Distributed Fault Tolerance

Problem Set 4

Jumilla, Sarah Ericka
Ramos, Ashley Kylle
Refuerzo, Lloyd Dominic
Rojo, Kate Lynn
STDISCM-S14

Key implementation steps

Node Set-up

- **AuthNode**
  - Responsible for user authentication and authorization.
  - Implemented using Spring Boot
  - Handles login requests, verifies credentials, and generates JWT tokens for authentication
  - Uses Spring Data JPA to interact with the database for student/faculty data
  - Secures communication with JWT.

Key implementation steps

Node Set-up

- **CourseNode**
  - This node manages course-related information and enrollment.
  - Implemented using Spring Boot
  - Provides endpoints for:
    - Viewing available courses and sections
    - Enrolling students in courses
    - Dropping courses
  - Uses Spring Data JPA to interact with the database for course, section, and enrollment data.
  - Implements an *EnrollmentService* to handle enrollment and drop logic.

Key implementation steps

Node Set-up

- **ViewNode**
  - Responsible for the presentation layer of the application.
  - Implemented using Spring Boot.
  - Handles user interaction and displays data to the user.
  - Uses Spring MVC to handle web requests and render views
  - Communicates with the *AuthNode* for authentication and the *CourseNode* for course data via RESTful APIs.
  - Uses RestTemplate to make HTTP requests to other nodes.
  - Manages user sessions (for the view layer) using HttpSession.

Key implementation steps

Node Set-up

- **GradesNode**
  - Manages grade information.
  - Implemented using Spring Boot.
  - Provides an endpoint for:
    - Viewing grades by student ID.
  - Uses Spring Data JPA to interact with the database for grade data.

Key implementation steps

Database Integration

- Each node interacts with a database (application properties) to store persistent data.
- Spring Data JPA is used to simplify database interactions through repositories
- The database schema includes tables for students, faculty, courses, course sections, enrollments, and grades.

Key implementation steps

Communication

- The nodes communicate with each other using RESTful APIs.
- *ViewNode* sends HTTP requests to *AuthNode* and *CourseNode*.
- *AuthNode* and *CourseNode* respond with JSON data.
- JWT is used for secure communication, specifically for authentication. The *AuthNode* generates a JWT upon successful login, and this token can be used by the *ViewNode* to authenticate subsequent requests to the *CourseNode*.

Key implementation steps

Enrollment Logic

- The *EnrollmentService* in the *CourseNode* handles the core enrollment and drop operations.
- It checks for section availability, existing enrollments, and other constraints.
- It updates the enrollments table in the database to reflect enrollment changes.

Explanation on how fault tolerance is achieved

AuthNode Failure

- Users will not be able to log in or log out
- However, users who are already logged in (and have a valid JWT) can still access course information and perform enrollment operations, as long as the *CourseNode* is running.

CourseNode Failure

- Users will not be able to view course information, enroll in courses, or drop courses.
- However, users can still log in and log out through the AuthNode.

ViewNode Failure

- Users will not be able to access the web interface of the application. They won't be able to view course listings, enroll in courses, or log in/log out through the web browser.
- However, the core application logic on the *AuthNode and CourseNode* remains operational. The authentication and course management services are still running.

Explanation on how fault tolerance is achieved

GradesNode Failure

- Users will not be able to view their grades through the application.
- However, other functionalities, such as login/logout (handled by *AuthNode*) and course enrollment (handled by *CourseNode*), will remain operational.
- The core data (grades) is stored in the database. When the GradesNode is restored (or a new instance is brought online), the grade information becomes accessible again.