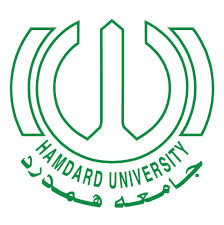
Hamdard University

Department of Computing

Final Year Project



**One-Source Construction**

**(FYP-025/FL24)**

**Software Design Specifications**

Submitted by

MadadAllah (1725-2021)

Hanzala Shahzad (2603-2021)

Burhan Khan (840-2018)

Supervisor(s)

Sir Muhammad Salman

Co-Supervisor  
Sir Maaz Ahmed

**Fall 2024**

**Document Sign off Sheet**

### Document Information

|  |  |
| --- | --- |
| **Project Title** | OneSource Construction |
| **Project Code** | FYP-025/FL24 |
| **Document Name** | Software Requirements Specifications |
| **Document Version** | 1.0 |
| **Document Identifier** | FYP-025/FL24-SDS |
| **Document Status** | Initial Version |
| **Author(s)** | MadadAllah, Hanzala, Burhan |
| **Approver(s)** | Sir Muhammad Salman Sir Maaz Ahmed |
| **Issue Date** | 22-May-2024 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Role** | **Signature** | **Date** |
| MadadAllah | Team Lead | WhatsApp Image 2025-01-19 at 19.27.39_c5544901 | 17/01/2025 |
| Hanzala Shahzad | Team Member 2 | sign | 17/01/2025 |
| Burhan Haider | Team Member 3 | WhatsApp Image 2025-01-19 at 19.29.50_0776ed58 | 17/01/2025 |
| Sir Muhammad Salman | Supervisor | WhatsApp Image 2025-01-19 at 19.32.43_ffcb19e2 | 17/01/2025 |
| Sir Maaz Ahmed | Co-Supervisor |  | 17/01/2025 |
| Ms. Maria Nasir Khan | Project Coordinator |  | 17/01/2025 |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 15-01-2025 | 1.0 | Initial | MadadAllah, Hanzala, Burhan |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Definition of Terms, Acronyms, and Abbreviations**

|  |  |
| --- | --- |
| **Term** | **Description** |
| One-Source | A centralized platform for managing construction services and products. |
| Contractor | Executes construction projects, e.g., building houses or offices. |
| Supplier | Provides construction materials like cement, bricks, and steel. |
| Agile | A flexible project management methodology using iterative development. |
| Scrum | Agile framework with daily meetings to track progress. |
| Frontend | The user-facing part of the platform (HTML, CSS, JavaScript). |
| Backend | The server-side logic (Node.js, Python/Django). |

# Table of Contents

[1. Introduction 8](#_Toc4677804)

[1.1 Purpose of Document 8](#_Toc4677805)

[1.2 Intended Audience 8](#_Toc4677806)

[2. Overall System Description 9](#_Toc4677808)

[2.1 Project Background 9](#_Toc4677809)

[2.2 Problem Statement 9](#_Toc4677810)

[2.3 Project Scope 9](#_Toc4677811)

[2.4 Not In Scope 9](#_Toc4677812)

[2.5 Project Objectives 9](#_Toc4677813)

[2.6 Stakeholders & Affected Groups 9](#_Toc4677814)

[2.7 Operating Environment 9](#_Toc4677815)

[2.8 System Constraints 9](#_Toc4677816)

[2.9 Assumptions & Dependencies 9](#_Toc4677817)

[3. External Interface Requirements 10](#_Toc4677818)

[3.1 Hardware Interfaces 10](#_Toc4677819)

[3.2 Software Interfaces 10](#_Toc4677820)

[3.3 Communications Interfaces 10](#_Toc4677821)

[4. System Functions / Functional Requirements 11](#_Toc4677822)

[4.1 System Functions 11](#_Toc4677823)

[4.2 Use Cases 12](#_Toc4677824)

[4.2.1 List of Actors 12](#_Toc4677825)

[4.2.2 List of Use Cases 12](#_Toc4677826)

[4.2.3 Use Case Diagram 13](#_Toc4677827)

[4.2.4 Description of Use Cases 13](#_Toc4677828)

[5. Non - Functional Requirements 16](#_Toc4677829)

[5.1 Performance Requirements 16](#_Toc4677830)

[5.2 Safety Requirements 16](#_Toc4677831)

[5.3 Security Requirements 16](#_Toc4677832)

[5.4 Reliability Requirements 16](#_Toc4677833)

[5.5 Usability Requirements 16](#_Toc4677834)

[5.6 Supportability Requirements 16](#_Toc4677835)

[5.7 User Documentation 16](#_Toc4677836)

[6. References 17](#_Toc4677837)

# Introduction

## Purpose of Document

*The purpose of this Software Requirements Specification (SRS) document is to outline the functional and non-functional requirements, architecture, and interaction mechanisms of the One-Source Construction platform. This document provides a detailed framework for developers, testers, and stakeholders to ensure the system is designed and developed effectively, meeting the needs of the construction industry.*

## Intended Audience

* *Development Team*
* *Project Supervisor*
* *Testers*
* *End-Users (Constructor,Supplier,Customer)*

## Document Convention

* *Headings: Arial, Bold, Size 16*
* *Subheadings: Arial, Bold, Size 14*
* *Body Text/Descriptions: Arial, Regular, Size 12*
* *Table Fields: Arial, Regular, Size 10*

## Project Overview

*OneSource Construction is a digital platform designed to streamline the construction process for customers in Pakistan by centralizing all construction-related services and products. The system will serve as a "one-window solution," enabling users to find contractors, purchase construction materials, and manage their construction projects efficiently.*

***3.4.1 Key Functionalities***

*Contractor Services: Users can browse and hire contractors for various projects, with details such as experience, ratings, and cost estimates provided for informed decision-making.*

*Material Marketplace: Suppliers will list construction materials like cement, steel, bricks, and sand on the platform, allowing users to compare prices and quality.*

*Project Management Tools: The platform will include tools to track budgets, materials, timelines, and quality.*

*Price Recommendations: The system will analyze previous ratings and prices to recommend optimal costs for services and materials.*

*User-Friendly Interface: An intuitive design ensures that users with minimal technical expertise can use the platform effectively.*

***3.4.2 Basic Design Approach***

*The One-Source Construction platform will adopt a modular, client-server architecture:*

*Frontend: Developed using HTML, CSS, and JavaScript to create a responsive and user-friendly interface.*

*Backend: Built with Python (Flask/Django) or Node.js, coupled with MySQL for efficient data storage and management.*

*Agile Methodology: The project will use the Scrum framework, emphasizing iterative development, continuous feedback, and rigorous testing.*

*Cloud Integration: AWS and Docker will be utilized for scalability, performance optimization, and secure deployment.*

## Scope

*In Scope:*

*Centralized platform for contractors and material suppliers.*

*Price comparison and recommendations for services and materials.*

*Tools to manage construction timelines, budgets, and quality.*

*User-friendly interface suitable for non-technical users.*

*Not in Scope:*

*Integration with external construction management systems.*

*Real-time syncing with external financial or government databases.*

*Automated quality checks for construction projects.*

# Design Considerations

*User Accessibility:*

*Issue: Users may have varying levels of technical expertise.*

*Consideration: Ensure the platform is intuitive, with clear navigation and minimal input errors.*

*Action: Focus on a minimalistic interface with clear labels, tooltips, and large interactive elements.*

*Data Storage and Scalability:*

*Issue: Large volumes of data, including supplier profiles, project details, and pricing information, must be stored and managed efficiently.*

*Consideration: Implement a scalable database to handle growth.*

*Action: Use MySQL with robust indexing and backup mechanisms.*

*Performance Optimization:*

*Issue: Potential performance bottlenecks due to large datasets and concurrent users.*

*Consideration: Optimize backend processing and data retrieval.*

*Action: Incorporate caching, load balancing, and optimized queries for high performance.*

*Secure Transactions:*

*Issue: Sensitive customer and payment data need protection.*

*Consideration: Ensure secure communication and storage of user data.*

*Action: Use HTTPS, secure authentication protocols, and data encryption.*

*Training and Support:*

*Issue: Users unfamiliar with technology may need assistance.*

*Consideration: Offer extensive help resources.*

*Action: Include FAQs, video tutorials, and customer support options.*

## Assumptions and Dependencies

***4.1.1 Modular Design for Future Extensibility***

*Assumption: The system may need to accommodate additional functionalities, such as support for advanced analytics or integration with third-party construction tools in the future.*

*Design Impact: The architecture should be modular, enabling new features to be added or updated without requiring significant changes to the core system.*

*Solution: Adopt a microservices architecture for the backend, with independent modules for contractor management, material marketplace, user management, and project tracking. This ensures extensibility while isolating the impact of new features.*

***4.1.2 Secure Payment Handling***

*Assumption: The platform will handle online transactions for purchasing materials and hiring contractors.*

*Design Impact: The system must integrate secure payment gateways while ensuring compliance with financial regulations.*

*Solution: Use established payment gateways like Stripe or PayPal, and implement encryption (SSL/TLS) for secure transaction handling.*

***4.1.3 User Support and Feedback Mechanisms***

*Assumption: Non-technical users may require guidance and support while using the platform.*

*Design Impact: The system must include features to address user issues and gather feedback for ongoing improvements.*

*Solution: Incorporate an in-app FAQ section, chat support, and feedback submission forms to provide assistance and gather user insights.*

***4.1.4 Cloud-Based Deployment***

*Assumption: The platform will require scalable and reliable hosting to handle varying traffic loads.*

*Design Impact: The infrastructure must support dynamic scaling and high availability.*

*Solution: Use cloud services like AWS or Azure for hosting, with autoscaling and load-balancing capabilities.*

## Risks and Volatile Areas

***4.2.1 Changing User Requirements***

*Risk: Customers or contractors may request new features or adjustments after the platform's launch, such as enhanced contractor rating systems or custom reporting tools.*

*Mitigation Strategy: Implement a feedback mechanism to continuously gather user suggestions. Use Agile methodology to accommodate iterative updates efficiently.*

***4.2.2 Data Growth and Storage Limitations***

*Risk: As the number of users increases, the volume of data (e.g., material orders, project details, user profiles) may outgrow the current storage capacity.*

*Design Approach: Use a scalable relational database (e.g., MySQL) with support for schema migrations and optimized indexing. Plan for periodic database maintenance and archiving.*

***4.2.3 Payment Security Vulnerabilities***

*Risk: Unauthorized access to payment transactions or data breaches could compromise user trust and platform integrity.*

*Mitigation Strategy: Implement robust security measures, such as tokenization, two-factor authentication (2FA), and secure APIs for payment gateway integration.*

***4.2.4 User Interface and Flow Modifications***

*Risk: User feedback may necessitate frequent changes to the platform's interface or workflows, potentially impacting system logic.*

*Design Approach: Decouple the presentation layer from the business logic and backend using RESTful APIs, allowing independent updates to the user interface without disrupting core functionality.*

***4.2.5 Technology Obsolescence***

*Risk: Rapid advancements in web and mobile technologies could render parts of the platform outdated.*

*Mitigation Strategy: Use widely supported and actively maintained technologies (e.g., Python, React Native). Plan periodic technology reviews and updates.*

# System Architecture

*The OneSource Construction platform follows a modular architecture, dividing functionality into distinct components to ensure scalability, maintainability, and flexibility. This architecture facilitates independent development, testing, and deployment of features while allowing seamless integration between components.*

*Overview of Architecture*

*Presentation Tier: Responsible for user interaction through a responsive, intuitive interface.*

*Business Logic Tier: Processes user requests, manages workflows, and handles data manipulation.*

*Data Tier: Stores and retrieves structured information such as user profiles, contractor details, and project data.*

## System Level Architecture

***5.1.1 User Interface (UI) Layer***

*Role: Provides an easy-to-use front-end experience where users can browse contractors, purchase materials, and manage projects.*

*Design Consideration: The interface prioritizes simplicity to accommodate users with varying technical expertise.*

*Interaction: Communicates with the business logic layer via API calls to fetch or update data dynamically.*

***5.1.2 Business Logic Layer***

*Role: Acts as the core processing unit, handling requests, workflows, and data validation.*

*Design Consideration: Modular design ensures new features can be added without disrupting the existing system.*

*Interaction: Interacts with the UI for user actions and with the data layer for storage and retrieval operations.*

***5.1.3 Data Management Layer***

*Role: Manages all data storage and retrieval tasks using a relational database (MySQL).*

*Design Consideration: Includes schema migration capabilities for future enhancements and data integrity checks.*

*Interaction: Interfaces with the business logic layer to provide secure, scalable access to stored data.*

***5.1.4 External System Integration***

*Future Integration: While the current system is self-contained, APIs will allow future integrations with third-party systems such as payment gateways or analytics tools.*

## Software Architecture

*The platform is designed using a three-layer architecture to ensure separation of concerns:*

***5.2.1 User Interface Layer***

*Role: Provides a responsive front-end developed using HTML, CSS, and JavaScript.*

*Interaction: Receives user inputs and forwards them to the business logic layer while displaying processed data.*

***5.2.2 Middle Tier (Business Logic Layer)***

*Role: Implements core application workflows, manages data validation, and enforces business rules.*

*Interaction: Acts as a bridge between the UI and data layers, ensuring data consistency and managing API calls.*

***5.2.3 Data Access Layer***

*Role: Provides secure and efficient access to the database for storing and retrieving data such as user accounts, contractor details, and project logs.*

*Interaction: Receives and processes requests from the middle tier, ensuring data integrity and optimal performance.*

***5.2.4 Major Physical Design Considerations***

*Cloud-Based Deployment: The platform will be hosted on AWS to ensure scalability and high availability.*

*Device Compatibility: Designed to work seamlessly on desktops, tablets, and mobile devices.*

*Error Handling:*

*Input Validation: Ensures all data input by users is consistent and valid.*

*Exception Handling: Provides clear error messages, such as “Network error, please try again,” to ensure users are not left confused.*

*Internet Dependency: The platform requires an active internet connection for real-time operations but includes local data storage for offline use, syncing once connectivity is restored.*

# *Design Strategy*

*The design strategy for OneSource Construction emphasizes robustness, scalability, security, and user accessibility to ensure the platform meets the needs of users in the construction industry. The key principles guiding this strategy include:*

***6.1.1 User Interface Paradigms***

*Strategy: A simple, intuitive, and responsive design will be used, prioritizing ease of navigation and usability for customers and contractors with varying levels of technical proficiency.*

*Clear labels, large buttons, and logical workflows will guide users through the platform’s functionalities, such as hiring contractors and purchasing materials.*

*Reasoning: A user-friendly interface ensures higher adoption rates by reducing the learning curve, particularly for users less familiar with technology.*

***6.1.2 Data Management (Storage, Distribution, Persistence)***

*Strategy: A relational database (MySQL) will be employed to handle structured data such as user accounts, contractor profiles, project timelines, and material inventory.*

*Data backup mechanisms will ensure integrity and security, while cloud integration will enable scalability.*

*Offline data persistence will allow users to save information locally and sync with the server when an internet connection is available.*

*Reasoning: Relational databases are reliable and efficient for managing large datasets, and cloud storage ensures scalability as user demand grows. Offline persistence improves usability for users in areas with inconsistent connectivity.*

***6.1.3 Concurrency and Synchronization***

*Strategy: The backend will use asynchronous processing and load balancing to handle concurrent user requests effectively. This includes managing multiple simultaneous operations like contractor search, material purchases, and project updates.*

*Reasoning: Efficient concurrency handling ensures system responsiveness and reliability, even during peak usage. This prevents bottlenecks and ensures data consistency.*

***6.1.4 Future System Extension or Enhancement***

*Strategy: A modular and flexible architecture will allow the addition of future features, such as advanced analytics, third-party integrations, and enhanced reporting tools, without disrupting the core system.*

*Reasoning: Scalability and modularity enhance the platform's adaptability to changing user needs and technological advancements.*

***6.1.5 Security and Privacy***

*Strategy: Data encryption, secure authentication mechanisms (e.g., two-factor authentication), and secure communication protocols (e.g., HTTPS) will be implemented to protect user data and transactions.*

*Reasoning: Robust security measures are essential for gaining user trust, particularly when handling sensitive information like payment details.*

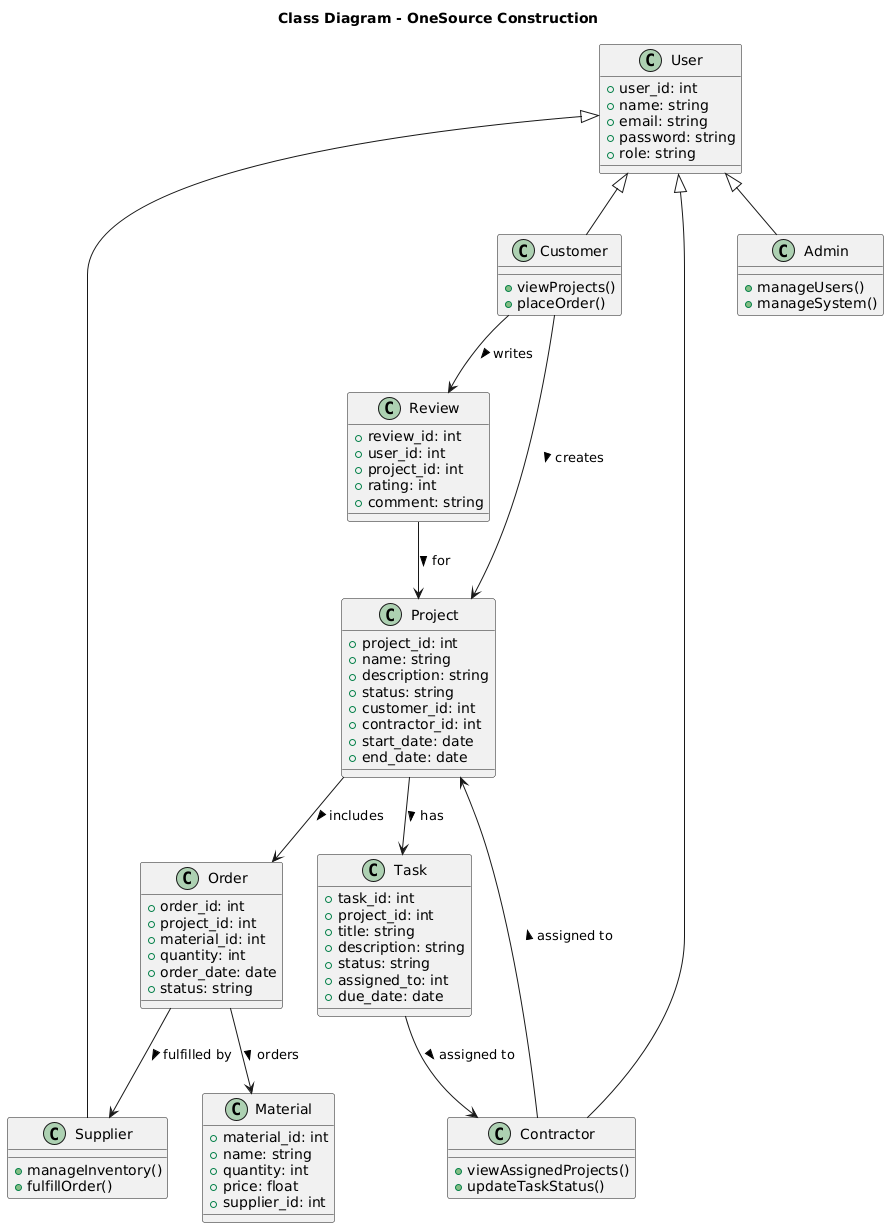
***6.1.6 System Reuse***

*Strategy: Reusable components, such as APIs for user management, contractor listings, and payment processing, will be implemented to promote consistency and reduce development effort for future updates or features.*

*Reasoning: Reusability reduces development time and effort while maintaining consistency across system functionalities.*

# *Detailed System Design*

# *Design Class Diagram*



## Database Design

#### *Logical Data Model (E/R Model)*

*The ER model contains the following key entities:*

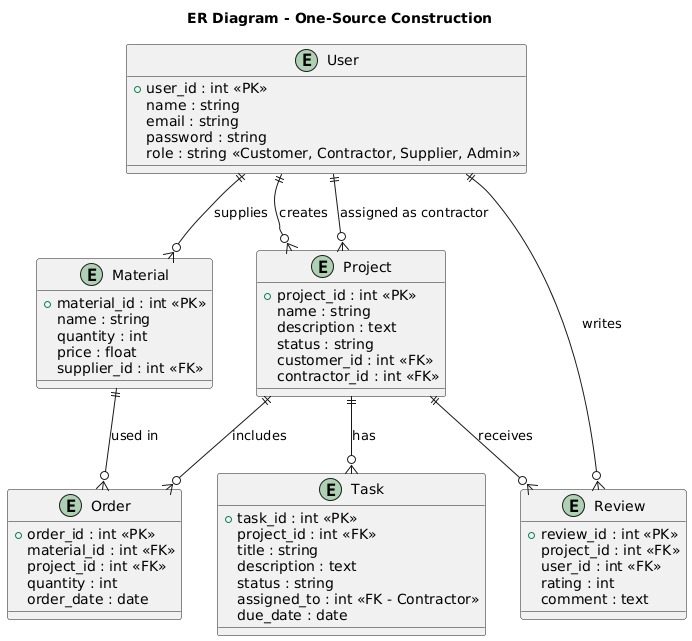
* *User****:*** *Primary key UserID.*
* ***Project****:* *Foreign key UserID, Primary key ProjectID.*
* *Contractor****:*** *Primary key ContractorID.*
* ***Material****:* *Foreign key SupplierID, Primary key MaterialID.*
* *Order****:*** *Foreign keys UserID, MaterialID, Primary key OrderID.*

#### *Detailed GUIs*

*The GUIs will include:*

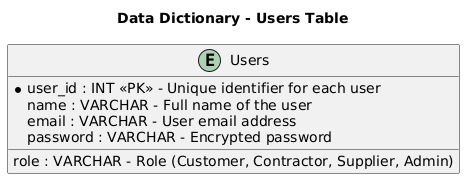
* *Home Page****:*** *Navigation for contractors, suppliers, and customers.*
* ***User Dashboard****:* *A detailed overview of projects, orders, and reviews.*
* *Material Marketplace****:*** *Listing of available materials with price comparison and availability.*
* ***Project Management Interface****:* *Tools for tracking budgets, timelines, and contractor assignments.*

### *ER Diagram*

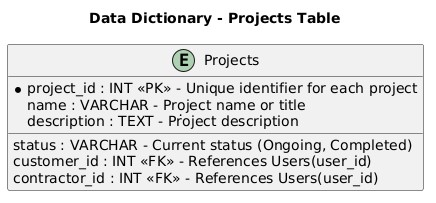


### *Data Dictionary*

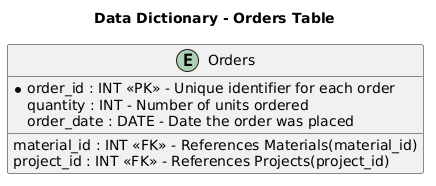
### **Data 1 user table**



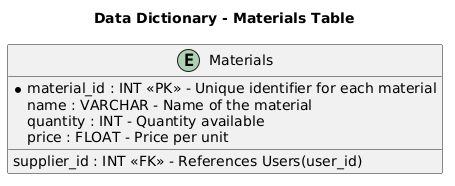
### **Data 2 Project Table**



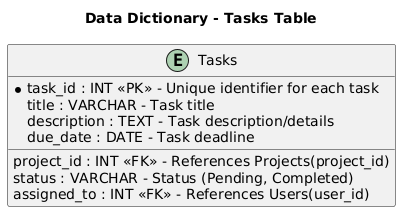
#### **Data 3 Order Table**



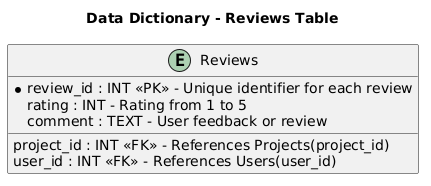
#### **Data 4 Materials Table**



#### **Data 5 Task Table**



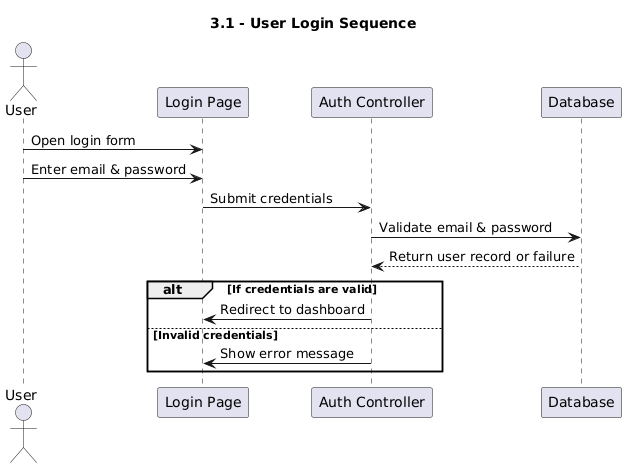
#### **Data 6 Reviews Table**



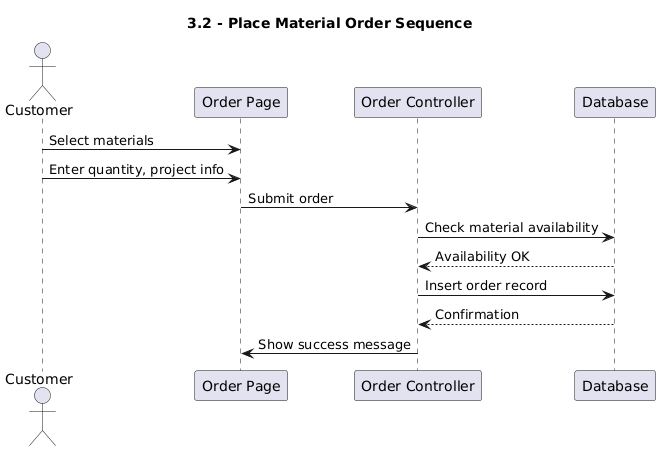
## Application Design

### Sequence Diagram

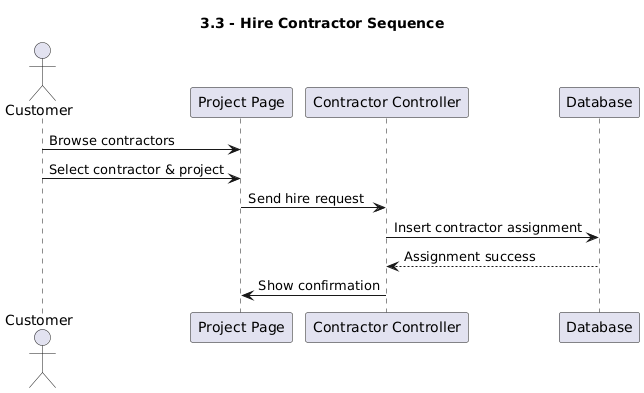
### User Login Sequence



### Place Material Order Sequence

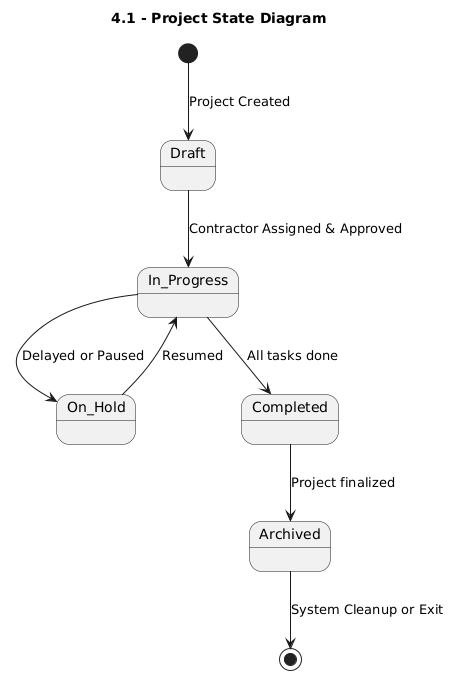


### Hire Contractor Sequence

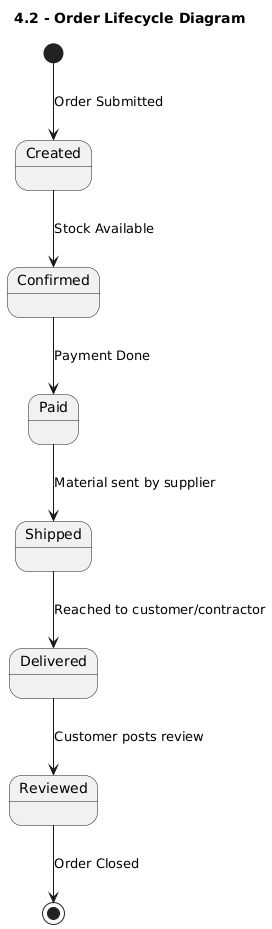


### State Diagram

### Project State Diagram



### Order Lifecycle Diagram



# References

* Valentino, Jonathan. "Web Development with Node.js and Express." PacktPublishing, 2020.
* Banks, Artemij. "React.js Essentials." Packt Publishing, 2015.
* Martin, Robert C. "Agile Software Development: Principles, Patterns, andPractices." Prentice Hall, 2002.
* Chodorow, Kristina. "MongoDB: The Definitive Guide." O'Reilly Media, 2013.
* Baron, Michael. "Cloud Computing with AWS." Packt Publishing, 2016.