PROJECT REPORT

PROJECT NAME	Java Spring Boot technology for Full Stack Web Development		**PROJECT NO.**	36
PROJECT MANAGER		**START OF THE PROJECT**	**END DATE**	**STATUS**
Haiyuan Lin		06.05.2024	31.05.2024	COMPLETED

MEMBERS

NAME	EMAIL	FIELD AND YEAR	SKILLS
Marat Rizakhanov	marat.rizakhanov@tuni.fi	SE, year: 1	Java, HTML, CSS, JS, python
Yannan Zhang	yannan.zhang@tuni.fi	SE, year: 1	Node.js
Ivan Lubnin	ivan.lubnin@tuni.fi	SE, year: 1	Java, HTML, CSS, JS
Alicja Kosak	alicja.kosak@tuni.fi	SE, year: 1	Java, python, CSS, JS

PROJECT SUMMARY

Our project uses Spring Boot with Gradle to build backend infrastructure for generating API routes for frontend interaction. The main goal is to provide endpoints for efficient data retrieval and manipulation, including fetching data, adding entities, and handling errors with comprehensive logging and exception management.

Key Features:

- **User Interaction**: Users can browse, comment on, and rate food recipes, with advanced filtering and searching to enhance experience.
- **Security**: Role-based authentication ensures only authorized users can publish or modify content.
- **Logging and Debugging**: A user log table tracks significant user actions for efficient troubleshooting.
- **API Refinement**: Postman collections help refine our API with JWT authentication, file upload validation, and advanced error handling.

Future Plans:

- **Deployment**: Deploy the backend and database to a public server.
- **Frontend Development**: Develop the frontend using React.

PROJECT TECHNICAL OVERVIEW

CATEGORY	DETAILS	COMMENTS
Classes	User, recipe, comment	Attributes of User, Recipe, Comment; Relations between User, Recipe, Comment; User Conversion Service and Service for Child Entities (Recipe or Comment)
Controllers	Controller for User, etc.; Controller for df. page and user log	Annotations, Data Manipulation and Handling Cross-Table Interactions; Webpage Deployment Using Thymeleaf, Testing Framework, User Log Repository and Data Structure,
Key features	JWT generator, Image uploader, Statistics controller	Authentication Controller, Security Enhancements; Validation, Directory Handling, File Transfer and Logging; StatisticsController, Dependencies, Endpoints and Efficiency
Frontend	Pages for the Appication	Home page, Login page, Main page, and Profile Page

KEY ISSUES

ISSUE NAME	OWNER	DESCRIBTION	COMMENT
Set up gitHub	Yannan Zhang	Creation of the kanban board and a repository in github	Link to repository: https://github.com/nancheung23/javaspringboot-pt1 Link to the kanban board: https://github.com/users/nancheung23/projects/1
Customer meeting	Marat Rizakhanov, Yannan Zhang, Ivan Lubnin, Alicja Kosak	Meeting with the customer	Discussion about the project and its goals
Food API	Alicja Kosak	Selection of an API that will be used for finding recipes	Used API: https://www.themealdb.com/api.php
Forking branches	Marat Rizakhanov, Yannan Zhang, Ivan Lubnin, Alicja Kosak	Creation of the separate branches, and a main one	Learning how to fork branches, from a private one
Backend merging	Yannan Zhang	Main branch of the project contains the backend	Merging the branch with the main one
MultiPartFile file uploader	Marat Rizakhanov, Yannan Zhang	MultiPartFile implementation and creation of whole procedure of uploading files or images.	File repository links to specific User, Recipe or Comment.

Data Structure	Ivan Lubnin, Alicja Kosak	Defining the data, and properties of the app	What elements should be implemented
Statistics for recipe and comment	Marat Rizakhanov, Yannan Zhang	Creation of the statistics for recipe and comment	Enhance user experience
Frontend	Ivan Lubnin, Alicja Kosak	Creation of the frontend for the application	Link to the repository: https://github.com/AlicjaKo/SpringBoot-frontend
JWT token generator	Marat Rizakhanov, Yannan Zhang	Implementation of the JWT (JSON web token) generator for User instance.	Implement JWT and can be added in business logic

TECHNICAL DESCRIBTION IN DETAIL

1. **Attributes of User, Recipe, Comment**
    - User: Username, Password, Nickname, Avatar, Email, Birthday, Age, Recipes, Comments
    - Recipe: Title, Content, Double, Views, Picture, User, Comments
    - Comment: Content, Time, Picture, User, Recipe

2. **Relations between User, Recipe, Comment**
    - User: @OneToMany (mappedBy = "user", fetch = FetchType.EAGER) and @JsonManagedReference ensure recipes and comments sets are bound to the User instance.
    - Recipe and Comment: @ManyToOne (fetch = FetchType.LAZY) and @JsonBackReference represent the backward connection to the User entity, storing a User object. For efficiency, the DTO classes store only the user id, with full conversion handled in the @Service classes.

    All three entities are stored in a PostgreSQL database, and each has a repository interface that extends CrudRepository or JpaRepository, enabling automatic data persistence.

3. **Entities' relations in `@Service`**
    - User Conversion Service:
        - Converts between DTO and entity objects.
        - Manages JWT verification using UserDetails for authentication and session management.
    - Service for Child Entities (Recipe or Comment):
        - Uses @Transactional annotation to ensure data integrity across multiple tables.
        - When storing a recipe (converted from a recipe DTO), cross-table interaction with the user table is required, making @Transactional essential.

    The description of these services clarifies the roles of Controllers, making them easier to understand.

4. **Controllers for User, Recipe, Comment**
    - Annotations:

    - Use @RestController at the beginning of the class.
    - Specify a prefix route for the class, e.g., @RequestMapping("/api/users").
    - Define specific route mappings for methods, e.g., @PostMapping("/login"), which simplifies to "/login" within the route cluster.
- Data Manipulation:
    - Controllers directly manipulate the database repository using methods like .save(), .delete(), .findAll(), and .findById().
    - The @Autowired annotation is required for dependency injection, either in the attribute or constructor.
- Handling Cross-Table Interactions: For actions involving multiple tables (e.g., posting a comment to a recipe by a user):
    - Use @RequestBody for the comment object.
    - Use user ID and recipe ID to identify the associated entities.
    - The process involves:
        1. Finding user and recipe in their repositories.
        2. If either is null, throw an exception.
        3. If both exist, link the comment with the user and recipe.
        4. Save the comment in its repository and update the user and recipe.

  Else return a ResponseEntity<CommentDTO> to filter sensitive information and provide a clear JSON response for the frontend.

5. Controllers for default page and user log

- Webpage Deployment Using Thymeleaf:
    - Before front-end and back-end separation, developers used templates (e.g., Thymeleaf) to create web pages stored in .../resources.
    - A @ConfigurationProperty(prefix='webApp') allows developers to set properties for the web application.
    - A @Controller can be used to deploy a webpage. For instance, returning a String value like home will map to home.html, which should invoke Thymeleaf.
- Testing Framework:
    - A default @Controller can serve a test router to check the framework's functionality.
    - Parameter values are defined in application.properties.
- User Log Repository and Data Structure
    - Logger Level Setting: In application.properties, logger levels can be adjusted from INFO to DEBUG to trace bugs.
    - User Log Table:
        1. To record user behavior, a logs table can be used.
        2. A specific service should be created to record logs.

3. Method example: Using LocalDateTime.now() for timestamp, and formatting details

6. Key Features

A) JWT Generator

- JwtUtility.java: Handles JWT generation using the HS256 algorithm, setting expiration from application.properties using @Value("${jwt.expiration}").
- JwtAuthenticationFilter.java: Inspects incoming requests for a "Bearer <token>" pattern, verifies the token, and transforms it into a UserDetails object for authentication.
- Authentication Controller: Issues tokens via /auth/authenticate endpoint, which accepts username and password. Token duration is controlled by settings in application.properties.
- Security Enhancements: Uses HS256 and hash salt methods for password encoding and token generation, deviating from default Spring Boot encoders.

B) Image Uploader

- ImageStorageService: Manages image uploads configured via @Value("${file.upload-dir}") in application.properties.
- Validation:
  - Limits uploads to specific formats (JPG, JPEG, PNG, GIF, WEBP).
  - Checks for null filenames, valid extensions, and non-empty files.
- Directory Handling: Creates directories if they don't exist and validates paths to prevent directory traversal attacks.
- File Transfer: Transfers validated files to storage and returns the absolute path.
- Logging: Uses SLF4J for logging actions and information.

C) Statistics Controller

- StatisticsController: Manages and presents statistical data for recipes, set up within the com.foodrecipes.webapp.controller package and responds to /api/statistics routes.
- Dependencies:
  - RecipeRepository and UserRepository for data access.
  - RecipeConversionService and UserConversionService for entity-to-DTO conversion.
- Endpoints:
  - Popular Recipes by Views (/popular-recipes-views): Retrieves and sorts recipes by views, returns top five as RecipeDTO.
  - Popular Recipes by Ratings (/popular-recipes-ratings): Sorts recipes by ratings, returns top ten as RecipeDTO.
- Efficiency: Utilizes Java Streams for data processing, improving response times and memory usage.

7. Frontend

- Home Page: the page that welcomes the users, it shows the title of the website, and it has a short description of the purpose of the website
- Login Page: user can register and login to the application
- Main Page: search for the recipes, and see the news
- Profile Page: data about the user that are editable