Web App Customization Task

1. Introduction

This project is based on the Django official tutorial application known as the Polls App. The objective of this assignment was to customize the existing application by extending its functionality and improving its user interface without rebuilding the system from scratch.

The primary aim of the customization was to enhance both the visual design and usability of the application while preserving its core structure. These modifications demonstrate an understanding of Django templates, static files, views, query handling, and front-end presentation.

The customization focused mainly on improving user interaction, data visibility, and visual presentation of poll information.

2. Description of Customization

Several meaningful enhancements were implemented in the Django Polls application.

Background Image Integration

The application interface was enhanced by adding background images using Django static files and CSS styling. Instead of the default plain background, the application now displays visually appealing background images. This modification required configuring static file paths and updating CSS styling rules.

Display of Total Vote Count on Home Page

The home page was modified to display the total number of votes for each poll. This was achieved by calculating the sum of votes from all related choices and passing this information from the view to the template. The index page template was updated to render the vote totals for each question.

Most Popular Poll Highlight

A new feature was introduced to identify and display the poll with the highest number of votes at the top of the home page. The system calculates total votes for each poll using Django query annotations and displays the most popular poll prominently.

Vote Percentage Visualization in Results Page

The poll results page was enhanced to visually represent vote distribution using percentage-based bars. Instead of only displaying numerical vote counts, users can now see graphical indicators showing the proportion of votes received by each choice. This required calculating vote percentages in the view and dynamically adjusting visual elements in the template.

User Interface Styling Improvements

Additional CSS styling was implemented to improve readability and layout. Link colors, hover effects, font sizes, and list formatting were customized to provide a cleaner and more structured interface.

3. Reason for Customization

The original Django Polls tutorial application provides only basic functionality and minimal visual design. While it allows users to vote and view results, it does not provide advanced visual feedback or enhanced presentation of poll data.

The purpose of these customizations was to improve both functionality and usability by:

- Providing better visual design
- Making poll information more informative
- Improving user engagement
- Enhancing data interpretation through visual representation

These changes make the application more suitable for real-world usage where both appearance and clarity of information are important.

4. Improvement to the Application

The implemented modifications significantly enhance the original application in several ways:

- Improves visual appearance through background images and styling
- Provides immediate insight into poll popularity through vote totals
- Highlights trending polls for easier navigation
- Enhances understanding of poll results using visual percentage bars
- Improves overall user experience and readability
- Demonstrates understanding of Django static files, templates, and view logic

These enhancements represent both functional and interface-level improvements, satisfying the requirement of meaningful customization.

Additional User Interface Enhancement – Hover Effect

An interactive hover effect was implemented on poll links using CSS styling. When a user moves the mouse pointer over a poll link, the link color changes to highlight the selection. This was implemented using the CSS: hover pseudo-class.

5. Conclusion

The Django Polls application was successfully customized by improving its visual presentation and extending its functionality. Background styling, vote count display, popularity highlighting, and visual result representation were implemented without altering the core architecture of the application.

These modifications demonstrate practical understanding of:

- Django Views and QuerySets
- Template Rendering
- Static File Management
- Front-end Styling with CSS
- Application Customization Techniques

The application is now more interactive, visually appealing, and informative compared to the original tutorial version, making it more suitable for practical use.