**ADDIS ABABA SCIENCE AND TECHNOLOGY UNIVERSITY  
  
College of Natural and Social Science  
  
  
Introduction to Software Engineering (GLaf/1002)**

**SECTION A**

**Group 1**

**Name ID NUMBER**

Abel Mekonnen ETS 0038/16

Abel Birhane ETS 0034/16

Abenezer Shiferaw ETS 0067/16

Addis Shiferaw ETS 0099/16

Abraham Aragie ETS 0089/16

Abraham Teramed ETS 0094/16

Addisalem Hailay ETS 0100/16

Submitted to: **Dr. Kassahun Ademek (PhD)**

Submission Date: **January 6, 2025 GC**

**Table of Contents**

Introduction----------------------------------------------------------------------------------------------3

The Planning Phase-------------------------------------------------------------------------------------4

Analysis Phase-------------------------------------------------------------------------------------------5

Design Phase---------------------------------------------------------------------------------------------5

Developmental Phase-----------------------------------------------------------------------------------6

Testing Phase--------------------------------------------------------------------------------------------7

Deployment Phase--------------------------------------------------------------------------------------9

Maintenance--------------------------------------------------------------------------------------------11

Sample Code ------------------------------------------------------------------------------------------14

Conclusion--------------------------------------------------------------------------------------------- 15

Sample Flowchart ------------------------------------------------------------------------------------ 16

**Introduction**

The **Utopia Hotel Reservation System** is a c++ programming language written program designed to help with the management of hotel room bookings, customer details, and room availability. It also provides a comprehensive and user-friendly interface for hotel staff and all the guests, allowing them to handle reservations, track room occupancy, and retrieve relevant information quickly, easily and efficiently.

In today's fast-paced world, managing hotel reservations manually can lead to errors, inefficiencies, and loss of valuable time making it quite tedious, and old fashion. The Utopia Hotel Reservation System tried to address all these challenges by automating the booking process, providing real-time information on room availability, and offering seamless ways to manage all the guest details.

This documentation provides the outlines to the design, functionality, and flow of the Utopia Hotel Reservation System, offering detailed explanations of its components, system architecture, and also how it serves both the hotel management team and customers. By using this system, hotels can enhance their operational efficiency, reduce errors in room booking, and improve the guest experience significantly

**Software Development Life Cycle (SDLC) Documentation for Utopia**

**Hotel Reservation System**

1. **The Planning Phase**

**Objective** We developed a system that helps people with the hotel room reservations, allowing users to book rooms, check availability, view room details, search users by ID, and manage user lists. The goal is to provide an efficient solution for managing hotel room bookings and user information to provide reduce the workload of the people working there.

**Scope**

* The system will support up to 100 rooms with 5 room types: Single, Double, Family, Deluxe, and Suite all to the people's personal preference.
* Users can view room availability, book rooms, check room details, search for users by ID, and leave rooms. We will ask for all this information to avoid any confusion when booking other rooms.
* Information for each room (e.g., price, type, availability) will be stored and displayed dynamically based on the current state of the availability of the rooms.

**Functional Requirements**

* **Room Booking:** Ability to book available rooms, register user details (name, ID, phone number).
* **Room Availability Check:** Users can view the available and the booked ones as well.
* **Room Details:** Users can see room details, including type, price, and availability.
* **Search by User ID:** Ability to search for users by their ID.
* **User List:** View a list of users who have reserved rooms.
* **Room Checkout:** Users can check out and free the room if it is not to their liking.

**Non-Functional** **Requirements**

1. **User Interface:** The system should be a console-based interface that is easy to navigate when needed.
2. **Error Handling:** The system must handle invalid inputs, displaying appropriate error messages.
3. **Scalability:** The system must efficiently handle room bookings and manage up to 100 rooms.
4. **Performance:** The system should provide accurate information about the room availability and user details without making and mistakes and in a short time with efficiency nonetheless.
   1. **Analysis Phase**

**System Input**

* **Room Number:** The number of the room a user wishes to book or check out.
* **User Details:** Information like: Name(to identify to whom the room was booked), ID(for detailed identification), and phone number(to call the customer when needed) to register a user during the room booking process.
* **Search ID:** A user ID used to search for a specific user's room reservation.

**System Outputs**

* **Confirmation/Failure Messages:** Messages confirming room booking, and successful checkout needs to be displayed, as well as the errors due to invalid operations.
* **Room Availability Information:** It must also display all of the rooms available for booking and also those that are already reserved.
* **Room Details:** Information about the type, price, and availability of each room needs to be displayed with the utmost transparency.
* **User List:** A list of all users who have booked rooms.

**Constraints**

* The system will be designed to manage up to 100 rooms.
* The input will be limited to room numbers ranging from 1 to 100.
* The system will operate as a command-line interface with no GUI.
  1. **Design Phase**

**System Architecture**

The Utopia Hotel Reservation System utilizes a modular architecture that is simple and effective. The system is built to be console-based, as it is following a layered architecture where each of its components is responsible for a singular specific task. The system is designed using a procedural programming model relies on arrays to store room types, prices, availability, and user details. Additionally, a prefix sum array is used to manage the room ranges efficiently.

**Components**

1. **Room Management:**
   * **Room Types**: Is an array storing the names of room types: Single, Double, Family, Deluxe, and Suite.
   * **Room Prices**: An array storing the price of each room type is used to display the prices of the required rooms.
   * **Room Sizes**: Is an array indicating the number of rooms available for each type is also used here.
   * **Room Availability**: Is a boolean array used for tracking whether a specific room is available for booking.
2. **User Management:**
   * **Personal Information**: A 2D array storing all the user details, including their name, ID, and phone number which is then associated with each of the ordered rooms.
3. **Functional Modules:**
   * **Book Room**: This module allows the users to book available rooms.
   * **Show room Availability**: This is used to display the status of all rooms (available or booked).
   * **Show Room Details**: This module shows a detailed information about each room.
   * **Show Users List**: Used for displaying a list of all users currently staying in the hotel.
   * **Search By Id**: Allows users to search for their booking using their ID.
   * **Leave Room**: Allows the users to check out of the rooms to free them
   1. **Development Phase**

The development phase aims to transform the design documents into actual code that implements the system's functionality. This phase involves writing, reviewing, and testing code in alignment with the project’s technical specifications and requirements.

The development of the Utopia Hotel Reservation System follows the iterative approach, allowing for continuous integration and incremental development. The code is developed using C++ as the primary programming language, and other libraries are used to facilitate input/output, string handling, and formatting.

The primary activities during the development phase include:

1. **Code Implementation:**

The C++ programming language was utilized to develop the Utopia Hotel Reservation System. Supporting libraries such as <iostream>, <iomanip>, and <string> were used for input/output operations, data formatting, and string manipulation.

Core components and methods implemented:

* + **Room Management:** Handled by bookRoom(), showRoomAvailability(), and leaveRoom() functions. A boolean array (roomAvailability[]) tracks room statuses.
  + **User Management:** Managed by registerInfo() and searchById(), storing user details in a multi-dimensional array (personalInfo[][]).
  + **Room Categorization:** Achieved using prefixRoomSizes[], mapping room types to their respective ranges.
  + **Display Module:** Functions like showRoomDetails() and showUsersList() format and output data.

1. **Version Control:**
   * Git was employed to manage the source code, with all changes tracked and managed through feature-specific branches. Merges into the main branch were performed after successful code reviews and testing.
2. **Code Review:**
   * Peer reviews were conducted to ensure compliance with coding standards and eliminate errors. Feedback from reviewers was addressed to refine the code.
3. **Completion of Development:**
   * The modules were integrated into a cohesive system, ensuring proper communication between components. All development tasks were completed and verified before transitioning to the testing phase.

**Challenges and Solutions**

1. **Room Availability Management:**

**Challenge:** Avoiding double-booking.

**Solution:** Implemented a boolean array (roomAvailability[]) and ensured state updates within critical functions.

1. **Input Validation:**

**Challenge:** Preventing invalid or unexpected inputs.

**Solution:** Input checks (cin.fail() and loops) embedded in functions like bookRoom() and showMenu().

* 1. **Testing Phase**

The testing phase done after the development phase was finalized. This phase aimed to validate the functionality, reliability, and performance of the Utopia Hotel Reservation System. Testing activities were conducted sequentially, as outlined below.

**Testing Activities**

1. **Unit Testing:**
   * Individual functions were tested to ensure they performed as expected.
   * **Examples:**
     + **bookRoom():** Tested with valid and invalid room numbers to ensure proper status updates.
     + **searchById():** Verified retrieval of user details based on unique IDs.
2. **Integration Testing:**
   * Ensured smooth interaction between modules.
   * **Example:** Verified that user registration data (registerInfo()) was correctly linked to room booking (bookRoom()).
3. **System Testing:**
   * Comprehensive end-to-end testing simulated user interactions.
   * **Scenarios:**
     + Booking a room, checking its availability, and then releasing it.
     + Searching for user details by ID.
4. **User Acceptance Testing (UAT):**
   * A group of end-users evaluated the system’s usability and functionality. And this test was done by selected three persons as indicated on table 1.
5. **Regression Testing:**
   * Conducted after each update to ensure no existing functionality was compromised.
6. **Performance Testing:**
   * Tested the system under high load conditions (e.g., multiple concurrent room bookings) to ensure stability and responsiveness.

**Bug Management**

1. **Bug Tracking:**
   * A detail Bug Tracking was done.
2. **Bug Categories:**
   * **Critical Bugs:** Prevented core functionalities (e.g., room booking) from working.
   * **Minor Bugs:** Included UI inconsistencies and non-critical glitches.
3. **Bug Resolution Workflow:**
   * Bugs were assigned to developers, resolved, and retested to confirm fixes.

**Our test cases**

*Table 1.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Case** | **Expected Outcome** | **Result** | **Trails** |
| 1 | Book a valid room | *“Room booked successfully”* | Pass✅ | 3 |
| 2 | Book an already booked room | “*Room already booked”* | Pass✅ | 3 |
| 3 | Search by a non-existent user Id | *“User not found”* | Pass✅ | 3 |
| 4 | Leave a room | *“Room* vacated *successfully”* | Pass✅ | 3 |
| 5 | Input invalid room number for booking | *“Invalid room number”* | Pass✅ | 3 |

* 1. **Deployment**

**Deployment** is the phase in the **Software Development Life Cycle (SDLC)** where the developed software is delivered and installed in the production environment, making it accessible and functional for end users. It ensures that the software is configured, tested, and deployed in a way that meets the stakeholders' expectations. This phase is critical as it transitions the software from development to operational status.

The deployment phase involves all the steps necessary to make the developed Hotel Management System operational for its intended users. This is a critical step to ensure a smooth transition from development to production, and it includes preparation, distribution, installation, and initial support.

* + 1. **Deployment Goals**
  + Deliver the application to the users in a functional and accessible manner.
  + Ensure that the system is installed and configured correctly on the target environment.
  + Minimize downtime and disruptions to the end-users during the deployment process.
    1. **Steps for Deployment**

**Step 1: Preparation**

* **System Requirements:**
  + Ensure the target machines meet the software's requirements, such as:

Operating system compatibility (Windows, Linux, etc.).

* + - Hardware specifications: RAM, CPU, and disk space.
    - Software dependencies: A C++ runtime environment if required.
* **Packaging:**
  + Compile the source code into an executable format.
  + Package the executable, along with:
    - A user manual.
    - Any necessary dependencies.
    - Installation instructions.
* **Backup Plan:**
  + Prepare for rollbacks in case the deployment encounters critical issues.

**Step 2: Installation**

* **Standalone Installation:**
  + For a single machine, directly copy the executable and run it.
  + Provide clear instructions for the installation process, such as placing the executable in a specified directory.
* **Network Installation:**
  + If the system is used on multiple machines, deploy it via a shared network or centralized server.
  + Configure access rights to ensure authorized usage.

**Step 3: Testing the Deployment**

* Test the deployed system in the production environment to confirm:
  + The program runs without errors.
  + User interaction is smooth and intuitive.
  + Data such as room availability updates correctly.

**Step 4: Training and Documentation**

* **User Training:**
  + Train hotel staff on how to:
    - Use the menu options (e.g., booking a room, checking availability).
    - Handle common tasks and troubleshoot basic issues.
* **Documentation:**
  + Include:
    - A quick-start guide for staff.
    - A technical manual for IT personnel detailing configuration, troubleshooting, and maintenance procedures.

**Step 5: Support Setup**

* Establish a channel for users to report issues (e.g., email support, chat support).
  + 1. **Types of Deployment**
* **Direct Deployment:** Install directly onto the production environment without intermediate steps. Suitable for small-scale systems like this one.
* **Phased Deployment:** Introduce the system gradually to parts of the hotel (e.g., one department) before full rollout.
* **Pilot Deployment:** Test the system with a subset of users to gather feedback.
  1. **Maintenance**

**Maintenance** refers to the continuous process of monitoring and improving the software after its deployment. It ensures that the system remains effective, bug-free, and adaptable to changing requirements over time. This phase is essential to prolong the software’s lifecycle and maintain user satisfaction.

Maintenance is an ongoing phase that starts after the system has been deployed. Its purpose is to ensure the system remains functional, efficient, and aligned with user requirements over time.

* + 1. **Goals of Maintenance**
* **System Stability:** Keep the system error-free and operational.
* **User Satisfaction:** Address user feedback and implement desired features.
* **Adaptability:** Ensure the system continues to meet changing business needs.

**2. Types of Maintenance**

**A. Corrective Maintenance**

* **Definition:** Fix bugs or errors encountered during use.
* **Examples:**
  + Fixing an issue where entering a non-integer value for room selection crashes the program.
  + Resolving formatting problems in the room details display.
* **Process:**
  + Identify the issue from logs or user feedback.
  + Debug and implement a fix.
  + Test the fix thoroughly before applying it to the production system.

**B. Adaptive Maintenance**

* **Definition:** Modify the system to respond to changes in the environment or requirements.
* **Examples:**
  + Adding support for more rooms as the hotel expands.
  + Updating the program to work with a newer version of the operating system.
* **Process:**
  + Assess the new requirements.
  + Plan and implement changes while ensuring backward compatibility.

**C. Perfective Maintenance**

* **Definition:** Enhance the system to improve performance or usability.
* **Examples:**
  + Adding a new feature for room cancellation.
  + Improving the room details display to sort rooms by price or type.
* **Process:**
  + Collect user feedback to identify potential enhancements.
  + Prioritize and implement improvements in stages.

**D. Preventive Maintenance**

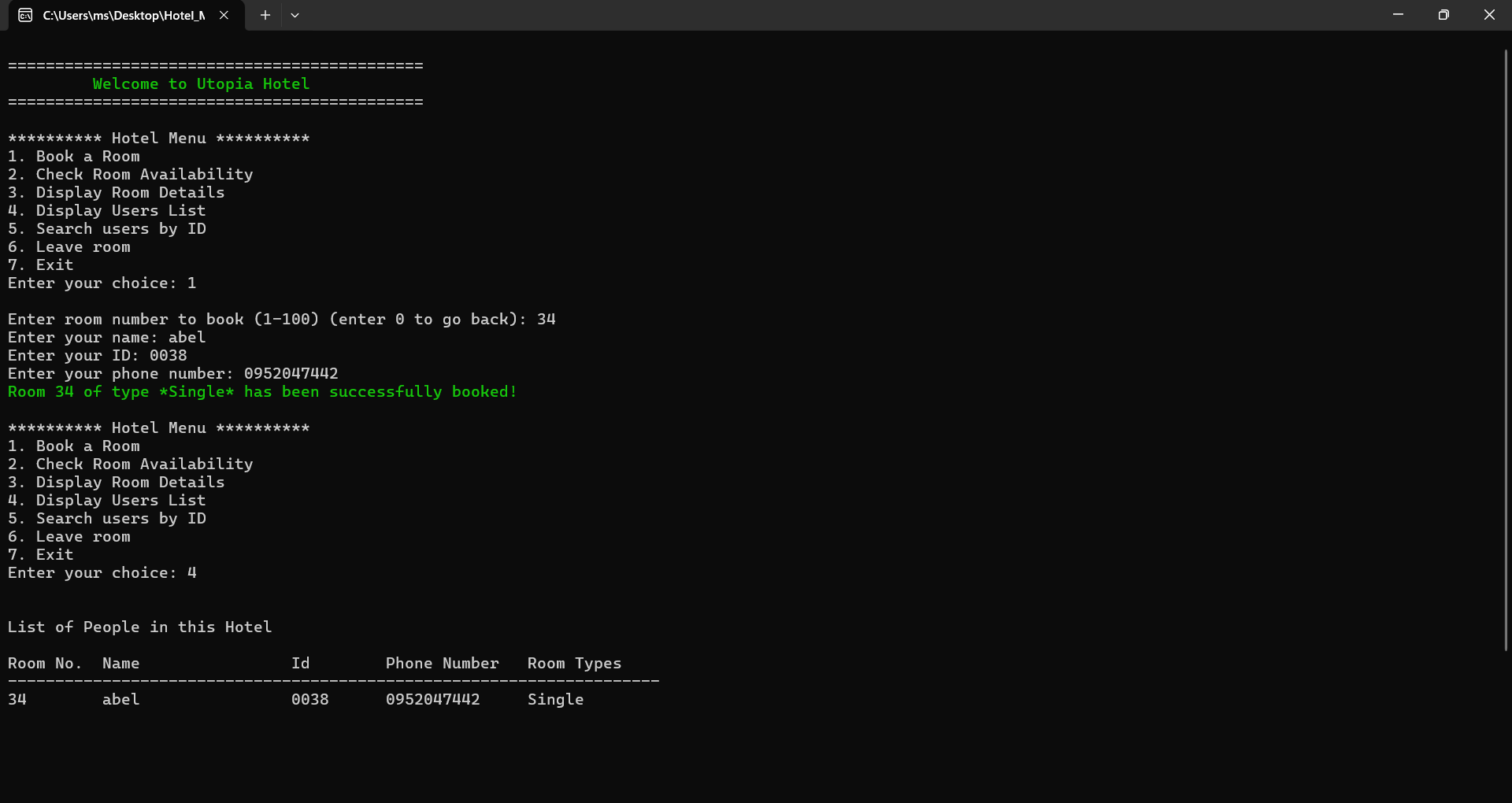
* **Definition:** Anticipate potential issues and address them before they occur.
* **Examples:**
  + Refactoring the code to make it more readable and maintainable.
  + Implementing input validation to prevent unexpected crashes.
* **Process:**
  + Conduct code reviews and performance audits regularly.
  + Optimize inefficient parts of the code.

1. **Maintenance Workflow**
   * 1. **Monitoring:**
   * Use logs or user reports to track system performance and identify issues.
   * Example: Monitor how frequently users encounter invalid input errors.
     1. **Issue Tracking:**
   * Maintain a database of known issues and their status (e.g., pending, resolved).
   * Example: Use a tool like GitHub Issues or JIRA to document and prioritize bugs.
     1. **Update Implementation:**
   * Plan and implement updates or patches in response to issues or feedback.
   * Example: Release a patch to add room cancellation functionality.
     1. **Testing Updates:**
   * Test updates in a controlled environment before deploying them to production.
   * Example: Ensure new features do not disrupt existing functionality.
     1. **Deployment of Updates:**
   * Deliver updates to users in a seamless manner, such as through downloadable patches.
2. **Tools for Maintenance**
3. **Version Control Systems (e.g., Git):**
   * Track changes to the codebase.
   * Allow rollbacks to previous versions if issues arise.
4. **Bug Tracking Tools (e.g., JIRA, Trello):**
   * Organize and prioritize bug fixes and feature requests.
5. **Testing Frameworks:**
   * Automate testing of core functionalities like booking and availability checking.
6. **Challenges in Maintenance**

* **User Adaptation:**
  + Users may find it difficult to adapt to changes, such as new features or workflows.
* **Unclear Requirements:**
  + Some user feedback may lack clarity, making it hard to implement improvements effectively.
* **Scalability Issues:**
  + As the system grows (e.g., more rooms, additional features), maintaining performance can be challenging.

**6. Long-Term Maintenance Goals**

* **Scalability:**
  + Transition from fixed arrays to dynamic structures like std::vector to handle an arbitrary number of rooms.
* **Integration:**
  + Connect the system to a database for persistent data storage.
* **Modernization:**
  + Upgrade the interface to a GUI or a web-based platform for better user experience.
* **Documentation Updates:**
  + Keep all user manuals and technical documentation up to date with system changes.



**Sample Code**

**Conclusion**

This project aimed to develop a comprehensive hotel reservation system that facilitates booking rooms, managing availability, and handling user interactions for a hotel with up to 100 rooms. By implementing key functionalities such as room reservations, user registration, room availability tracking, and searching users by ID, we have successfully created a system that simulates the operational processes of a hotel in a simple and efficient manner.

Throughout the development of this project, we gained valuable experience in designing and implementing a real-world application, applying the Software Development Life Cycle (SDLC) process from planning to deployment. We learned how to manage arrays effectively, handle user inputs and errors, and ensure that the system’s operations are both seamless and user-friendly. Furthermore, we enhanced our skills in modular programming, which allowed us to divide complex functionalities into manageable components, making the code easier to maintain and debug.

In conclusion, this project not only helped us improve our programming abilities, but it also deepened our understanding of the importance of user-centered design, clear system architecture, and thorough testing. It gave us practical exposure to solving real-world problems and reinforced the critical thinking skills necessary for software development. This project has laid a solid foundation for future work in developing more advanced and scalable systems.

**GitHub Repository**

[**https://github.com/bella-247/Hotel\_Management\_System\_cpp**](https://github.com/bella-247/Hotel_Management_System_cpp)

**Flowchart**

