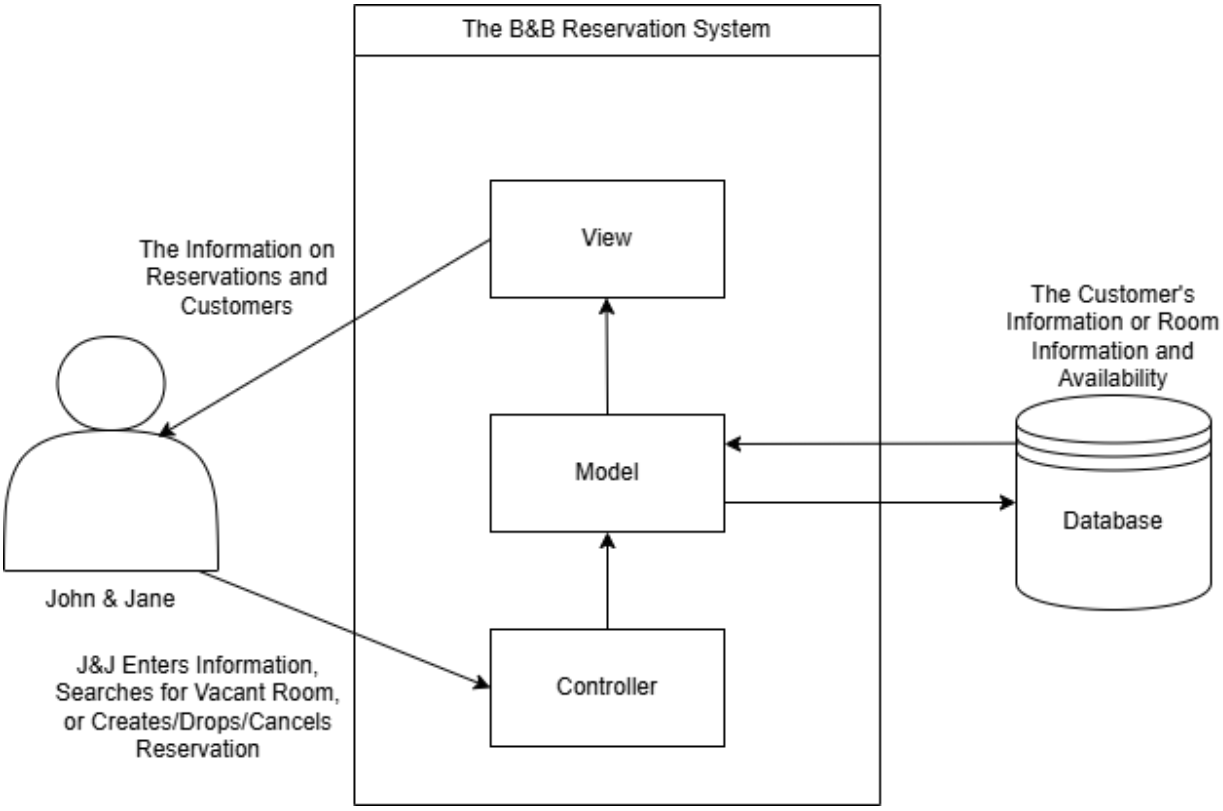
Name: Victoria Lee
Date: 11/9/2024
Class: CMSC 345 7380 Software Engineering Principles and Techniques.
Primary Instructor: John Lee
Week: 4 - Create a Design Model for a Small Bed & Breakfast Reservation System

1)
Rubric Criteria:
Create software system architecture diagram 10%
Your Response:

MVC Design Pattern Diagram:



2)
Rubric Criteria:
Explain approach, steps, and rationale of the software architecture diagram 20%
Your Response:

Approach:

For this software architecture diagram, I used an MVC Design Pattern. This software architecture pattern diagram design (MVC) separates an application into three main components: Model, View, and Controller, making it easier to manage and maintain the codebase (GeeksforGeeks, 2024). And each layer has its duties and dependencies. I decided to use this model as the other patterns were confusing and I wanted to find a pattern that best matches with the use case steps that were given. Since this

project seemed like e-commerce or similar to it, I decided to find the best software architecture design patterns for this scenario. I ended up finding that the MVC design pattern was as close to it as possible. This approach is significant as I wanted to make sure it also matches with the UML class diagram that I previously made. With the UML class diagram, I was able to create the MVC design pattern version of it that best matches the scenario provided. This model approach was helpful with GeeksforGeeks (2024) as they mention that an MVC diagram is best for Complex Applications, Frequent UI Changes, Reusability of Components, and Testing Requirements.

Steps followed as according to GeeksforGeeks (2024):

1. Firstly, I Identified the Users, System, Layer Logic, and Data Management (Database) as the starting parts of the diagram and then I added in the required details.
    a. Users: John and Jane as the system's users.
    b. System: The overall B&B reservation system.
    c. Layer Logic: Dividing the system into MVC components.
    d. Data Management: The database handling the storage and retrieval of information.
2. Next, I defined the responsibilities of each layer (Model, View, and Controller). This is shown in the next steps:
    a. Model: The Model component in the MVC (Model-View-Controller) design pattern demonstrates the data and business logic of an application. It is responsible for managing the application's data, processing business rules, and responding to requests for information from other components, such as the View and the Controller.
        i. This manages and handles reservations, customer details, room availability, and payment processing.
    b. View: Displays the data from the Model to the user and sends user inputs to the Controller. It is passive and does not directly interact with the Model. Instead, it receives data from the Model and sends user inputs to the Controller for processing.
        i. This displays the available rooms, room types, and showing reservation details.
    c. Controller: The controller acts as an intermediary between the Model and the View. It handles user input and updates the Model accordingly and updates the View to reflect changes in the Model. It contains application logic, such as input validation and data transformation.
        i. This processes a new reservation, handles cancellations, or handles dropping as well.
3. I provided more details by showing the responsibility communication between the components and the arrows to show how data and control movement across levels.
    a. For Model communication responsibility steps across the levels:
        i. First, receive the reservation dates from the Controller.
        ii. Then, check the availability of rooms for the requested dates.
        iii. Next, provide available room data to the Controller.
        iv. Then, update reservation data based on the Controller's input.
        v. Lastly, save the reservation details to the database.
    b. For View communication responsibility steps across the levels:
        i. Display the interface for John or Jane to enter reservation dates.
        ii. Show the list of available rooms and their prices during the requested dates.

iii. Collect the guest’s room selection.
iv. Display a form to enter or confirm the guest's name and guarantee payment details.
v. Render the updated UI based on the Controller's updates.

c. For Controller communication responsibility steps across the levels:
i. Acts as an intermediary between the Model and the View.
ii. Handles user input, updates the Model accordingly, and updates the View to reflect changes in the Model.
iii. Contains application logic such as input validation and data transformation.

d. User Interaction with View: John or Jane inputs the reservation start and end dates into the system through the View.
e. Controller Receives User Input: The View forwards the reservation dates to the Controller.
f. Controller Processes User Input: The Controller validates the input dates and sends them to the Model to check room availability.
g. Model Checks Availability: The Model checks the database for available rooms during the requested dates and sends the data back to the Controller.
h. Controller Updates View: The Controller receives the list of available rooms and their prices from the Model. The Controller updates the View to display this information.
i. View Displays Available Rooms: John or Jane informs the guest about the available rooms and prices. The guest selects a room.
j. User Interaction with View: John or Jane inputs the reservation start and end dates into the system through the View.
k. Controller Receives User Input: The View forwards the reservation dates to the Controller.
l. Controller Processes User Input: The Controller validates the input dates and sends them to the Model to check room availability.
m. Model Checks Availability: The Model checks the database for available rooms during the requested dates and sends the data back to the Controller.
n. Controller Updates View: The Controller receives the list of available rooms and their prices from the Model. The Controller updates the View to display this information.
o. View Displays Available Rooms: John or Jane informs the guest about the available rooms and prices. The guest selects a room.

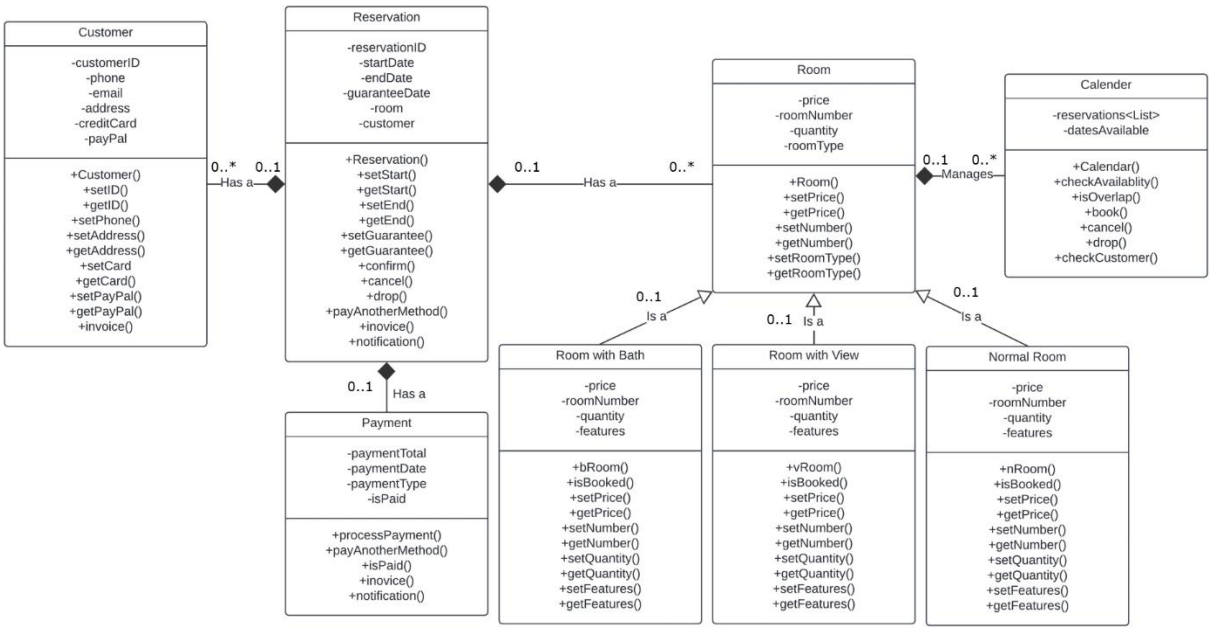
4. Finalize the diagram by double-checking that it matches the scenario and that it syncs with the UML class diagram that was previously made.
5. All steps and logical processes are assisted by GeeksforGeeks (2024).

Rationale:

The system uses a simplistic approach to comprehend and manage the system because of the layered designed. According to GeeksforGeeks (2024), this design pattern specifies that an application consists of a data model, presentation information, and control information. This is beneficial for software engineers as the pattern requires that each of these be separated into different objects making it easier to see what each component is responsible for. The MVC pattern separates the concerns of an application into three distinct components, each responsible for a specific aspect of the application’s

functionality. This separation of concerns makes the application easier to maintain and extend, as changes to one component do not require changes to the other components. This detailed MVC design pattern ensures that the B&B reservation system is well-organized, maintainable, and scalable. It enables modularity, making it easier to change certain layers. It helps developers add new features smoothly, makes testing simpler, and allows for better user interfaces (GeeksforGeeks, 2024). Overall, MVC helps keep everything organized and improves the quality of the software. This design also meets the demands of balancing software quality and efficiency.

3)
Rubric Criteria:
Create detailed UML class diagram 15%
Your Response:



4)
Rubric Criteria:
Explain approach, steps, and rationale of the detailed class diagram model 25%
Your Response:

Approach:

The class diagram has more details for developers and may be somewhat different as it was hard to make it exactly like the Use Case Diagram or the software architecture diagram. I made sure to make it as similar as possible, but if there were some minor changes, it is due to those additional details. I made sure to thoroughly read the scenario and apply it to other template examples online about the UML Use Case Diagrams to help me create it. The approach in creating the UML class diagram for the Bed & Breakfast (B&B) reservation system focuses on modeling the key entities and their relationships within the system. The objective is to provide a clear blueprint of the structure and behavior of the system components, ensuring they work together to support the reservation process (Tsui, Karam, & Bernal,

2014). For this one, the owners can only access and enter information, where the customer can guide the owners so that they can get the reservation and the customer's information into the system. First and foremost, the calendar is the main class of the entire system. Without the calendar, booking the available rooms would be difficult. This means that the calendar manages the system to reserve and book the rooms. The other classes use a has is relationships and room class have an is a relationship. The calendar manages the rooms and the reservations with the customer. The reservations have a relationship with the customer, payment, and the rooms. The rooms have subclasses with an is a relationship to specify the type of rooms the customer wants that are available according to the calendar. In other words, this setup ensures that the Calendar manages the availability and bookings of Rooms, while Reservations reference to Rooms, Customers, and Payment. Meanwhile, Rooms refer to the subclasses to specify the type of room. Finally, the details are added with the attributes, methods, functions, and relationships.

Steps:

1. The first step was to identify the key entities for each class and the class name. I made sure to determine the main entities involved in the system, such as Customer, Reservation, Room, Room types (3 of them), Payment, and Calendar.
2. The second step was to define the attributes and methods: To do this, I consulted the instructions for each object's attributes. I created the customer's attributes and methods as it was described in the scenario given and then did the same for the rest of the classes. The room reservation was straightforward, and the methods took a while to create to match the previous diagram. The calendar object was easy to create as it is the admin class of all of the rest of the classes. Without the calendar object, there is no reservation to make. For each entity, I made sure to identify the relevant attributes (data) and methods (functions) that describe the entity's state and behavior. The methods and functions were added after the attributes to match with the UML use case model given. I tried my best to make the methods and functions as close as possible to the UML use case model to make it accurate.
3. The third step was to establish the relationships. I determined the relationships between entities, such as associations, aggregations, compositions, and generalizations, and could optionally include the multiplicities where 0..1 means at least 0 and at most 1; 0..* means at least 0 at most infinity.
4. Finally, the last step was to finalize the UML Class Diagram. To do this, I made sure to review the diagram to ensure that I accurately represented the requirements based on the scenario given. In addition, I used UML notation to visually represent the classes, attributes, methods, and relationships.

Rationale:

The UML class diagram is different than the UML use case. It allowed me to see the layout of the system's development through attributes, entities, methods/functions, and objects. The class diagram includes all essential entities and their attributes and methods, providing a detailed overview of the system's structure. This allowed me to get a deeper understanding of the system through the classes, modification, and testing. This also allowed me to see how the system functions for this scenario with the B&B reservation system. Since I defined the relationships between entities, the diagram clarifies how different parts of the system interact and depend on each other. Using generalization (inheritance)

to model Room types such as Room with Bath, Room with View, and Normal Room, allows for code reusability and a noticeable hierarchy. Although not as simplistic as the UML use case model, this diagram helps create a blueprint of the goals that may or may not work based on the previous model. This can be used in conjunction with the use case diagram for the programming requirements. The class diagram provides the foundation for implementation, making the system easier to develop, scale, and maintain (Tsui, Karam, & Bernal, 2014). The clear definitions of classes and relationships help developers understand the system and make changes when needed (Tsui, Karam, & Bernal, 2014). This is quite useful for programming and most programmers utilize a UML class diagram or package. By following this approach, the UML class diagram provides a clear and detailed representation of the B&B reservation system and is clear for programmers (developers), ensuring that all components are defined and interact to support the overall functionality (Tsui, Karam, & Bernal, 2014). This helps John and Jane manage their business efficiently for the B&B reservation so that it will be a good reservation experience for their customers.

5)
Rubric Criteria:
Create user interface mockup 10%
Your Response:

Step 1: Call John & Jane's B&B Reservation.

Website: https://www.JJsBed&Breakfast.com/reservation

JJ's Bed & Breakfast | Home | About | Reservation | Rooms | Other |

Reservation

Hello! This is John and Jane with the B&B located in a small New England town.

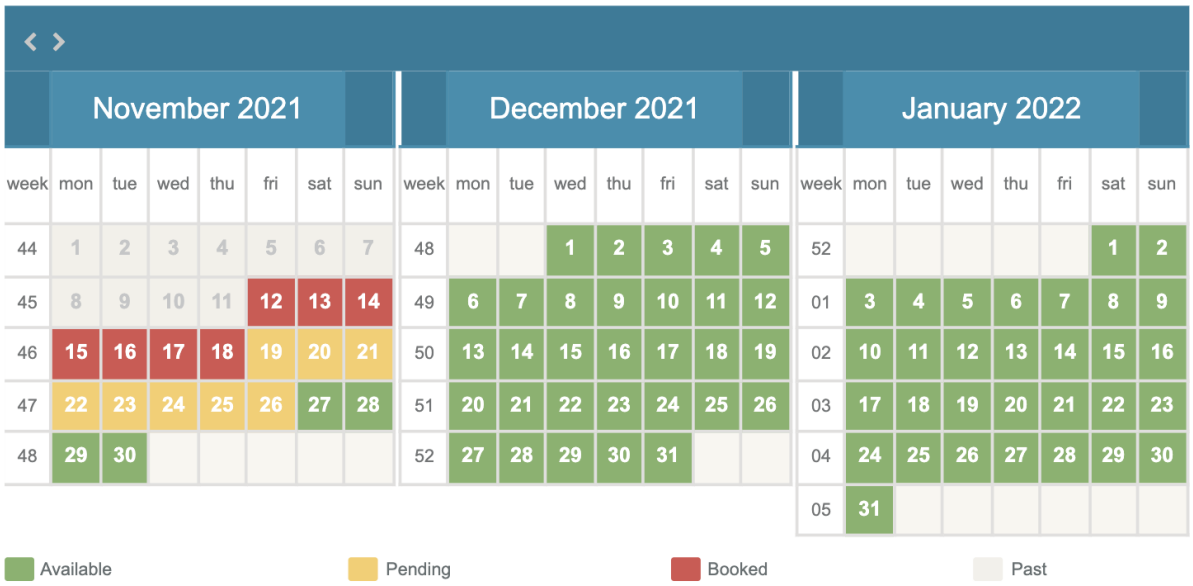
To make a reservation in this B&B, please call the number below!

This will help us book your reservation and guide you to fill out some information as well.

Phone: (123)-456-7890

Thank you!

Step 2: Check Callender Availability Overview



November 2021

week	mon	tue	wed	thu	fri	sat	sun
44	1	2	3	4	5	6	7
45	8	9	10	11	12	13	14
46	15	16	17	18	19	20	21
47	22	23	24	25	26	27	28
48	29	30					

December 2021

week	mon	tue	wed	thu	fri	sat	sun
48			1	2	3	4	5
49	6	7	8	9	10	11	12
50	13	14	15	16	17	18	19
51	20	21	22	23	24	25	26
52	27	28	29	30	31		

January 2022

week	mon	tue	wed	thu	fri	sat	sun
52						1	2
01	3	4	5	6	7	8	9
02	10	11	12	13	14	15	16
03	17	18	19	20	21	22	23
04	24	25	26	27	28	29	30
05	31						

Available Pending Booked Past

Figure 1: Callender availability created by PHPJabbers. (n.d.).

Step 3: Enter Reservation Dates

Enter Reservation Dates

Start: _____

End: _____

Check Availability:	

(Please enter the Start and End dates and Press Check Availability.)

(If available, it would pop up true/yes, if not false/no.)

Step 3: Select Room

Room with Bath Price: $150	Room with View Price: $100	Normal Room Price: $80

Select the Room Type by Clicking it.

Step 4: Reservation Details, Information, and Confirmation

Reservation Information Details

Room Selected:
Room with Bath //Room with Bath Image// Price: $150 All Features Listed

Dates:
Start: 11/27/2021
End: 12/3/2021

Tax Rate:
12.2%
Tax Price:
12.2% * $150 = $18.30

Total Price (w/ Tax Rate):
$1,050.00 + $18.30 = $1,068.30

Customer Information:	
Full Name:	Firstname Lastname
Address:	1234 England St.
Phone Number:	(234)-567-8901
Email:	Customer123@gmail.com
Zip Code:	12345

Debit/Credit Card Information (Payment Method):	
Card Number:	1234-5678-9012-3456
Expiration Date:	12/24
Security Code:	123
PayPal:	If PayPal, click on PayPal Link to sign in and retrieve information automatically.
Other Payment:	If other, select other payment options
	Create Reservation

1. Verify room information, customer information, reservation information, dates, and price. (If it is not in the system, include it in the system).
2. Enter/auto-populate customer information. (If the customer is already in the system, then retrieve the information. If not, then add the customer to the system).
3. Enter Customer Card Information
4. Click Create Reservation Button to Confirm (The system should verify and then notify the user)

Step 5: Reservation Agreement, Policies, and Confirmation Receipt

Policies and Agreement Sent to Customer | JJ's B&B Rules and Regulations created by ():

1. **Check-In and Check-Out:**
    I. Check-in and Check-out time opens at 9:00 A.M. Early check-ins and late check-outs are subject to availability and may incur additional charges.
2. **Reservation and Cancellation/ Dropping Policy:**
    I. A valid government-issued photo ID and a credit card are required at check-in and must be present. All Guests must adhere to the cancellation/drop policy specified at the time of booking to avoid charges. This policy may vary based on the type of reservation.
    II. Reservations must be guaranteed by 1 day's payment.
    III. Reservations will be held without guarantee for an agreed-upon time. If not guaranteed by that date, the reservation will be dropped.
3. **Payment:**
    I. All major credit cards are accepted. Cash payments may require a valid credit card on file. Payment for the entire stay, including taxes and fees, is required at check-in.
4. **Age Requirement:**
    I. Guests must be at least 18 years old to check in without a parent or legal guardian.
5. **Occupancy Limits:**
    I. The maximum occupancy for each room type is 3 or 4 people depending on the room type booked. Additional charges may apply for extra guests.
6. **Quiet Hours:**
    I. Quiet hours are observed from 10:00 P.M. to 7:00 A.M. Please be considerate of other guests during this period.
7. **No Smoking Policy:**
    I. Smoking is strictly prohibited in all indoor areas. Designated smoking areas are provided. A cleaning fee will be charged for smoking in non-smoking areas.

8. **Pet Policy:**
    I. Pets are not allowed unless explicitly stated otherwise. Service animals are exempt from this policy.
9. **Damages and Theft:**
    I. Guests are responsible for any damage to the property or its contents. Missing items will be charged as the replacement cost.
10. **Guest Conduct:**
    I. Disruptive or unlawful behavior will not be tolerated. Guests may be asked to leave without a refund if their conduct is deemed unacceptable.
11. **Facility Access:**
    I. Certain areas or amenities may have restricted access or specific hours. Please adhere to posted signs and guidelines.
12. **Lost and Found:**
    I. [Your Establishment Name] is not responsible for lost or stolen items. Please secure valuables in the provided safe, and report any lost items to the front desk.
13. **Emergency Procedures:**
    I. Familiarize yourself with emergency exits and procedures. In case of an emergency, contact the front desk or dial 911.
14. **Compliance with Laws:**
    I. Guests must comply with all local, state, and federal laws. Any illegal activities will be reported to the authorities.

By staying with us, you agree to abide by these rules and regulations. JJ's B&B reserves the right to refuse service or evict guests who violate these policies.

Confirmation Message

Reservation has been successfully made for [Customer Name], on [Dates], [Room Number], for [Price].

Thank you for choosing JJ's B&B! We wish you a pleasant stay!

Print/Email Details

Print	✓	Email	✓

Done

Check box which one to send the receipt and Confirm with Done.

Step 6: Cancelling / Dropping the Reservation (For certain scenarios)

Use the System to find if the user has breached the reservation policy:

Reservation Policy Breaches/Changes Check:

Produce List: Yes/No

Check Guarantee:	

(Enter the yes/no to allow it to produce a list of the possible reservation guarantee breaches or changes. Press Check Guarantee.)

It will produce all the Guarantee in a list and show up on the calendar. The red will signify a breach (false) and the yellow is a change (false). Green means guarantee is true.

Check the calendar to select the dates that have the customer's bookings to drop/cancel.

November 2021

week	mon	tue	wed	thu	fri	sat	sun
44	1	2	3	4	5	6	7
45	8	9	10	11	12	13	14
46	15	16	17	18	19	20	21
47	22	23	24	25	26	27	28
48	29	30					

December 2021

week	mon	tue	wed	thu	fri	sat	sun
48			1	2	3	4	5
49	6	7	8	9	10	11	12
50	13	14	15	16	17	18	19
51	20	21	22	23	24	25	26
52	27	28	29	30	31		

January 2022

week	mon	tue	wed	thu	fri	sat	sun
52						1	2
01	3	4	5	6	7	8	9
02	10	11	12	13	14	15	16
03	17	18	19	20	21	22	23
04	24	25	26	27	28	29	30
05	31						

Figure 1: Callender availability created by PHPJabbers. (n.d.).

(The calendar should list out the possible customer's names on that date and times specifically that have a guarantee true, changes false, or breaches false.)

Select the customer's name and information in the database.

Query Result

SQL | All Rows Fetched: 4 in 0.003 seconc

	CUSTOMERUSERID	FIRSTNAME	LASTNAME
1	1	Katarina	Alves
2	2	Linda	Sobieska
3	3	Negan	Smith
4	4	Nina	Williams

Cancel/Drop the Reservation after selecting from the database list

Customer selected: _____

(The customer's name is here and all reservation information is listed below)

Customer and Reservation Information here:

Cancel/Drop:_____

Receipt:_____

Receipt email/print: _____

(a box will pop up to select one option: cancel/drop and yes/no for receipt)

(If email, the user will have to fill out the email or select print. The options will not matter if the user selects “no” in the previous “Receipt” option).

Confirm Changes

(Selecting and clicking on Confirm Changes cancels/drops the reservation)

6)
Rubric Criteria:
Explain approach, steps, and rationale of the user interface mockup 10%
Your Response:

Approach:

The approach to designing the user interface mockup focuses on simplicity and a clear layout to ensure that John and Jane can easily manage reservations, view availability, and process payments. This approach I decided to do was based on the UML Class diagram and the Use case provided with the Use case in the previous assignment. I also used a similar base of the Payroll System UI mockup that I found online to start off how I would build the UI mockup for this system (Creately, n.d.). According to Nulab (n.d.), a UI mockup is a full-sized model of a design or a device used for product presentations or demonstrations. Mockups are static, high-fidelity designs that show how the product will look, while prototypes are interactive, allowing users to experience how the product will work (Nulab, n.d.). The UI needs to support John and Jane's workflow effectively, ensuring smooth handling of reservations, customer information, and payment processing. This allowed me to develop the closest user interface mockup for John and Jane’s B&B reservation system, ensuring that all necessary information is easily accessible to them. The design aims to make the reservation process organized while providing a clear and intuitive user experience. In my previous use case, since it was like the requirements that were provided, I decided to use the previous use case for more details on how the cancellation and drop

process works. The additional details are as follows that I made sure to include: If the customer calls before the reservation date to cancel, the owner proceeds to cancel the reservation through the system. The customer calls the owner before the reservation date or time that was set up. The customer requests a cancellation and provides the reservation information. The owner finds the reservation in the calendar and deletes the reservation. The deletion of the reservation alerts the system to send a notification via digital communication methods such as email, phone, or mail. The room becomes available and open to reservations for the deleted dates or times. If the customer does not call before or fails to pay before the day, the owner may proceed to drop the reservation or charge a late fee through the system. The owner checks if the initial invoice has not been paid a day before the reservation date. The owner looks at the calendar, and the system shows that the reservation Boolean (T=Did Pay, F=Did not pay) is false. The owner selects the reservation and the options of either 1) paying through another method or 2) dropping the reservation. The drop alerts the system to send a notification through digital communication methods such as email, phone, or mail. The room becomes available and open to reservations for the dropped dates or times. Through this approach, it included the cancellation and the drop steps for the UI mockup design. I also made sure to include all the requirements of the use case given. This approach is to make sure that it follows that John and Jane will have four bedrooms for guests with varying features and prices. When a potential customer calls for a reservation, they will check the calendar, and if there is a vacancy, they will enter the customer's name, address, phone number, reservation dates, room price, credit card number, and room numbers. Following this approach of both different types of use cases, it allowed me to build the UI mockup. The UML class diagram also helped me sort out what I needed to include (attributes, guarantee, etc) in the UI mockup design.

Steps are followed according to Creately (n.d.), Nulab (n.d.), and Tsui, Karam, & Bernal (2014), it mentions to do these steps:

1. Firstly, I went to identify all the User's needs. This includes understanding the tasks John and Jane will perform, managing reservations, checking availability, and processing payments. In other words, it means to identify the main components and user flow.
    a. Main Components: Calendar for checking room availability; Room listings and details; Customer information form; Payment processing form; Confirmation and notification system
    b. User flow: Receive call → Check availability → Inform customer → Manage room listings → Enter customer information → Confirm reservation → Process payment → Send confirmation. (Exceptions to this method is the cancellation and dropping which is different and is mentioned previously how I tackled it).
2. Secondly, I designed the wireframes and mockup layout of the UI. This involves creating basic wireframes/mockup to map out the layout of the UI, focusing on key areas like the reservation form, calendar, and navigation.
    a. Main Screen: Header with navigation (Home, Reservations, Calendar, Reports, Settings); Main section with reservation form and calendar view.
    b. Reservation Form: Input fields for start and end dates; Dropdown or list for room selection; Customer information form.
    c. Calendar View: Display available and booked dates; Highlight vacancy and booked rooms.

3. Thirdly, I developed a detailed Mockups about the registration process, information, database, system, payment, and other tasks. This involves adding details to the wireframes/mockup, including input fields, buttons, and navigation elements. I made sure that the UI is intuitive and follows a logical flow from one task to the next.
    a. Main Screen: Header with navigation (Home, Reservations, Calendar, Reports, Settings); Main section with reservation form and calendar view.
    b. Reservation Form: Input fields for start and end dates; Dropdown or list for room selection; Customer information form.
    c. Calendar View: Display available and booked dates; Highlight vacancy and booked rooms.
4. Fourthly, I incorporated some changes and reviewing the final design based on the the use cases. This includes refining the design, ensuring it meets their specific needs and preferences. It also means implementing the final design after testing to ensure the UI is user-friendly and efficient.
    a. For instance, I made sure to include the drop and cancellation, but that was after I realized I needed to create it when testing out the final results.

Rationale:

It is crucial to comprehend UI mockups to build this B&B mockup version so that it is user-friendly and efficient. When working with mockups, I learned to tackle with a mid to high-fidelity visualization of your design, usually showcasing the layout structure, the color schemes, typography, images, and the overall navigation interface of the website or app (Nulab, n.d.). This taught me to learn how these are essential for testing, visualization, and design approval. By creating the design this way, it keeps the interface clean and straightforward, with essential elements easily accessible to John and Jane. The form-based layout ensures that all necessary information is collected in a logical sequence, minimizing the chance of errors. According to Tsui, Karam, & Bernal (2014), Quick access to key functions like checking for availability, managing room listings, processing reservations, and confirming reservations helps make the process efficient. When included the navigation bar allows easy access to other system functions, like viewing the calendar and generating reports, making the system versatile and comprehensive. By focusing on the specific needs of John and Jane, the UI mockup design ensures they can manage their B&B reservations effectively, even without extensive technical skills (Tsui, Karam, & Bernal, 2014). The struggles I had were mostly trying to make it as close as possible to the use cases and the UML class diagram. I made sure to follow the requirements and make sense. Overall, this mockup ensures that all necessary steps in the reservation process are covered.

7)
Rubric Criteria:
Reflect on the learning experience and lessons learned 10%
Your Response:

Creating the software system architecture diagram, class diagram, and user interface mockup diagrams for the Bed & Breakfast (B&B) reservation system involved understanding the requirements and translati ng them into structured, visual representations. The process required a detailed analysis of the business needs and a clear understanding of the interactions between different system components. For the first part of the software system architecture diagram, I decided to use an MVC design pattern. According to

GeeksforGeeks (2024), the MVC (Model-View-Controller) design pattern breaks an application into three parts: the Model (which handles data), the View (which is what users see), and the Controller (which connects the two). This makes it easier to work on each part separately, add new features smoothly, make testing simpler, and allow for better user interfaces. MVC helps keep everything organized and improves the quality of the software. This is beneficial and important for software engineers as it is good for apps with many features and user interactions, like e-commerce sites. This MVC design pattern also helps organize code and manage complexity and MVC allows changes to the View without affecting the underlying logic for the UI process. The fact that this design is simplistic also taught me that it is reusable to use in other parts like in other projects, due to MVC's modular structure. After building the diagram, it is so flexible that it can be tested easily to compare with other diagrams ensuring that it is adaptable to different types of projects. This allowed me to test each component separately for better quality control (GeeksforGeeks, 2024). Secondly, the project allowed me to experience the UML class diagram stage process. I had to alter and modify the class diagram from the previous assignment, which had to somewhat synchronize with the newer diagrams asked to draw. The relationships helped me pay attention to how classes interacted with each other. Positioning the subclasses and superclasses reminded me of the importance and dynamics of inheritance. According to GeeksforGeeks (2024), I learned that the UML class diagram is a visual tool that represents the structure of a system by showing its classes, attributes, methods, and the relationships between them. By learning to create the UML diagram, it taught me to understand how the system is organized and how its components interact. Class diagrams provide a high-level overview of a system's design, helping to communicate and document the structure of the software. Class diagrams represent the system's classes, attributes, methods, and relationships, providing a clear view of its architecture. They are a fundamental tool in object-oriented design and play a crucial role in the software development lifecycle. I learned that this UML diagram can appropriately depict various aspects of the OOPs concept. It also demonstrates that a proper design and analysis of applications can be faster and more efficient. Learning this helped me comprehend the base for deployment and component diagrams. It also taught me to learn forward and reverse engineering. I learned that after creating the UML class, it allowed me to start building the layout of the other two diagrams (software system architecture and user interface mockup design). According to Tsui, Karam, & Bernal (2014), I learned that the UI mockup design is significant in programming as it can help designers create visual representations of a website or app's user interface (UI) before the final product is developed. This allowed me to comprehend the steps of creating a UI mockup and how it benefits programmers in software engineering. Using this method, it taught me to Visualize the final product as mockups show how different elements work together in the layout, including color, typography, icons, and more. I also taught me to test design ideas as mockups allow designers to try out different ideas and see what works best. Lastly, the UI mockup taught me to ensure that the design meets expectations and meets the user's needs. Then once designed and reviewed, I learned that the next the design process is usually to create a prototype (Tsui, Karam, & Bernal, 2014). These diagrams provide a high-level view of the system's functionality and ensure it meets the requirements. Writing detailed use case scenarios provides a step-by-step guide on how the system should behave, helping to ensure that all functional requirements are met. Through research, some situations to utilize this are when the user wants to comprehend the interactions with the system, the early stages of requirements gathering to identify key functionalities and involving stakeholders in the requirements process. Moreover, creating the UML class diagram involved identifying the key entities and their attributes and methods (GeeksforGeeks, 2024). This placed the importance of accurately

modeling real-world entities to ensure the system's data structure is reflective of actual business processes. Specifically, I made sure to start off with the UML Class diagram as previously taught and used. Then, I implemented it again to this project with modifications to sync with the system architecture diagram. Next, I had to create the UI mockup after the two diagrams were created. This allowed me to pay attention to how things came together for the last diagram. However, the difference was that the UML class diagram and the clients' specifications were easier to implement than the UI which was daunting at first. According to Tsui, Karam, & Bernal (2014), to overcome this obstacle, I had to follow the hierarchy from the previous step which allowed me to establish the elements all together. I made sure to make the UI as similar as possible to the other two diagrams to match the scenario. This taught me how to notice the difference between the UI and the software system architecture. I had to continue adding on what I learned previously and further expand in different branches like with the other two newer diagrams I had to create. This allowed me to focus on the relationships each element had to one another and comprehend the intersections and differences alike. I learned that there are a variety of different preparations that go into software development. One of the most valuable lessons learned is the importance of iteration and refinement in the modeling process. There were moments when I made each part either the class or the use case that I realized I needed to improve on and keep on adding additional changes. This made me realize how important planning and formatting is in software development. Establishing relationships between entities helps in understanding how the components of the system interact (Tsui, Karam, & Bernal, 2014). The designing of diagrams and refining the previous diagrams taught me the significance of why software engineering utilizes them (GeeksforGeeks, 2024). These lessons helped me learn to catch errors early and ensure the final design is efficient. This makes the system easier to scale and maintain through the clear definitions of classes and relationships. Overall, creating the diagrams and design for the B&B reservation helped me reinforce the importance of clear and detailed modeling by understanding the business requirements. This experience is beneficial in future software development projects.

References

Creately. (n.d.). *Payroll system UI mockup example*. Creately Diagrams. https://creately.com/diagram/example/ipt6bcpc2/payroll-system-ui-mockup-example

GeeksforGeeks. (2024). *UML class diagram notation*. GeeksforGeeks. https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/#uml-class-notation

GeeksforGeeks. (2024). MVC design Pattern. GeeksforGeeks. https://www.geeksforgeeks.org/mvc-design-pattern/

Nulab. (n.d.). *What are mockups in design and UX?*. Nulab Learn. https://nulab.com/learn/design-and-ux/mockups/

PHPJabbers. (n.d.). *Rental property booking calendar*. Phpjabbers. https://www.phpjabbers.com/rental-property-booking-calendar/

Tsui, F., Karam, O., & Bernal, B. (2014). *Essentials of software engineering* (3rd ed.). Jones and Bartlett Le arning. https://library-books24x7-com.ezproxy.umgc.edu/toc.aspx?site=VGX8U&bookID=51648