**1. System Overview**

The system consists of three main microservices:

* **User Management**: Manages user authentication, registration, and profiles.
* **Document Editing**: Allows for document creation, real-time editing, and change tracking.
* **Version Control**: Manages version history, rollback capabilities, and user contributions tracking.

Additionally, the system includes:

* **API Gateway**: Routes requests to appropriate microservices.
* **REST APIs**: Each microservice exposes RESTful endpoints for communication.
* **JUnit Tests**: Tests are implemented for all services to ensure functionality.

**2. Required Tools and Libraries**

* **Spring Boot**: For rapid microservice development.
* **Swagger**: For API documentation.
* **Spring Cloud Gateway**: For setting up the API gateway.
* **JUnit**: For unit and integration testing.
* **Spring Data JPA**: For data persistence.
* **WebSocket** (optional): For real-time editing in the Document Editing service.

**3. Microservices Design**

**A. User Management Service**

* **Operations**:
  1. **Register User**: Create a new user account with details like username, email, and password.
  2. **Authenticate User**: Validate user credentials for login and issue a JWT token.
  3. **Manage Profile**: Update user information, such as profile picture or bio.
* **API Endpoints**:
  1. POST /users/register
  2. POST /users/authenticate
  3. PUT /users/{userId}/profile
* **Technologies**:
  1. **JWT Authentication**: Secure endpoints with JSON Web Tokens.
  2. **Database**: Use MySQL or PostgreSQL for storing user data.

**B. Document Editing Service**

* **Operations**:
  1. **Create Document**: Allow users to create new documents.
  2. **Edit Document**: Enable collaborative editing with WebSocket for real-time changes.
  3. **Track Changes**: Store a log of changes made to the document for tracking purposes.
* **API Endpoints**:
  1. POST /documents
  2. PUT /documents/{docId}/edit
  3. GET /documents/{docId}/changes
* **Technologies**:
  1. **WebSocket**: For real-time collaborative editing.
  2. **Database**: Use MongoDB for storing document data and changes.

**C. Version Control Service**

* **Operations**:
  1. **Save Version**: Create a version snapshot whenever a significant change is made.
  2. **Revert to Version**: Allow users to revert a document to a previous version.
  3. **Track Contributions**: Log contributions made by each user for accountability.
* **API Endpoints**:
  1. POST /versions/{docId}/save
  2. GET /versions/{docId}
  3. POST /versions/{docId}/revert
* **Technologies**:
  1. **Database**: Use MySQL or PostgreSQL to manage version history and contributions.

**4. API Gateway**

* **Spring Cloud Gateway**: Set up routing rules to forward requests to the appropriate microservices.
* **Endpoints**:
  + User Management endpoints (e.g., /api/users/\*\*) are forwarded to the User Management service.
  + Document Editing endpoints (e.g., /api/documents/\*\*) are forwarded to the Document Editing service.
  + Version Control endpoints (e.g., /api/versions/\*\*) are forwarded to the Version Control service.

**5. Data Flow and Communication**

* **Authentication**: User logs in through the User Management service and receives a JWT token.
* **Authorization**: JWT token is attached to requests made to Document Editing and Version Control services to ensure only authenticated users can access or modify documents.
* **Document Editing**: Uses WebSocket to provide real-time collaborative editing. Each change is broadcast to all users viewing the document.
* **Version Control**: The Document Editing service calls the Version Control service to save document versions at specified intervals or upon user request.

**6. Implementation Steps**

1. **Setup Project Structure**:
   * Create separate Spring Boot applications for each microservice (User Management, Document Editing, Version Control).
   * Set up the Spring Cloud Gateway as a standalone project.
2. **Develop Microservices**:
   * **User Management**: Implement endpoints for registration, authentication, and profile management with JWT-based security.
   * **Document Editing**: Implement endpoints for document creation and real-time editing with WebSocket support.
   * **Version Control**: Implement endpoints for versioning operations and contributions tracking.
3. **Configure API Gateway**:
   * Set up Spring Cloud Gateway routes to direct requests to the appropriate microservices based on URL paths.
4. **Swagger Integration**:
   * Add Swagger to each microservice to auto-generate API documentation.
5. **Testing with JUnit**:
   * Write unit tests for each operation in all three microservices.
   * Create integration tests to verify the interaction between microservices.
6. **Real-Time Editing Feature**:
   * Use WebSocket in the Document Editing service to support live collaboration.
   * Broadcast updates to all users viewing the document.