50.003 : Elements of Software Construction

C4G1 - Project Meeting 3 Report

Name	Student ID
Ong Jung Yi	1006655
Luvyn Sequeira	1007081
Stavya Sharma	1006933
Zhuang Yang Kun	1006933
Yee Jia Zhen	1006969
Lee Jun Hui Ryan	1006652
Luong Hung	1007160

Task Distribution

Tasks	Group Members
Frontend (Figma, react implementation, design changes)	Stavya, Yang Kun, Luvyn
Backend (SQL query, API calls, routing, controller, models)	Jung Yi, Ryan, Jia Zhen, Hung
Testing (Unit Testing, Integration Testing, Test Cases, Graph)	Jung Yi, Ryan, Jia Zhen, Hung, Stavya, Yang Kun, Luvyn

Changes made since PM2

Feedback	Changes
Sequence Diagram missing reply message	Added reply message
Domain Class Diagram missing	Added Domain Class Diagram
Use case names “Manage Booking” and “Manage Account” vague and unclear	Changed to “Update Account Details” and “View Booking” (clearer action taken)

Features Implemented

Front-end

1. **Landing Page**: Initial page users see when they visit the website.
2. **Hotel Search Page**: Allows users to search for hotels based on various criteria.
3. **Booking Page**: Page where users can book selected hotels.
4. **Payment Page**: Facilitates the payment process for hotel bookings.
5. **Hotel Room Details**: (Not yet implemented)

Back-end

1. **Search Models Implementation**:
   - Search based on destination, number of guests, check-in, and check-out dates.
   - Display a list of hotels filtered by these criteria.
2. **Hotel Listing by Destination**:
   - Use multiple parameters to refine the search results.
   - Fetch hotel information, including prices and additional parameter filters.
     i. Filters for ratings, price, amenities not completed yet. Search is only constrained by the number of guests, dates, and destination currently.
3. **Second Phase**:
   - Plan to introduce additional filters for a more refined search experience.

SQL Querying

1. **Controller Implementation**:
   - Integrates SQL queries to handle various search and filter functionalities.
   - Not yet integrated into the system.

Frontend Unit Testing

Unit Testing for Search Hotel Use Case

Unit testing for src/HotelSearchPage/filterSection.js and src/HotelSearchPage/filterSection.css

Test Case ID	Description	Precondition	Input	Expected Output	Post Condition	Function/Module	Log
TC_FC_001	Check if 'Filter By' heading is rendered	FilterSection component is available	Render FilterSection component	'Filter By' heading should be in the document	None	FilterSection component	Pass (Time: 31 ms) - The 'Filter By' heading was rendered successfully.
TC_FC_002	Check if 'Total Price' label is rendered	FilterSection component is available	Render FilterSection component	'Total Price' label should be in the document	None	FilterSection	Pass (Time: 6 ms) - The 'Total Price' label was rendered successfully.
TC_FC_003	Check if 'Max' and 'Min' input fields are rendered	FilterSection component is available	Render FilterSection component	'Max' and 'Min' input fields should be in the document	None	FilterSection	Pass (Time: 9 ms) - The 'Max' and 'Min' input fields were rendered successfully.
TC_FC_004	Check if 'Star Rating' label is rendered	FilterSection component is available	Render FilterSection component	'Star Rating' label should be in the document	None	FilterSection	Pass (Time: 5 ms) - The 'Star Rating' label was rendered successfully.
TC_FC_005	Check if star rating buttons (1 to 5) are rendered	FilterSection component is available	Render FilterSection component	Star rating buttons (1 to 5) should be	None	FilterSection	Pass (Time: 7 ms) - The star rating buttons (1 to 5) were

				in the document			rendered successfully.
TC_FC_006	Check if 'Guest Rating' label is rendered	FilterSection component is available	Render FilterSection component	'Guest Rating' label should be in the document	None	FilterSection	Pass (Time: 6 ms) - The 'Guest Rating' label was rendered successfully.
TC_FC_007	Check if guest rating radio buttons ('Any', 'Wonderful 9+', 'Very Good 8+', 'Good 7+') are rendered	FilterSection component is available	Render FilterSection component	Guest rating radio buttons ('Any', 'Wonderful 9+', 'Very Good 8+', 'Good 7+') should be in the document	None	FilterSection	Pass (Time: 8 ms) - The guest rating radio buttons ('Any', 'Wonderful 9+', 'Very Good 8+', 'Good 7+') were rendered successfully.
TC_FC_008	Check if the correct guest rating radio button is selected	FilterSection component is available	Click on 'Any' and 'Wonderful 9+' radio buttons	'Any' and 'Wonderful 9+' radio buttons should be checked respectively	None	FilterSection	Pass (Time: 10 ms) - The correct guest rating radio button ('Any' and 'Wonderful 9+') was selected successfully.

Unit Testing for destinationSearch.js

Test Case ID	Description	Precondition	Input	Expected Output	Post Condition	Function/Module	Log
TC001	Test input field renders correctly	Component is mounted	None	Input field with placeholder 'Where to?' is rendered	Input field is visible	DestinationSearch	Input field rendered successfully
TC002	Test debounced search functionality	Component is mounted	User types 'Par' in the input field	Search function is called with 'Par' after debounce	Search results are displayed	DestinationSearch	Search function debounced successfully
TC003	Test suggestion list renders correctly	User has typed a query	User focuses on the input field	Suggestion list is rendered with filtered destinations	Suggestion list is visible	DestinationSearch	Suggestion list rendered successfully
TC004	Test suggestion click functionality	Suggestions are displayed	User clicks on a suggestion	Query is updated with the clicked suggestion and suggestions are hidden	Input field contains clicked suggestion	DestinationSearch	Query updated and suggestions hidden successfully

Unit Testing for pageHeader.js

Test Case ID	Description	Precondition	Input	Expected Output	Post Condition	Function/ Module	Log
TC001	Test header renders correctly	Component is mounted	None	Header element is rendered with a list of items	Header is visible	PageHeader	Header rendered successfully
TC002	Test logo renders correctly	Component is mounted	None	Logo image is rendered with correct src	Logo is visible	PageHeader	Logo rendered successfully
TC003	Test Sign In link functionality	Component is mounted	User clicks on Sign In link	User is navigated to #signin	User is on the sign-in section	PageHeader	Sign In link navigated successfully
TC004	Test Contact Us link functionality	Component is mounted	User clicks on Contact Us link	User is navigated to #contactus	User is on the contact us section	PageHeader	Contact Us link navigated successfully
TC005	Test currency button functionality	Component is mounted	User clicks on SGD button	Currency dropdown is displayed	Currency dropdown is visible	PageHeader	Currency button clicked successfully

Unit testing for dateInput.js

Test Case ID	Description	Precondition	Input	Expected Output	Post Condition	Function/Module	Log

TC001	Test DateRangePicker renders correctly	Component is mounted	None	DateRangePicker is rendered with placeholders 'Check-in' and 'Check-out'	DateRangePicker is visible	DateInput	DateRangePicker rendered successfully
TC002	Test date range selection functionality	Component is mounted	User selects a date range	setDateRange function is called with the selected date range	Date range is updated	DateInput	Date range selected successfully
TC003	Test renderInput function	Component is mounted	None	Two TextFields are rendered with the selected dates	Two TextFields are visible	DateInput	TextFields rendered successfully
TC004	Test start date input field	Component is mounted	User enters a start date	TextField for start date is updated with the entered date	Start date field contains entered date	DateInput	Start date input updated successfully
TC005	Test end date input field	Component is mounted	User enters an end date	TextField for end date is updated with the entered date	End date field contains entered date	DateInput	End date input updated successfully
TC006	Test localization provider	Component is	None	LocalizationProvider uses AdapterDayjs	Dates are handled	DateInput	LocalizationProvider

		mounted		for date handling	using Dayjs adapter		initialized successfully

Backend Unit Testing (Search Hotel - HBS_UC_1)

Unit Testing for *controller/searchHotelController.js:* This tests are meant for input validation, Hotel retrieval and filtering as well as the implemented caching mechanism

Test Case ID	*Description*	*Precondition*	*Input*	*Expected Output*	*Post Condition*	*Function/Module*	*Log*
TC_UT_001	Validate retrieval of hotels by destination	DestinationCache is empty	req.params.id = '123', req.body = { checkin: '2024-11-01', checkout: '2024-11-10' }	List of hotels and their prices for the given destination ID	Destination '123' cached	searchHotelByDestination	Destination 123 cached
TC_UT_002	Ensure caching mechanism for destination hotels	DestinationCache contains hotels for id '123'	req.params.id = '123', req.body = { checkin: '2024-11-01', checkout: '2024-11-10' }	List of hotels and their prices retrieved from cache	Cache hit for destination '123'	getCacheHotelsByDestination	Cache accessed: destination 123
TC_UT_003	Validate filtering by star rating	HotelPrices and Hotels Maps are populated	filters = { starRatingFloor: 4 }	List of hotels with a star rating of 4 or higher	Filter applied and valid hotels returned	filterHotels	Filter applied successfully
TC_UT_004	Validate filtering by price range	HotelPrices and Hotels	filters = { priceFloor: 100,	List of hotels with prices within the	Filter applied and valid	filterHotels	Filter applied successfully

		Maps are populated	priceCeil: 200 }	specified range	hotels returned		
TC_UT_005	Validate retrieval of hotel details by ID	Hotel ID is valid and room prices are available	req.params.id = '123', req.body = { destination_id: '456', checkin: '2024-11-01' }	Hotel details and room prices for the given hotel ID	Hotel details and room prices retrieved	searchHotelById	Hotel details retrieved
TC_UT_006	Ensure error handling for missing mandatory fields in hotel search	None	req.params.id = '', req.body = { checkin: '', checkout: '' }	Error message indicating missing mandatory fields	Error message displayed	searchHotelByDestination	Error handled correctly
TC_UT_007	Validate the filtering placeholder function	HotelPrices and Hotels Maps are populated	filters = {}	Unfiltered list of hotels and their prices	Placeholder filter applied successfully	placeholderFilterHotels	Placeholder filter applied

Unit testing for *models/hotel.js* are meant to test the fetchHotelsByDestination and fetchHotel functions.

Test Case ID	*Description*	*Precondition*	*Input*	*Expected Output*	*Post Condition*	*Function*	*Log*
TC_UT_008	Test fetchHotelsByDestination() returns map of hotel instances	None	destination_id: RsBU	Map of hotel instances identified by its corresponding hotel id	Hotels by destination fetched successfully	fetchHotelsByDestination	Pass
TC_UT_009	Test fetchHotel() returns Hotel instance	None	id: 'diH7'	Hotel instance for the given ID	Hotel fetched successfully	fetchHotel	Pass

TC_UT_010	Ensure error handling when fetching hotels by destination failed	None	destination_id: 'Invalid'	Error thrown and logged	Error handled correctly	fetchHotelsByDestination	Pass
TC_UT_011	Ensure error handling when fetching a hotel by hotel id failed	None	id: 'Invalid'	Error thrown and logged	Error handled correctly	fetchHotel	Pass

Unit testing for the *models/hotelPrices.js* are meant to test the fetchHotelPricesByDestination function.

Test Case ID	*Description*	*Pre-condition*	*Input*	*Expected Output*	*Post Condition*	*Function/Module*	*Log*
TC_UT_012	Validate fetching hotel prices with successful API response	API response is valid and complete	destination_id: 'D123', checkin: '2024-11-01', checkout: '2024-11-10', lang: 'en_US', currency: 'USD', guests: '2'	Map of hotel prices with HotelPrice instances	Hotel prices fetched successfully	fetchHotelPricesByDestination	
TC_UT_013	Validate handling of incomplete API response with polling	API response is incomplete initially	destination_id: 'D123', checkin: '2024-11-01', checkout: '2024-11-10', lang: 'en_US', currency: 'USD', guests: '2'	Map of hotel prices after polling completes	API polling handled correctly	fetchHotelPricesByDestination	
TC_UT_014	Ensure error handling	API request fails	destination_id: 'D123', checkin:	Error thrown and logged	Error handled correctly	fetchHotelPricesByDestination	Error fetching hotel prices

	when API fails		'2024-11-01', checkout: '2024-11-10', lang: 'en_US', currency: 'USD', guests: '2'				
TC_UT_015	Validate handling of exceeded API poll limit	API response never completes	destination_id: 'D123', checkin: '2024-11-01', checkout: '2024-11-10', lang: 'en_US', currency: 'USD', guests: '2'	Error indicating poll limit exceeded	Error handled correctly	fetchHotelPricesByDestination	

Unit testing of the *models/hotel.js* are meant to test the fetchHotelsByDestination and fetchHotel functions.

Test Case ID	*Description*	*Pre condition*	*Input*	*Expected Output*	*Post Condition*	*Function/Module*	*Log*
TC_UT_016	Validate fetching of hotels by destination	None	destination_id: 'D123'	Map of hotels with hotel instances	Hotels fetched successfully	fetchHotelsByDestination	Hotels fetched for destination D123
TC_UT_017	Ensure error handling when fetching hotels by invalid destination	None	destination_id: 'Invalid'	Error thrown and logged	Error handled correctly	fetchHotelsByDestination	Error fetching hotels by destination
TC_UT_018	Validate fetching a hotel by ID	None	id: 'H1'	Hotel instance for the given ID	Hotel fetched successfully	fetchHotel	Hotel fetched for ID H1

TC_UT_019	Ensure error handling when fetching a hotel by invalid ID	None	id: 'Invalid'	Error thrown and logged	Error handled correctly	fetchHotel	Exceeded API poll limit

Unit testing of the *models/room.js* are meant to test the fetchRoomPrices functions.

Test Case ID	*Description*	*Precondition*	*Input*	*Expected Output*	*Post Condition*	*Function/Module*	*Log*
TC_UT_020	Test fetchRoomPrices() can return list of Room instances	API response is valid and complete	id: 'H123', destination_id: 'D123', checkin: '2024-11-01', checkout: '2024-11-10', lang: 'en_US', currency: 'USD', guests: '2'	Array of Room instances	Rooms specific to hotel id fetched successfully	fetchRoomPrices	API polled successfully
TC_UT_021	Validate handling of incomplete API response with polling	API response is incomplete initially	id: 'H123', destination_id: 'D123', checkin: '2024-11-01', checkout: '2024-11-10', lang: 'en_US', currency: 'USD', guests: '2'	Array of Room instances after polling completes	API polling handled correctly	fetchRoomPrices	API polled multiple times

TC_UT_022	Ensure error handling when API fails	API request fails	id: 'H123', destination_id: 'D123', checkin: '2024-11-01', checkout: '2024-11-10', lang: 'en_US', currency: 'USD', guests: '2'	Error thrown and logged	Error handled correctly	fetchRoomPrices	Error fetching room prices
TC_UT_023	Validate handling of exceeded API poll limit	API response never completes	id: 'H123', destination_id: 'D123', checkin: '2024-11-01', checkout: '2024-11-10', lang: 'en_US', currency: 'USD', guests: '2'	Error indicating poll limit exceeded	Error handled correctly	fetchRoomPrices	Poll limit exceeded

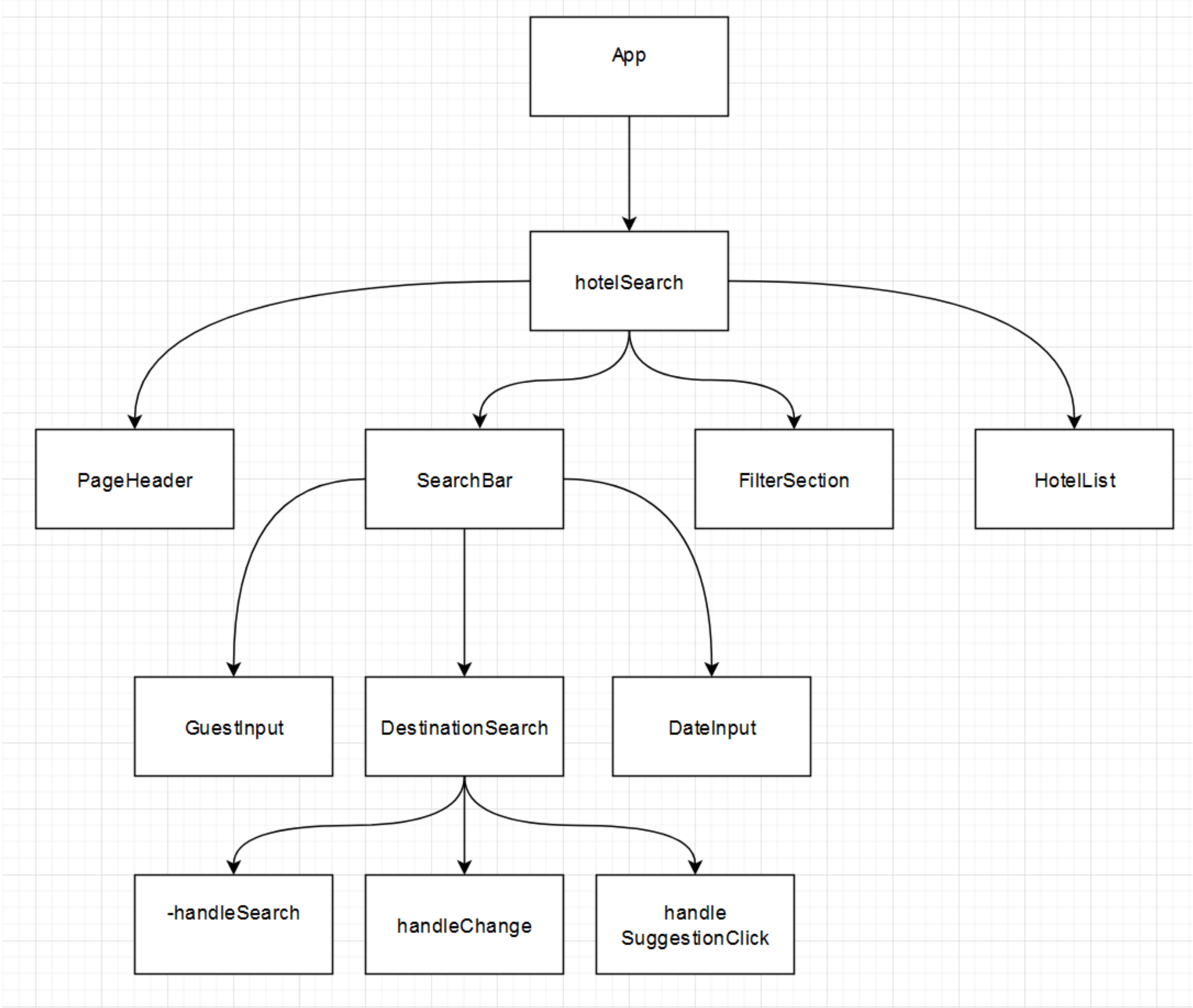
Testing Plan

The team is currently in the process of integrating the frontend and backend functionalities. As such we have included our plans for integration testing under this section for our testing plan.

Phase	Activity	Duration
Unit Testing and Integration Testing	Completion of Unit Testing and Begin Integration Testing	1 week (Week 11)
Integration Testing and E2E testing	Completion of Integration Testing and Begin E2E testing	1 week (Week 12)
Week 13	Completion of E2E testing and project handover	1 week (Week 13)

Integration Testing

For the integration testing we would like to ensure that the various components within the system are working together correctly once unit testing of the backend controllers and models have been concluded. We plan to use the bottom-up approach for our integration testing.



Integration Test Cases

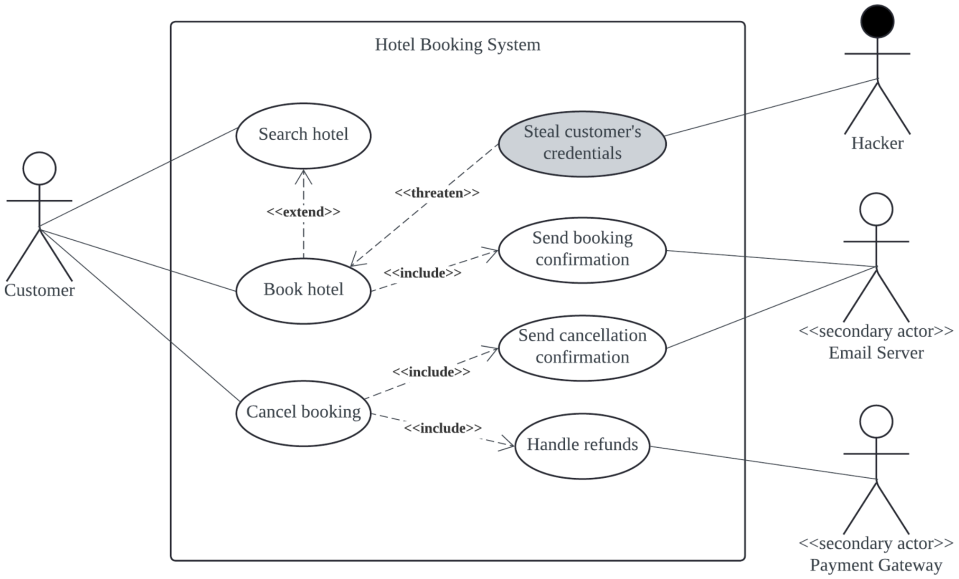
Test Case ID	Description	Precondition	Input	Expected Output	Function/Module	Log
IT001	Test DestinationSearch component independently	Component is isolated	User types in search field	Filtered destinations are displayed	DestinationSearch	

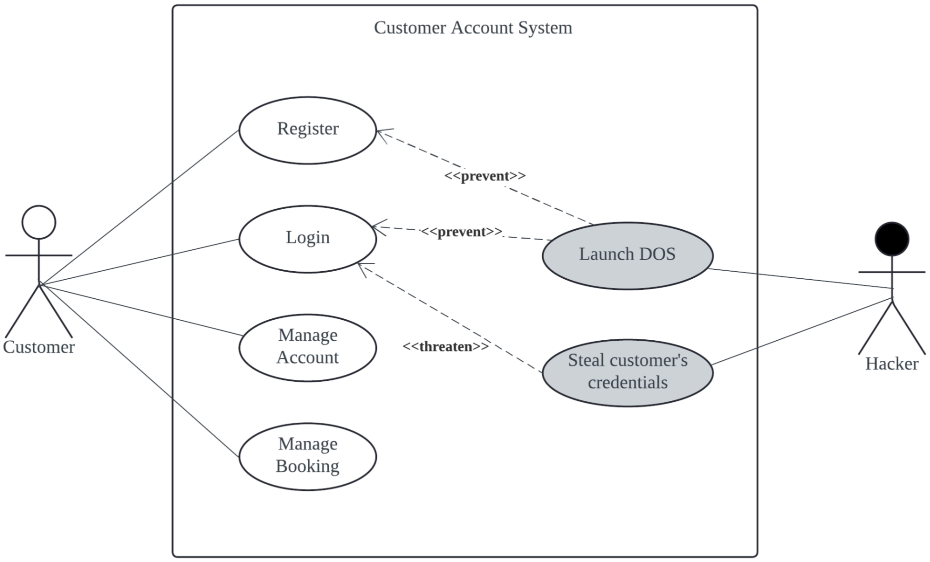
IT002	Test DateInput component independently	Component is isolated	User selects a date range	Selected date range is updated	DateInput	
IT003	Test GuestInput component independently	Component is isolated	User selects number of guests	Selected number of guests is updated	GuestInput	
IT004	Integrate SearchBar with DestinationSearch	SearchBar and DestinationSearch components are available	User types in search field within SearchBar	Filtered destinations are displayed	SearchBar, DestinationSearch	
IT005	Integrate SearchBar with DateInput	SearchBar and DateInput components are available	User selects a date range within SearchBar	Selected date range is updated	SearchBar, DateInput	
IT006	Integrate SearchBar with GuestInput	SearchBar and GuestInput components are available	User selects number of guests within SearchBar	Selected number of guests is updated	SearchBar, GuestInput	
IT007	Test FilterSection component independently	Component is isolated	User interacts with filters	Filters are updated accordingly	FilterSection	
IT008	Integrate hotelSearch with SearchBar	hotelSearch and SearchBar components are available	User interacts with SearchBar	Search criteria are updated	hotelSearch, SearchBar	
IT009	Integrate hotelSearch with FilterSection	hotelSearch and FilterSection components are available	User interacts with FilterSection	Filters are applied to search results	hotelSearch, FilterSection	

IT010	Integrate hotelSearch with HotelList	hotelSearch and HotelList components are available	Search criteria and filters are applied	Filtered hotel list is displayed	hotelSearch, HotelList	
IT011	Integrate hotelSearch with PageHeader	hotelSearch and PageHeader components are available	User interacts with header options	Header interactions are handled correctly	hotelSearch, PageHeader	
IT012	Full integration test of hotelSearch	All components are available	User interacts with search, filters, and header	Correctly filtered hotel list is displayed	hotelSearch, SearchBar, FilterSection, HotelList, PageHeader	

UML Diagrams

Use Case Diagram

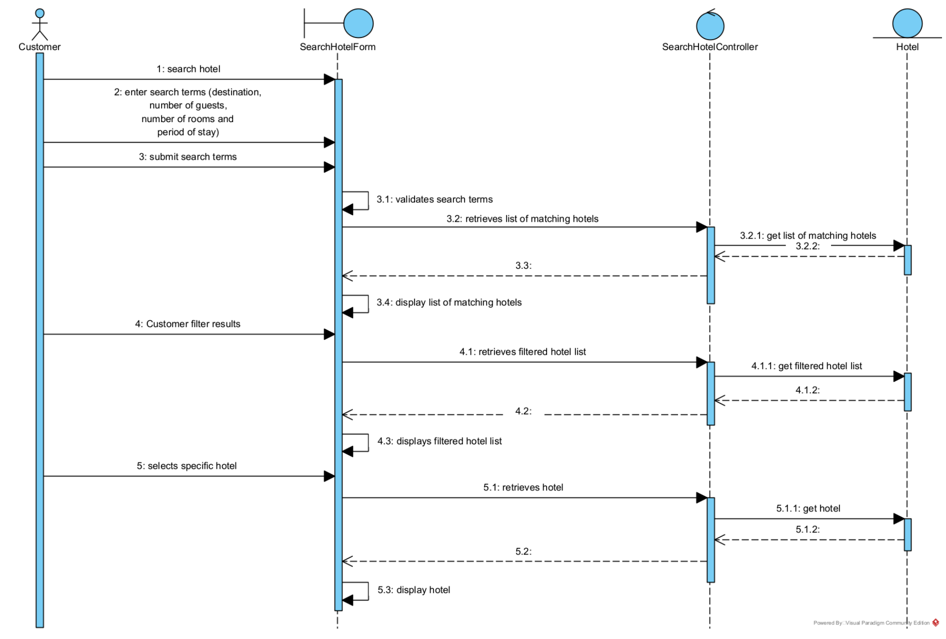




Use Case Descriptions

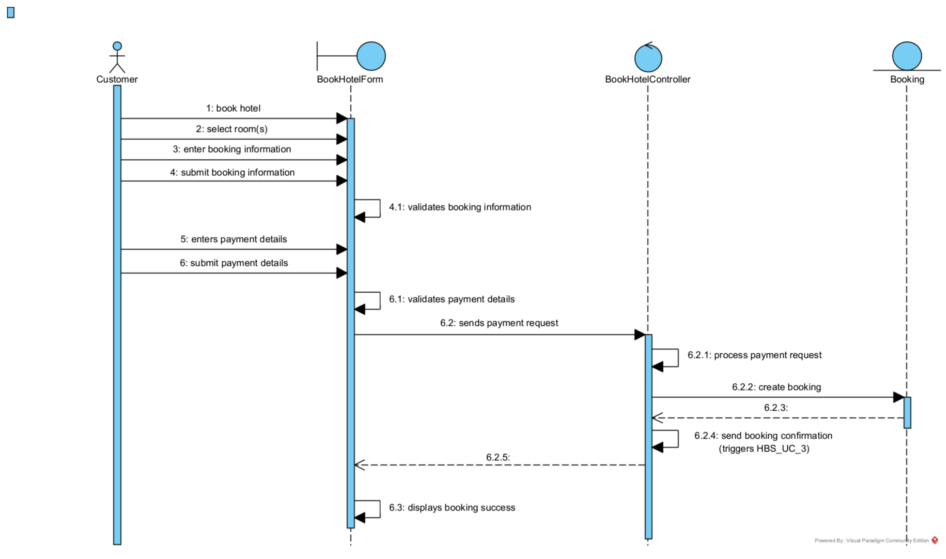
Use Case ID	HBS_UC_1
Use Case Name	Search Hotel
Author	Ryan Lee
Date Created	3/7/2024
Description	This use case allows customers to search for hotels by destination, dates of stay, number of guests, and number of rooms. For each field, the customer selects input values from a predefined list. Upon submission, the system displays a list of matching hotels along with the cheapest room for each hotel. Customers can filter the list by star ratings, guest ratings, and price range. The customer can then select a specific hotel from the list to view a list of matching room types provided and any other additional information.
Primary Actor	Customer
Secondary Actor	Ascenda Hotel API
Precondition	1. Hotel Listing API is available and functioning
Postconditions	1. System displays details for specific hotel
Main Flow	1. Customer inputs destination, date of stay, number of guests and number of rooms (select from list of valid input values) 2. Customer submits inputs 3. System validates inputs 4. System retrieves list of matching hotels 5. System displays list of matching hotels 6. Customer filters results by star ratings, guest ratings, and price range 7. System retrieves list of filtered matching hotels 8. System displays list of filtered matching hotels 9. Customer selects specific hotel from result list 10. System displays details of the selected hotel such as matching room details, ratings, and any other additional information 11. Extension point : customer books hotel (HBS_UC_2) 12. Use case ends
Alternate Flow	3a. Missing mandatory search criterion/criteria 1. System prompts customer for the missing mandatory search fields 2. Use case resumes at main flow step 1

Sequence Diagram


Customer
SearchHotelForm
SearchHotelController
Hotel
1: search hotel
2: enter search terms (destination,
number of guests,
number of rooms and
period of stay)
3: submit search terms
3.1: validates search terms
3.2: retrieves list of matching hotels
3.2.1: get list of matching hotels
3.2.2:
3.3:
3.4: display list of matching hotels
4: Customer filter results
4.1: retrieves filtered hotel list
4.1.1: get filtered hotel list
4.1.2:
4.2:
4.3: displays filtered hotel list
5: selects specific hotel
5.1: retrieves hotel
5.1.1: get hotel
5.1.2:
5.2:
5.3: display hotel
Powered By: Visual Paradigm Community Edition

Use Case ID	HBS_UC_2
Use Case Name	Book Hotel
Author	Ryan Lee
Date Created	3/7/2024
Description	This use case allows the customer to book a hotel room and make payment via payment gateway. To book a hotel room, the customer enters the following information: first name & last name, phone number & email address, special requests to hotel, payment information (credit card number, expiry date, CVV/CVC), and the billing address. If the customer is logged in, then certain fields are automatically filled in (depending on what information the customer has saved in their account). After payment is processed and booking is confirmed, the system will trigger the send booking confirmation use case (HBS_UC_3) in which the email server will send an email to the customer with booking confirmation and information.
Include Use Cases	1. Send Booking Confirmation (HBS_UC_3)
Extends Use Cases:	1. Search hotel (HBS_UC_1)
Primary Actor	Customer
Secondary Actor	Payment Gateway
Precondition	1. Customer has selected (searched up) a hotel 2. Payment Gateway is available and functioning
Postconditions	1. Hotel room is booked and customer receives a booking confirmation via email 2. Booking details are recorded in database (via Create Booking API)
Main Flow	1. Customer selects room(s) to book 2. Customer enters required booking information (e.g. customer's email) 3. Customer submits booking information 4. System validates booking information 5. Customer enters payment details 6. Customer submits payment details 7. System validates payment details 8. System sends payment request to Payment Gateway 9. Payment Gateway processes the payment and returns a

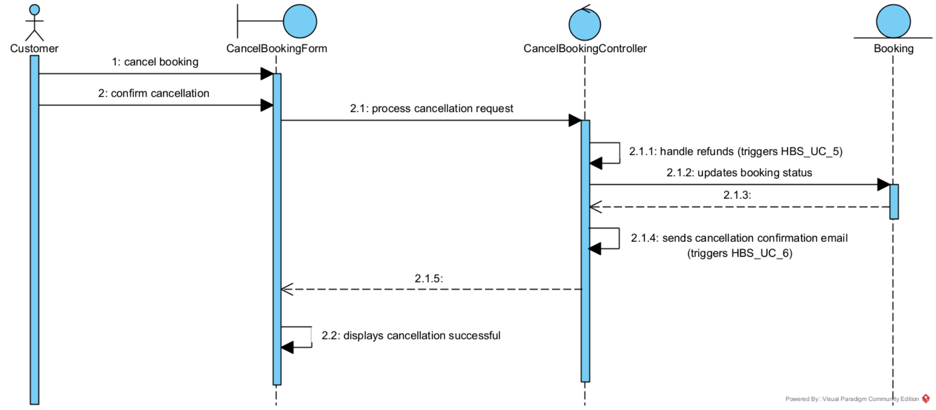
	confirmation 10. System confirms the booking and create booking in database 11. System triggers send booking confirmation use case (HBS_UC_3) 12. System displays “booking success” message 13. Use case ends
Alternate Flow	2a. Customer is logged in 1. System auto fills form based on information saved to customer’s account 2. Customer fills in remaining fields 3. Use case resumes at main flow step 3 4a. Invalid or missing booking information 1. System prompts customer for missing/incorrect information 2. Use case resumes at main flow step 2 5a. Customer is logged in and has saved payment information 1. System auto fills payment details form based on payment information saved to customer’s account 2. Customer can modify the fields 3. Use case resumes at main flow step 7 6a. Customer is logged in and does not have payment information saved to account 1. System asks customer if they wish to save payment information to account 2. System says payment information to account if customer indicates to do so 3. Use case resumes at main flow step 8 7a. Invalid or missing payment details (that can be detected without Payment Gateway) 1. System prompts customer to fill in missing details or correct incorrect information 2. Use case resumes at main flow step 6 9a. Payment declined by Payment gateway 1. System informs customer that the payment was declined 2. a) Customer can retry the payment with different payment details by resuming use case at main flow step 6 b) Customer can cancel the booking process. Use case ends.

Sequence Diagram	

Customer
BookHotelForm
BookHotelController
Booking
1: book hotel
2: select room(s)
3: enter booking information
4: submit booking information
4.1: validates booking information
5: enters payment details
6: submit payment details
6.1: validates payment details
6.2: sends payment request
6.2.1: process payment request
6.2.2: create booking
6.2.3:
6.2.4: send booking confirmation
(triggers HBS_UC_3)
6.2.5:
6.3: displays booking success
Powered By: Visual Paradigm Community Edition

Use Case ID	HBS_UC_3
Use Case Name	Send Booking Confirmation
Author	Ryan Lee
Date Created	3/7/2024
Description	This use case allows the system to send a booking confirmation email to a customer (via the email server) after they have made a successful booking. This use case is included in the book hotel use case (HBS_UC_2) as it is part of the booking process, triggered when the customer's booking is confirmed. The email should notify customer that their booking was successful and provide them all relevant information: destination ID, hotel ID, booking display information (number of nights, start date, end date, adults, children, message to hotel, room types), price, booking reference (booking ID), guest information (salutation, first name, last name), payee information (payment ID, payee ID). The email should also contain a unique URL that redirects the customer to a page that allows them to cancel that booking if they so wish.
Primary Actor	
Secondary Actor	Email Server
Precondition	1. Email server is available and functioning 2. Customer made successful booking (system triggers this use case)
Postconditions	1. Customer receives a booking confirmation email with all relevant details
Main Flow	1. System compiles customer's booking details 2. System generates booking confirmation from compiled data 3. System composes an email with booking confirmation details 4. System sends composed email to the customer's email address via the email server 5. Use case ends
Alternate Flow	4a. Email server fails to send email 1. System retries sending the email via email server a specified number of times 2. If the email still cannot be sent, the system logs the failure and notifies system administrator 3. Use case ends

Use Case ID	HBS_UC_4
Use Case Name	Cancel Booking
Author	Ryan Lee
Date Created	3/7/2024
Description	This use case enables customers to cancel their previously made hotel booking. This use case includes the send handle refunds use case (HBS_UC_5) which triggers after the system processes cancellation, in which the system interacts with the payment gateway for any necessary refunds. Upon successful cancellation, the system will trigger the included send cancellation confirmation use case (HBS_UC_6), in which a cancellation confirmation email is sent to the customer. This use case is reached by customers via the special URL in their booking confirmation email, unique to that particular booking.
Include Use Cases	1. Handle Refunds (HBS_UC_5) 2. Send Cancellation Confirmation (HBS_UC_6)
Primary Actor	Customer
Secondary Actor	
Precondition	1. Customer has existing booking that they wish to cancel 2. Payment gateway and email server are available and functioning
Postconditions	1. Booking is cancelled 2. Customer receives a cancellation confirmation email 3. Booking status is updated to ‘cancelled’ in database
Main Flow	1. Customer navigates to booking cancellation page unique to their booking, via URL in booking confirmation email 2. System prompts customer to confirm cancellation 3. Customer confirms booking cancellation 4. System processes cancellation request 5. System triggers handle refunds (HBS_UC_5) use case 6. System updates booking status to ‘cancelled’ in database 7. System triggers send cancellation confirmation (HBS_UC_6) use case 8. System displays “Booking cancelled successfully.” 9. Use case ends
Alternate Flow	4a. Too late to cancel booking 1. System notifies customer that it is too late to cancel the booking 2. Use case ends

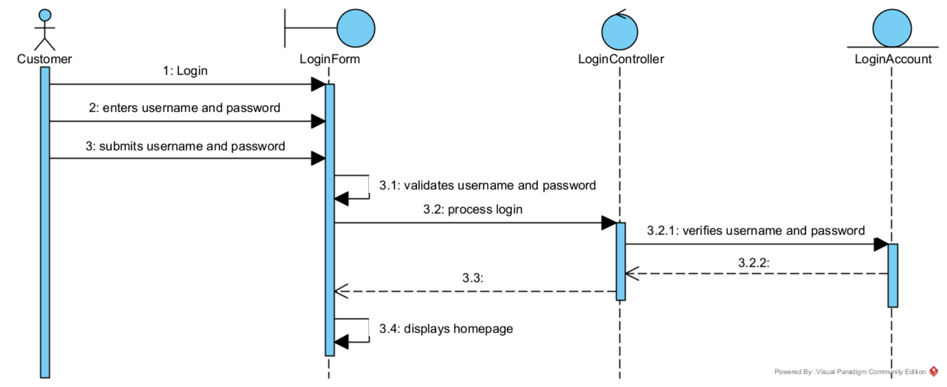
	4b. Booking status is already cancelled 1. System notifies customer that the booking is already cancelled 2. Use case ends 5a. Refund request rejected by payment gateway 1. System informs customer that refund request has been rejected and thus booking cancellation request is also incomplete 2. Use case ends
Sequence Diagram	 Customer CancelBookingForm CancelBookingController Booking 1: cancel booking 2: confirm cancellation 2.1: process cancellation request 2.1.1: handle refunds (triggers HBS_UC_5) 2.1.2: updates booking status 2.1.3: 2.1.4: sends cancellation confirmation email (triggers HBS_UC_6) 2.1.5: 2.2: displays cancellation successful Powered By: Visual Paradigm Community Edition

Use Case ID	HBS_UC_5
Use Case Name	Handle Refunds
Author	Ryan Lee
Date Created	3/7/2024
Description	This use case enables the process of refunding the customer when they cancel a booking. This use case is included in the cancel booking use case (HBS_UC_4), triggered after the cancellation request is processed (refund request allowed by system). The system interacts with the payment gateway to process the refund request.
Primary Actor	
Secondary Actor	Payment Gateway
Precondition	1. Booking cancellation request is processed and confirmed by system 2. Payment gateway is available and functioning
Postconditions	1. Payment gateway processes and confirms refund request transaction
Main Flow	1. System fetches booking details needed for refund request 2. System sends a refund request to the payment gateway with the calculated amount and transaction details 3. The payment gateway processes the refund 4. The payment gateway returns confirmation of refund to the system 5. Use case ends
Alternate Flow	3a. Refund request rejected by payment gateway 1. System informs customer that refund request has been rejected and thus booking cancellation request is also incomplete 2. Use case ends

Use Case ID	HBS_UC_6
Use Case Name	Send Cancellation Confirmation
Author	Ryan Lee
Date Created	3/7/2024
Description	This use case allows the system to send a cancellation confirmation email to a customer (via the email server) after they have successfully cancelled a booking. This use case is included in the cancel booking use case (HBS_UC_4), triggered when the booking cancellation is confirmed. The email should notify the customer that their booking cancellation was successful.
Primary Actor	
Secondary Actor	Email Server
Precondition	1. Email server is available and functioning 2. Customer successfully cancelled booking (refunded)
Postconditions	1. Customer receives an email confirming their booking cancellation
Main Flow	1. System composes an email notifying customer of successful booking cancellation 2. System sends composed email to the customer’s email address via the email server 3. Use case ends
Alternate Flow	2a. Email server fails to send email 1. System retries sending the email via email server a specified number of times 2. If the email still cannot be sent, the system logs the failure and notifies system administrator 3. Use case ends

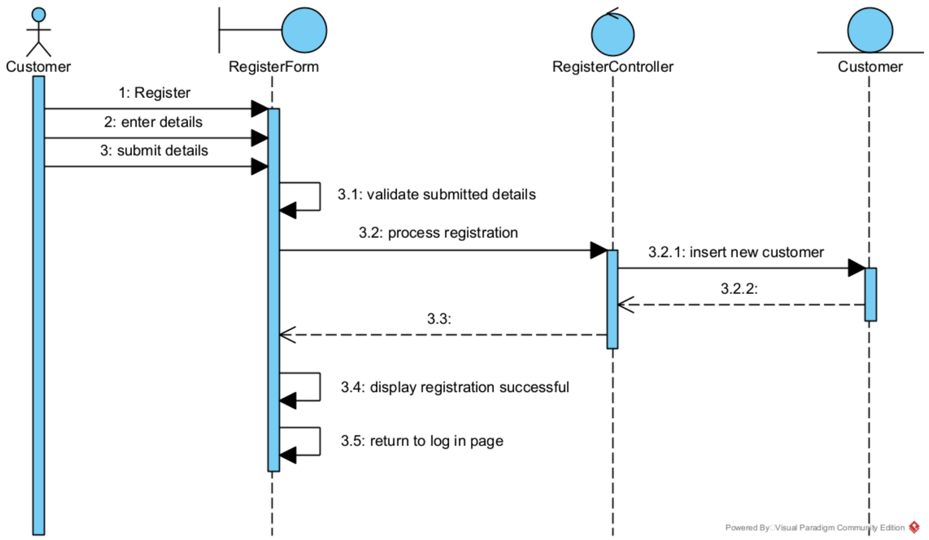
Use Case ID	CAS_UC_1
Use Case Name	Login
Author	Ryan Lee
Date Created	3/7/2024
Description	This use case enables customers to login to their accounts before making a booking. While logging in is not required for booking, it provides additional convenience by autofilling the customer’s saved details and preferences when making a booking. Once a customer is logged in, they can proceed to fill in and save said details and preferences to their account.
Primary Actor	Customer
Secondary Actor	
Precondition	1. Customer has to have valid account
Postconditions	1. System displays the relevant homepage with customer authenticated and logged in.
Main Flow	1. Customer navigates to login page 2. Customer enters username and password 3. Customer submits username and password 4. System validates the username and password 5. System processes login 6. System verifies the username and password 7. System returns to homepage (logged in) 8. Use case ends
Alternate Flow	4a. Missing username and/or password 1. System prompts customer for username and password 2. Use case resumes at main flow step 2 6a. Invalid username and/or password 1. System displays “Invalid username and/or password” message 2. System prompts customer for username and password 3. Use case resumes at main flow step 2 7a. Customer wishes to save booking details & preferences 1. Customer navigates to “Manage Account” page 2. Customer enters details & preferences they wish to save to their account (for autofill purposes) 3. Customer submits form 4. System saves information to database 5. Use case ends

Sequence Diagram

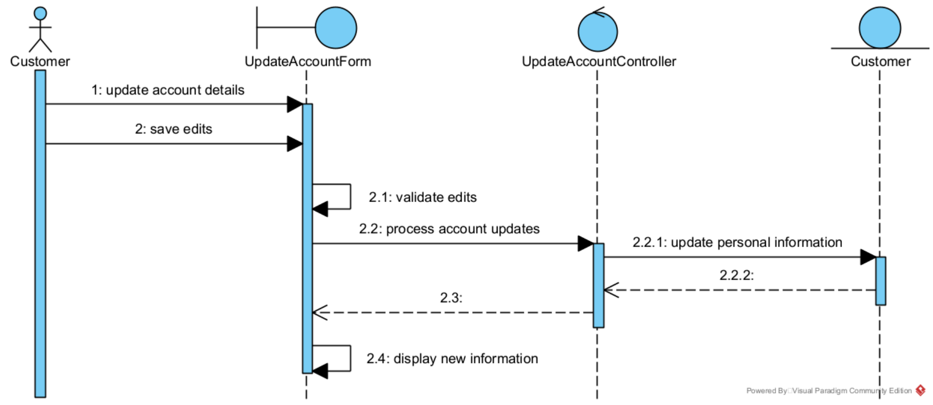

Customer
LoginForm
LoginController
LoginAccount
1: Login
2: enters username and password
3: submits username and password
3.1: validates username and password
3.2: process login
3.2.1: verifies username and password
3.2.2:
3.3:
3.4: displays homepage
Powered By: Visual Paradigm Community Edition

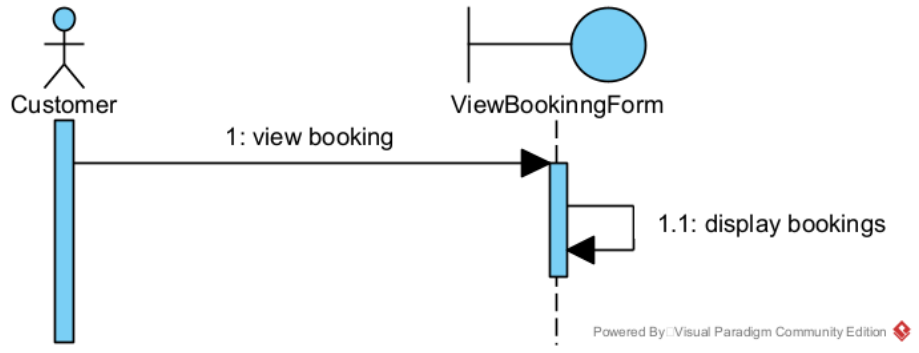
Use Case ID	CAS_UC_2
Use Case Name	Register
Author	Yee Jia Zhen
Date Created	4 July 2024
Description	This use case is for new users to create an account for the hotel booking web application.
Primary Actor	Customer
Secondary Actor	
Precondition	1. Customer is not an existing user
Postconditions	1. Customer account created
Main Flow	1. Customer navigates to the register account page 2. Customer fill in their details 3. Customer submit the register account form 4. System validates submitted details 5. System process registration 6. System adds customer to database 7. System displays registration successful message 8. System returns to log in page 9. Use case ends
Alternate Flow	4a. Username or email taken 1. System prompts customer for new username or email 2. Use case resumes at main flow step 3 4b. Weak password 1. System prompts customer for a strong password 2. Use case resumes at main flow step 3

Sequence Diagram

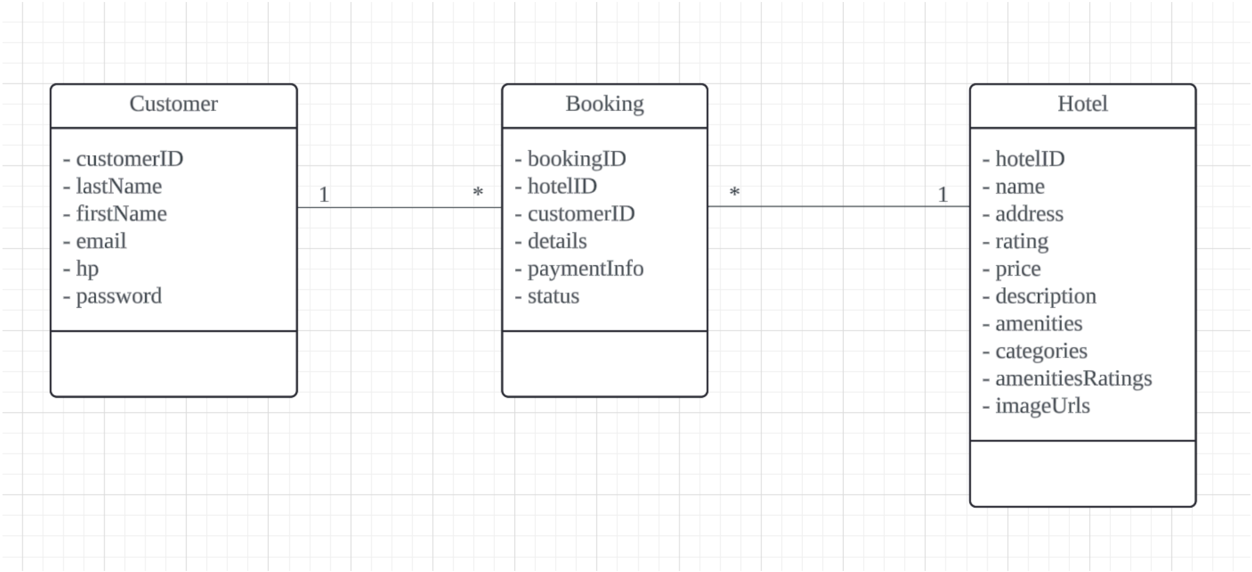


Customer
RegisterForm
RegisterController
Customer
1: Register
2: enter details
3: submit details
3.1: validate submitted details
3.2: process registration
3.2.1: insert new customer
3.2.2:
3.3:
3.4: display registration successful
3.5: return to log in page
Powered By: Visual Paradigm Community Edition

Use Case ID	CAS_UC_3
Use Case Name	Update Account Details
Author	Yee Jia Zhen
Date Created	4 July 2024
Description	This use case allows customers to view and amend their personal information that they have provided during registration.
Primary Actor	Customer
Secondary Actor	
Precondition	1. Customer has a valid account
Postconditions	1. Customer account details successfully updated
Main Flow	1. Customer navigates to account page 2. Customer edit their personal information 3. Customer save their edits 4. System validates their personal information 5. System process account updates 6. System updates the database 7. System display new information 8. Use case ends
Alternate Flow	4a. Username or email taken 3. System prompts customer for new username or email 4. Use case resumes at main flow step 3 4b. Weak password 3. System prompts customer for a strong password 4. Use case resumes at main flow step 3
Sequence Diagram	 Customer UpdateAccountForm UpdateAccountController Customer 1: update account details 2: save edits 2.1: validate edits 2.2: process account updates 2.2.1: update personal information 2.2.2: 2.3: 2.4: display new information Powered By: Visual Paradigm Community Edition

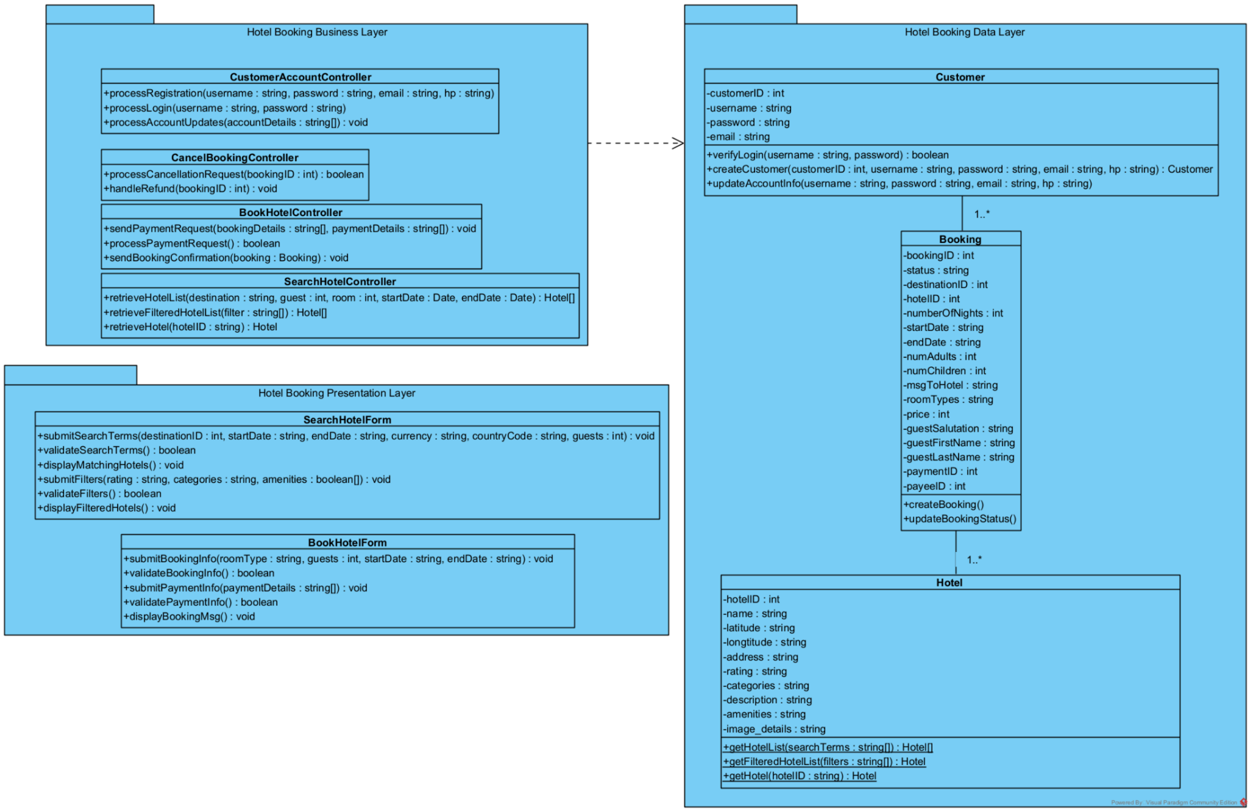
Use Case ID	CAS_UC_4
Use Case Name	View Booking
Author	Yee Jia Zhen
Date Created	4 July 2024
Description	This use case allows customers to view their hotel booking(s) information and gives them the option to cancel their booking(s).
Primary Actor	Customer
Secondary Actor	
Precondition	1. Customer has a valid account
Postconditions	1. Customer's booking(s) successfully cancelled 2. Customer booking(s) successfully changed.
Main Flow	1. Customer navigates to bookings page 2. System displays customer's booking(s) 3. Use case ends
Alternate Flow	2a. Customer wishes to cancel booking 1. Customer selects cancel booking option, triggering cancel booking use case (HBS_UC_4) 2. Use case ends
Sequence Diagram	 Customer ViewBookinngForm 1: view booking 1.1: display bookings Powered By Visual Paradigm Community Edition

Domain Class Diagram



Solution Class Diagram

SCD for compulsory use cases. Still figuring out certain parameters and hence have generalised them as string arrays - will revise these later (iterative process).


Hotel Booking Business Layer
CustomerAccountController
+processRegistration(username : string, password : string, email : string, hp : string)
+processLogin(username : string, password : string)
+processAccountUpdates(accountDetails : string[]) : void
CancelBookingController
+processCancellationRequest(bookingID : int) : boolean
+handleRefund(bookingID : int) : void
BookHotelController
+sendPaymentRequest(bookingDetails : string[], paymentDetails : string[]) : void
+processPaymentRequest() : boolean
+sendBookingConfirmation(booking : Booking) : void
SearchHotelController
+retrieveHotelList(destination : string, guest : int, room : int, startDate : Date, endDate : Date) : Hotel[]
+retrieveFilteredHotelList(filter : string[]) : Hotel[]
+retrieveHotel(hotelID : string) : Hotel
Hotel Booking Presentation Layer
SearchHotelForm
+submitSearchTerms(destinationID : int, startDate : string, endDate : string, currency : string, countryCode : string, guests : int) : void
+validateSearchTerms() : boolean
+displayMatchingHotels() : void
+submitFilters(rating : string, categories : string, amenities : boolean[]) : void
+validateFilters() : boolean
+displayFilteredHotels() : void
BookHotelForm
+submitBookingInfo(roomType : string, guests : int, startDate : string, endDate : string) : void
+validateBookingInfo() : boolean
+submitPaymentInfo(paymentDetails : string[]) : void
+validatePaymentInfo() : boolean
+displayBookingMsg() : void
Hotel Booking Data Layer
Customer
-customerID : int
-username : string
-password : string
-email : string
+verifyLogin(username : string, password) : boolean
+createCustomer(customerID : int, username : string, password : string, email : string, hp : string) : Customer
+updateAccountInfo(username : string, password : string, email : string, hp : string)
1..*
Booking
-bookingID : int
-status : string
-destinationID : int
-hotelID : int
-numberOfNights : int
-startDate : string
-endDate : string
-numAdults : int
-numChildren : int
-msgToHotel : string
-roomTypes : string
-price : int
-guestSalutation : string
-guestFirstName : string
-guestLastName : string
-paymentID : int
-payeeID : int
+createBooking()
+updateBookingStatus()
1..*
Hotel
-hotelID : int
-name : string
-latitude : string
-longtitude : string
-address : string
-rating : string
-categories : string
-description : string
-amenities : string
-image_details : string
+getHotelList(searchTerms : string[]) : Hotel[]
+getFilteredHotelList(filters : string[]) : Hotel
+getHotel(hotelID : string) : Hotel
Powered By: Visual Paradigm Community Edition