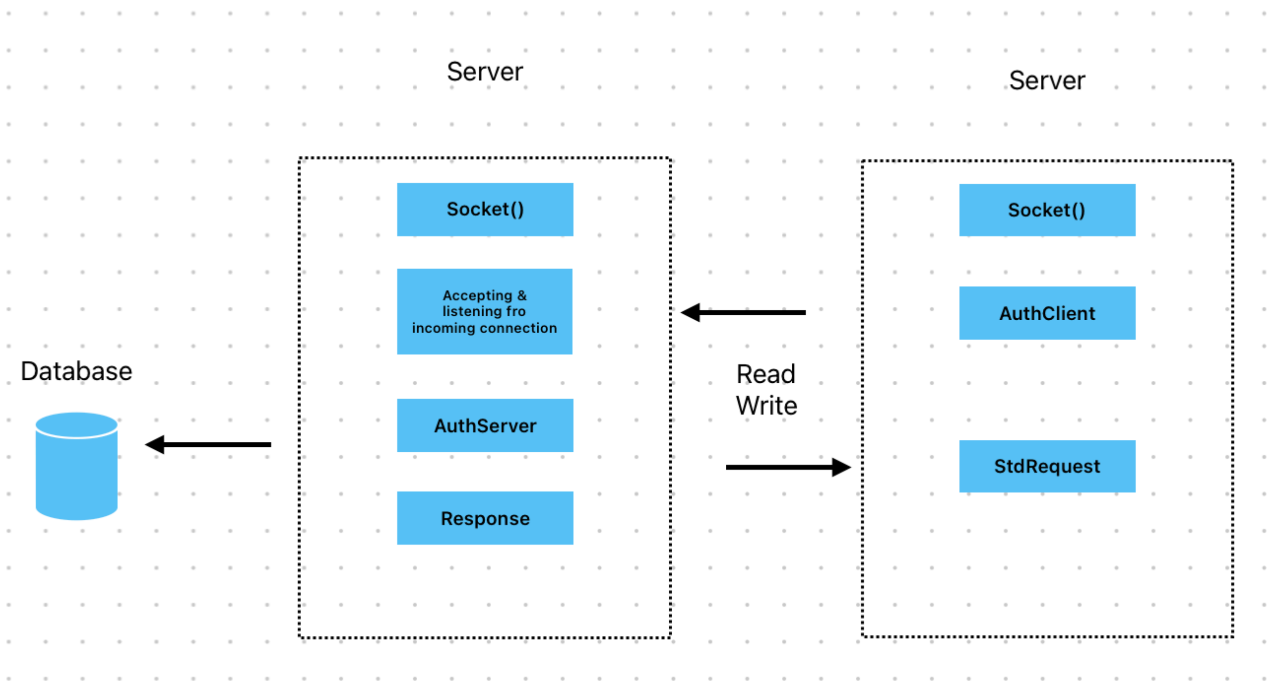
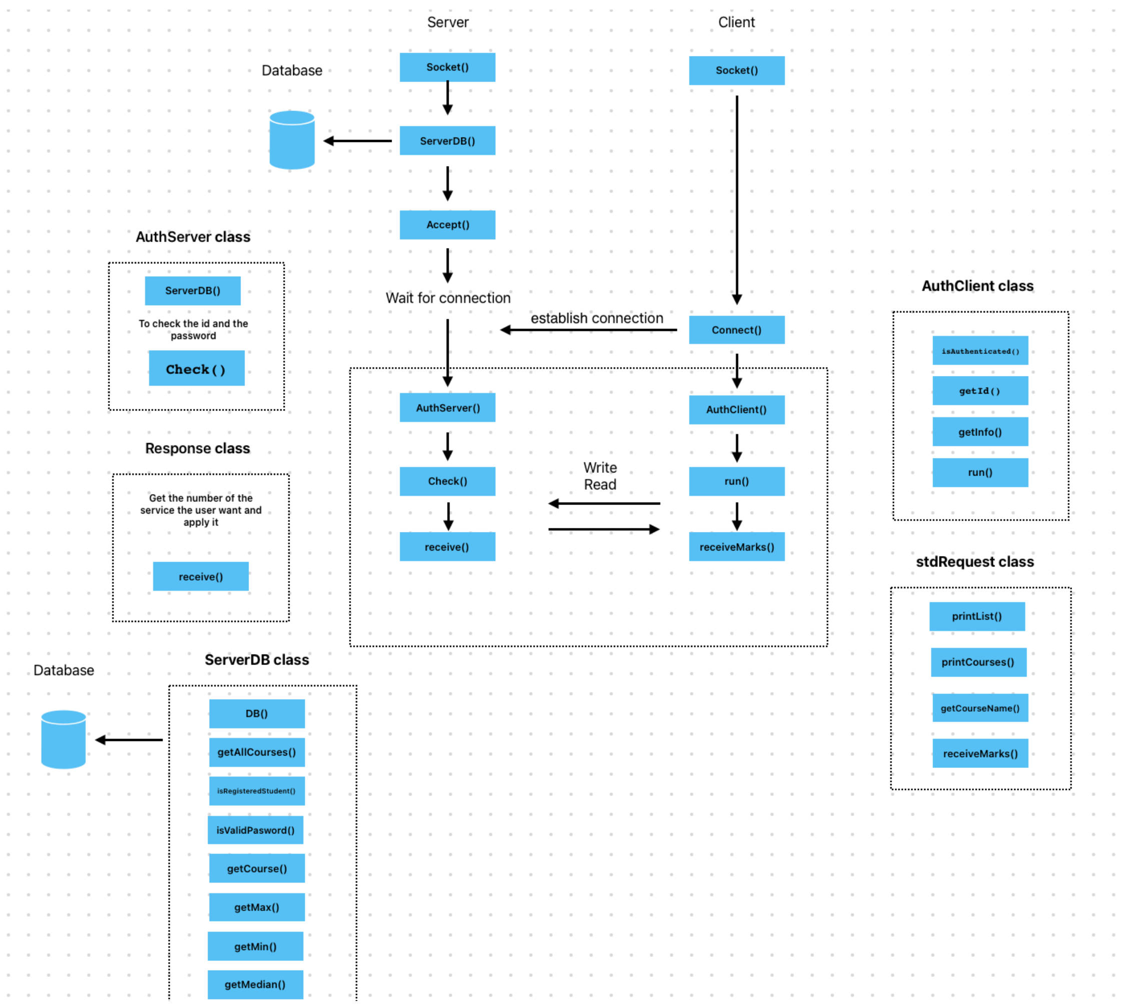
Alaa Abdelqader
Student Grading System Assignment

Part 1: Command-line / Sockets and JDBC Backend Implementation.





simple client-server system that allows a client to connect to a server, authenticate, and perform various operations related to a database of student grades

Technology Choices:

- **MySQL Database.**
- **JDBC:** Java Database Connectivity (JDBC) is used for interacting with the MySQL database

Client Authentication: The client initiates an authentication process with the server by sending an authentication request(via run() funtion in the Authclient class). The server verifies the client's credentials (student ID and password) against a database and responds accordingly

Database Query: After successful authentication, the client can request various operations related to the database, such as retrieving all marks in all courses getting the mark in a specific course calculating average maximum minimum & median marks for a specific course

Database Query Algorithms: diff algorithms are used to calculate statistics such as average, maximum, minimum, and median marks for a specific course. These algorithms involve SQL queries to fetch relevant data from the database

Important Design Decisions:

Threaded Server: The server implementation uses a threaded approach to handle multiple client connections concurrently. This to ensure that the server can handle multiple clients simultaneously without blocking

Error Handling: The code includes error handling mechanisms to deal with exceptions that may occur during socket communication database queries or authentication processes

Analytical Thinking:

1.Client-Server Architecture:

Strengths:

- **Scalability:** Both client and server components are implemented as separate entities.
- **Modularity:** The separation of client and server logic facilitates easier maintenance & updates to each component independentlly
- **Concurrency:** The threaded server implementation enables handling multiple client connections concurrently, improving performance and responsiveness.

2. Authentication Process:

Strengths:

- **Security:** Authentication helps ensure that only authorized users can access the system, enhancing security.

Potential Vulnerabilities**:** there might be vulnerabilities such as brute-force attacks or session hijacking :(

3. Database Interaction:

Strengths:

- **Data Management:** Integration with a database allows for efficient storage, retrieval, and management of student grade information.

Weaknesses:

- Database Dependencies

Challenges and Solutions:

1.Error Handling :

Challenge: Dealing with errors, failures, and unexpected conditions
Solution: Implement error handling mechanisms

2.Scalability and Performance:

Challenge: Making sure that the system can deal with more people using it at the same time without slowing down.
Solution:

- Create the system with growth in mind, using methods like spreading the work across many servers, keeping them grouped together … Make sure to handle database connections smartly to avoid wasting resources. Check & improve the most important parts of the code, database requests and how data travels over the network to make everything faster.

Security Considerations:

Security Measures:

- The system requires clients to authenticate themselves using a username (student ID) & (password)
- The server verifies the client’s credentials before granting access to the data

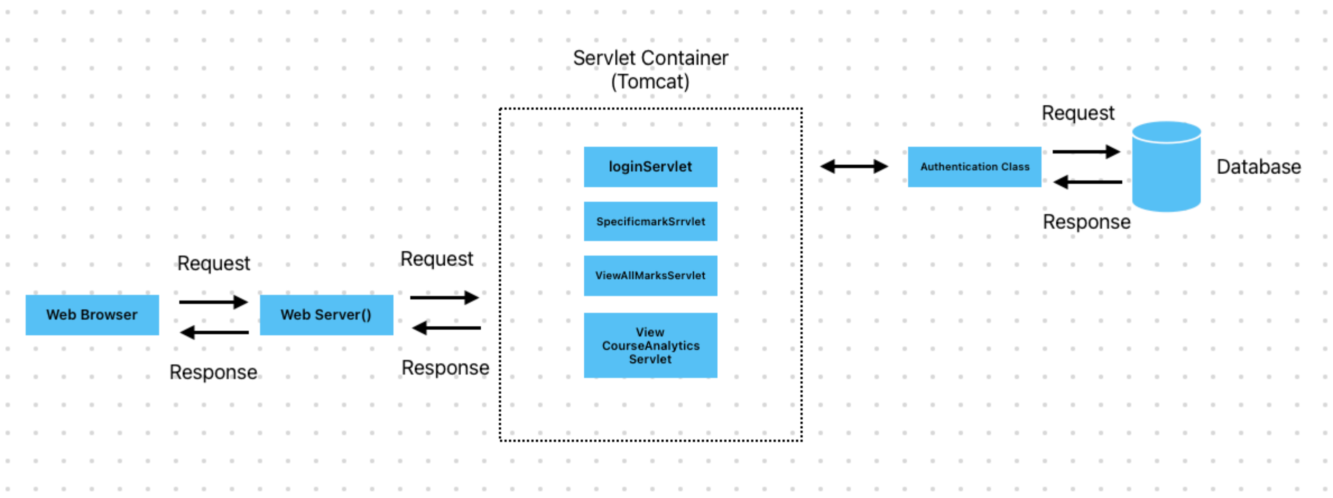
Future Enhancements for the Client-Server System

1. Two-Factor Authentication (2FA).

2. Role-Based Access Control (RBAC).

3.Intrusion Detection and Prevention System (IDPS).

Part 2: Web App / Tradi&onal MVC Servlets and JSPs Implementation.



- **AuthenticationService ->** This class handles user authentication. It interacts with the database (DB class) to validate user credentials.

    - Algorithms/Techniques Used: Implements authentication logic using the (authenticate) method, which verifies user credentials by checking if the student ID exists in the DB and if the entered password matches the stored password.

- **CourseINFO** -> Represents information about a specific course, including its name and corresponding mark.

    - Algorithms/Techniques Used: Simple data class storing course name and mark. And provides getters and setters for encapsulation.

- **DB Class ->** handles database interactions including connecting to the database, retrieving course statistics, fetching student marks, and validating student information.

    - Algorithms/Techniques Used: Implements methods to fetch course statistics (average, minimum, maximum), retrieve all marks for a student, fetch a specific course mark for a student, check if a student is registered, and validate student password.

- **Servlets ->** Servlets handle HTTP requests and responses for diff functionalities like user login, viewing single marks, viewing all marks, and viewing course analytics.

    - Algorithms/Techniques Used: Implements (doGet) and (doPost) methods to handle HTTP GET and POST requests, respectively. Utilizes request dispatching to forward requests to appropriate JSP pages for for showing them on the website.

Design Decisions:

- Uses servlet annotations (@WebServlet) for mapping servlet URLss
- Implements logic to check user authentication status before serving certain requests
- Provides error handling for servlet operatins

Overall Design Considerations:

- **Security**: Implements pass validation to ensure secure user authentication.
- **Efficiency**: Utilizes design patterns like Singleton for resource optimization & efficient object management.
- **Error Handling**: To deal with exceptions.

Challenges and Solutions:

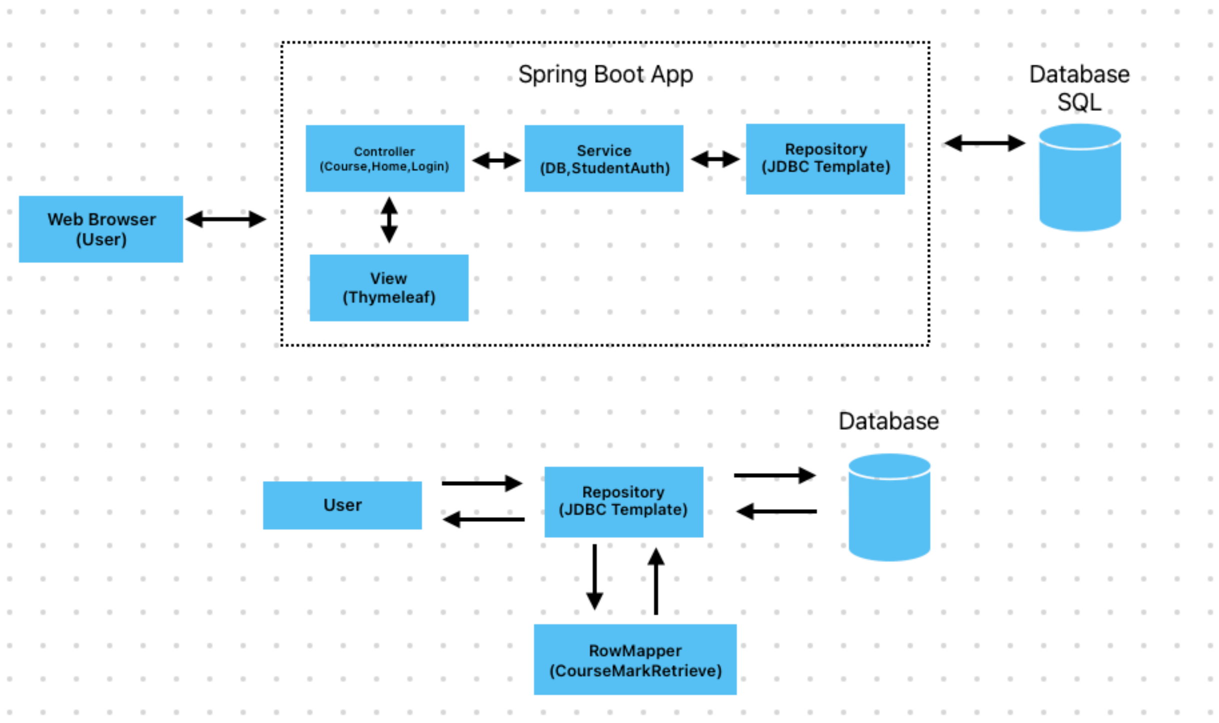
Identification of Challenges:

- **Database Connection Management**: Making sure the connection to the database works smoothly all the time without causing any slowdowns or wasting resources.
- **Error Handling**: Handling exceptions & errors gracefully to maintain application stability and provide meaningful feedback to users.

Solutions:

- **A Singleton Database Connection** means we use a special method to make sure there's only one connection to the database and this helps us to manage the database connection efficiently
- **Exception Handling**: setting up strong ways to deal with errors by using try-catch blocks

Part 3: Web App / Spring MVC and Spring REST Implementation



Spring Boot App
Database SQL
Controller (Course,Home,Login)
Service (DB,StudentAuth)
Repository (JDBC Template)
Web Browser (User)
View (Thymeleaf)
Database
User
Repository (JDBC Template)
RowMapper (CourseMarkRetrieve)

Overview

Student System developed using the Spring Boot framework. It aims to provide functionalities related to managing student data, authentication, and accessing course information.

Project Structure:

- The project follows a standard Maven directory structure for organizing source code, resources, and configurations.
- Java classes are organized into packages based on functionality (controllers, models, service components, and row mappers)

Technologies Used:

- **Spring Boot:** the primary framework for building the application. It simplifies the setup & development of Spring-based applications by providing defaults for configuration and allowing developers to start coding right away.
- **Spring MVC (Model-View-Controller):** The application follows the MVC architectural pattern, with controllers handling incoming requests, models representing the data, and views responsible for rendering the user interface :)
- **Thymeleaf:** the templating engine for generating HTML views. It allows for seamless integration of dynamic content into web pages, making it easier to create interactive user interfaces.
- **MySQL Database:** the relational database management system for storing student and course-related data. The application interacts with the database using Spring Data JDBC, simplifying database operations through abstraction.
- **JDBC Template:** executing SQL queries and managing database connections. It provides a higher-level abstraction over traditional JDBC, reducing boilerplate code and enhancing productivity.

Project Components (Implementation):

- Controllers:
  - LoginController: Handles both GET and POST requests for login, validating student credentials against the database. Upon successful authentication, it redirects the user to the home page.
    - Challenges faced: Ensuring secure handling of sensitive user data like passwords.
  - HomeController: Handles requests related to the home page. It retrieves the authenticated user's ID and passes it to the view for personalized content rendering.
  - CourseController: Contains endpoints for retrieving student marks, calculating averages, maximum, minimum, and median marks for courses.
    - Challenges: Efficient retrieval of marks for different courses.
    - Solution: Using a row mapper to map database query results to Course objects, facilitating easy retrieval and processing of marks.
- Models:
  - Student: Represents a student entity with attributes (ID, password, and name)
  - Course: Represents a course entity with attributes for name and mark.
- Services:
  - DB: Provides methods for interacting with the MySQL database, including validation of student credentials, retrieval of marks, and calculation of statistical measures.
    - Challenges: efficient calculation of analytics like average, maximum, minimum, and median marks.
    - Solutions: Implementing SQL queries for aggregate functions like AVG, MAX, MIN and utilizing subqueries to calculate the median mark.
- Row Mappers:
  - CourseMarkRetrive: Maps rows from the database query result to Course objects, facilitating the conversion of database records to Java objects.

Strengths:

- Use of Spring Boot simplifies application setup and development.
- Clear separation of concerns through MVC architecture enhances & improve maintainability.
- Integration of Thymeleaf allows for dynamic content rendering in HTML views.
- JdbcTemplate provide efficient database operations.

Security Considerations:

Access Control: restrict access to certain functionalities based on user roles and permissions. For example: only authenticated users are allowed to view sensitive info (student marks).