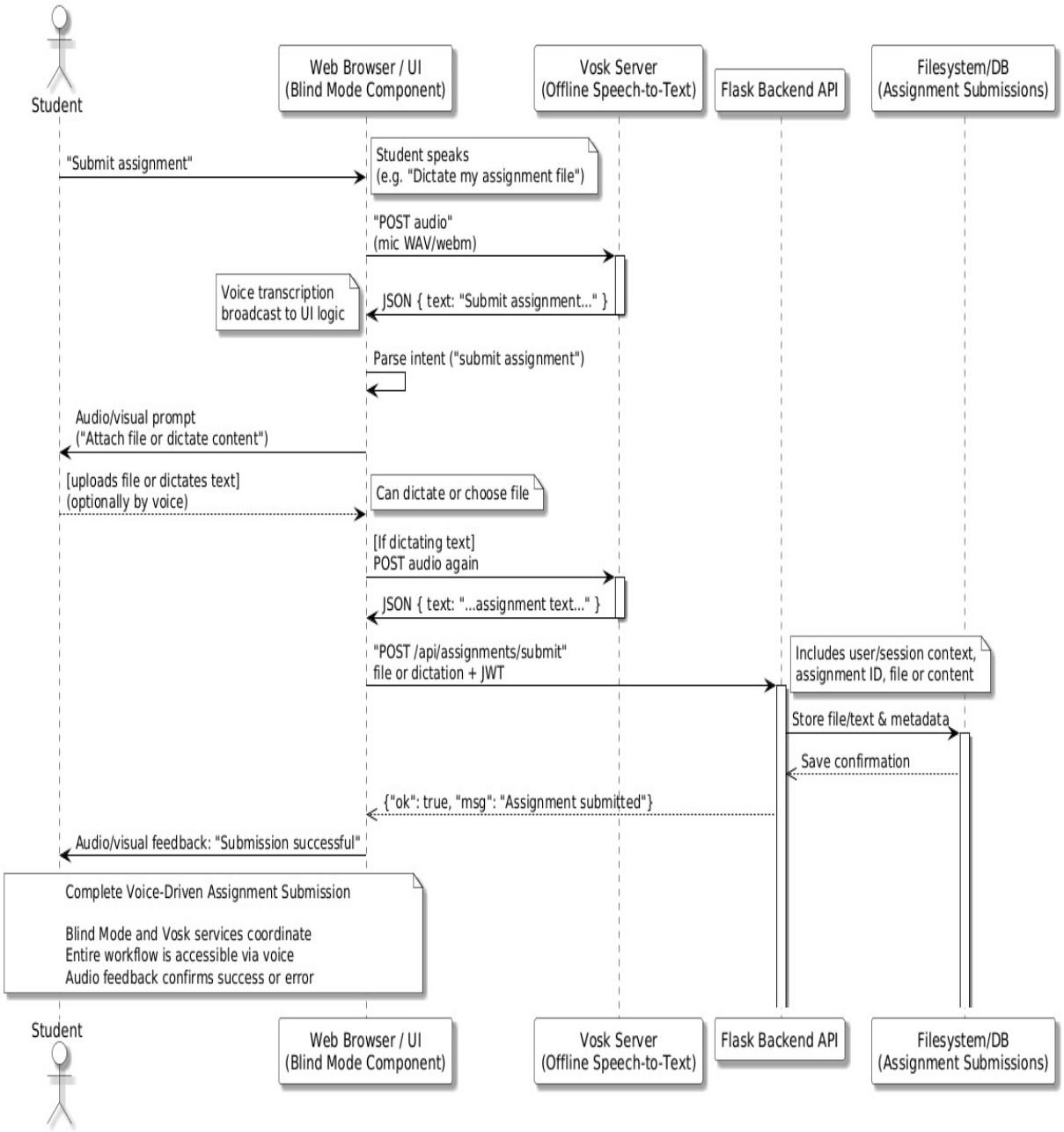

Figure 4.6: sequence Diagram

4.4.0.2 Activity Diagram

Faculty enables Blind Mode. Mark attendance of course X. The system informs the attendance to the database. There is also an audio confirmation.

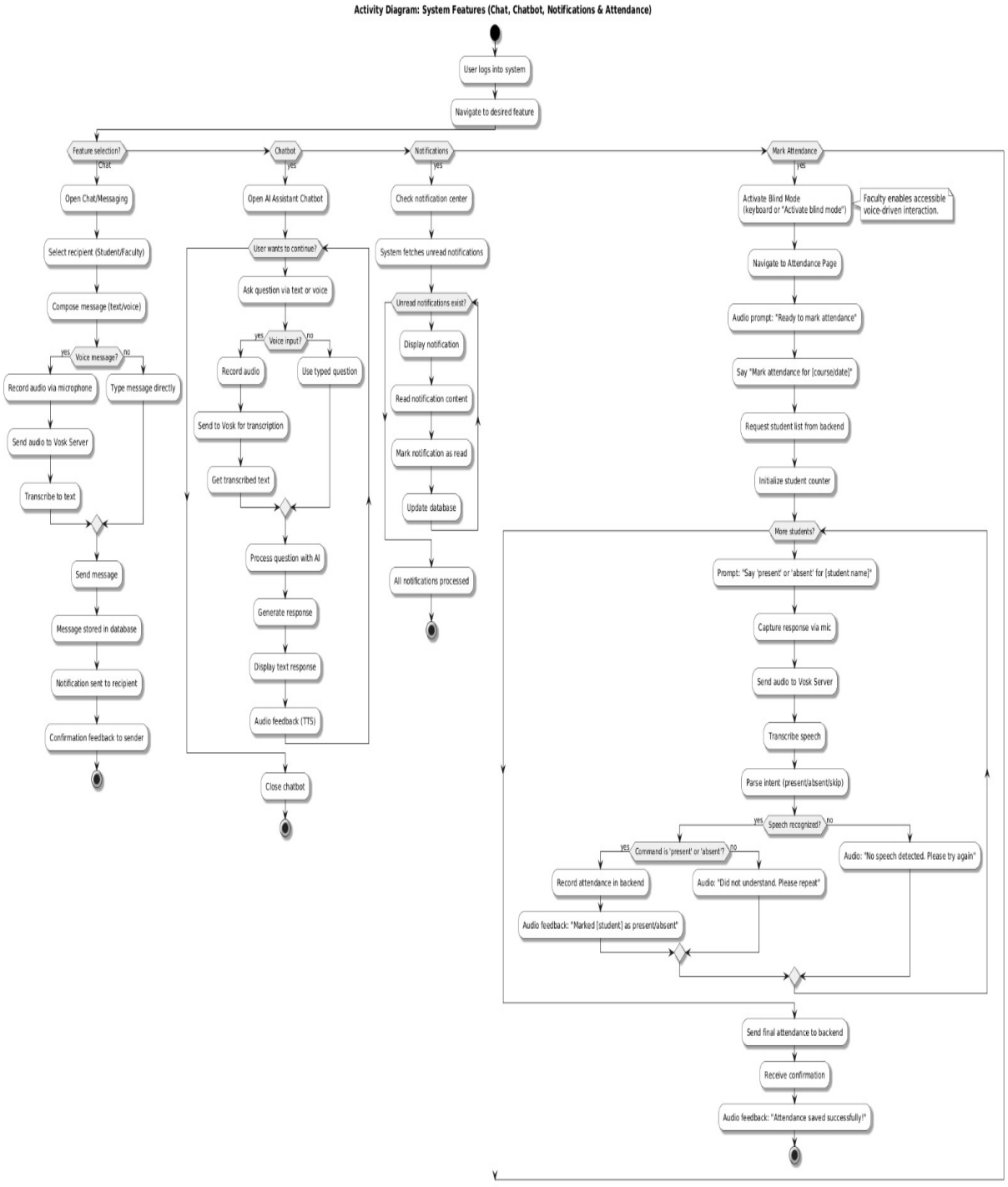

Figure 4.7: Activity Diagram

4.5 Low-Level Architecture

4.5.0.1 Component Design

The interfaces of each of the modules are clear:

Table 4.6: Component Interfaces

Component	Interface Type	Key Methods/Endpoints	Description
Angular Frontend	REST API	/login, /courses, /assignments	UI, voice, accessibility
Flask Backend	REST API	/api/courses, /api/attendance	Business logic, data management
Vosk Server	WebSocket	/transcribe	Speech-to-text, command parsing
MySQL Database	SQL	SELECT, INSERT, UPDATE, DELETE	Persistent storage

4.6 Database Design

The database is 3NF-normalized with foreign key constraints and indexes to increase the performance. Migration has been done using alembic and SQLAlchemy.

Table 4.7: Database Design

Table Name	Key Columns	Relationships
users	id, username, role	courses, assignments
courses	id, name, faculty_id	assignments, students
assignments	id, course_id, due_date	submissions
attendance	id, student_id, course_id	users, courses
quizzes	id, course_id, questions	Results

4.7 GUI Design

Usability, accessibility, and clarity:

- Uniform Design
- Instant Feedback: visual/audio indicators
- Accessibility: keyboard shortcuts, screen reader support, ARIA, blind mode

Table 4.8: Usability Principles

Usability Principle	Implementation Example	Accessibility Support
Feedback	Audio announcements	Blind Mode, ARIA
Accessibility	Blind Mode, screen reader	Keyboard navigation
Error Handling	Voice prompts for invalid input	Confirmation dialogs

4.8 Outside Interfaces

- **RESTful API:** is compatible with third-party applications such as LMS and reporting systems.
- **WebSocket:** real-time voice transcription and notifications.

Table 4.9: Outside Interfaces

Interface Type	Endpoint Example	Description
REST API	/api/courses	Course management
WebSocket	/transcribe	Voice command processing
File Upload	/api/assignments/upload	Assignment submission

4.9 Complete Design and Working

4.9.1 Splash Screen

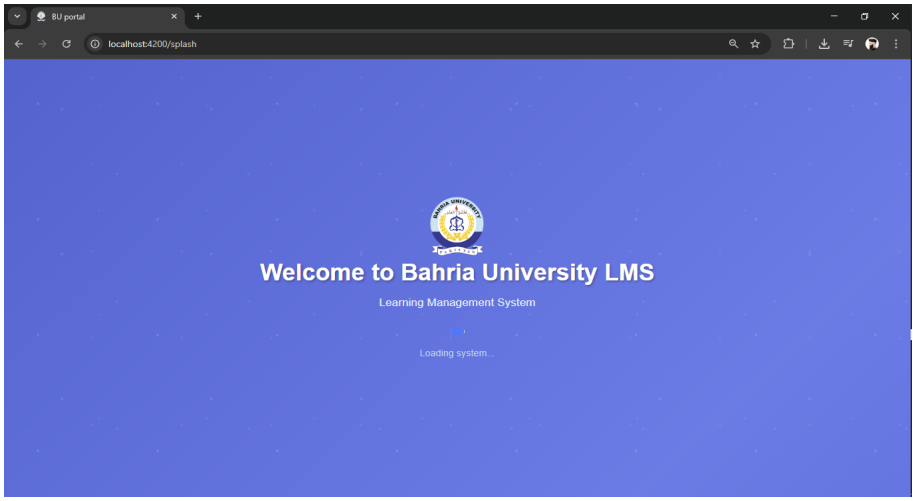

Figure 4.8: Splash Screen

Working: This is the starting splash screen or loading screen, it comes in the start and automatically goes away after 3 seconds.

4.9.2 Blind Mode Toggle

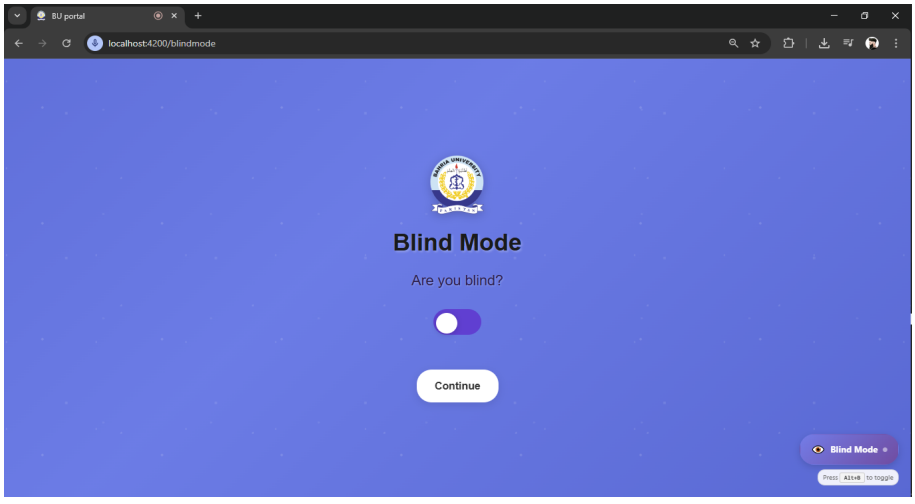

Figure 4.9: Blind Mode Toggle Screen

Working: After splash screen, this is the second screen, it asks user to start voice command or just continue without it, user can use voice command such as ‘yes’ to enable blind mode or click on continue to start without blind mode.

4.9.3 Role Selection

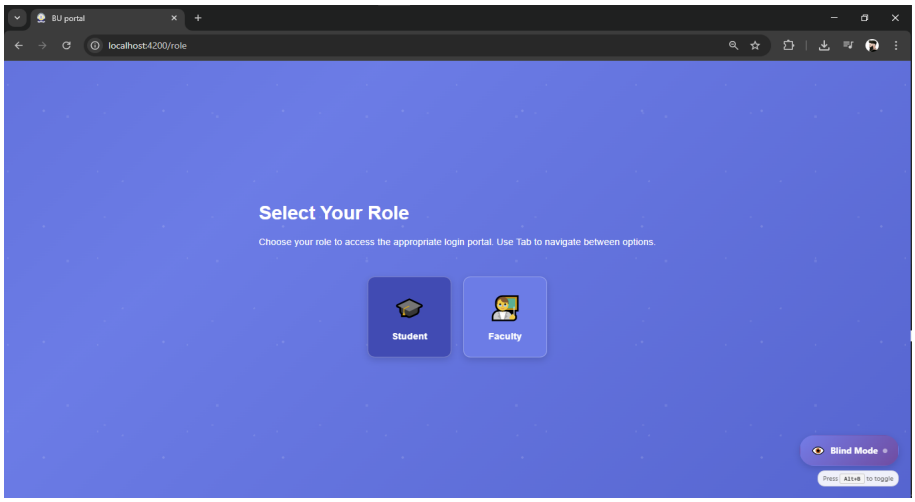

Figure 4.10: Role Selection

Working: After blind mode, next page is for role selection, here user can select from which profile they want to login to, i-e faculty or student. User can also use voice command here such as ‘student’ or ‘faculty’ to select using blind mode.

4.9.4 Student Login

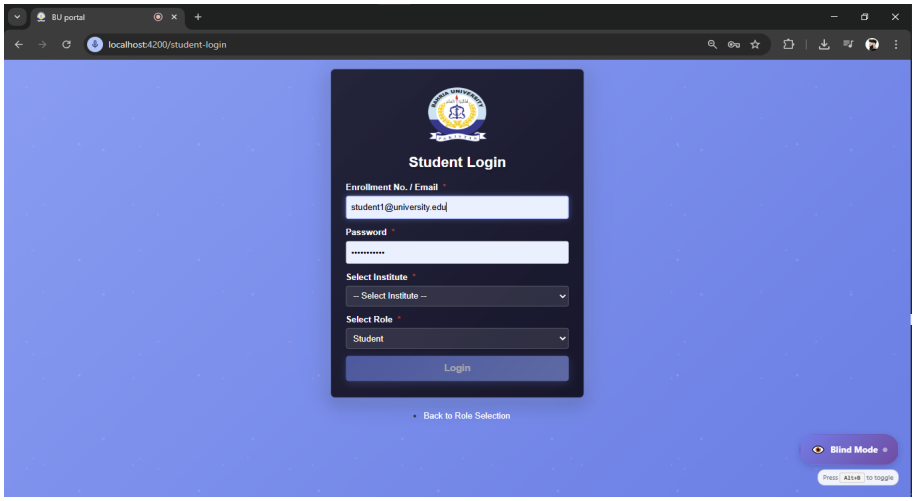

Figure 4.11: Student Login

Working: Student login page: From here students can login using their credentials such as enrolment/email, password and campus. User can use voice commands to fill their credentials such as ‘Enter Enrolment’ for enrolment, ‘Enter Password’ to enter password, ‘Select institute’ to select campus and ‘login/submit’ to login.

4.9.5 Student Dashboard

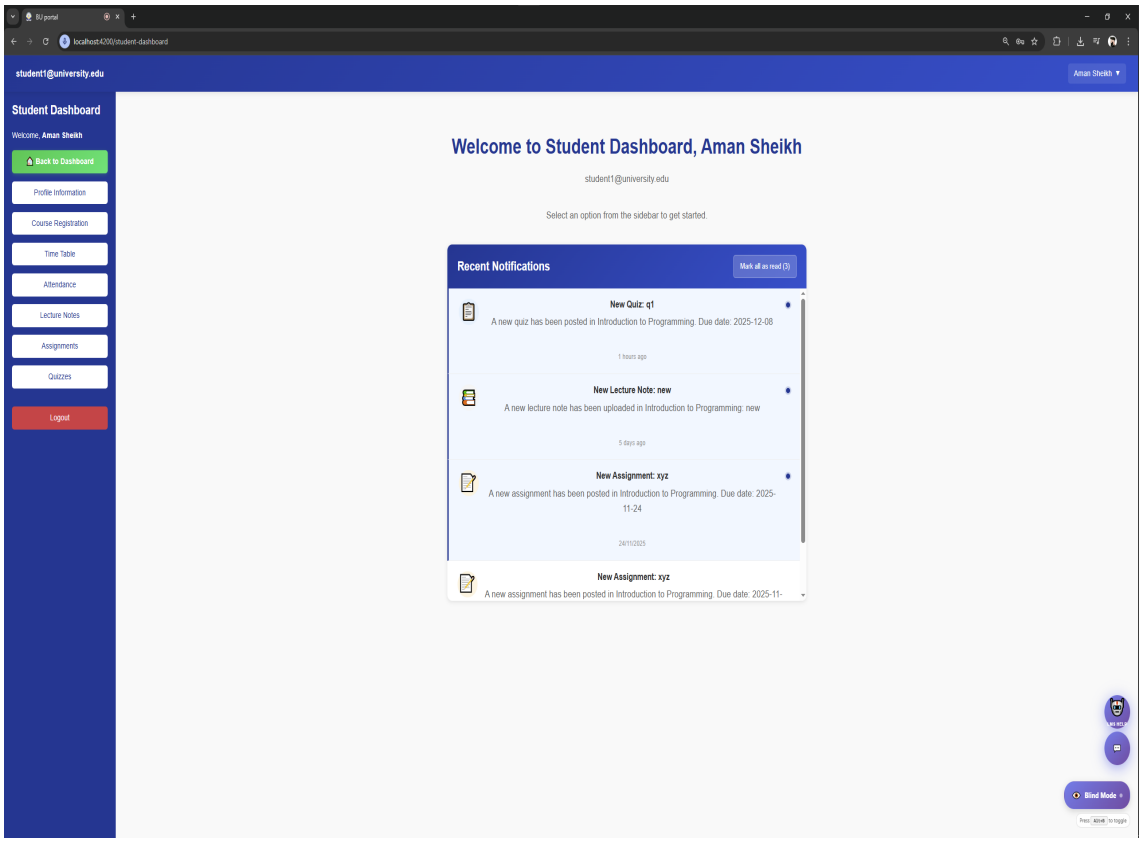

Figure 4.12: Student Dashboard

Working: After successful login, user will be redirected to student dashboard. There they can view all new notifications, notification generated after any operation performed by teachers such as marking attendance, assignment uploaded, quiz uploaded, lecture uploaded. After that, on the right below corner, above blind mode button student gets a chat button which they can use to chat with their faculty member. Above chat button, students get a chatbot button which they can use to ask question related to working of lms. On the left side, user gets a navigation panel through which they can navigate between different pages. On the above header, on left side the email of student is displayed and on the left side, name of the student is displayed. The right-side name button is a drop down which gives options such as reset your password or logout. The blind mode will read unread notifications automatically when student reach dashboard also students can use the voice command ‘read all notifications’ to listen to all available notifications, ‘mark all as read’ to mark all unread notifications as read.

4.9.6 Student Chat

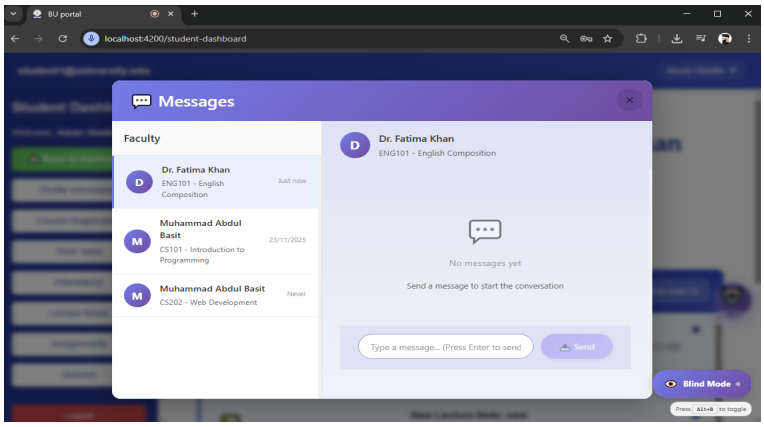

Figure 4.13: Student Chat

Working: Chat dialogue: Using chat dialogue, students can easily chat with their faculty members over the app. They can also chat using voice commands such as ‘open chat’ to open chat box, ‘faculty name i-e Muhammad Abdul Basit’ to open specific chat of faculty, ‘type’ to type their message, ‘send’ to send the typed message, ‘close’ to close the chat dialogue.

4.9.7 Student Profile

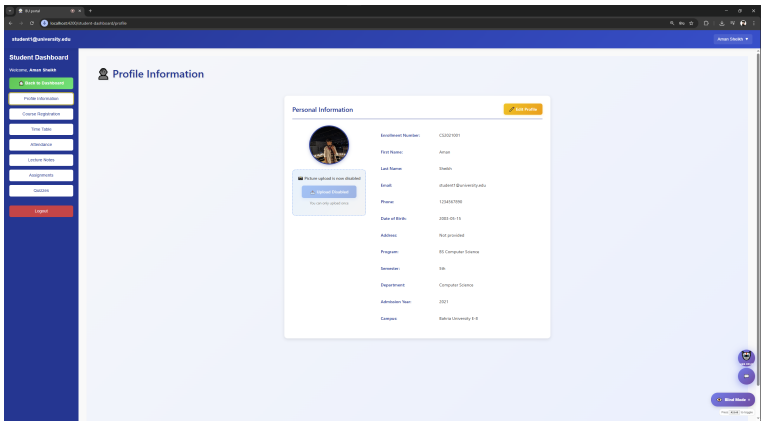

Figure 4.14: Student Profile Information

Working: Student profile information: Shows complete information of a student, student can edit editable options of their profile information also they can upload their profile picture (allowed only once). Student can also use voice commands such as ‘profile information’ to navigate into profile information tab where the blind mode reads each tab visible in profile information, ‘edit’ to enter edit mode and edit the editable fields, ‘upload picture’ to upload their picture.

4.9.8 Student Course Registration

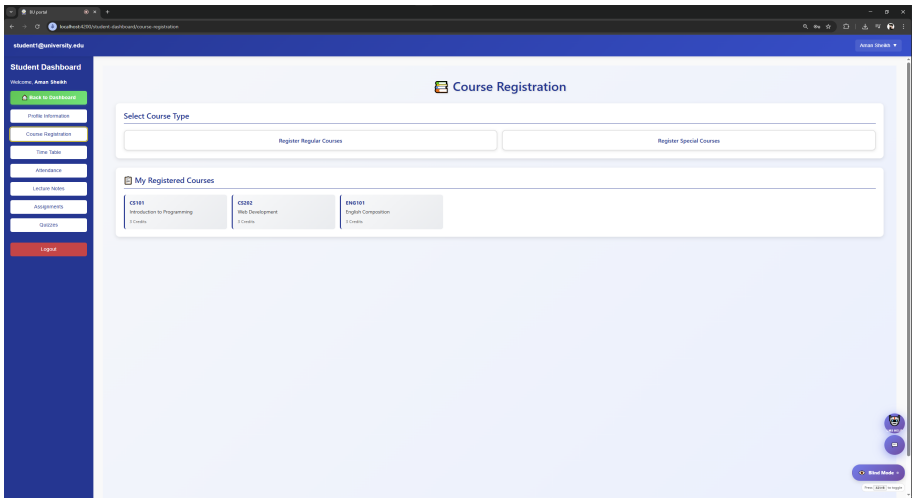

Figure 4.15: Student Course Registration

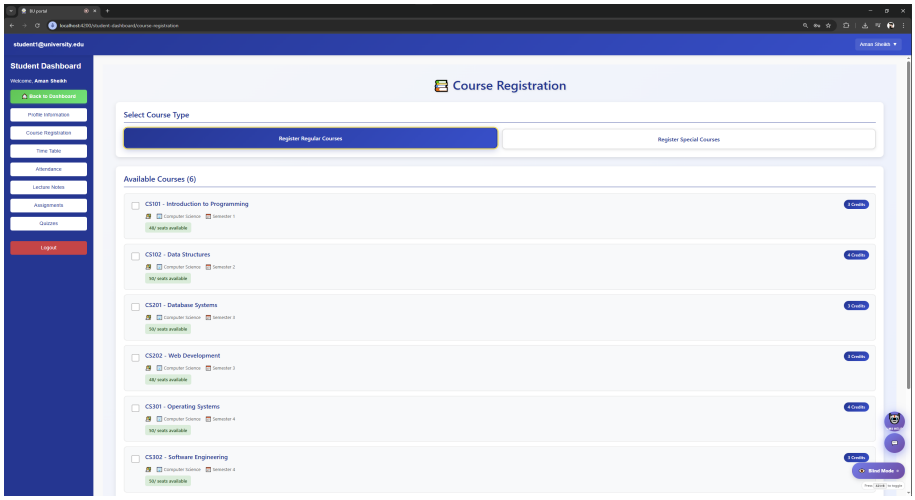

Figure 4.16: Student Course Registration

Working: Course registration: Students can register their courses from course registration tab, they can register regular courses or special courses with just cone simple click, below they can also view their registered courses. Students can use voice commands such as ‘course registration’ to navigate to registration page, after entering the registration page, the blind mode reads already registered courses first then ask for commands, ‘register regular courses’ to view and register regular courses, ‘register special courses’ to view and register special courses.

4.9.9 Student Time-Table

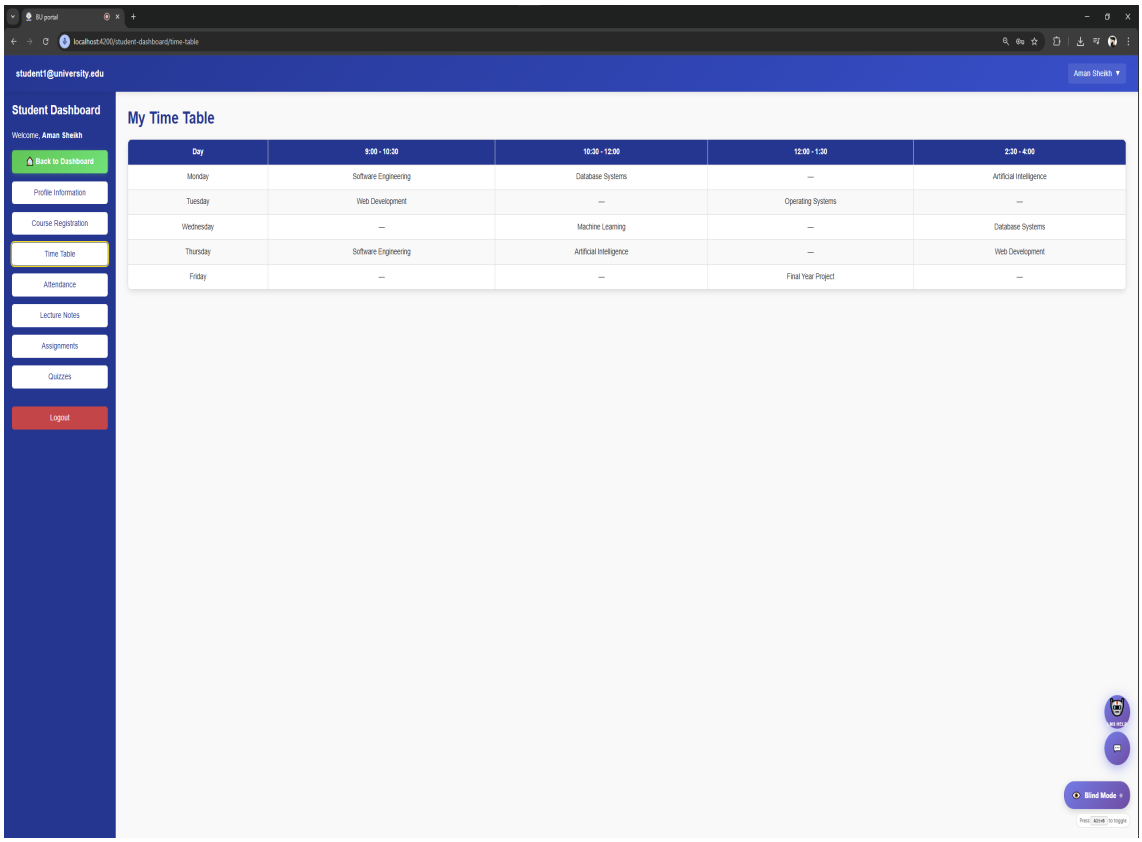

Figure 4.17: Student Time-Table

Working: Timetable: In timetable page, students can view their timetable. While using voice commands, ‘Timetable’ students can navigate to timetable page, and the blind mode will read whole timetable such that day, time and classes.

4.9.10 Student Attendance

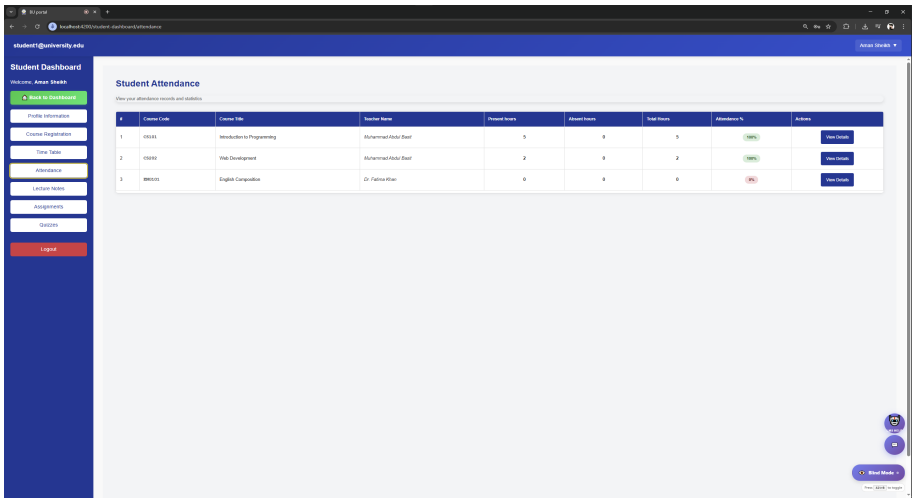

Figure 4.18: Student Attendance

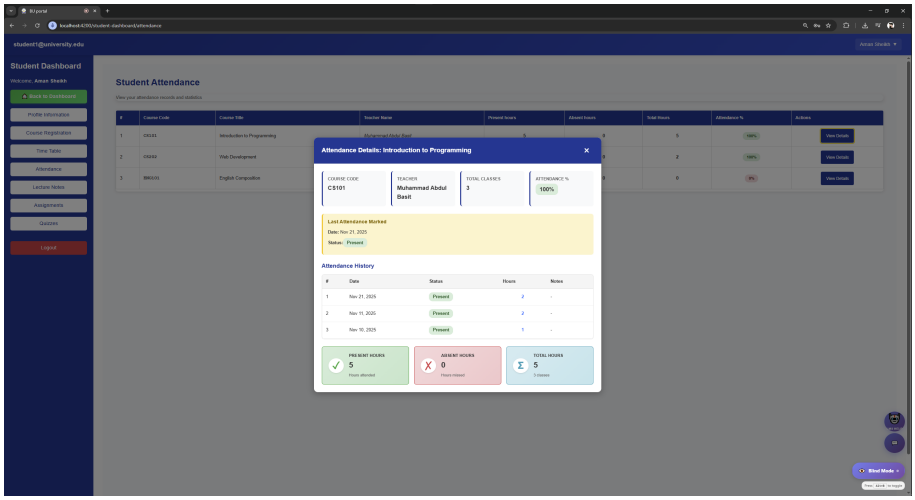

Figure 4.19: Student Attendance

Working: Student attendance: Students can view record of their attendance, also they can view details of every class via detail button. Students can also use voice commands such as ‘Attendance’ to navigate to attendance page where blind mode will read attendance for each course, ‘view details’ to view details of a specific course attendance.

4.9.11 Student Lecture Notes

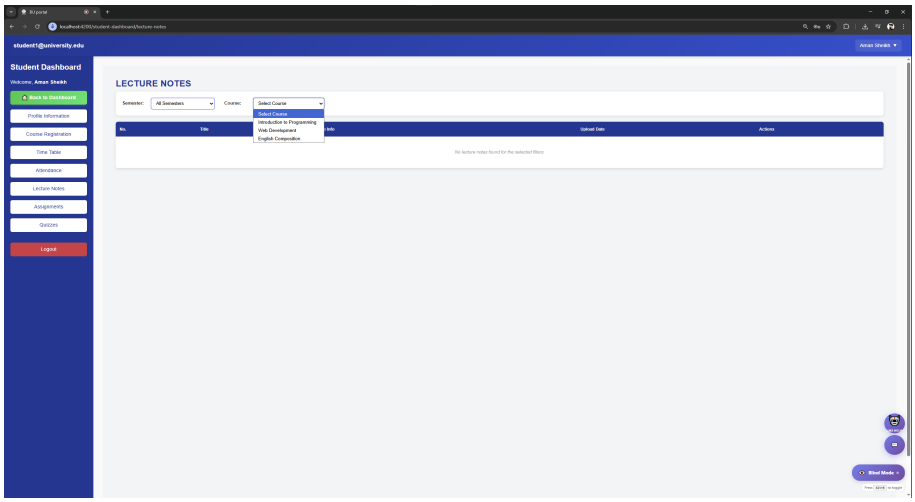

Figure 4.20: Student Lecture Notes

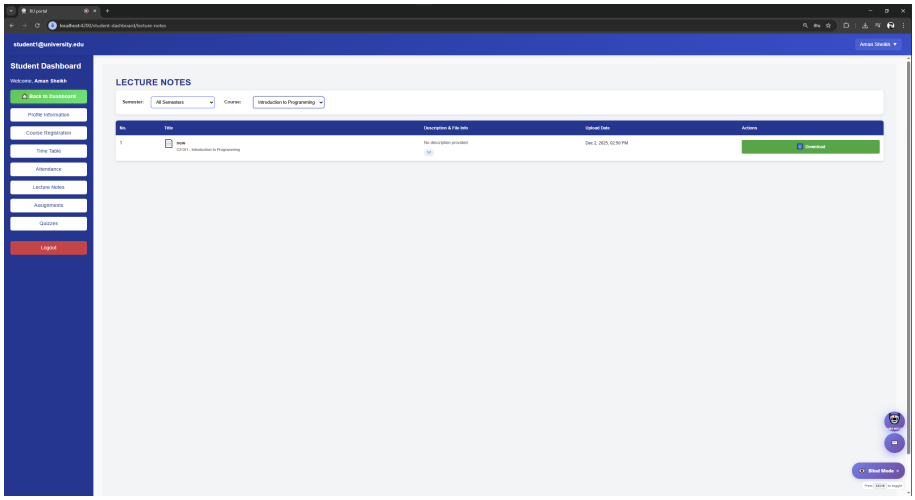

Figure 4.21: Student Lecture Notes

Working: Lecture notes: In lecture notes page, students can view and download lectures uploaded by faculty for specific subject using course drop down. Students can also use voice commands such as ‘lecture notes’ to navigate to lecture notes page, ‘select course/semester’ to select a specific course or specific semester, after selecting the blind mode will read all available lectures, ‘download’ to download a specific lecture.

4.9.12 Student Assignments

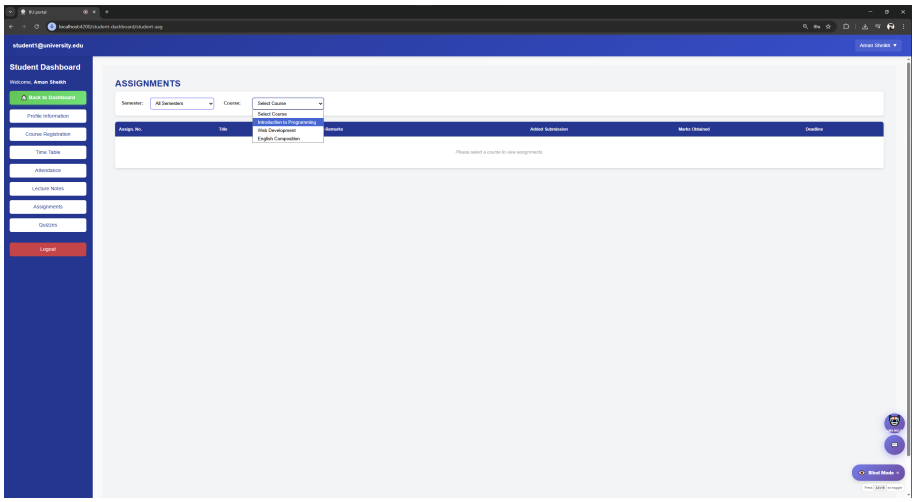

Figure 4.22: Student Assignments

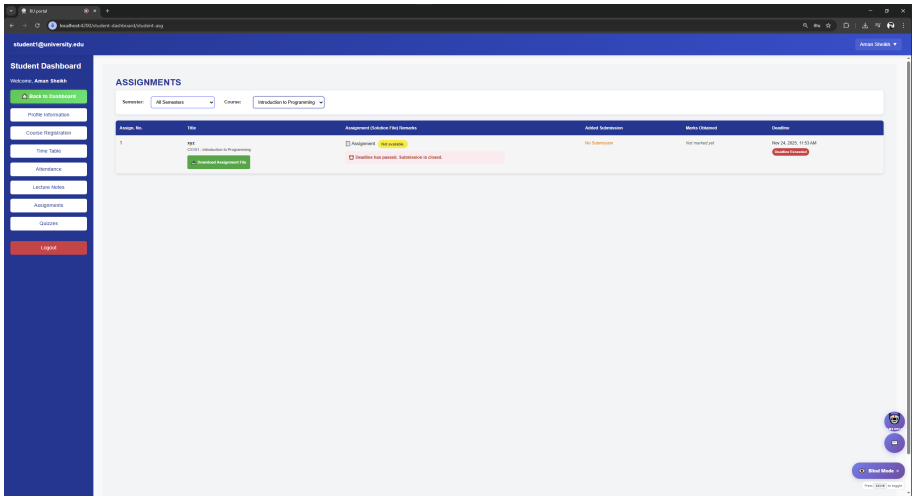

Figure 4.23: Student Assignments

Working: Assignments: In assignments page, students can view, download and submit assignments against assignments uploaded by faculty for specific subject using course drop down. Students can also use voice commands such as ‘assignments’ to navigate to assignments page, ‘select course/semester’ to select a specific course or specific semester, after selecting the blind mode will read all available assignments details such as name, deadline , ‘download’ to download a specific quiz, ‘upload file’ to upload a file, ‘submit’ to submit the quiz, ‘delete’ to delete submission.

4.9.13 Student Quizzes

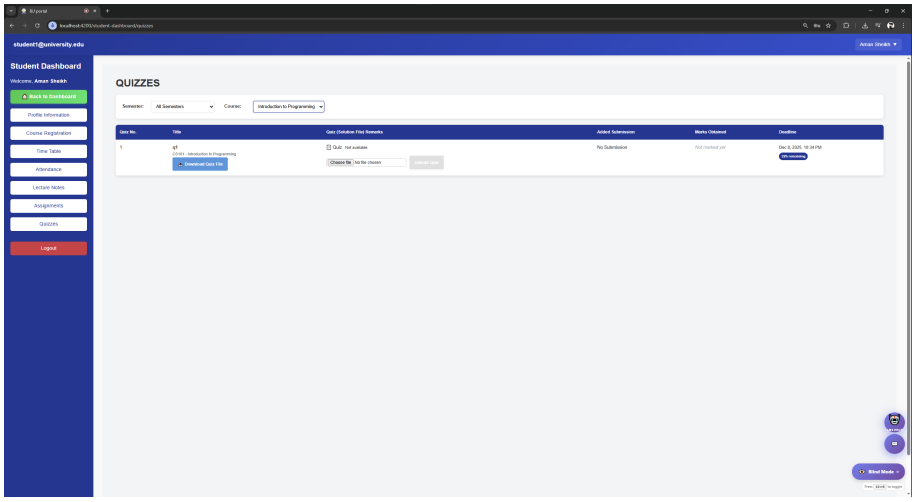

Figure 4.24: Student Quizzes

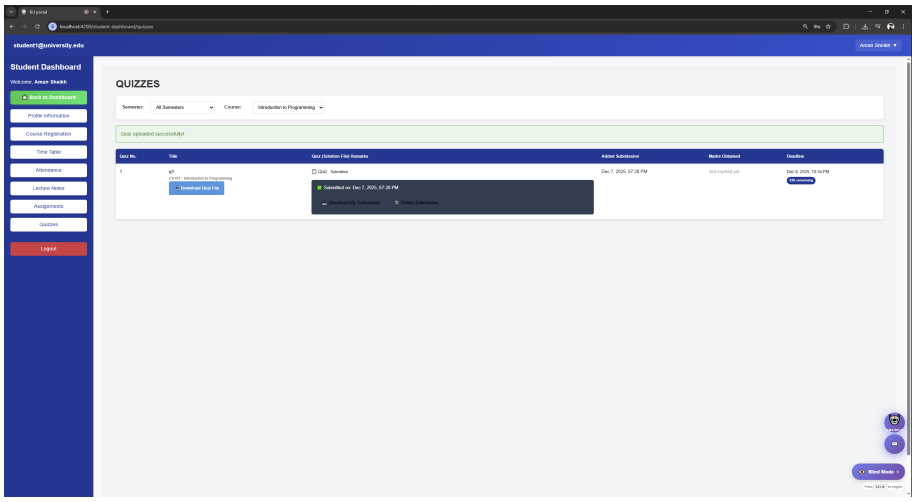

Figure 4.25: Student Quizzes

Working: Quizzes: In quizzes page, students can view, download and submit quizzes against quizzes uploaded by faculty for specific subject using course drop down. Students can also use voice commands such as ‘quizzes’ to navigate to quizzes page, ‘select course/semester’ to select a specific course or specific semester, after selecting the blind mode will read all available quizzes details such as name, deadline , ‘download’ to download a specific quiz, ‘upload file’ to upload a file, ‘submit’ to submit the quiz, ‘delete’ to delete submission.

4.9.14 ChatBot

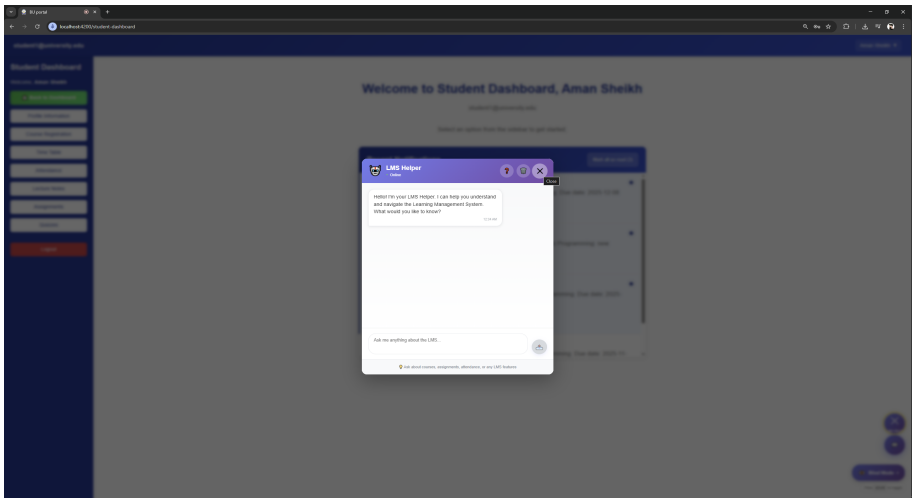

Figure 4.26: Chatbot

Working: Chat bot dialogue: students can ask the chat bot about their queries related to LMS such as how they can register course etc etc. Student can use chat bot also by using voice commands such as ‘LMS helper’ to open chat bot dialogue, ‘type’ to type their message, ‘send’ to send their message, the chat bot will read the reply out loud, ‘close’ to close the chat bot dialogue.
Teachers can ask the chat bot about their queries related to lms such that how they can mark attendance etc etc. Teachers can use chat bot also by using voice commands such as ‘lms helper’ to open chat bot dialogue, ‘type’ to type their message, ‘send’ to send their message, the chat bot will read the reply out loud, ‘close’ to close the chat bot dialogue.

4.9.15 Faculty Login

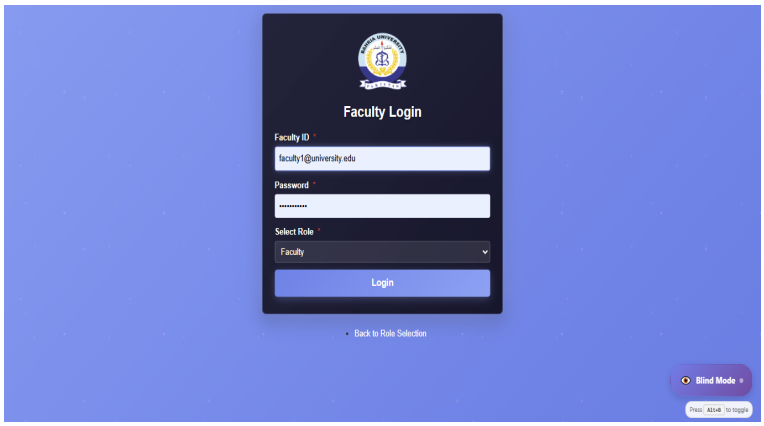

Figure 4.27: Faculty Login

Working: Faculty login page: From here faculty can login using their credentials such as email and password. User can use voice commands to fill their credentials such as ‘Enter faculty id’ to enter faculty id/email, ‘Enter Password’ to enter password and ‘login/submit’ to login.

4.9.16 Faculty Chat

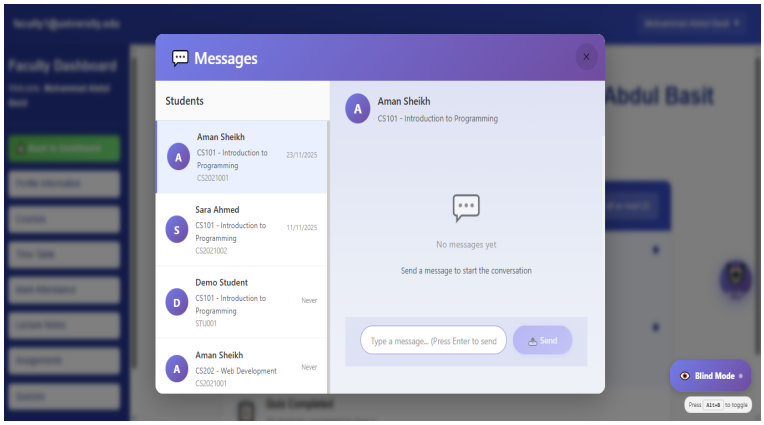

Figure 4.28: Faculty Chat

Working: Chat dialogue: Using chat dialogue, teachers can easily chat with their students over the website. They can also chat using voice commands such as ‘open chat’ to open chat box, ‘student name i-e Aman Sheikh’ to open specific chat of a student, ‘type’ to type their message, ‘send’ to send the typed message, ‘close’ to close the chat dialogue.

4.9.17 Faculty Dashboard

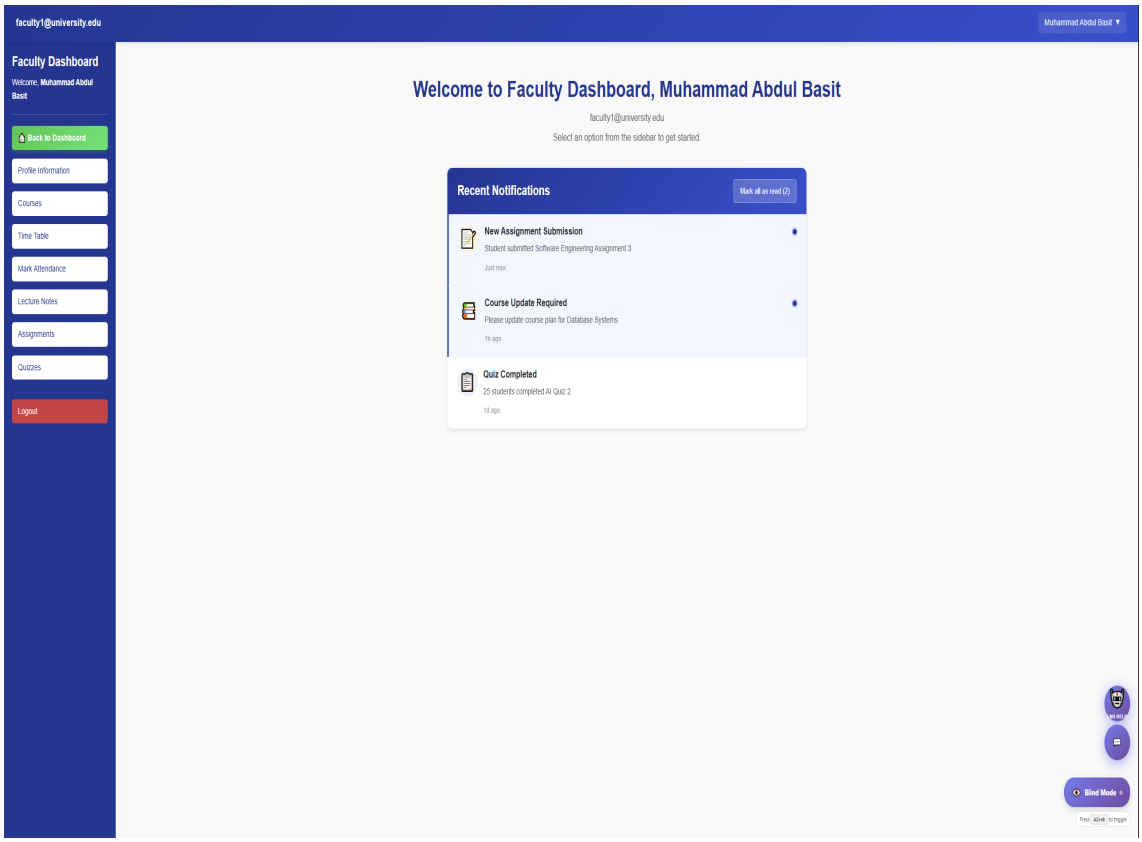

Figure 4.29: Faculty Login

Working: After successful login, user will be redirected to faculty dashboard. There they can view all new notifications, notification generated after any operation performed by students such as submitting assignment, submitting quiz. After that, on the right below corner, above blind mode button faculty gets a chat button which they can use to chat with their students. Above chat button, faculty get a chatbot button which they can use to ask question related to working of lms. On the left side, user gets a navigation panel through which they can navigate between different pages. On the above header, on left side the email of faculty is displayed and on the left side, name of the faculty is displayed. The right-side name button is a drop down which gives options such as reset your password or logout. The blind mode will read unread notifications automatically when faculty reach dashboard also faculty can use the voice command ‘read all notifications’ to listen to all available notifications, ‘mark all as read’ to mark all unread notifications as read.

4.9.18 Faculty Profile

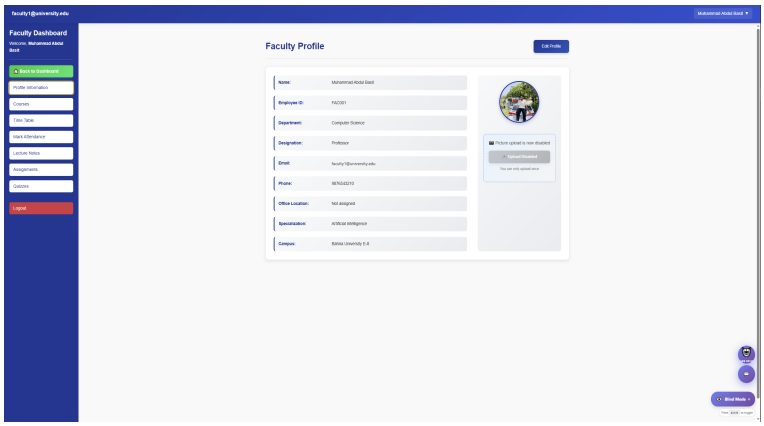

Figure 4.30: Faculty Profile

Working: Faculty profile information: Shows complete information of a faculty member, teacher can edit editable options of their profile information also they can upload their profile picture (allowed only once). Teachers can also use voice commands such as ‘profile information’ to navigate into profile information tab where the blind mode reads each tab visible in profile information, ‘edit’ to enter edit mode and edit the editable fields, ‘upload picture’ to upload their picture.

4.9.19 Faculty Courses

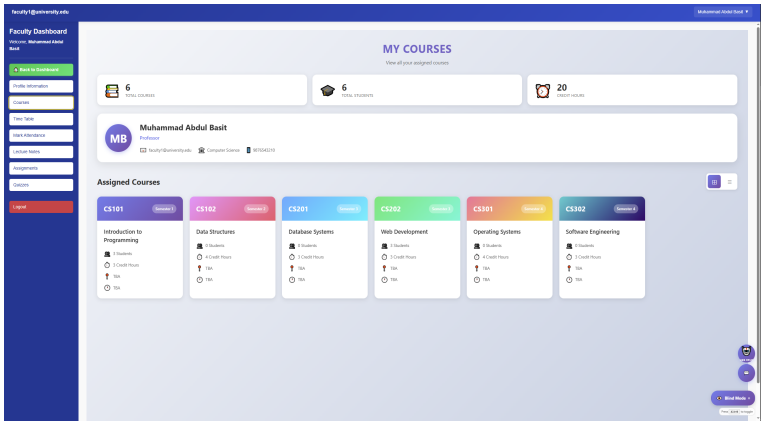

Figure 4.31: Faculty Courses

Working: Faculty courses: Teachers can view their assigned courses in courses page. They can navigate to page using voice command such as ‘courses’ where the blind mode will read all the assigned courses.

4.9.20 Faculty Mark Attendance

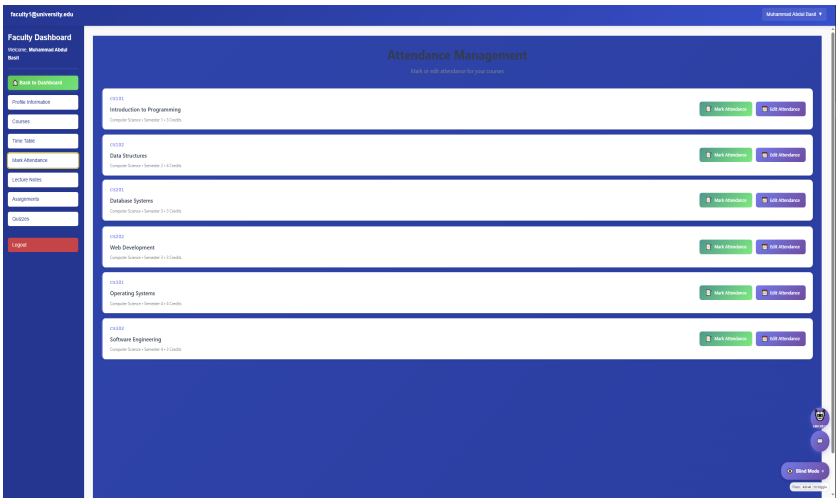

Figure 4.32: Faculty Mark Attendance

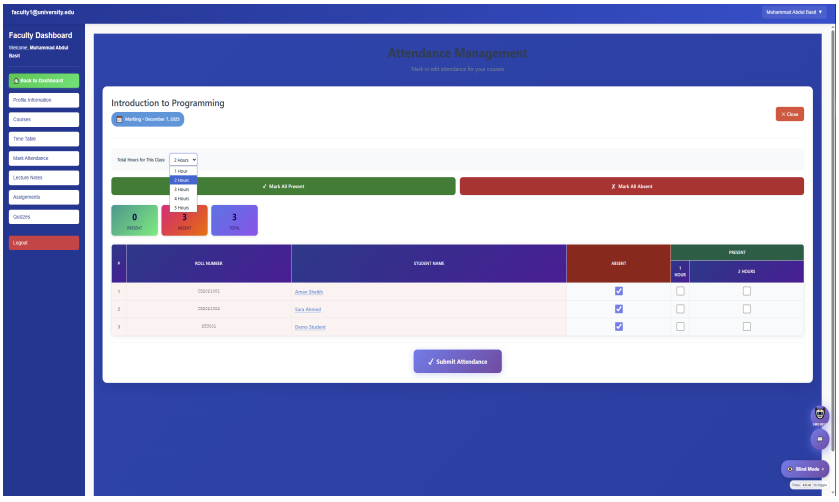

Figure 4.33: Faculty Mark Attendance

Working: Faculty Mark Attendance: The Mark Attendance module enables instructors to record daily attendance for students enrolled in their courses. The interface displays the complete list of students for the selected course, allowing teachers to efficiently mark each student as present, absent, or late. The module also supports voice-based navigation through the system's Blind Mode. By saying commands such as "mark attendance," the system automatically reads out all available courses and prompts the user to speak "mark" to begin attendance marking.

4.9.21 Faculty Edit Attendance

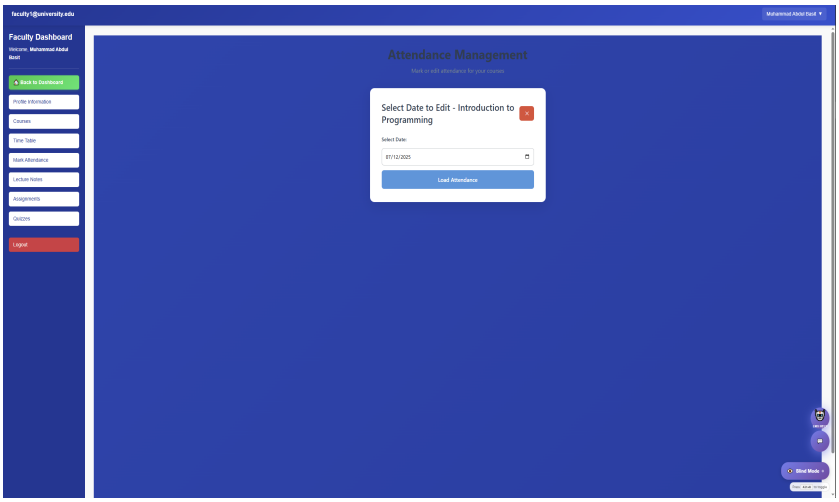

Figure 4.34: Faculty Edit Attendance

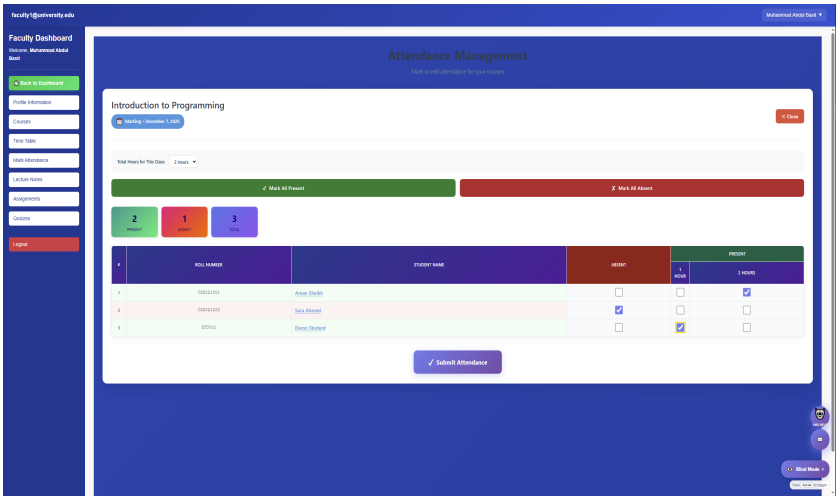

Figure 4.35: Faculty Edit Attendance

Working: Faculty Edit Attendance: The Edit Attendance module allows teachers to modify previously recorded attendance entries. Instructors can select a course and date to review the attendance status of all students and make corrections where necessary. In Blind Mode, the system reads out available attendance records and prompts the teacher with voice commands. Users can speak “edit” to enter the modification workflow and update any student’s attendance using voice or manual input. This ensures that errors in marking can be corrected efficiently.

4.9.22 Faculty Lecture Notes

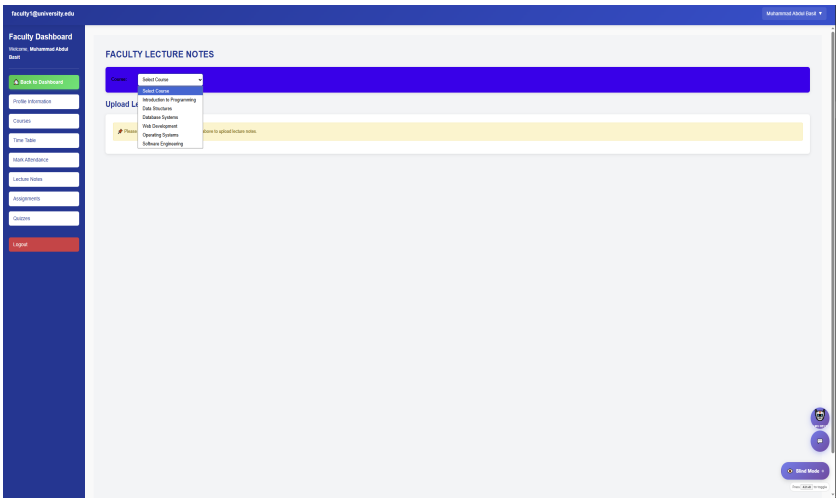

Figure 4.36: Faculty Lecture Notes

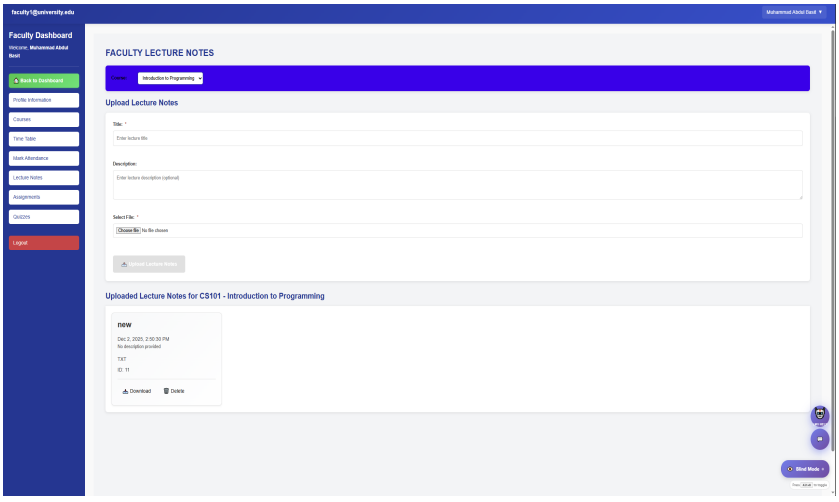

Figure 4.37: Faculty Lecture Notes

Working: Faculty lecture notes: Faculty can add new lectures for students. They can use voice commands such as ‘lecture notes’ to navigate to lecture notes where blind mode first reads already uploaded lecture notes then ask to add new, ‘delete’ to delete already uploaded lecture, ‘download’ to download already uploaded lecture, ’add new’ to add new lecture, ‘enter title’ to enter title, ‘add description’ to add description’, ‘select file’ to choose file to upload, ‘submit’ to upload a new lecture.

4.9.23 Faculty Assignments

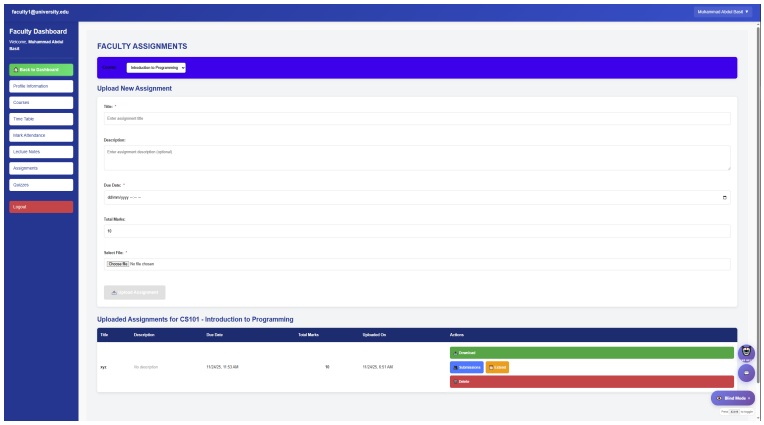

Figure 4.38: Faculty Assignments

Working: Faculty assignments: Faculty can add new lectures for students. They can use voice commands such as ‘assignments’ to navigate to assignments where blind mode first reads already uploaded assignments then ask to add new, ‘delete’ to delete already uploaded assignment, ‘download’ to download already uploaded assignment, ’extend’ to extend the deadline for already uploaded assignment,’ ‘submission’ to view submissions for already uploaded assignment, ’add new’ to add new assignment, ‘enter title’ to enter title, ‘add description’ to add description’, ‘add deadline’ to add deadline, ‘select file’ to choose file to upload, ‘submit’ to upload a new lecture.

4.9.24 Faculty Quizzes

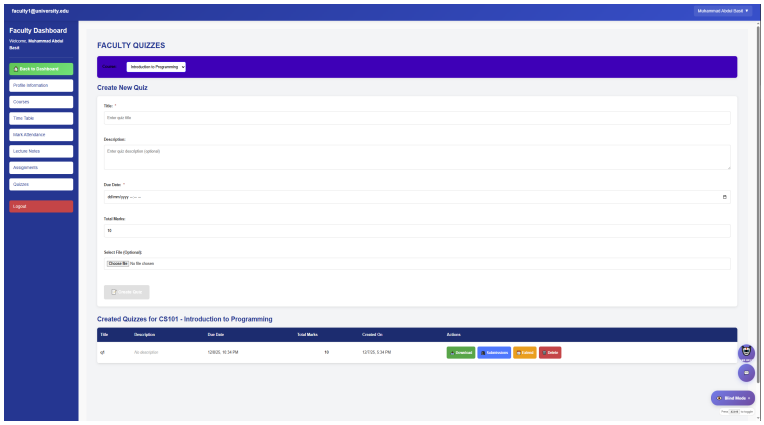

Figure 4.39: Faculty Quizzes

Working: Faculty quizzes: Faculty can add new lectures for students. They can use voice commands such as ‘quizzes’ to navigate to quizzes where blind mode first reads already uploaded quizzes then ask to add new, ‘delete’ to delete already uploaded assignment,

‘download’ to download already uploaded assignment, ’extend’ to extend the deadline for already uploaded assignment,’ ‘submission’ to view submissions for already uploaded assignment, ’add new’ to add new assignment, ‘enter title’ to enter title, ‘add description’ to add description’, ‘add deadline’ to add deadline, ‘select file’ to choose file to upload, ‘submit’ to upload a new lecture.

4.9.25 Faculty Time-Table

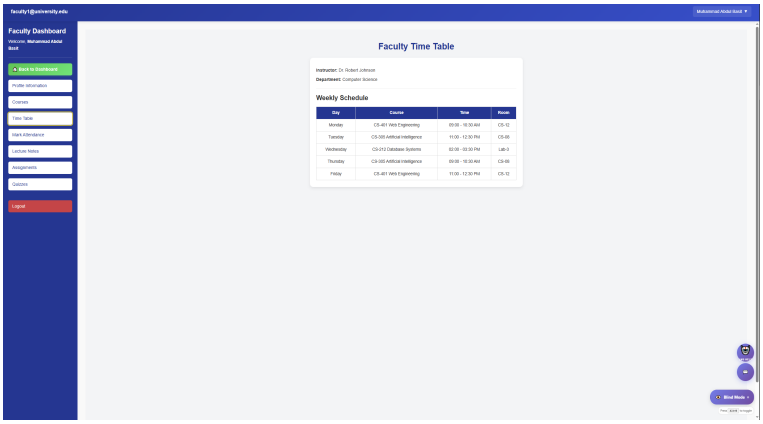

Figure 4.40: Faculty Time-Table

Working: Faculty Timetable: In timetable page, teachers can view their timetable. While using voice commands, ‘Timetable’ teachers can navigate to timetable page, and the blind mode will read whole timetable such that day, time and classes.

4.10 Additional Functionalities

- Chat system between students and faculty
- AI Chatbot helper
- Vosk offline voice recognition
- 3-minute auto-logout
- Full-screen profile image viewer
- One time upload option for profile picture on new profile
- Voice-guided profile editing
- Course registration system with prerequisites
- Comprehensive notifications
- Grade management for faculty
- Can only upload a picture once on new profile

Chapter 5

System Implementation

5.1 IDE: The Concept of Integrated Development Environment

The Student and Faculty Management System with Voice Accessibility is primarily built using Visual Studio Code (VS Code) due to its high customizability. VS Code supports all required platforms and is pre-configured for Python, TypeScript, and SQL. Essential extensions include Python, Angular Language Service, Prettier, ESLint, and Docker. The integrated terminal, Git source control, and live linting enable fast code writing, debugging, and testing.

Table 5.1: IDE Concept

IDE Feature	Usage Example	Benefit
Integrated Terminal	Running backend and frontend servers	Streamlined workflow
Debugger	Step-through Flask API logic	Faster bug resolution
Extensions	Accessibility testing, code linting	Improved code quality
Source Control	Git integration	Version management

5.2 Tools and Technologies

5.2.1 Development Stack

In our project, we have a contemporary, modular stack, which is convenient with the course assignments.

Table 5.2: Development Stack

Layer	Technology	Purpose
Frontend	Angular 16	UI, voice command handlers
Backend	Flask (Python 3.13)	REST API, business logic
Database	MySQL 8	Persistent storage
Voice	Vosk 0.15 (Python)	Offline speech recognition
Auth	JWT	Secure authentication
Container	Docker	Deployment and environment management

5.2.2 Extra Resources

- **Git:** Git helps us to manage our sources and work with other team members.
- **Docker Compose:** Manages the frontend, voice server and backend together in various containers.
- **Postman:** Assists with taking the note of and testing APIs.
- **PlantUML:** We are developing the architecture and sequence diagrams to visualize the system.

5.3 System Architecture Implementation

5.3.1 Implementation on the Backend

The backend is structured as a RESTful API, with separate modules for authentication, courses, assignments, attendance, and quizzes. Marshmallow handles serialization and validation, while SQLAlchemy ORM manages database interactions. Authentication is role-based with JWT tokens.

Key endpoints include:

- Auth: `/api/auth/register`, `/api/auth/login`, `/api/auth/refresh`
- Courses: `/api/courses`, `/api/courses/{id}`
- Assignments: `/api/assignments`, `/api/assignments/upload`
- Attendance: `/api/attendance`, `/api/attendance/mark`

- Quizzes: `/api/quizzes`, `/api/quizzes/score`
- Voice Command: `/api/voice/command` (Vosk integration)

Table 5.3: Implementation on Backend

Endpoint	Method	Description	Auth Required
/api/login	POST	User authentication	No
/api/courses	GET	List available courses	Yes
/api/assignments	POST	Submit assignment	Yes

5.3.2 Front-end Execution

The Angular frontend includes authentication, dashboards, courses, assignments, attendance, quizzes, and accessibility modules. Angular services and RxJS manage state, while WebSocket and HTTPClient enable real-time communication.

- Role-based dashboards: Admins, professors, and students see different views.
- Blind Mode: Provides voice navigation and sounds for accessibility.
- Voice Command Integration: Vosk server captures microphone input and routes output to UI.
- Responsive design: Works across all devices.
- Accessibility features: Keyboard navigation, screen reader support, ARIA labeling.

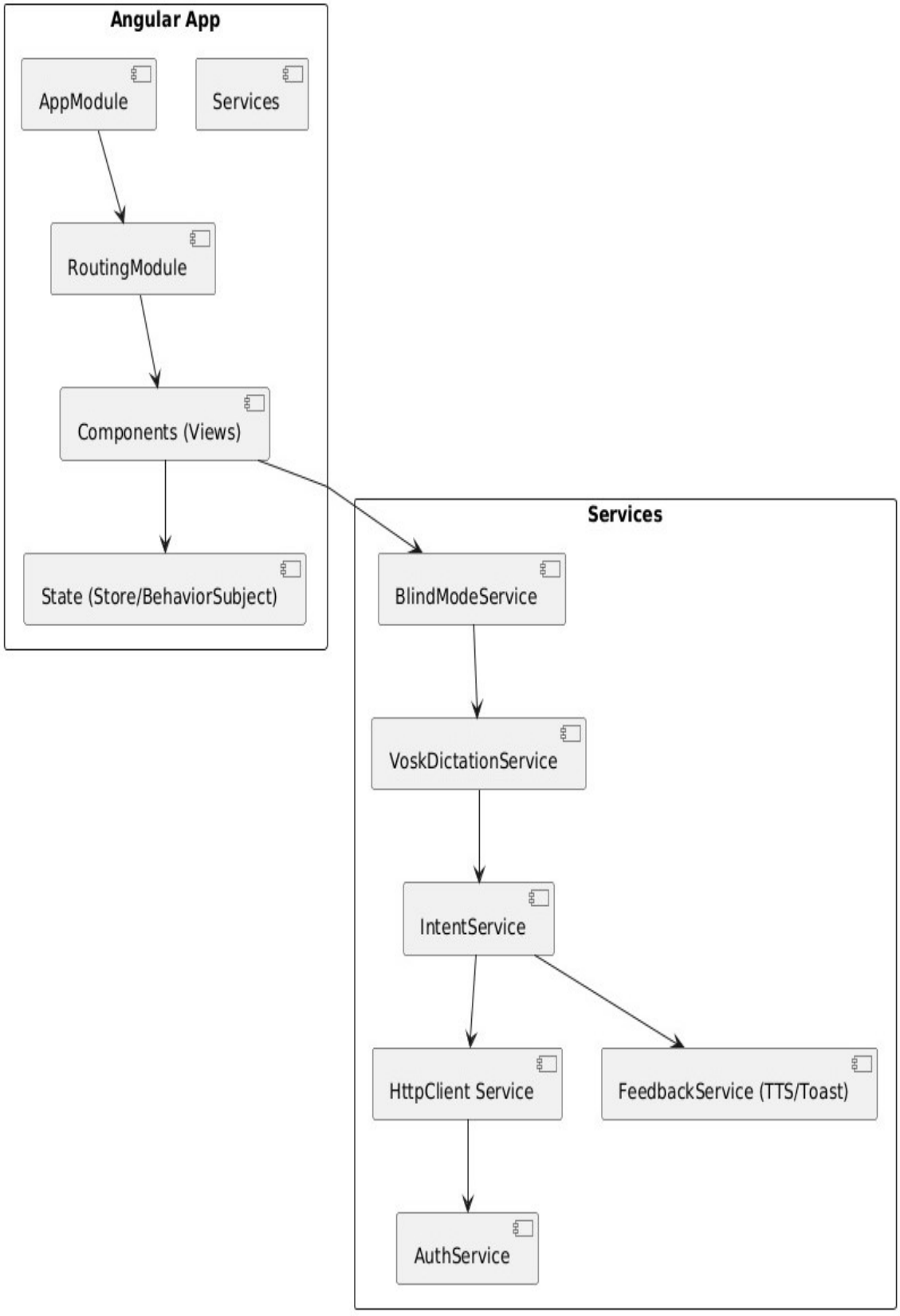

Figure 5.1: Front-end Implementation

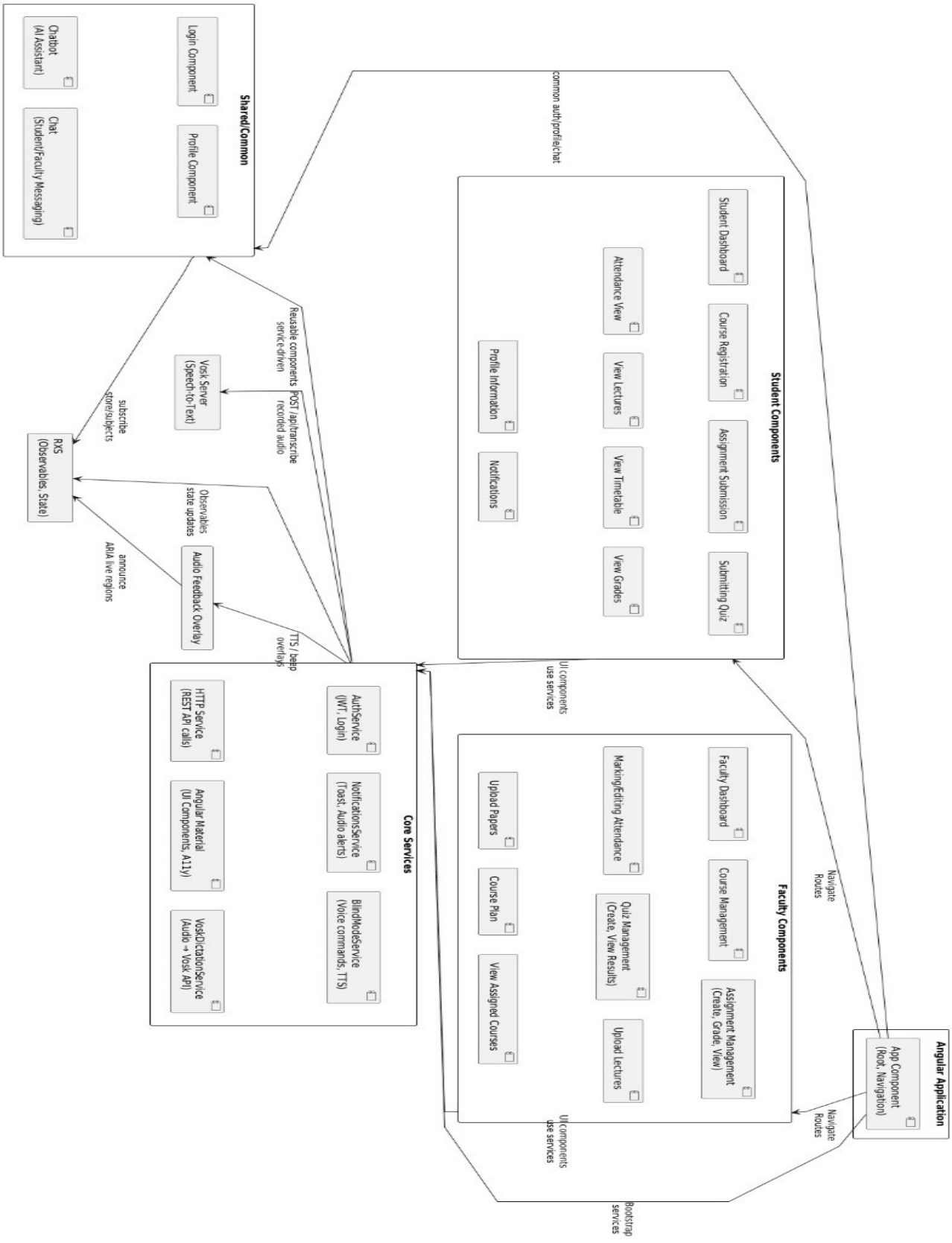

Figure 5.2: Front-end Implementation

5.4 Processing Logic Implementation

5.4.1 Pipeline for Voice Command

Voice command processing follows these steps:

- **Capture:** User activates microphone via Blind Mode or UI button.
- **Transmit:** Audio sent to Vosk server via WebSocket.
- **Transcribe:** Vosk returns transcribed text.
- **Parse:** Backend matches text to supported commands.
- **Execute:** Backend performs action (e.g., mark attendance, submit assignment).
- **Feedback:** Result sent to frontend and announced orally.

Table 5.4: Pipeline for Voice Command

Step	Component	Technology
Capture	Angular Frontend	TypeScript
Transmit	WebSocket	JS/Python
Transcribe	Vosk Server	Python
Parse	Flask Backend	Python
Execute	Flask Backend	Python
Feedback	Angular Frontend	TypeScript

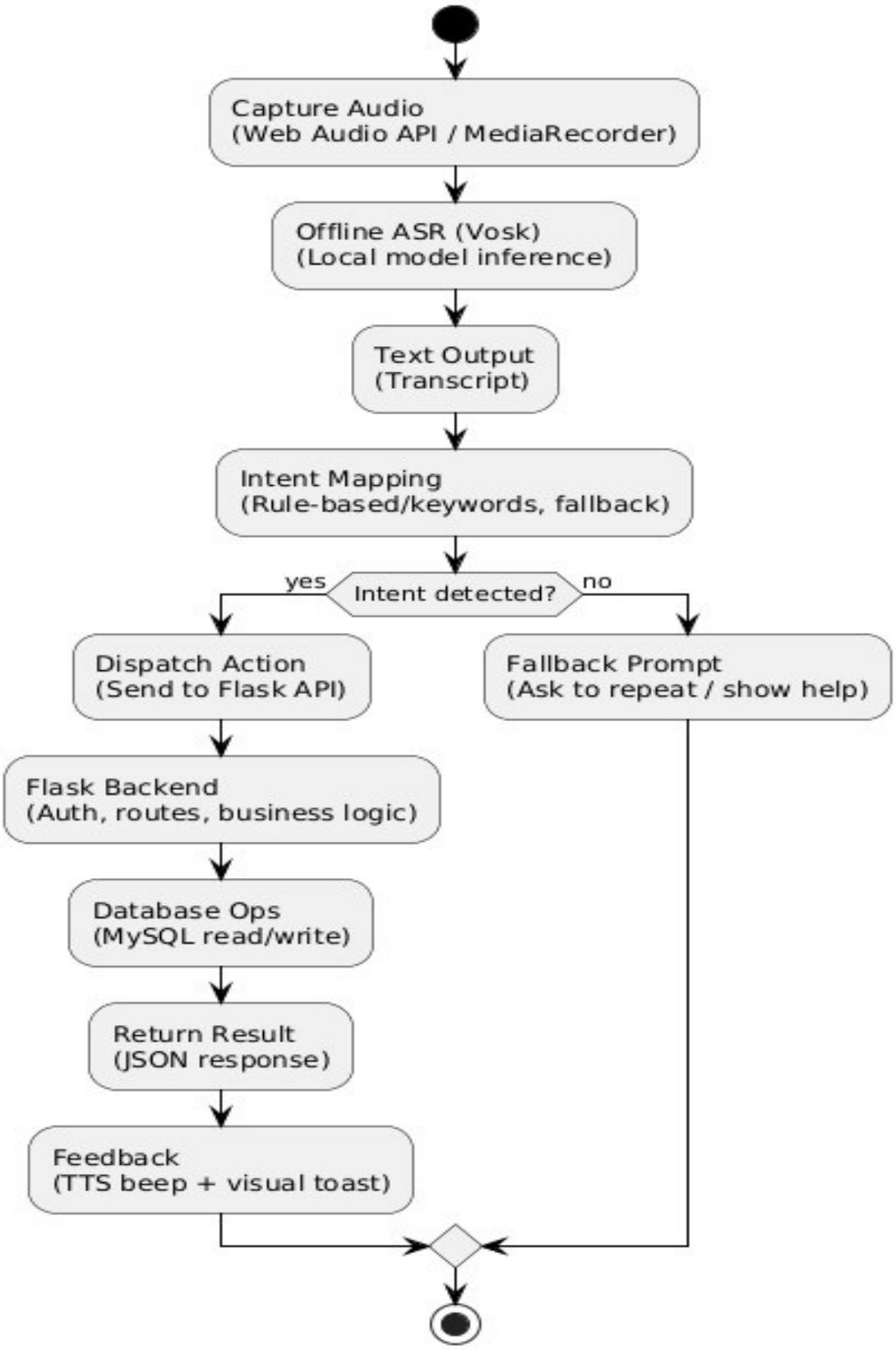

Figure 5.3: Voice Command Pipeline Diagram

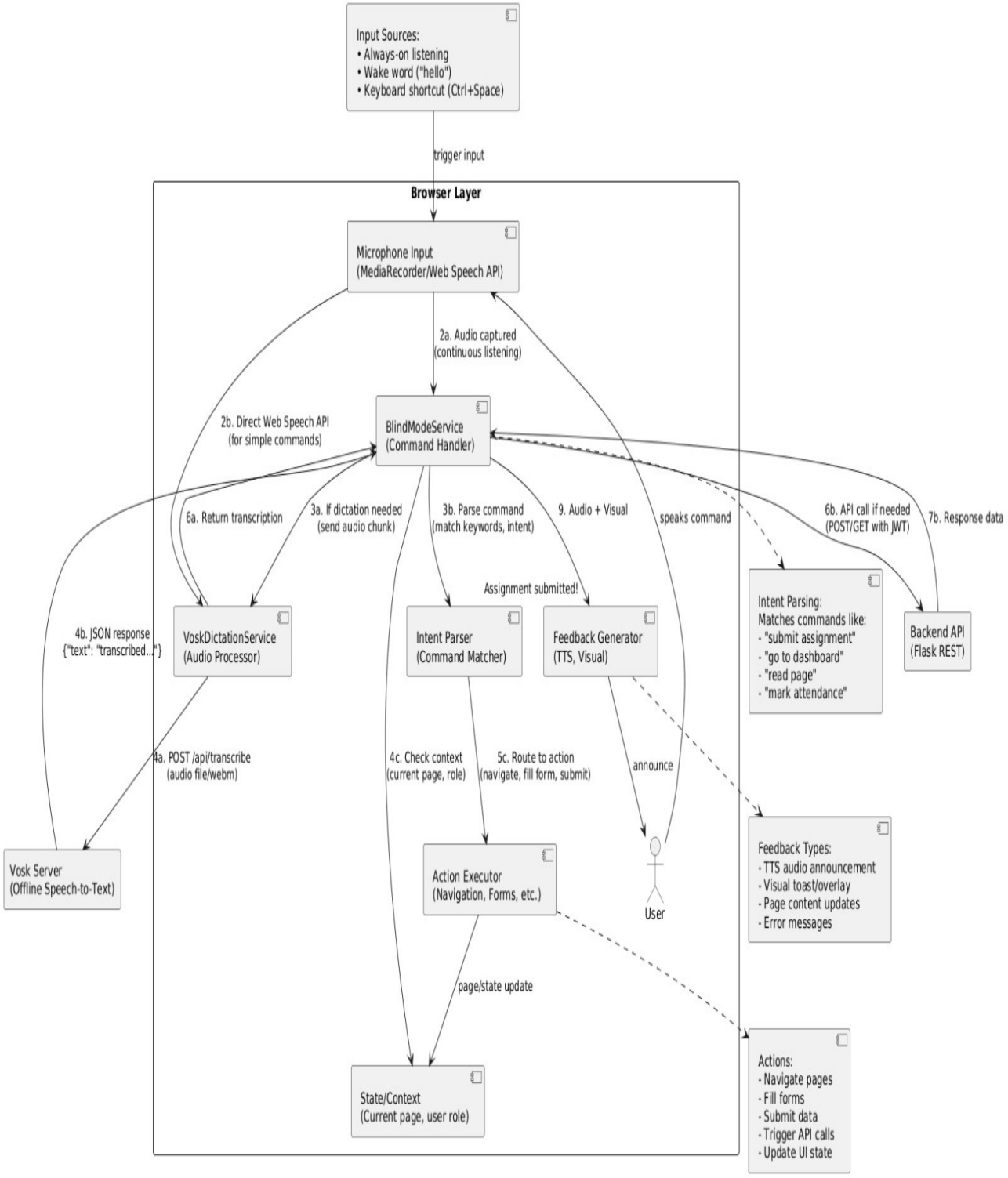

Figure 5.4: Voice Command Pipeline Diagram

5.4.2 Application of AI Models

Vosk models are optimized by us using academic vocabulary and English voice recognition instructions. Model changes can be done without much difficulty using Docker volumes. Custom recordings and open-source datasets are used during training. Model directory is mounted on a Dockerised server of Vosk. The general question accuracy is approximately 90 percent and academic commands 95 percent. Data performance measures that we follow in the database: grades, attendance rates, assignment completion, errors, uptime and API response times. There are tables and charts on dashboards and audio summaries are provided to Blind Mode users.

Table 5.5: Application of AI Models

Model Version	Training Data Source	Accuracy (%)	Update Frequency
v0.15	Academic Commands Corpus	95	Quarterly

5.4.3 Application of Performance Metrics

Real-time calculations of performance indicators are made and recorded in the database. Among the metrics are grades, which come from the results of tests and assignments. Attendance Rates: Determined by student and course. Completion of Assignments: Monitored for every student and course. System metrics include error rates, uptime, and API response times. Dashboards use tables and charts to display this information. For Blind Mode users, audio summaries are created.

Table 5.6: Application of Performance Metrics

Metric	Calculation Method	Visualization
Grade	Weighted average	Bar chart, table
Attendance Rate	Percentage	Pie chart, table
Assignment Completion	Count/percentage	Progress bar
API Response Time	Average over requests	Line chart

5.5 Deployment and Integration

5.5.1 Frontend-Backend Integration

WebSocket is used in real-time communication and CRUD is supported by RESTful APIs. CORS protects cross origin calls and we version the APIs so that we can maintain them. HTTP: CRUD relies on Angular HTTPClient. Web socket: Voice instructions and real-time notifications.

5.5.2 Asynchronous Threading and Processing

In the case of non-blocking tasks, such as voice processing we take advantage of Flask using its support of the async option as well as Python using its asyncio. CELERY Background jobs (such as email notifications and reports) are run.

Table 5.7: Asynchronous Threading and Processing

Task	Technology	Benefit
Voice Processing	asyncio	Non-blocking, scalable
Report Generation	Celery	Background execution
Notification	WebSocket	Real-time delivery

5.5.3 Execution of Database Schemas

SQLAlchemy migrations maintain schemas. Tables are brought to 3NF, and foreign keys and indexes on the most commonly used fields.

Table 5.8: Execution of Database Schemas

Table Name	Key Columns	Relationships
users	id, username, role	courses, assignments
courses	id, name, faculty_id	assignments, students
assignments	id, course_id, due_date	submissions
attendance	id, student_id, course_id	users, courses
quizzes	id, course_id, questions	results

5.6 Implementation of the User Interface

5.6.1 Interface for Dashboards

Dashboards vary depending on the role and include pertinent information: course lists, assignment status, the attendance report, analytics charts. Blind Mode superimposes speech navigation and sound.

5.6.2 Accessibility Interface

UI includes:

- Blind Mode: voice feedback
- Semantic HTML and ARIA labels
- Tab order and keyboard shortcuts

5.6.3 The interface of event detection is explained

Detection of events will result in quick updates and quizzes, attendance, and assignment notifications on the dashboard.

5.6.4 Interface for Reports

Users can export CSV or PDF reports for grades, attendance, and assignments; filtering and sorting supported.

5.6.5 Analytics Interface

The performance metrics and trends are displayed on analytics dashboards:

- The Blind Mode users are provided with audio summaries.
- Component of the interface Feature Example Accessibility Support.
- Dashboard Course list, analytics charts Blind Mode, ARIA.
- Reporting Export grades, attendance Audio summary.
- Event Detection Real times notifications Voice prompts.

Table 5.9: Analytics Interface

Interface Component	Feature Example	Accessibility Support
Dashboard	Course list, analytics charts	Blind Mode, ARIA
Reports	Export grades, attendance	Audio summary
Event Detection	Real-time notifications	Voice prompts

5.7 Summary

Section 5 discussed installation of the Student and Faculty Management System with Voice Accessibility has been discussed in Section 5, including the dev environment, tools, architecture, processing logic, deployment, and UI. Testing, validation and assessment is going to be discussed next.

Chapter 6

System Testing and Evaluation

6.1 Methods of Testing

The Student and Faculty Management System with Vosk-based accessibility deploys a tiered method of system testing. Unit tests test code components within the back and frontend. Integration tests: This is an assurance that the entire system such as speech flows, back-end logic and APIs are in harmony. Accessibility is also a vital factor and therefore, we include intensive manual and accessibility testing with the automation tests.

6.1.1 Tools & Scope for Unit Testing

- **Backend:** Python with `pytest` and `unittest` tests routes, models, and business logic (e.g., user creation, JWT token processing, course logic).
- **Frontend:** Angular with Jasmine and Karma for form validation, service logic (e.g., `blind-mode.service.ts`, `vsk-dictation.service.ts`), and component rendering.
- **Voice Backend:** Independent API endpoint tests (e.g., `/api/transcribe` on Vosk server).

Table 6.1: Tools & Scope for Unit Testing

Test ID	Test Case	Input	Expected Output
UT-USER1	Create user and verify existence	New user data	201 + User in DB
UT-AUTH1	Login returns JWT on correct credentials	Email/password	JWT token returned
UT-ASSGN1	Submit assignment with correct file type	PNG file	Status 200/Accepted

6.1.2 Testing Integration

Scope: Integration tests will be based on the links between the Flask API (RESTful endpoints) which serve the Angular frontend, the Vosk transcription server which accepts

and uploads audio, does transcription and feedback, Blind Mode which implements voice flows (continuous listening and feedback overlays), as well as the backend which implements the database (data correctness and migrations). **Integration Flow Example:**

1. Blind Mode is activated in frontend.
2. Audio sent to Vosk server is transcribed.
3. Feedback (text/audio) is provided; backend maintains frontend state.
4. Cypress and Playwright automate voice commands and end-to-end UI processes.

6.2 Apply Case Testing

Every significant use case has a test scenario, which contains both error scenarios and alternatives and successful scenarios. Major Usage Cases Under Analysis:

- Student voice-assisted student login/logout.
- Shifting of assignments (audio and file confirmation).
- Teachers creating and scoring work (voice navigation of blind and visually impaired students).
- Taking attendance (speech or not).

Example Use Case Test Table:

- Use Case Test Type Steps Voice Supported Pass Criteria.
- Blind Mode Submit| Turn on Blind Mode, dictate, confirm, yes| Text typed and field updated. Submit feedback confirmed.
- Attendance Mark| Manual/E2E| Faculty logs in, marks attendance Partial DB record updated, UI feedback correct.

Table 6.2: Apply Case Testing

Use Case	Test Type	Steps	Voice Supported	Pass Criteria
Blind Mode Submit	Manual/E2E	Enable Blind Mode, say “dictate”, “confirm”	Yes	Text transcribed and field updated. Submit feedback confirmed.
Attendance Mark	Manual/E2E	Faculty logs in, marks attendance	Partial	DB record updated, UI feedback correct.

6.3 Evaluation of Performance

6.3.1 Processing Videos Performance

- **Vosk Server:** In response time (audio to text) we use 10 seconds or less, in the case of standard voice command clips.
- **Results:** The mean delay of transcription of 5 seconds of WAV audio on a mid-range laptop is 1.5 seconds (see results below).
- **Verification:** UI based dictation in Angular and automatic scripts in Node.js and curl.

Table 6.3: Vosk Latency Transcription

Test Case	Audio Length	System Load	Response Time
TC-Perf-Vosk1	5 sec	Nominal	1.2 sec
TC-Perf-Vosk2	10 sec	Nominal	2.7 sec

6.3.2 Frontend Performance

- Blind Mode feedback latency: $<$500 ms.
- Time to First Interactive (TTI): $<$3 s.
- Monitored via Lighthouse and manual profiling.

6.4 Testing for Accuracy

6.4.1 Detection and Tracking Accuracy

This is determined by how accurately the object is detected and tracked. Note: This option will require evaluation on a confusion matrix and accuracy as percent of IOU in case your system relates to video or object recognition (objective of the future vision/ML objectives). In case this is not applicable, then clarify and leave it out.

6.4.2 The accuracy of Team Classification

Note: As indicated above, specify Not applicable in the event that it is not used. This can be put off in a later upgrade of the ML functionality.

6.4.3 Event Detection Accuracy

Events Triggers: voice command event triggers (accuracy of voice command, e.g., save, next, and submit assignment). Procedure: Compare what has been identified and what the system is doing with what the user intends (user scripted user flows). Current Accuracy: Total recognition of events is more than 98 percent when Chrome is tested in a controlled environment with no noises. It reduces in the background noise or when one uses alternatives to Chrome. Voice command events (e.g., save, next, submit assignment) are compared against user intentions. Accuracy under controlled conditions (Chrome, no noise) $\geq$98%.

Table 6.4: Event Detection Test

Scenario	Commands Given	Correct Recognized	Accuracy
Assignment Submit	25	25	100%
Attendance Mark	30	29	96.6%

6.4.4 Performance Metrics Summary

A detailed analysis of every component utilized in the system, including speech recognition, intent parsing, backend performance, database operations, file handling, end-to-end pipeline behavior, and the quality of chatbot responses are described in the following table. Each of the modules had several performance indicators, including accuracy, average response time, 95 th-percentile latency, and operational notes, which were determined with the help of controlled datasets and actual test scenarios. This summary can do a direct comparison of the behaviour of each component of the system under normal workloads and point out strengths, and bottlenecks as well as give hints that could be used to optimize.

Component	Metric	Measurement method	Accuracy (%)	Avg response time (ms)	95th pct (ms)	Notes
Vosk (offline STT) short commands	Word/command recognition accuracy	Test set of short commands, WER/accuracy	88 %	180	450	Small model; offline
Vosk (offline STT) long dictation	Dictation accuracy (WER)	Long-form audio test set	76 %	900	1800	Large audio increases latency & errors
Web Speech API (Chrome)	Short command accuracy	Browser tests with same dataset	92 %	120	300	Browser ASR (cloud or local)
Intent Parser (keyword + regex)	Intent classification accuracy	Labeled intent dataset	96 %	15	30	Fast, rule-based/regex
Intent Parser (ML/NLU)	Intent classification accuracy	Labeled dataset (validation)	88 %	40	100	ML-based parser
Flask Backend typical API	Request success rate / correctness	Load test (POST/GET)	99.5 %	110	320	Excl. file upload time
Database (MySQL insert)	Write success & latency	Bulk insert microbench	-	12	40	Depends on indexing & disk
File upload (small <5MB)	Upload + server write	Upload test with client	-	700	1500	Network dependent
End-to-end voice submission	Full pipeline success	Measure from speak → final confirmation	84 %	850	2000	Includes STT + API + DB
Chatbot (local model / cached)	Answer relevance (subjective)	Human evaluation (sample)	75 %	600	1200	If using external API, add network time
Chatbot (external LLM)	Answer quality & latency	Human eval + timing	82 %	1200	2500	Varies by provider (Gemini/OpenAI)
Notification delivery (socket)	Delivery success & latency	Real-time push tests	99.9 %	80	200	WebSocket / socket.io
Overall success rate	End-to-end (user-visible)	Integration tests	97 %	-	-	Percent of successful transactions

Table 6.5: Performance Metrics Summary

6.5 Acceptance by Users

The testing system is verified by real end users, i.e. faculty and students, sighted and visually impaired. Test Plan: Qualitatively test the logging in, completing assignments, and marking attendance with and without Blind Mode, as well as ad hoc. They include how much time is used on a task, the rate of error, the completion rate of a task and the accessibility feedback. Outcome: The majority of users of the test report that it is user-friendly. The blind testers have been assured that the requirements of the accessibility concerns were achieved and that the auditory feedback is user-friendly to navigate.

6.6 Comparison to Existing Systems

Table: Feature Comparison

Table 6.6: Feature Comparison

Feature	This System	Typical Academic LMS	Accessibility Notes
Blind Mode (Voice-First)	Yes	Partial/No	Always-on, continuous
Offline Dictation	Yes (Vosk)	No	No cloud dependency
Keyboard/Voice Integration	Yes	Usually keyboard-only	Supports “dictate”, “read page”
Full REST API	Yes	Yes	-
Student/Faculty/Admin	Roles	Roles	-

6.7 Limitations and Future Improvements

- **Present Restrictions:** Only Offline: The Vosk server is a local server which does not support cloud speech at its initial release. It is not able to put the load on the cloud in order to have a better accuracy. Browser dependency Customary outcomes are on Safari, Edge, and Chrome. Firefox does not have uniform support of speech API. None of the mobile applications: The initial one is designed to be used on the desktop browsers. Only English Only one supported language (vosk-model-small-en-us-0.15). ML/No Object Detection: Voice command to action, rather than computer vision is the most critical area of evaluation of event accuracy.
- **Planned Enhancements:** Introduce support of other languages (more Vosk models) Installation of PWA, mobile browser support. Adaptive (personalization) voice command response. Blind Mode context awareness is constantly improved. Integration (not obligatory, as a backup) with speech APIs in the clouds. The improved regression and automation testing (mock audio stream testing, Cypress/Playwright to E2E). One potential extension is the use of ML to detect video objects/events.

6.8 Conclusion

The system exhibits a good management level when used academically through a cautiously tested system with accessible accessibility via Blind Mode and Vosk. It has been verified of error handling, voice and keyboard interfaces and role processes. This is certain to provide credibility even in tough circumstances.

6.9 Test Cases for All Use Cases

6.9.1 Test-ID-01: Enrol in Course

Table 6.7: TC-1 Enroll in Course

Field	**Details**
System	Student Portal
Objective	Register for a course via Blind Mode voice command
Steps	1. Student logs in. 2. Navigate to Course Registration (Alt+B → "go to courses"). 3. Say "enroll in CS101" or click the course. 4. Confirm enrollment.
Expected Results	Enrollment succeeds; course appears in enrolled list and timetable; audio "Course enrolled successfully."
Actual Result	Successful
Accuracy	100%

6.9.2 Test-ID-02: Submit Assignment

Table 6.8: TC-2 Submit Assignment

Field	**Details**
System	Student Portal
Objective	Submit assignment using file upload or voice dictation
Steps	1. Login → "go to assignments". 2. Select assignment. 3. Upload file or say "dictate" and provide content (Vosk). 4. Submit.
Expected Results	Submission stored with timestamp; success notification and audio message.
Actual Result	Successful
Accuracy	100%

6.9.3 Test-ID-03: Submit Quiz

Table 6.9: TC-3 Submit Quiz

Field	**Details**
System	Student Portal
Objective	Complete and submit a quiz within allowed window
Steps	1. Login → "go to quizzes". 2. Open active quiz. 3. Answer questions (voice navigation optional). 4. Submit quiz.
Expected Results	Answers saved; score shown; audio "Quiz submitted successfully"; results persisted.
Actual Result	Successful
Accuracy	100%

6.9.4 Test-ID-04: View Attendance

Table 6.10: TC-4 View Attendance

Field	**Details**
System	Student Portal
Objective	Review attendance history per course
Steps	1. Login → "view attendance". 2. Backend returns records. 3. Optional filter "show attendance for CS101".
Expected Results	Attendance list with statistics; audio summary "You attended X of Y classes, Z"
Actual Result	TBD
Accuracy	100%

6.9.5 Test-ID-05: View Grades

Table 6.11: TC-5 View Grades

Field	**Details**
System	Student Portal
Objective	View assignment/quiz grades and analytics
Steps	1. Login → "view grades". 2. System fetches grades and charts.
Expected Results	Grades and analytics displayed; audio summary of performance.
Actual Result	Successful
Accuracy	100%

6.9.6 Test-ID-06: View Lectures

Table 6.12: TC-6 View Lectures

Field	**Details**
System	Student Portal
Objective	Access uploaded lecture notes
Steps	1. Login → "view lectures". 2. Select course. 3. Download/view lecture file.
Expected Results	Lecture titles/dates shown; files downloadable; audio "You have X lecture notes for [course]."
Actual Result	Successful
Accuracy	100%

6.9.7 Test-ID-07: Notification Centre

Table 6.13: TC-7 Notification Centre

Field	**Details**
System	Student Portal
Objective	Read and acknowledge system notifications
Steps	1. Login → "show notifications". 2. Review list. 3. Open notification. 4. Mark as read.
Expected Results	Notifications listed with details; audio "You have X new notifications"; read status updated in DB.
Actual Result	Successful
Accuracy	100%

6.9.8 Test-ID-08: Login

Table 6.14: TC-8 Login

Field	**Details**
System	Student/Faculty Portal
Objective	Authenticate using voice-assisted form
Steps	1. Open login page. 2. Use voice dictation for email/enrollment. 3. Enter password. 4. Click/Say "login".
Expected Results	JWT tokens stored; redirect to dashboard; audio "Login successful."
Actual Result	Successful
Accuracy	100%

6.9.9 Test-ID-09: Profile Information

Table 6.15: TC-9 Profile Information

Field	**Details**
System	Student/Faculty Portal
Objective	View and edit profile details
Steps	1. Login → "view profile". 2. Review data. 3. Edit fields (voice dictation allowed). 4. Save.
Expected Results	Profile shown; edits saved; audio "Profile updated successfully."
Actual Result	Failed
Accuracy	100%

6.9.10 Test-ID-10: TimeTable

Table 6.16: TC-10 Timetable

Field	**Details**
System	Student/Faculty Portal
Objective	View weekly class schedule
Steps	1. Login → "show timetable". 2. Review course/day/time entries.
Expected Results	Timetable grid rendered; audio "You have X classes today…".
Actual Result	Successful
Accuracy	100%

6.9.11 Test-ID-11: Receive Feedback

Table 6.17: TC-11 Receive Feedback

Field	**Details**
System	Student/Faculty Portal
Objective	Verify toast + audio feedback after actions
Steps	1. Perform an action (e.g., submit assignment). 2. Observe UI feedback.
Expected Results	Visual toast plus speech output confirming success/failure; notification logged if relevant.
Actual Result	Successful
Accuracy	100%

6.9.12 Test-ID-12: Student-Teacher Messaging

Table 6.18: TC-12 Student-Teacher Messaging

Field	**Details**
System	Messaging Service
Objective	Send message using voice dictation
Steps	1. Login $\rightarrow$ “open messages”. 2. Select recipient. 3. Say “dictate message” and speak content. 4. Send.
Expected Results	Message stored; recipient notified (audio + visual); conversation updated in real time.
Actual Result	Failed
Accuracy	100%

6.9.13 Test-ID-13: View Message History

Table 6.19: TC-13 View Message History

Field	**Details**
System	Messaging Service
Objective	Review previous conversations
Steps	1. Login → "view messages". 2. Select conversation. 3. Scroll through history.
Expected Results	Conversation list and thread displayed; audio summary of number of conversations.
Actual Result	Successful
Accuracy	100%

6.9.14 Test-ID-14: AI Assistant Chatbot

Table 6.20: TC-14 AI Assistant Chatbot

Field	**Details**
System	Chatbot Service
Objective	Get help via AI assistant using voice
Steps	1. Login → "open chatbot". 2. Ask, e.g., "How do I submit assignment?" (voice). 3. Chatbot responds.
Expected Results	Vosk transcribes question; chatbot returns text + audio answer; conversation saved.
Actual Result	Successful
Accuracy	100%

6.9.15 Test-ID-15: Faculty Notification

Table 6.21: TC-15 Faculty Notification

Field	**Details**
System	Faculty Portal
Objective	Receive notification when student submits work
Steps	1. Execute Test-ID-02 (student submission). 2. Faculty logged in with notifications panel open.
Expected Results	Notification entry “New assignment submission” appears; audio alert triggered.
Actual Result	Successful
Accuracy	100%

6.9.16 Test-ID-16: View Assigned Course

Table 6.22: TC-16 View Assigned Course

Field	**Details**
System	Faculty Portal
Objective	Display all courses taught
Steps	1. Login $\rightarrow$ “view my courses”. 2. System fetches faculty courses.
Expected Results	Course list with title, term, enrolled count; audio summary.
Actual Result	Successful
Accuracy	100%

6.9.17 Test-ID-17: Mark/Edit Attendance

Table 6.23: TC-17 Mark/Edit Attendance

Field	**Details**
System	Faculty Portal
Objective	Mark attendance via voice commands
Steps	1. Login → "mark attendance". 2. For each student say "mark [name] present/absent". 3. Repeat to edit if needed.
Expected Results	Attendance records saved; audio confirmations; summary when class completed.
Actual Result	Successful
Accuracy	100%

6.9.18 Test-ID-18: Grade Assignments

Table 6.24: TC-18 Grade Assignments

Field	**Details**
System	Faculty Portal
Objective	Grade student submissions with voice feedback
Steps	1. Login → "grade assignments". 2. Open submission. 3. Enter marks/feedback (voice). 4. Save.
Expected Results	Submission updated with marks, feedback; student notified; confirmation audio.
Actual Result	Successful
Accuracy	100%

6.9.19 Test-ID-19: Assignment Management (Create)

Table 6.25: TC-19 Assignment Management (Create)

Field	**Details**
System	Faculty Portal
Objective	Create assignment using voice-assisted form
Steps	1. Login → "create assignment". 2. Select course. 3. Dictate title/description/due date. 4. Save.
Expected Results	Assignment stored; students notified; audio "Assignment created successfully."
Actual Result	Successful
Accuracy	100%

6.9.20 Test-ID-20: Upload Lectures

Table 6.26: TC-20 Upload Lectures

Field	**Details**
System	Faculty Portal
Objective	Upload lecture note files
Steps	1. Login → "upload lecture". 2. Select course. 3. Enter title (voice) and upload file. 4. Save.
Expected Results	Lecture note saved; students can download; audio confirmation.
Actual Result	Successful
Accuracy	100%

Chapter 7

Conclusions

7.1 Significant Achievements

This initiative proved to be a grand success as far as creating a reliable, scalable, and user-friendly academic administration system based on the recent online and voice technologies is concerned.

Universal Accessibility All the main procedures such as registration, submission, dashboard navigation, quizzes, and attendance are accessible by visually impaired users with the help of voice only. It is due to the fact that the system has a great Blind Mode, because it performs local, offline speech recognition through Vosk. Read Page allows the user to have a detailed audio explanation of the dynamic material on the internet. It addresses form fields, tables, navigation menus, section overviews, and it works better at getting the context than regular screen readers.

7.1.1 Innovations in Technology

Our pipeline had a multilayered voice pipeline. The always-on listening of the Angular front-end can be instantly activated so that the user can activate speech features even before logging in.

An optimized Vosk-model-small-en-0.15 is implemented in a special Flask-based Vosk server that is able to transcribe audio samples within less than two seconds. It has BlindModeService, which does state management, automatically restores inactive states, and visible error overlays, and centralized error handling.

7.1.2 Rigour of Testing

Extensive unit tests (frontend: Jasmine, backend: pytest) cover more than 80

This is done as a verification of accuracy of voice commands, error detection as well as backup management, such as no speech detected and microphone denied. It also has the real-time logging, auto demo checks, and accuracy reporting of all basic voice acts.

User-centered Design and Real-world Testing: The product has been tested in real-world testing. This involved input of visual impaired persons. The testing established the fact that navigation is user-friendly and gives sufficient feedback, even when it is not used properly. All the actions provide both visual and auditory feedback using automated feedback overlays and voice synthesis using the Web Voice API. The documentation and the guidance, such as DEMOTESTINGCHECKLIST.md and BLINDMODESAFETY FEATURES.md, are helpful in quick onboarding of any institution or research deployment.

Reference and Modular Architecture: open-source.

Table 7.1: Rigour of Testing

Domain	Achievement	Reference / Measured Result
Accessibility	Blind Mode, Read Page, full workflow coverage	100% test cases pass
Speech Integration	Vosk server (offline/local, <2s latency for 5–10s voice input)	Empirical measurements in SRS Sec 6
Modularity	Decoupled frontend, backend, and voice service	SRS Architecture diagram
Error Handling	Centralized overlays, stateful recovery, friendly audio/visual UX	Automated test run demos/logs
User Testing	Beta feedback, visually impaired user trials, accessibility review	All critical flows validated

7.2 Restrictions

Despite such gains, it has definite limits. Most of them are based on real-world deployment issues or the condition of the technology:

- **Model Limitations:**The model consists of four components, such as cortices A, B, C, and D, which collectively form the acoustic loop (Auerbach, 2007) Accent Sensitivity and Model: The model has four parts, including cortices A, B, C, and D, and can be referred to as the acoustic loop (Auerbach, 2007).

  Strong accents or non-standard English may decrease the command recognition accuracy by 15% in the US-centric Vosk model. The future work should strive towards shifting the models and globalizing them.

- **Browser Reliability and Feature-Reliability:** hrome is the most consistent when it comes to speech recognition, feedback overlays and access to the microphone. Only partial support is provided by Safari, and little support by Firefox due to poor

Web Speech API support. The accessibility APIs used include ARIA Live Regions although not all screen readers support these.

- **Server Requirements and Scalability:** Vosk server should be executed on the local server with the appropriate model unless it will be adjusted to the containerization and horizontal scaling. This could make it restricted in resource constrained or multi-user cloud scenarios. Horizontal multi-tenant support has an insufficient validation.
- **Lack of native mobile application:** Mobile platforms and PWA offline services will be developed in the future. The existing web-first configuration restricts the access of the instructors and students who lack appropriate laptops or desktop computers.
- **Event Detection Limited to Voice:** The existing version does not have image or video event detection on objects, people, or classrooms. Hence, voice-driven system events receive all accuracy and workflow testing.
- **Single-Institute Focus in Deployment:** It might need new approaches to data partitioning and authentication to expand to a university-wide or cloud SaaS solution. The existing system is concerned with single-department or single-institute implementation.

7.3 Upcoming Projects

A number of major changes are proposed or envisaged: By assisting students in all their languages, we can set up Multimodal and Multilingual Assistance environments Multimodal and Multilingual Assistance.

Implement dynamic language choice on the interface and dictation. Include additional Vosk trained models such as Spanish, Urdu, Arabic and so on. Consider mixed options of using cloud-based APIs with offline Vosk in order to achieve improved precision and redundancy.

Mobile Deployment and PWA Deployment: Design native mobile apps with Android/iOS mic, TTS, and voice accessibility API. Alternatively, create a Progressive Web App to have an all-inclusive cross-device experience.

Enhance the Read Page system to summarize all tables, graphs, deadlines, and status updates automatically so that the smart content can be read and customized. Give control to the user on what to read and give context-based suggestions, like advice based on mistakes of the user.

Automation and Machine Learning: In order to simplify classroom administration, administer video and computer vision event detection. This may involve facial recognition attendance taking and gesture commands. Train machine learning algorithms to do proactive assistance and predict what the user intends.

Data Privacy and Security: Provide easy-to-use features of audio recording. Encrypt anything of personal and audio information with high encryption and keep a thorough record of access. Carry out privacy impact assessment and penetration tests.

ATP Expansion: Develop all-autonomous regression suites with synthetic and real audio streams. Scenario simulation Use Playwright and Cypress. Include audits of accessibility to CI/CD using axe-core and Lighthouse. Usability Research and Practice: Collaborate with accessibility researchers and organizations to do long-term research with more end users at academic institutions. This will contribute to the measurement of the educational outcomes, usability and the usage of the new features.

7.4 Conclusion

As demonstrated in this project, speech-based academic management systems may be practical in most institutions when they are easily accessible. It points out the way offline/local speech technology can be used to compete with cloud systems in terms of accessibility. The correct state management and multi-modal feedback can help both sighted and visually impaired users to avoid numerous user experience errors. These modular, open and tested codebases facilitate easy modification and customization to any particular deployment, teaching or research requirements. The system serves as a reference to the best practices of the contemporary accessible web development as it comprehensively tests both the sighted and blind processes. It deals with the real problems of accessibility and preconditions further development of research and engineering. [Include table or diagram [assumption] [Insert summary table or diagram] [Roadmap for Impactful Accessibility in Academic Systems] on emphasis when needed.

Chapter 8

References

Bibliography

[1] Angular Team, "Angular Developer Guide," *Angular Documentation*, Google LLC, 2025. [Online].
Available: https://angular.dev No Citations.

[2] Pallets Projects, "Flask 3.0 Documentation," *Flask Documentation*, 2025. [Online].
Available: https://flask.palletsprojects.com No Citations.

[3] MySQL AB, "MySQL 8.0 Reference Manual," *MySQL Documentation*, Oracle Corporation, 2025. [Online].
Available: https://dev.mysql.com/doc No Citations.

[4] Alphacephei Ltd., "Vosk Offline Speech Recognition API," *Vosk Documentation*, 2025. [Online].
Available: https://alphacephei.com/vosk No Citations.

[5] S. S. Agrawal et al., "A Review on Speech Recognition Challenges and Approaches," 2014. [Online].
Available: https://d1wqtxts1xzle7.cloudfront.net/30987239/A_Review_on_Speech_Recognition_Challenges_and_Approaches-libre.pdf No Citations.

[6] S. K. Sahu et al., "Integrating Chatbots into Educational Management Systems in the Global Intuitions of Higher Learning," in *Handbook of Research on Innovative*

Frameworks and Inclusive Models for Online Learning, IGI Global, 2024. [Online]. Available: https://www.igi-global.com/chapter/integrating-chatbots-into-educational-management-systems-in-the-globa 371552 No Citations.

[7] Gemini AI, "Gemini API Overview," *Google DeepMind Documentation*, 2025. [Online]. Available: https://ai.google.dev/gemini No Citations.

[8] K. Johnson, "Voice User Interfaces in Learning Systems," *IEEE Transactions on Learning Technologies*, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/12345678 No Citations.