Panacea – A Smart ERP with Dynamic Scheduling

A MINOR PROJECT REPORT SUBMITTED TO

THE NATIONAL INSTITUTE OF ENGINEERING, MYSURU
(An Autonomous Institute under Visvesvaraya Technological University, Belagavi)

in partial fulfillment for the award of degree of

Bachelor of Engineering
in
Computer Science and Engineering

Submitted By

Pranav Jagdish	**4NI22CS150**
Ninad Parate	**4NI23CS125**
Pabitra Ranjan Jena	**4NI23CS134**
Pragyanshu Mishra	**4NI23CS143**

Under the guidance of

Ms. Usha K Patil

Assistant Professor

Department of CS&E
NIE, Mysuru

Department of Computer Science & Engineering
THE NATIONAL INSTITUTE OF ENGINEERING
(Autonomous Institution)
Mysuru - 570 008
2025-26

THE NATIONAL INSTITUTE OF ENGINEERING

(An Autonomous Institute under Visvesvaraya Technological University, Belagavi)

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project work entitled "Panacea – A Smart ERP with Dynamic Scheduling" is a Bonafide work carried out by Pranav Jagdish (4NI22CS150), Ninad Parate (4NI23CS125)**,** Pabitra Ranjan Jena (4NI23CS134)**,** Pragyanshu Mishra (4NI23CS143) in partial fulfillment for the award of degree of **Bachelor of Engineering in Computer Science and Engineering**, of Visvesvaraya Technological University, Belagavi, during the year **2024-25**. It is certified that all corrections / suggestions indicated during internal assessment have been incorporated and the corrected copy has been deposited in the department library. This project report has been approved in partial fulfillment for the award of the said degree as per academic regulations of The National Institute of Engineering (Autonomous Institution).

Ms. Usha K Patil
Assistant Professor
Dept. of CS&E
NIE, Mysuru

Dr. Anitha R
Professor and Head
Dept. of CS&E
NIE, Mysuru

ABSTRACT

Efficient academic and administrative management continues to be a major challenge for many colleges, where essential tasks such as timetable creation, attendance tracking, and information sharing are still carried out manually. These conventional practices are time-consuming, error-prone, and often inconvenient for students, faculty, and administrators. To address these issues, this project proposes a **Smart Academic ERP System** featuring **Automated Timetable Generation** and **Real-Time Notifications**, providing a unified platform for academic, administrative, and communication services.

A key component of the system is its **intelligent timetable generator**, which automatically produces conflict-free schedules by considering factors such as classroom availability, faculty workload, and course requirements. It also supports dynamic adjustments—such as class cancellations or rescheduling—and instantly notifies all affected users to minimize disruptions.

In addition to scheduling, the platform integrates several essential modules, including attendance management, course registration (with provisions for backlog and debarred students), fee payment, result announcements, and e-library access. A dedicated announcements portal enables the publication of important notices, regulations, case filings, fines, and disciplinary actions, ensuring transparency and easy accessibility for all stakeholders.

To improve communication, the ERP includes a **real-time chat interface** that supports seamless interaction between students and faculty. A centralized admin panel allows administrators to manage users, permissions, and institutional policies efficiently. The system is built with scalability in mind, employing modern web technologies, reliable backend frameworks, and real-time communication protocols to ensure robust performance.

By automating critical processes and consolidating institutional services, this Smart Academic ERP System significantly reduces manual workload, enhances operational efficiency, and fosters a more connected campus environment. The final product demonstrates strong technical implementation skills while addressing real-world challenges commonly faced in higher education institutions.

ACKNOWLEDGMENT

I sincerely owe my gratitude to all the persons who helped and guided me in completing this mini project.

I am thankful to **Dr. Rohini Nagapadma**, Principal, NIE College, Mysuru, for all the support she has rendered.

I thank **Dr. Anitha R**, Professor and Head, Department of Computer Science and Engineering, for her constant support and encouragement throughout the tenure for the mini project work.

I would like to sincerely thank my guide **Ms. Usha K Patil**, Assistant Professor, Department of Computer Science and Engineering, for providing relevant information, valuable guidance, and encouragement to complete this minor project.

Pranav Jagdish	4NI22CS150
Ninad Parate	4NI23CS125
Pabitra Ranjan Jena	4NI23CS134
Pragyanshu Mishra	4NI23CS143

TABLE OF CONTENTS

CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Managing academic and administrative activities in higher education institutions is often complex and resource-intensive. Tasks such as timetable preparation, attendance tracking, result publication, and faculty–student communication are frequently performed manually or spread across multiple disconnected systems. These fragmented processes lead to inefficiencies, scheduling conflicts, delays, and poor coordination. With the increasing adoption of digital technologies, there is a clear need for an integrated solution that ensures accuracy, transparency, and streamlined access to essential services.

The proposed project, **Smart Academic ERP with Automated Timetable Generation and Real-Time Notifications**, aims to address these challenges through a unified and automated digital platform. The system's core feature is an intelligent timetable generator that produces optimized, conflict-free schedules. Beyond this, the platform incorporates a comprehensive suite of academic and administrative modules, including attendance management, course registration, fee payment, e-library access, and built-in communication tools. By consolidating these functions into a single system, the project enhances operational efficiency and supports a more organized and responsive academic environment.

1.1 Background

In most colleges, timetable preparation is performed manually by faculty coordinators, often requiring several iterations to eliminate scheduling conflicts. This manual approach is time-consuming and inefficient, especially for institutions with large student populations and diverse courses. Furthermore, existing processes such as result declaration, fee collection, and announcement circulation often depend on physical notices or isolated software modules, which makes coordination difficult. Recent developments in campus management software have demonstrated the benefits of centralization, automation, and real-time communication. However, many commercial solutions are costly and too complex for small to medium-sized institutions. Hence, there is a pressing need for an academic management system that is both efficient and scalable, designed specifically for institutional convenience and student accessibility.

1.2 Purpose

The purpose of this project is to design and implement a lightweight yet comprehensive academic management platform that:

- Automates repetitive academic tasks such as timetable scheduling and attendance logging.
- Provides real-time updates and notifications to minimize communication delays.
- Unifies multiple academic and administrative functions on a single, accessible portal.
- Demonstrates the feasibility of a scalable ERP solution tailored to academic institutions.

1.3 Objectives

The specific objectives of the project are:

1. To develop a **conflict-free automatic timetable generator** that considers faculty availability, classroom allocation, and course requirements.
2. To provide features for **dynamic class management**, including cancellations, rescheduling, and notifications.
3. To integrate academic functions such as **attendance tracking, result publishing, and course registration**.
4. To offer administrative modules including **fee payment, rules and regulations, case filings, and fines management**.
5. To enable **real-time communication** between students, faculty, and administration via chat and notification systems.
6. To build an **admin panel** for centralized management of users, roles, and institutional policies.
7. To ensure that the system is **scalable, user-friendly, and adaptable** to different institutional requirements.

Chapter 2

LITERATURE REVIEW

[1] **Khan & Sonawane, "College Timetable Generation using Graph Neural Networks and Reinforcement Learning," 2025.**

This paper presents a timetable generation model using GNNs paired with reinforcement learning. The system treats subjects, time slots, and rooms as graph nodes and optimizes scheduling using iterative training. It significantly reduces conflicts and dynamically adjusts faculty availability. However, it requires substantial training data and computational power, limiting adoption in small institutions.

[2] **Mumcu & Çebi, "The Role of Push Notifications in Shaping Students' Engagement, Self-Regulation, and Academic Procrastination," International Journal of Educational Technology in Higher Education, 2025.**

The study evaluates the impact of personalized academic notifications on learning behaviors. Results show that timely notifications reduce procrastination and improve engagement. The study suggests moderation because excessive alerts may distract learners. The paper highlights the relevance of notification-driven platforms within educational ERP systems.

[3] **Sajithabanu & Mahmootha, "Intelligent College Management System with Real-Time Monitoring and Visual Analytics," Journal of Data Mining and Management, 2025.**

This work proposes an intelligent ERP platform capable of monitoring administrative and academic activities in real time. Dashboards provide actionable analytics to users. The system integrates modules such as finance, academics, and communication. While promising, deployment complexity remains a challenge for smaller institutions.

[4] **Wu et al., "Digital Real-Time Learning System with Push Notifications," Journal of Medical Internet Research, 2022.**

The authors develop an online learning system that sends real-time alerts to remind students about sessions, attendance, and feedback. Implementation improves participation and class responsiveness. The research underscores the growing importance of real-time notification services to foster

engagement within digital academic platforms.

[5] **Uma et al., "Automatic Timetable Generation," International Journal for Research in Applied Science & Engineering Technology (IJRASET), 2023.**

This paper explores heuristic-based scheduling using Simulated Annealing and Tabu Search. The system focuses on minimizing teacher and room conflicts while satisfying institutional constraints. It is computationally efficient; however, the algorithm may not always produce globally optimal schedules under heavy constraints.

[6] **Ahmed & Salman, "Cloud-Based Attendance Management System Using Mobile Authentication," International Journal of Emerging Technologies in Learning (IJET), 2021.**

This paper proposes a mobile-based attendance system that integrates cloud storage and QR authentication. Attendance records are updated in real time and stored securely. Although reliable, the system requires stable network connectivity, limiting usage in low-coverage areas.

[7] **Park & Lee, "AI-Assisted Academic Resource Planning for Higher Education," Applied Sciences Journal, 2024.**

This study focuses on using AI models to allocate academic resources such as classrooms, instructors, and time slots. The authors demonstrate that AI solutions reduce administrative workload significantly. The prototype system showed improved scheduling accuracy, but AI explainability remained a concern.

[8] **Sharma & Thomas, "A Web-Based Academic ERP for University Management," International Journal of Computer Applications, 2021.**

The work describes an ERP built to centralize examination management, attendance logs, and timetable records. Offering role-based dashboards, the system simplifies operations through automation. Nevertheless, limited scalability and integration constraints reduce long-term adoption potential.

Chapter 3

EXISTING SYSTEM AND PROPOSED SYSTEM

3.1 Existing System

In most colleges and universities, academic and administrative tasks are handled either manually or through disjointed software modules.

- Timetable Preparation: Typically managed manually by faculty coordinators using spreadsheets or paper-based methods. This is a time-consuming process and often results in scheduling conflicts among classrooms, courses, and faculty. Adjustments for cancellations or extra classes require repeated manual intervention.
- Attendance Management: Conducted through physical registers or isolated attendance software that lacks integration with other systems.
- Course Registration & Result Processing: Carried out through separate portals or offline forms, creating redundancy and delays.
- Fee Payment: Still heavily dependent on offline modes in many institutions or semi-digital systems without full integration into the academic record.
- Communication & Notifications: Announcements are typically made via notice boards, emails, or classroom messaging apps. These methods lack consistency, timeliness, and centralization.
- E-Library and Resources: Often scattered across multiple platforms or basic repositories without integration into the main student portal.

3.1.1 Drawbacks of Existing System

- Manual timetable creation is time-consuming
- Frequent scheduling conflicts occur
- No centralized portal for academic information
- Attendance must be tracked separately (paper/software)
- Students lack real-time access to attendance/announcements
- Separate systems cause data redundancy
- Poor communication between faculty & students
- Result publishing and fee management are slow

3.2 Proposed System

The proposed system, **Smart Academic ERP with Automated Timetable Generation and Real-Time Notifications**, offers a **unified, automated, and scalable solution** that addresses the limitations of current practices.

- **Automated Timetable Generation**: Uses constraint-based algorithms to automatically generate conflict-free schedules, taking into account faculty availability, room capacity, and course requirements.
- **Integrated Attendance Management**: Teachers can upload attendance directly into the system, and students can view records instantly. This eliminates manual registers and ensures transparency.
- **Course Registration & Results**: Provides a seamless portal for course selection. Results are published in real-time through the portal.
- **Unified Fee Payment Portal**: Secure online fee payment with automated receipt generation and integration with the student's academic record.
- **Real-Time Notification**: Facilitates instant communication between teachers, students, and administrators through push notifications for timetable updates, announcements, and deadlines.
- **E-Library & Academic Resources**: Offers a centralized repository of digital content (e-books, lecture notes, past papers) integrated within the same platform.
- **Administrative Functions**: Includes portals for rules, regulations, case filings, fines, and disciplinary actions.
- **Admin Panel**: Provides centralized control for managing users, roles, permissions, academic content, and system settings.

3.2.1 Advantages of Proposed System

- Automated conflict-free timetable generation
- Real-time scheduling updates & notifications
- Integrated attendance system with instant visibility
- Unified portal → no redundant data entries
- Smooth communication via chat & announcements
- Improved transparency for students & faculty
- Handles re-registrations and special cases
- Secure access via user authentication

Chapter 4

SYSTEM REQUIREMENTS

4.1 Hardware Requirements

Minimum (for development/testing)

- Processor: Intel Core i3 (2.0 GHz or above) / AMD equivalent
- RAM: 4 GB
- Storage: 250 GB HDD / 120 GB SSD
- Display: 1366×768 resolution
- Network: Stable internet connection (5 Mbps or above)

Recommended (for deployment/server)

- Processor: Intel Core i5/i7 (Quad-core, 2.5 GHz or above)
- RAM: 8–16 GB
- Storage: 512 GB SSD (for faster database queries)
- Network: High-speed broadband (10 Mbps+ with static IP if hosted locally)
- Backup: External storage/cloud backup for database

4.2 Software Requirements

- Operating System: Windows / Linux (Ubuntu preferred)
- Backend: Python (Django / Flask)
- Frontend: ReactJS / Angular / Vue / Next.js
- Database:
    - PostgreSQL / MySQL (primary)
    - Redis / MongoDB (optional – caching & live chat)
- Authentication: JWT / OAuth2
- IDE: VS Code / PyCharm
- Version Control: Git + GitHub/GitLab
- Browser (Testing): Chrome / Firefox

Functional Requirements:

No.	Requirement
FR1	System must allow user authentication using registered credentials.
FR2	Admin must be able to manage users (create/update/delete).
FR3	System must generate conflict-free timetable based on faculty & room availability.
FR4	Faculty must be able to upload attendance.
FR5	Students must be able to view attendance records.
FR6	Admin/Faculty must be able to post announcements.
FR7	Users must receive real-time push notifications.
FR8	System must allow course registration.
FR9	System must allow result publication & viewing.
FR10	System must support fee payment & receipt download.
FR11	System must store and provide access to notes/resources.
FR12	Faculty must be able to schedule quizzes/tests.
FR13	Students must be able to attempt quizzes.
FR14	System must maintain complete academic session history.
FR15	System must generate attendance and academic reports.

Table 4.1 – Functional Requirements

Non-Functional Requirements:

Category	Requirement
Performance	• System should load dashboard within 3 seconds. • Attendance storage operations < 2 seconds.
Reliability	• System uptime must be ≥ 99%.
Scalability	• System must support increasing users and courses without major changes.
Security	• User data must be protected with authentication + role-based access. • All communication through HTTPS.
Availability	• System should be accessible 24×7 except maintenance hours.
Usability	• UI should be simple, consistent, and easy to navigate.
Portability	• Works on major browsers & adaptable to mobile.
Maintainability	• Code should follow modular design for easy updates.
Backup/Recovery	• Automated database backups daily; recovery must be possible within 2–4 hours.
Data Integrity	• No duplicate records; updates must maintain consistency.

Table 4.2 – Non-Functional Requirements

Chapter 5

SYSTEM DESIGN

5.1 System Architecture

1. High-Level Architecture

The system follows a three-tier architecture:

1. **Presentation Layer (Frontend):**
   - Web Portal (for students, faculty, and admin)
   - Mobile Access (optional)
   - Handles user interactions (login, timetable view, attendance check, notifications, fee payment, etc.).
2. **Application Layer (Backend):**
   - Implements business logic (automatic timetable generation, attendance handling, course registration).
   - Handles user authentication, notifications, and admin controls.
   - Provides APIs for frontend communication.
3. **Data Layer (Database):**
   - Relational DB for structured academic data (students, teachers, courses, timetable, attendance, fees).
   - Optional NoSQL/Redis for chat and notifications.

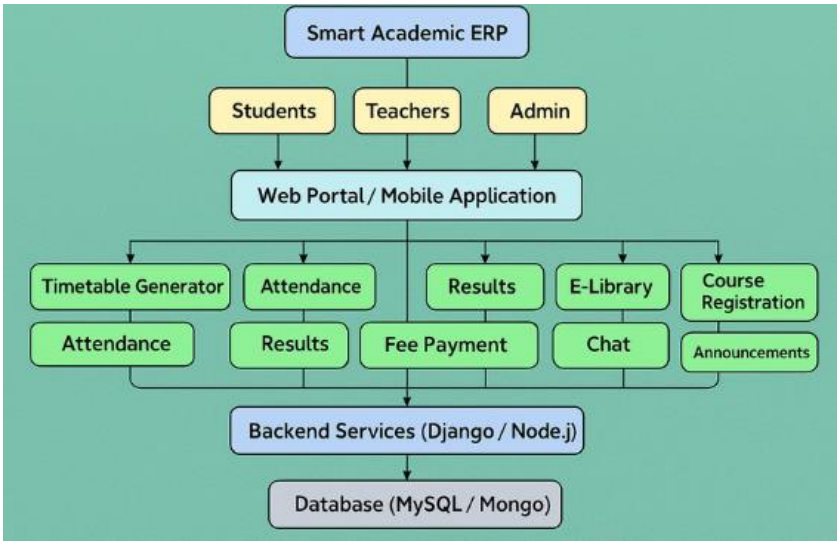

Fig 5.1 – Flow Chart

1. **Overall Architecture**
    - N-tier / MVC architecture built on Django
    - Separation of concerns – Presentation (templates + static files), View/Controller layer (Django views), Service/Business-logic layer (functions or class-based services), Data-access layer (Django ORM).
2. **Main subsystems (modules)**
    - Authentication & User Management – custom CustomUser model, CustomUserManager, role-based user_type (HOD / Staff / Student).
    - Academic Core – Course, Subject, Session, Staff, Student.
    - Timetable Engine – Room, TimetableEntry, auto-generation algorithm.
    - Attendance Service – Attendance, AttendanceReport.
    - Leave & Feedback Service – LeaveReportStudent/Staff, FeedbackStudent/Staff.
    - Notification Service – NotificationStudent/Staff, push via Firebase Cloud Messaging (FCM).
    - Library Service – Book, Library, IssuedBook.
    - Content Service – Note, MCQTest, MCQQuestion, MCQOption, MCQSubmission, MCQAnswer.
3. **Data-flow overview**
    - User request (browser → Django view).
    - View validates request → calls a service (business logic).
    - Service interacts with ORM models → DB read / write.
    - Service returns data → view renders HTML / JSON → response back to browser.
4. External interfaces
    - FCM – HTTP POST to Google for push notifications.
    - File storage – uploaded notes, profile pictures, etc., stored under MEDIA_ROOT and served via Django static file handling.
    - Optional API – the same view logic can be called by a SPA or mobile client (JSON responses).

5. Security model

- Login with email-only (USERNAME_FIELD = 'email').
- RBAC enforced by checking request.user.user_type in every view (HOD = 1, Staff = 2, Student = 3).
- CSRF protection on all POSTs, @csrf_exempt only on AJAX endpoints that already verify the user session.

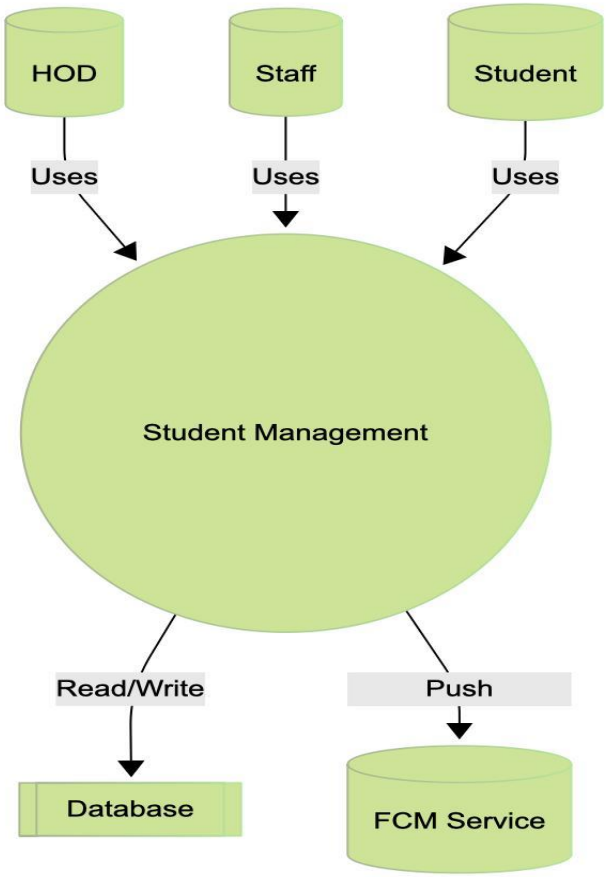

Fig 5.2 – Context Diagram

2. Detailed-Level Architecture

1. Authentication & User Management

Classes

CustomUser – extends AbstractUser, drops username, adds email, user_type, profile_pic, address, gender, fcm_token.

CustomUserManager – methods _create_user, create_user, create_superuser (hashes password with make_password).

Signals – post_save creates a linked profile (Admin, Staff, Student) based on user_type.

Permissions – Decorators in views (@login_required) and inline checks (if request.user.user_type != 2: return HttpResponseForbidden()).

2. Academic Core

Model Key fields Important constraints

Course name unique name (application-level).

Session start_year, end_year used for grouping students and timetable.

Subject name, staff FK, course FK, credits credits = max periods per week.

Staff admin FK, course FK one-to-one with CustomUser.

Student admin FK, course FK, session FK one-to-one with CustomUser.

3. Attendance Service

Attendance – session, subject, date.

AttendanceReport – FK to Student & Attendance, status (True = present).

Workflow (staff → save_attendance view):

Create a new Attendance row.

Loop through posted student list → create AttendanceReport for each with the submitted status.

All operations wrapped in a DB transaction so the whole batch succeeds or fails together.

Reporting – Student view aggregates AttendanceReport rows to compute present/absent percentages.

4. Notification Service

Model – NotificationStudent / NotificationStaff (message, timestamps).

Push flow –

Admin/HOD writes a message → admin_notify_* view stores it.

staff_fcmtoken / student_fcmtoken endpoint updates the user's fcm_token.

When a notification is sent, a JSON payload is POSTed to FCM URL with the token; on success the message is saved in the corresponding Notification* table.

5. Content Service (Notes & MCQ)

Notes – Note stores title, FK to Subject & Staff, and a file upload (file).

MCQ – hierarchy: MCQTest → MCQQuestion → MCQOption.

Submission flow –

Student opens a test (student_take_test).

On POST, a MCQSubmission is created (score = 0).

For each posted answer, an MCQAnswer is created linking the submission, question, and selected option.

If the selected option's is_correct flag is true, increment a local score counter.

After all questions processed, update submission.score.

6. Result Management

StudentResult – FK to Student & Subject, fields test (continuous-assessment), exam.

Update flow – Staff view (staff_add_result) either creates a new result row or updates an existing one (via get_or_create pattern).

7. API / URL Mapping (quick reference)

URL (pattern) HTTP verb Owner Core operation

/staff_take_attendance/ GET Staff Show subject & session pickers

/save_attendance/ POST Staff Persist Attendance + many AttendanceReport rows

/student_take_test/<id>/ GET/POST Student Render MCQ test / evaluate submission

/manage_timetable/ GET/POST HOD View existing entries, add rooms, add single entry, or auto-generate

/send_staff_notification/ POST HOD Push via FCM + store in DB

/add_book/ POST Staff Create Book record

/issue_book/ POST Staff Create IssuedBook record

/admin_view_profile/ GET/POST HOD View / edit own profile

8. Error handling & validation

Model clean() methods raise ValidationError (e.g., TimetableEntry.clean).

Forms (ModelForm subclasses) perform field-level validation (clean_<field>()).

Views catch generic Exception only to log/notify; user-facing errors are displayed via Django messages.

All POST endpoints that are AJAX-driven return JSON (JsonResponse) with proper HTTP status codes.

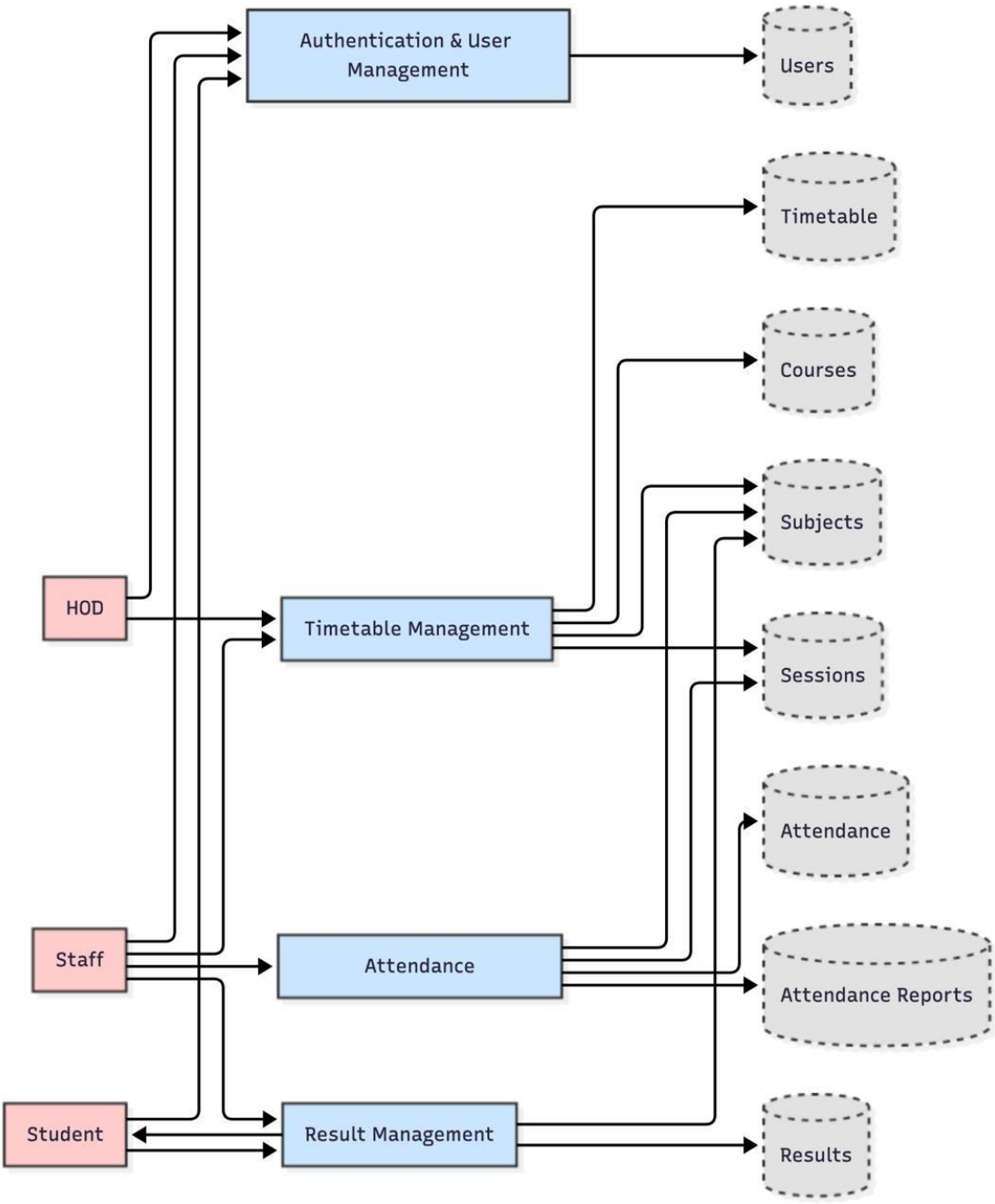

Fig 5.3 – Administrative and Service Sub-Systems

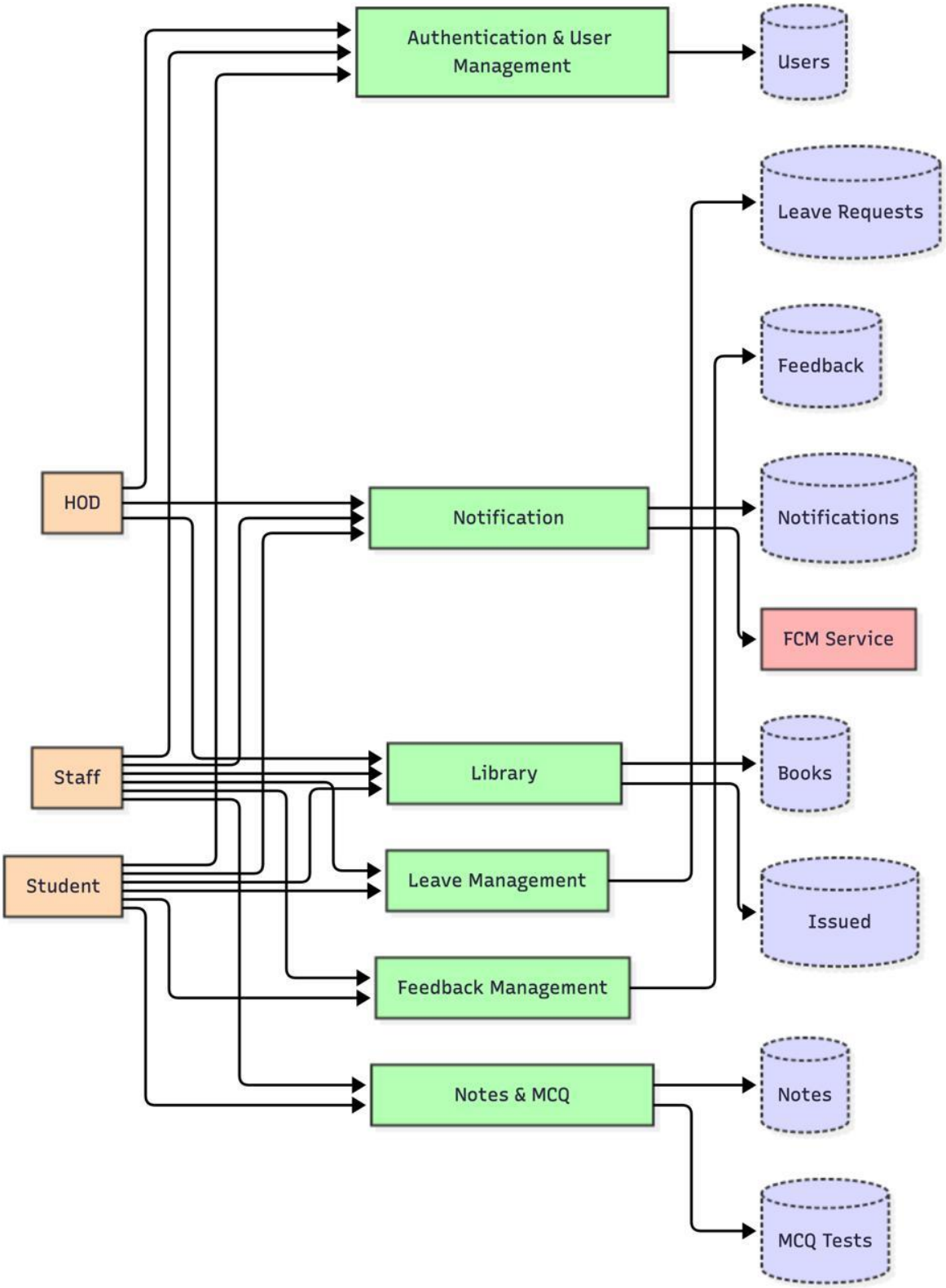

Fig 5.4 – Academic Core Sub-Systems

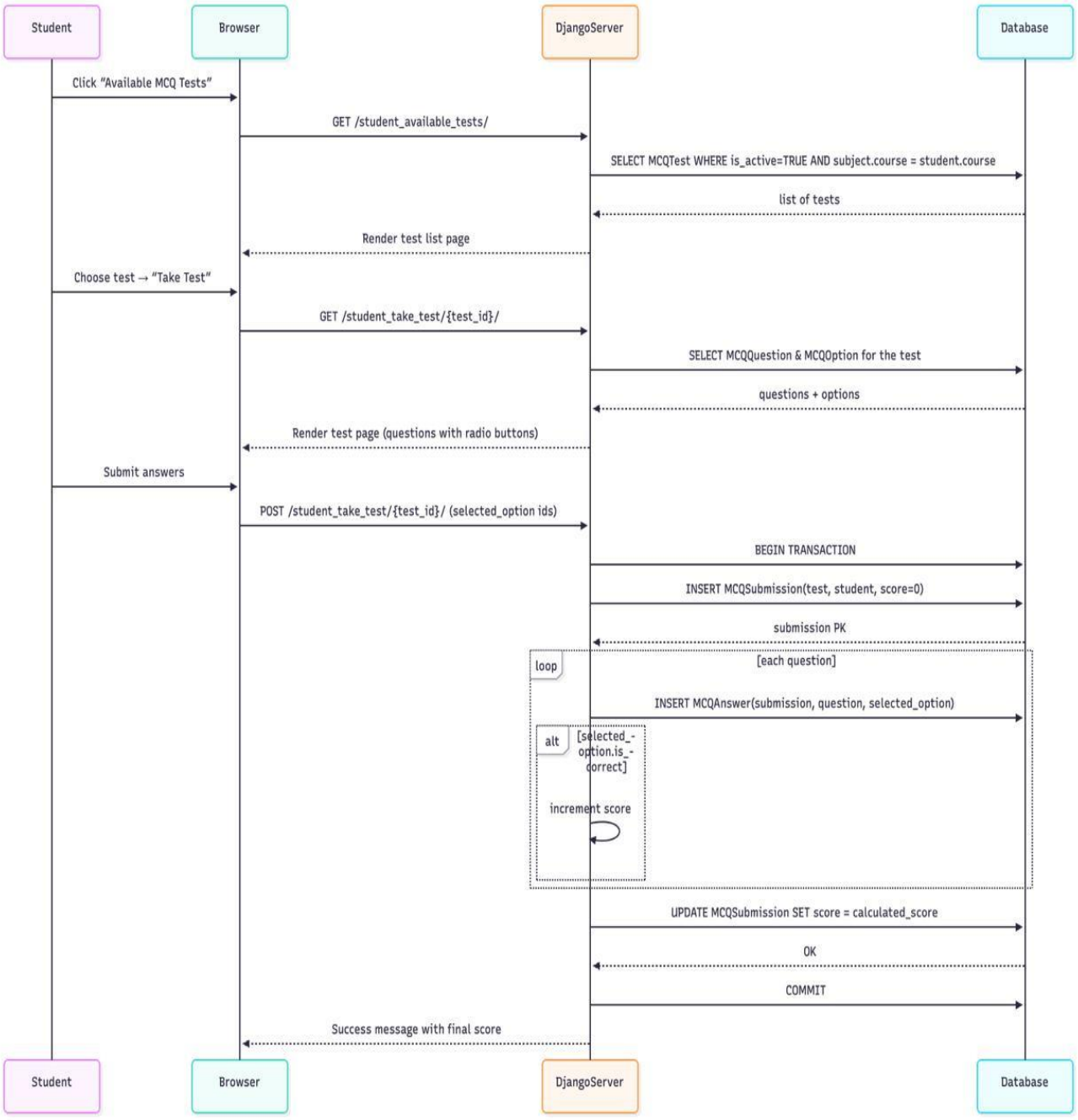

Fig 5.5 – Sequence Diagram Quiz

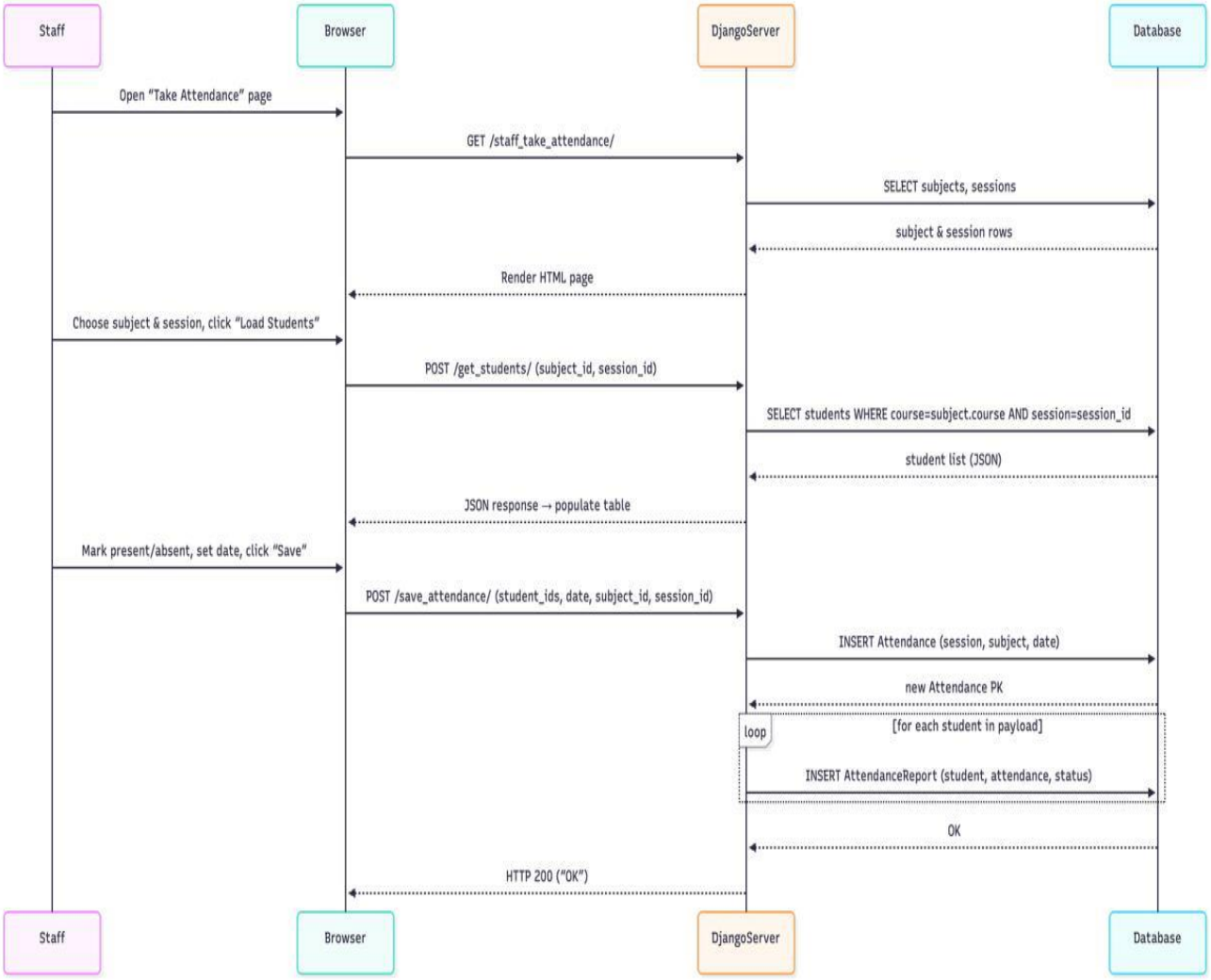

Fig 5.6 – Sequence Diagram Staff Records

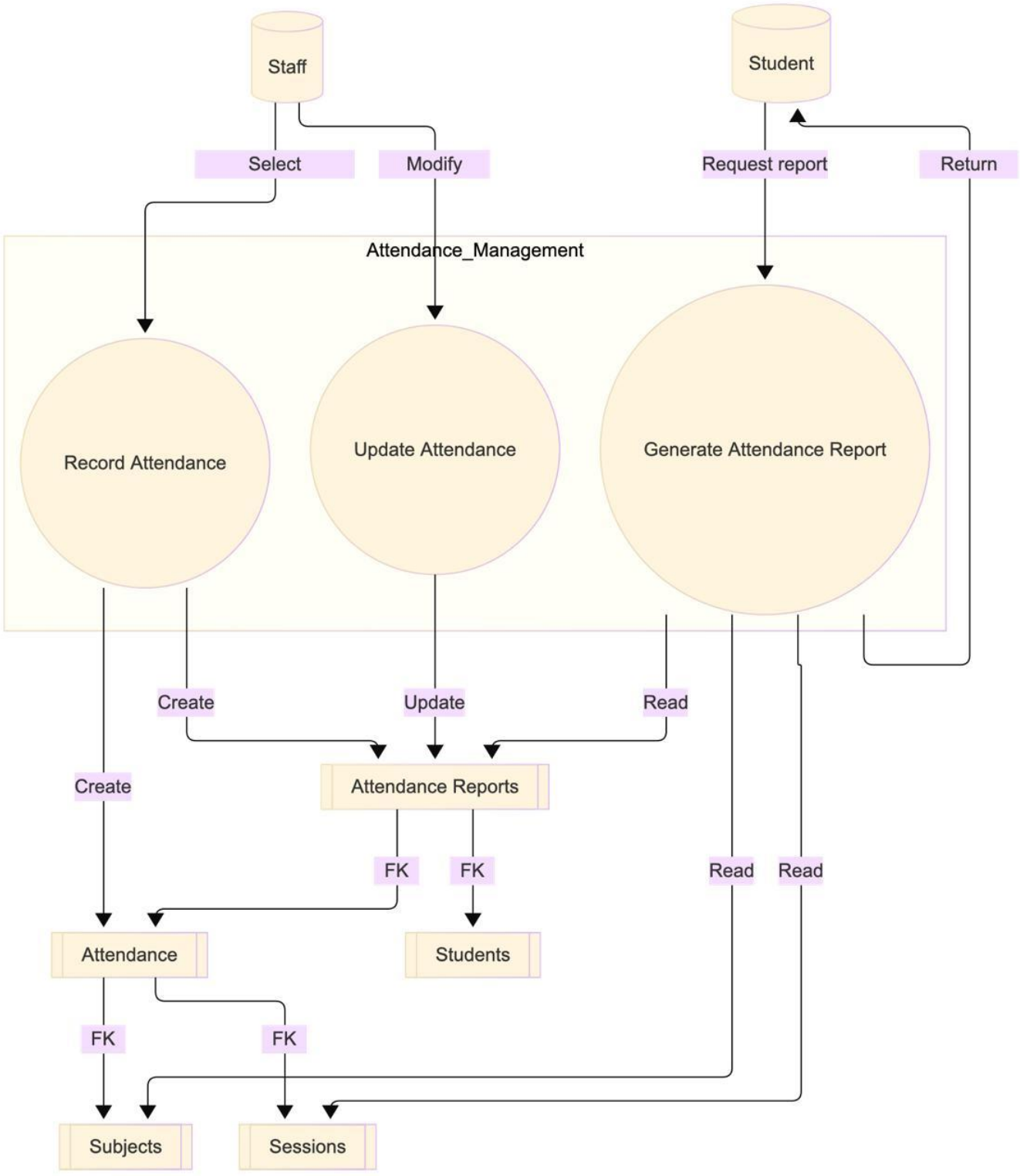

Fig 5.7 – Detailed Attendance Management

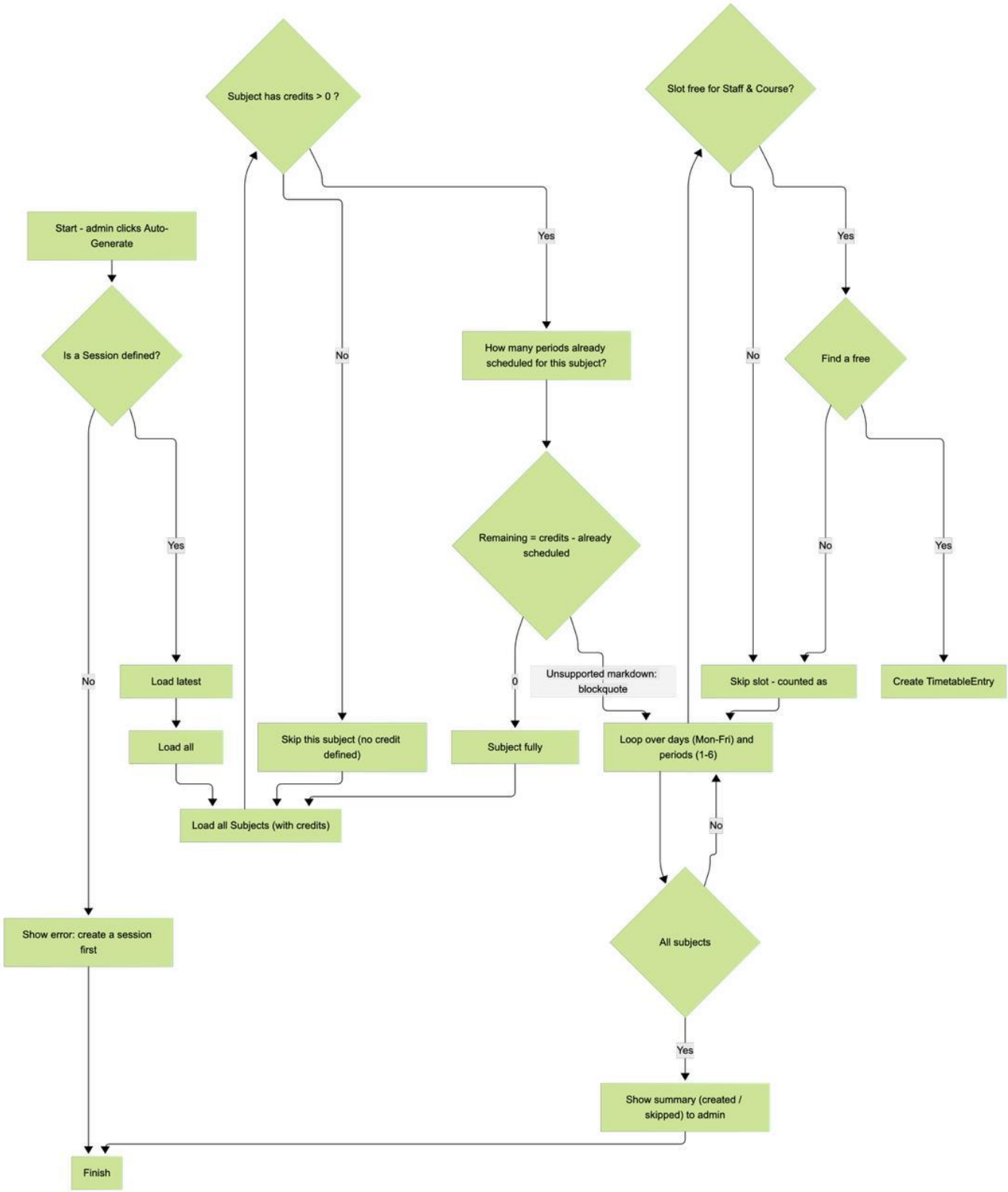

Fig 5.8 – Automated Generation of Timetable Flowchart

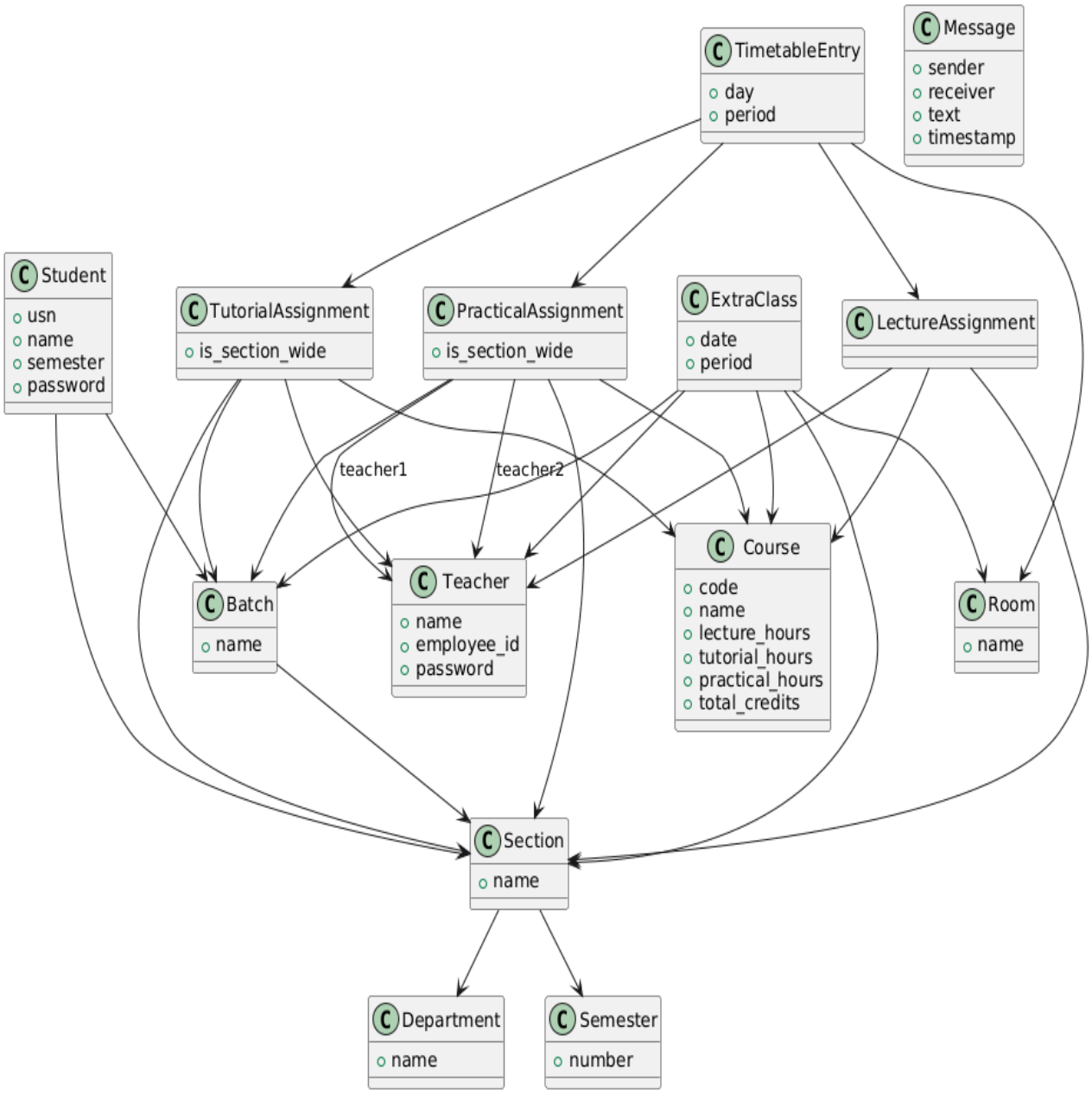

Fig 5.9 – ER Diagram Core Systems

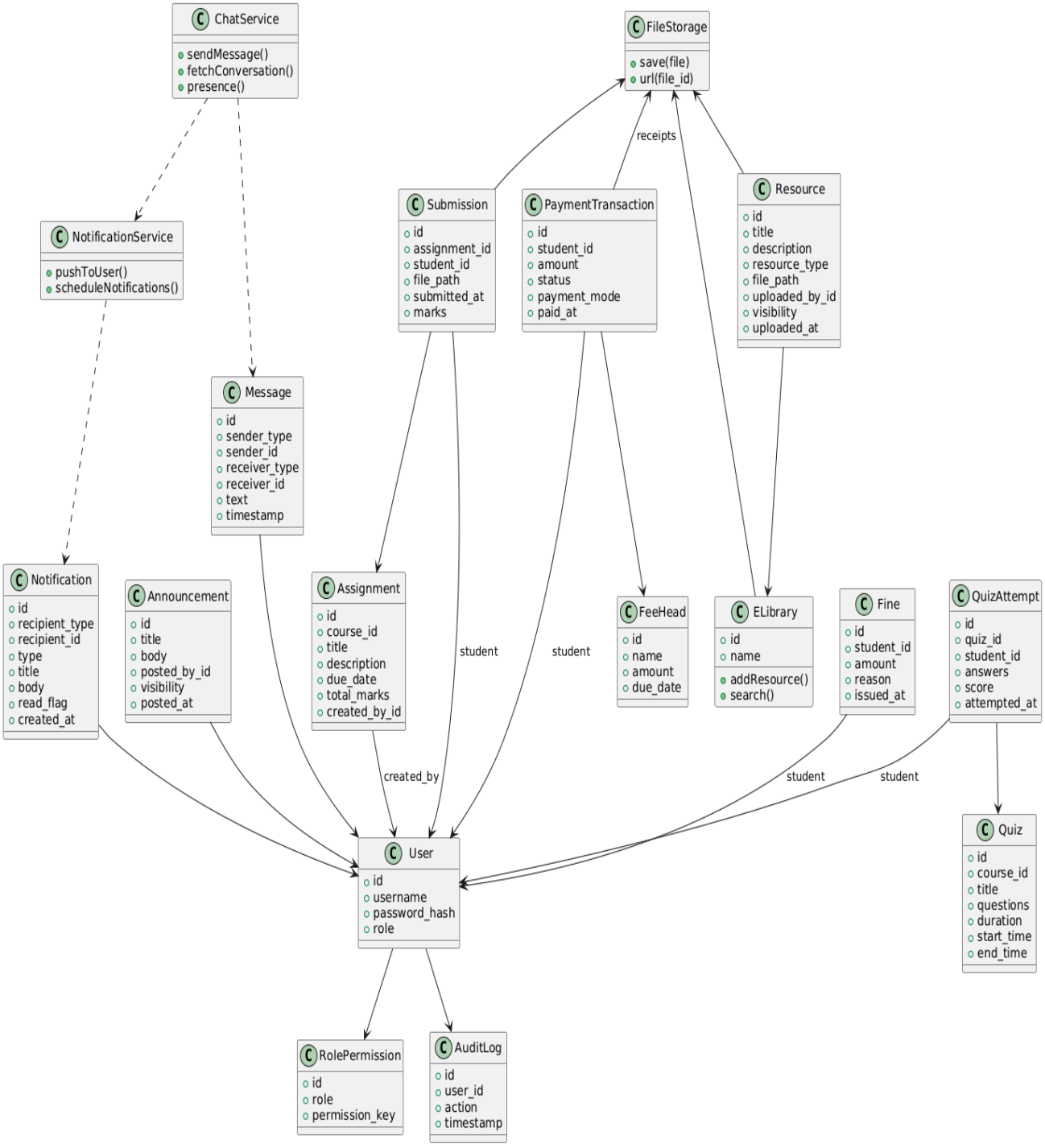

Fig 5.10 – ER Diagram Extended Services

5.2 Use Case Diagram

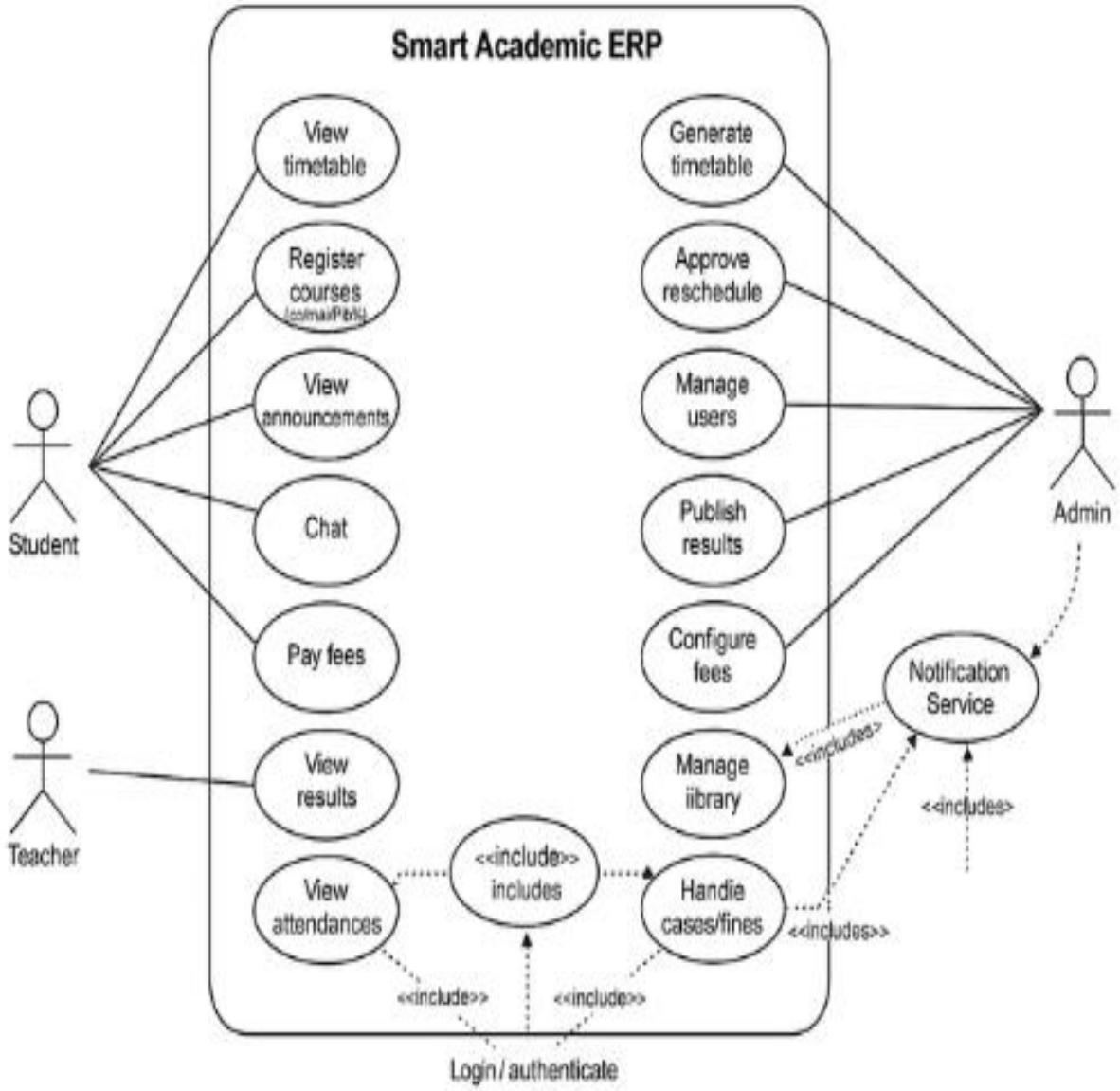

Fig 5.11 – Use-Case Diagram

Chapter 6

IMPLEMENTATION

Pseudocode

```
// ====== BOOT ======
MAIN():
  load_config()
  connect_database()
  start_web_server(routes)

// ====== AUTH ======
LOGIN(email, password):
  u ← DB.users.find_by_email(email)
  IF !u OR !verify(password, u.hash): return ERROR("Invalid")
  token ← JWT.sign({id:u.id, role:u.role})
  return OK(token)
REGISTER(email, pass, role):
  IF DB.users.exists(email): return ERROR("Exists")
  u ← {email, hash:hash(pass), role, verified:false}
  DB.users.insert(u)
  send_verification(email)
  return OK("Check email")


// ====== ROLE GUARD ======
AUTHZ(token, required_role):
  claims ← JWT.verify(token)
  IF claims.role ∉ required_role: return ERROR("Forbidden")
  return OK(claims.id)
```

```
// ====== TIMETABLE (Constraint-based) ======
GENERATE_TIMETABLE(session):
  slots ← all_days×periods
  FOR each subject IN session.subjects:
    need ← subject.credits
    WHILE need > 0:
      s ← pick_best_slot(slots, subject, constraints)
      IF violates(s, subject, constraints): s ← next_slot()
      place(subject, s); mark_used(s); need--

constraints:
  no_staff_double_book, no_room_double_book,
  course_one_subject_per_slot, room_capacity_ok

// ====== ATTENDANCE ======
ENABLE_ATTENDANCE(class_id, by_staff):
  code ← random_4_digits()
  window ← now()..now()+X mins
  DB.attn_window[class_id] ← {code, window}
  notify_class(class_id, "Attendance open")
  return OK(code)

MARK_ATTENDANCE(student_id, class_id, code, location):
  w ← DB.attn_window[class_id]
  IF !w OR now() ∉ w.window OR code ≠ w.code: return ERROR("Closed/Code")
  IF !within_radius(location, lecturer_location(class_id)): return ERROR("Out of range")
  DB.attendance.upsert({student_id, class_id, date:today}, {status:true})
  return OK("Marked")


// ====== NOTIFICATIONS ======
NOTIFY(user_ids[], title, body):
```

```
  FOR id IN user_ids: push_fcm(id, title, body)
  DB.notifications.bulk_insert(user_ids, title, body)

// ====== COURSES / REGISTRATION ======
ADD_COURSE(admin, course):
  AUTHZ(admin, ["ADMIN"])
  DB.courses.insert(course)

REGISTER_COURSE(student_id, course_id):
  IF !prereqs_met(student_id, course_id): return ERROR("Prereq")
  DB.enrollments.insert({student_id, course_id})

// ====== RESULTS ======
UPLOAD_RESULT(staff, course_id, student_id, grade):
  AUTHZ(staff, ["STAFF","ADMIN"])
  DB.results.upsert({student_id, course_id}, {grade})

VIEW_RESULT(student_id):
  return DB.results.find({student_id})

// ====== MATERIALS ======
UPLOAD_NOTE(staff, subject_id, file):
  AUTHZ(staff, ["STAFF","ADMIN"])
  url ← storage.put(file)
  DB.notes.insert({subject_id, staff_id:staff, url})

GET_NOTES(subject_id):
  return DB.notes.find({subject_id})
```

```
// ====== REPORTS ======
ATTENDANCE_REPORT(student_id, range):
  recs ← DB.attendance.find({student_id, date∈range})
  pct ← present(recs)/total(recs)
  return {recs, pct}
TIMETABLE_VIEW(role_id):
  return DB.timetable.view_for(role_id)

// ====== ADMIN ======
CREATE_USER(admin, data):
  AUTHZ(admin, ["ADMIN"])
  DB.users.insert(data)
SET_ROLE(admin, user_id, role):
  AUTHZ(admin, ["ADMIN"])
  DB.users.update(user_id, {role})

// ====== ERROR HANDLING ======
API(handler):
  TRY: return handler()
  CATCH ValidationError e: return ERROR("Bad Request", e.msg)
  CATCH Exception e: log(e); return ERROR("Server")

// ====== UTIL ======
within_radius(p1, p2):
  return haversine(p1, p2) ≤ ALLOWED_METERS
pick_best_slot(slots, subject, constraints):
  return argmin_s(conflict_score(s, subject, constraints))
```

Chapter 7

TESTING

7.1 Types of Testing

7.1.1 Unit Testing

Each module such as login, timetable, notifications, attendance, and database integration was individually tested for proper behaviour.

7.1.2 Integration Testing

Checked smooth communication among modules:

- Login → Dashboard
- Timetable → Notifications
- Attendance → Reports

7.1.3 Functional Testing

Validated system functions:

- Admin creates/updates timetable
- Faculty can record attendance
- Students can view attendance, results
- Admin pushes announcements

7.1.4 Authentication Testing

- Incorrect credentials rejected
- Valid email required

7.1.5 Exception Handling

- Invalid/empty login
- Scheduling conflicts
- Unavailable faculty/room

7.1.6 Performance Testing

- Data fetch and updates remained stable under multi-user access
- Timetable/attendance loads < 2 seconds under testing conditions

7.1.7 User Acceptance Testing

Faculty and students used the system; feedback confirmed:

- Easy navigation
- Faster communication
- Accurate scheduling

7.2 Application Testing Checklist

S. No.	Test Case / Input	Expected Output	Actual Output	Result (P/F)
1	Open application	Application launches successfully	As expected	P
2	Enter valid credentials (Admin/Staff/Student)	Login successful → Dashboard opens	As expected	P
3	Enter invalid credentials	Display “Invalid username/password”	As expected	P
4	Login without email verification	Display “Verify email first”	As expected	P
5	Access system without login	Blocked → redirect to login	As expected	P
6	Admin creates timetable	Timetable stored & visible	As expected	P
7	Create timetable with conflicting slot	Display conflict error msg	As expected	P
8	Faculty marks attendance	Attendance recorded	As expected	P
9	Student views attendance	Attendance percentage shown	As expected	P
10	Duplicate attendance entry	Block duplicate entries	As expected	P
11	Student receives notification	Notification delivered	As expected	P
12	Admin posts announcement	Announcement visible to students	As expected	P
13	Student views notes	Notes displayed properly	As expected	P
14	Result upload by staff	Results saved	As expected	P
15	Student views results	Correct result displayed	As expected	P
16	Fee payment initiated	Redirect to secure payment	As expected	P
17	Invalid payment attempt	Display error	As expected	P
18	Admin adds user	User account created	As expected	P
19	Admin removes user	Account removed	As expected	P
20	Internet disconnected	Show connection error	As expected	P

Table 7.1 Testing Checklist

Chapter 8

SCREENSHOTS

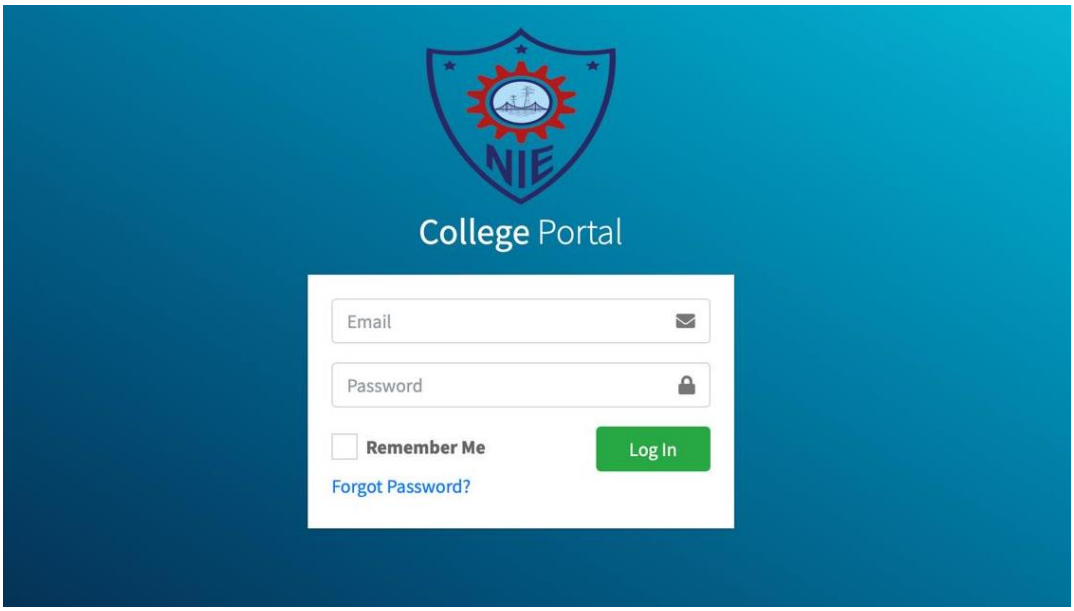

Fig 8.1 – Unified Login Page

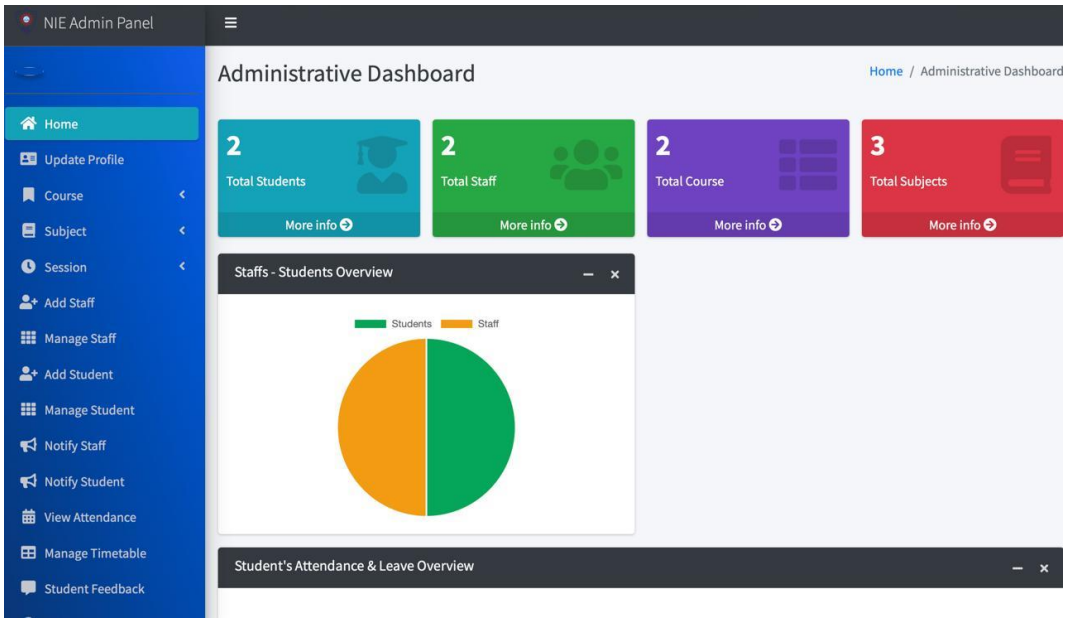

Fig 8.2 – Admin Dashboard

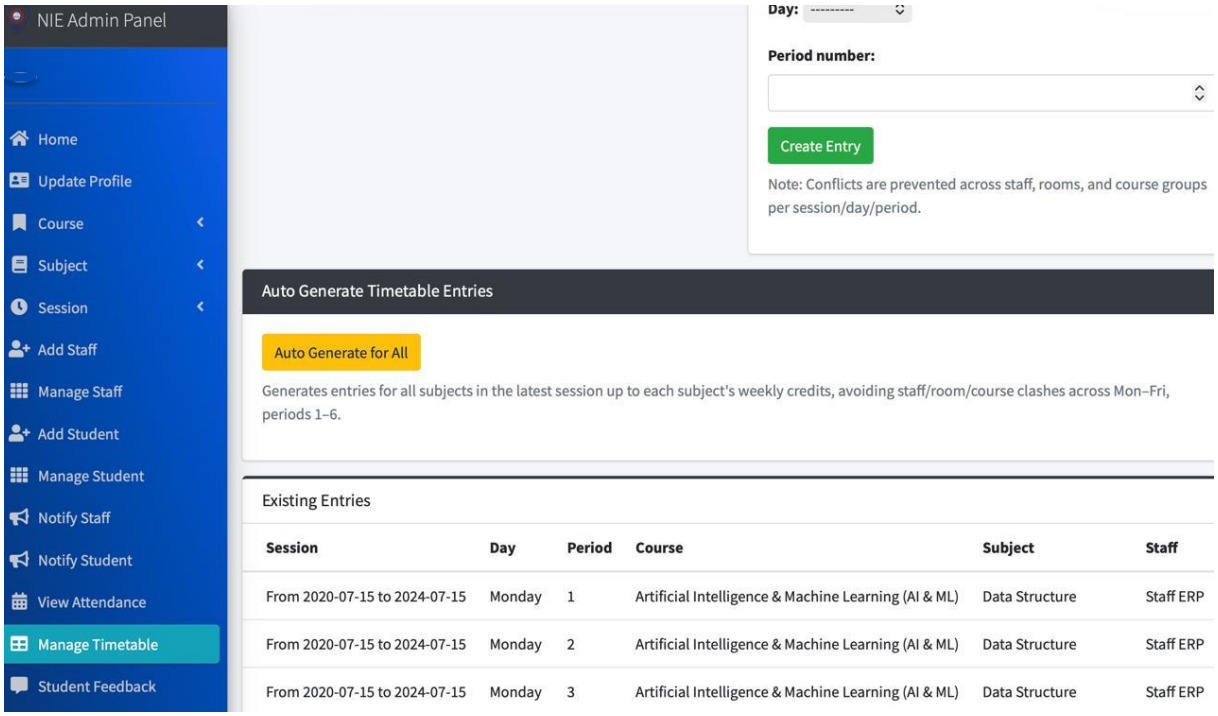

Fig 8.3 – Timetable entry and Automated Generation

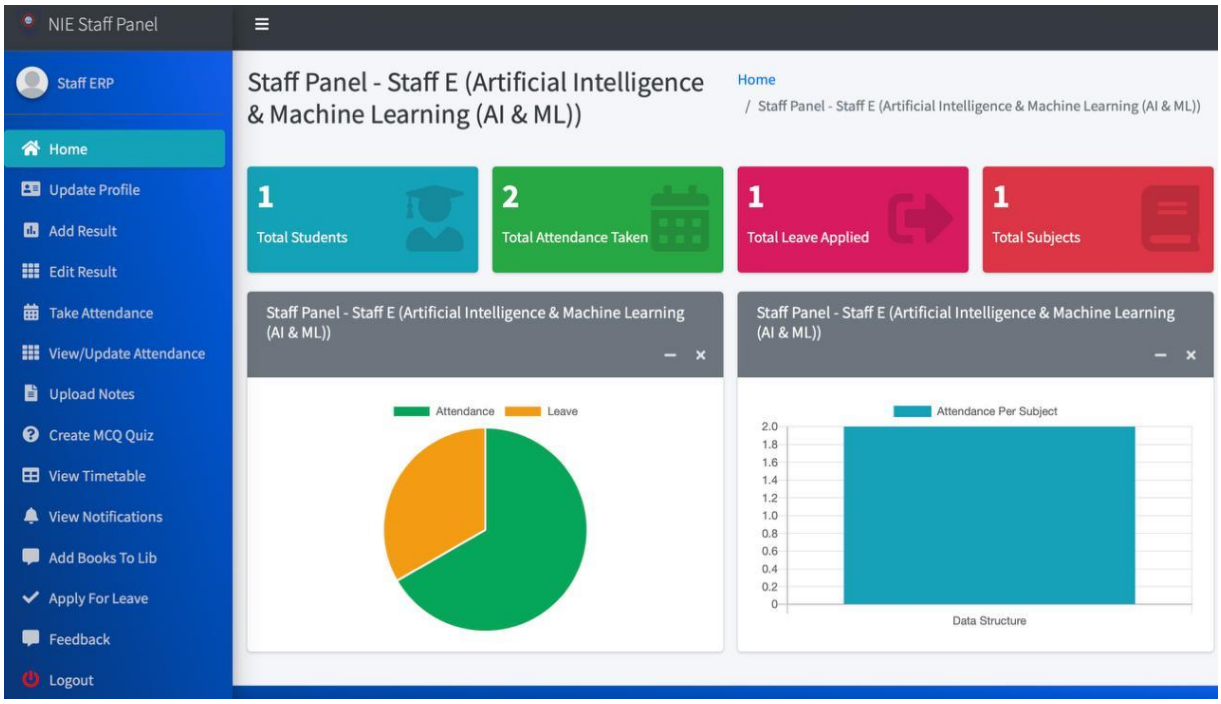

Fig 8.4 – Staff Dashboard

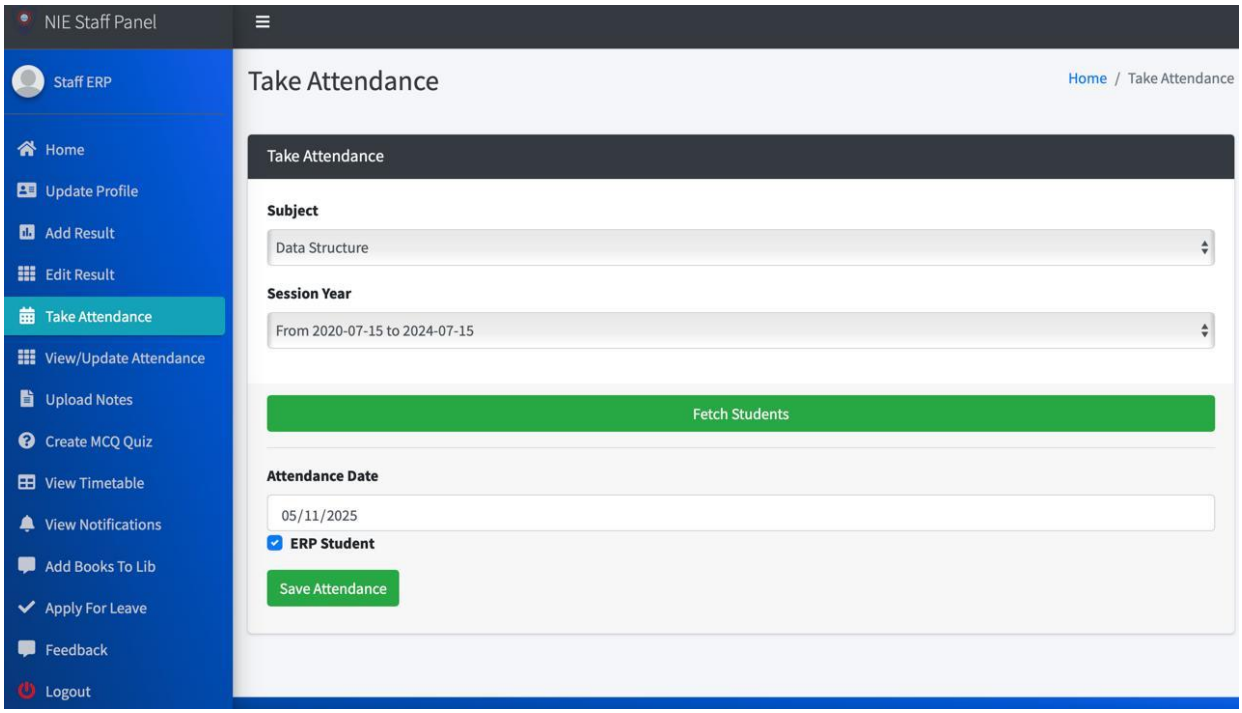

Fig 8.5 – Attendance Call

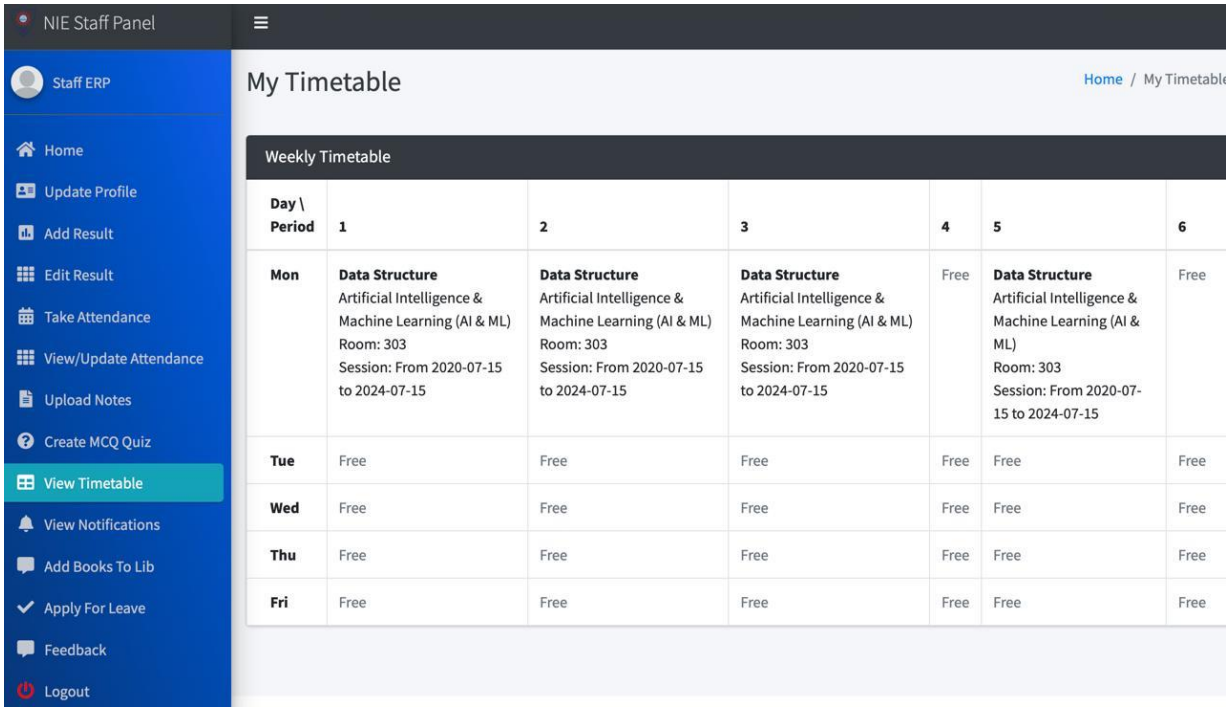

Fig 8.6 – Staff Timetable

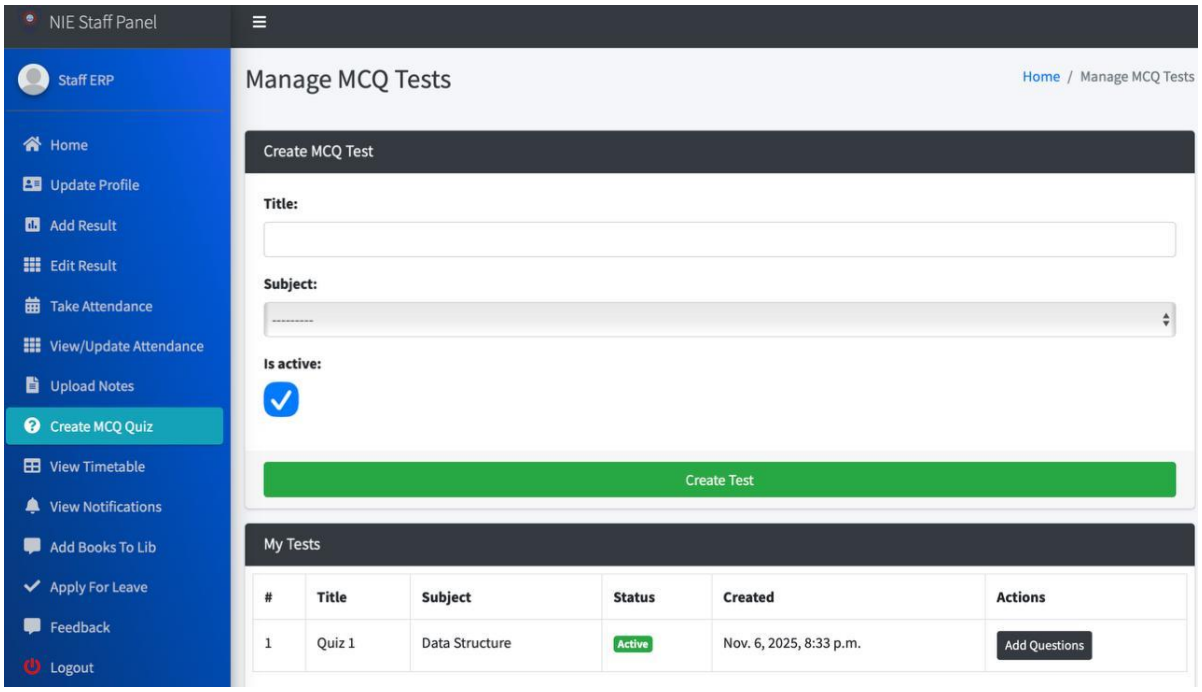

Fig 8.7 – Setting of Quiz

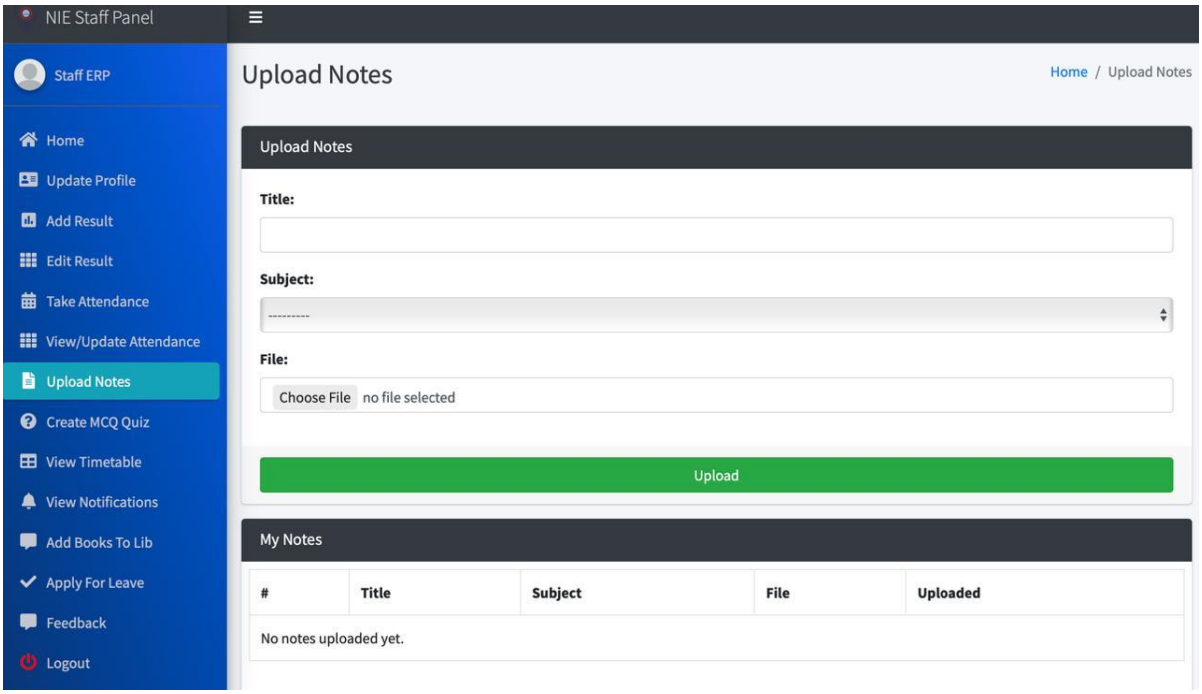

Fig 8.8 – Upload Window for Notes

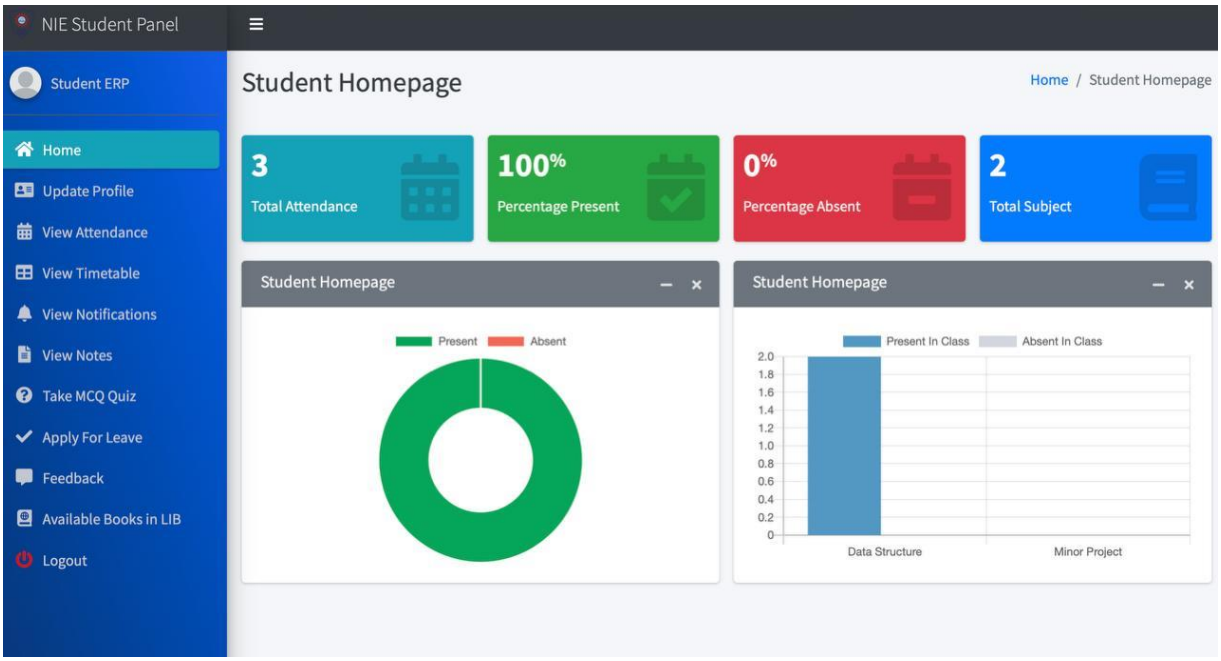

Fig 8.9 – Student Dashboard

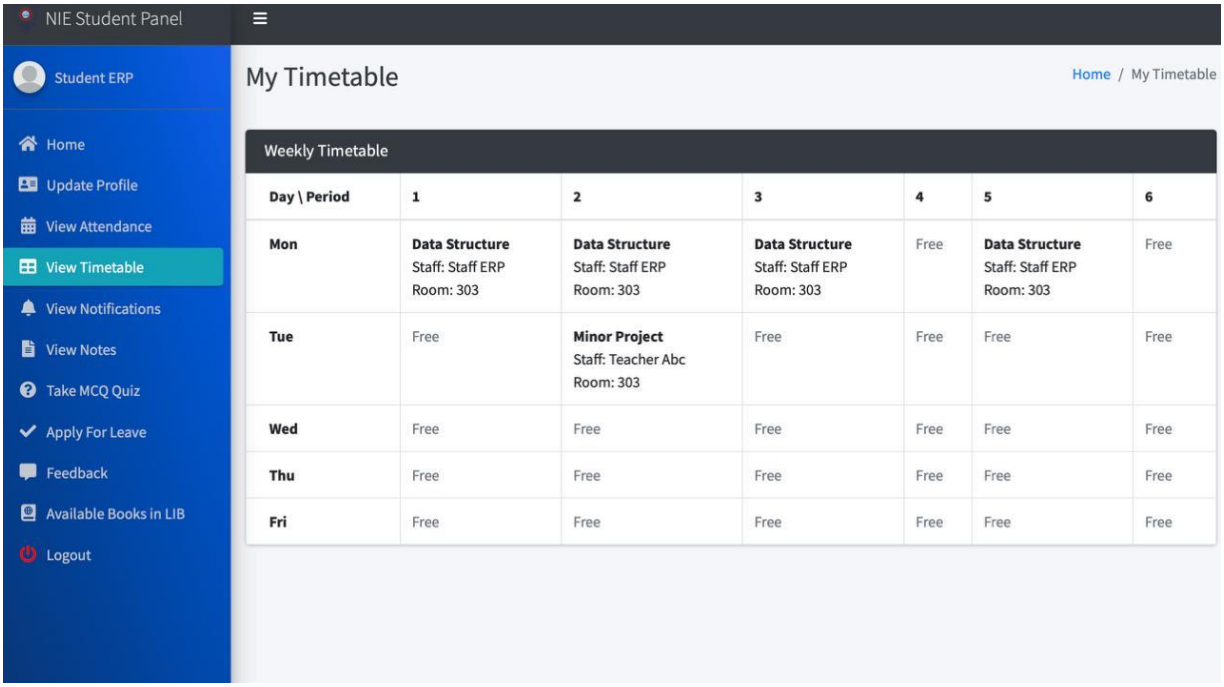

Fig 8.10 – Student Timetable

FUTURE ENHANCEMENT AND CONCLUSION

Conclusion

The Panacea Smart Academic ERP system proves to be an effective solution for streamlining academic and administrative activities within educational institutions. By integrating multiple features—such as automated timetable generation, attendance monitoring, course registration, announcements, fee management, and resource sharing—the platform minimizes manual intervention and significantly reduces the chances of human error.

The system's centralized architecture enables students, faculty, and administrators to interact within a common digital space, improving communication and ensuring timely access to essential information. Real-time notifications and intuitive interfaces further enhance user convenience and operational efficiency.

Overall, Panacea demonstrates how technology can address institutional challenges, promote organized academic planning, and foster a more structured and transparent learning environment. The project establishes a strong foundation for future academic digitalization and can readily be extended to support larger, more complex requirements.

Future Enhancements

- AI-driven Timetable Optimization: Future versions of the system can incorporate AI-based algorithms to automatically refine timetable allocation by considering instructor workload, student course combinations, classroom capacity, and historical usage trends. This would help improve resource utilization and reduce scheduling conflicts.
- Biometric/Location-Based Attendance: To ensure higher authenticity and prevent proxy entry, biometric technologies—such as fingerprint scanning or facial recognition—can be integrated. Alternatively, location-based verification using GPS or Wi-Fi mapping may further enhance attendance reliability.
- Dedicated Mobile Application: A lightweight Android/iOS companion app can be developed to provide real-time access to attendance, notifications, timetables, and study materials. Offline access and cached storage will enhance usability in low-connectivity regions.
- Cloud-Backed Multi-Campus Support: Deploying the system on scalable cloud infrastructure (Azure/AWS/GCP) would facilitate centralized data management across multiple campuses, allowing unified authentication, reporting, and policy enforcement.

REFERENCES

[1] G. Khan and S. Sonawane, "College timetable generation using graph neural networks and reinforcement learning," *Utilitas Mathematica Journal*, 2025. Available: https://utilitasmathematica.com/index.php/Index/article/view/2381

[2] F. K. Mumcu and A. Çebi, "You have a notification: The role of push notifications in shaping students' engagement, self-regulation, and academic procrastination," *Int. J. Educ. Technol. High. Educ.*, vol. 22, no. 1, 2025. doi: 10.1186/s41239-025-00537-x

[3] K. Sajithabanu and S. Mahmootha, "Intelligent college management system with real-time monitoring and visual analytics," *J. Data Min. Manag.*, vol. 10, no. 1, pp. 45–53, 2025. Available: https://matjournals.net/engineering/index.php/JoDMM/article/view/1953

[4] H.-T. Wu, C.-H. Chiu, and H.-W. Chou, "Establish a digital real-time learning system with push notifications," *Journal of Medical Internet Research*, vol. 24, no. 3, p. e34512, 2022. Available: https://pubmed.ncbi.nlm.nih.gov/35250711/

[5] S. Uma, P. Sharvesh, K. Pradeep, R. Sathishkumar, and T. Senthilnathan, "Automatic timetable generation," *IJRASET*, vol. 11, no. 4, pp. 2110–2115, 2023. Available: https://www.ijraset.com/research-paper/automatic-timetable-generation

[6] M. Ahmed and S. Salman, "Cloud-based attendance management system using mobile authentication," *Int. J. Emerg. Technol. Learn. (iJET)*, 2021. doi: 10.3991/ijet.v16i20.25141

[7] D. Park and S. Lee, "AI-assisted academic resource planning for higher education," *Applied Sciences Journal*, 2024. Available: https://www.mdpi.com/

[8] R. Sharma and A. Thomas, "A web-based academic ERP for university management," *Int. J. Comput. Appl.*, 2021. Available: https://www.ijcaonline.org/