

|  |  |  |
| --- | --- | --- |
| Module Code | : | AAPP003-4-2  Introduction to Software Engineering |
| Intake Code | : | UCDF2307ICT(SE) |
| Lecturer Name | : | Ts. Dr. Law Foong Li |
| Hand-in Date | : | 10-12-2024 |
| Tutorial Group | : | Tutorial 10 |
| Group No. | : | Group 33 |
| Group Leader | : | Mah Zheng Yang |

|  |  |
| --- | --- |
| TP Number | Team Member |
| TP076394 | Anderson Tey Le Tien |
| TP074894 | Chek Ka Nic |
| TP074822 | Ching Ke Jing |
| TP076546 | Chong Wei Jie |
| TP076666 | Mah Zheng Yang |

**Table of Contents**

[Group Component 4](#_Toc184762816)

[1.0 Introduction 4](#_Toc184762817)

[1.1 Project Background 4](#_Toc184762818)

[1.2 Objectives 4](#_Toc184762819)

[1.3 Scopes 4](#_Toc184762820)

[2.0 Planning and Requirements Analysis 5](#_Toc184762821)

[2.1 Explanation of The Software Process 5](#_Toc184762822)

[2.2 Gantt Chart 8](#_Toc184762823)

[2.4 Functional and Non-functional Requirements 9](#_Toc184762824)

[3.0 Logical Design (Part 1) 13](#_Toc184762825)

[3.1 Use Case Diagram 13](#_Toc184762826)

[3.2 Class Diagram 20](#_Toc184762827)

[4.0 Physical Design 22](#_Toc184762828)

[5.0 Implementation 32](#_Toc184762829)

[5.1 Hardware Requirements 32](#_Toc184762830)

[5.2 Software Requirements 32](#_Toc184762831)

[6.0 Prototype 33](#_Toc184762832)

[7.0 Conclusion 33](#_Toc184762833)

[7.1 Limitation 33](#_Toc184762834)

[7.2 Challenges 33](#_Toc184762835)

[7.3 Further Enhancement 33](#_Toc184762836)

[Individual Component 34](#_Toc184762837)

[8.0 Logical Design (Part 2) 34](#_Toc184762838)

[8.1 Anderson Tey Le Tien TP076394 34](#_Toc184762839)

[8.2 Chek Ka Nic TP074894 36](#_Toc184762840)

[8.3 Ching Ke Jing TP074822 38](#_Toc184762841)

[8.4 Chong Wei Jie TP076546 40](#_Toc184762842)

[8.5 Mah Zheng Yang TP076666 42](#_Toc184762843)

[9.0 Methodology 44](#_Toc184762844)

[9.1 Prototyping Model (Anderson Tey Le Tien TP076394) 44](#_Toc184762845)

[9.2 Waterfall Model (Chek Ka Nic TP074894) 45](#_Toc184762846)

[9.3 SSADM (Ching Ke Jing TP074822) 48](#_Toc184762847)

[9.4 Spiral Model (Chong Wei Jie TP076546) 52](#_Toc184762848)

[9.5 V-Model (Mah Zheng Yang TP076666) 54](#_Toc184762849)

[10.0 Testing 57](#_Toc184762850)

[10.1 White Box Testing (Anderson Tey Le Tien TP076394) 57](#_Toc184762851)

[10.2 Black Box Testing (Chek Ka Nic TP074894) 58](#_Toc184762852)

[10.3 Black Box Testing (Ching Ke Jing TP074822) 58](#_Toc184762853)

[10.4 White Box Testing (Chong Wei Jie TP076546) 59](#_Toc184762854)

[10.5 Black Box Testing (Mah Zheng Yang TP076666) 59](#_Toc184762855)

[References 60](#_Toc184762856)

# Group Component

## 1.0 Introduction

### 1.1 Project Background

The Swiss Garden Hotel, which is centrally located in Kuala Lumpur, offers dining option entertainment, event hosting and hotel reservations. However, its reliance on a manual system to manage visitor statistics, payment, reservations, event booking and reporting has become insufficient and prone to errors as demand has grown. To solve these problems, the hotel’s owner, Mr. David, recommends implementing a computerized system to enhance decision-making, customer service and process speed.

### 1.2 Objectives

The objectives are to improve customer service, reduce errors, boost operational effectiveness and provide real-time data for informed decision-making. Future enhancements like online reservations and personalized services will be made possible by the system’s adaptable architecture. By using this technology, the Swiss Garden Hotel intends to keep its competitive edge, raise guest happiness and simplify operations.

### 1.3 Scopes

The Swiss Garden Hotel Management System will automate hotel reservations, event planning, billing and payment processing. Centralized data administration, automated booking confirmations, real-time availability checks, precise billing and support for multiple payment methods are some of its advantages. In order to assist with strategic planning, it will also produce real-time analytics and reporting on revenue, events and occupancy.

## 2.0 Planning and Requirements Analysis

### 2.1 Explanation of The Software Process

**Specification**

The primary objectives of the specification phase, the initial stage of software development process, is to identify and define the features that the finished product should have. In order to determine both functional requirements which are specific tasks that the system must complete and non-functional requirements which are quality elements like performance, security and usability, it entails collecting requirements from stakeholders. Several requirements are listed in order to develop a distinctive and recognized software vision. Since it guarantees that the development team is aware of the issue domain and user requests, this phase forms the basis for all others.

**Design and Implementation**

Through Design and Implementation, the needs are converted into a workable system and a well-organized plan. During the design phase, the system architecture is created, typically using models such as UML diagrams to show the components, relationship and procedures. This phase includes developing complex design for the operations, data structures and user interface. Implementation is the process of writing the actual code using programming languages and development tools to make the concept a reality. This iterative phase often includes prototyping, testing during software development and fine-tuning the program to ensure it fulfils the concept.

**Validation**

Validation is the process of ensuring that the produced software meets its specifications and performs as planned. At this stage, a variety of testing methodologies are employed such as unit testing (testing individual components), integrating testing (testing the relationship between modules), system testing (testing the entire system) and acceptance testing (testing by customers). To ensure that the software is reliable and ready for deployment, errors must be identified and corrected at this point.

**Evolution**

Evolution is the ongoing process of improving, updating and changing software after it has been implemented. It involves fixing issues, improving performance and adding new features in order to adjust to changing needs or take advantage of new opportunities. Software will remain relevant, helpful and efficient as long as keeps involved to meet changing business needs and technological breakthroughs. To guide updates and enhancement, this ongoing cycle usually necessitates user participation and system performance monitoring.

**Implementation of Software Process to Case Study**

**Specification**

Collecting specific demands from stakeholders such as hotel personnel and owner, Mr. David, is one of the prerequisites for the Swiss Garden Hotel Management System. Functional needs include things like enabling online reservations for visitors, giving event planners the authority to plan events and automating the invoicing and payment processes. Non-functional criteria include things like system dependability, data security for sensitive customer information and usability for inexperienced hotel staff. At this stage, the development team is informed of the hotel’s requirements as well as the challenges posed by the manual procedures that are currently in use.

**Design and Implementation**

As part of the design process for the Swiss Garden Hotel, models and diagrams are created to construct the system’s architecture. A use case diagram illustrates the connections between system activities (booking, billing, report generation) and system users (guests, receptionists, event coordinators). A class diagram can be used to identify entities like Guest, Room, Event and Payment as well as their attributes and relationships. The major objective of interface design is to create user-friendly interfaces for things like making hotel reservations and planning events.  
Implementation would involve developing a system prototype with tools such as Figma for interface design or Axure clickable mockup. Programming languages like Python or Java could be used to provide functional components like automatic room availability checks, report generating and invoice computation. In this phase, a functional system prototype is created from the conceptual designs.

**Validation**

Validation ensures that the prototype or implemented system functions in compliance with the specifications of the hotel system. To guarantee correctness, individual modules like room booking tools would go through unit testing. Effective communication between components such as billing and room availability checks would be ensured by integration testing. To ensure that everything functions as it should, including managing end-to-end duties like processing reservations, generating receipts and updating visitor data, the system would be tested. Hotel staff would evaluate the prototype as part of User Acceptance Testing (UAT) to ensure that it meets their operational needs and is easy to use. Future enhancements would be guided by UAT feedback.

**Evolution**

The Swiss Garden Hotel Manegement System would require ongoing improvements and updates if it were implemented. For instance, based on feedback from employees and clients, the system may offer extra features like a connection to a mobile app for hotel reservations or a customer loyalty program to reward frequent visitors. Performance improvements and bug fixes would be made to ensure smooth operations. The system may need to handle additional data or link to other systems for improved analytics as the hotel grows. This phase ensures that the system will remain relevant and meet the evolving needs of the hotel throughout time.

### 2.2 Gantt Chart

A graph with multiple colored bars

Description automatically generated with medium confidence

This Gantt chart illustrates the four primary phases of the Swiss Garden Hotel Management System development project: specification, design and implementation, validation, and evolution. The specification phase's weeks 1-4 are devoted to activities like obtaining requirements, confirming the platform's functionality, and evaluating business processes. Weeks four through nine of the Design & Implementation phases include building a physical model, designing the user interface, and beginning development. Weeks nine through twelve are set out for validation activities like acceptability, system, and unit testing. Weeks 14–16 will be dedicated to the last stage, Evolution, which focuses on adding additional features. This schedule offers a precise road map for finishing tasks and setting project objectives.

**2.3 Workload Matrix**

A screenshot of a document

Description automatically generated

### 2.4 Functional and Non-functional Requirements

#### 2.4.1 Functional Requirements

|  |  |
| --- | --- |
| Actor | Functional Requirement |
| Customer | * Check room availability for specific date * Book room online and specify preferences * Receive booking confirmation via email or SMS * Submit event booking requests * Receive event booking confirmation and updates * Pay bills using multiple methods * Receive receipts * Provide feedback on services |
| Staff | * Manually add, update or cancel room booking * View room availability * Assign rooms to guests * Add or update event bookings with details * Coordinate with other department via system * Add or update guests’ records * Search and retrieve guest history for repeat visitor * Generate and print bills for guests * Record payments made via cash, cards or other methods * Track outstanding balances for guests or events * Receive reminders for upcoming check-in, check-out or events |
| Administrator | * Add, update or remove user accounts for staff * Assign access permission for staff * Generate and review reports * Export reports in any formats of file * Monitor inventory levels for hotel supplies and equipment * Receive alerts for low stock levels |

#### 2.4.2 Non-functional Requirements

|  |  |  |
| --- | --- | --- |
| Non-functional Requirement | Detail | |
| Product Requirement | Performance Requirement | * The system must handle at least 100 concurrent users without performance degradation * Response times for room availability checks, bookings and report should not exceed 3-5 seconds * Report must be generated within 5 seconds |
| Usability Requirement | * The system must provide an intuitive and user-friendly interface for customers and staff. * Interfaces should be mobile-responsive and accessible on tablets, smartphones, and desktops. * Multilingual support should be available for diverse users * The system must comply with accessibility standards |
| Reliability Requirement | * System uptime must be 99.9%, with minimal downtime for maintenance. * Automated daily backups should prevent data loss and support disaster recovery. * Error handling mechanisms must ensure minor faults do not disrupt overall functionality. |
| Security Requirement | * All sensitive data (e.g., payment details, personal information) must be encrypted during storage and transmission. * Role-based access control must restrict features based on user roles (e.g., receptionist, manager). * Multi-factor authentication is required for staff login. * Regular security audits should be conducted to identify and mitigate vulnerabilities. |
| Organizational Requirement | Operational Requirement | * Automated notifications should be sent to relevant departments (housekeeping, event management) based on customer interactions (e.g., room bookings, check-outs). * Regular automated backups should be performed to ensure data security and prevent data loss. |
| Development Requirement | * The system must be developed using Agile methodology to enable iterative development and constant improvement. * A modular design approach should be used to allow easy updates, feature additions, and future scalability. |
| Environmental Requirement | * Optimize the system for minimal power consumption to reduce environmental impact. * Utilize cloud platforms with sustainability initiatives |
| External Requirement | Regulatory Requirement | * The financial management system should adhere to industry standards for accounting and taxation. * Online payment processing must meet PCI DSS (Payment Card Industry Data Security Standards) for secure handling of payment information. |
|  | Legislative Requirement | * Personal data collected from guests must be managed and protected in accordance with privacy laws such as GDPR and local legislation. * Legal agreements with third-party service providers (e.g., payment processors, booking platforms) must be documented and enforced. |
|  | Ethical Requirement | * The system should ensure that guest data is only used for legitimate purposes and protected against unauthorized access. * The system must follow transparency guidelines, providing users with clear information about how their data will be used. |

## 3.0 Logical Design (Part 1)

### 3.1 Use Case Diagram

**Use Case Diagram**

A diagram of a network

Description automatically generated

**Use Case Description**

**Guest Use Cases**

**1. Make Room Reservation**

|  |  |
| --- | --- |
| **Use Case Name** | Make Room Reservation |
| **Actors** | Guest, Hotel Staff |
| **Goal** | To reserve a room for the guest. |
| **Preconditions** | The guest must have an account, and the hotel must have available rooms. |
| **Postconditions** | The room is reserved, and the guest receives a confirmation. |
| **Main Flow** | 1. Guest logs into the system 2. Guest selects room type and dates. 3. Staff checks room availability and confirms the reservation. 4. Guest receives reservation confirmation. |
| **Alternate Flows** | A1.1 If no rooms are available, the system notifies the guest of unavailability.  A1.2 If guest did not select room type and dates, the system prompt user to fill in all required fields.  A1.3 If guest encounter poor internet signal, the system will show error message and ask user to reload the page. |
|  |  |

**2. Booking an Event**

|  |  |
| --- | --- |
| **Use Case Name** | Booking an Event |
| **Actors** | Guest, Hotel Staff |
| **Goal** | Reserve a venue for an event with specified details. |
| **Preconditions** | The event venue must be available for the requested date and time. |
| **Postconditions** | The event is booked, and the guest receives a confirmation. |
| **Main Flow** | 1. Guest logs into the system. 2. Selects event date, duration, and venue. 3. Specifies guest count and activities. 4. Hotel staff checks availability. 5. Guest receives booking confirmation. |
| **Alternate Flows** | A2.1 If no venue is available, staff will suggest alternate dates or venues.  A2.2 If guest did not specifies some event details, the system prompt user to fill in all required fields.  A2.3 If guest encounter poor internet signal, the system will show error message and ask user to reload the page. |
|  |  |
|  |  |

**3. Booking a Taxi**

|  |  |
| --- | --- |
| **Use Case Name** | Booking a Taxi |
| **Actors** | Guest |
| **Goal** | Request transportation to a specified location. |
| **Preconditions** | The guest must specify pick-up/drop-off locations and passenger details. |
| **Postconditions** | Taxi is booked, and confirmation is sent to the guest. |
| **Main Flow** | 1. Guest selects taxi booking option. 2. Provides required details (pickup, drop-off, passenger count). 3. System confirms availability. 4. Guest receives booking confirmation. |
| **Alternate Flows** | A3.1 If no transportation is available, notify the guest with alternatives.  A3.2 If guest did not select pick-up/drop-off locations and passenger details, the system prompt user to fill in all required fields.  A3.3 If guest encounter poor internet signal, the system will show error message and ask user to reload the page. |

**4. Requesting Room Service**

|  |  |
| --- | --- |
| **Use Case Name** | Requesting Room Service |
| **Actors** | Guest |
| **Goal** | Request services such as cleaning or food delivery to the room. |
| **Preconditions** | The Guest must be checked into a room. |
| **Postconditions** | The requested service is provided. |
| **Main Flow** | 1. Guest accesses room service options. 2. Selects desired service (cleaning, food). 3. System records the request. 4. Staff fulfils the request. |
| **Alternate Flows** | A4.1 If service is unavailable (e.g., kitchen closed), notify the guest. |
|  | A4.2 If guest did not select room service option, the system prompt user to fill in all required fields.  A4.3 If guest encounter poor internet signal, the system will show error message and ask user to reload the page. |

**5. Make Payment**

|  |  |
| --- | --- |
| **Use Case Name** | Make Payment |
| **Actors** | Guest |
| **Goal** | Complete payment for services or bookings. |
| **Preconditions** | The guest must have an outstanding balance. |
| **Postconditions** | Payment is processed, and a receipt is issued. |
| **Main Flow** | 1. Guest accesses payment section. 2. Selects payment method (cash, card). 3. System processes payment. 4. Receipt is generated and sent to the guest. |
| **Alternate Flows** | A5.1 If payment fails, request retry or an alternate method. |
|  | A5.2 If guest did not have sufficient balance in the banking account, request retry or an alternate method.  A5.3 If guest encounter poor internet signal, the system will show error message and ask user to reload the page. |

**6. Log into Account**

|  |  |
| --- | --- |
| **Use Case Name** | Log into account |
| **Actors** | Guest |
| **Goal** | Authenticate the user and grant access to the system. |
| **Preconditions** | The user must have a valid account with login credentials. |
| **Postconditions** | The user is authenticated and granted access to their dashboard. |
| **Main Flow** | 1. User navigates to the login page. 2. Enters username and password. 3. System verifies credentials. 4. Grants access to the user’s account dashboard. |
| **Alternate Flows** | A6.1 If credentials are invalid, prompt the user to retry or recover the account.  A6.2 If guest encounter poor internet signal, the system will show error message and ask user to reload the page. |

**7. Give Feedback**

|  |  |
| --- | --- |
| **Use Case Name** | Give Feedback |
| **Actors** | Guest |
| **Goal** | Allow guest to provide feedback or rate their experience. |
| **Preconditions** | |  | | --- | | The guest must have used a service or made a booking. |  |  | | --- | |  | |
| **Postconditions** | Feedback is stored in the system for review by staff. |
| **Main Flow** | 1. Navigates to the feedback section. 2. Selects a service or experience to review. 3. Provides feedback and ratings. 4. Submits feedback. 5. System stores the feedback and sends a confirmation. |
| **Alternate Flows** | A7.1 If feedback submission fails, notify the customer and request retry. |
|  | A7.2 If guest encounter poor internet signal, the system will show error message and ask user to reload the page. |

**Staff Use Cases**

**1. Manage Guest Activities**

|  |  |
| --- | --- |
| **Use Case Name** | Manage Guest Activities |
| **Actors** | Hotel Staff |
| **Goal** | Organize and monitor guest activities. |
| **Preconditions** | The guest must be registered in the system. |
| **Postconditions** | Guest activities are logged and managed. |
| **Main Flow** | 1. Staff views guest activity requests. 2. Assigns resources or services. 3. Updates guest activity records. |
| **Alternate Flows** | A1.1 If resources are unavailable, notify guests or offer alternatives. |
|  | A1.2 If staff encounter poor internet signal, the system will show error message and ask user to reload the page. |

**2. Generate Total Bill**

|  |  |
| --- | --- |
| **Use Case Name** | Generate Total Bill |
| **Actors** | Hotel Staff |
| **Goal** | Provide a comprehensive billing summary for a guest. |
| **Preconditions** | All guest services must be logged. |
| **Postconditions** | Bill summary is generated and issued to the guest. |
| **Main Flow** | 1. Staff retrieves guest service records. 2. Calculates total charges. 3. Generates a detailed bill. 4. Provides the bill to the guest. |
| **Alternate Flows** | A2.1 If discrepancies are found, reconcile charges before generating the bill.  A2.2 If staff encounter poor internet signal, the system will show error message and ask user to reload the page. |

**3. Generate Annual Report**

|  |  |
| --- | --- |
| **Use Case Name** | Generate Annual Report |
| **Actors** | Hotel Staff |
| **Goal** | Produce a comprehensive report for management review. |
| **Preconditions** | The system must have records of financial and operational data. |
| **Postconditions** | The annual report is generated and shared with management. |
| **Main Flow** | 1. Staff accesses the reporting module. 2. Specifies report parameters (year, metrics). 3. System generates the report. 4. Staff reviews and submits the report to management. |
| **Alternate Flows** | A3.1 If data is incomplete, request additional inputs before generating. |
|  | A3.2 If staff encounter poor internet signal, the system will show error message and ask user to reload the page. |

### 3.2 Class Diagram

A diagram of a computer

Description automatically generated with medium confidence

The provided class diagram represents a detailed design of Swiss Garden Hotel Management System, illustrating the key entities, their attributes, methods, and relationships. The primary actors in the system are the Guest and Staff, each represented as separate classes. The Guest class captures user details such as username, email, and phone number, and provides functionalities like booking rooms, events, and taxis, as well as requesting room services and leaving feedback. Each guest is associated with multiple transactions, including room reservations, event bookings, and payments, demonstrating a many-to-one relationship with classes like Room, Event, Taxi, and Payment.

The Staff class, on the other hand, manages administrative tasks such as approving bookings, checking availability, confirming taxi arrangements, and generating reports. The staff interacts with core system entities like Room, Event, and Taxi, performing operations to ensure smooth management. Additionally, the Staff class is associated with the Report class, allowing staff members to generate summary reports, which is crucial for hotel performance tracking.

The diagram also highlights transactional and service-oriented components such as Payment, Feedback, and Room Service. The Payment class manages financial transactions, linking directly to the Guest class and enabling guests to complete bookings or services via specified payment methods. The Feedback class allows guests to provide ratings and comments on their experience, creating a link between customer satisfaction and system functionality. Furthermore, the Room Service class captures additional services requested by guests, such as cleaning or extra amenities, showcasing its strong composition relationship with the Room class.

Key relationships in the diagram include the many-to-many association between Guest and Room/Event/Taxi, reflecting that a guest can book multiple services, while staff ensures these bookings are approved and managed. The inclusion of multiplicity notations such as 1..\* and 0..\* ensures clarity in how classes interact. Overall, this class diagram provides a comprehensive view of the system's structure, emphasizing the functional and relational dependencies that underpin the efficient operation of the hotel management system.

## 4.0 Physical Design

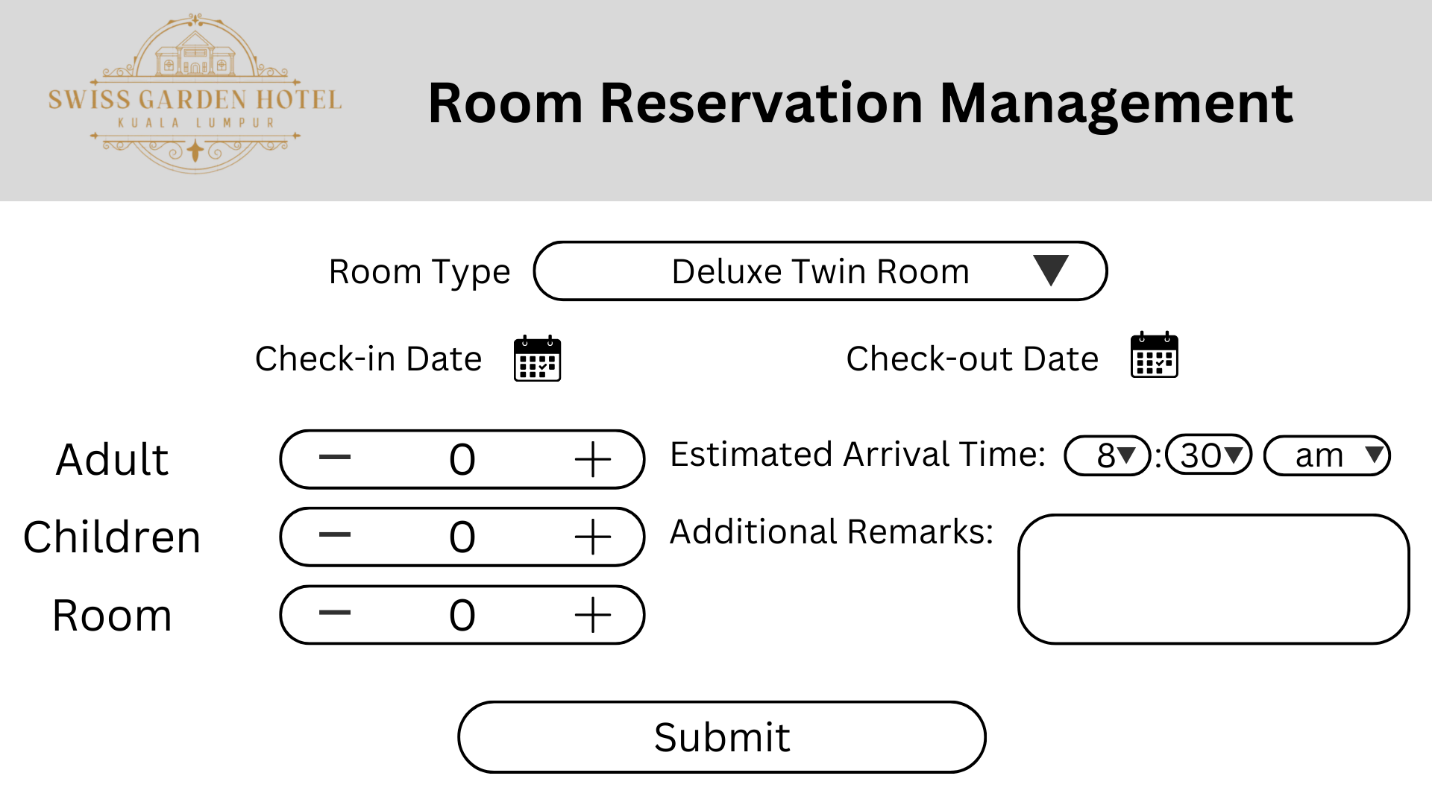
**Menu Page**



This is the logo of Swiss Garden Hotel. When user clicks this logo at the menu page, the page will be refreshed.

These six buttons are the main function for admin to use. Users can click the function they want to perform, and it will bring user to the page they want.

**Room Reservation Management Page**



This is the logo of Swiss Garden Hotel. The user can click it to return to the menu page and the system will be refreshed automatically.

This button is to choose the room type that customer reserves. When user clicks on the arrow, every room type will be shown to choose.

These two buttons are to choose the date of room reserved by customer. When it is clicked, a calendar will be shown and user can choose the date by using it.

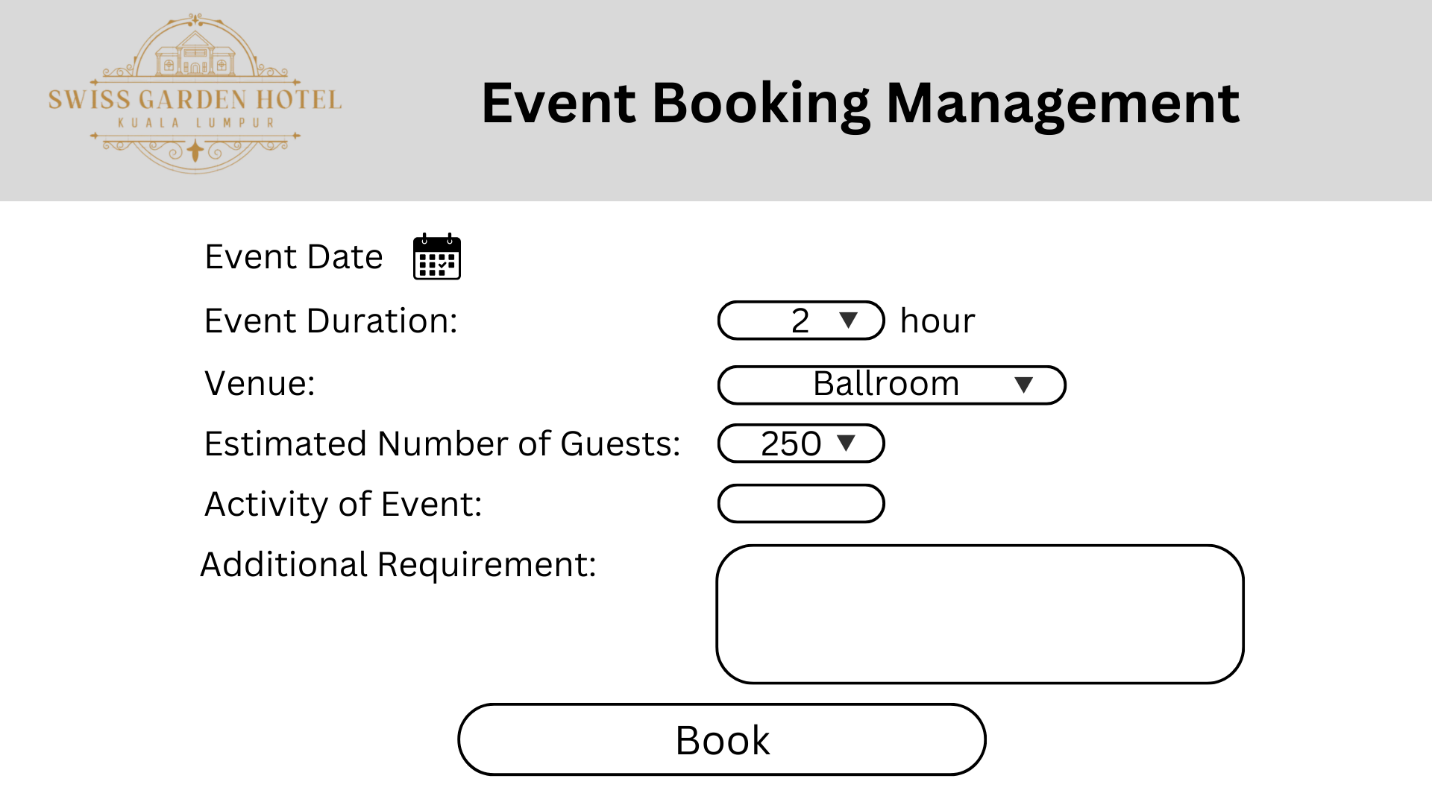
These three buttons are to input the number of guests and rooms. User can click the “-” or “+” button to enter the number.

These buttons are to input the estimated arrival time of guests at the hotel. When user clicks the buttons, every time with 10 minutes gap will be shown to choose. “am” and “pm” will be displayed too.

This area is to allow user to type the additional remarks such as the request.

This button is to submit all the information once it is complete.

**Event Booking Management Page**



This button is to input the date of event booking. When user clicks this button, a calendar will be shown and user can choose the date.

This button is to input the number of hours of event duration. When it is clicked, every number will be shown to choose.

This button is to choose the venue of event. When it is clicked, every available venue will be shown to choose.

This button is to choose the estimated number of guests. When it is clicked, every number with 10 gap will be displayed to choose.

This area is to input the activity of event. User can type the activity at here.

This area is to input the additional requirement. User can type the request at here.

This button is to book the event once all information is complete.

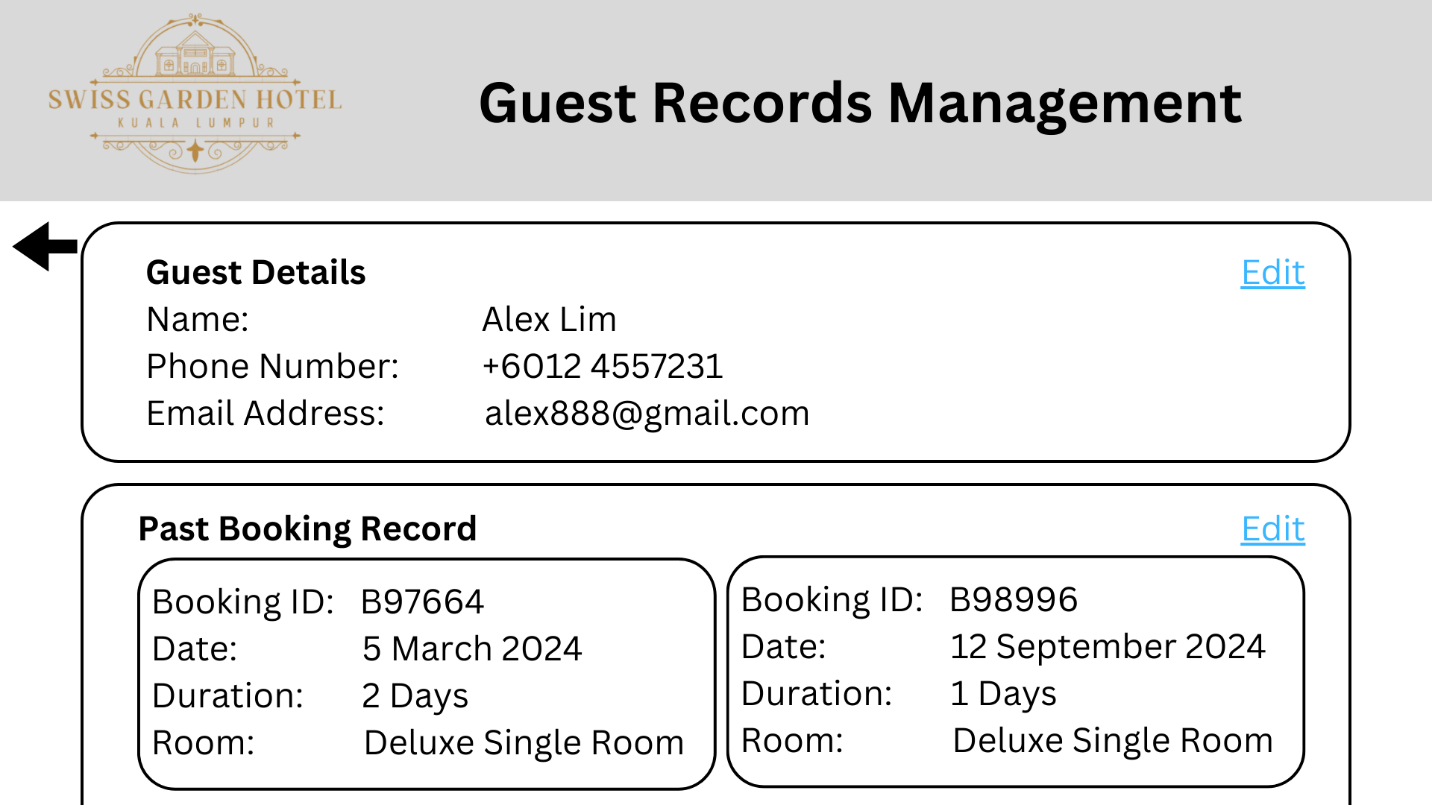
**Guest Records Management Page (1)**



At this area, detail personal information of guests will be shown. The information included name, phone number and email address.

This “View” button is to allow user to view the information which is more detailed about guest’s booking information. User can edit the personal information and booking information after clicking on the "View" button.

**Guest Records Management Page (2) displayed after clicking “View” button at previous page**



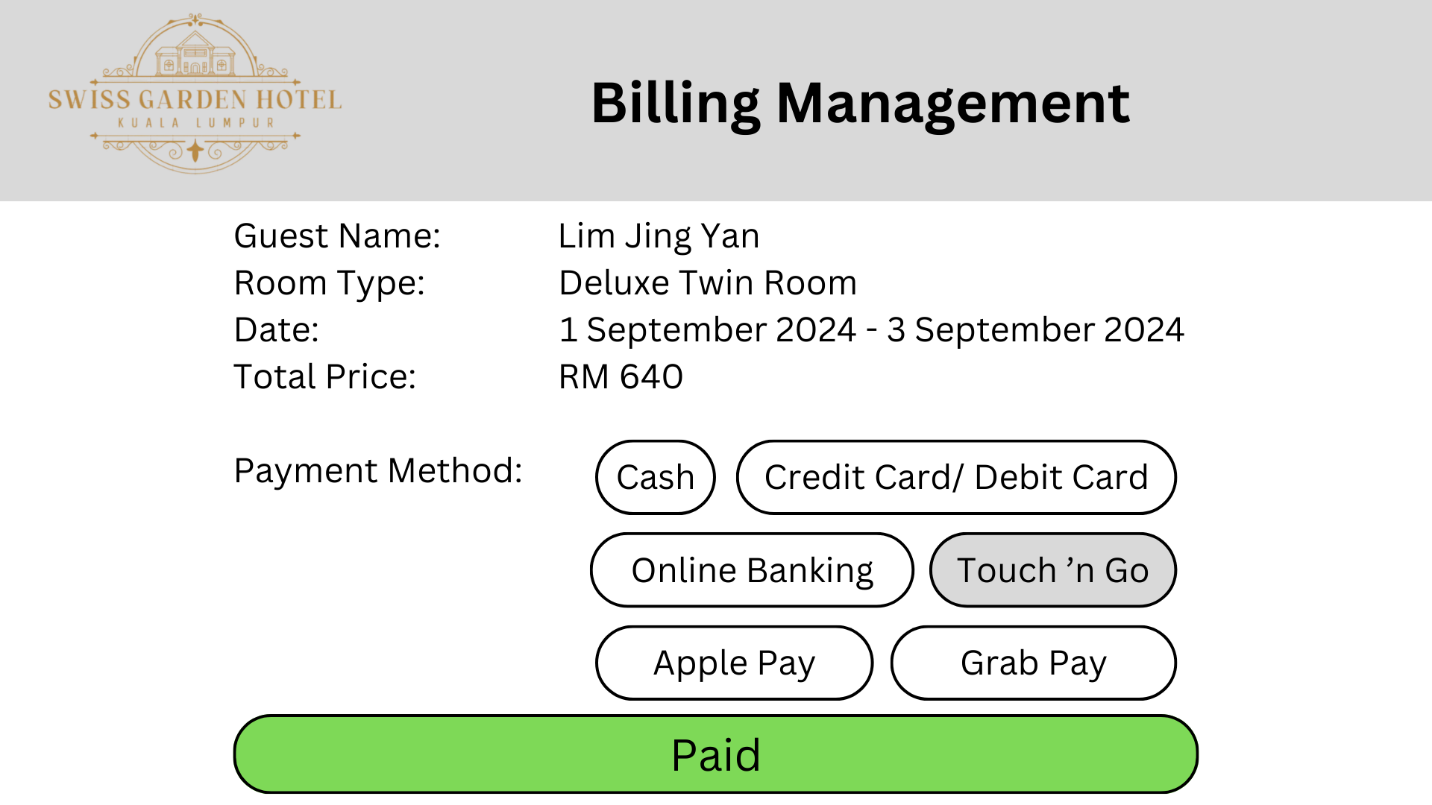
This button is to allow user to go back to the previous Guest Records Management Page(1).

This area is showing the personal information of guest.

This area is showing the past booking record information of guest. The information included booking ID, date, duration and room type.

This two buttons allow user to edit the guest’s personal information and past booking information.

**Billing Management Page**

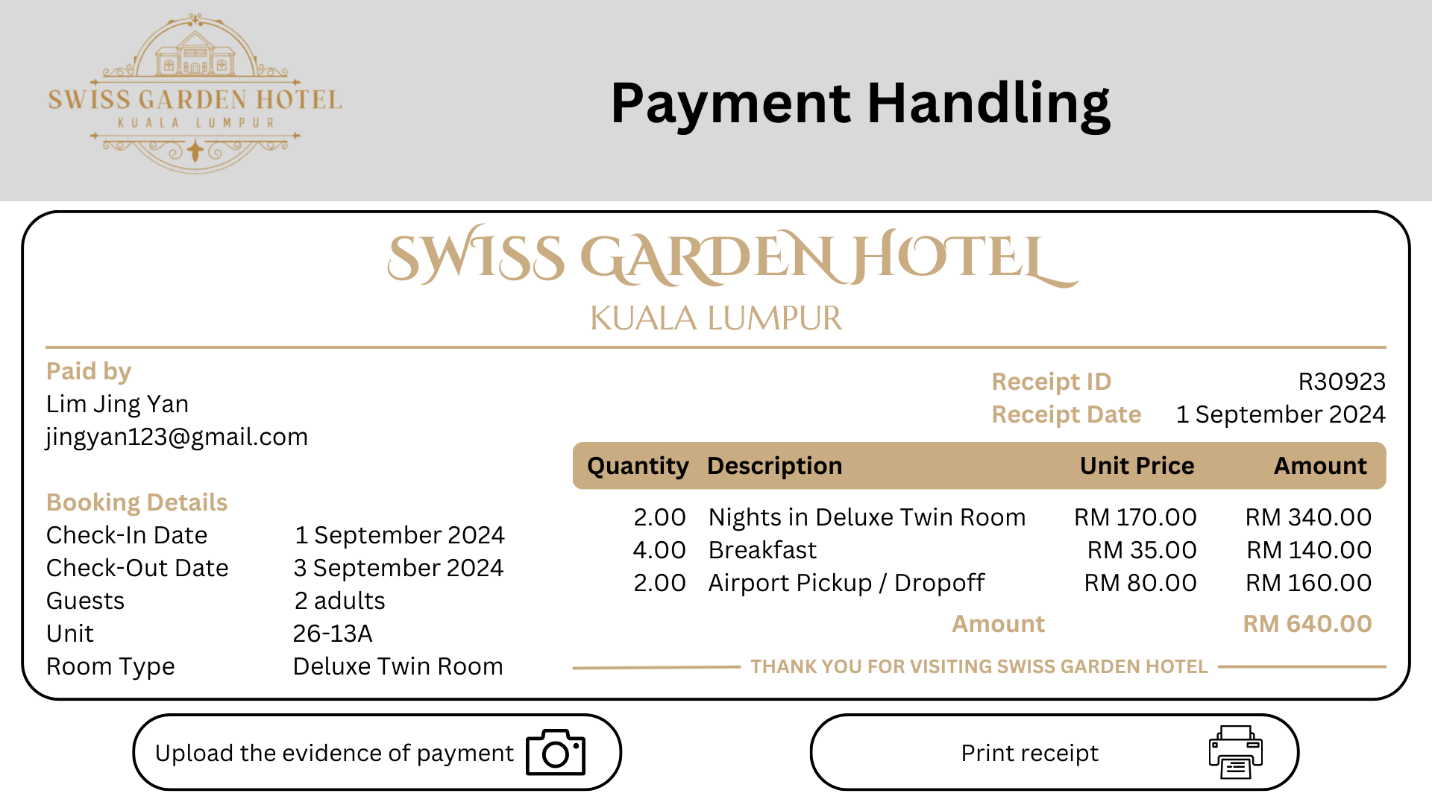


This area shows guest’s booking information and price of booking. The information included guest name, room type, date and total price.

This area allows user to select the payment method made by the guest.

This button is to confirm the information above once the information is complete.

**Payment Handling Page**

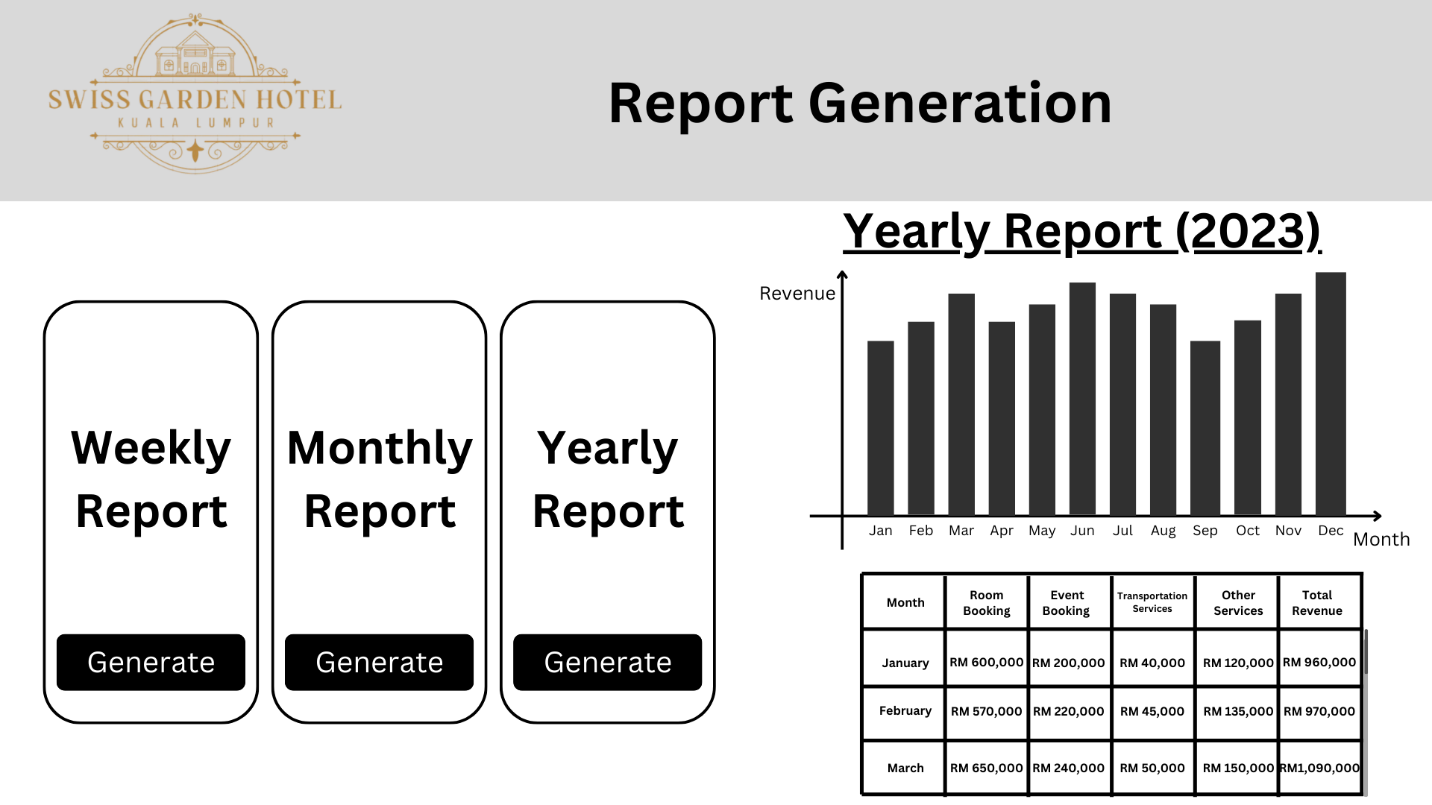


This area is showing the receipt of guest’s booking. The receipt includes complete information of guest’s personal information, booking information, receipt information and price information.

This button allows user to upload the evidence of payment from guest. Once user clicks this button, the camera will show up and capture the evidence.

This button is to print the physical receipt using the printer.

**Report Generation Page**

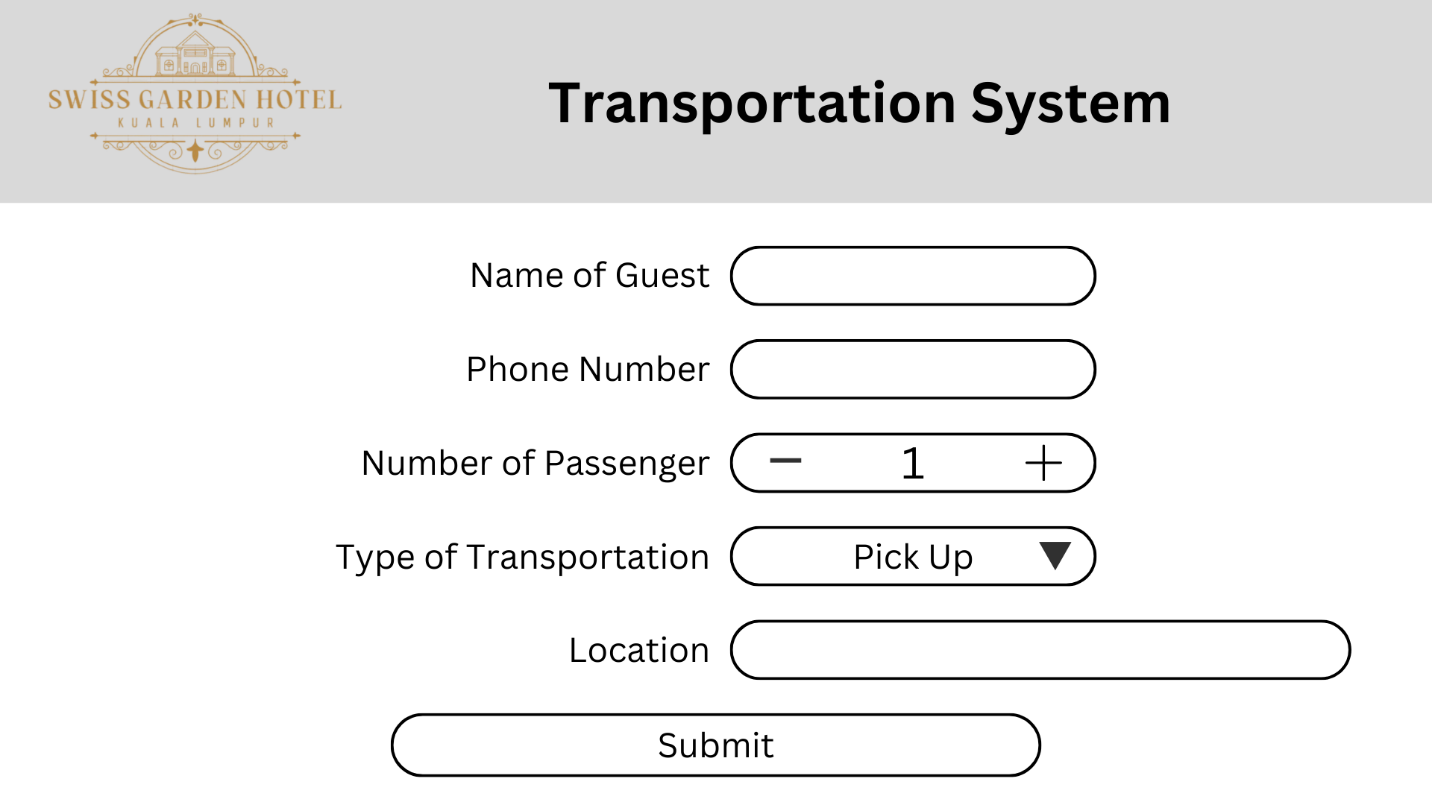


These three buttons allow user to generate weekly report, monthly report and yearly report. Once the button is clicked, the appointed report will be shown at the right.

This area shows the report appointed by user. The data is represented by a bar chart.

This area is showing the table that represents the amount of revenue earned from each part of service in each month.

**Transportation System Page**



This area is to input the name of guest. User can type the name in the area.

This area is to input the phone number of guest. User can type the phone number in the area.

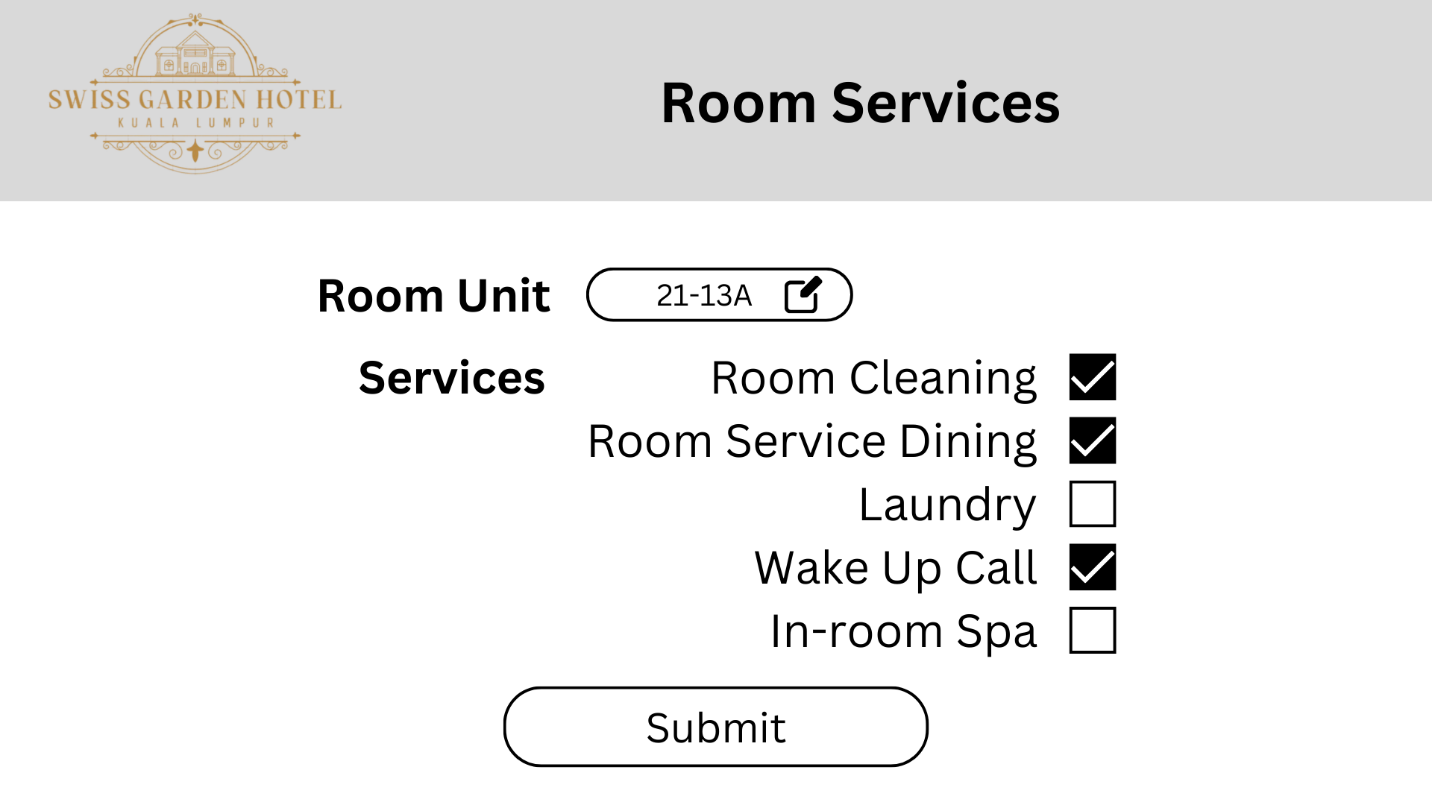
This area is to input the number of passenger. User can click the “+” or “-” button to select the number.

This area allows user to choose the type of transportation. When user clicks the arrow, “Pick Up” and “Drop Off” will be displayed to choose.

This area allows user to input the location of destination. User can type the location in the area.

This button is to submit the information of transportation once it is filled.

**Room Services Page**



This button is to choose the room unit that requests for room services. When user clicks this button, every room unit will be shown to choose.

This area shows the services that provided by Swiss Garden Hotel. User can click on the checkbox for the service that requested by the guest. One or more than one services can be chosen at one time.

This button is to submit the room service information once it is complete.

## 5.0 Implementation

### 5.1 Hardware Requirements

To guarantee effectiveness, reliability and scalability, the Swiss Garden hotel Management System’s hardware and software components must be carefully selected. A high-performance server is required to host the system and core database; cloud-based options like AWS or Microsoft Azure are recommended for scalability and remote access. Modern PCs with at least i5 processor, 8GB RAM and 256GB of SSD storage are essential for receptionists and employees. Secure LAN/Wi-Fi connectivity and a reliable network architecture are necessary to enable smooth device-to-device communication. Additionally, the system will require peripherals such as point-of-sale (POS) terminals to process payments, cloud storage or backup devices to save data and printers and scanners for administrative tasks.

### 5.2 Software Requirements

Stable operating systems like Windows Server or Linux for server and Windows 10 / 11 or MacOS for PCs will serve as the foundation for the system’s software. A database management system (DBMS), such as Microsoft SQL Server, PostgreSQL or MySQL will house visitors, transaction and reservation data. An application needs to feature modules for reservations, event bookings, billing and reporting whether it is off-the-shelf or custom-built. Connectivity to secure payment gateways like PayPal or Stripe will facilitate digital transaction. Modern web browsers are the sole way to access the system’s interface and antivirus software and encryption are crucial security measures to protect data and comply with regulatory requirements

Program installation and hardware configuration will occur after software design, development or change. Existing records will be transferred to the new system via data migration and thorough testing will guarantee that every module works as planned. To limit disruptions, the implementation will be done in phases and staff members will receive training on how to utilize the system effectively. System backups, frequent software updates and ongoing technical assistance will preserve dependability and foster the hotel’s future expansion. The Swiss Garden Hotel will be able to maximize operations and provide guests with an exceptional experience thanks to this mix of reliable software and technology.

## 6.0 Prototype

\*\*A video is submitted to give explanation of the prototype\*\*

## 7.0 Conclusion

### 7.1 Limitation

In the proposed system, one of the limitations is that our system is dependent on Internet connectivity. Our system is highly relying on cloud-based functionalities and the data such as guest information, room reservation record, room information, and event bookings are stored in the cloud-based database. Therefore, an Internet outage could disrupt the entire operation of the hotel management system. Besides, another limitation presents in the system is the maintenance and support challenges. The system should undergo maintenance, bug fix and technical support regularly to ensure the system is working properly and smoothly. This may require an ongoing expense to maintain the system.

### 7.2 Challenges

We have faced some challenges in our development process. One of them is the problem in the integration with existing processes. The hotel is currently using a manual workflow like the room availability checks, report generation and receipt generation. The transition from manual workflows to a digital system is a huge problem for the staff who are not familiar with technology and find it hard to adapt to the new system. Other than that, another challenge we have faced is the data migration problem. Until now, the hotel has accumulated quite a number of guest records and bookings records. The data migration from the old workflows to the new system is significantly important. Mistakes in the migration process are not allowed because it will leads to data inconsistencies or data loss in the system.

### 7.3 Further Enhancement

Our suggestion to enhance the system is to add an auto-backup functionality. With this function, if there is any interruption of Internet connection, the data will be backed-up and stored in an offline database that allows access without Internet connection.

# Individual Component

## 8.0 Logical Design (Part 2)

### 8.1 Anderson Tey Le Tien TP076394

**Request Room Service**

|  |  |
| --- | --- |
| Guest | Staff |
| Yes  No  Confirm ordering?  Select Service | Provide Services  Receive order |

The **request room service** activity diagram shows the process need for user to ordering a room service. First, guests need to select the service they need such as room cleaning, food ordering, or requesting for extra sheets. Guest needs to confirm their order after they make their choice, they are allowed to select again if not confirm for the order. Staff will only receive their requirement and prepare for the order when customer confirm their order.

**Manage Guest Activities**

|  |  |
| --- | --- |
| System | Staff |
| Display requirements | Fulfil requirement  No  Yes  Arrange activity  Requirement fulfilled?  Check Customer Order |

The **manage guest activities** activity diagram displays the flow of system delivering guests requirement to staff. Customers’ requirements such as room service, room reservation, and event booking will be shown by the system to allow staff to manage their order effectively. Staff need to check whether the requirement is done and need to assign the order to worker in charge if the requirement have not been fulfilled. Then, staff will carry out their job immediately after being assigned.

### 8.2 Chek Ka Nic TP074894

**Login Account**

A diagram of a computer

Description automatically generated

The **Login Account** activity diagram outlines the steps a user follows to log into the system. The process starts with the user accessing the login page and entering their credentials (username and password). The system then validates the credentials. If the credentials are valid, the user is granted access to the homepage, and the login status is displayed. If the credentials are invalid, an error message prompts the user to re-enter their credentials, ensuring the system remains secure and accessible only to authorized users.

**Taxi Booking**

A diagram of a business process

Description automatically generated

The **Taxi Booking** activity diagram showcases the process a guest booking a taxi. The guest initiates the booking by specifying booking details like the pick-up and drop-off locations, passenger count, and departure time. The staff verifies taxi availability. If a taxi is available, the staff confirms the booking, and the guest receives a booking confirmation. If no taxi is available, the staff notifies the guest and suggests alternate options, ensuring efficient communication and service delivery.

### 8.3 Ching Ke Jing TP074822

**Event Booking**

A diagram of a function

Description automatically generated

This **Event Booking** activity diagram shows how the guest and the system interact during the venue reservation process. By selecting a day, a venue, the approximate number of attendees and the duration of the event, the guest starts the process. The availability of the venue is then confirmed by the system. After the system verifies the reservation and checks to see if the place is open, the guest pays to finalize the transaction. If the initial location isn’t available, the guest must return and select another and they must continue doing this until they find one. The flow is finished once the payment has been processed successfully.

**Room Reservation**

A diagram of a flowchart

Description automatically generated

This **Room Reservation** activity diagram clarifies the hotel reservation procedure and illustrates how the guest engages with the system. Important details including the kind of room, the dates of check-in and check-out, the number of passengers and the expected arrival time are first provided by the guest. The system examines this information to determine whether rooms are available. If there are rooms available, the system allows the guest to complete the booking process by confirming the reservation and making payment. If there are no rooms available, the system will ask the guest to change their selection before rechecking the availability. This cyclical strategy ensures both guest pleasure and procedural continuity by allowing the guest to review all their options prior to making a reservation.

### 8.4 Chong Wei Jie TP076546

**Generate Total Bill**

**A diagram of a process

Description automatically generated**

In **Generating Total Bill** process, guest will start to proceed to pay bill. Then, staff will help to check guest personal information, booking information and price of booking. After that, staff will input the information and price into the system. Then, the receipt will be generated and staff can print the receipt to give it to guest. Staff will need to confirm the payment method that guest wants to perform. If the guest wants to pay the bill cashlessly, he can pay with a bank card or online payment and staff will capture the evidence of payment. If the guest pays with cash, then staff will receive the cash correctly. Last, staff will receive the review and comment from the guest for improvement. End.

**A diagram of a diagram

Description automatically generatedGenerate Report**

In the beginning of **Generate Report** process, the staff will request the system to generate annual report. The staff can choose to generate weekly, monthly or yearly report. Then, the system will check the data stored in the database. If the data is incomplete, the system will require staff to input additional data then generate bar chart and revenue table. If the data is complete, bar chart and revenue table will be generated automatically. End.

### 8.5 Mah Zheng Yang TP076666

**Make Payment**

A diagram of a process

Description automatically generated

The **Make Payment** activity diagram shows the steps for the users to follow to make payments. The process starts with the system displaying the payment page and prompts the users to enter guest details and choose a payment method. The system will then process the payment. If the payment is successful, the system will generate a receipt and the users will be able to view the receipt. If the payment is unsuccessful, the system will display an error message to the user and prompt the user to enter guest details again and choose another payment method.

**Leave Feedback**

A diagram of a process

Description automatically generated

The **Leave Feedback** activity diagram outlines the steps for the users to leave feedback. The process starts with displaying the feedback page and prompts the users to select a service to review. Users will be asked to provide ratings and comments on the specific service and submit the feedback. If the feedback is submitted successfully, the feedback will be stored in the database. If the feedback is not submitted successfully, the system will display an error message to the user and prompt the user to submit the feedback again.

## 9.0 Methodology

### 9.1 Prototyping Model (Anderson Tey Le Tien TP076394)

#### 9.1.1 Introduction

A Prototyping model is a system development model which suitable for project that users are not confirm with their requirement. During the prototyping model, process such as building, testing, and modifying a prototype will be repeated until finish the project. The initial version of software will be used to present their concepts and design options. Therefore, users can add more requirements after the demonstration to improve the system. They can also know the weakness and error exists in previous requirement.

#### 9.1.2 Implementation

In this project, prototyping model had been distributed into 4 parts including prototyping plan, outline definition, executable prototyping, and evaluation report. In the first step, we are required to determine the objective and timeline for prototype development. The purpose of prototype needs to be planned according to user’s requirement.

The following step is to define the functionality of prototype by arranging the workflow and features. We need to gather the requirement from stakeholders and define whether the requirement is functional or non-functional. After that, we will process both functional requirement into document and create a flow diagram for the system.

In the executable prototype process, a prototype which focus on main function will be created. It means that some important features need to be implemented in this step. Not only that, developers are also required to create a functional user interface for end-user.

The last step in prototyping model is gathering the feedback from users and compile it into document. Several hotel staff and guests will take part in the testing session, and their feedback on the functionality and interface of system will be recorded. Therefore, improvement will be done according to the documentation of feedback.

#### 9.1.3 Advantages and Disadvantages

There are several advantages when using prototyping model. When developers keep getting feedback from users, unnecessary functions will be removed and users can make sure the development of system are on the right way. Therefore, the efficiency of system implementation will increase and system with a high-quality design can be developed with less development effort.

However, there are some disadvantages with prototyping model. First, even the same prototype being presented, different feedback will be given from different users. It will confuse the developers and hard to fulfil every requirement. Not only that, the prototype shown to users might not be same with final system because it keeps changing when new problem faced. Moreover, the tester of prototype is not the typical system users, and they may not understand what customers really want.

#### 9.1.4 Justification

### The reason I selected the Prototyping model as the methodology for the Swiss Garden Hotel management system is that the prototyping model ensures the efficiency of the system's interfaces and functionalities. It allows smoother operations in various features such as room booking, billing management, and report generation.

### 9.2 Waterfall Model (Chek Ka Nic TP074894)

#### 9.2.1 Introduction

The Waterfall Model is a linear and sequential software development methodology in which each phase—requirements analysis, system design, implementation, testing, deployment, and maintenance—is completed before the next begins. Developed by Winston W. Royce in 1970, it follows a structured approach where progress flows downward like a waterfall (Royce, 1970). This model is widely used in projects with well-defined requirements and minimal expected changes.

#### 9.2.2 Implementation

**1. Requirements Analysis Phase**

In this phase, the system requirements are gathered by analysing the use case diagram. Each actor's interactions with the system, such as guests making reservations or staff managing bookings, are examined to ensure all functionalities (e.g., login, feedback, taxi booking) are clearly defined. This phase sets the foundation for understanding what the system must achieve.

**2. System Design Phase**

The use cases guide the creation of the system's architecture and components. Modules like Room Reservation, Event Booking, and Billing Management are designed based on the interactions and workflows from the use case diagram. Relationships between actors and modules, such as staff confirming bookings or generating reports, are mapped out to define the system's structure.

**3. Implementation Phase**

The design is translated into code, and the system is developed module by module. Features such as user authentication for login, workflows for taxi booking, and interfaces for feedback submission are implemented. Each use case is directly linked to a functional software component.

**4. Testing Phase**

Each use case is tested to ensure it performs as expected, including scenarios like valid/invalid credentials for login or unavailable rooms for reservations. Integration testing ensures modules work seamlessly together, while user acceptance testing verifies that both guests and staff can effectively use the system.

**5. Deployment Phase**

The system is deployed in the hotel environment, enabling guests to access features like reservations and feedback through the website or mobile app. Staff are trained to use internal functionalities such as managing guest activities and generating reports. The system becomes operational.

**6. Maintenance Phase**

After deployment, the system is monitored for performance and updated as needed. Bugs are fixed, and improvements are made to enhance functionalities like generating reports or handling payments. New features may also be added based on user feedback and evolving hotel needs.

#### 9.2.3 Advantages and Disadvantages

**Advantages**

The advantages of the Waterfall Model lie in its simplicity and clarity. Its structured approach makes it easy to manage and ideal for projects with clear goals. Each phase is well-documented, enabling easy knowledge transfer and accountability (Balaji & Murugaiyan, 2012).

**Disadvantages**

However, the drawbacks include limited flexibility, as it is difficult to accommodate changes once a phase is completed. It is also less suitable for projects with evolving requirements or for iterative processes, which may lead to costly revisions if changes arise late in development (Pressman, 2020). Despite its limitations, the Waterfall Model remains effective for projects requiring high levels of predictability and documentation.

#### 9.2.4 Justification

Developing a hotel management system typically involves well-defined and predictable requirements (e.g., room reservations, taxi bookings, payment processing). The system's functionalities are unlikely to change significantly during development. The Waterfall model is ideal in such cases because it requires a detailed understanding of requirements upfront before moving to subsequent phases. Moreover, the Waterfall model emphasizes a linear, phase-based approach (Requirement Analysis → Design → Implementation → Testing → Deployment → Maintenance). This suits the development of a hotel management system where tasks like designing room booking, taxi booking, or payment workflows must follow a logical and sequential order to ensure system integrity (Balaji & Murugaiyan, 2012). In the Waterfall model, each phase is thoroughly documented, which is essential for a system that integrates multiple functions (e.g., guest services, staff management, billing). Comprehensive documentation ensures that all stakeholders (developers, staff, and management) understand the system's design and functionality, aiding future maintenance and updates.

### 9.3 SSADM (Ching Ke Jing TP074822)

#### 9.3.1 Introduction

The Structured System Analysis and Design Methodology (SSADM), a well-liked paradigm for system development, employs rigorous and systematic techniques for software system analysis, design and implementation. By aligning its fundamental components with the four main phases of software production which are software specification, software design and implementation, software validation and software evolution. SSADM ensures a methodical and efficient approach to system development.

The Software Specifications in SSADM is preceded by a feasibility study, which assesses the project’s viability based on cost, time and technical resources. By identifying the project’s objectives, constraints and scope, this step establishes the framework for the work. After that, a comprehensive needs analysis is conducted. SSADM uses tools like entity relationship diagram (ERD) and data flow diagrams (DFD) to assess current systems and gather requirements. These systematic approaches ensure that the requirements for the software are accurate, comprehensive and consistent with those of the stakeholders.

#### 9.3.2 Implementation

The SSADM in the Software Design and Implementation phase consists of both the logical and physical design phases. Logical design’s primary objective is to create an abstract model of the system. This means developing data structures and system functionalities while ensuring that all requirements from the specification stage are satisfied. The use of thorough DFDs and ERDs enables a design that is both accurate and understandable. After the logical design is finished, the physical design phase begins. In this instance, the abstract model is converted into a concrete implementation strategy that compromises technical requirements for both software and hardware, database schema and user interface designs. By serving as a roadmap for system development, these processes ensure that implementation closely adheres to the planned design.

Throughout the software validation process, SSADM emphasizes how important it is to ensure that the developed system meets the criteria. The initial validation step, prototyping, entails stakeholders building and assessing models of essential elements. This continuous feedback system will ensure that users’ needs are being met by the technology. Comprehensive testing follows which comprises functional testing to verify requirements compliance and user acceptability testing (UAT) to guarantee usability and efficacy. These validation processes ensure the reliability and quality of the system before it is deployed.

Finally, the Software Evolution phase is supported by SSADM's emphasis on structured analysis and documentation. Since systems will eventually need to be updated and modified, SSADM provides a systematic approach to change management. Thorough documentation of the concept, implementation and standards enables east maintenance and modification. Without compromising its integrity, the process makes it easier to reassess and adapt the system to changing user needs. Because of this systematic approach, the product will continue to be successful and relevant for the rest of its existence.

#### 9.3.3 Advantages and Disadvantages

**Advantages**

* **Structured Approach**: SSADM provides a clear, systematic framework that guides the entire system development process. Each stage is well-defined, ensuring that no crucial step is overlooked.
* **Thorough Documentation**: The methodology emphasizes detailed documentation at every stage, which aids in clear communication between stakeholders and developers. It also facilitates future system maintenance and upgrades.
* **Clear Requirements Analysis**: SSADM places significant emphasis on understanding user requirements before designing the system. This ensures that the final product meets the actual needs of users and stakeholders.
* **Risk Reduction**: By thoroughly analyzing feasibility, requirements, and designs early in the process, SSADM helps to identify potential risks and issues before they affect implementation, thus reducing the likelihood of project failure.
* **Standardized Methodology**: SSADM is a standardized methodology widely recognized in the industry, providing a common language for development teams and stakeholders, especially in large projects.

**Disadvantages**

* **Time-Consuming**: The detailed documentation and analysis required at each stage of SSADM can make the process slow, especially for smaller projects where such extensive planning may not be necessary.
* **Rigidity**: SSADM’s structured approach may not be flexible enough for rapidly changing requirements or environments. Adjusting the system based on new insights can be challenging once the design phase is completed.
* **High Resource Demand**: Due to its emphasis on documentation, analysis, and planning, SSADM may require more resources, including time, personnel, and expertise, making it less suitable for smaller teams or projects with limited budgets.
* **Complexity in Large Projects**: In large projects, SSADM can become overly complex, as the extensive documentation may lead to difficulty in tracking changes and maintaining consistency across different stages of the system development.
* **Lack of Emphasis on Prototyping**: SSADM tends to focus more on comprehensive documentation and planning, potentially neglecting rapid prototyping or iterative feedback loops, which can delay the identification of usability issues or design flaws.

#### 9.3.4 Justification

The Swiss Garden Hotel Management System case study provides strong support for SSADM's structured approach, which is appropriate given the scale and complexity of the hotel's operations. The methodology's well-defined basis will allow for a full examination and documentation of every facet of the hotel administration system, including event and room reservations, billing, and report preparation. This is crucial because it ensures that the system satisfies the requirements of each stakeholder, including management, staff, and visitors, by determining their specific needs. The focus on requirements analysis ensures that the final system will meet the operational needs of the hotel and streamline processes such as reservation and payment processing, reducing errors and inefficiencies in the present manual technique.

### 9.4 Spiral Model (Chong Wei Jie TP076546)

#### 9.4.1 Introduction

Spiral Model is a risk-driven software process framework that combines design and prototyping. It is a meta-model and can be used by other models. Spiral Model was proposed by Boehm in 1988. Spiral Model contains many loops in the process and each of the loops represents each phase. In this model, the development team begins with a small set of requirements. Then, the team will be going through every phase. (Alshamrani & Bahattab, 2015)

#### 9.4.2 Implementation

There are four phases in Spiral Model, Planning, Risk Analysis and Mitigation, Development and Review and Evaluation.

**Planning Phase**

In the planning phase, the development team will communicate with customer to make sure the team fully understand the requirements. The development team will record the information of requirements and design a development plan. The distribution of work, budgetary plan and time schedule will be planned.

**Risk Analysis and Mitigation Phase**

In the second phase, the development team will discuss about the potential risk of the project. For example, the risk can be failure of transfer payment data to the receipt of bill. This may cause the price to be calculated wrongly and the amount of money receive by customer is incorrect. Solutions to the risk will be produced too. Then, a prototype is created. The prototype is a basic model and it helps the customer to give approval or disapproval to the development team.

**Development Phase**

In Development phase, the actual software is created. The software engineers will improve the software according to the comments of customer. Hence, the software can further improve and meet customer satisfaction. For example, the system has included every function required by the customer. The function are: Room Reservation Management, Event Booking Management, Guest Records Management, Billing Management, Payment Handling and Report Generation.

**Review and Evaluation Phase**

A diagram of a product

Description automatically generatedIn the final phase of Spiral Model, the development team will test the final software. Then, the customer will review the software. If the software has successfully met all the requirements and the customer is satisfied with the software, the development process is complete. (Risener, 2022)

Figure 1: Spiral Model

#### 9.4.3 Advantages and Disadvantages

**Advantages**

In fact, Spiral Model has several advantages. First, the most important advantage is that it can analyze large amount of risk, even if the risk is enhanced. Besides, it is also a suitable software development model for large and mission-critical projects. Furthermore, the software is produced early in the life cycle.

**Disadvantages**

However, Spiral Model also carries some disadvantages. First, Spiral Model is not a good software development model for small projects. Apart from that, the success of the project is dependent on the risk analysis phase. Lastly, Spiral Model may require a high cost. (Nabil & Govardhan, 2010)

#### 9.4.4 Justification

Spiral Model is a very suitable system development model for Swiss Garden Hotel's staff system. This is because the system is a large and complex structure while Spiral Model is truly efficient in managing complex system. Besides, the Spiral Model is high risk tolerance for critical projects. This means that Spiral Model can handle some failures of the project.

### 9.5 V-Model (Mah Zheng Yang TP076666)

#### 9.5.1 Introduction

V-Model is known as the Verification and Validation Model. It is a structured software development approach emphasizing detailed and early testing throughout the Software Development Life Cycle (SDLC) (Pham, 2024). The V-Model is named after its shape, the letter “V”. The left side of the “V” represents the verification phase while the right side of the “V” represents the validation phase (Oppermann, 2023).

A diagram of software testing

Description automatically generated

Figure 2: V-Model

#### 9.5.2 Implementation

**Verification Phases**

**Requirement Design**

In this phase, the hotel’s needs are identified and the system requirements are gathered using a use case diagram. The primary requirements include logging in to the account, reserving a room, booking an event, booking a taxi, requesting room service, making payment, leaving feedback, managing guest activities, generating bills, and generating reports. These requirements and needs will be documented in a Requirement Specification Document (SRS) as a foundation of subsequent phases.

**System Design**

The functionalities of the system are planned, and a system architecture is created. The modules are designed based on the use case diagram including the interactions and relationships between actors and modules. The output of this phase is a System Design Document which provides an overview of the system’s function flow.

**Architecture Design**

In this phase, the system is divided into specific modules that have a particular set of functionalities. For instance, the guest will handle room reservations and payments while the staff will be able to generate report and bills. A High-Level Design (HLD) Document will be created to outline how the modules will interact with each other.

**Module Design**

During this phase, internal workings like input processes and output processes are defined. For example, check-in and check-out dates, guest details, and payment receipts. This phase produced a Low-Level Design (LLD) Document which includes internal logic and data flow between the modules.

**Coding**

In the coding phase, design documents are transformed into a working system. All the use cases such as room reservations, event booking and taxi booking are implemented into the system and become functional. The output of this phase is a fully implemented code.

**Validation Phases**

**Unit Testing**

In this phase, every module is tested to ensure they function properly. For example, the staff report generation function is tested to confirm that it generates an accurate and true report. Unit testing ensures that each module functions as expected.

**Integration Testing**

After unit testing, integration testing focuses on testing the interactions between modules. For instance, after the guests make their payment, integration testing makes sure the receipt generation module works properly and generates a correct receipt.

**System Testing**

In system testing, it evaluates the entire system’s performance in an end-to-end scenario. For instance, the system is tested to ensure that guests can log in to their account, reserve a room, and make payment while the staff can generate receipts and manage bookings. Other than that, the staff is able to generate reports. This phase ensures that the system is working properly.

**Acceptance Testing**

In acceptance testing, the system is tested by the clients to ensure that the system fulfills the requirements and functionalities. After the system is validated and verified by the client, the system is ready for deployment. Acceptance testing ensures that the system is validated and aligns with the client's expectations.

#### 9.5.3 Advantages and Disadvantages

**Advantages**

* Clear and structured approach - Allows the developers to understand and follow more easily
* Early detection of errors - Significantly helps in preventing unnecessary costs of fixing the system and ensures software quality.

**Disadvantages**

* Rigid and inflexible structure - Difficult to accommodate changes after the development process starts
* Late testing phase - Having the testing phase after the code is completed and if any issues arise, the project will be delayed

#### 9.5.4 Justification

Using the V-model, the development process of the system will be smoother and more efficient. It is because it has a clear structure that allows developers to follow and develop the software step by step. Besides, developers will be able to detect errors at the beginning of the development process. This helps the developers to avoid rework of the project and prevent the waste of time.

## 10.0 Testing

### 10.1 White Box Testing (Anderson Tey Le Tien TP076394)

#### 10.1.1 Testing Method

The testing method I choose is white box testing, which also defined as structural testing. White box testing usually carried out by system developers as it requires tester to know the internal structure of the process. This testing method is more efficient in debugging because every possible condition will be tested before implementation of next function. However, white box testing takes long time to be conducted as every single process will being tested. It can help to reduce unnecessary code which might cause trouble. Thus, developers must have enough knowledge in system implementation to carry out a white box testing. Not only that, specialised tools like source code analysers and debuggers are required to test the system (Bele,2022).

#### 10.1.2 Test Script

\*\*Test script are submitted separately in an excel file\*\*

### 10.2 Black Box Testing (Chek Ka Nic TP074894)

#### 10.2.1 Testing Method

In this given task, I have chosen black box testing as my approach to conduct the testing on the prototype. To be more specific, black box testing is a software testing approach that evaluates the functionality of an application without any knowledge of its internal structure, code, or implementation. Testers focus solely on providing inputs and verifying outputs against expected results, ensuring the system behaves as intended from an end-user perspective. This method emphasizes testing the software's external behaviour and is commonly used to validate functionality, usability, and reliability (Pressman, 2020).

#### 10.2.2 Test Script

\*\*Test script are submitted separately in an excel file\*\*

### 10.3 Black Box Testing (Ching Ke Jing TP074822)

#### 10.3.1 Testing Method

The Swiss Garden Hotel Management System case study provides strong support for SSADM's structured approach, which is appropriate given the scale and complexity of the hotel's operations. The methodology's well-defined basis will allow for a full examination and documentation of every facet of the hotel administration system, including event and room reservations, billing, and report preparation. This is crucial because it ensures that the system satisfies the requirements of each stakeholder, including management, staff, and visitors, by determining their specific needs. The focus on requirements analysis ensures that the final system will meet the operational needs of the hotel and streamline processes such as reservation and payment processing, reducing errors and inefficiencies in the present manual technique.

#### 10.3.2 Test Script

\*\*Test script are submitted separately in an excel file\*\*

### 10.4 White Box Testing (Chong Wei Jie TP076546)

#### 10.4.1 Testing Method

I chose white box testing as my testing method. It is also known as Clear Box Testing and Open Box Testing. White box testing generally relies on the internal structure of the software. A simple white box testing method needs single elements to execute. By using White Box Testing method, the internal structure of the item being tested is known to the tester. In fact, White Box Testing method includes statement and branch coverage which are very common. Besides, White Box Testing is very effective to generate the small number of additional tests that needed to satisfy a criterion. (Ostrand, 2002)

#### 10.4.2 Test Script

\*\*Test script are submitted separately in an excel file\*\*

### 10.5 Black Box Testing (Mah Zheng Yang TP076666)

#### 10.5.1 Testing Method

In this project, I have chosen black box testing as my testing method while testing the prototype. Black box testing is defined as a software testing method in which the tester tests the functionality of the software without a thorough knowledge of its internal structure (Ashtari, 2022). Black box testing gives priority to the comprehensive examination of the software functionalities and it is closely related to behavioral testing. Furthermore, there are few types of Black Box Testing such as functional testing, non-functional testing, boundary value analysis, equivalence partitioning, and error guessing. While using Black Box Testing, it helps to identify missing functionalities but once a defect is found, it is hard to determine where the problem originates.

#### 10.5.2 Test Script

\*\*Test script are submitted separately in an excel file\*\*

# References

Alshamrani & Bahattab. (2015). A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model. IJCSI International Journal of Computer Science Issues, 106-111. Retrieved from https://d1wqtxts1xzle7.cloudfront.net/36637147/SDLC-libre.pdf?1423940228=&response-content-disposition=inline%3B+filename%3DA\_Comparison\_Between\_Three\_SDLC\_Models\_W.pdf&Expires=1733772954&Signature=KYRGLCNcVBdkGKvrm99rHLMmZYDX90qUnZc9CaZV8WnCi8H018xVSQro0

Ashtari. (2022). *Black Box vs. White Box Testing: Understanding 3 Key Differences*. Retrieved from Spiceworks: https://www.spiceworks.com/tech/devops/articles/black-box-vs-white-box-testing/

Balaji & Murugaiyan, B. (29 June, 2012). WATERFALLVs V-MODEL Vs AGILE: A COMPARATIVE STUDY ON SDLC. International Journal of Information Technology and Business Management, 5. Retrieved from https://mediaweb.saintleo.edu/Courses/COM430/M2Readings/WATEERFALLVs%20V-MODEL%20Vs%20AGILE%20A%20COMPARATIVE%20STUDY%20ON%20SDLC.pdf

Nabil & Govardhan. (2010). A Comparison Between Five Models Of Software Engineering. IJCSI PUBLICATION 2010, 94-101. Retrieved from https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=3a4a2cb2328e2f416be0be012e5d580975943554#page=115

Oppermann. (2023). *What Is the V-Model in Software Development?* Retrieved from Built In: https://builtin.com/software-engineering-perspectives/v-model

Ostrand, T. (2002). White-Box Testing. Wiley Online Library. doi:https://doi.org/10.1002/0471028959.sof378

Pham. (2024). *V-Model (Software Development): A Comprehensive Handbook*. Retrieved from Orient Software: https://www.orientsoftware.com/blog/v-model-software-development/

Pressman, R. S. (2020). Software Engineering: A Practitioner’s Approach. 83. Retrieved from https://nlp.chonbuk.ac.kr/SE/ch03-04.pdf

Risener. (2022). A Study of Software Development Methodologies. ScholarWorks@UARK, 1-39. Retrieved from https://scholarworks.uark.edu/cgi/viewcontent.cgi?article=1105&context=csceuht

W.Royce, W. (1970). Managing the Development of Large Software Systems. 11. Retrieved from https://blog.jbrains.ca/assets/articles/royce1970.pdf