Trainee Management Portal

Technical Design Document

1. Introduction

The **Trainee Management Portal** is designed to efficiently plan, manage, and track trainee schedules and engagement. The system supports various user roles such as Coaches, Trainers, Mentors, and CRs, with role-based access control. The portal offers features like real-time updates, notifications, and automated attendance tracking to streamline management.

---

2. Objectives

- Manage and track trainee schedules effectively.
- Provide role-based access control.
- Ensure real-time updates for session attendance and progress.
- Allow seamless interaction between trainees and key stakeholders.

---

3. Features Overview

Core Functionalities

Role	Features
Lead	CRUD on Coaches
Coach	Assign CR, Trainer, Mentor; CRUD Connect Updates; TimeTable Management
Trainer	Contribution Hours, Topics Taught, Trainer Connect Updates
Mentor	CRUD Mentor Connect Updates

CR	Attendance Management, Connect Updates, Cohort Read Access
All Roles	Read Cohort Details, TimeTables, Calendar

Event Types

- **Trainer Session (Daily)** - Virtual/In-person
- **Coach Connect (Daily)** - Virtual/In-person
- **Mentor Connect (Weekly)** - Virtual/In-person
- **BH Session (Weekly)** - Virtual/In-person
- **Leadership Connect (Monthly)** - Virtual/In-person

Dashboards & Reports

- Personalized Calendar View
- Cohort Timetable View
- Attendance Reports
- Contribution Reports
- Connect Updates Dashboard (Daily, Weekly, Monthly Connects Overview)
- **Activity Log Dashboard** (Audit logs, User activity tracking)

Additional Features

- **Role Switcher:** For users with multiple roles (Coach + CR, etc.)
- **Notifications & Reminders:** Automated email or in-app reminders for upcoming connects
- **Export Timetables:** Download cohort timetables as PDF/CSV
- **Commenting & Remarks Section:** For each connect/session update
- **Audit Logs:** Track all major changes and updates by user and timestamp
- **Search & Filters:** For all lists, cohorts, reports, and updates

- **Bulk Upload (CSV):** Attendance and schedule entries
- **Video Conferencing Integration:** (Zoom/Google Meet links)
- **Dark Mode Support:** For accessibility and user preference

---

4. System Architecture

High-level Architecture

- **Frontend:** React.js + TailwindCSS + FullCalendar + Chart.js (for Reports)
- **Backend:** Spring Boot (Java)
- **Database:** PostgreSQL
- **Authentication:** JWT + Role-based Authorization (Spring Security)
- **Deployment:** Render.com / Railway.app
- **CI/CD:** GitHub Actions
- **Email Notifications:** (Optional, SendGrid)
- **Video Conferencing Links:** (Optional, Zoom API Integration)

Component Diagram

- **User Interface Layer**
    - Dashboards
    - Calendar Views
    - Reports
    - Activity Logs
- **Application Layer**
    - Auth Service

    - Cohort Management Service
    - Event/Connect Management Service
    - Notification Service
- **Data Layer**
    - PostgreSQL Database
- **External Integrations**
    - Email Notifications (Optional)
    - Video Conferencing Links (Optional)

---

5. Database Design (Schema Overview)

- **users (id, name, email, password, role, created_at)**
- **cohorts (id, name, domain, start_date, end_date, created_by)**
- **schedule (id, cohort_id, event_name, date, time, type, remarks, created_by)**
- **attendance (id, cohort_id, cr_id, date, status, remarks, created_by)**
- **contributions (id, trainer_id, cohort_id, hours, topics, date, created_by)**
- **connect_updates (id, cohort_id, type, date, remarks, updated_by)**
- **audit_logs (id, user_id, event_type, description, timestamp)**
- **notifications (id, user_id, message, type, is_read, created_at)**

---

6. Security Design

- JWT-based Authentication
- Role-based Authorization (Spring Security)

- Input Validation (Backend & Frontend)
- Password Hashing (BCrypt)
- Secure API Endpoints with HTTPS
- Audit Logging (Critical actions by user)
- Rate Limiting and CORS Configuration

---

7. API Design (Sample Endpoints)

- **Auth**
    - POST /api/auth/login
    - POST /api/auth/register
- **Cohorts**
    - GET /api/cohorts
    - POST /api/cohorts
- **Schedule**
    - GET /api/schedule/{cohortId}
    - POST /api/schedule
- **Attendance**
    - GET /api/attendance/{cohortId}
    - POST /api/attendance
- **Contributions**
    - GET /api/contributions/{trainerId}
    - POST /api/contributions
- **Connect Updates**

    - GET /api/connects/{cohortId}
    - POST /api/connects
- **Activity Logs**
    - GET /api/logs
- **Notifications**
    - GET /api/notifications
    - POST /api/notifications/mark-read

---

8. Non-Functional Requirements

- **Scalability:** Supports multiple cohorts and large user base
- **Reliability:** 99.9% uptime
- **Security:** Follows OWASP top 10 security guidelines
- **Usability:** Intuitive UI for all roles
- **Maintainability:** Modular codebase, clean architecture
- **Accessibility:** Support for dark mode, screen readers

UI Design: Dashboard Architecture

🧭 General Layout for All Dashboards

Section	Description
Top Navbar	Contains user info, role, profile picture, logout dropdown

Sidebar	Role-based navigation: Dashboard, Cohorts, Schedule, Assignments, Settings
Main Panel	Dynamic content based on selected menu
Modals	CRUD operation dialogs (Add/Edit/Delete entities, schedule updates)

---

1. Lead Dashboard (ROLE_LEAD)

UI Page Components

Cohort Management View

- **Components**:
  - `<Table />` with columns: `Cohort Name`, `Start Date`, `Status`, `Trainer`, `Mentor`, `BH Trainer`, `Coach`, `Actions`
  - `<Button />` for "Assign Roles", "Edit Schedule", and "View Details"
- **Triggers**:
  - Clicking "Assign Roles" opens a modal:
    - Drop-downs to select Trainer, Mentor, BH Trainer
    - PUT: `/api/cohorts/{id}/assign/{role}`
  - "Edit Schedule" opens a calendar/schedule grid editor
    - PUT: `/api/schedule/{scheduleId}`
- **Interactions**:
  - Inline filtering, search, pagination
  - Badge indicators for schedule completion or delayed cohorts

Trainer/Mentor/BH Trainer/Coach List Views

- Tabs for each entity:

    - Each contains:
        - `<Table />` with basic info (Name, Contact, Expertise, Cohorts Assigned)
        - `<Button />` for View Profile, Assign to Cohort
- View Profile expands accordion or opens a modal
- **API**:
    - GET `/api/lead/trainers`, `/mentors`, `/bhtrainers`, `/coaches`

Schedule Management

- **Grid calendar view (like Google Calendar)** per cohort
- Ability to:
    - Drag & drop sessions
    - Edit topic name, duration
    - Assign responsible trainer
    - Add a mentor/co-trainer
- **API**:
    - GET `/api/cohorts/{id}/schedule`
    - PUT `/api/schedule/{id}`

User Experience Elements

- Toasts for success/failure
- Validation for role assignment (no duplicate role assignment to the same cohort)
- Cohort cards with live status color-coded (green – active, red – overdue, blue – upcoming)

---

2. Coach Dashboard (ROLE_COACH)

UI Page Components

Assigned Cohorts View

- **Table Columns**:
  - `Cohort Name`, `Duration`, `Trainer`, `Mentor`, `Schedule`, `Actions`
- **Features**:
  - View full cohort details
  - Update schedule (similar UI as Lead)
  - Reassign any role (Trainer, Mentor, BH Trainer)
- **Triggers**:
  - GET: `/api/coach/cohorts`
  - GET: `/api/cohorts/{id}`
  - PUT: `/api/cohorts/{id}/assign/{role}`

Schedule Editor

- Editable grid per cohort
- Inline addition of new session
- Assign mentor, trainer from available list
- **API**:
  - PUT `/api/schedule/{id}` or `/api/cohorts/{id}/schedule`

Role Assignment Modal

- Form with searchable drop-downs for:
  - Trainer

    - Mentor
    - BH Trainer
- Dynamic validation: Only show users with that role
- Assign or reassign with confirmation modal

UX Enhancements

- History tracker: See who updated which schedule and when
- Notification panel: Show pending assignments, incomplete sessions
- Inline timeline for cohort progress

---

3. Trainer Dashboard (ROLE_TRAINER)

UI Page Components

Assigned Cohorts Table

- **Columns**:
    - `Cohort Name`, `Current Phase`, `Mentor`, `Schedule`, `Trainer Connect Status`
- Cohort Connect column shows:
    - if trainer connect submitted
    - if pending
- **API**:
    - GET `/api/trainer/cohorts`

Trainer Connect Submission View

- **Form Fields**:

    - Date (picker)
    - Topics Covered (textarea)
    - Trainer Notes
    - Duration
    - Upload Materials (file input)
- **Actions**:
    - Submit Trainer Connect
        - POST or PUT `/api/trainerconnect/{id}`
    - Mark Acknowledgment
        - PUT `/api/trainerconnect/{id}/acknowledge`
- **Validations**:
    - Date should not be future
    - Content is mandatory
    - One connect per day per cohort

View Past Connects

- Accordion view grouped by date
- Expands to show:
    - Notes, attachments, session health
    - Mentor remarks (if any)

UX Enhancements

- Calendar showing connects in green/red
- Toasts & inline form errors
- Auto-save drafts before submission

Security & Routing (All Dashboards)

Layer	Protection
Frontend Routes	Role-based protected routes (`/lead`, `/coach`, `/trainer`)
Backend APIs	Secured using Spring Security with JWT tokens
Sensitive Actions	POST/PUT requests protected via CSRF or secure tokens

Event Triggers Summary

Action	Triggered From	Resulting Effect
Assign Role to Cohort	Lead/Coach role assign modal	PUT API → Cohort updated
Submit Trainer Connect	Trainer dashboard	POST API → Record stored, UI updated
Update Schedule	Calendar/Grid Editor	PUT API → Timeline updates
Acknowledge Connect	Trainer Connect checkbox	PUT API → Flag updated

9. Day-wise Time Plan and Milestones

Day-wise Time Plan and Milestones

Week 1: April 28 - May 5

Goal: Requirement gathering, feature finalization, and wireframe creation.

Date	Activity	Milestone/Target
April 28	- Project kick-off meeting - Finalize requirements document	Requirement gathering complete
April 29	- Discuss the core features and user roles - Identify system interactions	Features list and system flow finalized
April 30	- Begin wireframe design for login page, dashboards, and cohort details	Basic wireframes for core pages (login, dashboard) completed
May 1	- Continue wireframe design for additional features (e.g., attendance, contribution)	Detailed wireframe design for user-specific pages
May 2	- Review wireframes with stakeholders - Revise wireframes based on feedback	Feedback incorporated into the wireframe

---

Week 2: May 6 - May 12

Goal: Database schema design, API design, and basic backend setup.

Date	Activity	Milestone/Target
May 6	- Design database schema (users, cohorts, schedule, etc.)	Database ERD created
May 7	- Design API endpoints for authentication and user management	API contract finalized for authentication and user endpoints
May 8	- Implement user registration and login functionality (JWT Auth)	User authentication API implemented
May 9	- Design API endpoints for cohorts, schedules	API contract finalized for cohorts and schedules
May 10	- Develop backend services for cohort management	Cohort management service implemented

---

Week 3: May 13 - May 19

Goal: Frontend setup, integration of calendar and timetable, and role-based access control.

Date	Activity	Milestone/Target
May 13	- Set up React.js frontend framework - Install TailwindCSS for styling	Basic frontend setup completed
May 14	- Implement user login page (frontend)	Login page integrated with backend

	- Connect login API	
May 15	- Design and implement the calendar view for cohorts	Calendar UI for cohort view integrated
May 16	- Design and implement timetable view for each cohort	Timetable UI integrated with backend
May 17	- Implement role-based navigation (Lead, Coach, Trainer)	Role-based UI navigation completed

---

Week 4: May 20 - May 26

Goal: Backend functionality for connect updates, attendance tracking, and integration with frontend.

Date	Activity	Milestone/Target
May 20	- Develop backend services for connect updates	Connect updates backend logic implemented
May 21	- Implement frontend for connect updates	Connect updates UI integrated
May 22	- Develop backend functionality for attendance tracking	Attendance management service implemented
May 23	- Integrate frontend with backend for attendance	Attendance UI fully integrated

May 24	- Set up video conferencing link integration (Zoom/Google Meet)	Video conferencing links integrated

---

Week 5: May 27 - June 2

Goal: Testing, bug fixing, and finalizing features.

Date	Activity	Milestone/Target
May 27	- Begin unit testing for APIs (authentication, cohorts, schedule)	Unit testing initiated
May 28	- Complete frontend testing (login, dashboard, timetable)	Frontend testing completed
May 29	- Integrate and test attendance and contribution tracking features	Full testing of attendance and contribution modules
May 30	- Conduct integration testing for the whole system	Integration testing completed
May 31	- Bug fixing based on testing feedback	Critical bugs fixed

---

Week 6: June 3 - June 9

Goal: Deployment preparation, final bug fixes, and documentation.

Date	Activity	Milestone/Target

June 3	- Set up CI/CD pipeline for automated deployment	CI/CD pipeline established
June 4	- Deploy application to Render.com or Railway.app	Application deployed to the staging environment
June 5	- Perform final testing on the staging environment	Final testing in the staging environment completed
June 6	- Deploy to production environment	Production deployment completed
June 7	- Finalize user documentation	User manuals and API documentation finalized

---

Week 7: June 10 - June 12

Goal: Review, project completion, and final handover.

Date	Activity	Milestone/Target
June 10	- Review all features and confirm all are working	All features confirmed working
June 11	- Complete project handover documentation	Documentation completed
June 12	- Final stakeholder meeting and project closure	Project successfully closed

---

Technical Milestones Overview

- **Week 1 (April 28 - May 2)**: Complete requirements gathering and wireframe designs.
- **Week 2 (May 5 - May 9)**: Finalize database schema and API design.
- **Week 3 (May 11 - May 15)**: Set up the frontend and integrate core features.
- **Week 4 (May 17 - May 21)**: Implement connect updates and attendance features.
- **Week 5 (May 21 - May 25)**: Testing and bug fixing.
- **Week 6 (May 27 - May 31)**: Deployment, final testing, and documentation.
- **Week 7 (June 2 - June 6)**: Project review, handover, and closure.

Project Monitoring & Verification

- **Code Reviews**: Regular code reviews will be conducted every week to ensure that the code quality is maintained and follows the project guidelines.
- **Daily Standups**: Team members will participate in daily standups to provide status updates, blockers, and next steps.
- **Weekly Milestone Check**: Every week, the project manager will verify that the milestones for the week are met and plan the next phase accordingly.
- **Final Testing**: After deployment, a comprehensive round of user acceptance testing (UAT) will be performed to ensure everything is functioning as expected.

3. Project Monitoring & Verification

- Code Reviews: Regular code reviews will be conducted every week to ensure that the code quality is maintained and follows the project guidelines.

- Daily Standups: Team members will participate in daily standups to provide status updates, blockers, and next steps.

- Weekly Milestone Check: Every week, the project manager will verify that the milestones for the week are met and plan the next phase accordingly.

- Final Testing: After deployment, a comprehensive round of user acceptance testing (UAT) will be performed to ensure everything is functioning as expected.