Internship Week (2) Day 2 Tasks For Students (Batch 4)

Project Title: Information Collector

Client: Avirant Enterprises

Project Overview:

The Information Collector project for Avirant Enterprises is designed to create a flexible and customizable data collection platform that offers the features of Google Forms while seamlessly integrating with Google Contacts for automated data storage. This solution will enable clients to efficiently gather and manage information tailored to their specific needs, enhancing their operational capabilities.

Objectives:

1. Customizable Data Collection:

- **Provide a user-friendly interface** that allows clients to create and modify forms based on their unique requirements.
- **Enable various question types**, including multiple-choice, text, checkboxes, and more, for diverse data collection scenarios.

2. Automated Data Storage:

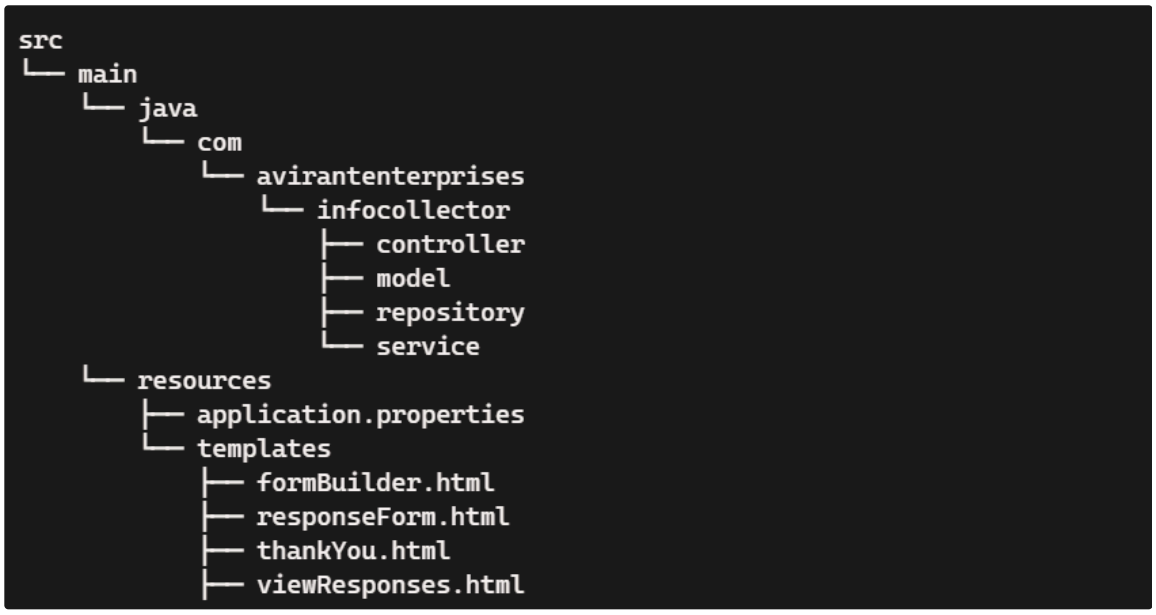
- **Implement an automated process** that directly stores collected data into Google Contacts, ensuring seamless data integration and management.
- **Maintain data accuracy** and reduce manual entry errors by automating the storage process.

3. Robust Reporting and Analytics:

- **Offer tools for clients to analyze collected data**, generate reports, and gain insights for informed decision-making.
- **Include features for exporting data** in multiple formats for further use.

Project Structure:

Directory Structure:



Detailed Development Steps:

1. Project Setup:

- **Create a New Spring Boot Project**:
  - Use Spring Initializr.
  - Add dependencies: Spring Web, Spring Data JPA, Thymeleaf, MySQL Driver.
  - Set project metadata:
    - Group: `com.avirantenterprises`
    - Artifact: `infocollector`
    - Name: `InfoCollector`
    - Description: `Information Collector`
    - Package Name: `com.avirantenterprises.infocollector`
  - Generate and import the project.
- **Configure Database**:
  - **application.properties**:

```
spring.datasource.url=jdbc:mysql://localhost:3306/infocollector
spring.datasource.username=root
spring.datasource.password=1234
spring.jpa.hibernate.ddl-auto=update
```

Customizable Data Collection Module

Step 1: Define Form Entity

Details to Store:

- **ID**: Unique identifier for each form.
- **Title**: Title of the form.
- **Questions**: List of questions associated with the form.

Step 2: Define Question Entity

Details to Store:

- **ID**: Unique identifier for each question.
- **Text**: Text of the question.
- **Type**: Type of question (e.g., text, multiple-choice, checkbox).

Step 3: Create Form Repository

- **Purpose**: Interface to manage database operations for forms.

Step 4: Create Question Repository

- **Purpose**: Interface to manage database operations for questions.

Step 5: Create Form Service

- **Service Interface**:
  - Methods:
    - `saveForm`: Save a new or updated form.
    - `findForms`: Retrieve a list of all forms.
    - `getFormById`: Retrieve a specific form by ID.

        - `deleteForm`: Delete a form by ID.
- **Service Implementation**:
    - Implement these methods to interact with the repository.

Step 6: Create Question Service

- **Service Interface**:
    - Methods:
        - `saveQuestion`: Save a new or updated question.
        - `findQuestions`: Retrieve a list of all questions.
        - `getQuestionById`: Retrieve a specific question by ID.
        - `deleteQuestion`: Delete a question by ID.
- **Service Implementation**:
    - Implement these methods to interact with the repository.

Step 7: Create Form Builder Controller

- **Endpoints**:
    - `/formBuilder`: Interface to create and modify forms.
    - `/saveForm`: Save a new or updated form.

Step 8: Create Form Builder UI Template

- **formBuilder.html**: User interface to create and modify forms with various question types.

Summary of Customizable Data Collection:

- **Form and Question Management**: CRUD operations for forms and questions.
- **UI**: Created for building and modifying forms.

This module ensures that users can create custom forms tailored to their specific needs, with various types of questions.

Automated Data Storage Module

Step 1: Set Up Google Contacts Integration

Goal:

- Automate the storage of collected data into Google Contacts to ensure seamless data integration and management.

Step 2: Obtain Google Contacts API Credentials

- **Google Cloud Console**:
    - Create a new project.
    - Enable the Google Contacts API.
    - Create OAuth 2.0 credentials (client ID and client secret).
    - Download the credentials JSON file.

Step 3: Configure Google Contacts API in the Project

- **application.properties**:
    - Add configuration for Google Contacts API credentials and token storage.

Step 4: Implement OAuth 2.0 Authentication

- **OAuth2 Service**:
    - Set up OAuth 2.0 authentication to authorize the application to access Google Contacts on behalf of the user.

Step 5: Create Data Storage Service

- **Service Interface**:
  - Methods:
    - `storeDataInGoogleContacts`: Store collected data into Google Contacts.
- **Service Implementation**:
  - Implement this method to interact with Google Contacts API and store data.

Step 6: Integrate Data Storage with Form Submission

- **Form Builder Controller**:
  - Update form submission handling to store data in Google Contacts upon form submission.

Step 7: Testing and Validation

- **Test the Integration**:
  - Submit a form and verify that the data is correctly stored in Google Contacts.
  - Check for any data discrepancies and resolve issues to ensure accuracy.

Summary of Automated Data Storage:

- **Google Contacts Integration**: Automated process to store collected data into Google Contacts.
- **OAuth 2.0 Authentication**: Secure authorization to access Google Contacts.
- **Data Storage Service**: Implementation to manage data storage in Google Contacts.

This module ensures that the collected data is seamlessly integrated into Google Contacts, maintaining accuracy and reducing manual entry errors.

Robust Reporting and Analytics Module

Step 1: Define Reporting and Analytics Objectives

Goals:

- Provide tools for clients to analyze collected data, generate reports, and gain insights for informed decision-making.
- Include features for exporting data in multiple formats for further use.

Step 2: Implement Data Analysis Tools

Features:

- **Performance Dashboards**: Visualize key metrics and data trends.
- **Custom Reports**: Generate detailed reports based on specific criteria.
- **Data Export**: Export collected data in various formats (CSV, PDF, etc.).

Step 3: Create Analytics Service

- **Service Interface**:
  - Methods:
    - `generateDashboardMetrics`: Generate metrics for performance dashboards.
    - `generateCustomReport`: Create a detailed report based on specific criteria.
    - `exportData`: Export collected data in the desired format.
- **Service Implementation**:
  - Implement these methods to analyze data and generate reports.

Step 4: Create Analytics Controller

- **Endpoints**:
  - `/dashboard`: View performance dashboards.
  - `/customReport`: Generate and view custom reports.
  - `/exportData`: Export collected data.

Step 5: Create Analytics UI Templates

- **dashboard.html**: Display performance dashboards.
- **customReport.html**: Interface to generate and view custom reports.
- **exportData.html**: Interface to export data in various formats.

Step 6: Testing and Validation

- **Test the Analytics Features**:
  - Generate sample reports and dashboards to ensure accurate data representation.
  - Validate data export functionality to ensure data is correctly formatted and exported.

Summary of Robust Reporting and Analytics:

- **Performance Dashboards**: Visualize key metrics and data trends.
- **Custom Reports**: Generate detailed reports based on specific criteria.
- **Data Export**: Export collected data in various formats for further use.

Overall InfoCollector Project Summary:

1. **Project Setup**: Spring Boot setup with necessary dependencies and database configuration.
2. **Customizable Data Collection**: CRUD operations and UI for creating and modifying forms with various question types.
3. **Automated Data Storage**: Integration with Google Contacts for automated data storage, maintaining accuracy and reducing manual entry errors.
4. **Robust Reporting and Analytics**: Tools for analyzing data, generating reports, and exporting data in multiple formats.