1. High-Level Design Description

Note-taking application that allows users to perform actions such as creating, saving, viewing, and deleting notes.

2. Interface Components (UI)

Required Components:

- **Annotation Form**: Allows users to enter text to create a new annotation.
- **Save Button**: Clicking will save the currently typed note.
- **Note List**: Displays all notes saved by the user, showing the title or an excerpt of the note for quick identification.
- **Delete Button**: Each note will have a delete button, allowing the user to remove unwanted notes.

Interactions:

- **Save Annotation**: After entering text and clicking save, the annotation is sent to the server via RESTful API to be saved in a database.
- **Delete Annotation**: Clicking the delete button next to each annotation in the list removes the corresponding annotation from the database and the displayed list in the user interface.

Wireframe:

A simple sketch might include a top navigation bar with the app title, followed by a note form centered on the screen. Below the form, a list of notes will be displayed, each with a delete button.

3. Data Model

User Model:

- **ID**: Unique user identifier.
- **Name**: User's name.
- **Email**: Unique user identification in the database.
- **Password**: Encrypted value stored in the database used together with email to access the app.
- **Creation Date**: Timestamp of user registration.

Note Model:

- **ID**: Unique identifier of the note.
- **Text**: Content of the annotation.
- **Creation Date**: Timestamp of note creation.

4. RESTful API

Resources:

- **GET /api/v1/notes**: Returns all user notes.
- **POST /api/v1/notes**: Creates a new note.
- **DELETE /api/v1/notes/{id}**: Deletes the note with the specified ID.

HTTP Verbs:

- **GET**: Retrieves all notes or a specific note.
- **POST**: Creates a new note with the note text in the message body.
- **DELETE**: Deletes an annotation based on the provided ID in the URL.

5. Web Server

RESTful API Implementation:

- **Business Logic**: Includes authentication and permission checks before allowing CRUD operations on annotations.
- **Persistence**: Annotations stored in a database, such as MongoDB or MySQL, depending on scalability and persistence needs.
- **Technologies**: Utilization of frameworks like Spring Framework for backend and MongoDB as a NoSQL database for storing annotations.
- **Security**: Uses user sessions for authentication and authorization, with encryption to safeguard sensitive data such as session tokens.

Considerations

This design assumes a fundamental user flow, focusing on core functionalities of the note-taking app. A comprehensive implementation would also encompass unit testing, robust error handling, and potential enhancements in performance and security.