SchoolTrackerFMI

12.27.2023

Aleksandar Kuzmov
Plovdiv University FMI

Overview

SchoolTrackerFMI is an educational management system designed to streamline and enhance the administrative and academic processes within educational institutions. The platform serves as a centralized hub for parents, students, teachers, and administrators, facilitating efficient communication, attendance tracking, and academic performance monitoring.

Goals

User Management:

- **Parent Portal**: Parents can view their children's attendance records, exam schedules, and academic performance.
- **Student Dashboard**: Students can access their attendance, exam schedules, and exam results.
- **Teacher Interface**: Teachers can manage classroom details, attendance records, and input exam results.

Attendance Tracking:

- Real-time tracking of student attendance with options to mark as **"attendant"** or **"non-attendant"** and add remarks if needed.
- Parents receive notifications for their children's attendance status.
- Course and Classroom Management:
- Admins can assign courses to classrooms, allocate teachers to specific courses, and manage course details.
- View classroom schedules and course outlines.

Exam Scheduling and Results:

Schedule different types of exams and manage exam details.

Record and display exam results for students, including grades and additional remarks.

Target Users:

Parents: Monitor children's attendance, exam schedules, and academic progress.

Students: Access personal attendance records, exam schedules, and exam results.

Teachers: Manage classroom details, attendance, and input exam results.

Administrators: Oversee overall system functionality and manage user permissions.

Milestones

I. Create Database structure in MySQL

1. Database Design:

Objective: Plan the structure and relationships among various tables.

Tasks:

Identify key entities: Parents, Students, Teachers, Courses, Classrooms, Exams, etc.

Define fields for each table, specifying data types, constraints, and relationships.

Draft an entity-relationship diagram (ERD) to visualize the database structure.

2. SQL Scripting:

Objective: Translate the database design into SQL scripts for execution.

Tasks:

Write SQL commands for creating tables, defining primary and foreign keys, establishing relationships, and adding constraints.

Include commands for creating junction tables (if needed) to manage many-to-many relationships.

Incorporate initial data population (optional) for some tables (e.g., Courses).

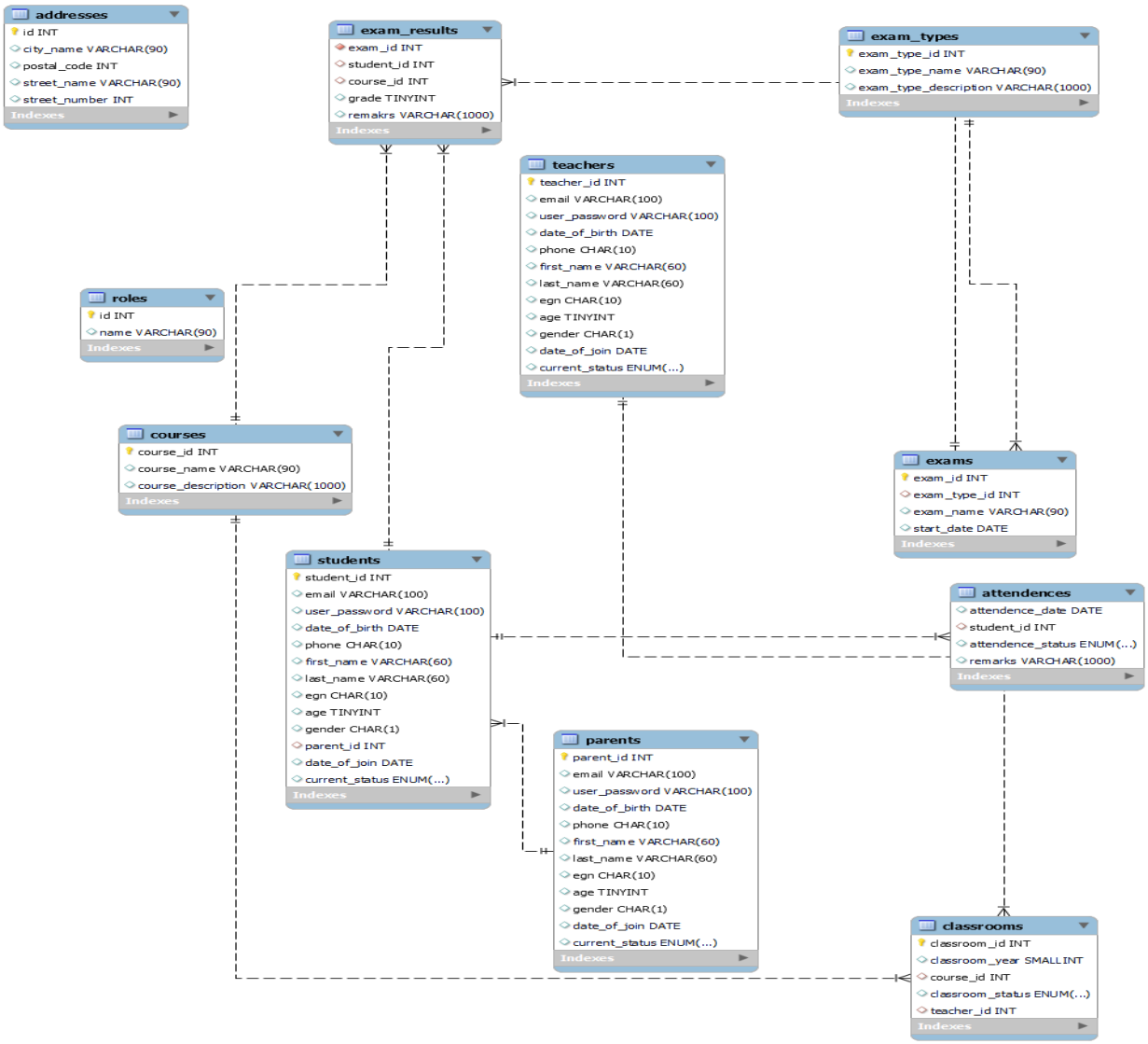
3. Implementation and Execution:

Objective: Execute the SQL script to create the database and its tables.

Tasks:

Execute the SQL script within the chosen database management system (e.g., MySQL).

Verify the successful creation of tables and relationships by checking the database schema.



II. Spring Boot Project Setup

Project Setup and Configuration:

Objective: Create a Spring Boot project and configure the development environment.

Tasks:

- Set up a Spring Boot project using Spring Initializr or a similar tool.
- Configure dependencies for web, data access, security, and other necessary modules.
- Establish database connectivity by configuring data source properties.

2. Entity Classes and Data Models:

Objective: Create Java classes representing entities that map to database tables.

Tasks:

- Develop entity classes (e.g., Parent, Student, Teacher) with annotations for mapping to database tables.
- Define relationships (One-to-One, One-to-Many, Many-to-Many) between entities using JPA annotations.
- Implement data models for core entities, ensuring alignment with the database schema.

3. Repository and Data Access Layer:

Objective: Implement the data access layer for interacting with the database.

Tasks:

- Create repository interfaces extending JpaRepository to handle CRUD operations for each entity.
- Define custom query methods if needed for complex data retrieval.
- Implement services to encapsulate business logic, utilizing repositories for data access.

4. Controllers and RESTful APIs:

Objective: Develop controllers to handle HTTP requests and expose RESTful endpoints.

Tasks:

- Create controller classes with mapping annotations to define endpoints for various functionalities (e.g., user management, attendance tracking).
- Implement methods within controllers to handle requests, interact with services

5. Security Implementation:

Objective: Secure endpoints and manage authentication/authorization.

Tasks:

- Configure Spring Security to manage authentication and authorization.
- Define security configurations to restrict access to certain endpoints based on user roles.
- Implement authentication mechanisms (e.g., JWT tokens) for secure API access.