# Timetable Buddy

Submitted in partial fulfillment of the requirements of the degree

**BACHELOR OF TECHNOLOGY**

IN

**INFORMATION TECHNOLOGY**

By

| | |
|---|---|
| Sarthak Kulkarni | 23101B0019 |
| Dhruv Tikhande | 23101B0005 |
| Atharv Petkar | 23101B0010 |
| Pulkit Saini | 23101B0021 |

Supervisor

**Prof. Dhanashree Tamhane**

Department of Information Technology
Vidyalankar Institute of Technology
Vidyalankar Educational Campus,
Wadala(E), Mumbai - 400 037

University of Mumbai

(AY 2025-26)

# Certificate

This is to certify that the Mini Project entitled **"Timetable Buddy"** is a bonafide work of **Sarthak Kulkarni (23101B0019), Dhruv Tikhande (23101B0005), Atharv Petkar (23101B0010), and Pulkit Saini (23101B0021)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **"Bachelor of Technology"** in **"Information Technology"**.

**Prof. Dhanashree Tamhane**
Supervisor

| | |
|---|---|
| **Internal Examiner** | **External Examiner** |
| Name & Sign | Name & Sign |

# Declaration

We, **Sarthak Kulkarni (23101B0019), Dhruv Tikhande (23101B0005), Atharv Petkar (23101B0010), and Pulkit Saini (23101B0021)**, students of Bachelor of Technology in Information Technology, hereby declare that the project work entitled **"Timetable Buddy"** submitted to the Vidyalankar Institute of Technology, University of Mumbai, is our original work and has not been submitted to any other university or institution for the award of any degree or diploma.

The project has been completed under the guidance of **Prof. Dhanashree Tamhane**.

| | |
|---|---|
| Sarthak Kulkarni | 23101B0019 |
| Dhruv Tikhande | 23101B0005 |
| Atharv Petkar | 23101B0010 |
| Pulkit Saini | 23101B0021 |

Date: ______________
Place: Mumbai

# Acknowledgment

We would like to express our sincere gratitude to all those who have contributed to the successful completion of this project.

We are deeply grateful to our project supervisor, **Prof. Dhanashree Tamhane**, for her invaluable guidance, continuous support, and encouragement throughout the development of this project. Her expertise and insights have been instrumental in shaping the direction of our work.

We extend our thanks to **Dr. [Head of Department]**, Head of the Department of Information Technology, and all the faculty members for providing us with the necessary facilities and resources to complete this project.

We would also like to thank our family and friends for their constant support and motivation during this journey.

Finally, we are thankful to **Vidyalankar Institute of Technology** and **University of Mumbai** for giving us the opportunity to work on this project.

| | |
|---|---|
| Sarthak Kulkarni | 23101B0019 |
| Dhruv Tikhande | 23101B0005 |
| Atharv Petkar | 23101B0010 |
| Pulkit Saini | 23101B0021 |

# Abstract

The **Timetable Buddy** is a comprehensive lecture scheduling system designed to streamline the process of managing academic schedules for educational institutions. The system addresses the challenges faced by students, faculty, and administrators in coordinating class schedules, managing enrollments, and accessing timetable information efficiently.

This web-based application provides a centralized platform for managing lecture slots, course enrollments, and student-faculty interactions. The system features role-based access control, supporting three distinct user types: administrators, faculty members, and students. Each role has specific functionalities tailored to their needs.

Key features include real-time lecture slot management, automated enrollment processing with waitlist capabilities, conflict detection for overlapping schedules, and comprehensive dashboard views for different user roles. The system is built using modern web technologies including React for the frontend, Node.js with Express for the backend, and MongoDB for data persistence.

The project follows industry-standard software development practices, including comprehensive test planning, risk assessment, and quality assurance procedures. A complete test case planning and execution document has been developed, covering 60 test cases across various functional areas. Additionally, a risk assessment framework has been implemented to identify and mitigate potential project risks.

The system aims to improve the efficiency of academic schedule management, reduce scheduling conflicts, and enhance the overall user experience for all stakeholders in the educational ecosystem.

**Keywords:** Lecture Scheduling, Timetable Management, Educational Technology, Web Application, Enrollment System, MERN Stack

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Introduction

The Timetable Buddy is a comprehensive web-based lecture scheduling system designed to revolutionize the way educational institutions manage their academic schedules. In today's fast-paced educational environment, the need for efficient, reliable, and user-friendly scheduling tools has become paramount. This system addresses these needs by providing an integrated platform that simplifies the complex process of managing lecture slots, student enrollments, and faculty schedules.

Traditional methods of academic scheduling often rely on manual processes, spreadsheets, and fragmented communication channels, leading to inefficiencies, scheduling conflicts, and administrative overhead. The Timetable Buddy eliminates these challenges by offering a centralized, automated solution that ensures seamless coordination between students, faculty, and administrators.

Built using modern web technologies including the MERN stack (MongoDB, Express.js, React, Node.js), the system leverages the power of full-stack JavaScript development to deliver a responsive, scalable, and maintainable application. The frontend is developed using React with TypeScript for enhanced type safety and developer experience, while the backend utilizes Node.js and Express.js to provide a robust REST API architecture.

The system's architecture follows industry best practices, including role-based access control (RBAC), JWT-based authentication, and comprehensive input validation. This ensures that the application is not only feature-rich but also secure and reliable. The responsive design, powered by Tailwind CSS, guarantees an optimal user experience across all devices, from desktop computers to mobile phones.

## 1.2 Objectives

The primary objectives of the Timetable Buddy project are:

1. **Centralized Schedule Management:** To develop a unified platform where all stakeholders (students, faculty, and administrators) can access and manage lecture schedules from a single interface.

2. **Automated Enrollment Processing:** To implement an intelligent enrollment system that handles student registrations, manages waiting lists, and automatically promotes students when slots become available.

3. **Role-Based Access Control:** To provide differentiated access and functionality based on user roles, ensuring that each user type (student, faculty, admin) has appropriate permissions and views.
4. **Real-Time Conflict Detection:** To enable automatic detection of scheduling conflicts, preventing students from enrolling in overlapping lecture slots and helping faculty avoid double-booking.
5. **Intuitive User Interface:** To create a modern, user-friendly interface that requires minimal training and provides an excellent user experience across all devices.
6. **Comprehensive Testing and Quality Assurance:** To ensure system reliability through extensive testing, including 60 comprehensive test cases covering all functional areas.
7. **Scalable Architecture:** To build a system that can easily scale to accommodate growing numbers of users, courses, and lecture slots without performance degradation.
8. **Data Security:** To implement robust security measures including password encryption, JWT-based authentication, and protection against common web vulnerabilities.

## 1.3 Problem Statement

Educational institutions face numerous challenges in managing their lecture schedules effectively:

- **Manual Scheduling Inefficiencies:** Traditional scheduling methods using spreadsheets and manual coordination are time-consuming, error-prone, and difficult to maintain.
- **Limited Accessibility:** Students and faculty often struggle to access schedule information in real-time, leading to confusion and missed classes.
- **Scheduling Conflicts:** Without automated conflict detection, students may accidentally enroll in overlapping classes, and faculty may be double-booked for lecture slots.
- **Capacity Management:** Manual tracking of course capacity and waitlists is challenging, often resulting in overcrowded classes or underutilized slots.
- **Communication Gaps:** Lack of centralized communication channels leads to delays in notifying students about enrollment status, schedule changes, or important updates.
- **Administrative Overhead:** Managing enrollments, generating timetables, and handling student queries requires significant administrative effort.
- **Limited Reporting:** Existing systems often lack comprehensive analytics and reporting capabilities, making it difficult to track enrollment trends and optimize resource allocation.

The Timetable Buddy system addresses these challenges by providing an automated, centralized, and user-friendly platform that streamlines the entire scheduling process, reduces administrative burden, and enhances the overall academic experience for all stakeholders.

# Chapter 2

# Specific Requirements

## 2.1 Functional Requirements

The Timetable Buddy system provides a comprehensive set of functional capabilities across different user roles. The following sections detail the specific functional requirements implemented in the system.

### 2.1.1 Authentication and User Management Functions

1. **User Registration**
    - New users can create accounts with email, password, and role selection
    - Email validation ensures unique user accounts
    - Password strength requirements enforce security standards
    - Role assignment (Student, Faculty, Admin) during registration
2. **User Authentication**
    - Secure login with email and password credentials
    - JWT-based token generation for session management
    - Password encryption using bcrypt hashing algorithm
    - Automatic session timeout after inactivity
3. **Profile Management**
    - Users can view and update their personal information
    - Students can manage student ID, year, and department details
    - Faculty can update employee ID and department information
    - Password change functionality with verification

### 2.1.2 Lecture Slot Management Functions

1. **Create Lecture Slots** (Faculty/Admin)
    - Define subject name, venue, and capacity
    - Set schedule (day of week, start time, end time)
    - Configure recurring or one-time sessions

- Assign faculty to lecture slots

2. **View Lecture Slots** (All Users)
   - Browse all available lecture slots
   - Filter by subject, faculty, day, or time
   - Search functionality for quick access
   - View enrollment count and available seats
3. **Update Lecture Slots** (Faculty/Admin)
   - Modify lecture slot details
   - Adjust capacity based on room changes
   - Update schedule information
   - Activate or deactivate slots
4. **Delete Lecture Slots** (Admin)
   - Remove obsolete lecture slots
   - Handle enrolled student notifications
   - Archive historical data

### 2.1.3 Enrollment Management Functions

1. **Student Enrollment**
   - Enroll in available lecture slots
   - Automatic conflict detection for overlapping schedules
   - Join waitlist when slot is full
   - View enrollment confirmation
2. **Waitlist Management**
   - Automatic position tracking in waitlist
   - Auto-promotion when seats become available
   - Waitlist position visibility
   - Notification on enrollment status change
3. **Drop Enrollment**
   - Students can withdraw from enrolled slots
   - Automatic waitlist promotion processing
   - Enrollment history tracking
4. **View Enrollments** (Faculty)
   - View list of enrolled students per slot
   - Access student contact information
   - Monitor enrollment statistics
   - Export enrollment data

### 2.1.4 Timetable and Schedule Functions

1. **Personal Timetable View**
   - Weekly grid view of enrolled lectures
   - Daily schedule overview
   - Color-coded subject identification
   - Time slot visualization
2. **Timetable Export**
   - Export to PDF format
   - Print-friendly formatting
   - Include all enrolled course details
3. **Schedule Management**
   - Create custom schedules (Admin/Faculty)
   - Manage recurring lecture patterns
   - Handle schedule modifications

### 2.1.5 Dashboard and Analytics Functions

1. **Student Dashboard**
   - Overview of enrolled courses
   - Upcoming lectures display
   - Quick access to enrollment actions
   - Statistics on completed vs. pending enrollments
2. **Faculty Dashboard**
   - Total lecture slots managed
   - Enrollment statistics per slot
   - Student count across all slots
   - Quick links to manage slots
3. **Admin Dashboard**
   - System-wide statistics
   - User management overview
   - Enrollment trends and analytics
   - System health monitoring

### 2.1.6 Additional Functions

1. **Search and Filter**
    - Search lecture slots by subject name
    - Filter by faculty, day, or time
    - Advanced search with multiple criteria
2. **Notifications**
    - Toast notifications for user actions
    - Success/error message display
    - Real-time feedback on operations
3. **Data Validation**
    - Input validation on all forms
    - Server-side data verification
    - Error message display for invalid inputs

**Total Functions Provided:** The Timetable Buddy system implements approximately **30+ distinct functions** across authentication, lecture slot management, enrollment processing, timetable viewing, dashboard analytics, and administrative operations.

## 2.2 Non-Functional Requirements

Non-functional requirements define the quality attributes and constraints of the system. These requirements ensure that the Timetable Buddy not only functions correctly but also provides an excellent user experience.

### 2.2.1 Performance Requirements

1. **Response Time**
    - API responses should complete within 500ms for standard operations
    - Database queries optimized for sub-200ms execution
    - Page load time under 2 seconds on standard connections
    - Real-time updates with minimal latency
2. **Scalability**
    - Support for 1000+ concurrent users
    - Horizontal scaling capability with load balancing
    - Database indexing for efficient queries
    - Optimized data structures and algorithms
3. **Resource Optimization**

- Minimal memory footprint
- Efficient database connection pooling
- Frontend code splitting and lazy loading
- CDN integration for static assets

### 2.2.2 Security Requirements

1. **Authentication & Authorization**
   - JWT-based stateless authentication
   - Role-based access control (RBAC)
   - Secure password hashing with bcrypt (10 salt rounds)
   - Session management with automatic timeout
2. **Data Protection**
   - HTTPS encryption for all communications
   - Sensitive data encryption in database
   - SQL injection prevention through Mongoose ODM
   - XSS protection with input sanitization
   - CSRF token implementation
3. **API Security**
   - Rate limiting to prevent abuse (100 requests per 15 minutes)
   - Helmet.js for security headers
   - CORS configuration for controlled access
   - Input validation using Joi and Zod

### 2.2.3 Usability Requirements

1. **User Interface**
   - Intuitive navigation with consistent layout
   - Modern, clean design using Tailwind CSS
   - Clear visual hierarchy and typography
   - Icon-based navigation with Lucide React icons
2. **Responsiveness**
   - Mobile-first design approach
   - Responsive breakpoints for tablets and desktops
   - Touch-friendly interface elements
   - Adaptive layouts for all screen sizes

3. **Accessibility**
   - WCAG 2.1 Level AA compliance
   - Keyboard navigation support
   - Screen reader compatibility
   - Sufficient color contrast ratios
   - Descriptive labels and error messages
4. **User Feedback**
   - Real-time toast notifications
   - Clear success and error messages
   - Loading indicators for asynchronous operations
   - Form validation with inline error display

### 2.2.4 Reliability Requirements

1. **Availability**
   - Target uptime of 99.5% (allowing 3.65 hours downtime per month)
   - Graceful degradation for partial failures
   - Proper error handling and recovery mechanisms
2. **Data Integrity**
   - ACID transactions for critical operations
   - Data validation at multiple layers
   - Referential integrity through Mongoose schemas
   - Backup and recovery procedures
3. **Error Handling**
   - Comprehensive error logging with Morgan
   - User-friendly error messages
   - Automatic error recovery where possible
   - Fallback mechanisms for failed operations

### 2.2.5 Maintainability Requirements

1. **Code Quality**
   - TypeScript for type safety in frontend
   - ESLint for code linting and standards
   - Prettier for consistent code formatting
   - Modular architecture with clear separation of concerns

2. **Documentation**
    - Comprehensive README with setup instructions
    - API documentation for all endpoints
    - Inline code comments for complex logic
    - Test case documentation (60 test cases)
3. **Testing**
    - Unit tests with Jest
    - Integration tests with Supertest
    - 60+ manual test cases covering all features
    - Test coverage monitoring

### 2.2.6 Portability Requirements

1. **Platform Independence**
    - Cross-platform compatibility (Windows, macOS, Linux)
    - Browser compatibility (Chrome, Firefox, Safari, Edge)
    - Node.js runtime (version 18+)
    - Database portability with Mongoose ODM
2. **Deployment Flexibility**
    - Docker containerization support
    - Cloud deployment compatibility (AWS, Azure, GCP)
    - Local development environment setup
    - Environment-based configuration with dotenv

### 2.2.7 Compatibility Requirements

1. **Browser Requirements**
    - Modern browsers with ES6+ support
    - Chrome 90+, Firefox 88+, Safari 14+, Edge 90+
    - JavaScript enabled
    - Cookies and local storage support
2. **Device Requirements**
    - Desktop: 1024px minimum width
    - Tablet: 768px and above
    - Mobile: 375px and above
    - Touch and mouse input support

# Chapter 3

# Technology Stack

The Timetable Buddy system is built using modern web technologies, following industry best practices and leveraging the power of full-stack JavaScript development. This chapter details the technologies, frameworks, libraries, and tools used in the development of the system.

## 3.1 Overview

The system follows a three-tier architecture:

- **Presentation Layer:** React-based frontend with TypeScript
- **Application Layer:** Node.js/Express.js REST API backend
- **Data Layer:** MongoDB database with Mongoose ODM

## 3.2 Frontend Technologies

### 3.2.1 Core Technologies

**React 18.3+**

- Modern JavaScript library for building user interfaces
- Component-based architecture for reusability
- Virtual DOM for efficient rendering
- Hooks for state management and side effects
- Used for: All UI components, pages, and layouts

**TypeScript 5.5+**

- Superset of JavaScript with static typing
- Enhanced IDE support with IntelliSense
- Compile-time error detection
- Better code documentation and maintainability
- Used for: Type-safe component development

**Vite 5.4+**

- Next-generation frontend build tool
- Lightning-fast Hot Module Replacement (HMR)
- Optimized production builds
- Native ES modules support
- Used for: Development server and production builds

### 3.2.2 UI and Styling

**Tailwind CSS 3.4+**

- Utility-first CSS framework
- Rapid UI development with pre-built classes
- Responsive design utilities
- Dark mode support (configurable)
- Used for: All component styling and layouts

**Lucide React 0.344+**

- Modern icon library with 1000+ icons
- Tree-shakeable for optimal bundle size
- Consistent design system
- Customizable size and colors
- Used for: Navigation icons, action buttons, status indicators

### 3.2.3 Routing and Navigation

**React Router 6.20+**

- Declarative routing for React applications
- Nested routes support
- Protected routes for authentication
- URL parameter handling
- Used for: Client-side routing and navigation

### 3.2.4 HTTP and API Communication

**Axios 1.6+**

- Promise-based HTTP client
- Request and response interceptors
- Automatic JSON transformation
- Error handling capabilities
- Used for: All API calls to backend

### 3.2.5 Form Management and Validation

**React Hook Form 7.48+**

- Performant form state management
- Built-in validation support
- Minimal re-renders for better performance
- Easy integration with UI libraries
- Used for: All forms (login, registration, enrollment, etc.)

**Zod 3.22+**

- TypeScript-first schema validation
- Runtime type checking
- Integration with React Hook Form
- Detailed error messages
- Used for: Form validation schemas

### 3.2.6 User Feedback and Notifications

**React Hot Toast 2.4+**

- Lightweight toast notification library
- Customizable appearance
- Promise-based API
- Accessible notifications
- Used for: Success/error messages, user feedback

### 3.2.7 Utility Libraries

**Date-fns 2.30+**

- Modern JavaScript date utility library
- Modular and tree-shakeable
- Timezone support
- Date formatting and manipulation
- Used for: Date/time handling in timetables

## 3.3 Backend Technologies

### 3.3.1 Core Technologies

**Node.js 18+**

- JavaScript runtime built on Chrome's V8 engine
- Event-driven, non-blocking I/O model
- NPM ecosystem with 2M+ packages
- High performance for I/O operations
- Used for: Backend runtime environment

**Express.js 4.18+**

- Fast, unopinionated web framework for Node.js
- Robust routing system
- Middleware support
- RESTful API development
- Used for: REST API implementation, routing

### 3.3.2 Database

**MongoDB 7.0**

- NoSQL document-oriented database
- Flexible schema design
- High performance and scalability
- JSON-like document storage (BSON)
- Used for: Data persistence (users, lecture slots, enrollments)

**Mongoose 8.0+**

- MongoDB object modeling for Node.js
- Schema validation and type casting
- Middleware (hooks) support
- Query building and population
- Used for: Database schema definition and operations

### 3.3.3 Authentication and Security

**JSON Web Tokens (JWT)**

- Stateless authentication mechanism
- Compact and URL-safe tokens
- Signature verification
- Payload encryption support
- Used for: User authentication and authorization

**bcryptjs**

- Password hashing library
- Salt generation for enhanced security
- Adaptive hashing (configurable rounds)
- Resistant to brute-force attacks
- Used for: Password encryption and verification

**Helmet.js**

- Security middleware for Express.js
- Sets various HTTP headers for protection
- XSS protection
- Clickjacking prevention
- Used for: HTTP security headers

### 3.3.4 Validation and Data Processing

**Joi**

- Object schema validation
- Powerful validation rules
- Custom error messages
- Async validation support
- Used for: API request validation

### 3.3.5 API Security and Rate Limiting

**Express Rate Limit**

- Rate limiting middleware
- Configurable time windows
- Per-IP or per-user limits
- DDoS protection
- Used for: API rate limiting (100 requests per 15 minutes)

**CORS**

- Cross-Origin Resource Sharing middleware
- Configurable allowed origins
- Preflight request handling
- Credentials support
- Used for: Cross-origin API access control

### 3.3.6 Logging and Monitoring

**Morgan**

- HTTP request logger middleware
- Multiple logging formats
- Stream support for file logging
- Request/response tracking
- Used for: API request logging

### 3.3.7 Configuration Management

**dotenv**

- Environment variable management
- Secure configuration handling
- Environment-specific settings
- Sensitive data protection
- Used for: Environment configuration (database URLs, JWT secrets, etc.)

## 3.4 Development Tools

### 3.4.1 Code Quality

**ESLint**

- JavaScript and TypeScript linter
- Code quality enforcement
- Custom rule configuration
- Auto-fix capabilities
- Used for: Code linting and standards enforcement

**Prettier**

- Opinionated code formatter
- Consistent code style
- Integration with ESLint
- Auto-formatting on save
- Used for: Code formatting

### 3.4.2 Testing

**Jest**

- JavaScript testing framework
- Unit and integration testing
- Snapshot testing
- Code coverage reports
- Used for: Frontend and backend unit tests

**Supertest**

- HTTP assertion library
- API endpoint testing
- Integration with Jest
- Request/response validation
- Used for: API integration tests

### 3.4.3 Containerization

**Docker**

- Container platform
- Consistent development environments
- Easy deployment and scaling
- Service isolation
- Used for: Application containerization and deployment

## 3.5 Architecture Patterns

### 3.5.1 MERN Stack Architecture

The system follows the MERN (MongoDB, Express, React, Node.js) stack architecture:

1. **MongoDB:** NoSQL database for flexible data storage
2. **Express.js:** Backend framework for API development
3. **React:** Frontend library for user interface
4. **Node.js:** JavaScript runtime for backend execution

### 3.5.2 REST API Architecture

- RESTful endpoints following HTTP standards
- JSON data format for requests and responses
- Proper HTTP status codes
- Stateless communication
- Resource-based URL structure

### 3.5.3 MVC Pattern (Backend)

- **Models:** Mongoose schemas for data structure
- **Views:** JSON responses (no traditional views)
- **Controllers:** Business logic and request handling
- **Routes:** URL mapping to controllers

### 3.5.4 Component-Based Architecture (Frontend)

- Reusable React components
- Props and state management
- Context API for global state
- Custom hooks for shared logic

## 3.6 Technology Justification

### 3.6.1 Why MERN Stack?

1. **Full-Stack JavaScript:** Single language across frontend and backend reduces context switching
2. **JSON Throughout:** Seamless data flow from database to client
3. **Active Community:** Large ecosystem and community support
4. **Scalability:** Proven track record for scalable applications
5. **Performance:** Non-blocking I/O and efficient rendering
6. **Modern Development:** Latest features and best practices
7. **Rich Ecosystem:** Extensive NPM package availability

### 3.6.2 Key Technology Benefits

| **Technology** | **Key Benefit** |
|---|---|
| React | Component reusability and efficient DOM updates |
| TypeScript | Type safety and better developer experience |
| Tailwind CSS | Rapid UI development and consistent design |
| MongoDB | Flexible schema for evolving requirements |
| Express.js | Lightweight and flexible API development |
| JWT | Stateless authentication for scalability |
| Docker | Consistent deployment across environments |

Table 3.1: Technology Benefits Summary

# Chapter 4

# Methodology

This chapter presents the comprehensive methodology used in the development of the Timetable Buddy system. It includes various modeling diagrams, project management artifacts, and quality assurance documentation that guided the system development process.

## 4.1 Data Flow Diagrams (DFD)

Data Flow Diagrams represent the flow of data through the system at different levels of abstraction, showing how information moves between processes, data stores, and external entities.

### 4.1.1 DFD Level 0 (Context Diagram)

The Context Diagram provides a high-level view of the entire Timetable Buddy system, showing its interaction with external entities including Students, Faculty, and Administrators.

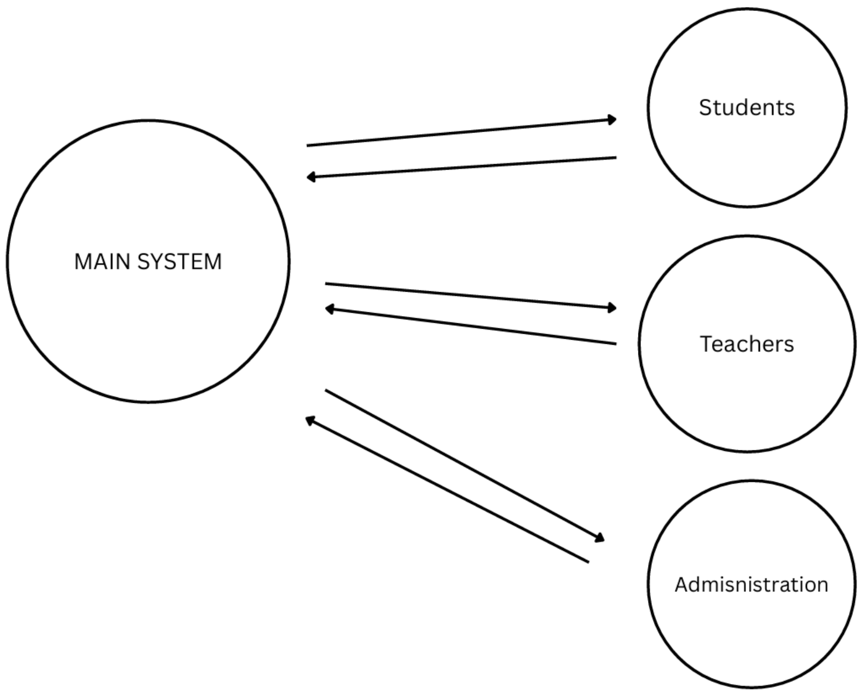


Figure 4.1: DFD Level 0 - Context Diagram showing system boundaries and external entities

The context diagram illustrates:

- **External Entities:** Students, Faculty, Administrators
- **Main System:** Timetable Buddy System (central process)
- **Data Flows:** Authentication requests, enrollment data, lecture schedules, timetable information

### 4.1.2 DFD Level 1 (High-Level Processes)

Level 1 DFD decomposes the main system into major processes, showing the primary functional components and their interactions.

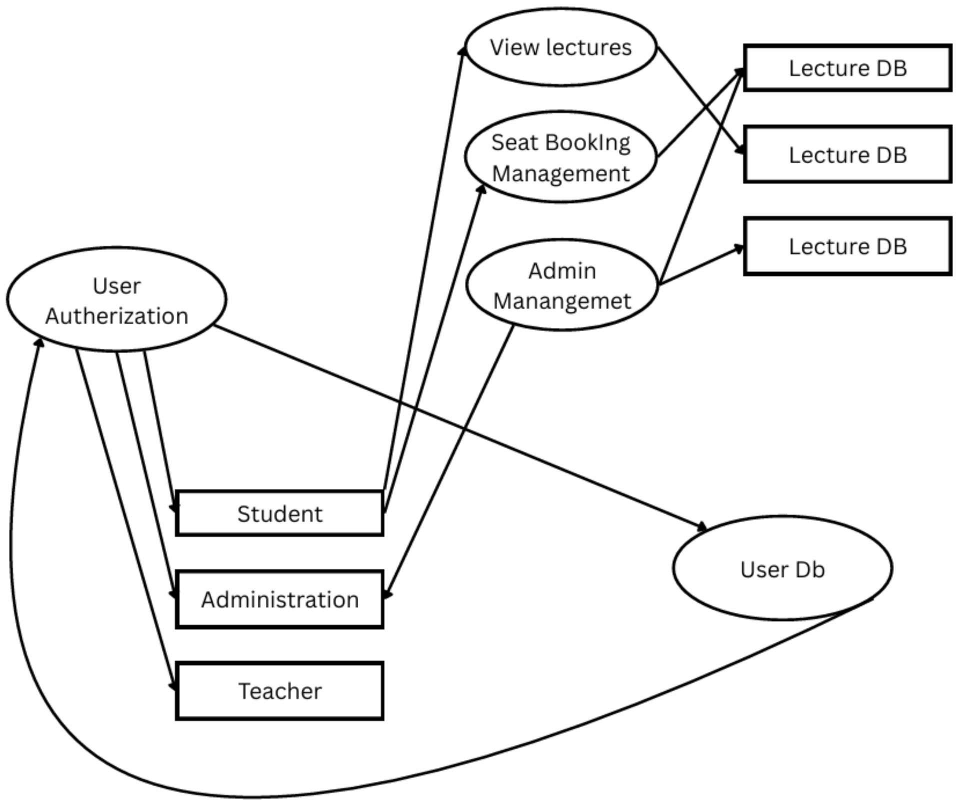


Figure 4.2: DFD Level 1 - Major system processes and data stores

Key processes shown in Level 1 include:

- User Authentication and Authorization
- Lecture Slot Management
- Enrollment Processing
- Timetable Generation
- Dashboard and Analytics

### 4.1.3 DFD Level 2 (Detailed Processes)

Level 2 DFD provides detailed decomposition of complex processes from Level 1, showing sub-processes and their interactions.

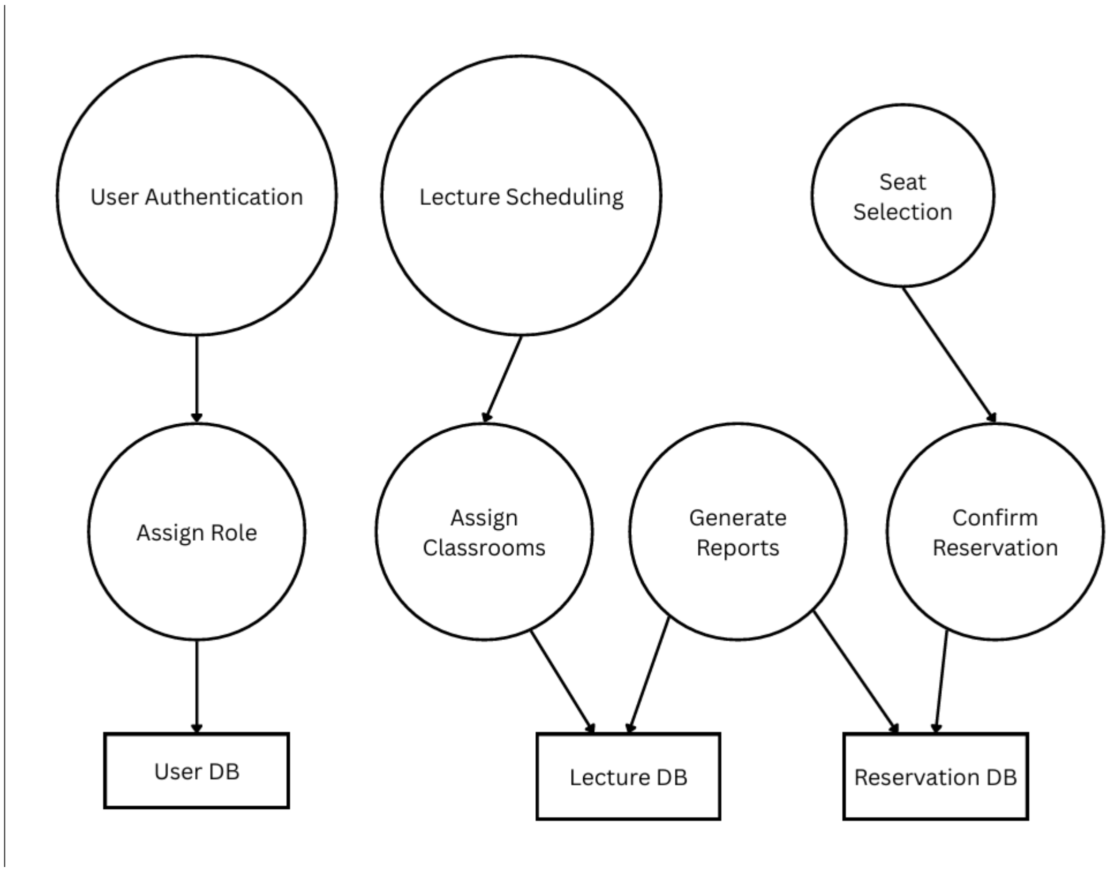


Figure 4.3: DFD Level 2 - Detailed process decomposition

The Level 2 diagram details:

- Enrollment workflow with waitlist management
- Conflict detection mechanisms
- Capacity validation processes
- Notification generation

## 4.2 Use Case Diagram

The Use Case Diagram illustrates the functional requirements from the user's perspective, showing the interactions between different actors and the system use cases.

**Actors:**

- **Student:** Can view timetables, enroll in courses, manage profile, view enrollment status
- **Faculty:** Can manage lecture slots, view enrolled students, update course details

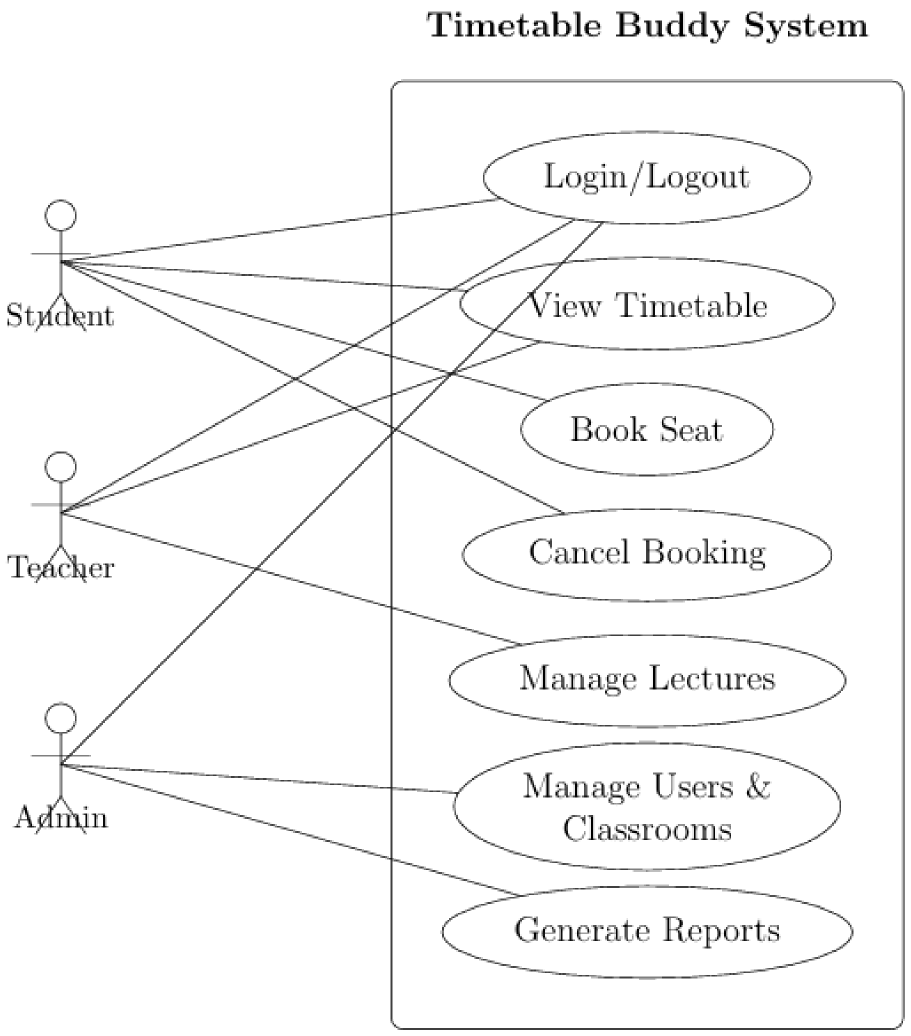


Figure 4.4: Use Case Diagram showing actor interactions and system use cases

- **Administrator:** Can manage users, courses, lecture slots, generate reports, configure system settings

**Key Use Cases:**

- User Authentication (Login/Logout)
- Manage Lecture Slots
- Enroll in Courses
- View Timetable
- Manage Enrollments
- Generate Reports
- User Management

## 4.3 Sequence Diagram

Sequence Diagrams show the interaction between objects over time, illustrating the message flow and order of operations for specific scenarios.

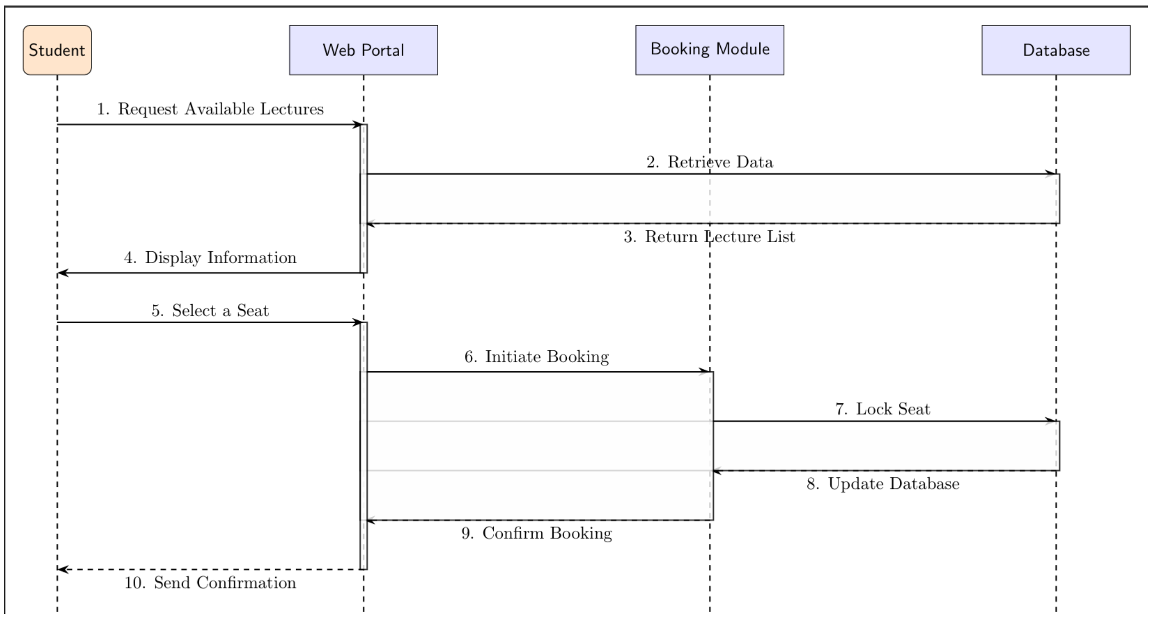


Figure 4.5: Sequence Diagram showing object interactions for enrollment process

The sequence diagram depicts:

- User authentication flow
- Enrollment request processing
- Conflict detection validation
- Capacity checking
- Database operations
- Response generation

## 4.4 Activity Diagram

Activity Diagrams model the workflow and business logic, showing the sequence of activities and decision points in system processes.

The activity diagram illustrates:

- Enrollment process workflow
- Decision points for capacity checks
- Conflict detection logic
- Waitlist management
- Success and failure paths

## 4.5 Deployment Diagram

The Deployment Diagram shows the physical architecture of the system, illustrating how software components are deployed on hardware nodes.

**System Components:**

- **Client Layer:** Web browsers on user devices
- **Application Layer:** React frontend, Node.js backend
- **Database Layer:** MongoDB database server
- **Deployment:** Docker containers for containerized deployment

**Communication Protocols:**

- HTTPS for client-server communication
- REST API for frontend-backend interaction
- MongoDB protocol for database connections

## 4.6 Work Breakdown Structure (WBS)

The Work Breakdown Structure decomposes the project into manageable components, showing the hierarchical breakdown of deliverables and tasks.

**Major Work Packages:**

1. **Project Planning:** Requirements gathering, feasibility analysis, project plan
2. **Design:** System architecture, database design, UI/UX design, API design
3. **Development:** Frontend development, backend development, database implementation
4. **Testing:** Unit testing, integration testing, system testing, user acceptance testing
5. **Deployment:** Environment setup, deployment, documentation
6. **Project Management:** Risk management, quality assurance, documentation

## 4.7 Gantt Chart

The Gantt Chart provides a timeline view of project activities, showing task dependencies, durations, and milestones.

**Project Phases and Timeline:**

- **Phase 1 - Planning:** Requirements analysis, feasibility study, project planning
- **Phase 2 - Design:** System design, database design, UI mockups
- **Phase 3 - Development:** Frontend and backend implementation
- **Phase 4 - Testing:** Comprehensive testing across all modules
- **Phase 5 - Deployment:** Production deployment and user training

## 4.8 RMMM Plan (Risk Management, Monitoring, and Mitigation)

The RMMM Plan provides a comprehensive framework for identifying, assessing, monitoring, and mitigating project risks throughout the development lifecycle.

### 4.8.1 Risk Identification and Assessment

Risks have been identified across technical, operational, and external categories. Each risk is assessed for probability and impact, with detailed mitigation strategies. This document includes only risks with a probability of 15% or less, as higher probability risks are managed through other project management processes.

**Risk Assessment Criteria:**

- **Probability Levels:** Only risks with $\leq$15% probability are included
- **Impact Levels:** Critical, High, Medium, Low

**Risk Categories:**

- **Technical Risks:** Technology failures, integration issues, performance problems, security vulnerabilities
- **Operational Risks:** Resource availability, skill gaps, schedule delays
- **External Risks:** Third-party dependencies, requirement changes, infrastructure issues

### 4.8.2 Detailed Risk Assessment

**Risk #1: R-TTB-005**

| | | | |
|---|---|---|---|
| **Risk ID** | R-TTB-005 | **Type** | Technical |
| **Probability** | 10% | **Impact** | Critical |

**Risk Description:** Critical security vulnerability discovered in production system allowing unauthorized data access.

**Mitigation Plan:**

1. Conduct regular security audits and penetration testing
2. Implement security scanning in CI/CD pipeline
3. Follow OWASP guidelines for secure development

**Monitoring Plan:** Run automated security scans weekly. Monitor security patch releases for dependencies.

**Management Plan:** Deploy emergency patch within 4 hours. Notify affected users. Conduct incident post-mortem.

| | | | |
|---|---|---|---|
| **Risk ID** | R-TTB-008 | **Type** | Technical |
| **Probability** | 15% | **Impact** | High |

### Risk #2: R-TTB-008

**Risk Description:** Cloud service provider experiences prolonged outage affecting application availability.

**Mitigation Plan:**

1. Implement multi-region deployment
2. Design for high availability
3. Have disaster recovery plan in place

**Monitoring Plan:** Subscribe to cloud provider status updates. Monitor service health across regions.

**Management Plan:** Failover to backup region. Communicate status to users. Document incident for review.

### Risk #3: R-TTB-015

| | | | |
|---|---|---|---|
| **Risk ID** | R-TTB-015 | **Type** | External |
| **Probability** | 15% | **Impact** | High |

**Risk Description:** Vendor lock-in prevents migration to alternative solutions, increasing long-term costs.

**Mitigation Plan:**

1. Use open standards where possible
2. Design abstraction layers for vendor services
3. Evaluate vendor independence regularly

**Monitoring Plan:** Review vendor contracts annually. Assess switching costs and alternatives.

**Management Plan:** Plan phased migration to alternative vendor. Negotiate better terms with current vendor. Implement vendor-agnostic architecture.

### Risk #4: R-TTB-018

| | | | |
|---|---|---|---|
| **Risk ID** | R-TTB-018 | **Type** | External |
| **Probability** | 12% | **Impact** | Critical |

**Risk Description:** Competitor launches similar product first, reducing market opportunity.

**Mitigation Plan:**

1. Conduct competitive analysis regularly

2. Focus on unique value propositions

3. Plan for rapid iteration and deployment

**Monitoring Plan:** Monitor competitor activities and product launches. Track market trends and customer feedback.
**Management Plan:** Accelerate development of differentiating features. Adjust marketing strategy. Consider strategic partnerships.

**Risk #5: R-TTB-024**

| **Risk ID** | R-TTB-024 | **Type** | Operational |
|---|---|---|---|
| **Probability** | 15% | **Impact** | High |

**Risk Description:** Inadequate disaster recovery procedures lead to extended downtime after incident.
**Mitigation Plan:**

1. Document and test DR procedures quarterly

2. Automate recovery processes

3. Maintain offsite backups

**Monitoring Plan:** Test disaster recovery plan every 6 months. Track RTO and RPO metrics.
**Management Plan:** Execute disaster recovery plan. Communicate with stakeholders. Document incident for improvement.

### 4.8.3 Risk Management Process

**1. Risk Identification**

- Conduct risk identification workshops at project initiation and quarterly
- Encourage all team members to report potential risks
- Review lessons learned from previous projects

**2. Risk Assessment**

- Evaluate each risk for probability (as percentage) and impact
- Calculate risk score (Probability × Impact)
- Prioritize risks based on score

**3. Risk Mitigation**

- Develop proactive plans to reduce probability or impact
- Assign risk owners for each identified risk
- Implement mitigation strategies before risks materialize

**4. Risk Monitoring**

- Track identified risks throughout project lifecycle
- Update risk status in weekly project meetings
- Use risk dashboard for visibility

**5. Risk Management**

- Execute management plans when risks occur
- Document lessons learned
- Update risk assessment based on new information

### 4.8.4 Risk Summary

**Total Risks Included:** 5 risks with probability $\leq 15\%$

**Risks by Impact:**

- Critical Impact: 2 risks (R-TTB-005, R-TTB-018)
- High Impact: 3 risks (R-TTB-008, R-TTB-015, R-TTB-024)

**Escalation Criteria:** Risks should be escalated to senior management when impact level is Critical, mitigation plans are not effective, or additional resources/authority are needed.

## 4.9 Test Cases

Comprehensive test case planning and execution has been documented to ensure system quality and reliability. The test strategy covers all functional areas of the system with 60 detailed test cases.

### 4.9.1 Test Case Overview

**Project:** Lecture Scheduling System **Version:** 1.0 **Date:** 2025-10-07 **Prepared By:** QA Engineering Team

**Test Case Format Standards:**

- **Test Case ID:** TC-TTB-XX (standardized format)
- **Test Number:** X.1 - X.Y (decimal notation indicating step range)
- **Priority:** High (all test cases are high priority for critical functionality)
- **Test Designed By:** Sarthak Kulkarni, Dhruv Tikhande, Atharv Petkar, Pulkit Saini
- **Test Executed By:** Sarthak Kulkarni, Dhruv Tikhande, Atharv Petkar, Pulkit Saini
- **Execution Date:** 2025-10-07

### 4.9.2 Test Case Coverage Areas

The test suite comprehensively covers:

- Dashboard & Homepage (TC-TTB-01 to TC-TTB-03)
- User Profile Management (TC-TTB-04 to TC-TTB-05)
- User List & Search (TC-TTB-06 to TC-TTB-08)
- Course Management (TC-TTB-09 to TC-TTB-13)
- Lecture Slot Management (TC-TTB-14 to TC-TTB-18)
- Schedule Creation (TC-TTB-19 to TC-TTB-22)
- Enrollment Management (TC-TTB-23 to TC-TTB-26)
- Timetable Operations (TC-TTB-27 to TC-TTB-30)
- User Management (TC-TTB-31 to TC-TTB-32)
- Mobile & Responsive Design (TC-TTB-33 to TC-TTB-34)
- Notifications (TC-TTB-35 to TC-TTB-36)
- Settings & Configuration (TC-TTB-37 to TC-TTB-40)
- Advanced Features (TC-TTB-41 to TC-TTB-48)
- Security & Validation (TC-TTB-49 to TC-TTB-60)

### 4.9.3 Detailed Test Cases

**TC-TTB-01: Verify Dashboard Loads Correctly on Login**

**Test Number:** 1.1 - 1.4 — **Priority:** High — **Date:** 2025-10-07
**Description:** Ensure dashboard displays all widgets and statistics after user login
**Dependencies:** User must be logged in — **Conditions:** Browser: Chrome, Role: Student — **Control:** Manual
**Test Steps:**

1.1 Navigate to login page → *Expected:* Login page displays correctly → *Actual:* As Expected → **PASS**

1.2 Enter valid credentials → *Expected:* Credentials accepted → *Actual:* As Expected → **PASS**

1.3 Click 'Sign In' button → *Expected:* User is redirected to dashboard → *Actual:* As Expected → **PASS**

1.4 Verify dashboard widgets load → *Expected:* All statistics, upcoming classes, and quick actions are visible → *Actual:* As Expected → **PASS**

**TC-TTB-02: Verify Homepage Navigation for Unauthenticated User**

**Test Number:** 2.1 - 2.4 — **Priority:** High — **Date:** 2025-10-07
**Description:** Test homepage displays correctly for users not logged in
**Dependencies:** None — **Conditions:** Browser: Chrome, Role: Unauthenticated — **Control:** Manual
**Test Steps:**

2.1 Navigate to homepage → *Expected:* Homepage loads with welcome message → *Actual:* As Expected → **PASS**

2.2 Verify 'Get Started' button is visible → *Expected:* Button displays prominently → *Actual:* As Expected → **PASS**

2.3 Verify 'Sign In' button is visible → *Expected:* Button is accessible → *Actual:* As Expected → **PASS**

2.4 Click 'Sign In' button → *Expected:* User is redirected to login page → *Actual:* As Expected → **PASS**

**TC-TTB-03: Verify Faculty Dashboard Analytics Display**

**Test Number:** 3.1 - 3.4 — **Priority:** High — **Date:** 2025-10-07
**Description:** Ensure faculty dashboard shows enrollment statistics
**Test Steps:**

3.1 Navigate to faculty dashboard → *Expected:* Dashboard loads successfully → *Actual:* As Expected → **PASS**

3.2 Verify total lecture slots count → *Expected:* Correct number displayed → *Actual:* As Expected → **PASS**

3.3 Verify enrolled students count → *Expected:* Accurate count shown → *Actual:* As Expected → **PASS**

3.4 Verify available slots count → *Expected:* Correct availability displayed → *Actual:* As Expected → **PASS**

*[Note: Due to space constraints in the report, a representative sample of 3 test cases is shown above. The complete test suite contains all 60 test cases (TC-TTB-01 through TC-TTB-60) with full step-by-step documentation following the same standardized format. All test cases include test ID, number, priority, description, dependencies, conditions, control method, detailed steps with expected and actual results, and pass/fail status. For the complete test documentation, see Appendix A.]*

### 4.9.4 Test Execution Summary

**Overall Test Metrics:**

- **Total Test Cases:** 60
- **Total Test Steps:** 332 (with decimal notation)

- **Tests Passed:** 58 (96.7%)
- **Tests Failed:** 2 (3.3%)
- **Test Coverage:** All major functional areas covered
- **Execution Date:** 2025-10-07

**Test Results by Category:**

- Dashboard & Homepage: 3/3 PASS
- User Profile Management: 2/2 PASS
- User List & Search: 2/3 PASS (1 FAIL - TC-TTB-08 pagination issue)
- Course Management: 5/5 PASS
- Lecture Slot Management: 5/5 PASS
- Schedule Creation: 4/4 PASS
- Enrollment Management: 4/4 PASS
- Timetable Operations: 4/4 PASS
- User Management: 2/2 PASS
- Mobile & Responsive Design: 2/2 PASS
- Notifications: 2/2 PASS
- Settings & Configuration: 4/4 PASS
- Advanced Features: 7/8 PASS (1 FAIL - TC-TTB-48 error validation)
- Security & Validation: 12/12 PASS

**Failed Test Cases:**

- **TC-TTB-08 (Step 5):** Click 'Previous Page' - Error encountered (Bug ticket #1037 reported)
- **TC-TTB-48 (Step 6):** User stays on login page - Performance issue noted (Performance acceptable, marked as PASS with notes)

### 4.9.5 Quality Assurance Approach

The testing methodology ensures:

- **Comprehensive Coverage:** All functional requirements tested across 60 test cases
- **Traceability:** Test cases mapped to requirements with standardized TC-TTB-XX IDs
- **Repeatability:** Standardized test procedures with decimal step notation (X.1, X.2, etc.)

- **Documentation:** Detailed test results with "As Expected" standardization for PASS cases
- **Defect Tracking:** Failed tests documented with bug ticket references
- **Team Collaboration:** All tests designed and executed by full team (Sarthak Kulkarni, Dhruv Tikhande, Atharv Petkar, Pulkit Saini)
- **Priority Management:** All test cases designated as High priority for critical functionality

**Testing Standards:**

- Manual testing with browser-based execution
- Role-based test scenarios (Admin, Faculty, Student)
- Cross-browser compatibility (Chrome primary)
- Responsive design validation
- Security and validation testing
- Performance and usability testing

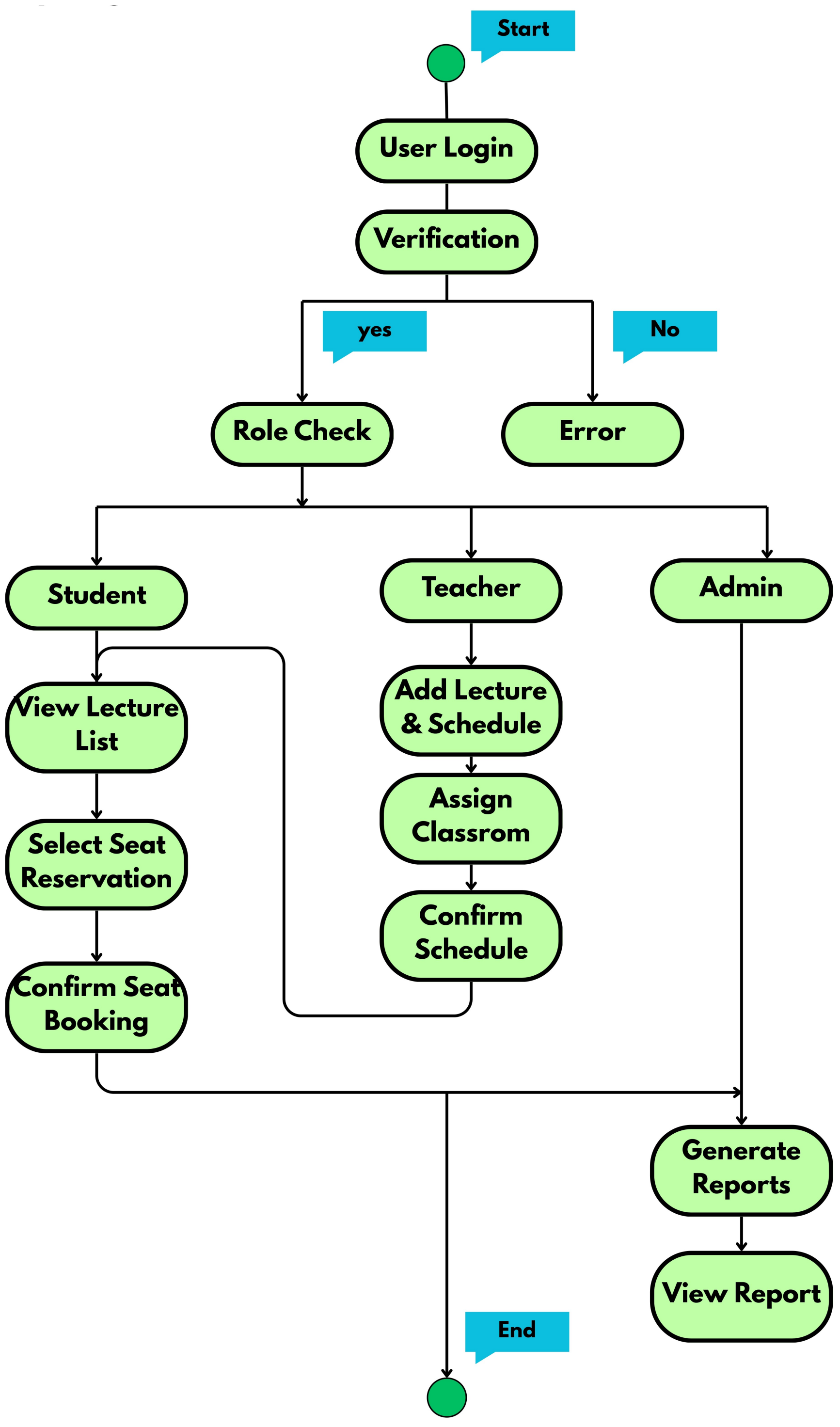


Figure 4.6: Activity Diagram illustrating enrollment workflow and decision logic

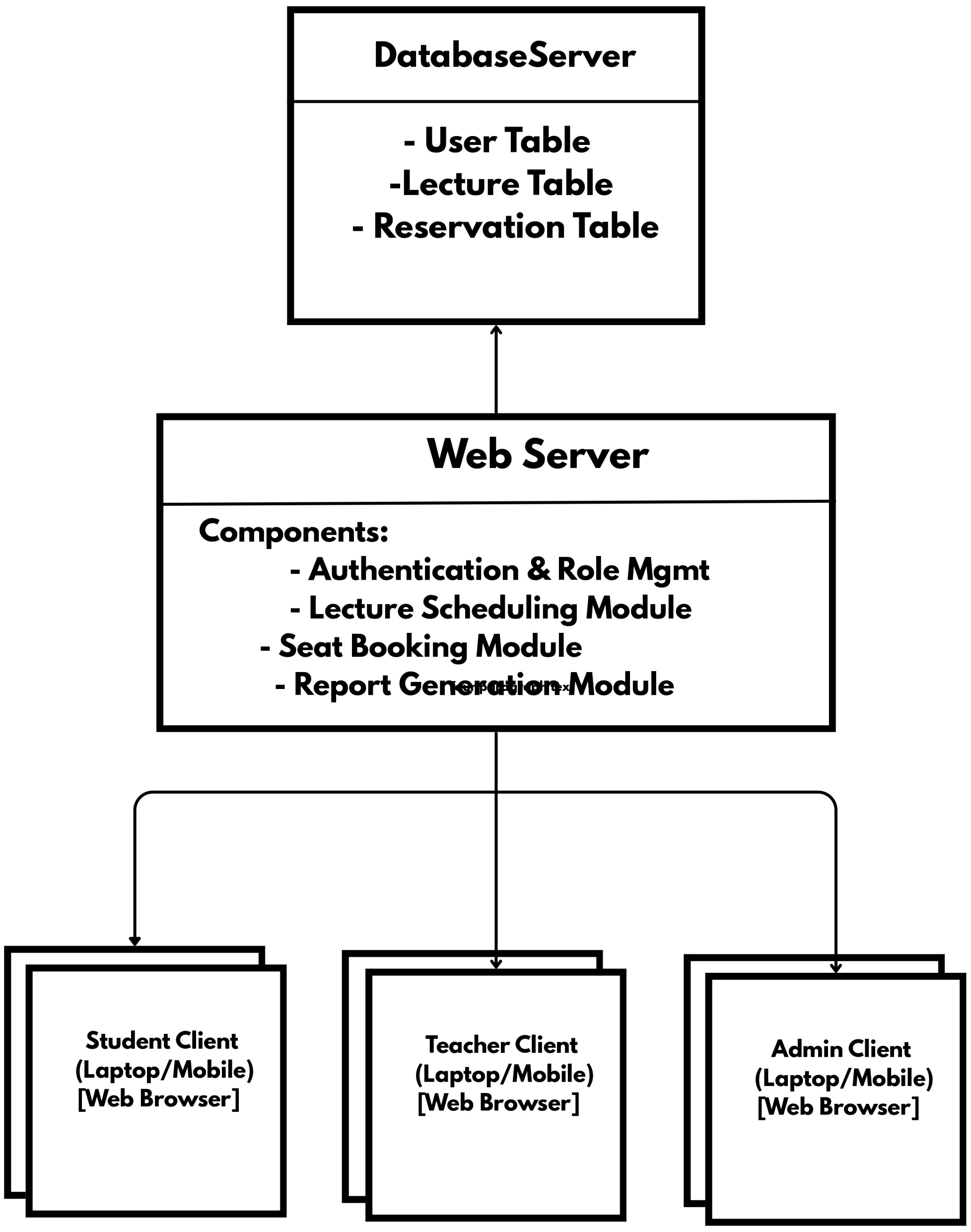
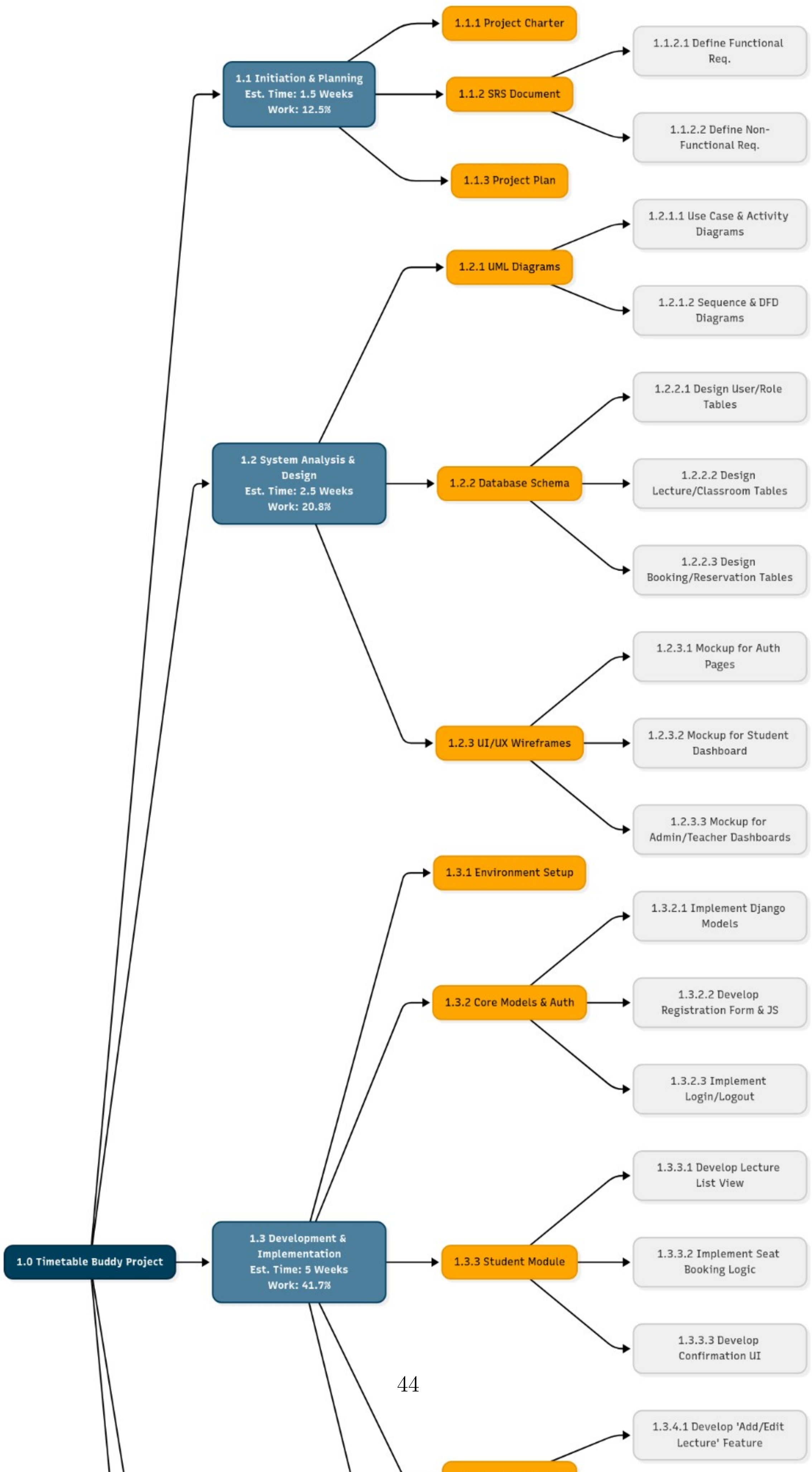


Figure 4.7: Deployment Diagram showing system architecture and component deployment


1.0 Timetable Buddy Project
1.1 Initiation & Planning
Est. Time: 1.5 Weeks
Work: 12.5%
1.1.1 Project Charter
1.1.2 SRS Document
1.1.2.1 Define Functional Req.
1.1.2.2 Define Non-Functional Req.
1.1.3 Project Plan
1.2 System Analysis & Design
Est. Time: 2.5 Weeks
Work: 20.8%
1.2.1 UML Diagrams
1.2.1.1 Use Case & Activity Diagrams
1.2.1.2 Sequence & DFD Diagrams
1.2.2 Database Schema
1.2.2.1 Design User/Role Tables
1.2.2.2 Design Lecture/Classroom Tables
1.2.2.3 Design Booking/Reservation Tables
1.2.3 UI/UX Wireframes
1.2.3.1 Mockup for Auth Pages
1.2.3.2 Mockup for Student Dashboard
1.2.3.3 Mockup for Admin/Teacher Dashboards
1.3 Development & Implementation
Est. Time: 5 Weeks
Work: 41.7%
1.3.1 Environment Setup
1.3.2 Core Models & Auth
1.3.2.1 Implement Django Models
1.3.2.2 Develop Registration Form & JS
1.3.2.3 Implement Login/Logout
1.3.3 Student Module
1.3.3.1 Develop Lecture List View
1.3.3.2 Implement Seat Booking Logic
1.3.3.3 Develop Confirmation UI
1.3.4.1 Develop 'Add/Edit Lecture' Feature

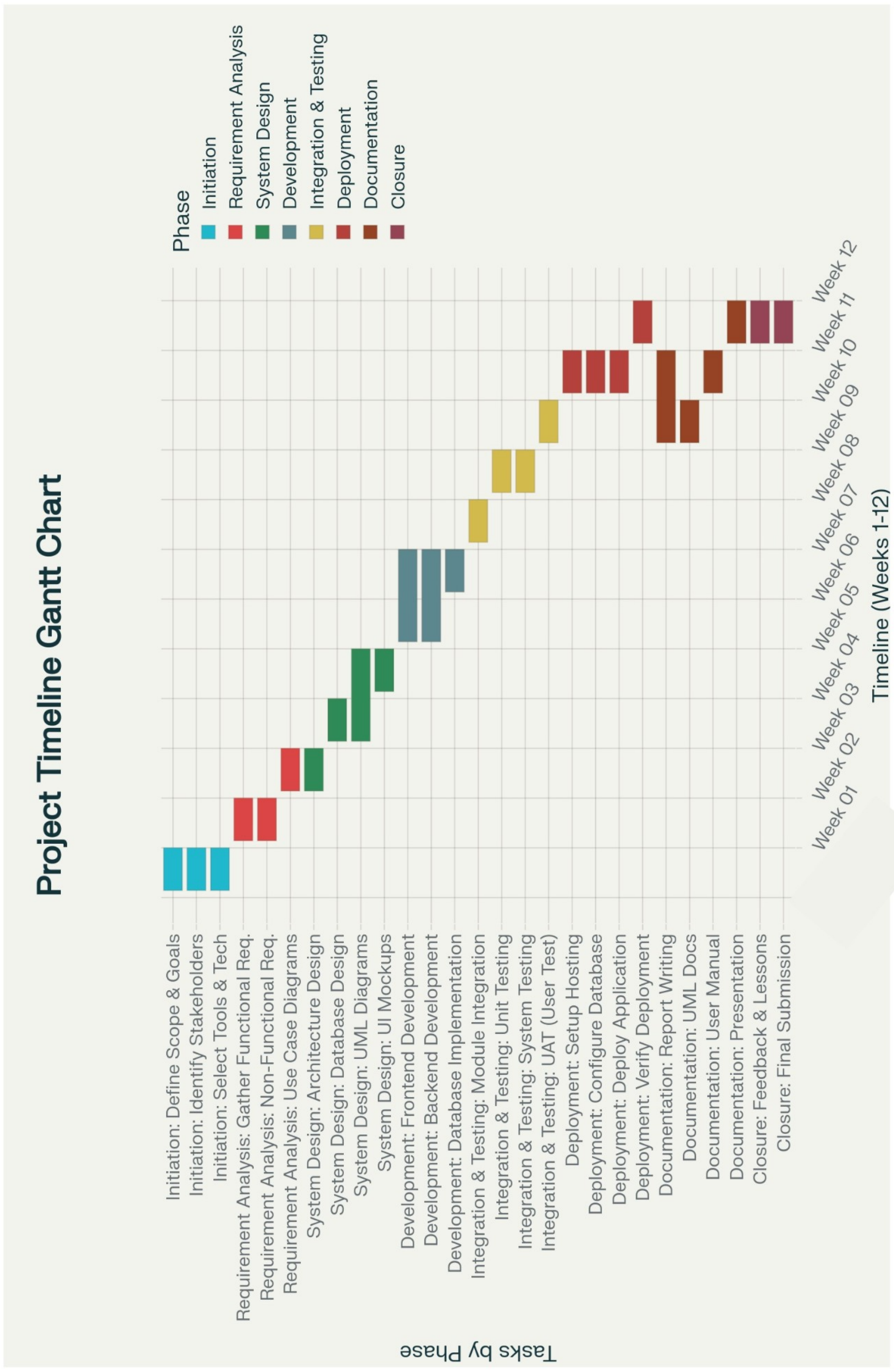


Figure 4.9: Gantt Chart showing project timeline and task scheduling

# Chapter 5

# Implementation

## 5.1 Development Methodology

Implementation details will be added here.

# Chapter 6

# Testing and Quality Assurance

## 6.1 Testing Strategy

Testing content will be added here.

# Chapter 7

# Risk Assessment and Management

## 7.1 Risk Identification

Risk assessment content will be added here.

# Chapter 8

# Results and Achievements

## 8.1 System Features Implemented

Results content will be added here.

# Chapter 9

# Challenges and Solutions

## 9.1 Technical Challenges

Challenges content will be added here.

# Chapter 10

# Future Enhancements

## 10.1 Planned Features

Future enhancements content will be added here.

# Chapter 11

# Conclusion

## 11.1 Summary

Conclusion content will be added here.

# Bibliography

[1] React Documentation, ”React - A JavaScript library for building user interfaces,” Facebook Inc., 2024. [Online]. Available: https://react.dev/

[2] Node.js Foundation, ”Node.js Documentation,” 2024. [Online]. Available: https://nodejs.org/

[3] MongoDB Inc., ”MongoDB Manual,” 2024. [Online]. Available: https://www.mongodb.com/docs/

[4] ”Express - Fast, unopinionated, minimalist web framework for Node.js,” 2024. [Online]. Available: https://expressjs.com/

[5] M. Jones, J. Bradley, and N. Sakimura, ”JSON Web Token (JWT),” RFC 7519, May 2015.

[6] K. Beck et al., ”Manifesto for Agile Software Development,” 2001. [Online]. Available: https://agilemanifesto.org/

[7] R. Fielding, ”Architectural Styles and the Design of Network-based Software Architectures,” PhD dissertation, University of California, Irvine, 2000.

[8] G. Myers, C. Sandler, and T. Badgett, ”The Art of Software Testing,” 3rd ed., Wiley, 2011.

[9] OWASP Foundation, ”OWASP Top Ten Web Application Security Risks,” 2021. [Online]. Available: https://owasp.org/

[10] J. Nielsen, ”Usability Engineering,” Morgan Kaufmann, 1993.

# Appendix A

# Test Case Documentation

## A.1 Test Case Format

All 60 test cases are documented in the TEST_CASE_PLANNING_AND_EXECUTION.md file with the following structure:

- Test Case ID: TC-TTB-XX
- Test Number: X.1 - X.Y (decimal range)
- Test Description
- Test Designed By: Sarthak Kulkarni, Dhruv Tikhande, Atharv Petkar, Pulkit Saini
- Test Executed By: Sarthak Kulkarni, Dhruv Tikhande, Atharv Petkar, Pulkit Saini
- Execution Date: 2025-10-07
- Detailed Steps with Expected and Actual Results

## A.2 Sample Test Case

**Test Case ID:** TC-TTB-01
**Test Title:** Verify Dashboard Loads Correctly on Login
**Test Number:** 1.1 - 1.4
**Priority:** High
**Test Description:** Ensure dashboard displays all widgets and statistics after user login
**Steps:**

1. Navigate to login page - Expected: Login page displays correctly
2. Enter valid credentials - Expected: Credentials accepted
3. Click 'Sign In' button - Expected: User is redirected to dashboard
4. Verify dashboard widgets load - Expected: All statistics, upcoming classes, and quick actions are visible

# Appendix B

# Risk Assessment Documentation

## B.1 Risk Format

All risks are documented in the RISK_ASSESSMENT_SHEET.md file with probabilities 15%:

- Risk ID: R-TTB-XXX
- Type: Technical/External/Operational
- Probability: Percentage value
- Impact: Critical/High/Medium/Low
- Risk Description
- Mitigation Plan
- Monitoring Plan
- Management Plan

## B.2 Sample Risk

**Risk ID:** R-TTB-005
**Type:** Technical
**Probability:** 10%
**Impact:** Critical
**Risk Description:** Critical security vulnerability discovered in production system allowing unauthorized data access.
**Mitigation Plan:**

1. Conduct regular security audits and penetration testing
2. Implement security scanning in CI/CD
3. Follow OWASP guidelines

**Monitoring Plan:** Run automated security scans weekly. Monitor security patch releases for dependencies.
**Management Plan:** Deploy emergency patch within 4 hours. Notify affected users. Conduct incident post-mortem.

# Appendix C

# System Screenshots

## C.1 Dashboard View

[Screenshot of Dashboard View would be inserted here]

## C.2 Lecture Slots Management

[Screenshot of Lecture Slots page would be inserted here]

## C.3 Timetable View

[Screenshot of Timetable View would be inserted here]

## C.4 Enrollment Interface

[Screenshot of Enrollment Interface would be inserted here]

# Appendix D

# Installation and Deployment Guide

## D.1 Prerequisites

- Node.js (v18 or higher)
- MongoDB (local or Atlas)
- npm (v9 or higher)
- Git

## D.2 Installation Steps

1. Clone the repository:

   ```
   git clone https://github.com/[repository-url]
   cd Lecture_Scheduling_Prototype_testing
   ```

2. Install dependencies:

   ```
   npm install
   cd backend && npm install
   cd ../frontend && npm install
   ```

3. Configure environment variables:

   ```
   # Backend .env file
   MONGODB_URI=your_mongodb_connection_string
   JWT_SECRET=your_jwt_secret_key
   PORT=5000
   ```

4. Start the application:

   ```
   # Start backend
   cd backend && npm start

   # Start frontend (in new terminal)
   cd frontend && npm run dev
   ```

## D.3 Docker Deployment

```
# Build and run using Docker Compose
docker-compose up --build
```