**Fashion Trend Analysis Dashboard Documentation**

**Project Overview**

**Project Name**

Fashion Trend Analysis Dashboard

**Project Description**

The Fashion Trend Analysis Dashboard is a web-based application designed to provide valuable insights into the latest fashion trends, designer collections, and trend analytics. This project will use data from the September 2023 London Fashion Weeks in New York, London, Milan and Paris.

**Project Objectives**

* Develop a web-based dashboard for analysing and visualizing fashion trends.
* Implement a user-friendly interface for users to explore fashion trends and designer collections.
* Provide trend analytics and insights to users.
* Allow user interaction features such as comments, likes, and trend sharing.
* Ensure data security, integrity, and user authentication.

**Project Technologies**

* **Spring MVC:** The project will be built using the Spring MVC framework, which provides a structured and modular approach to building web applications.
* **Maven:** Maven will be used for project management and dependency management, simplifying the build process.
* **JDBC (Java Database Connectivity):** JDBC will be used to interact with the database to store and retrieve fashion trend data.

**Technical Stack**

**Backend:**

* **Spring Boot:** As the chosen framework, Spring Boot provides a powerful foundation for building Java-based web applications.
* **Maven:** Maven for project management and dependency resolution. It simplifies the build process and allows you to manage project dependencies efficiently.
* **Spring MVC:** Implement the Model-View-Controller (MVC) architecture provided by Spring for structuring your application.
* **Spring Data JDBC Template or JPA:** Choose either Spring Data JDBC Template or Java Persistence API (JPA) for interacting with the database and performing CRUD operations.
* **MySQL Database:** For storing fashion trend data, you can use a relational database like MySQL. Ensure proper schema design and normalization.
* **JUnit 5:** Write unit tests to ensure the reliability and correctness of your application. JUnit 5 is a popular testing framework for Java.
* **Spring DI and Annotations:** Utilise Spring's Dependency Injection (DI) and annotations for component management and configuration.
* **Error Handling Framework:** Implement custom error handling using Spring's Exception Handling capabilities.

**Frontend:**

* **HTML, CSS, JavaScript:** The frontend interface can be developed using the fundamental web technologies like HTML, CSS for styling, and JavaScript for interactivity.
* **Thymeleaf:** Thymeleaf is a templating engine that integrates well with Spring MVC. It allows you to create dynamic web pages easily.
* **Bootstrap:** Bootstrap provides responsive and pre-designed UI components that can expedite frontend development.
* **Data Visualisation Libraries:** Consider using data visualisation libraries like D3.js or Chart.js if you plan to display trends graphically.

**Version Control and Collaboration:**

* **Git:** Use Git for version control. It allows you to track changes, collaborate with team members, and synchronise your project with platforms like GitHub.

**Deployment and Hosting:**

* **Deployment Server:** Choose a server for deploying your Spring Boot application. Popular choices include Apache Tomcat.

**Feature Implementation and Testing**

**Backend Development:**

* **Create Trend Data Models:** Develop Java classes to represent fashion trend data, including attributes like trend name, description, date, and source.
* **REST API Endpoints:** Implement RESTful API endpoints for CRUD operations.
* **Data Validation:** Implement server-side data validation to ensure data integrity and security.
* **Error Handling:** Create custom error messages and handle exceptions gracefully.

**Database Setup:**

* **Schema Creation:** Execute SQL scripts to create the database schema with appropriate tables.
* **Data Population:** Use SQL scripts to populate the database with sample fashion trend data.
* **Data Access Layer:** Develop data access objects (DAOs) or repositories to interact with the database using JDBC Template or JPA.

**Frontend Development:**

* **UI Components:** Design and implement UI components for the dashboard, including:
  + Trend listing with filtering options.
  + Data entry forms for adding and updating trends.
  + Charts and graphs for trend visualisation.
* **Integration with Backend:** Establish communication with the backend through REST API calls to fetch and display trend data.
* **Data Validation:** Implement client-side data validation for user inputs in forms.
* **Responsive Design:** Ensure the UI is responsive for various screen sizes and devices.

**Testing:**

* **Unit Testing:** Write unit tests for each backend component using JUnit 5 or another Java testing suite. Test CRUD operations, data validation, and error handling.
* **Integration Testing:** Perform integration tests to ensure the frontend and backend components work together seamlessly. Test API endpoints and data flow.
* **UI Testing:** Conduct user interface testing to verify that the dashboard's components function correctly and provide a good user experience.
* **Data Validation Testing:** Validate that data validation rules are enforced on both the client and server sides.
* **Error Handling Testing:** Test error scenarios to confirm that custom error messages are displayed when necessary.

**Documentation:**

* **API Documentation:** Create documentation for the REST API, including endpoint descriptions and sample requests/responses.
* **Testing Documentation:** Document the testing process, including test cases and results.

**Deployment:**

* Deploy the application to a web server or cloud hosting platform for accessibility.
* Ensure proper configuration and security measures for production deployment.

**Timescale**

**Step 1: Project Setup and Planning (September 21st)**

**1.1. Requirements Gathering and Analysis:**

* Clearly define the project's objectives, scope, and target audience.
* Identify key features and functionalities, including complex endpoints.
* Determine the technical stack and tools required.

**1.2. System Design:**

* Create wireframes and UI designs for the dashboard's frontend.
* Design the database schema (ERD) for storing fashion trend data.
* Plan the MVC architecture, outlining the structure of the Model, View, and Controller components.
* Define the API endpoints, both basic and complex ones.

**Step 2: Backend Development (September 22nd - September 24th)**

**2.1. Model (Data Layer):**

* Implement Java classes to represent fashion trend data.
* Set up a database, either relational (e.g., MySQL) or non-relational (e.g., MongoDB), as per your project requirements.
* Create SQL or NoSQL scripts to create tables and populate initial data.

**2.2. Controller (Application Logic Layer):**

* Develop the Controller layer to manage data integration and business logic.
* Implement basic CRUD endpoints (Create, Read, Update, Delete) for fashion trend data.
* Create complex endpoints, such as trend analysis and filtering, to meet the project's advanced requirements.

**Step 3: Frontend Development (September 25th - September 26th)**

**3.1. View (Presentation Layer):**

* Build the frontend interface using your chosen technology (JavaScript, React, Angular, etc.).
* Implement data visualization components, charts, and tables for presenting fashion trend data.
* Incorporate wireframes and UI design principles to ensure a user-friendly interface.

**Step 4: Data Integration (September 24th - September 26th)**

**4.1. Data Extraction and Transformation:**

* Develop data extraction processes to gather fashion trend information from articles.
* Use web scraping libraries (e.g., Jsoup) or APIs of fashion news websites to collect data.
* Transform and structure the extracted data to fit the Model's data classes.

**4.2. Integrate Data into the System:**

* Implement Controller methods for fetching and storing trend data from articles.
* Schedule periodic data updates to keep trend information current.
* Ensure that data retrieved from articles is stored in the database for easy access.

**Step 5: API Creation (September 23rd - September 25th)**

**5.1. RESTful API Development:**

* Create RESTful API endpoints for interacting with fashion trend data.
* Include endpoints for basic CRUD operations and complex trend analysis.
* Implement server-side data validation to ensure data integrity.

**Step 6: Testing (September 24th - September 26th)**

**6.1. Unit Testing:**

* Write unit tests for each component, including Controller methods and API endpoints.
* Utilise testing frameworks like JUnit 5 to ensure functional correctness.

**6.2. UI/UX Testing:**

* Perform UI/UX testing to ensure the frontend components render correctly and are user-friendly.

**Step 7: Error Handling and Refinement (September 26th - September 27th)**

**7.1. Error Handling:**

* Implement custom error messages and error handlers as per project requirements.
* Ensure that errors are appropriately categorized and handled.

**7.2. Refinement and Optimisation:**

* Review the codebase for optimization opportunities and potential improvements.
* Conduct performance testing to identify and address bottlenecks.

**Step 8: Documentation (Throughout the Development Process)**

**8.1. Documentation of Code and APIs:**

* Continuously document code, APIs, and endpoints using comments.
* Generate API documentation to provide clear instructions for usage.

**Step 9: Deployment and Presentation (September 27-29th)**

**9.1. Deployment:**

* Deploy the application to a server.
* Ensure that all components are operational and accessible.

**9.2. Project Presentation:**

* Prepare a comprehensive project presentation to showcase your Fashion Trend Analysis Dashboard to your team.
* Highlight the features, functionality, and technical aspects of the project.

Ideas for frontend: automatic Carousal – no need for user to click to show next.