

|  |  |
| --- | --- |
| **Project title** | **Project Design** |
| **Course Code** | **DSE211/03** |
| **Course Name** | **Application Development Capstone** |
| **Project Start date** |  |
| **Project Submission Date** |  |

## The Project Design Document is a critical component of your capstone project, serving as the blueprint for your application’s development. This document will detail the design of the user interface, the backend architecture, and the integration of frameworks to ensure a cohesive and functional application. The design document should be based on the principles and concepts learned in IU2 (UI Frameworks), IU3 (Backend Development), and IU4 (Framework Integration).

## Design Document Components

## 1. User Interface (UI) Design

## 1.1 UI Frameworks Selection:

## Overview: Describe the UI frameworks you have chosen for the project. Discuss why these frameworks are suitable for your application, considering factors such as ease of use, community support, and compatibility with other technologies.

## Activity: Identify the UI frameworks (e.g., Bootstrap, Material-UI, Angular) that will be used in your project. Justify your choices with reference to your project’s requirements.

## 1.2 Wireframes and Mockups:

## Definition: Wireframes and mockups are visual representations of the user interface, showing the layout and structure of the application without detailed design elements.

## Activity: Create wireframes and mockups for the key screens of your application. These should include:

## The main dashboard or homepage.

## User registration and login screens.

## Any other key screens relevant to your application (e.g., product pages, user profiles, etc.).

## 1.3 Responsive Design:

## Implementation: Describe how you will implement responsive design to ensure your application works well on various devices, including desktops, tablets, and smartphones.

## Activity: Outline the responsive design strategies you will employ, such as media queries, flexible grid systems, and responsive images. Include examples of how your design will adapt to different screen sizes.

## 2. Backend Architecture

## 2.1 Server-Side Frameworks:

## Overview: Discuss the server-side framework you have selected for your project. Explain why this framework is appropriate for your application, considering scalability, performance, and security.

## Activity: Identify the backend framework (e.g., Node.js with Express, Django, Spring Boot) you will use and justify your choice based on the application’s needs.

## 2.2 RESTful API Design:

## Definition: RESTful APIs allow for communication between the client and server by defining a set of operations (e.g., GET, POST, PUT, DELETE) that can be performed on resources.

## Activity: Design the RESTful APIs for your application. This should include:

## A list of API endpoints with their respective HTTP methods.

## A description of each endpoint’s purpose and the data it will handle.

## Example requests and responses for each endpoint.

## 2.3 Database Design and Integration:

## Database Choice: Describe the database system (e.g., MySQL, PostgreSQL, MongoDB) you have selected and explain why it is suitable for your project.

## Schema Design: Provide a detailed schema of your database, including tables, fields, and relationships between entities.

## Integration: Explain how the database will be integrated with your backend, including how data will be accessed, manipulated, and secured.

## Activity: Create an Entity-Relationship Diagram (ERD) that visualizes the structure of your database. Detail how the backend will interact with the database through the RESTful APIs.

## 3. Framework Integration

## 3.1 Framework Integration Strategy:

## Overview: Describe how you will integrate the chosen frontend and backend frameworks. Discuss how these frameworks will work together to create a seamless user experience.

## Activity: Outline your integration strategy, detailing how data will flow between the frontend and backend. Include diagrams that illustrate the architecture of your application, showing how the different components will interact.

## 3.2 Code Optimization for Performance:

## Techniques: Discuss the techniques you will use to optimize your code for performance. This includes minimizing load times, reducing resource usage, and ensuring scalability.

## Activity: Identify specific code optimization techniques (e.g., code minification, lazy loading, caching) that you will implement in your project. Explain how these techniques will improve the performance of your application.

## 3.3 Security Considerations:

## Security Measures: Outline the security measures you will implement to protect your application from common threats, such as SQL injection, cross-site scripting (XSS), and data breaches.

## Activity: Detail the security best practices you will follow, including secure authentication methods, data encryption, and regular security audits. Explain how these measures will be integrated into both the frontend and backend.

## Design Document Submission

## Deadline: [Insert Deadline Date]

## Format: Submit the design document in a PDF format. The document should be clear, well-structured, and include all required sections.

## Evaluation Criteria: The design document will be evaluated based on the thoroughness of the UI design, the robustness of the backend architecture, the effectiveness of framework integration, and the attention to performance and security.

### **Project Design Document Rubric**

| **Criteria** | **5 - Excellent** | **4 - Good** | **3 - Satisfactory** | **2 - Needs Improvement** | **1 - Unsatisfactory** |
| --- | --- | --- | --- | --- | --- |
| **UI Design** | UI design is exceptionally clear, visually appealing, and fully aligned with project goals. Wireframes and mockups are detailed, functional, and demonstrate a deep understanding of user experience and responsive design. | UI design is clear, visually appealing, and mostly aligned with project goals. Wireframes and mockups are well-detailed and functional, with good consideration of user experience and responsive design. | UI design is adequate but may lack some clarity or visual appeal. Wireframes and mockups are somewhat functional but may need further refinement in user experience and responsiveness. | UI design is vague or lacks visual appeal. Wireframes and mockups are incomplete or do not adequately address user experience and responsive design. | UI design is unclear, visually unappealing, or missing. Wireframes and mockups are absent, incomplete, or poorly executed, with minimal consideration for user experience or responsiveness. |
| **Backend Architecture** | Backend architecture is exceptionally well-designed, scalable, and secure. The RESTful API design is comprehensive, with clear and well-documented endpoints. Database schema is well-structured, with clear relationships and data integrity. | Backend architecture is well-designed and mostly scalable and secure. RESTful API design is clear with well-documented endpoints. Database schema is adequately structured, with minor improvements needed. | Backend architecture is satisfactory but may lack some scalability or security features. RESTful API design is functional but may need better documentation. Database schema is somewhat structured but may lack clarity or completeness. | Backend architecture is incomplete or lacks scalability/security. RESTful API design is unclear or poorly documented. Database schema is poorly structured or lacks detail. | Backend architecture is missing or critically flawed. RESTful API design is absent, unclear, or non-functional. Database schema is missing, incomplete, or does not support the application’s requirements. |
| **Framework Integration** | Framework integration is seamless, with well-documented data flow and interaction between frontend and backend. Code optimization techniques are thoroughly implemented, ensuring high performance and scalability. Security considerations are comprehensive and well-integrated. | Framework integration is effective, with clear data flow between frontend and backend. Code optimization techniques are implemented but may need minor refinements. Security considerations are well-integrated but may need additional detail. | Framework integration is functional but may have gaps in data flow or interaction. Code optimization techniques are partially implemented, with room for improvement. Security considerations are included but may lack depth. | Framework integration is poorly executed, with significant gaps in data flow or interaction. Code optimization techniques are minimal or ineffective. Security considerations are incomplete or poorly integrated. | Framework integration is absent or critically flawed. Code optimization techniques are missing or irrelevant. Security considerations are absent or inadequate, leaving the application vulnerable. |
| **Clarity and Professionalism** | The design document is exceptionally well-organized, clearly written, and free of errors. It presents the design in a professional and convincing manner, with a strong emphasis on quality and detail. | The design document is well-organized, clearly written, with minor errors. It presents the design in a professional manner, with good attention to quality and detail. | The design document is adequately organized and written but may contain some errors or lack some professionalism. Quality and detail are sufficient but could be improved. | The design document is poorly organized, with significant errors or lack of clarity. Professionalism is lacking, and the document may not effectively communicate the design. | The design document is disorganized, difficult to read, with numerous errors and a lack of professionalism. Quality and detail are minimal or absent, making the document ineffective. |