Collaborative Whiteboard Project Documentation

1. Introduction

The Collaborative Whiteboard is an interactive web application designed for seamless online collaboration. It allows multiple users to draw, annotate, and brainstorm ideas on a shared canvas. The project leverages core web technologies such as HTML, JSP, CSS, MySQL, Maven, Java Servlets, DAO, JTest, and JavaScript for validation.

Key Features:

- **Dynamic whiteboard for user interaction.**
- **Persistent storage of whiteboard sessions using MySQL.**
- **User authentication and session management.**
- **Scalable and modular design using Maven.**
- **Validation using DAO, JTest, and JavaScript.**

---

2. System Requirements

Hardware Requirements:

- **Processor: Minimum dual-core processor.**
- **RAM: 4GB or higher.**
- **Storage: At least 500MB free space.**

Software Requirements:

- **Operating System: Windows, macOS, or Linux.**
- **JDK: Version 8 or higher.**
- **Apache Tomcat: Version 9.0 or higher.**
- **Browser: Latest versions of Chrome, Firefox, or Edge.**

---

3. Technology Stack

Frontend:

- **HTML and JSP: HTML provides the structure for web pages, while JSP is used to dynamically generate content on the server-side before rendering in the browser.**
- **CSS: Used to create visually appealing layouts and responsive designs.**
- **JavaScript: Enables interactive and dynamic elements on the client side, such as real-time validation and drawing tools.**

Backend:

- **Java Servlets: Manage server-side logic, process HTTP requests, and control the flow of data between the frontend and the database.**
- **DAO Pattern: Ensures separation of concerns by handling all database-related operations in a structured manner.**

Database:

- **MySQL: A robust relational database used for storing user credentials, session data, and whiteboard content securely.**

Build Tool:

- **Maven: Manages project dependencies, automates the build process, and ensures consistent project structure.**

Testing:

- **JTest: Used for validating server-side logic and ensuring reliable code through unit and integration tests.**

---

4. Installation Guide

Step 1: Clone the Repository

git clone https://github.com/Celestial-1/Collaborative-Whiteboard.git

cd Collaborative-Whiteboard

Step 2: Configure MySQL Database

1. **Create a new database named `whiteboard`.**
2. **Run the SQL script provided in the `db` directory to set up tables.**

Step 3: Configure Environment Variables

Update the `db.properties` file with your MySQL credentials:

db.url=jdbc:mysql://localhost:3306/whiteboard

db.username=<your_username>

db.password=<your_password>

Step 4: Build and Deploy the Application

Use Maven to build the project:
mvn clean install

1.
2. **Deploy the generated WAR file to Apache Tomcat:**
   - **Copy the WAR file from the `target` directory to the `webapps` folder of your Tomcat server.**
3. **Start the Tomcat server and access the application at `http://localhost:8080/Collaborative-Whiteboard`.**

---

5. Project Architecture

High-Level Architecture:

- **Frontend: Static pages with JSP and dynamic styling using CSS.**
- **Backend: Java Servlets managing HTTP requests and responses.**

- Database: MySQL for persistent data storage.

Components:

1. User Interface (UI): Built using HTML, JSP, CSS, and JavaScript.
2. Servlets: Handle backend logic and business processes.
3. DAO: Encapsulates database interactions.
4. Database: Stores user data and whiteboard content.

---

6. Frontend Design

Overview:

The frontend includes tools for:

- Drawing on a virtual canvas.
- Selecting colors and brush sizes.
- Clearing the canvas.

Key Components:

- Canvas Page: Displays the interactive whiteboard.
- Control Panel: Contains options for drawing tools and color selection.

---

7. Backend Implementation

Servlets:

- **WhiteboardController: Central controller for handling HTTP requests and directing the user to appropriate JSP pages (e.g., signup, signin, or the main whiteboard).**

Business Logic:

- **User actions are processed server-side using Java Servlets.**
- **DAO handles database interactions for cleaner code separation.**
- **Data is persisted in MySQL through JDBC connections.**

---

8. Database Design

Data Models:

1. **User:**
    - `id`: Integer (Primary Key)
    - `username`: String
    - `email`: String
    - `password`: String (hashed)
2. **Whiteboard Session:**
    - `session_id`: Integer (Primary Key)
    - `user_id`: Foreign key referencing `User` table
    - `data`: Blob storing serialized whiteboard state

---

9. Core Functionalities

Drawing Tools:

- **Freehand drawing.**
- **Selection of brush size and color.**
- **Erasing specific sections or clearing the entire canvas.**

Data Persistence:

- **Save whiteboard state to the database for future access.**
- **Retrieve previous sessions for continuity.**

---

10. User Authentication

Features:

- **Secure registration and login using hashed passwords.**
- **Persistent sessions managed through cookies.**

---

11. Collaboration Features

Multi-user Collaboration:

- **Users can join the same session and contribute interactively.**
- **Synchronization is handled via periodic AJAX calls to the server.**

---

12. Error Handling

Common Issues:

- **Invalid login credentials.**
- **Database connection failures.**

Solutions:

- **Display user-friendly error messages.**
- **Log server-side errors for debugging.**

---

13. Scalability

Strategies:

- **Use connection pooling for efficient database access.**
- **Optimize SQL queries for performance.**

---

14. Testing

Types of Testing:

1. **Unit Testing: Validate individual Servlets and DAO operations.**
2. **Integration Testing: Ensure end-to-end functionality.**
3. **Validation Testing: Ensure input data is validated using JTest and JavaScript.**

4. UI Testing: Validate the behavior of the JSP pages.

---

15. Deployment

Hosting:

- Deploy on an Apache Tomcat server.

Deployment Process:

1. Build the project using Maven.
2. Deploy the WAR file to the Tomcat `webapps` directory.
3. Start the Tomcat server and verify deployment.

---

16. Usage Guide

Accessing the Application:

1. Navigate to `http://localhost:8080/Collaborative-Whiteboard`.
2. Register or log in to access the whiteboard.
3. Use the drawing tools to interact with the canvas.

Saving and Loading:

- Save the current whiteboard state using the "Save" button.
- Load a saved session using the "Load" button.

---

17. Future Enhancements

Planned Features:

- Real-time synchronization using WebSockets.
- Export whiteboard content as images or PDFs.
- Add advanced shapes and text tools.

---

18. Troubleshooting

Issues:

1. Unable to log in: Check database connection and user credentials.
2. Whiteboard not saving: Ensure database is properly configured.

Steps to Resolve:

- Check server logs for error messages.
- Verify database credentials in the `db.properties` file.

---

19. Project Contribution

Guidelines:

1. Fork the repository and create a new branch for your feature.
2. Commit changes with clear messages.

3. **Open a pull request for review.**

---

20. JavaScript for Validation

Role in the Project:

- **Form Validation: Ensure inputs such as email, username, and password meet the required format and constraints before submission.**
- **Interactive Feedback: Provide real-time error messages to users for incomplete or incorrect inputs.**
- **Security: Reduce server-side validation load by catching errors on the client side.**

Examples of Validations:

- **Email Validation: Check for proper email format.**
- **Password Strength: Ensure passwords meet complexity requirements.**
- **Field Completeness: Prevent empty or invalid fields from being submitted.**

---

21. Team Member :

- **Yash Kumar Singh (leader),**
- **Ritik,**
- **Shubham Mishra.**