Here is a detailed point-wise explanation of the whole project:

1. **Project Aim:**
   * The project is a Student Management System designed to manage students and courses.
   * It provides RESTful APIs and a simple web UI to perform CRUD operations on students and courses.
   * The system uses an in-memory H2 database for data storage.
2. **Project Structure:**
   * **Model Layer:** Contains entity classes Student and Course representing database tables.
   * **Repository Layer:** Interfaces StudentRepository and CourseRepository extend Spring Data JPA repositories for database operations.
   * **Service Layer:** StudentService and CourseService contain business logic and interact with repositories.
   * **Controller Layer:**
     + REST controllers StudentController and CourseController expose API endpoints for CRUD operations.
     + View controllers StudentViewController and ViewController serve Thymeleaf templates for frontend UI.
   * **Templates:** Thymeleaf HTML templates students.html and courses.html render data in tabular format.
   * **Configuration:** application.properties configures the H2 database and other Spring Boot settings.
   * **Build:** Managed by Maven with dependencies for Spring Boot, Spring Data JPA, Web, Thymeleaf, H2, and Lombok.
3. **Working and Workflow:**
   * The application starts with StudentManagementSystemApplication class.
   * Data is stored in the H2 in-memory database.
   * REST API endpoints allow clients to create, read, update, and delete students and courses.
   * The frontend UI pages fetch data from the service layer via controllers and display it using Thymeleaf templates.
   * Users can view lists of students and courses by navigating to /students and /courses URLs.
   * Thymeleaf templates dynamically render data passed from controllers into HTML tables.
   * The system supports easy extension for additional features or UI enhancements.
4. **Technologies Used:**
   * Java 17, Spring Boot 3.5.3
   * Spring Data JPA for ORM
   * H2 in-memory database
   * Thymeleaf for server-side rendering
   * Maven for build and dependency management
   * Lombok for reducing boilerplate code
5. **How to Use:**
   * Run the application using mvn spring-boot:run.
   * Access REST APIs at http://localhost:8080/api/students and http://localhost:8080/api/courses.
   * Access frontend UI at http://localhost:8080/students and http://localhost:8080/courses.
   * Use H2 console at /h2-console to view and manage database data.