PRD — University Portal (Student + Teacher + Admin) with Attendance, Timetable, Analytics & AI Chatbot

1. Product Overview

1.1 Product Name

UniPortal — University Student, Teacher & Admin Portal

1.2 Problem Statement

Universities need a centralized web portal where students can track their academic journey (attendance, timetable, marks, exams, results) and teachers can manage academic operations (attendance marking, timetable, exam creation, marks entry) with strong access control.

Additionally, universities require an **Admin Portal** to configure and manage all master data, users (teachers/students), and system-wide operations with analytics.

1.3 Solution Summary

A web portal that provides:

- Role-based access for **Students**, **Teachers**, and **Admins**
- Student dashboard with **graphs** for marks + attendance + subject-wise performance
- Teacher tools to manage class-related operations, attendance, leave approval, and marks
- Admin portal for managing all users, timetable, exams, branches, sections, subjects and system analytics
- **Leave application** module initiated by students, approved by teachers, and sent via **email from the portal**
- An AI chatbot that answers queries about performance, marks, attendance, timetable (with strict security controls based on role)

---

2. Target Users

2.1 Student

- Logs in to view own academic information
- Needs quick access to: timetable, attendance %, marks/results, leave application

2.2 Teacher

- Logs in to manage students in their assigned classes
- Marks attendance for their classes
- Approves leaves
- Adds marks only for the subjects they handle
- Views analytics of students/classes they are assigned to

2.3 Admin

- Configures the university master data
- Manages users (teachers and students)
- Manages timetables, exams, branches, subjects
- Views full system analytics across all branches/classes

---

3. Goals & Success Metrics

3.1 Goals

- Provide a single portal for student academic tracking
- Reduce manual record keeping (attendance + marks)
- Ensure proper role-based control:
    - student → self data
    - teacher → assigned class + assigned subject
    - admin → full access
- Provide dashboards and analytics for all roles
- Provide chatbot to reduce repetitive queries and improve UX

3.2 Success Metrics

- **100% role-based access compliance** (no unauthorized access)
- Students can access within **2 clicks**:
    - timetable
    - attendance summary

    - marks/results
- Teachers can complete attendance marking within **< 2 minutes per class session**
- Teachers can upload marks for their subject within **< 5 minutes per exam**
- Admin can bulk upload users via CSV successfully with < 1% failure rate
- Email leave workflow works end-to-end with logs
- Dashboards load within **3 seconds** with graphs

---

4. Scope

4.1 In Scope (Must Have)

- Student/Teacher/Admin login
- Academic structure: branches, classes, sections, semesters
- Subjects linked to semester/class/branch
- Teacher assignment to:
    - subjects
    - classes
- Attendance tracking
- Leave application + approval + email sending
- Timetable per class
- Exams and marks entry with strong restrictions
- Student dashboard with graphs
- Teacher dashboard with graphs (assigned scope)
- Admin dashboard with graphs (full scope)
- AI Chatbot with role-based restrictions

4.2 Out of Scope (Optional Enhancements)

- Fees/payment module
- Library management
- Hostel management
- Parent login
- LMS/video classes
- Placement module
- Events/announcements module

---

5. Key User Journeys (End-to-End)

5.1 Student Journey

1. Student logs in
2. Lands on **Student Dashboard**
3. Views:
    - attendance status & percentage
    - marks summary
    - subject-wise performance chart
4. Navigates tabs:
    - Timetable
    - Attendance & Leaves (apply leave + track status)
    - Marks / Results
5. Uses chatbot:
    - "What is my attendance this month?"
    - "Show my marks for Mid 1"

5.2 Teacher Journey

1. Teacher logs in
2. Lands on **Teacher Dashboard**
3. Views:
    - attendance pending (only assigned classes)
    - leave requests pending (only assigned classes)
    - exam/marks work pending
4. Teacher opens a class they handle:
    - marks attendance
    - approves/rejects leaves
    - enters marks **only for assigned subject(s)** and **only for assigned class(es)**

5.3 Admin Journey

1. Admin logs in
2. Lands on **Admin Dashboard**
3. Can manage:
    - branches / sections / classes
    - subjects
    - timetables

    - exams
    - teachers
    - students
4. Bulk uploads users via CSV
5. Views global analytics:
    - branch-wise performance
    - attendance defaulters
    - exam pass rates

---

6. Functional Requirements

6.1 Authentication & Role-Based Access Control

Roles

- **STUDENT**
- **TEACHER**
- **ADMIN**

Role Access Rules (Critical)

▯ Student:

- Must view only own:
    - profile
    - attendance
    - timetable
    - marks/results
    - leave requests
- Chatbot restricted to own data only

▯ Teacher:

- Can view/manage only:
    - assigned classes (classes they attend/handle)
    - assigned subjects
    - assigned students in those classes
- Can enter marks **only for their subject** and **only for their assigned classes**
- Can mark attendance only for assigned classes
- Can approve/reject leave only for assigned classes
- Chatbot restricted to their scope

▯ Admin:

- Can view and manage everything
- Can manage users, masters, exams, timetable
- Full analytics access
- Chatbot full data access

6.2 Academic Structure Module

Supports:

- Branches (CSE, ECE...)
- Sections (A/B/C)
- Classes (year + semester mapping)
- Semesters (1–8)
- Academic year

Requirements

- Admin manages structure
- Teachers can only view relevant structure
- Students can only view their assigned academic mapping

6.3 Users Management (Admin Controlled)

Admin Requirements

Admin must be able to:

- Add teacher accounts
- Add student accounts

- Edit/update teacher/student information
- Deactivate users (soft delete)
- Reset passwords (optional)
- Assign teachers to:
    - classes
    - subjects

CSV Import (Mandatory)

Admin must be able to:

- Upload a CSV file to bulk add students
- Upload a CSV file to bulk add teachers

CSV upload must support:

- Validation errors reporting (invalid email, duplicate roll no, missing fields)
- Import summary:
    - total rows
    - inserted
    - failed
    - skipped duplicates

6.4 Subjects Management

Requirements

- Subjects must be mapped to:
    - branch
    - semester
- Admin manages subject creation
- Teacher assignment must link:
    - teacher → subject(s)
    - teacher → class(es)
- Students can view subjects of their semester/branch only

6.5 Timetable Module (Must Have)

Student View

- Weekly timetable tab
- Shows:
    - day vs period
    - subject
    - teacher

Teacher View

- Teacher can view timetable for their assigned classes
- Teacher can view their personal schedule timetable

Admin View

- Admin can manage timetables for all classes and teachers
- Admin can update timetable entries

6.6 Attendance Module (Must Have)

Student Requirements

- View daily attendance calendar
- View overall attendance %
- View monthly attendance trend graph

Teacher Requirements

- Mark attendance only for classes they attend/handle
- Optional: mark period-wise attendance

Admin Requirements

- Can view attendance reports for all classes
- Can generate attendance defaulters report (<75% or configurable threshold)

6.7 Leave Management Module (Must Have)

Student Requirements

- Apply leave:
  - date range
  - leave type
  - reason
  - attachment optional
- Track status:
  - pending/approved/rejected

Teacher Requirements

- Approve/reject leave requests only from assigned classes
- Add comments

Admin Requirements

- Can view all leave requests (read-only)
- Configure leave types (optional)

Email Notifications (Mandatory)

- Leave applied email → sent to teacher/class coordinator
- Leave decision email → sent to student

6.8 Exams & Marks Module (Core Feature)

Requirements

- System must support exams:
  - Mid 1 / Mid 2 / Semester / Final (flexible)
- Results should be displayed exam-wise and subject-wise

Student View

- Can view:
  - marks per subject for each exam
  - totals and percentages
  - pass/fail status
- Must not view other students' results

Teacher View (Very Important Restriction)

Teachers can:

- Add/edit marks ONLY:
  - for their assigned subject(s)
  - for their assigned class(es)
- Teachers must not be able to enter marks for:
  - other subjects
  - other classes
  - other branches not assigned

Admin View

Admin can:

- Create/manage exams for any class
- Assign exams to semesters/classes
- Publish results (optional)
- Override marks (optional with audit logs)

6.9 Dashboards & Graphs (Must Have)

Student Dashboard

Cards

- Attendance % (overall)
- Attendance this month
- Latest exam total marks

- Best subject
- Weakest subject

Graphs

1. Subject-wise marks chart
2. Attendance trend chart
3. Exam comparison chart
4. Performance index score (marks + attendance)

Teacher Dashboard

Teacher dashboard should show graphs only for assigned scope: **Cards**

- Assigned classes
- Pending attendance entries
- Pending leave requests
- Upcoming exams

Graphs

1. Class avg per subject (their subject only)
2. Pass/fail distribution (their class scope)
3. Low performers list (subject/class scope)
4. Attendance defaulters in assigned classes

Admin Dashboard

Admin gets full portal analytics: **Cards**

- Total students
- Total teachers
- Total classes
- Leaves pending
- Attendance defaulters count

Graphs

1. Branch-wise overall performance
2. Class-wise attendance percentages
3. Subject-wise difficulty/average marks across branches
4. Pass rate per semester exam
5. Student growth trends (optional)

6.10 AI Chatbot Module (Mandatory)

Purpose

Chat assistant to answer:

- marks queries
- attendance queries
- timetable queries
- performance analytics queries

Student Chatbot Rules

□ Allowed:

- "What is my attendance percentage?"
- "Show my marks for Mid 2"
- "Compare my Mid 1 and Mid 2"
- "What is my timetable tomorrow?"
- "Which subject is weak?"

□ Not allowed:

- "Who is topper?"
- "Show marks of roll number X"
- "Class average"

Teacher Chatbot Rules

□ Allowed (limited to assigned scope):

- "Topper in my class for my subject"
- "List students below 40% in my subject"
- "Attendance defaulters in my class"
- "Pass percentage for my class"

Admin Chatbot Rules

□ Allowed:

- all statistics and analytics across entire portal
- cross-branch comparisons
- overall toppers and ranks
- defaulters across all classes

Output Requirements

Chatbot must respond with:

- human readable answer
- optional structured data output for chart rendering

Security Requirement (Critical)

Chatbot must strictly enforce:

- student → only self
- teacher → assigned class + assigned subject only
- admin → full access

---

7. UX / UI Requirements

Navigation Structure

After login show role-based sidebar menu:

Student Sidebar

- Dashboard
- Timetable
- Attendance
- Leave Requests
- Marks / Results
- Chatbot

Teacher Sidebar

- Dashboard
- My Classes
- Attendance
- Leave Approvals
- Marks Entry
- Timetable
- Chatbot

Admin Sidebar

- Admin Dashboard
- Manage Branches
- Manage Classes
- Manage Subjects
- Manage Timetable
- Manage Exams
- Manage Teachers
- Manage Students
- Bulk Upload (CSV)
- Reports & Analytics
- Chatbot

UI Principles

- Clean & minimal design
- Responsive layout
- Filters/search tools for teacher/admin
- Students UI should avoid admin complexity

---

8. Notification Requirements

Email Notifications (Mandatory)

- Leave applied
- Leave approved/rejected

Optional:

- Results published
- Attendance shortage (<75%) warnings
- Exam reminders

---

9. Non-Functional Requirements

9.1 Security

- Authentication required everywhere
- Students must not access other student data
- Teachers restricted to assigned class + subject
- No leakage via chatbot
- Optional: audit logs for edits

9.2 Performance

- Dashboard < 3 seconds
- Charts should load efficiently

9.3 Reliability

- Missing marks should show "Not yet uploaded"
- Missing attendance should show "Not marked"
- Proper handling of network errors

9.4 Maintainability

- Modular design
- Reusable components
- Standardized coding and formatting

---

10. Deliverables

10.1 Application Delivery

- Working portal with role-based access
- Complete modules:
  - timetable
  - attendance
  - leaves + email
  - exams + marks
  - dashboards + analytics
  - chatbot

10.2 Documentation

- Setup instructions
- Role access matrix
- Screenshots + demo video

---

11. Acceptance Criteria

The project is accepted when:

▯ Students can login and view ONLY:

- their timetable
- their attendance + leave requests
- their marks/results
- dashboards with graphs
- chatbot answers restricted to self data

▯ Teachers can login and:

- mark attendance for assigned classes only
- approve/reject leave for assigned classes only

- add marks ONLY:
    - for their subject(s)
    - for their class(es)
- view dashboards limited to their assigned scope
- chatbot restricted to assigned scope

 Admin can:

- manage branches/classes/subjects/exams/timetable
- manage teachers/students
- bulk upload teachers/students via CSV
- view full analytics dashboards and graphs
- chatbot can query entire system

 No unauthorized access or data leakage exists (including chatbot restrictions)

Technology Stack Requirements

Frontend

The frontend must be implemented using the following technologies and libraries:

- **React JS** (core UI framework)
- **shadcn/ui** (UI component system for consistent, modern design)
- **React Query** (server-state management: fetching, caching, refetching, pagination)
- **React Context API** (global app state: auth session, role, user profile, UI preferences)
- **React Router** (recommended) for role-based navigation and routing

Frontend Expectations

- Clean, responsive UI
- Role-based UI rendering:
    - Student UI
    - Teacher UI
    - Admin UI
- Graph/Chart support using a React charting library (e.g., Recharts)

Backend

The backend must be developed using:

- **FastAPI** (Python web framework for API development)
- **PostgreSQL** (primary relational database)
- **JWT Authentication** (access token-based authentication)

Backend Expectations

- Secure authentication and authorization
- Strict role-based access enforcement:
    - Students can access only their data
    - Teachers can access only assigned class + subject data
    - Admin has full access
- Proper validations, Pagination and consistent error handling
- Logging and monitoring-friendly structure (recommended)

Important Note to Internship Candidates

The requirements mentioned in this document are provided as a **reference baseline** and should be treated as **mandatory product requirements** for the internship project.

Internship candidates are expected to carefully review all the requirements and prepare a complete **Solution & Design Document** that includes (but is not limited to) the following:

Mandatory Submission Requirements

Candidates must submit a detailed document covering:

1. **Database Schema Design**

    - Tables / entities required
    - Relationships between entities
    - Constraints and validations
    - Role-based access considerations in schema design

2. **Screen List + UI Planning**

    - List all screens required for:
        - Student portal
        - Teacher portal
        - Admin portal

    - Define navigation structure (sidebar/tabs)

3. **User Journeys**

    - Define end-to-end user flows for each role, such as:
        - Student: login → dashboard → timetable → attendance → marks → chatbot
        - Teacher: login → class view → attendance → marks entry → leave approvals
        - Admin: login → manage users → manage timetable/exams → analytics

4. **Screen-wise Functionalities**

    - For every screen, clearly mention:
        - What data is shown
        - What actions are possible
        - What validations and access restrictions apply
        - Role-based conditions (what should be visible/hidden)

5. **Implementation Approach**

    - Clear explanation of how the candidate plans to build the project:
        - frontend structure and state management
        - backend architecture/modules
        - authentication and authorization logic
        - handling attendance, marks, leave workflows
        - chatbot approach and access control

6. **API Requirements**

    - Define the APIs required per screen/module
    - Each API should mention:
        - purpose
        - input payload
        - output response structure
        - permission required (student/teacher/admin)
    - Candidates should prepare a screen-to-API mapping

7. **Overall Solutioning**

    - Architecture diagram (optional but recommended)
    - Assumptions and trade-offs
    - Edge cases and failure handling
    - Security considerations (data privacy + role restrictions)

Expected Outcome

Candidates should deliver a well-structured, professional solution document that demonstrates:

- product understanding
- system design thinking
- secure role-based portal planning
- ability to translate requirements into implementable technical design