Accommodation Reservation Management System

Syrone Robinson

SWEN 646 Software Design and Implementation

Date: 07/29/203

Table of Contents

[1 INTRODUCTION 2](#_Toc141548095)

[1.1 Purpose 2](#_Toc141548096)

[1.2 Scope 2](#_Toc141548097)

[1.2.1 Features 2](#_Toc141548098)

[1.2.2 Out of Scope 2](#_Toc141548099)

[1.3 Overview 2](#_Toc141548100)

[2 SYSTEM OVERVIEW 2](#_Toc141548101)

[2.1 Background Information 2](#_Toc141548102)

[2.2 Major Functionality 2](#_Toc141548103)

[2.2.1 Account Management 2](#_Toc141548104)

[2.2.2 Reservation Creation 2](#_Toc141548105)

[2.2.3 Reservation Modification 3](#_Toc141548106)

[2.2.4 Reservation Cancellation 3](#_Toc141548107)

[2.2.5 Pricing and Billing 3](#_Toc141548108)

[2.2.6 Accommodation Types 3](#_Toc141548109)

[2.3 System Context 3](#_Toc141548110)

[2.3.1 Users 3](#_Toc141548111)

[2.3.2 Data Management 4](#_Toc141548112)

[2.4 Design Approach 4](#_Toc141548113)

[3 System Architecture 4](#_Toc141548114)

[3.1 Architectural Design 4](#_Toc141548115)

[3.2 Decomposition Description 5](#_Toc141548116)

[3.2.1 Main Class 5](#_Toc141548117)

[3.2.2 Manager Class 6](#_Toc141548118)

[3.2.3 Account Class 6](#_Toc141548119)

[3.2.4 Reservation Class (Abstract) 6](#_Toc141548120)

[3.2.5 Specific Reservation Classes (“CabinReservation”, “HotelReservation”, “HouseReservation”) 7](#_Toc141548121)

[3.2.6 Data Flow 7](#_Toc141548122)

[3.3 Exception Handling 8](#_Toc141548123)

[3.3.1 DuplicateObjectException 8](#_Toc141548124)

[3.3.2 IllegalLoadException 8](#_Toc141548125)

[3.3.3 IllegalOperationException 9](#_Toc141548126)

[3.4 Design Rationale 9](#_Toc141548127)

[3.4.1 Modularity 9](#_Toc141548128)

[3.4.2 Reusability 9](#_Toc141548129)

[3.4.3 Extensibility 9](#_Toc141548130)

[3.4.4 Flexibility 10](#_Toc141548131)

[3.4.5 Exception Handling 10](#_Toc141548132)

[3.4.6 Alternative Architectures Considered 10](#_Toc141548133)

[4 Data Design 10](#_Toc141548134)

[4.1 Data Description 10](#_Toc141548135)

[4.1.1 Data Storage Files 10](#_Toc141548136)

[4.2 Data Dictionary 12](#_Toc141548137)

[4.2.1 Account 12](#_Toc141548138)

[4.2.2 Reservation 13](#_Toc141548139)

[5 Component Design 14](#_Toc141548140)

[6 HUMAN INTERFACE DESIGN 24](#_Toc141548141)

[6.1 Overview of User Interface 24](#_Toc141548142)

[6.1.1 Create Account 24](#_Toc141548143)

[6.1.2 View Account Information 24](#_Toc141548144)

[6.1.3 Create Reservation 24](#_Toc141548145)

[6.1.4 Update Account 25](#_Toc141548146)

[6.1.5 Update Reservation 25](#_Toc141548147)

[6.1.6 Cancel Reservation 25](#_Toc141548148)

[6.1.7 Exit 25](#_Toc141548149)

[6.2 Screen Images 25](#_Toc141548150)

[6.2.1 Main Menu Screen 25](#_Toc141548151)

[6.2.2 Create Account Screen 25](#_Toc141548152)

[6.2.3 View Account Information Screen 26](#_Toc141548153)

[6.2.4 Create Reservation Screen 26](#_Toc141548154)

[6.2.5 Update Account Screen 26](#_Toc141548155)

[6.2.6 Update Reservation Screen 27](#_Toc141548156)

[6.2.7 Cancel Reservation Screen 27](#_Toc141548157)

[6.3 Screen Objects and Actions 27](#_Toc141548158)

[6.3.1 Reservation System's User Interface 27](#_Toc141548159)

[7 Requirement Matrix 28](#_Toc141548160)

# INTRODUCTION

## Purpose

The purpose of this Software Design Document is to describe the architecture and system design of the Accommodation Reservation System. This document is intended for developers, architects, and stakeholders involved in the development and maintenance of the system.

## Scope

The Accommodation Reservation System is designed to facilitate the reservation of various types of lodgings, including hotels, cabins, and houses. The system allows customers to create accounts, make reservations, view and modify existing reservations, and cancel reservations if needed. The goals of the system are to provide a user-friendly and efficient platform for customers to reserve accommodations and to help accommodation providers manage their reservations effectively.

### Features

* Account Management: Customers can create accounts, update their personal information, and manage their reservations.
* Reservation Creation: Customers can create new reservations for different types of accommodations (hotels, cabins, or houses).
* Reservation Modification: Customers can modify existing reservations, such as changing dates or the number of guests.
* Reservation Cancellation: Customers can cancel reservations within a specified time frame before the reservation start date.
* Pricing: The system automatically calculates reservation prices based on accommodation type, duration, and additional amenities.

### Out of Scope

The system does not handle payment processing or financial transactions. Additionally, the system does not include user authentication or authorization features, as it assumes that users have already been authenticated through external means.

## Overview

This Software Design Document is organized into several sections, each focusing on different aspects of the Accommodation Reservation System. The document provides an overview of the system's architecture, modules, and the relationships between them. It also includes detailed descriptions of the major components, classes, and their functionalities. Additionally, this document explains the data structures, file handling, and the flow of information within the system.

The subsequent sections of this document will delve into the architecture, classes, and methods of the system, describing how it achieves its objectives and functionalities in a comprehensive manner.

# SYSTEM OVERVIEW

The Accommodation Reservation System is a comprehensive software application that facilitates the reservation of various types of lodgings, including hotels, cabins, and houses. The system provides a user-friendly platform for customers to create accounts, make reservations, modify existing reservations, and cancel reservations when necessary. Accommodation providers can efficiently manage reservations and lodging information through the system.

## Background Information

The Accommodation Reservation System was developed to address the growing demand for a centralized and automated reservation system in the hospitality industry. Traditional reservation methods often involve manual processes and may not provide real-time availability information to customers. This system aims to streamline the reservation process, enhance customer experience, and optimize accommodation management for providers.

## Major Functionality

### Account Management

Customers can create accounts on the platform by providing their personal information, including mailing address, email address, and phone number. Account holders can also update their account details.

### Reservation Creation

The system allows customers to create new reservations for their desired accommodation type (hotel, cabin, or house). During the reservation process, customers can specify the lodging's physical and mailing addresses, start date, number of nights, number of guests, and other relevant details.

### Reservation Modification

Customers have the flexibility to modify their existing reservations based on their changing travel plans. They can change reservation dates, the number of nights, or other preferences, subject to availability and specific rules.

### Reservation Cancellation

Customers can cancel their reservations within a designated time frame before the reservation start date. Upon cancellation, the system updates the reservation status and applies any applicable cancellation policies.

### Pricing and Billing

The system automatically calculates reservation prices based on the accommodation type, duration, and additional amenities selected by customers. It presents the total price to customers during the reservation process.

### Accommodation Types

The system supports three main types of accommodations: hotels, cabins, and houses. Each accommodation type may have specific amenities and pricing structures.

## System Context

The Accommodation Reservation System is a standalone application that interacts with customers through a user-friendly web interface. The system also maintains a database to store account information, reservations, and relevant lodging details.

### Users

The primary users of the system are customers seeking accommodation reservations and accommodation providers managing their lodging offerings.

### Data Management

The system stores account data, reservation information, and lodging details in text files. The database (text files) facilitates efficient data retrieval, updates, and retrieval of reservation records.

## Design Approach

The Accommodation Reservation System adopts a modular and object-oriented design approach. It consists of several key modules, such as the Account Management Module, Reservation Module, and Pricing Module, each responsible for specific functionalities. The use of object-oriented design promotes code reusability, maintainability, and scalability.

The system also employs file handling mechanisms to store account and reservation data. Accounts are stored in individual text files, while reservations are saved as separate text files within the respective account folders.

Overall, the design emphasizes usability, reliability, and efficiency, ensuring a seamless experience for customers and accommodation providers alike.

# System Architecture

## Architectural Design

The Accommodation Reservation System follows a layered architectural design to promote separation of concerns and maintain a modular structure. The major subsystems and their interconnections are represented in the class diagram below:

A screenshot of a computer

Description automatically generated

## Decomposition Description

The Accommodation Reservation System is decomposed into several subsystems, each represented by specific classes that work together to provide the overall functionality of the system. Below is an explanation of how these classes interact, their relationships, and the data flow through the system:

### Main Class

* The “Main” class serves as the entry point for the program.
* It initializes the “Manager” class, which is the central controller of the system.
* The “Main” class interacts with the user through the console to receive commands and input.

### Manager Class

* The “Manager” class acts as the central controller of the system, coordinating interactions between different components.
* It holds collections of “Account” objects and manages reservations associated with each account.
* The “Manager” class is responsible for loading and saving data to and from the data storage files.
* It provides methods for adding new accounts, creating reservations, modifying reservation details, and retrieving account and reservation information.

### Account Class

* An “Account” object represents a single user account in the system.
* Each “Account” object contains a collection of “Reservation” objects associated with that account.
* The “Account” class provides methods to add, remove, and retrieve reservations.
* It also allows the modification of account information such as mailing address, email address, and phone number.

### Reservation Class (Abstract)

* The “Reservation” class is an abstract class that represents a reservation made by an account.
* It contains common attributes like reservation number, lodging address, start date, number of nights, etc., and provides common methods like “calculatePrice()” for calculating reservation prices.
* This class is subclassed by specific types of reservations, such as “CabinReservation”, “HotelReservation”, and “HouseReservation”.
* Subclasses of “Reservation” define additional attributes and methods specific to each reservation type.
* The `Reservation` class implements the reservation status using the “ReservationStatus” enumeration.

### Specific Reservation Classes (“CabinReservation”, “HotelReservation”, “HouseReservation”)

* Each specific reservation class represents a particular type of lodging reservation (cabin, hotel, house).
* They inherit from the “Reservation” class and provide additional attributes and methods specific to their respective types of reservations.
* Each specific reservation class overrides the “calculatePrice()” method to calculate the price based on its unique criteria.
* The “Manager” class creates instances of these reservation classes when adding new reservations to an account.

### Data Flow

* The data flow through the system begins when the “Main” class starts and initializes the “Manager”.
* The “Manager” class loads existing accounts and reservations from the data storage files and stores them in collections of “Account” objects.
* As the user interacts with the system through the console, the “Manager” class processes the commands and calls appropriate methods to perform the required operations.
* When new reservations are added or account information is modified, the changes are reflected in the “Account” and “Reservation” objects held by the “Manager”.
* The “Manager” class also handles data persistence, ensuring that any changes made to accounts and reservations are saved back to the data storage files.
* Additionally, the system may throw custom exceptions (e.g., “AccountNotFoundException”, “ReservationNotFoundException”, “DataLoadException”, “DataSaveException”, etc.) when exceptional situations occur, providing appropriate error handling and feedback to the user.

## Exception Handling

The Accommodation Reservation System utilizes custom exceptions to handle exceptional situations that may occur during the execution of the program. These custom exceptions provide meaningful error messages and help the system respond appropriately to exceptional conditions.

### DuplicateObjectException

* Description: This exception is thrown when attempting to add a duplicate object (e.g., reservation) to a collection that does not allow duplicates.
* Constructor(s): "DuplicateObjectException(String message)"
* Message: {message} (Custom message describing the reason for the exception)
* Thrown by: Various methods in "Account" and "Manager" classes when handling reservations and accounts.

### IllegalLoadException

* Description: This exception is thrown when an attempt is made to load data from a file in an illegal or unsupported format.
* Constructor(s): "IllegalLoadException(String message)"
* Message: {message} (Custom message describing the reason for the exception)
* Thrown by: Methods in "Manager" class when loading data from files, and if the data is in an illegal or unsupported format.

### IllegalOperationException

* Description: This exception is thrown when an illegal operation is attempted (e.g., modifying a reservation with an invalid state).
* Constructor(s): "IllegalOperationException(String message)"
* Message: {message} (Custom message describing the reason for the exception)
* Thrown by: Various methods in "Reservation" and "Manager" classes when validating reservation operations.

## Design Rationale

The selected layered architectural design provides a clear separation of concerns and promotes modularity within the system. The rationale behind this architecture includes:

### Modularity

By decomposing the system into separate subsystems, each responsible for specific tasks, it becomes easier to maintain, extend, and modify individual components without affecting the others.

### Reusability

The Reservation class and its subclasses promote code reuse by inheriting common attributes and behaviors from the parent class. This enables efficient development and maintenance of various accommodation-specific reservation types.

### Extensibility

The use of an abstract Reservation class allows for the addition of new reservation types in the future by simply extending the base class and providing accommodation-specific details.

### Flexibility

The Manager acts as a mediator between the user interface and the underlying data model. This provides flexibility to change the data storage mechanism without affecting the user interface.

### Exception Handling

User-defined exceptions allow for more granular error handling and provide meaningful feedback to users in case of exceptional situations.

### Alternative Architectures Considered

* **Component-Based Architecture**: We considered a component-based architecture where each major component (User Interface, Manager, Account, Reservation, etc.) would be implemented as separate reusable components. However, we chose the layered architecture as it provides a more straightforward structure for a small-scale application like the Accommodation Reservation System.
* **Microservices Architecture**: While the microservices architecture offers scalability and independent deplorability, it may introduce unnecessary complexity for this relatively small project. The layered architecture suits the current requirements and allows for future enhancements if needed.

# Data Design

## Data Description

The Accommodation Reservation System stores and organizes its data using text files for persistence. The major entities in the system are "Account" and "Reservation", and they are stored in separate files in the data directory. Below is a detailed description of the data storage format:

### Data Storage Files

**Accounts Data File: "data/accounts/Acc-accountNumber/acc-accountNumber.txt"**

* Naming Convention: The data for each account is stored in a separate folder named "Acc-accountNumber," where "accountNumber" is the unique account identifier.
* File Format: The account data is stored in plain text format.
* Example Contents:

Account Number: 64245385

Mailing Address: 123 Main Street

Email Address: test@gmail.com

Phone Number: (123) 456-7890

**Reservations Data Files: "data/accounts/Acc-accountNumber/res-reservationNumber.txt"**

* Naming Convention: Each reservation for an account is stored in a separate file named "res-reservationNumber," where "reservationNumber" is the unique reservation identifier.
* File Format: The reservation data is stored in plain text format.
* Example Contents (for a Cabin Reservation):

Reservation Number: 0317543823

Physical Address: 456 Oak Avenue

Mailing Address: 123 Main Street

Start Date: 2023-08-10

Number of Nights: 3

Number of Beds: 2

Number of Bedrooms: 1

Number of Bathrooms: 1

Lodging Size: 800

Full Kitchen: true

Price: $145.0

Status: COMPLETED

## Data Dictionary

### Account

- Type: Class

- Description: Represents a user account for accommodation reservations.

- Attributes:

1. "accountNumber" (String): The unique identifier for the account.
2. "mailingAddress" (String): The mailing address associated with the account.
3. "reservations" (List<Reservation>): A collection of reservations associated with the account.
4. "emailAddress" (String): The email address associated with the account.
5. "phoneNumber" (String): The phone number associated with the account.

- Methods:

1. "getAccountNumber()": Returns the account number.
2. "getMailingAddress()": Returns the mailing address.
3. "setMailingAddress(String address)": Sets the mailing address.
4. "getReservations()": Returns the list of reservations.
5. "getEmailAddress()": Returns the email address.
6. "setEmailAddress(String email)": Sets the email address.
7. "getPhoneNumber()": Returns the phone number.
8. "setPhoneNumber(String phone)": Sets the phone number.
9. addReservation(Reservation reservation)": Adds a new reservation to the account.

### Reservation

- Type: Abstract Class (Superclass for specific reservation types)

- Description: Represents a lodging reservation made by an account.

- Attributes:

1. "accountNumber" (String): The account number associated with the reservation.
2. "reservationNumber" (String): The unique identifier for the reservation.
3. "lodgingPhysicalAddress" (String): The physical address of the lodging.
4. "lodgingMailingAddress" (String): The mailing address of the lodging.
5. "startDate" (Date): The start date of the reservation.
6. "numberOfNights" (int): The number of nights for the reservation.
7. "numberOfBeds" (int): The number of beds in the lodging.
8. "numberOfBedrooms" (int): The number of bedrooms in the lodging.
9. "numberOfBathrooms" (int): The number of bathrooms in the lodging.
10. "lodgingSize" (int): The size of the lodging in square feet.
11. "price" (double): The price of the reservation.
12. "status" (ReservationStatus): The status of the reservation (e.g., DRAFT, COMPLETED, CANCELLED).

- Methods:

1. "getAccountNumber()": Returns the account number associated with the reservation.
2. "getReservationNumber()": Returns the reservation number.
3. "getLodgingPhysicalAddress()": Returns the physical address of the lodging.
4. "setLodgingPhysicalAddress(String address)": Sets the physical address of the lodging.
5. "getLodgingMailingAddress()": Returns the mailing address of the lodging.
6. "setLodgingMailingAddress(String address)": Sets the mailing address of the lodging.
7. "getStartDate()": Returns the start date of the reservation.
8. "getNumberOfNights()": Returns the number of nights for the reservation.
9. "getNumberOfBeds()": Returns the number of beds in the lodging.
10. "getNumberOfBedrooms()": Returns the number of bedrooms in the lodging.
11. "getNumberOfBathrooms()": Returns the number of bathrooms in the lodging.
12. "getLodgingSize()": Returns the size of the lodging.
13. "getPrice()": Returns the price of the reservation.
14. "getStatus()": Returns the status of the reservation.
15. "setStatus(ReservationStatus status)": Sets the status of the reservation.
16. "calculatePrice()": Abstract method to calculate the price of the reservation.

# Component Design

**Class Name: Main**

Class Description/Purpose: Handles the user interface and interactions for the lodging reservation system. It serves as the entry point, provides a text-based menu, and interacts with the "Manager" class.

Class Modifiers: Public

Class Inheritance: None

Class Attributes:

- "private static final Scanner scanner = new Scanner(System.in)": Static Scanner object to read user input.

- "private static final Manager manager = new Manager()": Static Manager object to manage interactions and data.

Exceptions Thrown: IllegalOperationException

Class Constructors: None

Class Methods:

- "public static void main(String[] args)": Entry point, displays menu, and handles user interactions.

- "private static int getChoice()": Reads and returns user's menu choice.

- "private static void createAccount()": Creates an account and saves it to a file.

- "private static void viewAccountInformation()": Displays information about a specific account.

- "private static void createReservation()": Creates a reservation for an account and saves it to a file.

- "private static void updateAccount()": Updates account information and saves changes to a file.

- "private static void updateReservation()": Updates reservation information and saves changes to a file.

- "private static void cancelReservation()": Cancels a reservation and updates its status in the file.

- "private static Date parseDate(String dateString)": Helper method to parse a date from a string.

**Class Name: Manager**

Class Description/Purpose: Manages the lodging reservation system, including loading accounts and reservations from files, handling account and reservation data, and saving changes to files.

Class Modifiers: Public

Class Inheritance: None

Class Attributes:

- "private final List<Account> accounts": List to store account objects.

- "private static final String ACCOUNTS\_DIRECTORY = "data/accounts"": Constant representing the directory where account data is stored.

- Exceptions Thrown: DuplicateObjectException, IllegalLoadException

Class Constructors:

- "public Manager()": Initializes the manager.

Class Methods:

- "private void loadAccountsFromFolders()": Loads accounts from account folders in the "data/accounts" directory.

- "private void loadReservationsForAccount()": Loads reservations for a given account from reservation files in the account folder.

- "private Account parseAccountData()": Parses account data from the file content and returns an Account object.

- "private int determineReservationType(String reservationData)": Determines the reservation type based on the content of the reservation file.

- "private Reservation parseReservationData(String reservationData)": Parses reservation data from the file content and returns a Reservation object.

- "public Account getAccount(String accountNumber)": Retrieves an account-by-account number.

- "public void addAccount(Account account)": Adds an account to the manager.

- "public void updateAccountFiles(Account account)": Updates the account file with the latest account information.

- "public void addReservation(String accountNumber, Reservation reservation)": Adds a reservation to an account.

- "public void completeReservation(String accountNumber, String reservationNumber)": Completes a reservation by marking it as completed and calculating its price.

- "public void cancelReservation(String accountNumber, String reservationNumber)": Cancels a reservation by marking it as canceled.

- "public double calculatePricePerNight(String reservationNumber)": Calculates the price per night for a reservation.

- "public double calculateTotalPrice(String reservationNumber)": Calculates the total price for a reservation.

- "public Reservation getReservation(String reservationNumber)": Retrieves a reservation-by-reservation number.

- "private boolean reservationExists(String reservationNumber)": Checks if a reservation already exists.

- "public void updateReservation(String accountNumber, Reservation updatedReservation)": Updates a reservation.

- "public void saveAccount(Account account, String directoryPath)": Saves an account to a file.

- "public void saveReservation(Reservation reservation, String directoryPath)": Saves a reservation to a file.

- “public List<Reservation> getAllReservationsForAccount(String accountNumber)”:

Retrieves all reservations associated with a given account number, or it returns an empty list if the account is not found.

**Class Name: Account**

Class Description/Purpose: Represents an account in the lodging reservation system, holding information such as account number, contact details, and associated reservations.

Class Modifiers: Public

Class Inheritance: None

Class Attributes:

- "private final String accountNumber": Holds the account number.

- "private String mailingAddress": Holds the mailing address.

- "private final List<Reservation> reservations": List to store associated reservations.

- "private String emailAddress": Holds the email address.

- "private String phoneNumber": Holds the phone number.

Exceptions Thrown: None

Class Constructors:

- "public Account(String accountNumber, String mailingAddress, String emailAddress, String phoneNumber)": Initializes the account with the provided details.

Class Methods:

- "public String getAccountNumber()": Returns the account number.

- "public String getMailingAddress()": Returns the mailing address.

- "public void setMailingAddress(String mailingAddress)": Sets the mailing address.

- "public List<Reservation> getReservations()": Returns the list of associated reservations.

- "public String getEmailAddress()": Returns the email address.

- "public void setEmailAddress(String emailAddress)": Sets the email address.

- "public String getPhoneNumber()": Returns the phone number.

- "public void setPhoneNumber(String phoneNumber)": Sets the phone number.

- "public void addReservation(Reservation reservation)": Adds a reservation to the account.

- "@Override public String toString()": Generates a formatted string representation of the Account object.

**Class Name: Reservation**

Class Description/Purpose: Represents an abstract reservation in the lodging reservation system, providing common attributes and methods for various types of reservations.

Class Modifiers: Abstract

Class Inheritance: None

Class Attributes:

- "private final String accountNumber": Holds the account number associated with the reservation.

- "private final String reservationNumber": Holds the reservation number.

- "private String lodgingPhysicalAddress": Holds the physical address of the lodging.

- "private String lodgingMailingAddress": Holds the mailing address of the lodging.

- "private final Date startDate": Holds the start date of the reservation.

- "private final int numberOfNights": Holds the number of nights for the reservation.

- "private final int numberOfBeds": Holds the number of beds in the lodging.

- "private final int numberOfBedrooms": Holds the number of bedrooms in the lodging.

- "private final int numberOfBathrooms": Holds the number of bathrooms in the lodging.

- "private final int lodgingSize": Holds the size of the lodging.

- "private double price": Holds the price of the reservation.

- "private ReservationStatus status": Holds the status of the reservation.

Exceptions Thrown: None

Class Constructors:

- "public Reservation(String accountNumber, String reservationNumber, String lodgingPhysicalAddress, String lodgingMailingAddress, Date startDate, int numberOfNights, int numberOfBeds, int numberOfBedrooms, int numberOfBathrooms, int lodgingSize)": Initializes the reservation with the provided details.

Class Methods:

- "public String getAccountNumber()": Returns the account number associated with the reservation.

- "public String getReservationNumber()": Returns the reservation number.

- "public String getLodgingPhysicalAddress()": Returns the physical address of the lodging.

- "public void setPrice(double price)": Sets the price of the reservation.

- "public String getLodgingMailingAddress()": Returns the mailing address of the lodging.

- "public Date getStartDate()": Returns the start date of the reservation.

- "public int getNumberOfNights()": Returns the number of nights for the reservation.

- "public int getNumberOfBeds()": Returns the number of beds in the lodging.

- "public int getNumberOfBedrooms()": Returns the number of bedrooms in the lodging.

- "public int getNumberOfBathrooms()": Returns the number of bathrooms in the lodging.

- "public int getLodgingSize()": Returns the size of the lodging.

- "public double getPrice()": Returns the price of the reservation.

- "public ReservationStatus getStatus()": Returns the status of the reservation.

- "public void setStatus(ReservationStatus status)": Sets the status of the reservation.

- "public abstract void calculatePrice()": Abstract method to calculate the price of the reservation (to be implemented by subclasses).

- "@Override public String toString()": Generates a formatted string representation of the Reservation object.

- "public void setLodgingPhysicalAddress(String lodgingPhysicalAddress)": Sets the physical address of the lodging.

- "public void setLodgingMailingAddress(String lodgingMailingAddress)": Sets the mailing address of the lodging.

“ public void setReservationNumber(String newReservationNumber)”: This method sets the reservation number with the given input while retaining only the numeric part and adding a prefix.

**Class Name: CabinReservation**

Class Description/Purpose: Represents a cabin reservation in the lodging reservation system, providing specific attributes and methods for cabin reservations.

Class Modifiers: Public

Class Inheritance: Inherits from the "Reservation" class.

Class Attributes:

- "private boolean fullKitchen": Indicates whether the cabin reservation includes a full kitchen or not.

Exceptions Thrown: None

Class Constructors:

- "public CabinReservation(String accountNumber, String reservationNumber, String lodgingPhysicalAddress, String lodgingMailingAddress, Date startDate, int numberOfNights, int numberOfBeds, int numberOfBedrooms, int numberOfBathrooms, int lodgingSize, boolean fullKitchen)": Initializes the cabin reservation with the provided details.

Class Methods:

- "public boolean isFullKitchen()": Returns "true" if the cabin reservation includes a full kitchen; otherwise, returns "false".

- "public void setFullKitchen(boolean fullKitchen)": Sets the value of the "fullKitchen" attribute.

- "@Override public void calculatePrice()": Overrides the parent class method to calculate the price of the cabin reservation based on specific rules.

- "@Override public String toString()": Generates a formatted string representation of the CabinReservation object, including the parent class's "toString()" output along with the value of "fullKitchen".

**Class Name: HotelReservation**

Class Description/Purpose: Represents a hotel reservation in the lodging reservation system, providing specific attributes and methods for hotel reservations.

Class Modifiers: Public

Class Inheritance: Inherits from the "Reservation" class.

Class Attributes:

- "private boolean kitchenette": Indicates whether the hotel reservation includes a kitchenette or not.

- Exceptions Thrown: None

Class Constructors:

- "public HotelReservation(String accountNumber, String reservationNumber, String lodgingPhysicalAddress, String lodgingMailingAddress, Date startDate, int numberOfNights, int numberOfBeds, int numberOfBedrooms, int numberOfBathrooms, int lodgingSize, boolean kitchenette)": Initializes the hotel reservation with the provided details.

Class Methods:

- "public boolean hasKitchenette()": Returns "true" if the hotel reservation includes a kitchenette; otherwise, returns "false".

- "public void setKitchenette(boolean kitchenette)": Sets the value of the "kitchenette" attribute.

- "@Override public void calculatePrice()": Overrides the parent class method to calculate the price of the hotel reservation based on specific rules.

- "@Override public String toString()": Generates a formatted string representation of the HotelReservation object, including the parent class's "toString()" output along with the value of "kitchenette".

**Class Name: HouseReservation**

Class Description/Purpose: Represents a house reservation in the lodging reservation system, providing specific attributes and methods for house reservations.

Class Modifiers: Public

Class Inheritance: Inherits from the "Reservation" class.

Class Attributes:

- "private int numberOfFloors": Indicates the number of floors in the house reservation.

- Exceptions Thrown: None

Class Constructors:

- "public HouseReservation(String accountNumber, String reservationNumber, String lodgingPhysicalAddress, String lodgingMailingAddress, Date startDate, int numberOfNights, int numberOfBeds, int numberOfBedrooms, int numberOfBathrooms, int lodgingSize, int numberOfFloors)": Initializes the house reservation with the provided details.

Class Methods:

- "public int getNumberOfFloors()": Returns the number of floors in the house reservation.

- "public void setNumberOfFloors(int numberOfFloors)": Sets the value of the "numberOfFloors" attribute.

- "@Override public void calculatePrice()": Overrides the parent class method to calculate the price of the house reservation based on specific rules.

- "@Override public String toString()": Generates a formatted string representation of the HouseReservation object, including the parent class's "toString()" output along with the value of "numberOfFloors".

# HUMAN INTERFACE DESIGN

## Overview of User Interface

The Accommodation Reservation System's user interface aims to provide a simple and intuitive experience for users to interact with the system. Users can perform various actions related to managing their accounts and reservations. The main functionalities from the user's perspective include:

### Create Account

Users can create a new account by providing necessary details like mailing address, email address, and phone number.

### View Account Information

Users can view their account information, including contact details and existing reservations.

### Create Reservation

Users can create a new reservation by specifying the lodging details, start date, and other required information.

### Update Account

Users can update their account information, including mailing address, email address, and phone number.

### Update Reservation

Users can modify the details of an existing reservation, such as lodging address, start date, and other information.

### Cancel Reservation

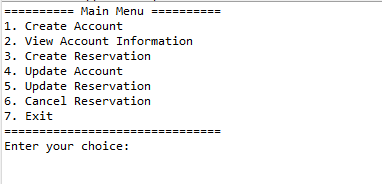
Users can cancel a reservation, and the system will update the reservation status accordingly.

### Exit

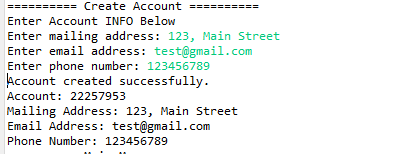
Users can exit the system and end their session.

## Screen Images

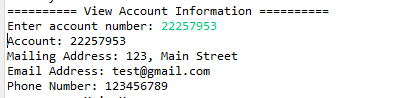
### Main Menu Screen



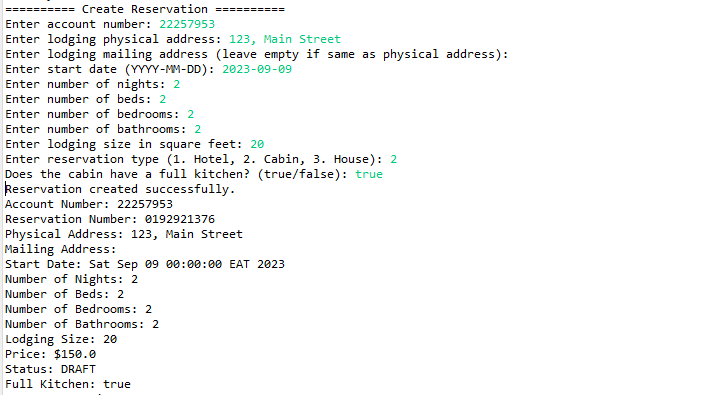
### Create Account Screen



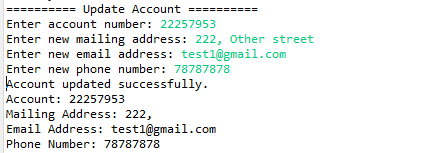
### View Account Information Screen



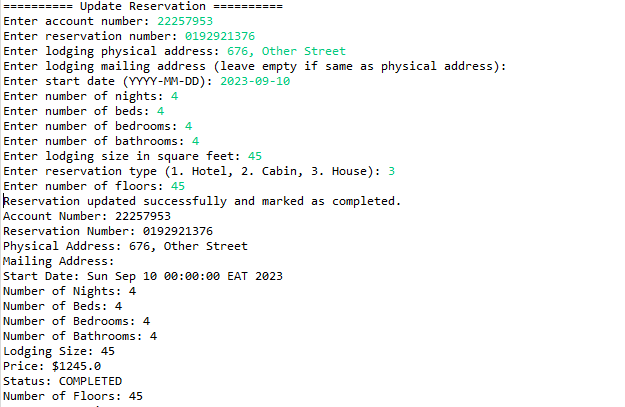
### Create Reservation Screen



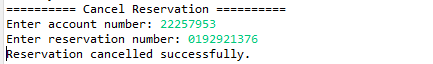
### Update Account Screen



### Update Reservation Screen



### Cancel Reservation Screen



## Screen Objects and Actions

The following screen objects and associated actions will be present in the Accommodation

### Reservation System's User Interface

#### Menu Options

Users can interact with menu options to select the desired functionality (e.g., create account, view account information, etc.).

#### Text Input

Users can input information, such as mailing address, email address, phone number, and reservation details, using text inputs from the console.

#### Status Information

Shows the status of reservations, indicating whether they are in the draft, completed, or cancelled state.

#### Navigation

Provides easy navigation between different sections of the application.

#### Confirmation Messages

Shows confirmation messages when an action, such as creating a reservation or updating an account, is successfully completed.

The screen objects and actions are designed to ensure a user-friendly experience, enabling users to manage their accounts and reservations efficiently and receive feedback on their actions.

# Requirement Matrix

|  |  |  |
| --- | --- | --- |
| **Requirement ID** | **Requirement Description** | **Satisfied By** |
| REQ001 | Create Account | Main.createAccount() method, Account class |
| REQ002 | View Account Information | Main.viewAccountInformation() method, Account class |
| REQ003 | Create Reservation | Main.createReservation() method, Reservation classes (HotelReservation, CabinReservation, HouseReservation) |
| REQ004 | Update Account | Main.updateAccount() method, Account class |
| REQ005 | Update Reservation | Main.updateReservation() method, Reservation classes (HotelReservation, CabinReservation, HouseReservation) |
| REQ006 | Cancel Reservation | Main.cancelReservation() method, Reservation classes (HotelReservation, CabinReservation, HouseReservation) |