HuaChang Growmax

Software Design and Research Report

Musang King

List of your Names:

Name	**Position**	**email**	**phone**
Benjamin Tan Chen Hern	Leader	104477174@student.swin.edu.au	+601110660387
Wallace Iglesias Chandrio	Member	104180579@student.swin.edu.au	+60166082100
Hein Htet Naing	Member	104329055@student.swin.edu.au	+60134942987
Mahanthe Acharige Sachindri Sudeepa Chandrasiri	Member	104338967@student.swin.edu.au	+60108932476

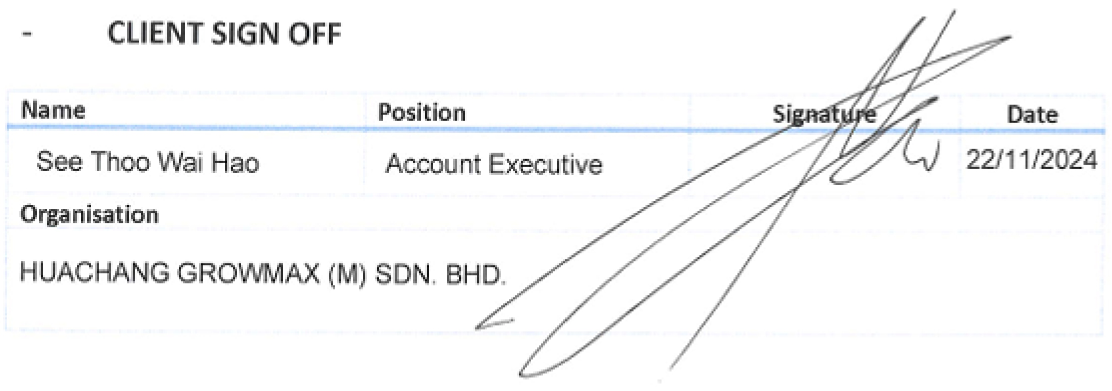
SWE40001 Software Engineering Project A, Year 2 Semester 2, 27 / 10 / 2024

1 DOCUMENT CHANGE CONTROL

Version	Date	Authors	Summary of Changes
1	27/10/24	Benjamin Tan Chen Hern Wallace Iglesias Chandrio Hein Htet Naing Mahanthe Acharige Sachindri Sudeepa Chandrasir	Finished the first version of the Software Design and Research Report (SDRR)

- DOCUMENT SIGN OFF

Name	Position	Signature	Date
Benjamin Tan Chen Hern	Leader	[illegible]	17/09/2024
Wallace Iglesias Chandrio	Member	[illegible]	17/09/2024
Hein Htet Naing	Member	[illegible]	17/09/2024
Mahanthe Acharige Sachindri Sudeepa Chandrasiri	Member	[illegible]	18/09/2024



- CLIENT SIGN OFF

Name	Position	Signature	Date
See Thoo Wai Hao	Account Executive	[illegible]	22/11/2024
Organisation			
HUACHANG GROWMAX (M) SDN. BHD.			

1. Introduction

This Software Design and Research Report details the development of a web-based order management system for Hua Chang Growmax. The goal of the system is to make ordering easier for warehouse workers, administrators, and salespeople. It would include functions including client management, order status monitoring, product availability, and safe payment processing using an online banking interface. This report analyzes the issue being addressed, describes the design methodology for the system, and offers solutions based on the specified software requirements.

This document's objective is to present a thorough summary of the system's research and design elements. In order to understand the design methodology, underlying architecture, and technical choices, developers, project managers, and stakeholders are supposed to use it. The client (Hua Chang Growmax), the development team, and other project stakeholders are among the target readers.

1.1 Overview

The design and development of the order management system are guided by this guideline. It offers:

- a high-level examination of the issue that the system is meant to address.
- a summary of the aims, purposes, and presumptions of the system.
- a thorough explanation of the architecture, design, and interactions of the system.
- a summary of the technologies and research used to satisfy the system's needs.

After analyzing the system requirements from the SRS, the paper discusses the design solutions and important factors that were taken into account during the development process.

1.2 Definitions, Acronyms and Abbreviations

GUI: Graphical User Interface – a visual interface that users interact with.

SRS: Software Requirements Specification – a document that outlines the system’s requirements.

API: Application Programming Interface – a set of tools for building software and facilitating communication between software components.

UAT: User Acceptance Testing – the process of testing the system by the end users to ensure it meets their needs.

HTTPS: Hypertext Transfer Protocol Secure – a secure communication protocol for transmitting data over the web.

DBMS: Database Management System – software that manages and organizes databases.

2. **Problem Analysis**

A high-level evaluation of the web-based order management system's Software/System Requirements Specification (SRS) is provided in this part. In order to provide a solution that effectively and efficiently meets the demands of the client, the analysis focuses on the system's goals, objectives, and requirements. This study makes sure the system design satisfies the technical and operational requirements of its users while also being in line with Hua Chang Growmax's overall business objectives.

2.1. **System Goals and Objectives**

The system's primary objective is to give Hua Chang Growmax a web-based order management platform that makes daily tasks easier for administrators, salespeople, and warehouse staff. The system seeks to increase overall efficiency by combining order management, stock availability, customer management, and secure payment processing.

Key objectives include:

- Order management: Salespeople are able to originate, modify, and monitor orders at any stage. Customers can view order progress once the order status has been updated.
- Stock Availability: Salespeople can reduce stockouts and delays by adjusting orders based on automated stock checks that confirm product availability.
- Customer management: By having access to customer profiles, which include previous purchases and preferences, salespeople may provide better customer care and make well-informed decisions.
- Role-based Access: The system offers safe, role-specific access, guaranteeing that users only engage with essential information and assignments, upholding security and accountability.
- Payment Integration: Customers may safely and effectively conduct purchases thanks to the platform's integration with an online payment gateway.
- Notifications: To improve communication, the system notifies stakeholders about order status, delivery, and possible problems.

These goals are in line with the system specifications, guaranteeing that the platform satisfies both technical and user requirements.

2.2. **Assumptions**

In order to keep the plan effective and in line with Hua Chang Growmax's expectations, a number of assumptions were made throughout the system design development process.

- Internet connection: In order to communicate with the system, it is expected that every user—including salespeople, administrators, and warehouse staff—will have dependable and constant internet connection. Due to the web-based nature of the platform, order management tools, transaction processing, and order status communication all depend on constant internet connectivity.

- User Knowledge: It is also assumed that users will be technically proficient. It is expected of salespeople, administrators, and warehouse staff to be able to use the web interface, enter order information, and update data as needed. To guarantee that every user can do duties like creating orders, checking stocks, and updating statuses effectively, training may be given.

- Third-party Integration: Throughout the system's duration, successful integration with third-party systems, like online payment gateways like TouchnGo and QR Payment, is presumed. To enable smooth financial transactions for clients placing orders, the payment system needs to be operational and always accessible. Assuming that these external services won't experience any interruptions, the system's architecture supports these integrations.

2.3. **Simplifications (if any)**

The system design was simplified in a number of ways to control development complexity and guarantee on-time delivery. The goal of these simplifications is to strike a compromise between functionality and the project's time and resource limitations.

- Stock Accuracy: It is acknowledged that the stock data does not always need to be 100% accurate, even if the system will include automated stock availability checks. Although the system is built with flexibility to allow salesmen to make manual adjustments, it will deliver stock information based on current revisions. Without putting undue pressure on maintaining 100% accuracy at all times, this simplification guarantees that stock levels are instructive and aid in directing sales decisions.

- Basic Payment Integration: To speed up the payment process, the system will first be coupled with a single online banking payment gateway (such as TouchnGo or QR Payment). Future revisions may take into account additional payment methods in response to user input and changing company requirements. This enables the project to concentrate on a single safe and dependable payment option for its initial deployment phase.

- Scalability: As Hua Chang Growmax's operations expand, the system is built to be scalable, meaning it can manage a growing number of users, orders, and data. The architecture will enable the system to accommodate more users or higher transaction volumes in the future without requiring major redesigns, even if the current implementation concentrates on the essential features needed by salespeople, administrators, and warehouse staff. This guarantees that even as the company grows, the system will continue to be dependable and effective.

3. High–Level System Architecture and Alternatives

This section outlines the high-level architecture of the Sales Order Application for Hua Chang Growmax. The chosen architecture is designed to support a scalable, secure, and user-friendly web-based order management system. Additionally, two alternative architectures are discussed, along with the rationale for selecting the chosen architecture over these alternatives.

3.1. System Architecture

The system follows a three-layer architecture, divided into Presentation Layer, Application Layer, and Data Layer. This layered approach provides separation of concerns, allowing each layer to handle specific responsibilities, thus enhancing maintainability, scalability, and security.

1. Presentation Layer

- Purpose: This layer provides the user interface for salespeople, administrators, and warehouse staff to interact with the system.
- Components:
    - User Interface (UI): A web-based GUI accessible through web browsers on both desktop and mobile devices. It includes screens for order creation, order tracking and inventory checks.
    - Authentication: This component ensures that users are authenticated via secure login. Role-based access controls are implemented to allow users only to access functionalities pertinent to their roles.
- Interaction: The Presentation Layer interacts with the Application Layer via secure API calls, sending user requests and displaying responses, such as order status or inventory availability.

2. Application Layer

- Purpose: Acts as the core processing unit of the system, handling business logic and coordinating data flow between the Presentation and Data Layers.
- Components:
    - Order Management Module: Manages the creation, modification, and tracking of orders, updating order status and facilitating order-related notifications.
    - Inventory Management Module: Checks product availability in real-time and communicates stock information to users in the Presentation Layer.
    - Customer Management Module: Handles customer data retrieval and management, allowing salespeople to view customer profiles and order histories.
    - Notifications Module: Sends notifications to users (e.g., order confirmations, status updates) as needed to enhance communication.

- Interaction: The Application Layer communicates with the Data Layer for CRUD (Create, Read, Update, Delete) operations on data and with the Presentation Layer to display processed data back to the users.

3. Data Layer

- Purpose: Stores and retrieves data used by the system, including order details, inventory data, customer information, and user credentials.
- Components:
    - Database Management System (DBMS): Houses all data for the system in an organized, relational format. It includes separate tables for orders, inventory, customers, and users.
    - Data Access Layer (DAL): Manages interactions between the Application Layer and the DBMS, ensuring secure and efficient data access.
    - Backup and Recovery: Provides mechanisms for regular data backups and recovery, ensuring data integrity and minimizing potential data loss.
- Interaction: The Data Layer responds to queries and updates from the Application Layer, allowing data retrieval and storage as required by the business logic.

3.2. **Other Alternative Architectures Explored**

1. Monolithic Architecture

In a monolithic approach, the entire application is built as a single, unified codebase, combining all three layers into a single deployable unit.

- Pros:
    - Easier initial setup and deployment due to its single codebase.
    - Fewer complexities with data sharing since all components reside in a single application.
- Cons:
    - Scalability issues as the application grows, making it challenging to add new features without significant rework.
    - Limited flexibility in scaling specific components or modules independently.
    - Higher risk of system downtime; an error in one part of the application could impact the entire system.
- Reason for Rejection: Given Hua Chang Growmax's expected growth, a monolithic approach would limit future scalability and flexibility. It also increases maintenance challenges and reduces resilience, making it less suited for a growing, multi-user system.

2. Microservices Architecture

A microservices approach divides the application into smaller, independent services, each responsible for a specific functionality, such as order management or customer management.

- Pros:
    - Enhanced scalability as each service can be scaled independently based on demand.
    - Increased fault tolerance since issues in one service do not necessarily impact others.
    - Flexible technology stack, allowing different services to use different programming languages or databases if needed.
- Cons:
    - Higher complexity in initial setup, as each microservice requires its own deployment and management.
    - More challenging inter-service communication, requiring robust APIs and potential overhead for data consistency.
    - Increased operational burden, especially for smaller teams, due to managing multiple services.
- Reason for Rejection: While microservices offer long-term benefits, the complexity and resources required to manage multiple services would exceed the scope and resources available for this project. For a student project with limited time and budget, a simpler architecture is more practical.

Justification for Chosen Architecture

The three-layer architecture was selected as it strikes a balance between simplicity and scalability. By separating the Presentation, Application, and Data Layers, this architecture allows:

- Scalability: The system can be scaled by enhancing specific layers as Hua Chang Growmax's business grows.
- Maintainability: Each layer has a defined role, making it easier to troubleshoot and modify.
- Security: Sensitive data and business logic are safeguarded in separate layers, reducing exposure to potential security risks.

The three-layer architecture effectively meets the needs of Hua Chang Growmax while remaining manageable for the development team.

4. **Detailed Design (using Object Orientation or alternative)**

The goal of the system is to create a web-based order management platform for Hua Chang Growmax, focused on improving daily operations for administrators, salespeople, and warehouse staff. Core functions include order management, stock availability tracking, customer management, role-based access, secure payment processing, and notifications.

To achieve this, we'll follow an object-oriented design (OOD) approach, specifically using responsibility-driven design (RDD) to structure components based on clear responsibilities, relationships, and role assignments.

Justification for Using OOD/RDD

- Clear Responsibility Assignment: RDD structures classes and objects around their responsibilities and collaborations. This makes it easier to maintain and extend as each part of the system has a single focus.
- Modularity: OOD's emphasis on modularity is aligned with agile development, enabling iterative design and testing.
- Scalability and Reusability: By assigning clear roles, RDD supports growth, as each component can evolve independently while staying cohesive with the overall design.

While domain-driven design (DDD) could also be considered, it would add unnecessary complexity given that the system's primary needs center around operational efficiency and user interaction rather than domain modeling. Therefore, RDD is a more suitable choice.

4.1. **The Detailed Design and Justification**

Object-Oriented Design (OOD) with Responsibility-Driven Design (RDD)

For the **GrowMax Sales Order System**, we will implement an object-oriented design (OOD) approach, specifically using responsibility-driven design (RDD). This structure allows us to model each component in the system based on its role, responsibilities, and collaboration needs, enhancing the overall system's modularity and scalability.

Core Components and Class Diagram

The design comprises several key components represented by classes, each with specific responsibilities. Below is a high-level overview of the core classes:

1. **Sales Coordinator**:
   - **Responsibilities**: Manages the entire order lifecycle, including creation, updates, and order status tracking.
   - **Collaborations**: Collaborates with Sales Person to check customer details and confirm transaction details.
2. **Salesperson**:
   - **Responsibilities**: Manages customer details, transaction details and updates stock of products.
   - **Collaborations**: Works with Sales Coordinator to confirm and sales transaction approval.
3. **Manager**:
   - **Responsibilities**: Has the authorities of both Sales Person and Sales coordinator. Oversees the entire transaction workflow.
   - **Collaborations**: Since the manager oversees the entire transaction lifecycle, they will need to work with both the salesperson and sales manager. On top of that, the manager will report straight to the master admin.

4. **Master Admin**:
    - **Responsibilities**: Holds the highest authorities of the sales order system. Has the same authorities as a manager plus the ability to create new accounts for the system.
    - **Collaborations**: Works with the manager to get reports of the transactions.
5. **NotificationService**:
    - **Responsibilities**: Manages email or SMS notifications related to orders.
    - **Collaborations**: Collaborates with salesperson and sales manager to send updates and confirmations.

Design Patterns Used

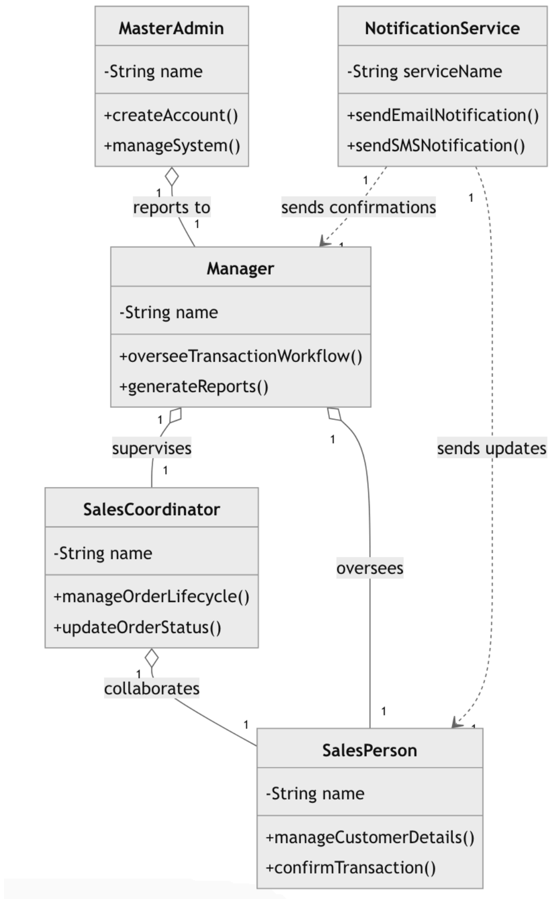
- ***Singleton Pattern**: Applied to NotificationService and UserAuthentication to ensure a single instance handles notifications and user authentication across the system.*
- ***Observer Pattern**: Implemented in NotificationService to enable real-time notifications on order updates.*
- ***Facade Pattern**: Used in Sales Person to provide a simple interface to handle the complexities of managing an order.*

Justification for Using OOD with RDD

- ***Clear Responsibility Assignment**: RDD structures classes and objects based on specific responsibilities and relationships, making each component easier to maintain and test individually.*
- ***Modularity**: The OOD approach aligns with modular development, allowing each component to be developed and tested independently, which is especially beneficial in an agile environment.*
- ***Scalability and Reusability**: RDD supports future scalability, as the system can expand by adding or modifying components without impacting the rest of the system.*

While alternative methods like Domain-Driven Design (DDD) offer in-depth domain modeling, they would add unnecessary complexity in this case, as the focus of this system is operational efficiency rather than domain intricacies.

Class Diagram



4.2. **Design Verification**

To verify this design, we will evaluate whether the system supports each use scenario from the Software Requirements Specification (SRS). Key scenarios include:

1. **Order Placement and Tracking**:
    - The design allows Salesperson to place orders using OrderManager, which interacts with InventoryManager to verify stock and CustomerManager to retrieve customer details. The NotificationService sends order confirmations, verifying that the order management process is handled efficiently and supports the specified requirements.
2. **Real-Time Inventory Updates**:
    - As part of each order transaction, InventoryManager updates stock levels. The WarehouseStaff and Admin roles can also access inventory data, and real-time alerts are provided by NotificationService if levels fall below set thresholds. This ensures that the system meets the real-time tracking and alerting requirement.
3. **Role-Based Access Control**:
    - The UserAuthentication and RoleManager classes ensure that only authorized users (e.g., Admins, Salespersons) can access specific functions. For example, only Admin can modify system settings, while Salesperson can place orders but not modify inventory.
4. **Notifications**:
    - NotificationService verifies that order confirmations, updates, and stock alerts are sent to relevant users. This ensures that users are informed at critical points in the process.

Through these checks, we validate that the design satisfies all key system functionalities and adheres to requirements, meeting operational needs and enhancing user interaction and efficiency for Huachang Growmax’s sales order process.

5. **Research and Investigations**

During the requirements analysis and design stages, various research areas were explored to understand and effectively address the needs of Hua Chang Growmax's sales order application.

1. **Domain Understanding**: The business model and order process requirements were analyzed to ensure the system would streamline tasks for salespeople, administrators, and warehouse staff. Key needs like order tracking, stock availability, and role-based access control were emphasized.
2. **System Design Research**: The report examined design patterns and architecture styles to enhance modularity, scalability, and security. The three-layer architecture (Presentation, Application, Data) was chosen to allow a clear separation of concerns and manageable growth as the business expands. Alternatives like monolithic and microservices architectures were considered but ultimately set aside due to scalability or complexity concerns.

3. **Technology Selection**: Research into suitable web-based technologies, security protocols, and databases was conducted to ensure reliable and secure transactions. This included API-driven communication for real-time updates and HTTPS for secure data transmission. Object-Oriented Design (OOD) principles with Responsibility-Driven Design (RDD) were selected to facilitate maintainable, modular development.

This research and investigation phase ensured that each aspect of the system was tailored to meet technical, operational, and future scalability needs.

6. References

- Fowler, M 2015, *Microservices: A Definition of This New Architectural Term*, MartinFowler.com, viewed 26 October 2024, https://martinfowler.com/articles/microservices.html
- Fowler, M 2015, *Monolithic First*, MartinFowler.com, viewed 3.October 2024, https://martinfowler.com/bliki/MonolithFirst.html