SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

50.003

Information Systems Technology and Design

Elements of Software Construction
Project 2 Ascenda Loyalty Hotel Booking System

CC3 Group C3I3

John-David Tan Ming Sheng 1005971

Tan Xynn Ee, Cordelia 1006138

Venkatakrishnan Logganesh 1006050

Wong Jia Kang 1006022

Rajavelu Sree Devi 1006591

Shjonathan Tan Tze Ern 1005987

Zachary Seow Shijie 1005979

Contents

1. Product Description

1.1 Purpose

The purpose of our project is to develop a high-performance hotel booking platform that powers white-labelled solutions for banks, airlines, and loyalty programs. The platform will enable customers to search, book, and redeem hotel night stays seamlessly. The project will implement the following functional features.

- Provide users with a fast and convenient booking experience by means of a fast auto complete search feature which suggests relevant destinations as users type.
- Allows users to plan their travel itinerary and finance costs beforehand through the incorporation of the date picker and options to specify the number of guests and rooms.
- Allows users to customize their own booking experience by providing detailed information about the selected hotel, especially the number of rooms available in the hotel.
- Allows users to navigate and have a pleasant, comfortable visiting experience on a foreign land with its various services to good use.

1.2 Scope

Our app allows users to access details of each hotel by selectively clicking on the hotel that they are interested in. Users will be able to find all the hotels located within the vicinity of their neighbourhood and select the cheapest hotels to minimize finance costs. User can create an account free of charge and register the account through the app to access the various services of the hotel booking platform and enjoy a comfortable booking experience.

1.3 Users and Stakeholders

The main stakeholders of our project include Ascenda and customers.

- Ascenda: As the provider of technology and services for the white-labeled platform, they are deeply interested in the successful implementation and adoption of the app by partner banks and airlines.
- Customers: The end customers who use these programs are the most important stakeholders since their satisfaction, engagement and adoption play a pivotal role in the success of this project.

1.4 Assumptions and Constraints

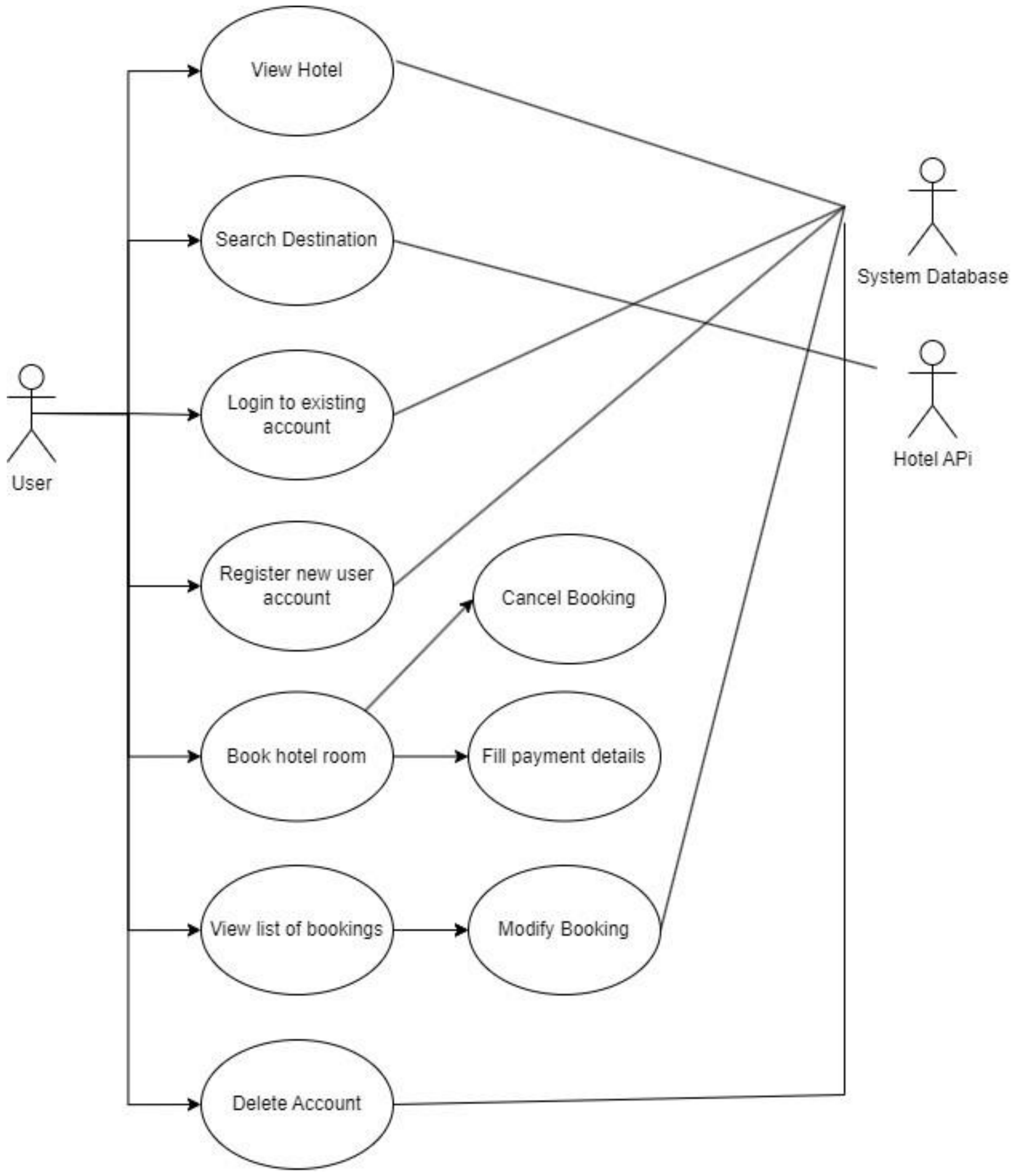
- App assumes that every user’s phone has an active internet connection.
- App assumes every user has an active google maps functionality on his phone.

1.5 Risks (Constraints) And Solutions

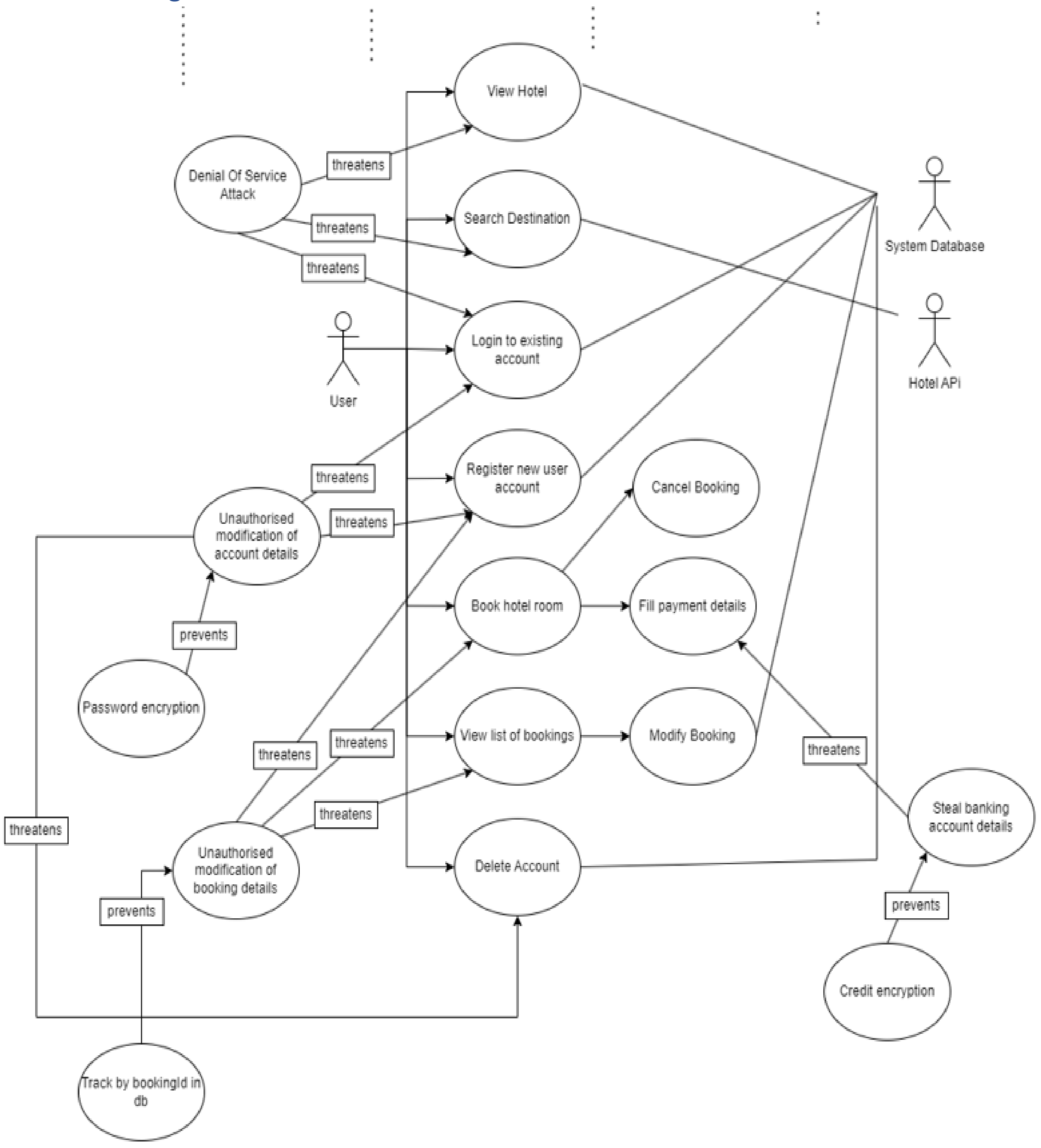
Risks Identified	**Potential Solutions/Mitigations**
Agile cycle makes it difficult to slot in last minute or sudden changes on the fly into the development cycle. Our group is unsure of how many these will exist, but it is a potential problem if too many last-minute changes come in and features are already being implemented and something to note just in case.	Collate all the last-minute changes and add them onto the next development cycle, or if it is the last one, add one more cycle to do all of it at one shot.
Inexperience of team members in web development could cause issues with the development process. Examples include using the wrong frameworks for the wrong features, implementing the system using bad code etc.	Study up ahead of time on the frameworks that we will be using (NodeJS, ReactJS, MongoDB) so that we are more familiar with the syntax. Search up guides also to ensure familiarity with the processes.
Multiple people working on the same part of the code could result in conflicts or overwritten bits of code when nothing is communicated about who is going to upload what.	Active communication between group members and proper division of labour can help mitigate this. A repository with proper branching can also help to spot these conflicts and adjust where necessary.
Integration of different subsystems could be difficult due to different inputs and output classes, possible messiness when trying to merge all subsystems together.	Try to mitigate by deciding on template inputs/outputs ahead of time and writing the code based on those predetermined decisions. Have a final check after completion to test if they work together.
Security constraints with using the API given to us as well as with the payment portal set up.	Share code using only private repositories and try to restrict access to it as much as possible to ensure that personal data and other forms of important code are not leaked out into the public.
Time constraints due to having to complete the whole project within the term period of around 3 months.	Plan ahead on what to do when. Schedule each activity and provide proper deadlines. Divide and ensure that each phase of the development cycle has an assigned time limit to ensure work can be done before each meeting.

2. Functional Requirements

2.1 Use Case Diagrams



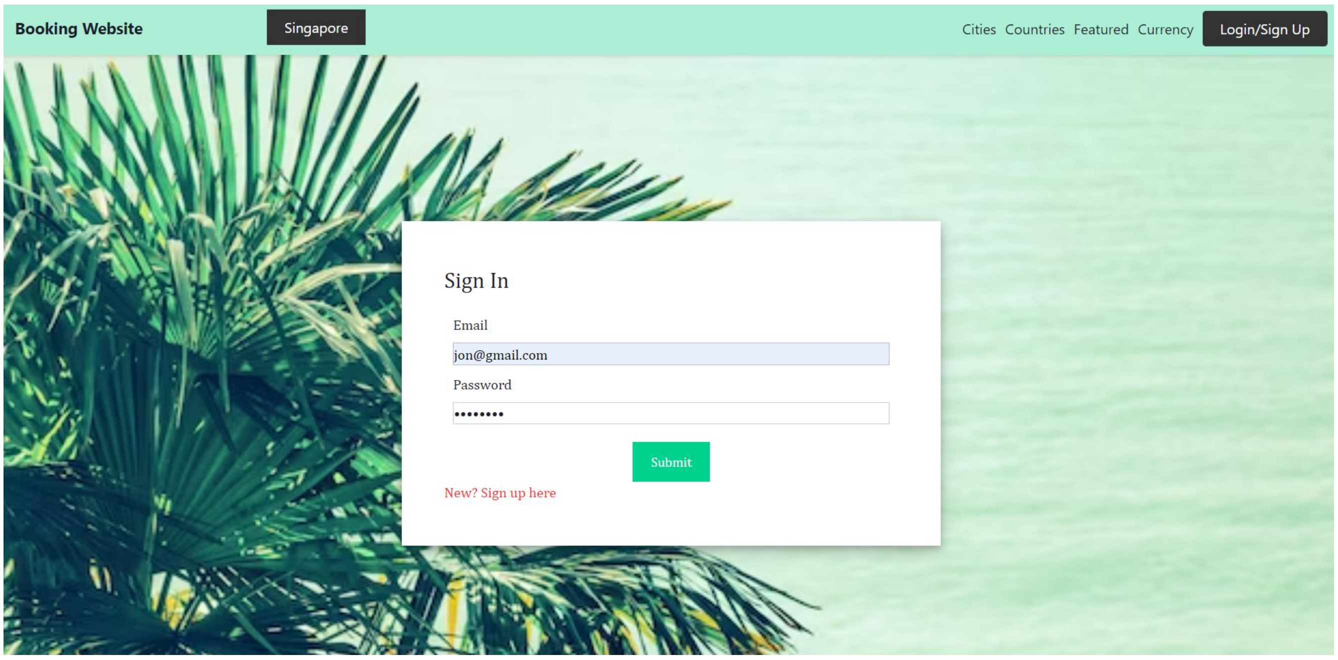
2.2 Misuse Case Diagrams

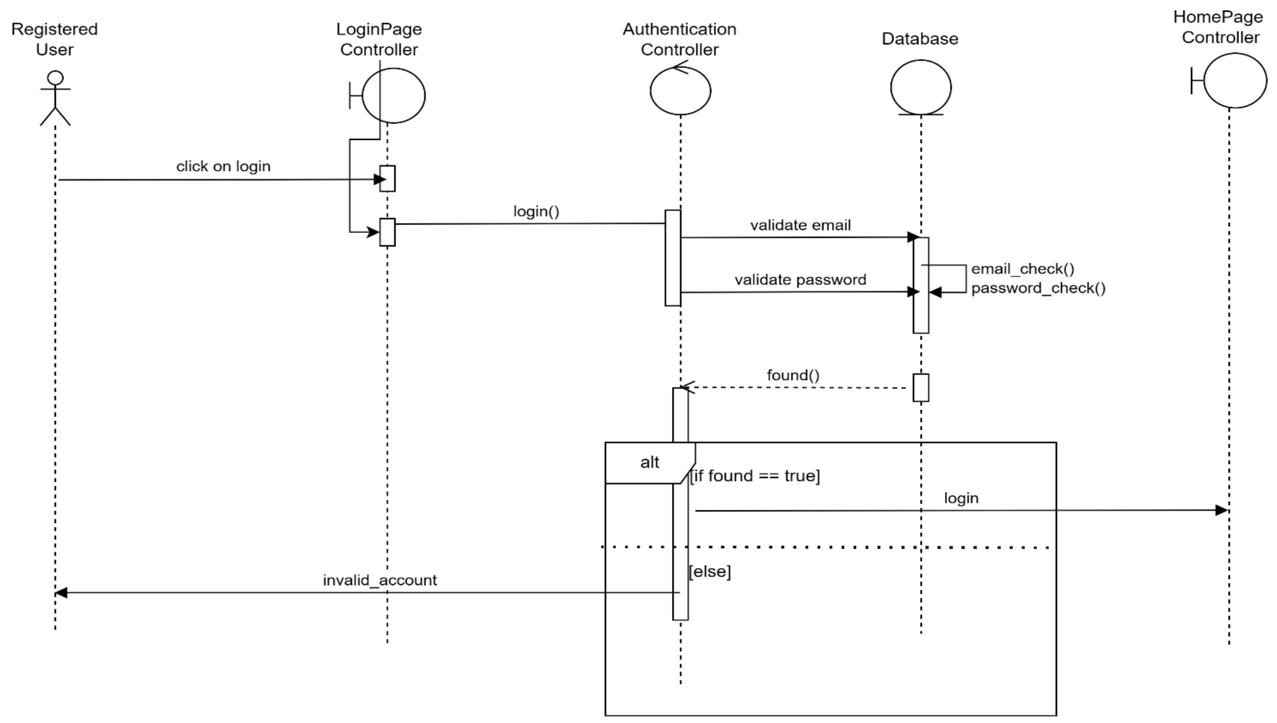


2.3 Case 1: Use Case Document, Sequence Diagrams and UI Illustration

1)Login to user account

At the sign in page user enters his email and password and logs into the app.



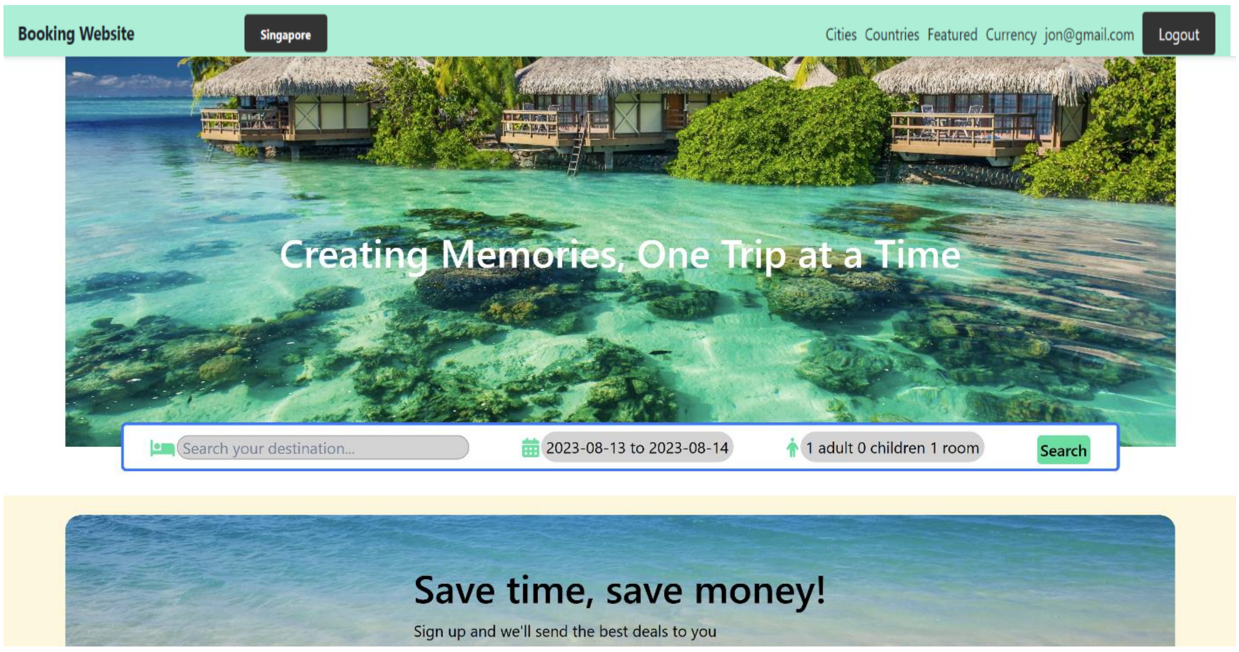


ID	HotelBooking_UC_01
Name	Login to user account
Objective	User to log into existing account
Pre-conditions	User must have existing account with app
Post-conditions	Success 1. User enters valid email address and password associated with the account/successful validation by the system 2. User successfully logs in to account Failure 1. User enters invalid credentials 2. User account not found in database
Actors	Primary 1. User Secondary 1. Database
Trigger	When user wants to log in
Normal flow	1. User enters login credentials – email address / username and password and submits request to log in 2. Database checks validity of credentials: email address / username belongs to a registered account and password matches the registered account's password 3. System authenticates login 4. User successfully logs in to user account
Alternative flow	2.a User enters invalid credentials; use case concludes with error notification 2.b Database unable to verify credentials; use case concludes with timeout notification
Interacts with	Account Database

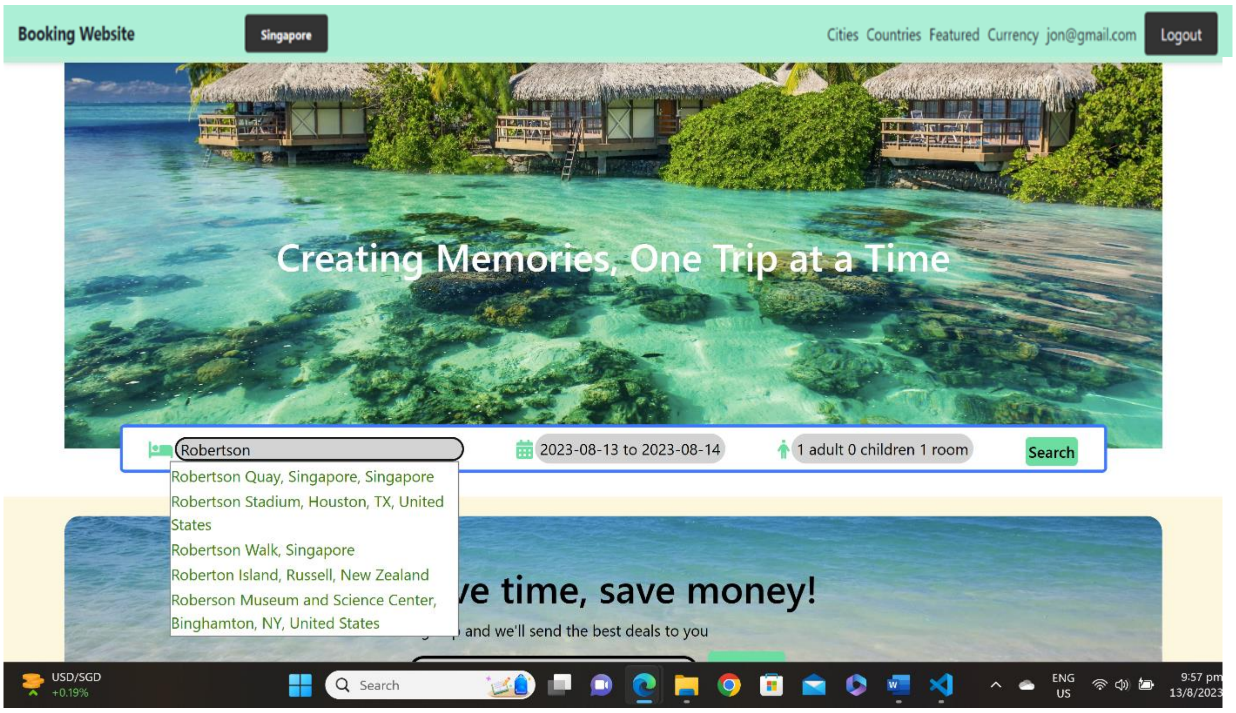
2.3 Case 2: Use Case Document, Sequence Diagrams and UI Illustration

2) User makes a search for the hotel

Step 1: User clicks on the 'Search your destination' textbox.

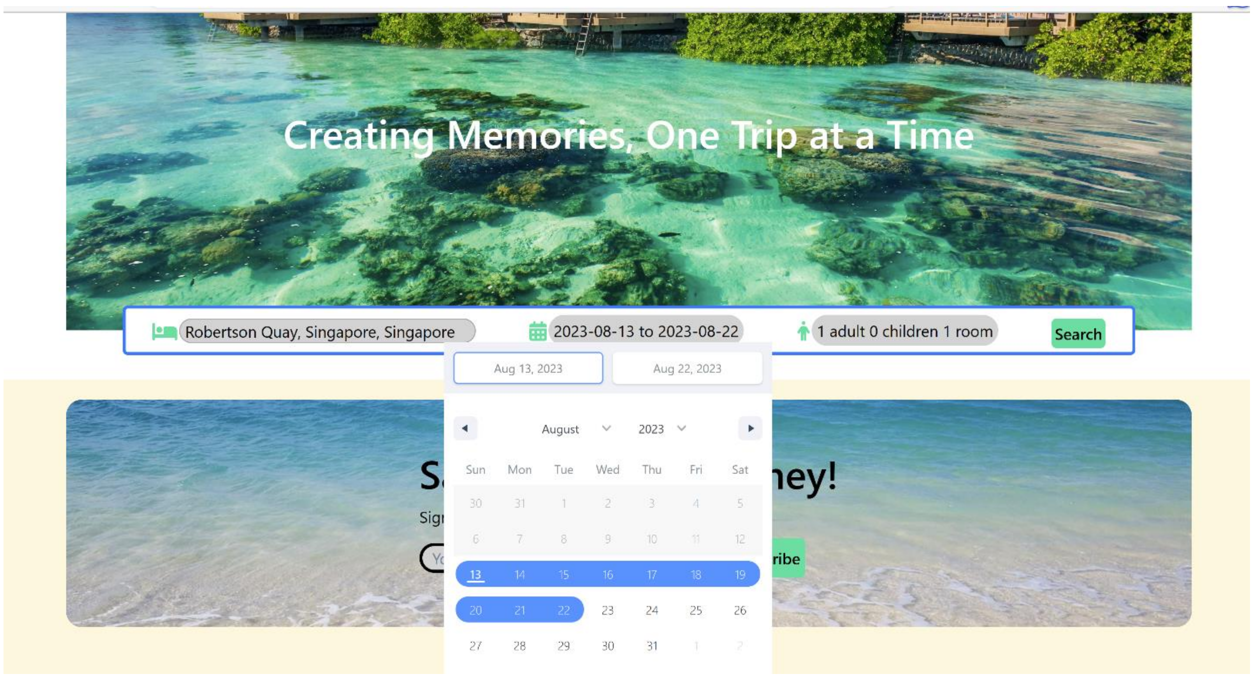

Booking Website
Singapore
Cities Countries Featured Currency jon@gmail.com
Logout
Creating Memories, One Trip at a Time
Search your destination...
2023-08-13 to 2023-08-14
1 adult 0 children 1 room
Search
Save time, save money!
Sign up and we'll send the best deals to you

Step 2: User starts typing text into the box and a dropdown list of recommendations of various destinations is shown below

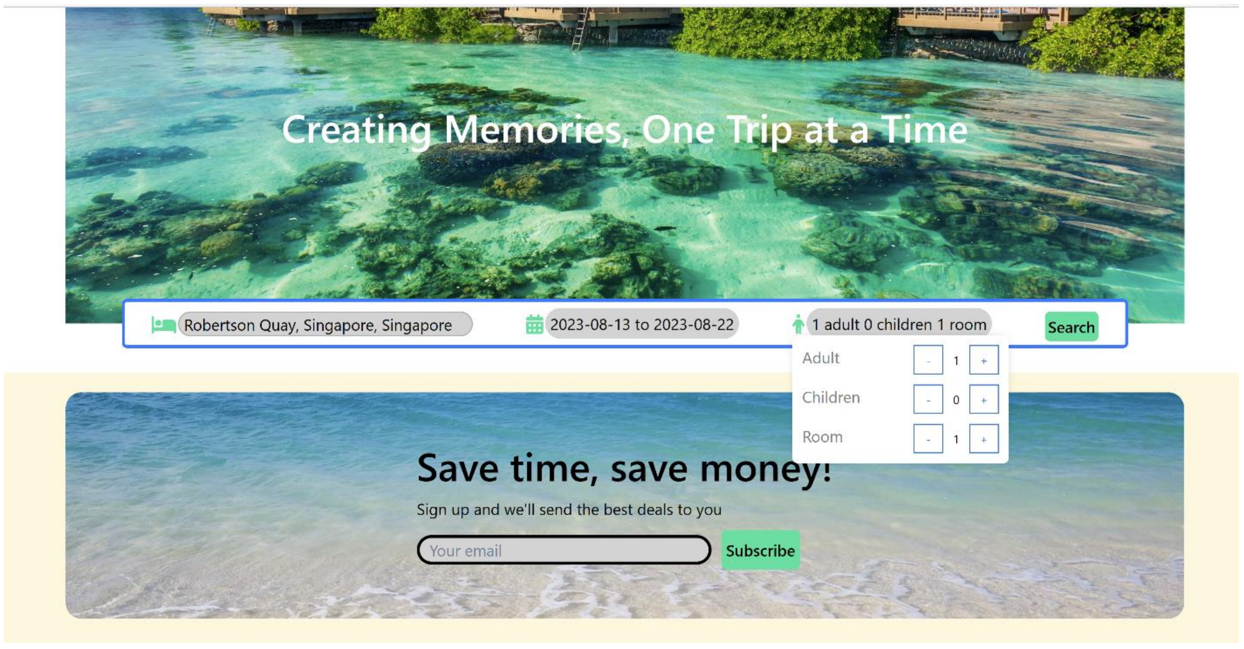

Booking Website
Singapore
Cities Countries Featured Currency jon@gmail.com
Logout
Creating Memories, One Trip at a Time
Robertson
2023-08-13 to 2023-08-14
1 adult 0 children 1 room
Search
ve time, save money!
and we'll send the best deals to you
USD/SGD +0.19%
Search
ENG US
9:57 pm 13/8/2023

Robertson Quay, Singapore, Singapore
Robertson Stadium, Houston, TX, United States
Robertson Walk, Singapore
Roberton Island, Russell, New Zealand
Roberson Museum and Science Center, Binghamton, NY, United States

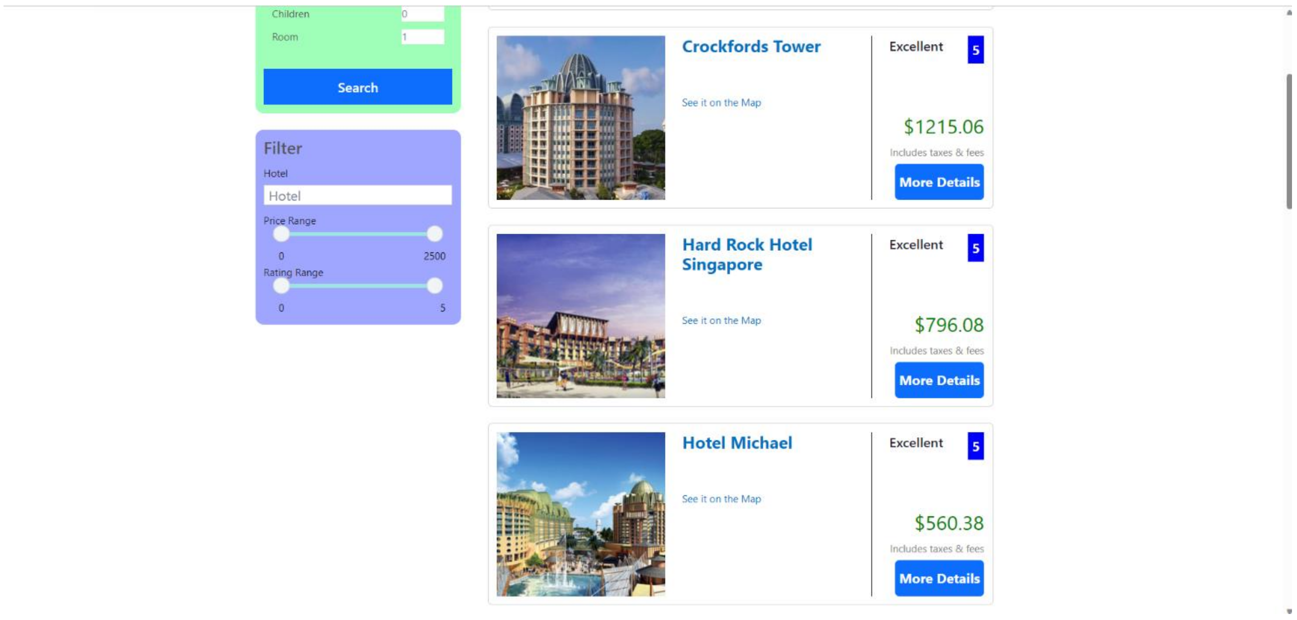
Step 3: User immediately selects one of these entries from the dropdown and indicates his period of stay on the calendar as shown below.


Creating Memories, One Trip at a Time
Robertson Quay, Singapore, Singapore
2023-08-13 to 2023-08-22
1 adult 0 children 1 room
Search
Aug 13, 2023
Aug 22, 2023
August
2023
Sun Mon Tue Wed Thu Fri Sat

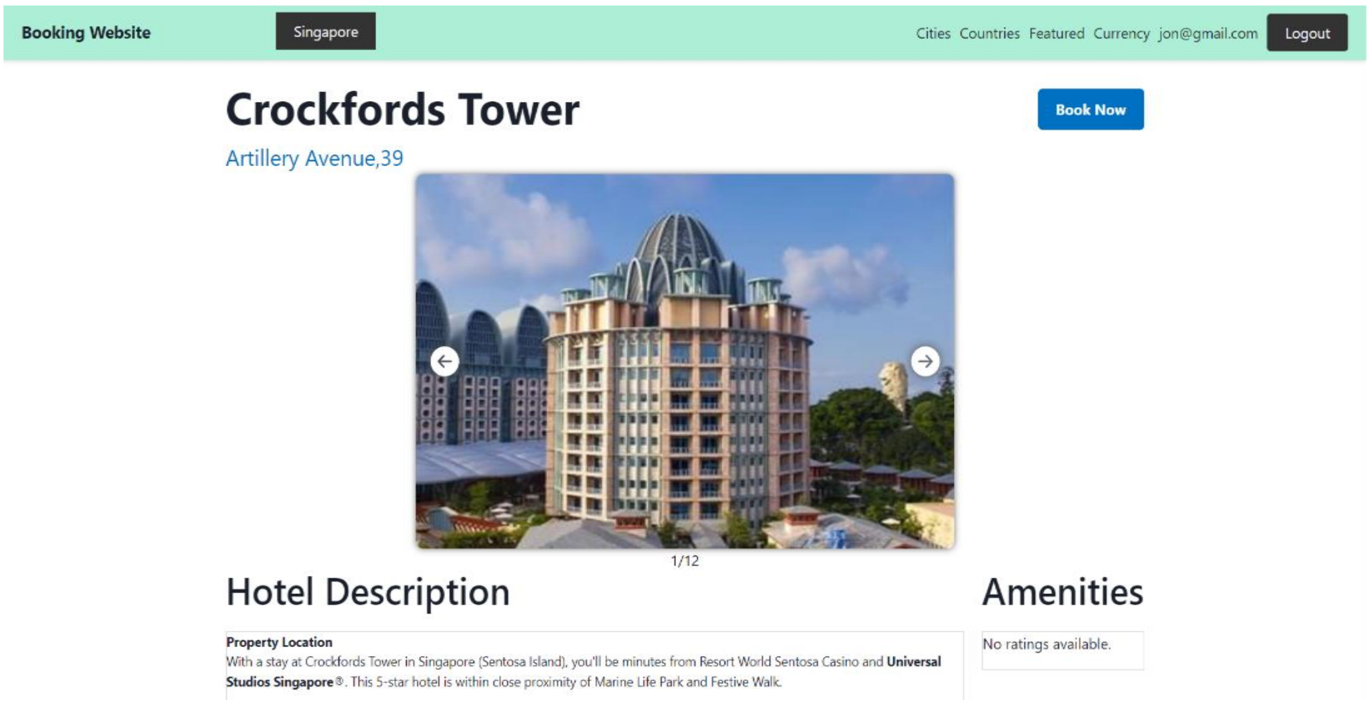
Step 4: User then selects the number of people accompanying him like one adult or one child by clicking on the relevant '+' or '-' icons as shown below and finally clicks on the 'Search' button as shown below.


Creating Memories, One Trip at a Time
Robertson Quay, Singapore, Singapore
2023-08-13 to 2023-08-22
1 adult 0 children 1 room
Search
Adult - 1 +
Children - 0 +
Room - 1 +
Save time, save money!
Sign up and we'll send the best deals to you
Your email
Subscribe

Step 5: User then clicks under 'More Details' related to the hotel of his choice. Let's assume the user selects the Crockford tower under Resorts World Sentosa



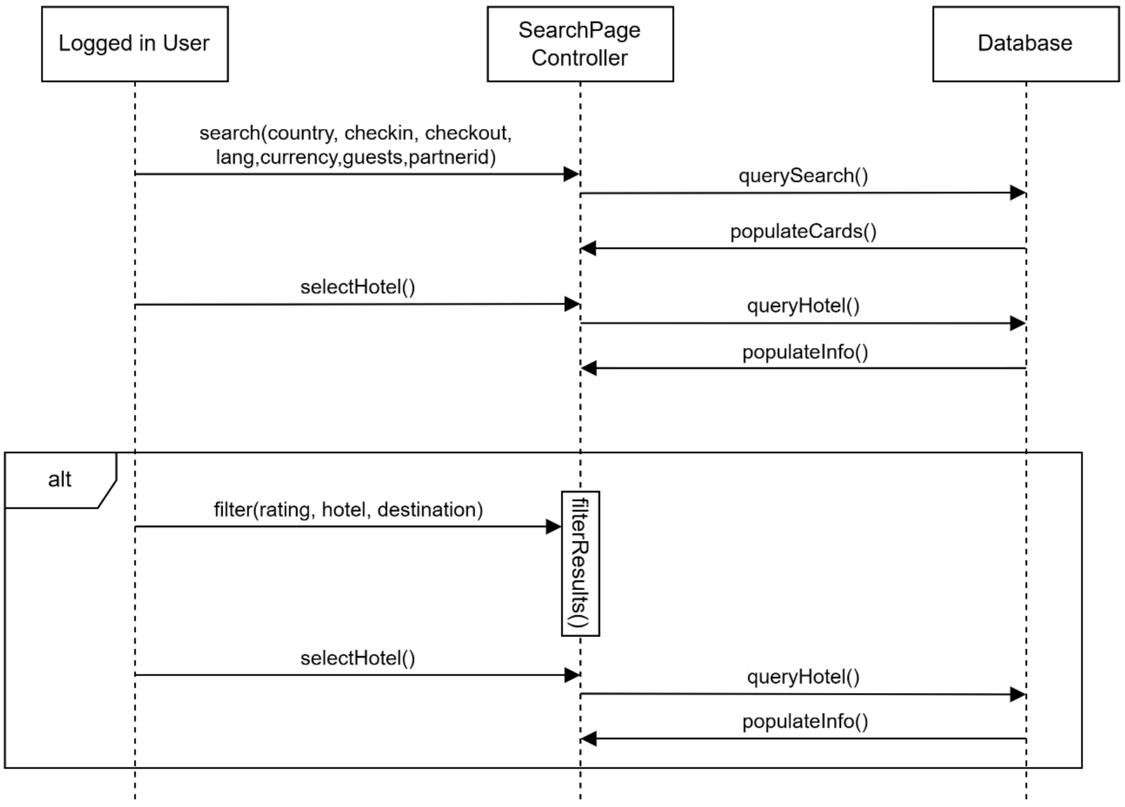
Step 6: User gets directed to a page where he can view hotel information, the cost of its various facilities and the amenities present in the hotel as well as make his booking



Property Location
With a stay at Crockfords Tower in Singapore (Sentosa Island), you'll be minutes from Resort World Sentosa Casino and **Universal Studios Singapore**®. This 5-star hotel is within close proximity of Marine Life Park and Festive Walk.

No ratings available.

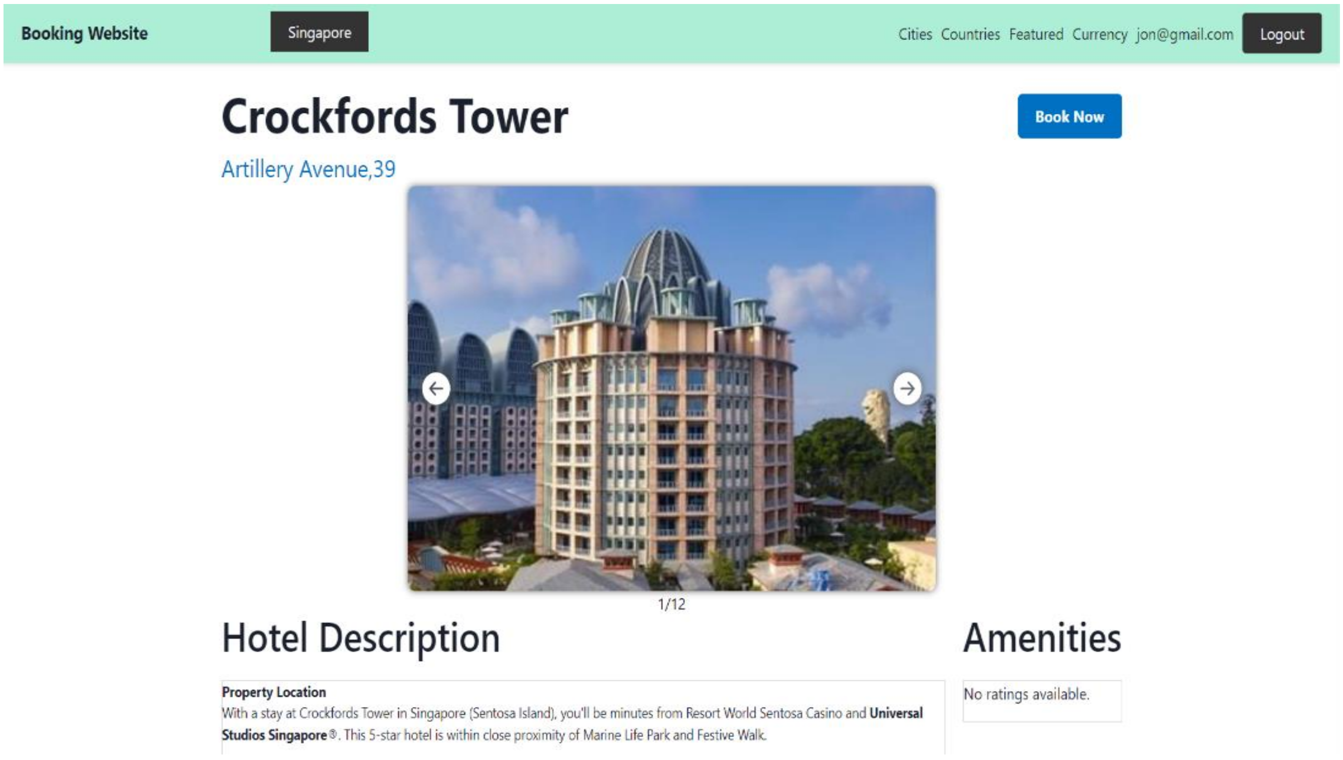
ID	HotelBooking_UC_02
Name	Search Hotels
Objective	User to search for a room that is suitable for them based on different filters
Pre-conditions	1. User has selected search bar
Post conditions	Success: 1. User must be able to view a suggested list of room to book from which fits the search filters Failure: 1. User's search requirements are not found
Actors	Primary 1. User Secondary 1. Hotel API 2. Database
Trigger	User clicks on search bar
Normal flow	1. User types in destination 2. System will provide auto-completion/suggestion based on User's query 3. User chooses check-in and check-out dates for stay 4. User selects number of guests, number of children and number of rooms 5. User views all suggested rooms that fit their criteria
Alternative flow	User searches for an invalid location or search result has no listings 1. An empty screen to show no listings 2. System returns to step 1 for normal flow and require user to search destination again
Interacts with	Hotels Database Hotel API


Logged in User
SearchPage Controller
Database
search(country, checkin, checkout, lang,currency,guests,partnerid)
querySearch()
populateCards()
selectHotel()
queryHotel()
populateInfo()
alt
filter(rating, hotel, destination)
filterResults()
selectHotel()
queryHotel()
populateInfo()

2.3 Case 3: Use Case Document, Sequence Diagrams and UI Illustration

2) User makes payment for the hotel booking--> UI covers for both the hotel booking, view existing booking information and making payment use cases

Step 1: After getting to step 6 of the search hotel use case, user now clicks on the 'Book Now'.



Crockfords Tower

Artillery Avenue,39

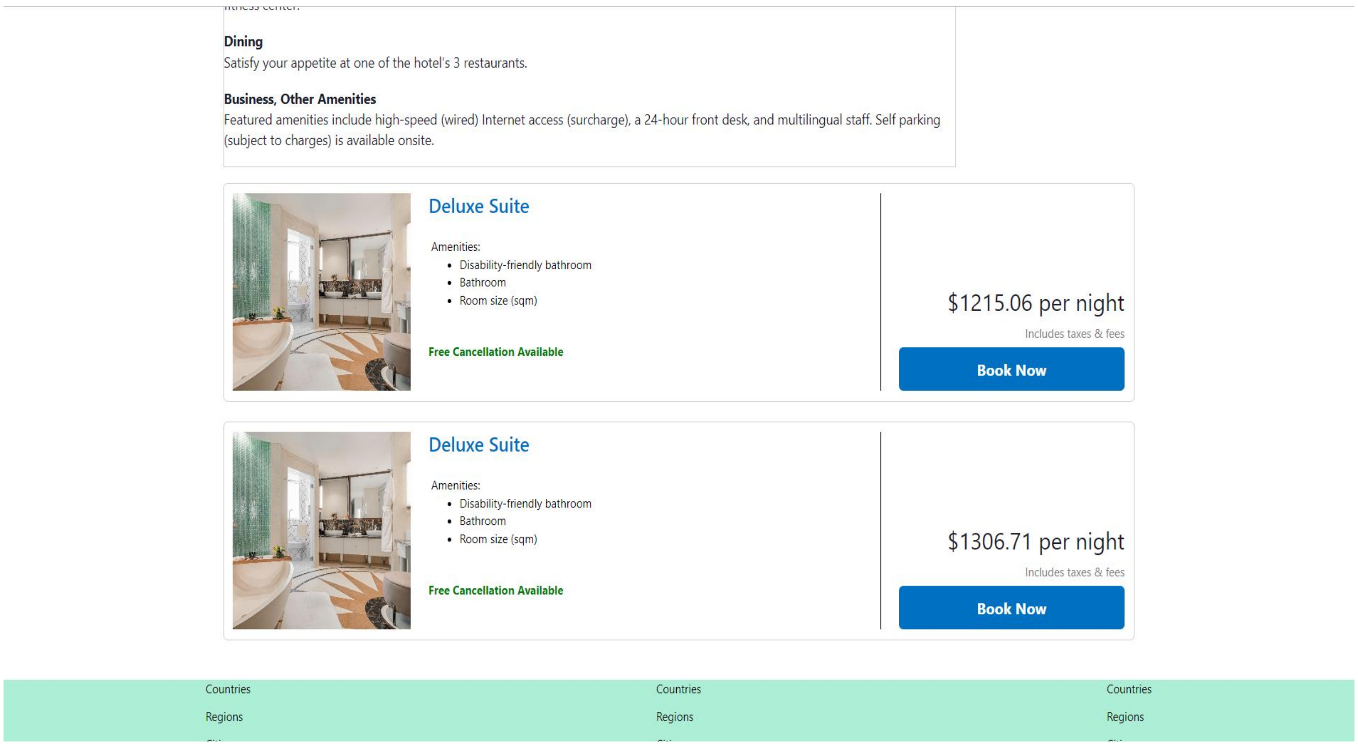
Hotel Description

Property Location
With a stay at Crockfords Tower in Singapore (Sentosa Island), you'll be minutes from Resort World Sentosa Casino and **Universal Studios Singapore**®. This 5-star hotel is within close proximity of Marine Life Park and Festive Walk.

Amenities

No ratings available.

Step2: The effect of step 1 scrolls him to a page where he can make his relevant bookings. In here, let’s assume the user selects the Deluxe Suit for example. User then presses on ‘Book Now’.



Dining
Satisfy your appetite at one of the hotel's 3 restaurants.

Business, Other Amenities
Featured amenities include high-speed (wired) Internet access (surcharge), a 24-hour front desk, and multilingual staff. Self parking (subject to charges) is available onsite.

Deluxe Suite

Amenities:
- Disability-friendly bathroom
- Bathroom
- Room size (sqm)

Free Cancellation Available

$1215.06 per night
Includes taxes & fees
Book Now

Deluxe Suite

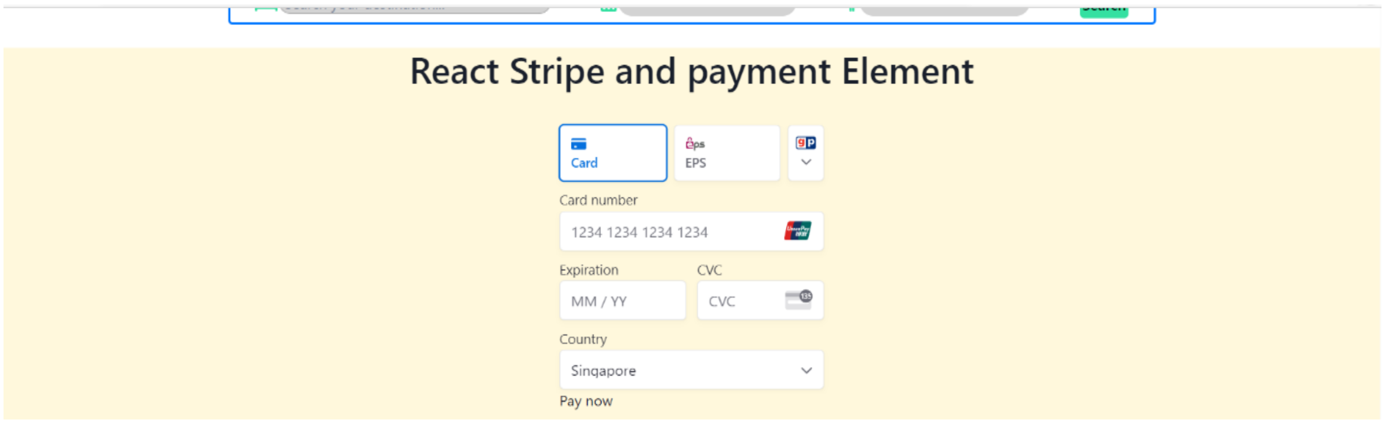
Amenities:
- Disability-friendly bathroom
- Bathroom
- Room size (sqm)

Free Cancellation Available

$1306.71 per night
Includes taxes & fees
Book Now

Countries Countries Countries
Regions Regions Regions

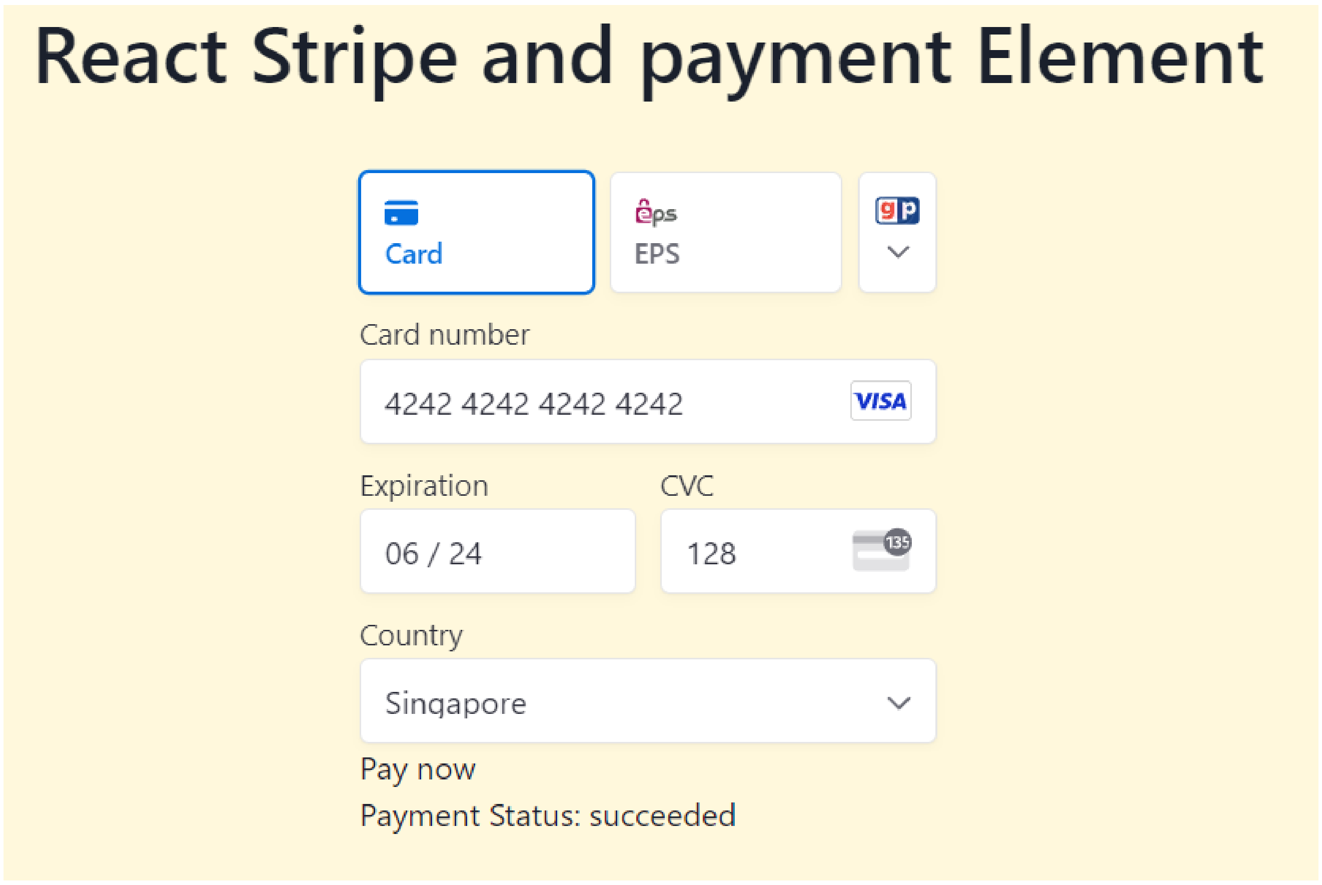
Step3: User now makes payment for the Deluxe Suit that he selected at Resorts World Sentosa.



React Stripe and payment Element

Card number: 1234 1234 1234 1234
Expiration: MM / YY
CVC: CVC
Country: Singapore
Pay now

Step4: If user's payment succeeds, a success message is displayed on screen as shown.


React Stripe and payment Element

Card number: 4242 4242 4242 4242

Expiration: 06 / 24

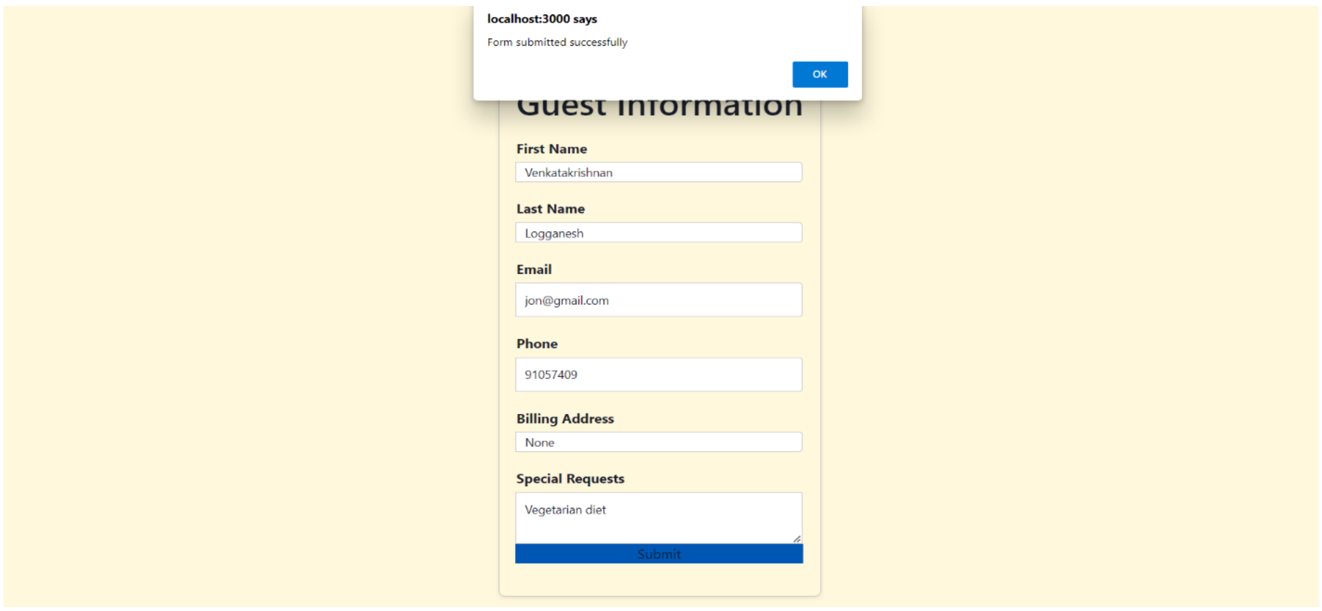
CVC: 128

Country: Singapore

Pay now

Payment Status: succeeded

Step5: User then enters all his details in the Guest Information form like his first name, last name, email, phone as well as special requests to hotel and clicks the 'Submit'. If user's submission is successful, a message is displayed on screen as shown below.


localhost:3000 says

Form submitted successfully

Guest Information

First Name: Venkatakrishnan

Last Name: Logganesh

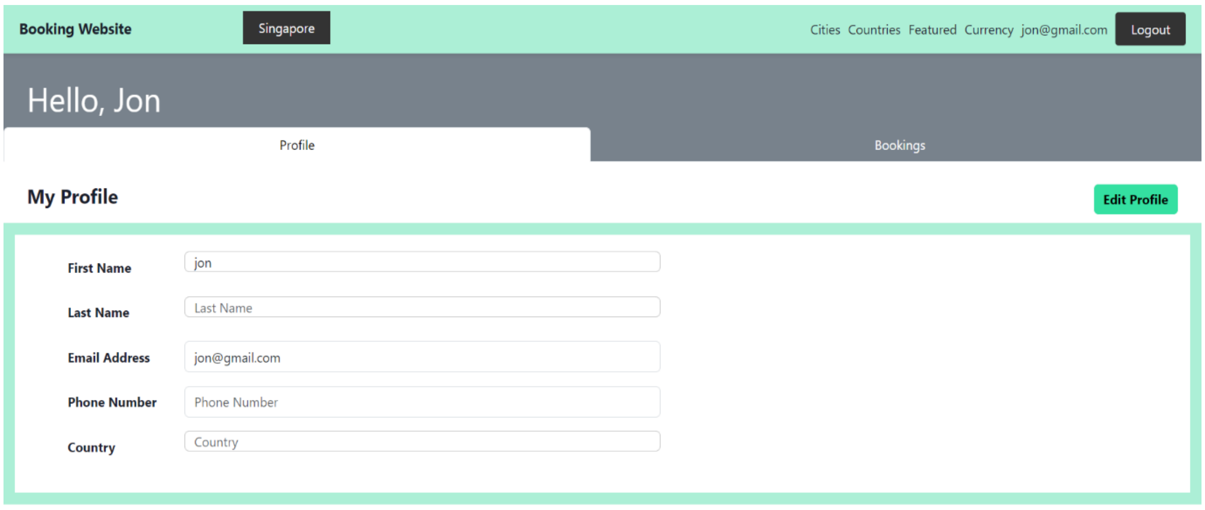
Email: jon@gmail.com

Phone: 91057409

Billing Address: None

Special Requests: Vegetarian diet

Step6: To view all his recent bookings, the user selects on the bookings tab as shown below.

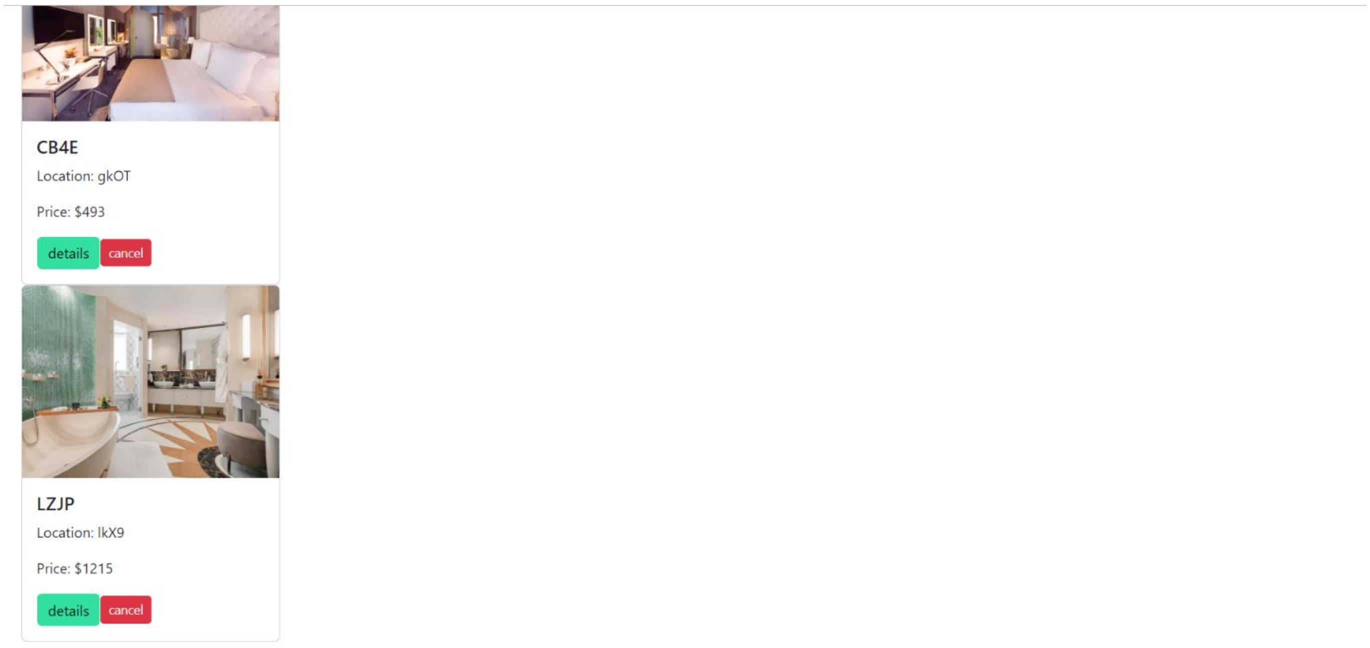


Hello, Jon

My Profile

First Name	jon
Last Name	Last Name
Email Address	jon@gmail.com
Phone Number	Phone Number
Country	Country

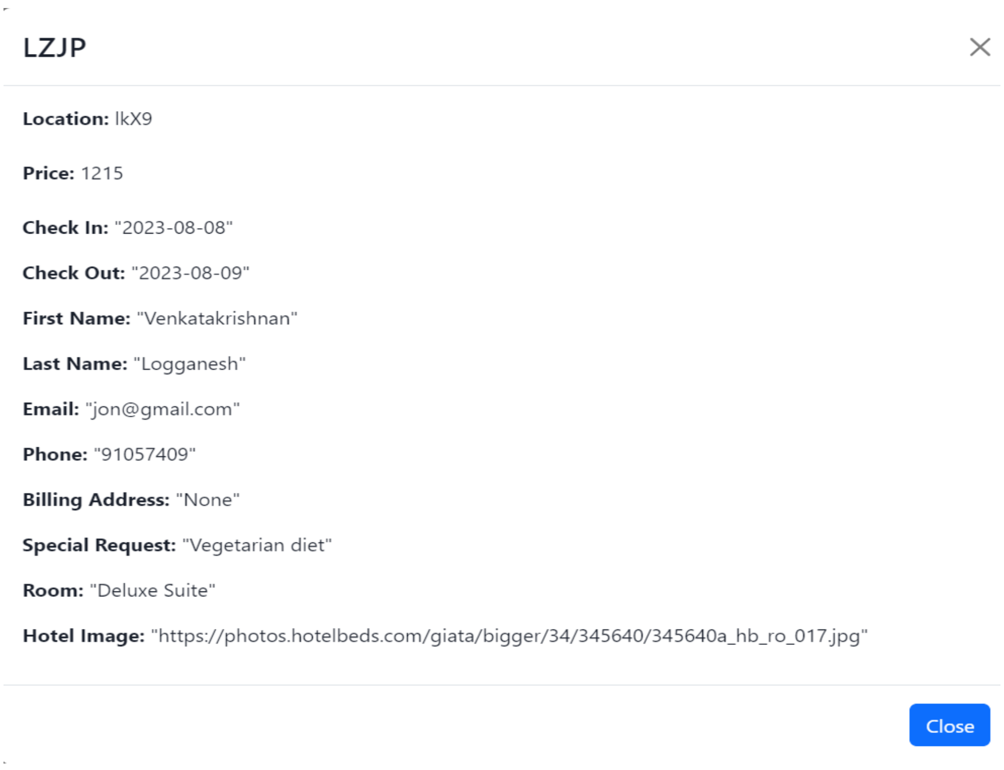
Step7: User must scroll on the way to the bottom of the page and click on 'details' to look at information regarding his most recent booking.



CB4E
Location: gkOT
Price: $493

LZJP
Location: lkX9
Price: $1215

Step8: Now the user sees all the details of the most recent submission that he made at the guest information page at step 5 being displayed on his screen as shown below.



LZJP

Location: IkX9

Price: 1215

Check In: "2023-08-08"

Check Out: "2023-08-09"

First Name: "Venkatakrishnan"

Last Name: "Logganesh"

Email: "jon@gmail.com"

Phone: "91057409"

Billing Address: "None"

Special Request: "Vegetarian diet"

Room: "Deluxe Suite"

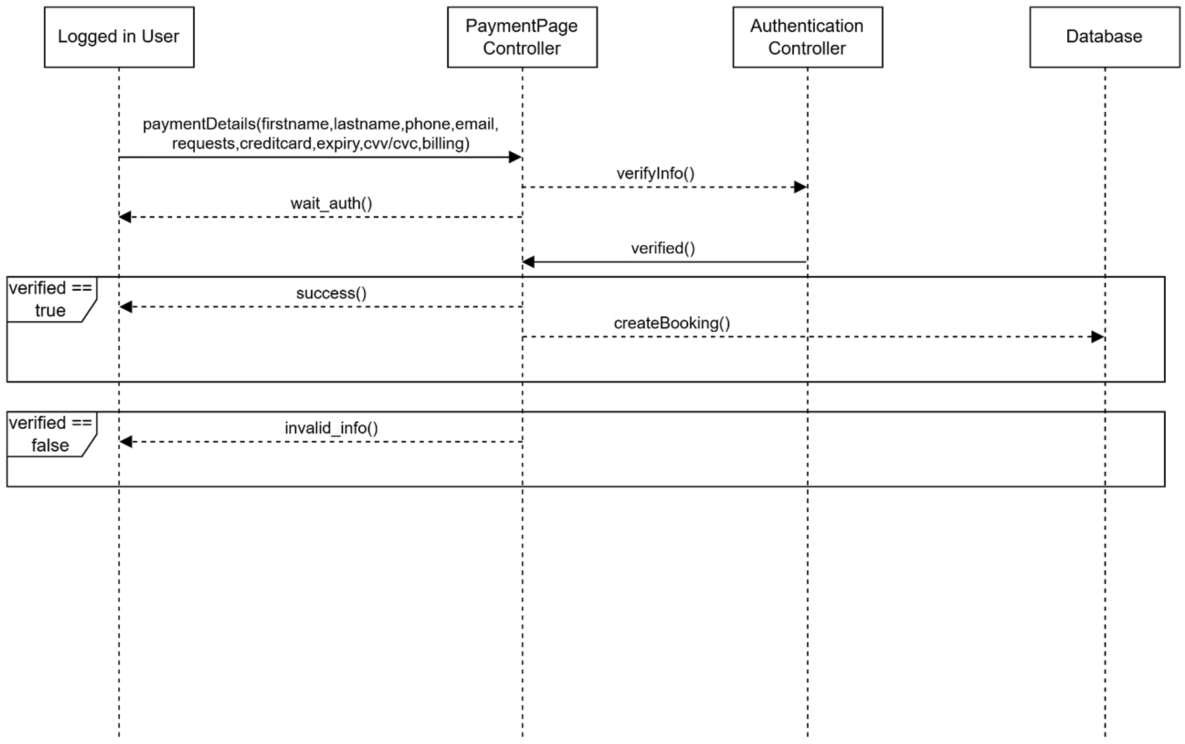
Hotel Image: "https://photos.hotelbeds.com/giata/bigger/34/345640/345640a_hb_ro_017.jpg"

Step9: *User then clicks 'Close' and concludes this booking journey.*

2.3 Case 4: Use Case Document, Sequence Diagrams and UI Illustration

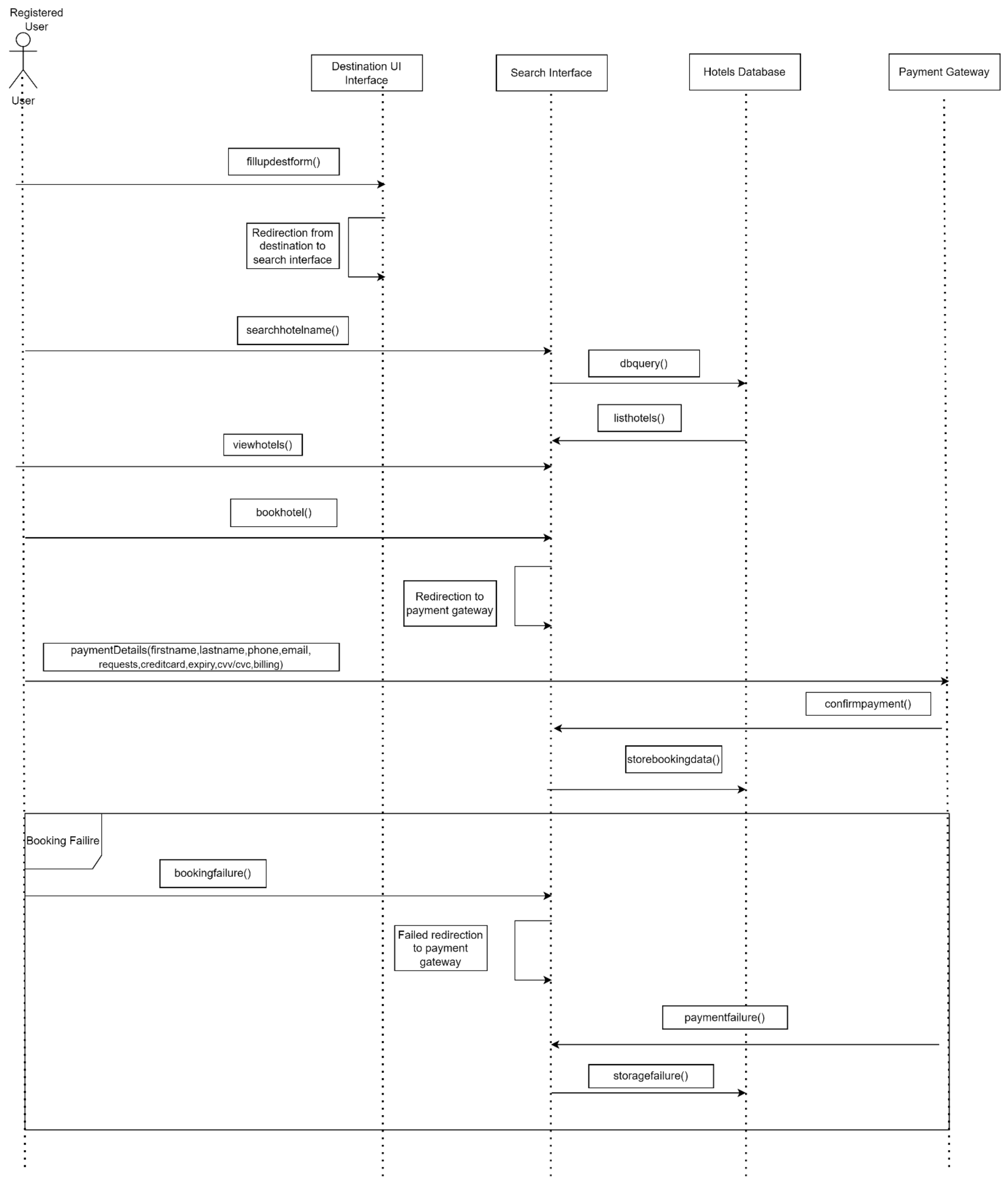
ID	HotelBooking_UC_04
Name	Make Payment
Objective	To submit personal particulars/payment information to pay for booking
Pre-conditions	User has selected a hotel of their interest, dates of stay and number of rooms
Post-conditions	Success: User has successfully paid for the booking and receives a confirmation message Failure: User has inputted invalid personal/payment information Payment is rejected and redirected back to “Make Payment” page
Actors	Primary: User Secondary: System & Database Booking API
Trigger	User clicks on “Book Hotel” button
Normal flow	Successful booking 1. User fills up personal particulars (Name, Phone number, Email, Special Requests to Hotel, Billing Address) and Payment Information 2. User clicks on “Submit” button that calls a “Create Booking” API 3. API creates a booking in the Database 4. System confirms booking is created 5. User receives a confirmation message
Alternative flow	Failure in making payment: 1. User inputs invalid personal particulars / Payment Information 2. API fails to create a booking due to invalid information 3. System confirms booking is not successfully created 4. User receives an error message and is redirected back to the

Interacts with	“Make Payment” page Bookings database and Accounts database



2.3 Case 5: Use Case Document, Sequence Diagrams and UI Illustration

ID	HotelBooking_UC_05
Name	Book hotel room
Objective	User wants to book a particular hotel room of his specific location on the specific dates.
Pre-conditions	User must be logged into Hotel Booking system User has already searched for destination with his check in/check out dates and number of rooms/adults/children User has clicked on one of the hotels in the search results User is viewing an individual hotel room page
Post-conditions	Success 1. User booking is successful. Failure 1. Unable to view the room's availability in a specific hotel in a specific location on the specific dates 2. Inaccurate details of hotel information or the hotel is fully booked.
Actors	Primary 1. User Secondary 1. Hotel booking administrator
Trigger	User wants to make a booking.
Normal flow	1. After viewing, user makes booking by pressing on the book now button. 2. User is directed to payment page. 3. If user makes payment by filling up the form 4. User then fills up the form with his first name, last name, phone number, billing address and special requests. 5. User submits form and booking successfully made. 6. Booking information is sent to booking's website mongo database
Alternative flow	Failure to display booking information: 1. Lag issues result in user not being able to confirm his booking. 2. Payment gateway is not working. Booking results are not showing up even though payment is made.
Interacts with	User interface consisting of buttons redirecting him to the relevant webpages.

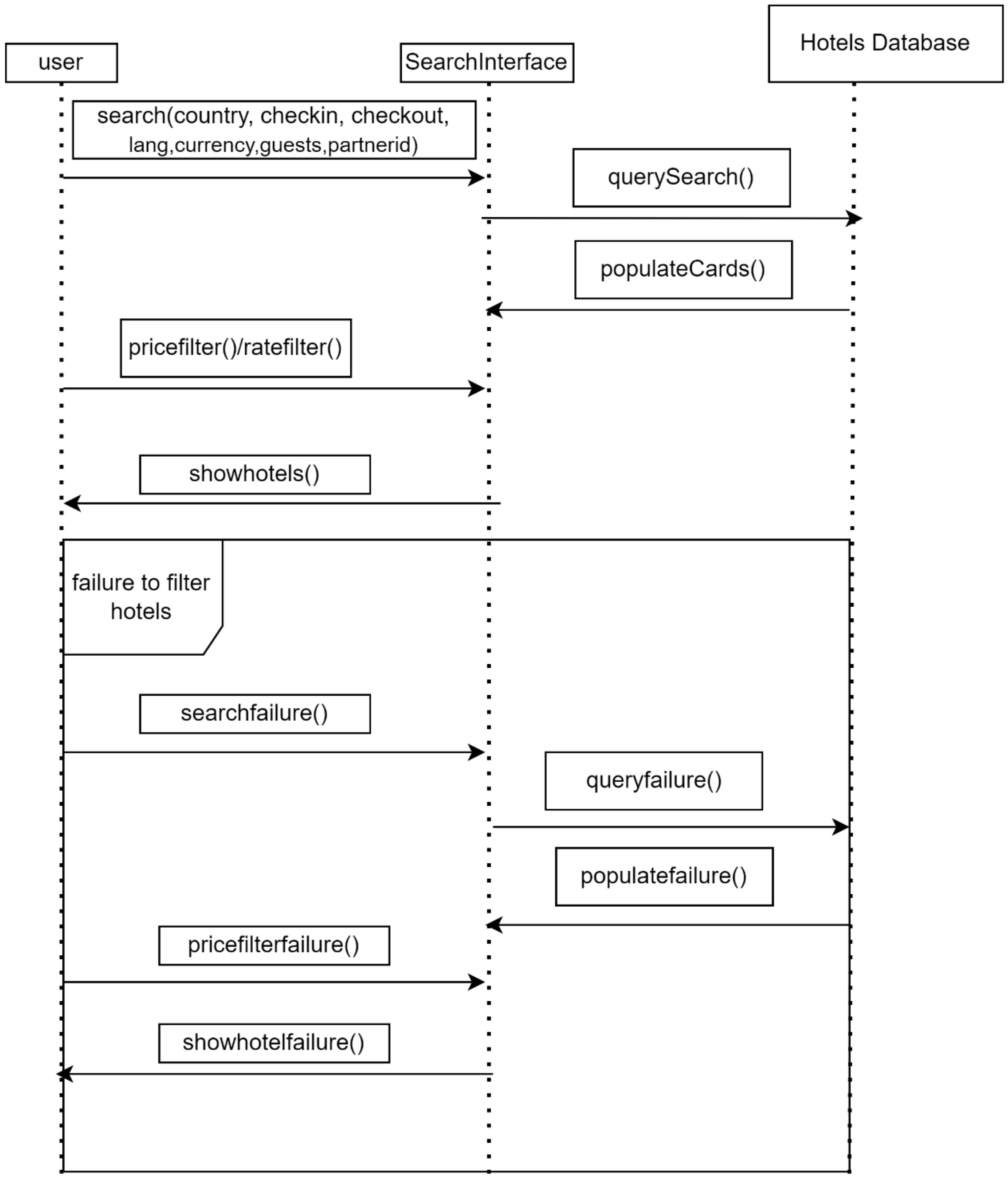

Registered User
User
Destination UI Interface
Search Interface
Hotels Database
Payment Gateway
fillupdestform()
Redirection from destination to search interface
searchhotelname()
dbquery()
listhotels()
viewhotels()
bookhotel()
Redirection to payment gateway
paymentDetails(firstname,lastname,phone,email, requests,creditcard,expiry,cvv/cvc,billing)
confirmpayment()
storebookingdata()
Booking Failire
bookingfailure()
Failed redirection to payment gateway
paymentfailure()
storagefailure()

2.3 Case 6: Use Case Document, Sequence Diagrams and UI Illustration

ID	HotelBooking_UC_06
Name	Delete User information / account
Objective	User wants to purge personal information from database
Pre-conditions	User must have an existing account User must be logged into Hotel Booking System
Post-conditions	Success: User account is deleted from database Failure: Database unable to delete user account
Actors	Primary: User Secondary: Account Database
Trigger	User wants to delete personal information due to inactivity on site
Normal flow	1. User clicks profile link 1. Proceeds to account security tab 3. Clicks delete account option 4. Confirms action 1. Logged out of account (should be unable to log back in)
Alternative flow	Failure to delete account: User does not authenticate his credentials and confirm his action of deleting account successfully, disabling him from deleting the account.
Interacts with	Accounts database

2.3 Case 7: Use Case Document, Sequence Diagrams and UI Illustration

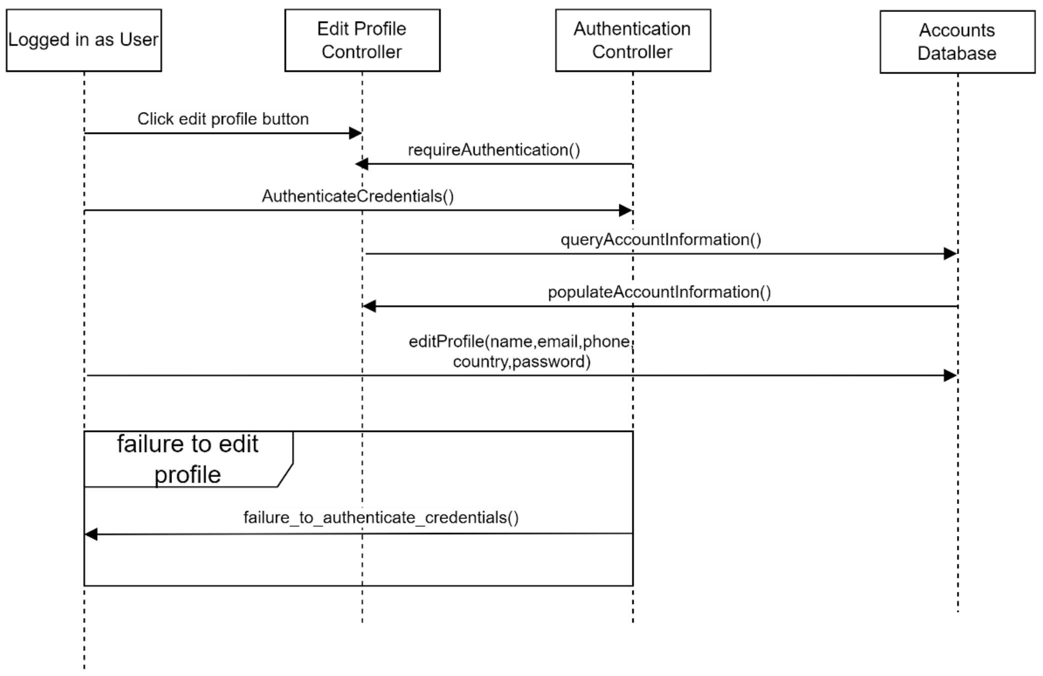
ID	HotelBooking_UC_07
Name	Search Hotel by Price
Objective	User wants to filter given list of hotels according to price range
Pre-conditions	User has searched for hotels according to requirements: destination, dates of stay and number of guests
Post-conditions	Success: List of hotels are filtered according to price range Failure: List of hotels is not filtered according to price range
Actors	Primary: User Secondary: Hotel Database
Trigger	User clicks on slider handles drags accordingly for filtering by price range
Normal flow	Success: 1. List of hotels is re-rendered 2. User browses new set of displayed hotels
Alternative flow	Failure of filter function: 1. Webpage does not render the hotel list based on the selected price range
Interacts with	Hotels database


user
SearchInterface
Hotels Database
search(country, checkin, checkout, lang,currency,guests,partnerid)
querySearch()
populateCards()
pricefilter()/ratefilter()
showhotels()
failure to filter hotels
searchfailure()
queryfailure()
populatefailure()
pricefilterfailure()
showhotelfailure()

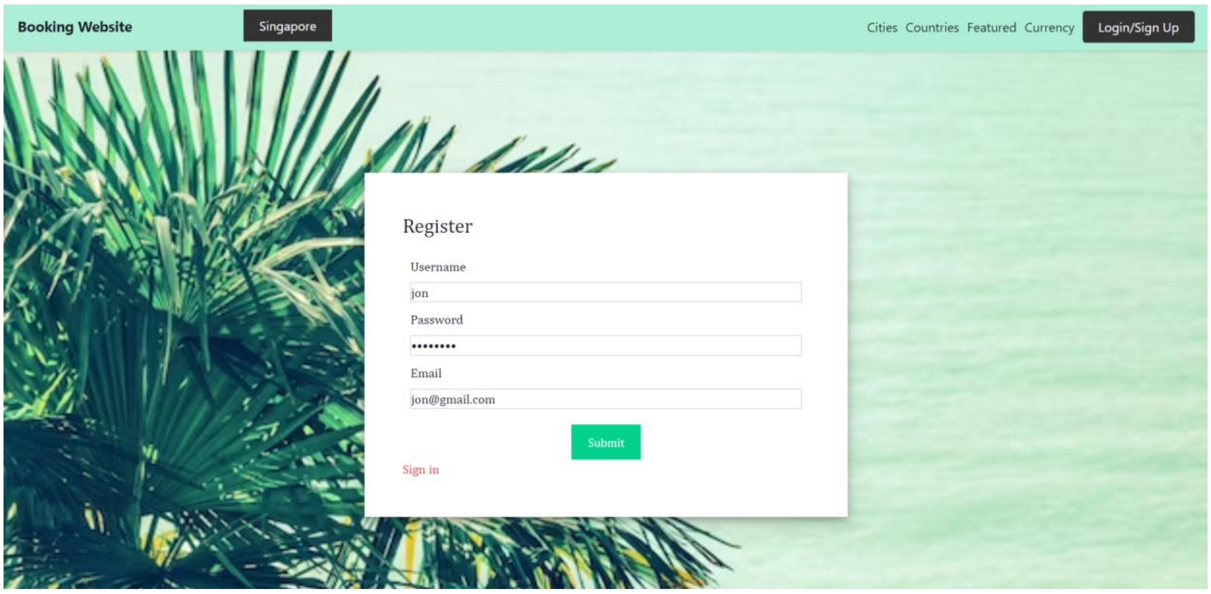
2.3 Case 8: Use Case Document, Sequence Diagrams and UI Illustration

ID	HotelBooking_UC_08
Name	Edit profile
Objective	User wants to edit his account information such as his first name, last name, email address, phone number, country and password
Pre-conditions	User must have an actively registered account in web application and must be logged into his account before he can edit his profile information.
Post-conditions	Success User can successfully update his account information for the relevant fields and successfully change his account information. Failure User is not able to edit his account details because of the failure of the 'Edit Profile' button interface
Actors	Primary : User Secondary : Hotel Database
Trigger	Use clicks on the edit profile button.
Normal flow	Success: 1.User clicks on his email address located at the top right-hand corner of the screen which directs him to the editing page. 2. User then clicks on the Edit Profile button which and goes on to edit the relevant information 3. After making all the relevant edits, the user must click on the 'Save Changes' button to successfully record all the modifications that he made to his account information so that they are preserved.
Alternative flow	Failure of login 1. User is not able to login to his account as he has forgotten his credentials. 2. Interface failure of the button functionalities in the backend causes the user to be unable to edit his profile, save the changes that he made to his account as well as refrains him from logging into his account.
Interacts with	Accounts database

Edit profile

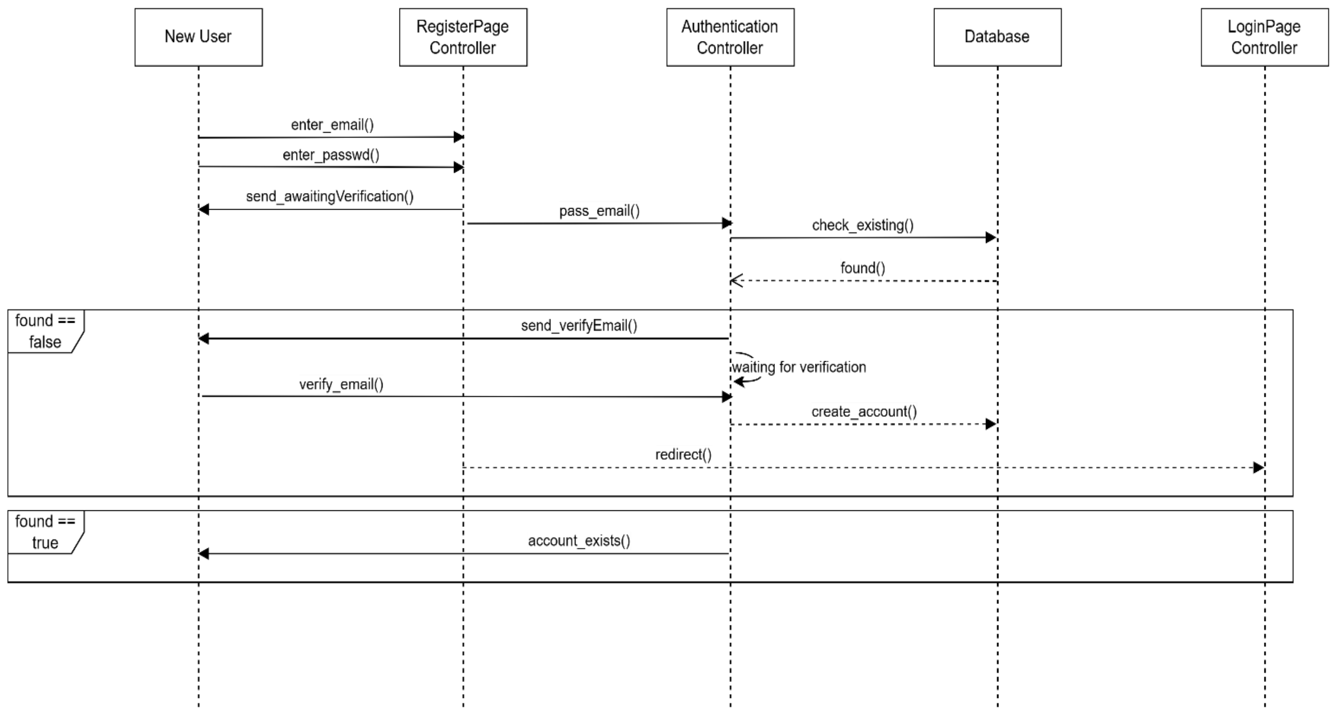

Logged in as User
Edit Profile Controller
Authentication Controller
Accounts Database
Click edit profile button
requireAuthentication()
AuthenticateCredentials()
queryAccountInformation()
populateAccountInformation()
editProfile(name,email,phone country,password)
failure to edit profile
failure_to_authenticate_credentials()

2.3 Case 9: Use Case Document, Sequence Diagrams and UI Illustration



ID	HotelBooking_UC_09

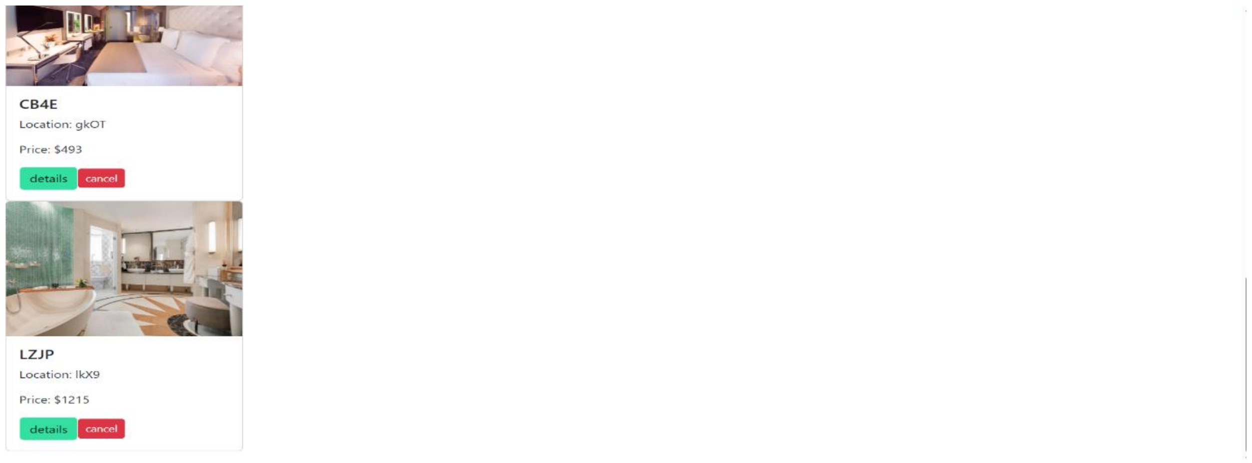
Name	Register profile
Objective	User wants to register his account information such as his first name, last name, email address, phone number, country and password
Pre-conditions	User must not have an actively registered account in web application
Post-conditions	Success User can type his account information into the relevant fields and successfully register his account information. Failure User is not able to register his account details because of lack of information provided User is not able to register as he already has an account associated with details provided
Actors	Primary : User Secondary: Hotel Database
Trigger Normal flow	1.User successfully enters the register account button. 2.User then keys in all the relevant information required for registration. 3.User then successfully registers his account upon clicking on register button
Alternative flow	Failure to register 1.User is not able to register his account as he has not filled in all required details 2. User is unable to register his account after filling in all required details as details are already associated with existing account.
Interacts with	Account database


New User
RegisterPage Controller
Authentication Controller
Database
LoginPage Controller
enter_email()
enter_passwd()
send_awaitingVerification()
pass_email()
check_existing()
found()
found == false
send_verifyEmail()
waiting for verification
verify_email()
create_account()
redirect()
found == true
account_exists()

2.3 Case 10: Use Case Document, Sequence Diagrams and UI Illustration

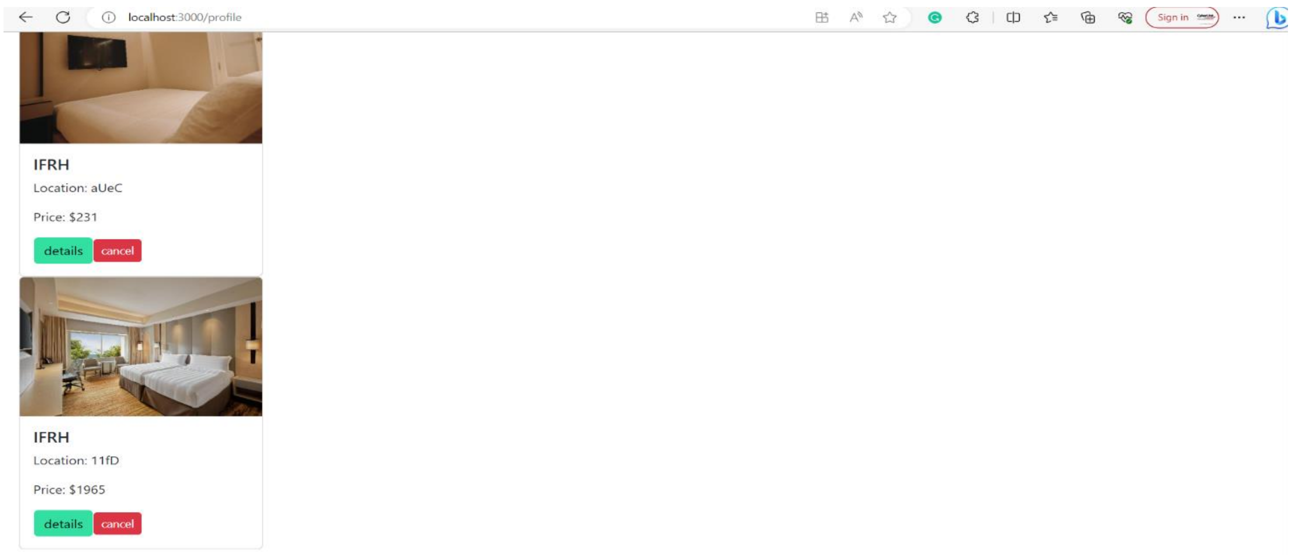
10)User cancels the booking.

Step1: Flowing from step 7 of the make hotel booking/make payment use case, user has to click on the ‘cancel’ button to remove his booking.


CB4E
Location: gkOT
Price: $493

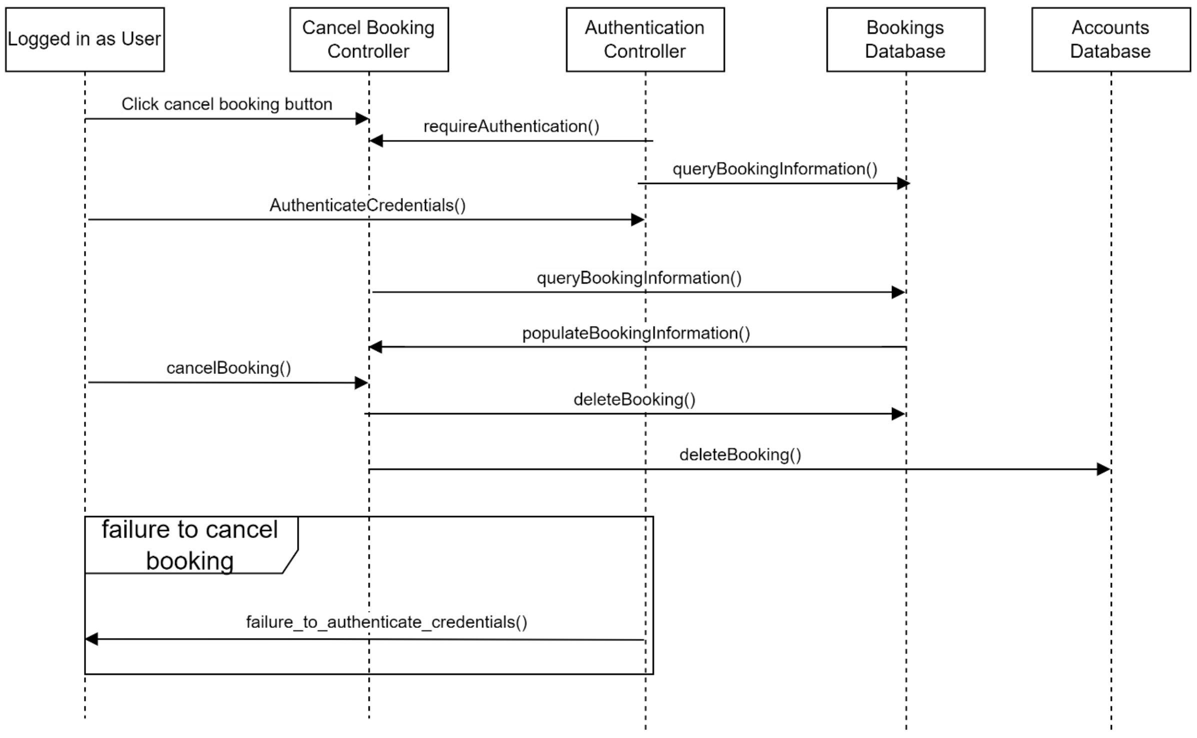
LZJP
Location: lkX9
Price: $1215

Step2: After cancellation, user follows step6 of the 'Make Payment’ use case to switch to the bookings tab. Now we see that ‘LZJP’ is removed from the bookings list.


localhost:3000/profile
IFRH
Location: aUeC
Price: $231

IFRH
Location: 11fD
Price: $1965

ID	HotelBooking_UC_10
Name	Cancel Booking
Objective	User wants to cancel the booking he created
Pre-conditions	User must have created a booking which exists in the web application and must have logged in to his account.
Post-conditions	Success: User can view and cancel his booking only if he is logged in and authenticates his cancellation Failure: User is not able to cancel his booking as he does not authenticate his account details accurately
Actors	Primary : User Secondary : Hotel Database
Trigger	User successfully logins to his account and presses the cancel booking button
Normal flow	User authenticates his cancellation and successfully cancels his booking.
Alternative flow	User is unable to cancel his booking as he does not authenticate his details accurately.
Interacts with	Bookings database and Accounts database


Logged in as User
Cancel Booking Controller
Authentication Controller
Bookings Database
Accounts Database
Click cancel booking button
requireAuthentication()
queryBookingInformation()
AuthenticateCredentials()
queryBookingInformation()
populateBookingInformation()
cancelBooking()
deleteBooking()
deleteBooking()
failure to cancel booking
failure_to_authenticate_credentials()

2.4 Test Case Documents

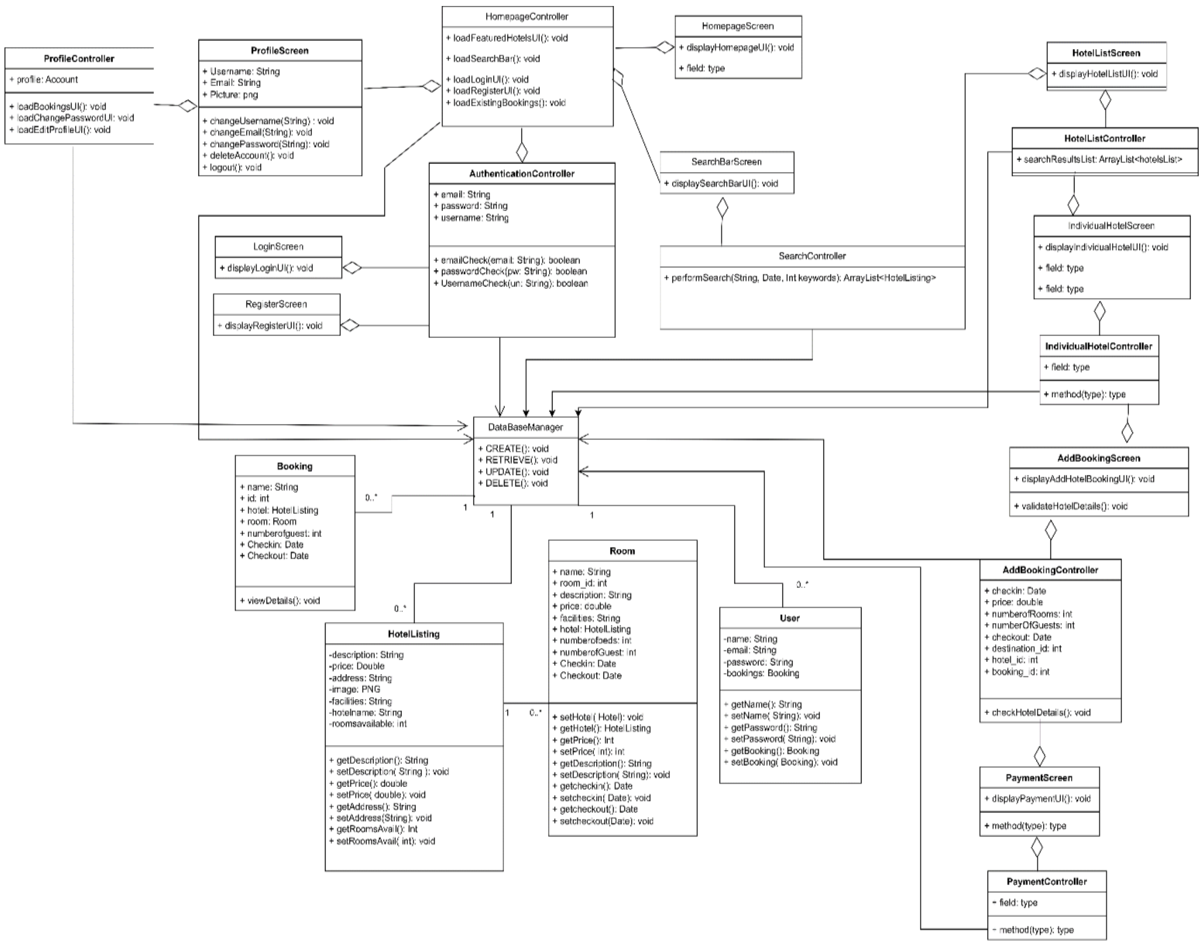
Test case ID	TC 1
Name	Search Hotels
Objective	User to search for a room that is suitable for them based on different filters
Pre-conditions	User has selected search bar
Event Sequence	1)User fills in destination and clicks on a destination, user clicks checkin date and checkout date, user adds number of rooms he/she would like to book, user adds number of adults, number of children, 2)User clicks search
Input	User navigates to https://localhost:3000
Output	Search bar is displayed
Input	User types destination, clicks on checkin and checkout dates, fills up number of adults, children and rooms
Output	Search bar has suggestions for destinations, calendar dates, number of adults, children and rooms are filled up and displayed to the user
input	User clicks search
output	User is directed to next page https://localhost:3000/hotels which displays a list of hotels available based on the information provided by the user
Post conditions	User either succeeds and has a list of hotels dynamically displayed based on the information provided, each hotel should show the “prices from”, hotel name, and hotel rating to the user Failure, user sees “hotels unavailable ” meaning no hotels are available for the period provided by the user

Test case ID	TC2
Name	Book hotel rooms
Objective	User is trying to book a hotel after he has confirmed his booking.
Pre-conditions	User has a registered an account and user is logged in.
Event Sequence	1)User is viewing the individual hotel page 2)User then presses on the book now button. 3) User is directed to payment page 4) if payment is successful, user has successfully booked Otherwise User is told payment is unsuccessful, booking fails, user is redirected to individual hotel page

Input	User navigates to https://localhost:3000/hotels/:id
Output	Hotels information is displayed such as description, amenities, rooms available and room pricing.
Input	User presses book now button for each room
Output	User is directed to https://localhost:3000/payment
Input	User enters correct payment information
Output	Payment success and user makes a successful booking
Post conditions	User books hotel successfully Or User booking fails

Test case ID	TC 3
Name	Make payment
Objective	User wants to make payment for his booking
Pre-conditions	User is logged in, user is in the individual hotel page, user has an active credit card account to which his payments can be attached.
Event Sequence	1)User clicks on book now button 2) user is redirected to payment page 3) user fills in payment details and payment succeeds. Otherwise User fills in payment details and payment fails
Input	User clicks on Book Now button in the page https://localhost:3000/hotels/:id and is redirected
Output	https://localhost:3000/payment user is directed to payment page where he sees a payment module to add card information and a form to collect user infromation
Input	User fills up payment information and user information, user clicks on submit button
Output	User is directed to the payment complete page https://localhost:3000/completion
Post conditions	User succeeds and sees “payment complete”, database stores all this information under profile under bookings Failure, user sees “payment failed to complete ” meaning payment didn’t complete

2.5 Class Diagrams



3. Non-Functional Requirements

In this project, there are a couple of non-functional requirements: Safety, Auditability, prohibition of illegal access to certain functionalities and error handling and debugging.

Personal information

1) Safety

User’s personal credentials such as their passwords are stored in an encrypted format using the bcrypt module the in the Mongo Database to prevent manipulation by Database managers. Moreover, encrypted passwords also guarantee safety and security since it assures the users that their data is not being leaked to external parties and modified without their consent. This ensures that users can only change their passwords upon their consent and approval. Therefore, the DB manager has no way of accessing user password, promoting confidentiality of user data. It avoids certain financial and legal consequences that the Ascenda faces in the long run.

2)Auditability

Every time a user makes a submission of his guest information at the payment page.The bookingId associated with this booking gets stored under his booking History for record keeping purposes. This ensures that the customer's mutual trust in the system is not compromised since all his records are maintained in the database. Information regarding who changes and owns a particular record in the database is controlled. In the application the customers' records requested are being well-organized, complete and compliant with the accounting standards.

3)Prohibition of Illegal access to certain functionalities

The user authentication at the login page ensures that only users who officially registered in the web app can access certain functionalities of the app. For example, an unregistered user will not be able to directly navigate to the payment or book hotel rooms.

Error Handling and Debugging

1) Modularity and Single Responsibility Principle

The different modules are well-separated from one another ensuring that each module serves only it single purpose. example, the authentication module is well-separated from the payment module to ensure that any error that occurs during code testing can be handled within the section of code operating that module.

2) Server and Client Web Architecture

Server module and client module is less tightly coupled and well separated from one another to ensure that the client deals with the frontend and the server deals with the routing in the backend. These two modules interact well with one another to promote the efficient rendering and functionality of the web app.

3) Coding and Documentation

The codes are well-documented with comments to ensure more coordinated time is spent on the testing of the system rather than understanding the workings of the code.

4) Concurrency and Asynchronous programming

All fetch() requests and axios.get() requests have await keyword or .then() or promise.all(). Fetch() starts a request and returns a promise. if the request fails, the promise is rejected. By utilising the await keyword, we will allow the asynchronous function to pause until the request is complete. Fetch() however doesn't throw errors when server returns a bad status. Fetch rejects only if a request cannot be made or response cannot be retrieved. We resolve this by using .catch() and throwing the error manually. Some require multiple request, thus we use promise.all() helper function to help us call parallel requests and if one fails an error will be thrown.

4. Testing

4.1 Unit Testing

Using Jest, we created and conducted both positive and negative unit test cases for functions regarding the searching of hotels, destination id searching, booking of hotel rooms and Accounts. For negative test cases, we confirmed the failure of the test case by entering inputs that, when queried, are not valid in our database. For positive test cases, we tested the various functions by giving inputs that are valid and check the results for their functions using expect(result).toBe(expectedresult), confirming the query from our mongoDB succeeded.

Bugs found from Unit testing and system testing	What will happen	Resolution
Localhost:3000/login Login has wrong email or password error not handled properly	Runtime error will occur	Adding a .catch() to the fetch() request on the front end, now the app will not crash but display 'invalid email or password'
Localhost:3000 Search dropdown bar does not refresh data when the user input changes	Dropdown bar gets extended with multiple repeated values	Adding a key to the .map() will remove any duplicates
Localhost:3000 search engine submit button will submit even if user has not clicked the dropdown bar	Error will be thrown as the ascenda api cannot submit a destination without location id	Adding a use state that checks where dropdown bar must be clicked at least once otherwise a prompt will come up when user presses submit button
Submit button in localhost:3000/payment allows not logged in user to submit the form	Error will be thrown as database cannot create unknown entry for user	Adding a use state to check if user is logged in or not
Localhost:3000/register is able to register a duplicate email	This will cause same email login which should not be the case	Adding a unique field to the backend database will resolve this
Localhost:3000/hotels will send multiple queries and get locked out	This will return results to be [] even if there is originally results.	Adding a setTimeout and sleep() function allows us to always get back results but at a maximum loading time of 30 seconds
Localhost:3000 If user types in wrong destination such as Singgpore instead of Singapore	No dropdown bar results will appear	The mongodb is set to autocomplete function, so even if user types singgpore, it will get the closest destination it can find to singgpore

4.2 Selenium Testing

The Selenium WebDriver library is employed to automate webpage interactions, effectively simulating user actions within our hotel booking application. These actions encompass tasks such as logging in, interacting with a calendar, performing searches, and navigating through diverse elements on the page. For the implementation of the WebDriver with the Google Chrome browser, the Chromedriver module is included.

Within this context, an asynchronous function is defined, denoted as "trial." Its purpose is to initialize a WebDriver instance tailored for the Chrome browser. In this context, the variable named "login_url" retains the URL fragment associated with the login page.

Following this setup, the WebDriver issues a GET request to "http://localhost:3000" and patiently waits for one second to allow the page to fully load. Subsequently, as the page becomes responsive, the WebDriver utilizes the "findElement" method to navigate to the login URL. It adeptly manages the input of values into the email and password fields, pausing for another second. The automation seamlessly clicks the login button and yet again pauses for a second.

Moving forward, the script orchestrates the action of clicking on a calendar icon, initiating a meticulously executed process to select a date range with precision. After the calendar icon is selected, the WebDriver adeptly interacts with the calendar interface, employing a series of strategic clicks to designate the desired date range. In continuation of the script's purposeful journey, the calendar icon is selected once more, successfully concluding the task of date range selection.

The process transitions to the inputting of the region into the search box, thereby triggering the search function. This sequence effectively simulates the act of entering a location, selecting a dropdown option, and initiating the search process. To ensure a complete loading of the hotel page, the script conscientiously waits for a duration of five seconds.

Continuing with its automated interactions, the WebDriver strategically clicks on various buttons on the hotel page, enabling navigation and interaction with the page's elements. As the script concludes its series of orchestrated tasks, the "finally" block safeguards against unexpected disruptions, assuring the proper closure of the browser session.

One more thing to note is that Selenium also allows for Regression testing of the whole application, as when each new feature is added, more code can be added to the end or middle of the selenium test to ensure that everything is running smoothly. This has helped to discover several major bugs, which have been fixed since then.

In sum, the Selenium WebDriver library proves indispensable in automating interactions within our hotel booking application. The script efficiently navigates, inputs, and engages with webpage elements, meticulously emulating user behaviour. By adhering to established grammar and professional style, the script's operational sequence is succinctly conveyed.

4.3 Pull Up and Tear Down Testing

Using mockData, we tested our sorting function sortBySearchRank used to sort Hotels based on their searchrank values by taking a few values from the Ascenda hotel endpoint and testing our function with their values, returning results and confirming that the hotels are successfully sorted based on their searchRank.

For integration testing for backend files accounts.test.js and bookings.test.js, we also ran create booking/account function in our initial pull up stage and delete booking/account function in our tear down stage. For our actual test we used our GET functions to check if the booking/account has been successfully created and we test our update booking/account function as well. This allows us to make sure that the various functions work together in the backend when needed.

---

5. Implementation Challenges

5.1 Challenges Faced

Some challenges faced were the coordination and integration between the frontend and backend frameworks. There are several queries, APIs and backend calls that are made for every functionality being implemented. For example, in implementing the display of the dropdown bar for countries in the home page and updating the displayed top 3 destinations, we had to query the API for searching of destinations and update the featured destinations accordingly. Incorrect querying as well as not triggering the useEffect() function and setting it to the selected option accurately in the client-side code would result in failure to update and render the accurate featured destination. The code for the querying the destinations from selected country and triggering the useEffect() function and updating can be found in hotelbookingsystem\bookingsys\client\src\components\featured\Featured.js and the implementation of the dropdown bar for countries in frontend framework can be found in hotelbookingsystem\bookingsys\client\src\components\navbar\Navbar.js. It was difficult passing data from 2 components that are on the same page. It was difficult querying the image data.

5.2 Lessons Learnt

We used agile development methodology and split the subsystems based on backend and frontend implementation work. Although the team initially split work based on frontend and backend portions of the project, we learnt that all team members need to familiarise with both backend and frontend frameworks as the integration requires knowledge of querying, the specific function of each query and how to use it to update and render the data required for all pages and their functionalities. We learnt about how to coordinate the tasks between the team members based on different functionalities that we aimed to provide on our web application.

6. Workload Distribution

Member	Planned	Actual
John-David Tan Ming Sheng	1. Frontend – Hotel Room Page, Hotel List Page, Register Page	1. Frontend design – Hotel Room page, Hotel List Page, Register Page 2. Integrate front end and back end – link up register page with accounts.js backend, hotel list & room page with hotels.js 3. Set up search context to pass data from page to page 4. Managed data and displaying data for hotel list & rooms page
Tan Xynn Ee, Cordelia	1. Frontend – profile, login, register, bookings pages	1. Frontend design– profile, login, register, bookings page 2. Integration of front end and back end - bookings page with bookings database, integration of edit profile page with accounts.js backend. View booking details and delete booking from both Accounts.js and Bookings.js 3. Css for profile, login, register and

		bookings page, navbar 4.
Venkatakrishnan Logganesh	1. Backend – CRUD operations with accounts and routing 2. Unit testing	1. Backend – Setting up hash password, accounts Schema and CRUD operations later refined by group members. 2. Frontend – Added GuestInfo form to the payment page 3. Integrated the submit button in the payment page with bookings.js 4. Report- Created use case tables for hotel booking, use case and misuse case diagrams, filter by price and overall hotel booking process sequence diagrams. Wrote the non-functional requirements, workings of the code for selenium testing based on what my group members did, test case tables for System Testing in terms of hotel booking and making payment. Added UI pictures
Wong Jia Kang	1. Backend – query Ascenda API	1. Backend – Worked with Zach

	2. Unit testing 3. Integration testing	to set up querying of Ascenda API for our search bar and getting hotel prices. 2. PUT function for bookings.js to update booking 3. Unit testing – search.test.js, accounts.test.js, bookings.test.js, hotels.test.js (negative testing included) 4. Pull up and tear down testing for sortBySearchRank () function in views/HotelsList.j s 5. Integration testing (Pull up and tear down) for bookings.test.js
Rajavelu Sree Devi	1. Backend – setting up of database with CRUD and routing 2. Unit testing	1. Backend – setting up of database with Node.js bookings and created schema using mongoose, set up authentication for the MongoDB database to our server 2. Frontend – countries dropdown in navbar 3. Helped with creating negative unit test cases 4. Report- use case tables, sequence

		diagrams implementation challenges, clarification of open issues, review
Shjonathan Tan Tze Ern	1. Frontend- home/landing page, payment page, infinite scrolling, 2. Integrate frontend and backend	1. Frontend – home/landing page, search bar engine, date picker, options, payment page using stripe js, infinite scrolling, top 3 cities and top 3 countries features on the home page 2. Integrate frontend and backend - linked search bar and hotelsList page with ascenda api, date picker, linked accounts.js routes with login page 3. Set up authcontext to pass user data around all pages when user is logged in.
Zachary Seow Shijie	1. Integrate frontend and backend 2. Integration testing 3. Selenium testing	1. Integrate frontend and backend and ensure that it was working properly. 2. Clean up and redid server backend to ensure ease of

		access when implementing the API for the app. 3. Work together with Jia Kang to implement autosearch in search.js as well as the other operations in it. 4. Set up the Selenium integration testing and system testing tables for the application. 5. Helped to setup pullup/pulldown for various test cases like in accounts testing.