One Clean Dude Detailing

Dakoda Meade

CST-452 Capstone Project Final Architecture & Design

Grand Canyon University

Instructor: Professor Mohammad Shahadat Ullah

Revision: 2

Date: 8/31/2024

**ABSTRACT**

The mobile automotive detailing company web application project aims to create a comprehensive, user-friendly platform that streamlines various aspects of the business. The application will host an online appointment booking, service reviews(hardcoded for now), and administrative tasks. The primary objective for this project is to enhance the user experience by providing a convenient way to schedule appointments and access services while enabling the company to manage operations more efficiently. This project is being undertaken as a favor for a friend, adding a personal commitment to deliver a high-quality application in a short amount of time. However, there will be more features added after the deadline is met for this class to ensure all the features promised for this class are done on time.

The web application will be developed using ASP.NET for the backend and MySQL for the database, ensuring a robust and scalable system. Following the Model-View-Controller (MVC) architecture, separating application business logic from the front-end logic. The User entity, with attributes such as role to differentiate between admins and customers, ensures proper access control and functionality tailored to different user types. The database schema includes tables for Users, Roles, Services, Appointments, and Appointment Statuses, essential for capturing all necessary data. The project's success will be measured by its functionality, user-friendly UI, and meeting the deadline. By integrating a Christian worldview, the project emphasizes ethical practices and community service, aligning goals with a commitment to moral values.

|  |
| --- |
| History and Signoff Sheet |

**Change Record**

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Revision Notes** |
| 7/31/2024 | Dakoda Meade | Initial draft for review/discussion |
| 8/31/2024 | Dakoda Meade | Final Draft and review |
|  |  |  |

|  |
| --- |
| **Overall Instructor Feedback/Comments** |

|  |
| --- |
| **Overall Instructor Feedback/Comments** |

**Integrated Instructor Feedback into Project Documentation**

Yes  No

**TABLE OF CONTENTS**

Design Overview 4

Detailed High-Level Solution Design 5

Detailed Technical Design 6

Appendix A – Technical Issue and Risk Log 7

Appendix B – References 8

Appendix C – External Resources 9

Design Introduction

1. Provide the high-level design of the proposed solution or business case with supporting narrative text. This design should include mock-up screen shots for the proposed user interface, pseudocode, or flowcharts that show the logic for the program, as well as the anticipated process flow. The purpose of the solution/business case design is to allow the stakeholder to approve the concepts before committing resources to the technical design.
2. Use the template to list the project deliverables that are to be included external to this Design Specification (Data Dictionary, API Design, etc.).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Deliverable Acceptance Log | | | | | |
| ID | Deliverable Description | Comments | Evaluator (internal or external as applicable) | Status | Date of Decision |
| 1 | Data Dictionary | Detailed documentation of the database schema, including table structures, relationships, and data types. | Internal | In Progress | TBD |
| 2 | MySQL DDL Scripts | Scripts to add the database with all necessary tables and columns. | Internal | In Progress | TBD |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |
| 5 |  |  |  |  |  |

***NOTE: If necessary, you may add subsections to those listed in order to match the requirements in the assignment description. Do not remove any top level sections (those that are listed in the Table of Contents).***

Detailed High-Level Solution Design

1. Provide a detailed overview of how the proposed design fits into the overall solution.
   1. Create diagrams to logically and physically depict the system. This can be illustrated using UML Component, UML Deployment, and UML Activity diagrams or simply block diagrams done in a drawing tool such as Visio.

A diagram of a diagram

Description automatically generated

UML Component Diagram for appointment information.

A diagram of a computer server

Description automatically generated

UML Deployment Diagram.

A diagram of a software program

Description automatically generated

UML Activity Diagram for scheduling an appointment.

* 1. This section should also include any solution configuration changes that will be required to develop and implement the proposed solution.
* Development Environment Setup:
  + - 1. Install Microsoft Visual Studio 2022 for the IDE.
* Server Configuration:
  + - 1. Install MAMP to set up Apache and MySQL server.
      2. Microsoft visual studio can run the web application for testing.
      3. Set up GitHub continuous integration and deployment (CI/CD) pipelines for automated deployments.
* Database Configuration:

1. Install MySQL workbench and setup schemas and tables.
2. Set up necessary indexes and relationships for efficient data retrieval.
   1. Describe the approach and resources required to assure non-functional requirements (such as security and performance) will be met with this solution.

* Security:

1. Use of registration and login for user authentication and authorization.
2. Utilize input validation, parameterized queries, and encryption so ensure best secure coding practices are used.

* Performance:

1. Use of optimized database queries.
2. Load testing is to be used to test for increased traffic.

* Reliability and Availability:

1. Set up a detailed logging system to detect and respond to issues promptly.
   1. The purpose of the detail solution architecture is to provide sufficient information for a developer to produce the system.

I am going to be using ASP.NET which uses MVC architecture. This solution architecture ensures that there is a clear understanding of the system’s components, structure and their interaction through a clear separation of logic.

* Models:

1. User Model: Model class that represents user data and information.
2. Service Model: Model class that represents service information.
3. Appointment Model: Model class that represents appointment details.

* Controllers:

1. Home Controller: Handles all the requests for the home page.
2. Appointments Controller: Controller class that handles all the requests for the appointments view which a list of the appointments model.
3. Appointment Controller: Similar to the appointments controller but only manages one appointment.
4. Reviews Controller: Controller class to handle request to the review view.
5. Schedule Appointment Controller: Controller class that manages appointment scheduling.
6. Register Controller: Handles registration for users.
7. Login Controller: Handles logging in for users.
8. Admin Dashboard Controller: Provides the administrative functions like managing services and appointments.
9. Service Controller:

* Views:

1. Home View: This view will explain what the company is about and have a schedule button for them to schedule an appointment.
2. Appointments View: This view is for displaying all the user’s upcoming and previous appointments.
3. Appointment View: View that displays appointment details and allows for canceling and completing the appointment.
4. Reviews View: Just displays some hardcoded made-up reviews.
5. Schedule Appointment View: This will be the view that allows a user to schedule an appointment. Will use JavaScript to allow doe client-side updates.
6. Register View: View to allow user to create an account to remember appointments and user information.
7. Login View: This view is to allow the user to log in after they create an account.
8. Admin Dashboard View: This view allows admins to manage services and appointments.
9. Service View: This is the view that allows an admin to view service details and be able to edit them.

* Services:

1. Security Service: Service class that allows the controllers to interact with the DAO classes for user CRUD operations.
2. Appointment Service: Service class that allows the controllers to interact with the DAO classes for appointment CRUD operations.
3. Service Service: Service class that allows the controllers to interact with the DAO classes for service CRUD operations.

* DAOs:

1. Security DAO: Data Access Object class that provides CRUD operations for the User Model by working with the database without implementation details.
2. Appointment DAO: Data Access Object class that provides CRUD operations for the Appointment Model by working with the database without implementation details.
3. Service DAO: Data Access Object class that provides CRUD operations for the Service Model by working with the database without implementation details.
4. Provide a detailed inventory of hardware and software technologies that will be used in the solution:
   1. List any Frameworks or third party libraries that will be used.

* **ASP.NET:** Web Application framework used to build the back-end logic and manage the data interactions with the user interface.
* **Entity Framework:** Used to interact with database for .NET applications.
* **Bootstrap:** Front-end framework for creating visually appealing, reactive, and responsive websites.
* **JavaScript/jQuery:** Library used to handle events and interactions on the client side of the application.
* **MySQL.Data:** Library to facilitate communication between the application and the database.
  1. List any Proof of Concepts to be completed (POC) to ensure that the technologies and frameworks selected are the best fit for purpose, cost effective, and proper to solve the problem. This section should also be updated with the purpose/rationale for the POC and the results of the POC.

Use the templates below to list the Proof of Concepts, hardware, and technologies.

|  |  |  |
| --- | --- | --- |
| Proof of Concepts | |  |
| **Description** | **Rationale** | **Results** |
| 1. Integrating ASP.NET application with MySQL. | Ensuring that there are no issues with ASP.NET interacting with MySQL for data storage and retrieval. | Be able to connect to that database and perform CRUD operations. |
| 2 - Implement some pages with Bootstrap. | Confirm Bootstrap is able to deliver the responsive design I am looking for. | Page displays correctly using developer mode in Google Chrome web browser simulation how it would appear on mobile and different resolution devices. |
| 3 – Using jQuery to make calls to the server without the need to reload a page. | Ensure jQuery can dynamically update pages without fully reloading the page. | Data retrieval and update achieved with no reloading of the pages. |
| 4 - |  |  |
| 5 - |  |  |

|  |
| --- |
| Hardware and Software Technologies |
| 1 – Visual Studio 2022 |
| 2 - My SQL Workbench |
| 3 - HTML5 |
| 4 - CSS3 |
| 5 - Bootstrap |
| 6 – ASP.NET MVC |
| 7 – MY SQL |
| 8 – C# |
| 9 – Entity Framework |
| 10 - |

**Logical Solution Design:**

Provide the proper diagrams and drawings that represent the high-level logical solution design.

A diagram of a person

Description automatically generated

Use case diagram.

A diagram of a service

Description automatically generated

Sequence Diagram of customer booking an appointment.

A diagram of a service

Description automatically generated

Sequence Diagram for an admin updating services.

**Physical Solution Design:**

Provide the proper diagrams and drawings that represent the high-level physical solution design.

A diagram of a computer

Description automatically generated

Screenshot of the Architecture Diagram.

Detailed Technical Design

**General Technical Approach:**

You should, in words, describe your approach and design here. You should also summarize any meeting notes, brainstorming sessions, etc. that you want to retain through the design of your project.

For the development of the automotive detailing company web application, the approach focuses on a structured and modular design to ensure maintainability and scalability. The project leverages ASP.NET MVC for the server-side logic, MySQL for database management, and standard web technologies such as HTML, CSS, and JavaScript for the front end. The design is structured around a Model-View-Controller (MVC) architecture, which separates concerns and allows for more manageable code and easier testing.

**Initial Planning Meeting**:

* Discussed project scope and objectives.
* Identified key features such as user login, appointment scheduling, and admin dashboard.
* Agreed on using ASP.NET MVC and MySQL as core technologies.

**Design Workshop Meeting**:

* Brainstormed user interface designs and layout.
* Created mockups for the login page, appointment booking page, and admin dashboard.
* Reviewed and finalized the data model and relationships for MySQL.

**Key Technical Design Decisions:**

Any final technical design decisions, such as framework decisions, etc., should be documented here. This should list the technology/framework, its purpose in the design, and why it was chosen. If necessary, the proper Proof of Concepts should be defined and implemented to validate the technical decision.

**Framework/Technology Decisions**

1. **ASP.NET MVC**

* **Purpose**: ASP.NET MVC (Model-View-Controller) is used to build the web application's backend. It provides a clear separation of concerns, allowing the application to handle user input, business logic, and data management efficiently.
* **Reason for Choice**: ASP.NET MVC was chosen due to its robust framework for building scalable and maintainable web applications. It offers built-in support for security features, data binding, and model validation. Additionally, it integrates well with the Microsoft ecosystem and has strong community support.

1. **MySQL**

* **Purpose**: MySQL serves as the relational database management system for storing and managing application data, including user information, appointments, and service details.
* **Reason for Choice**: MySQL was selected for its reliability, performance, and widespread use. It is cost-effective and offers strong support for complex queries and transactions. Its compatibility with ASP.NET through various connectors and libraries also influenced the choice.

1. **Bootstrap**

* **Purpose**: Bootstrap is used for front-end development to create responsive and visually appealing user interfaces. It provides pre-designed components and grid systems.
* **Reason for Choice**: Bootstrap was chosen for its ease of use, extensive documentation, and ability to create a responsive design that works across various devices. It helps streamline the UI development process and ensures a consistent look and feel.

1. **Entity Framework Core**

* **Purpose**: Entity Framework Core is used for Object-Relational Mapping (ORM) to interact with the MySQL database. It abstracts database operations and provides a more intuitive way to manage data.
* **Reason for Choice**: Entity Framework Core was selected due to its powerful ORM capabilities, support for LINQ queries, and integration with ASP.NET MVC. It simplifies data access and management, reducing the amount of boilerplate code needed.

1. **Razor Pages**

* **Purpose**: Razor Pages, part of the ASP.NET Core framework, is used for building dynamic web pages with a simplified syntax for server-side code.
* **Reason for Choice**: Razor Pages were chosen for their simplicity and ability to streamline the development of web pages, especially where minimal logic is required. It integrates seamlessly with ASP.NET MVC and enhances productivity.

1. **jQuery**

* **Purpose**: jQuery is used for client-side scripting to handle DOM manipulation and asynchronous operations.
* **Reason for Choice**: jQuery was chosen for its ease of use and compatibility with various browsers. It simplifies client-side interactions and AJAX requests, improving the user experience.

Proof of concepts were provided earlier.

**Database ER Diagram:**

A screenshot of a computer

Description automatically generated

Screenshot of current ER Diagram. This is subject to change as needed for the project.

**Database DDL Scripts:**

This should contain the DDL script showing all database constraints, etc.

These will be attached as MySQL files separately.

**Flow Charts/Process Flows:**

You should insert any flow charts or UML Activity diagrams here. Flow charts should document algorithms or workflows that will be implemented in your program.

A diagram of a company

Description automatically generated

A diagram of a flowchart

Description automatically generated

Flowchart of all ends of the application.

**Sitemap Diagram:**

Image file of your Sitemap diagram.

A diagram of a company

Description automatically generated

Sitemap diagram of the application.

**User Interface Diagrams:**

You should insert any wireframe drawings or white board concepts that were developed to support your application. If you have no supporting documentation, explain the rationale why you are able to leave this section as N/A.

A screenshot of a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a login form

Description automatically generatedA screenshot of a login form

Description automatically generated

A screenshot of a login form

Description automatically generatedA screenshot of a calendar

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated

**UML Diagrams:**

You should insert any UML Class diagrams and UML Sequence diagrams here. If you have no supporting documentation, explain the rationale why you are able to leave this section as N/A.

All UML class diagrams have been updated to match how I am currently building and plan on building the classes. These are subject to change as I go through the project.

A screenshot of a computer

Description automatically generated

UML Class Diagrams for the models.

A screenshot of a computer

Description automatically generated

UML Class Diagrams for the views with their controllers.

A screenshot of a computer program

Description automatically generated

UML Class Diagrams for the services classes with their DAO classes.

A diagram of a service

Description automatically generated

A diagram of a service

Description automatically generated

**Service API Design:**

This section should fully document any third party Service Interface APIs being consumed or application specific Service API’s being published, how to access the service, what parameters are required by the API, and the detailed JSON data format specification that could be used by a third party developer to integrate with the service and API.

There are no current API’s being used or any plans for a 3rd party API to be used.

**NFR’s (Security Design, etc.):**

This section should outline how non-functional requirements will be supported by the design.

1. Security:
   1. Authentication and Authorization: Using authentication and authorization for users. This comes with role-based access that controls access to parts of the application.
   2. Encryption: All sensitive data like passwords will be encrypted before being stored in the database.
   3. Input Validation: All input fields will be validated to prevent SQL injection attacks and other common web application attacks.
2. Performance:
   1. Efficient Database Queries: Queries to the database will be well written for quick database requests.
   2. Optimized Code: Code base will be written efficiently to minimize response times. Using asynchronous programming where appropriate.
3. Maintainability:
   1. Modular Architecture: MVC architecture is used to separate concerns and improve code organization.
   2. Documentation: Code-based documentation will be extensive and well detailed. This will make it easy for future changes to the code base.
   3. Automated Testing: Unit tests, integration tests, and end-to-end tests will be used to catch bugs early and ensure code quality.
4. Usability:
   1. Clean and Organized UI: The UI/UX will be clean and organized. Navigation will be consistent and easy to use.
5. Reliability:
   1. Logging: Will be used to catch any issues early on and help diagnose the issues.
   2. Error Handling: Robust error handling will ensure the application can recover from errors with ease and no crashing.

**Operational Support Design:**

This section should fully document how your design supports monitoring and logging.

Logging:

ILogger, a logger built-in logger with ASP.NET core, will be utilized for the logging of the application. Logging is an essential part of operational support, providing a record of application events, errors, and user activities. This enables developers and administrators to diagnose issues, understand application behavior, and maintain the health of the system.

1. Error Logging:
   1. Capture and log any and all errors that can occur during runtime.
   2. Ensure that enough details are logged in order to diagnose the issue.
2. Activity Logging:
   1. Log all user actions.
   2. Significant or not all events and actions should be logged for records.

Monitoring:

No monitoring plans currently in scope.

**Other Documentation:**

You should insert any additional drawings, storyboards, white board pictures, project schedules, tasks lists, etc. that support your approach, design, and project. If you have no supporting documentation, explain the rationale why you are able to leave this section as N/A.

Appendix A – Technical Issue and Risk Log

1. Use the template to identify and monitor project issues and risks.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Issues and Risk Log | | | | | | | | |
| **Issue or Risk** | **Description** | **Project Impact** | **Action Plan/Resolution** | **Owner** | **Importance** | **Date Entered** | **Date to Review** | **Date Resolved** |
| I/R | What is the issue or risk? | How will this impact scope, schedule, and cost? | How do you intend to deal with this issue? | Who manages this issue? |  |  |  |  |
| R | Data breach | Compromise of customer information | Implement robust security measures and monitor access | Dakoda | High | 7/29/24 | Weekly |  |
| R | Project Incomplete by Deadline | Inability to deliver the project on time | Implement agile methodologies and regularly review progress | Dakoda | High | 7/29/24 | Daily |  |
| R | Technical Issues | Potential delays and additional costs due to tech problems | Plan ahead for any technical issues. | Dakoda | Medium | 7/29/24 | Daily |  |
| I | Scope Creep | Uncontrolled changes or continuous growth in project scope | Define clear project requirements and enforce change control procedures | Dakoda | Medium | 7/29/24 | Weekly |  |
| I | Stakeholder Availability | The application is being made for my friend so I need his input on decisions and he might not have availability. | Schedule regular meetings and provide multiple communication channels | Dakoda | Medium | 7/29/24 | Weekly |  |
| I | Outside Interferences | I have a baby on the way and a lot of work, unforeseen doctors’ appointments and issues with work can cause reduced time to work. | Plan ahead of time and make sure I plan for any issues that can occur. | Dakoda | Medium | 7/29/24 | Weekly |  |

Appendix B – References

*List all Project Documentation References*

*List all references using APA style*

Appendix C – External Resources

|  |  |
| --- | --- |
| **GIT URL:** | *https://github.com/DakodaMeade/OCDD* |
| **Hosting URL:** | *The Hosting URL (if applicable).* |