Survey Personalization Based on Dynamic Responses

09/07/2024
Project Report

Abishek SK
RA2111003040136

Key Aspects of the Work

Methodology

1. **Flask Application Setup**:
    - **Flask**: Used as the main framework to handle web routing, request handling, and template rendering.
    - **SQL Alchemy**: Used for database management to store and retrieve survey responses.
    - **HTML Templates**: Employed to render survey questions and capture user inputs.

2. **Dynamic Routing**:
    - **Route Handlers**: Each survey question has its own route handler to display the question and process responses.

3. **Database Schema**:
    - **Response Model**: A database model to store responses to each survey question. Fields include `q1`, `q2`, `q3`, etc., to track answers for each question.

4. **User Interaction Flow**:
    - **Starting the Survey**: The survey starts from the home page (`/`), creating a new `Response` record.
    - **Sequential Question Display**: Each question is displayed based on the current state of the `Response` object.
    - **Dynamic Question Routing**: After answering a question, the application decides the next question or the conclusion.

Findings

1. **Seamless User Experience**:
    - The dynamic routing approach ensures a smooth user experience by determining the next question based on previous responses, reducing the need for complex condition checks within each route.

2. **Maintainability**:
    - The use of it is to manage survey flow simplifies the codebase, making it easier to maintain and extend the survey with new questions or logic.

3. **Flexibility**:
    - The methodology allows for flexible survey paths, enabling personalized survey experiences. For example, different questions can be shown based on specific answers (e.g., if `q1 == 'Yes'`, proceed with questions `q2` to `q10`; otherwise, skip to question `q5`).

4. **Database Integration**:
    - The integration of SQLAlchemy for database management ensures that responses are stored persistently, facilitating data analysis and reporting.

Recommendations

1. **Enhancing User Feedback**:
    - Implement real-time validation and feedback for survey responses using JavaScript to improve the user experience further.

2. **Improving Scalability**:
    - For larger surveys, consider breaking down the logic into modular functions to handle more complex branching and conditions efficiently.

3. **Extending Functionality**:
    - Add features like user authentication to track individual progress, save partially completed surveys, and allow users to resume later.

4. **Data Analysis**:
    - Implement data analysis and reporting tools to analyze survey results directly from the application. This can be achieved by integrating libraries like Pandas and Matplotlib for data analysis and visualization.

5. **User Interface Design**:
    - Enhance the user interface with CSS and JavaScript frameworks (e.g., Bootstrap, React) to make the survey more engaging and responsive.

6. **Accessibility and Internationalization**:
    - Ensure the survey is accessible to all users by following web accessibility guidelines (WCAG) and consider internationalization support for multiple languages.

Summary

By implementing a dynamic survey in Flask, using a helper function to determine the next question, and integrating a database to store responses, the methodology presented offers a robust, flexible, and maintainable solution for personalized surveys. The findings highlight the effectiveness of this approach in providing a seamless user experience, while the recommendations suggest ways to further enhance the functionality, scalability, and usability of the survey application.