Project Title: Book a Cruise

Website Name: Cruising

Semester: Spring 2019

Group Number: 1

Team Members: Randall Byrd, Riyazh Dholakia, Syed Kalam, Sujal Patel, Biprojoyti Paul

Date of Submission: April 14th, 2019

Planning and Scheduling:

Assignee Name	Email	Task	Duration (hours)	Dependency	Due date	Evaluation
Riyazh Dholakia	rdholakia1@student.gsu.edu	Use Cases, Requirements, Test Cases, Architectural modeling, Readme, Front-End, Database, Sequence and Use Case Diagram, Testing Environment, Unit Testing, Bugs, Test Documentation	4 hours	Microsoft Visio Download, Visual Studio Code Download, Use Case(s), Requirements	4/14/19	100%
Sujal Patel	spatel287@student.gsu.edu	Requirements, Test Cases, Problem Statement, GitHub Project Board, Readme, Front-End, Database, Sequence and Use Case Diagram, Unit Testing, Test Documentation, Bugs	4 hours	Microsoft Visio Download, Visual Studio Code Download, Use Case(s), Requirements	4/14/19	100%
Randall Byrd	rbyrd11@student.gsu.edu	Requirements, Use Case Diagram, Test Cases, Architectural modeling, Front-End, Database, Sequence and Use Case Diagram, Unit Testing, Test Documentation, Screen shots	4 hours	Microsoft Visio Download, Visual Studio Code Download, Use Case(s), Requirements	4/14/19	100%

Syed Kalam	skalam1@student.gsu.edu	Use Cases, Class Diagram, Requirements, Test Cases, Behavioral modeling, Database, Testing Environment, Unit Testing, Test Documentation, Bugs Report	4 hours	Everyone's work to format project, Microsoft Visio Download, Visual Studio Code Download, Use Case(s), Requirements	4/14/19	100%
Biprojoyti Paul **(Coordinator)**	bpaul5@student.gsu.edu	Planning and Scheduling, Requirements, Test Cases, Behavioral modeling, Front-End, Database, Unit Testing, Bugs, Test Documentation	4 hours	Microsoft Visio Download, Visual Studio Code Download, Use Case(s), Requirements	4/14/19	100%.

Problem Statement:

People who want to go on vacation, have fun, new experience, explore the world, love being on a boat, traveling, food, etc. There are many reasons to go on a cruise. It is also better price wise, compared to flying. Some people feel safer on water rather than flying to places. The audience is endless, since anyone can access site. Our competitors are doing this now. The audience can also be controlled by the market and the client. Sometimes, it is hard to find a cruise booking place. Nowadays, many people want to go to their desired place and wanted to be close to the water and be with the family. The valuable thing is that the cruise gives them all things to a reasonable price. Cruise trips the best trips ever, that incorporate boats, attractions, clubs, family games, and more. People can book plane, car, or go by train, bus, but the fact that Cruise is different than others, tells you more people are will experience this. Anybody would love to be part of an amazing scenery and experience. Other alternatives can also be to book from competitors' site. The top-level objective is building a product to officiate, booking a cruise. Differentiators from other products is out will be simple, better UX, and other various packages other competitors do not offer. There is a lot of websites and apps already been made on this project. If somebody Googles, they will find many different places to book cruises. We have seen a lot of websites

where the look of the website does not appeal as much so that the customer will not be impressed to book the cruise from their website. We made a front-end and back-end/database which will connect to each other. We think we have enough resources to make that possible. If we need help, we can use Google, YouTube, and Udemy courses to sharpen our skills. The front-end system we can built by HTML, CSS, and JavaScript. To make it responsive we will use some jQuery. For database, we will use Firebase. On backend we will use node, JavaScript, hopefully Firebase will take care of server.

It will be fun and challenging to make such kind of project. The fact that we can build a system, which can book ticket with given customer information will be mined blowing, and we can use it later any similar system if we want. This technical idea is common process, hopefully we can introduce new tech into the scene. The fact that we can build a system, which can book ticket with given customer information will be mined blowing, and we can use it later any similar system if we want. This technical idea is common process, hopefully we can introduce new tech into the scene. This is also help us build our skills and gives a good resume project. These skills help us in our future career.

User Requirements:

Use Case Diagram:

Users can choose departures and destinations. Admins can change website design/UI and shut down the system. Both employees and admins are connected to the database

Use Cases:

Use Case #: 1
Name of the Use Case: User Sign Up
Actors: User, Admin, and Database
Description:

- The User will click on sign up option on login page.
- After clicking sign up, the user will have to fill a form out, with inputs in requirements.
- The form boxes are including so they are all required to be filled.
- After the inputs are complete, the User will have to hit submit.
- If successful, the User will get success message.
- If unsuccessful, the User will a try again message to try again.
- The information will save in Database if successful.
- After successful they will go to login in page to login in with their email and password.
- Admin can change the look and feel of website and anything in Database.

Exception Path: If the User already has an account with us, they will be prompted to Login.
Alternate Path: There is only one path in this case, unless error to try again.
Pre-condition: Browser, Mouse, Keyboard, Internet Connection.
Post-condition: Login in page after sign up is complete.

Use Case #: 2
Name of the Use Case: User Login
Actors: User, Admin, and Database
Description:

- The User will be given the login page, once accessing out site.
- The User will enter email and password. These are including.
- Recommend the password option is an exclude option.
- After the inputs are complete, the User will have to hit submit.
- If successful, the User will end up on the destination/departure page.
- If unsuccessful, the User will a try again message to try again.
- The information will be validated with database.
- Admin can change the look and feel of website and anything in Database like there email and password.

Exception Path: If the user does not have an account with us they will be prompted to Sign Up.
Alternate Path: There is only one path in this case (Departure/Destination site), unless error to try again.
Pre-condition: Valid account must be already made.
Post-condition: Entering our Destination/Departure page through browser after pre-condition has been met.

Use Case #: 3
Name of the Use Case: User selects
Destination(s) and Departure
Actors: User, Employee, Admin, and Database
Description:

- A user should be able to select a Destination(s) and Departure selection for their cruise.
- A Destination should include a Departure.
- A Destination can exclude multiple Destinations.
- Employee and Admin should be able to change the Destination and Departure locations through the database.
- Admin can turn off page and change the look and feel of it.

Exception Path: If the User already had a Destination and Departure selected and we saved it in a cookie/session.

Alternate Path: There is two alternate paths in this case. The one path is the "normal" path of the User to select Destinations and Departure. The other path is Employee and Admin, being able to change Destination and Departure in Database. Admin being able to turn off system and page.
Pre-condition: Browser, mouse, keyboard, internet connection, and valid login account.
Post-condition: Next page which is to select and room.

Use Case #: 4

Name of the Use Case: User selects Room

Actors: User, Database, and Employee

Description:

- The User will have to click the room options tab.
- Upon selection, the User will have to specify the number of adults and or children that will be in attendance
- After, the User will be prompted to select from a list of three room types: Silver, Gold, or Platinum Suite.
- After giving the desired room type, the Database will look to see if that room isavailable, and if not it will list of other room options that are.
- If there is no vacancy for the room, then an error message may be thrown, and a list of other available room options may be prompted on the screen, else finding the desired room will generate a success.
- Each room will have a price to determine of the cruise.
- Employee will change room types and room price in the Database system.

Exception Path: If the User already had a Room selected and we saved it in a cookie/session.
Alternate Path: The alternative path will be the room type options (silver, gold, platinum suites)
Pre-condition: Desired room type, being logged in, and after selecting Destination/Departure.
Post-condition: Room selection is complete and then you go to the next page which is to choose Dates.

Use Case #: 5

Name of the Use Case: User selects Dates

Actors: User, Admin, and Database

Description:

- The user will select the dates option
- After clicking, the user will select the start and end dates of his or her trip
- The database will then look up respective cruises during that time

- if for any reason a cruise is not available then the database will generate an error message to user. if there is availability then a list of cruises will be shown

Exception Path: If the User already had Dates selected and we saved it in a cookie/session.
Alternate Path: There is only one path, unless the dates yield error then the User will be prompted to enter new dates
Pre-condition: knowing the dates, Desired room type, being logged in, and after selecting Destination/Departure
Post-condition: the dates are chosen, and the User will be sent to the passenger information screen.

Use Case #: 6
Name of the Use Case: User enters Passenger(s) Information
Actors: User and Database
Description:

- The User will select the Passenger(s) Information option
- The Database will allocate information box(s) for each passenger in attendance, the number of passengers is retrieved from the select a room's page, given by the User.
- After, the User will be prompted to give the first name, last name and date of birth of each passenger starting from passenger 1
- If any passenger box is left unfilled then an error message to complete the form will be shown.
- If everything is complete the user will click next and will be brought to the payments page.

Exception Path: If the User already had a Passenger info entered before, we will have been saved in our Database.
Alternate Path: There is only one path on this page and if there is error the User will have to try again
Pre-condition: Having passengers' information, knowing the dates, Desired room type, being logged in, and after selecting Destination/Departure
Post-condition: the information is given, and the User is sent to the payments page.

Use Case #: 7
Name of the Use Case: User Pays for the Cruise
Actors: User and Database
Description:

- The user will enter the payment page.
- User must put all his information like first name, last name, address, country, city state, region, zip code, telephone number, email address.
- Then user must put the credit/debit card number.
- Our system will be able to take the American Express card, Discover, Master Card, Visa.
- User should be able to provide the security code, expiration date, Billing Address.
- The information will be sent to our Database system for the payment to be processed and saved for next time.

Exception Path: If they have a saved payment method already on our Database system.

Alternate Path: The User can call instead to book the cruise.
Pre-condition: User should confirm all other pages before coming here, User should know the dates, User must have money in the card.
Post-condition: If all the required fields of payment page is done then User will go to confirmation page.

Use Case #: 8
Name of the Use Case: User gets an Order Confirmation
Actors: User and Database
Description:

- After the payment page, the User would see a confirmation page.
- The confirmation page will have a confirmation number.
- User will have a confirmation email sent out to the given email address.

Exception Path: If the confirmation page is closed they can access it in their email.
Alternate Path: There is no alternate path for this case. The user should be able to see the confirmation page after the payment.
Pre-condition: The User must pay through his credit/debit card.
Post-condition: User should must get a confirmation number and a email confirmation.

Use Case #: 9
Name of the Use Case: User can Contact Cruising
Actors: User and Database and Employee
Description:

- Contact page in the navigation bar, which the User can access Contact page from here to reach out to Cruising
- After User is on the Contact page, they type in their comments and hit submit.
- The Database captures that information to give us an email, so Employee can reply to the User.

Exception Path: The other path is the User does not go through with the Comments.
Alternate Path: The User can leave to another page or cancel.
Pre-condition: The User must have a comment to ask
Post-condition: Going back to Home Page.

Use Case #: 10
Name of the Use Case: User can Logout
Actors: User
Description:

- After the order confirmation page, the user will be prompted to logout.
- If the user will have the option to go back to Departure/Destination page to book another cruise.

Exception Path: The other path can be if the user automatically gets logged off when the session/cookie to done.
Alternate Path: For user either they can log out or keep going through our website.
Pre-condition: Browser, Mouse, Keyboard, Internet Connection.

Post-condition: Logging out page after reservation is complete.

System Requirements (including test cases):

Requirement #: 1
Use Case #: 1
Name: Sign Up
Introduction: Easy look and feel for the form for the User to sign up with so they can login to our website, to book a cruise.
Rationale: Sign up process for user to login into the site, and to start booking a cruise.
Input: First Name, Last Name, Email, Password, Confirm Password, Phone Number, Full Address, DOB
Requirement Description: The User should be able to sign up. This is only possible with all requirements are met.

1. All fields must be completed.
2. Requirements from the system are met too.

Output: A successful screen, then routed to login to log back in.
Test Cases:

Test Case ID: 1.1
Description: Attempt Sign Up with any valid email with valid password.
Test Inputs: Email and password not currently in Database table linked to a User account.
Expected Results: Successful, then goes to Login Page to re-enter the nearly created Sign Up page.
Dependencies: None
Initialization: New user account (email and password)
Test Steps:

1. Submit a valid email and password that is not in our Database System
2. Click submit and wait for successful message to be redirected to Login Page.

Test Case ID: 1.2
Description: Attempt Sign Up with a current User's email and password
Test Inputs: Email and password that is not valid like a fake email or no email and password
Expected Results: Unsuccessful, an error message to try again.
Dependencies: None
Initialization: Invalid email and password
Test Steps:

1. Submit an invalid email and password.
2. See an alert for an error message to try again, and why it is invalid email or password.

Test Case ID: 1.3
Description: Attempt Sign Up with any invalid email with invalid password
Test Inputs: Email and password currently in Database table linked to a User account.
Expected Results: Unsuccessful, an error message to try again.
Dependencies: None
Initialization: Old User account with an email and password
Test Steps:

1. Submit a current email and password in our Database System.
2. See an alert for an error message to try again, and why it is invalid email or password.

Requirement #: 2
Use Case #: 2
Name: Login
Introduction: Easy look and feel for the 2-input form for the User to login to go to our Destination/Departure page.
Rationale: Login process for User to login into the site, and to start booking a cruise.
Input: Email and password
Requirement Description: The User should be able to sign up. This is only possible with all requirements are met.

1. All fields must be completed.
2. Requirements from the system are met too.
3. Database validation must be met also.

Output: Routed to our Destination/Departure page.
Test Cases:

Test Case ID: 2.1
Description: Attempt player login with valid user information
Test Inputs: Email and password corresponding to a valid User account.
Expected Results: Login Successful. User presented with player dashboard
Dependencies: None
Initialization: Users account must already have been created.
Test Steps:

1. Submit valid username and password on login page

Test Case ID: 2.2
Description: Attempt login with invalid user information.
Test Inputs: Email and password inconsistent with valid User accounts.
Expected Results: Login unsuccessful. User notified that login unsuccessful and told to retry again.
Dependencies: None
Initialization: Database table with correct user information to check against.
Test Steps:

1. Submit invalid username and password on login page.
2. Verify correct exception path is taken.

Requirement #: 3
Use Case #: 3
Name: Destination(s) and Departure
Introduction: Destination and Departure page to select where the User wants to go for their cruise. The page will be simple and easy to choose from.
Rationale: Destination/Departure page, this is the first step in the cruise booking process after login validation.
Requirement Description:

1. Destination/Departure page with easy check boxes.
2. There should at least 1 Destination selected
3. There should be only 1 Departure selected.
4. All information is pulled from Database when should locations.
5. The User can select places that are not offered. They should be not selectable (grayed out). Checking with Database constantly.

Output: Routed to Room page after all correct selections have been made.
Test Cases:

Test Case ID: 3.1
Description: Attempt to select an invalid departure and or departure
Test Inputs: Enter an invalid destination/departure
Expected Results: Unsuccessful booking and user gets notified to select a correct destination/departure.
Dependencies: None
Initialization: All the possible destinations and departures are given from the database (along with another option where users can enter a location that may not be shown).
Test Steps:

1. Enter an invalid location for destination/departure
2. Database will validate if the selection is valid or not.

Test Case ID: 3.2
Description: Attempt to select two or more departures and or destinations at the same time
Test Inputs: Two respective destinations/departures
Expected Results: Invalid destination or departure. User notified to pick one option.
Dependencies: None
Initialization: Database will generate a list of Destinations/departures
Test Steps:

1. Select two destinations/departures.
2. Make sure database only allows one.

Test Case ID: 3.3
Description: Attempt to skip past the departure page without selecting.
Test Inputs: Skip page
Expected Results: Invalid destination or departure. User notified to pick an option.
Dependencies: None
Initialization: Database will generate a list of Destinations/departures
Test Steps:

1. Try to press the next button without selecting.
2. Verify that the database requires a selection.

Requirement #: 4

Use Case #: 4

Name: Room

Introduction: Room selection is made readily simple for customers to choose desired room options.

Rationale: This is the second step in booking a cruise. A room needs to go with your Destination are this will determine price.
Input: Number of people in attendance and desired room type
Requirement Description:

1. User must input a valid number of people for correct search results for example 100 guest for one room would be far too many.
2. Database finds appropriate rooms requested by customers

Output: A room selection success is affirmed, and the User will be routed to the dates page.
Test Cases:

Test Case ID: 4.1
Description: Attempt to select a room that has already been booked
Test Inputs: Select a room that has just lost vacancy.
Expected Results: System will validate that the room is no longer available.
Dependencies: None
Initialization: Database has checked all room availabilities
Test Steps:

1. Select a room that is booked already
2. Verify that there is an exception.

Test Case ID: 4.2
Description: Attempt to select rooms that are not offered for the specified amount of people
Test Inputs: Enter a large amount of guest and try to book a room allocated for a small amount of people.
Expected Results: Unsuccessful booking and the system will generate an error exception.
Dependencies: None
Initialization: Database will compare the number of guests to the room size.
Test Steps:

1. Select 10 guest and try to book a room for two.
2. Verify that an exception is thrown

Test Case ID: 4.3
Description: Attempt to skip a room selection.
Test Inputs: Skip to next page
Expected Results: The database will require a selection before moving on the next page.
Dependencies: None
Initialization: The database will generate a list of room selections
Test Steps:

1. Skip to the next page
2. Verify that an exception is thrown

Requirement #: 5
Use Case #: 5
Name: Dates
Introduction: choosing respective dates will be easy and user friendly

Rationale: This the third step in booking a cruise. You need a Dates to know when you are going on the cruise.
Input: Start date and end date
Requirement Description:

1. Both start and end dates must be given to generate a search result
2. the database outputs an error or a list of cruises.

Output: A successful screen, then routed to the passenger information page.
Test Cases:

Test Case ID: 5.1
Description: Attempt to select the correct date with the prior cruise departure/destination.
Test Inputs: Cruise departure/destination is assigned with the selected date in database system.
Expected Results: System will allow the user to book the departure/destination date.
Dependencies: None
Initialization: Database will match the information with the selected date by the user.
Test Steps:

1. Enter the correct date and booked the cruise departure / destination
2. User can travel now on the selected date.

Test Case ID: 5.2
Description: User select the wrong date to book the cruise, the date is not specified in the database.
Test Inputs: Select a date which is not allocated in the database for the prior cruise.
Expected Results: The system will prompt a message that prior cruise is not available for the required date.
Dependencies: None
Initialization: Database will check the date requirement for the cruise.
Test Steps:

1. Select the correct date for the cruise departure / destination.
2. Database will validate if the selected date is matched with our database system.

Test Case ID: 5.3
Description: Attempt to select a date for the departure, which is already booked or full.
Test Inputs: t
Expected Results: System will generate a message with no space accepted.
Dependencies: None
Initialization: Database will generate the number of passenger, those are assigned for the required dates.
Test Steps:

1. Enter the correct date but it is already being full. Select the other dates for the departure which has capacity for the passengers.

Requirement #: 6
Use Case #: 6
Name: Passenger Information
Introduction: Entering passenger(s) information will be made easy and efficient

Rationale: This the third step in booking a cruise. The Database will have the exact amount of necessary spaces for each passenger in attendance, so the User does not have to do it.
Input: first name, last name, and date of birth
Requirement Description:

1. All fields must be completed
2. Database validation must be met also.

Output: No errors will persist, and the User will then move on to the payments page
Test Cases:

Test Case ID: 6.1
Description: User enters their information according to the required fields.
Test Inputs: Database will match the requirements with the given information.
Expected Results: It will allow the user to move forward.
Dependencies: None.
Initialization: Database will match the given information with the stored information
Test Steps:

1. By entering all the required fields, system will allow the user to move forward.

Test Case ID: 6.2
Description: User leave an empty field while entering their information.
Test Inputs: Database system required to fill each box for the information.
Expected Results: An error message will appear and display to try it again.
Dependencies: None
Initialization: All fields must be completed while putting user information because it is required for our database system
Test Steps:

1. Leave one box empty.
2. Database will not provide positive result until all fields are filled out correctly.

Test Case ID: 6.3
Description: The user is already signed up before and trying to insert his same information again, a message will appear saying that already an existing member.
Test Inputs: Database check the given information provided by the user.
Expected Results: Existing member tab will appear.
Dependencies: None
Initialization: Database system will match the user information with the stored information.
Test Steps:

1. Submit the existing information.
2. Database will allow one information, only one time.

Requirement #: 7
Use Case #: 7
Name: Payments
Introduction: Making a payment will be divergent for all Users
Rationale: This the fifth step in booking a cruise. You have pay to secure your spot on the cruise. The system will support multiple banking platforms (Visa, MasterCard, etc.).

Input: the first and last name on the card, the credit or debit card number, the security code, expiration date, and billing address
Requirement Description:

1. All fields must be entered correctly.
2. Database must ensure security.

Output: The credit/debit card will be accepted, sending the User to the order confirmation page.
Test Cases:

Test Case ID: 7.1
Description: User forget to fill out the box of security code for his/her credit card.
Test Inputs: Each box need fill out for the credit/debit card information.
Expected Results: An error message will be saying the payment cannot be processed.
Dependencies: None.
Initialization: For payment, all field should be completed for the credit/debit card because it is needed to get the money from the card.
Test Steps:

1. Give a random 3-digit number for security code.
2. Payment will not be processed because all the information from the card is needed to process the payment.

Test Case ID: 7.2 This will match with use case # 7
Description: User put all the information needed from the credit/debit card.
Test Inputs: payment information will be checked by the corresponding bank. Bank will check if the user has enough money in the bank account.
Expected Results: A message will say the payment is processed.
Dependencies: Bank Account.
Initialization: Bank database will check if the user information matched with given account number
Test Steps:

1. Fill out all the box needed for the card information.
2. Bank database will match and process the information.

Requirement #: 8
Use Case #: 8
Name: Order Confirmation
Introduction: Order history will be safe and secure in the Database
Rationale: The Database will generate a unique confirmation code to respective Users
Input: None
Requirement Description:

1. The Database system must save the order in the system.
2. The Database system must populate an order confirmation number and be then shown to the User on the screen.
3. The Database system will also have to email the confirmation number to the User.

Output: The User will receive an order confirmation number and will be prompted to Logout or got to Departure and Destination page.
Test Cases:

Test Case ID: 8.1
Description: The payment did not process, and user will not get a confirmation page.
Test Inputs: Leave one box empty on payment information page.
Expected Results: Error Message.
Dependencies: Payment Page. (This test is directly correlated with the payment page.
Initialization: our firebase database will check the payment information page.
Test Steps: None.

Test Case ID: 8.2
Description: The payment processed, and user will get a confirmation page.
Test Inputs: Fill out all the boxes for the credit card information page.
Dependencies: Payment Page. (This test is directly correlated with the payment page.
Initialization: our firebase database will check the payment information page. Our database will match the email address and send a confirmation email to that email address. The bank database will check the card information to match the card information.
Test Steps:

1. None.
2. After payment user will get a confirmation number and a confirmation email.

Requirement #: 9
Use Case #: 9
Name: Contact
Introduction: Contact page to help User contact us if any issues.
Rationale: To help User's facilitate an easy contact.
Input: Comment section then click submit.
Requirement Description: User should Logout after done with our site.

1. The Comment section in this page to type any message they click.
2. They hit submit to send Comments to our email, so we can reply too.

Output: An alert that says Thank You and then takes you back to Home Page(index.html) after you click Ok.
Test Cases:

Test Case ID: 9.1
Description: Contact page successful message to send to Employee.
Test Inputs: Type in Comments on Contact Page
Expected Results: User will get successful message.
Dependencies: User must be logged in.
Initialization: Logged in on contact page with comments boxed filled then submit button clicked.
Test Steps:

1. User must fill out comments.
2. User must be currently logged in.
3. User must hit submit to get successful message.

Test Case ID: 9.2
Description: Contact page unsuccessful message to send to Employee.
Test Inputs: Type in Comments on Contact Page

Expected Results: User will get unsuccessful message.
Dependencies: User must be logged in.
Initialization: Logged in on contact page with comments boxed filled then submit button clicked.
Test Steps:

1. User must fill out comments.
2. User must be currently logged in. User must hit submit to get unsuccessful message.

Requirement #: 10

Use Case #: 10

Name: Logout

Introduction: Logging out will be click of button easy for the user after they reserved the cruise.
Rationale:
Input: Click of button
Requirement Description: User should Logout after done with our site.

1. Give the User and option to logout.
2. If cookie/session is over having the User logout automatically.

Output: Logged out of the page, back to the login page.
Test Cases:

Test Case ID: 10.1
Description: User clicks on Logout which will be successful.
Test Inputs: Click on Logout button
Expected Results: User will be closed out the website.
Dependencies: The User must be signed.
Initialization: User can exit out of the website.
Test Steps:

1. It will be just click of button, on the navigation bar.
2. They will end on the Login Page.

Test Case ID:10.2
Description: User clicks on Logout and nothing happens or error.
Test Inputs: Click on Logout.
Expected Results: User will be still be on the current page due to error or invalid logout.
Dependencies: User must be signed out.
Initialization: none
Test Steps:

1. It will be just click of button, on the navigation bar.
2. Error message to try again.

System Modeling:

Class Diagram:

New Class Diagram with Singleton Model (Creational):

Database specification and analysis:

Since we are using Firebase, there are no tables (no SQL). All you need is JSON structure given by Firebase. This is can be found in our GitHub code. The link for the GitHub is at the end of our project in Appendix Section. Most of our "tables" have been implemented. Use Case #1-6 and 10 have all "tables" implemented in JSON struct since no SQL. Below is what our Database will look like if it is table structured. Access to our Firebase/Database is restricted due to Gmail/Google Login. If you like, please see screenshots in Appendix section.

<u>Parent tables:</u>

Customers: customer_id (PK), customer_first_name, customer_last_name, customer_phone_number, customer_email_address, customer_password, customer_address, customer_city, customer_state, customer_zipcode, customer_dob

Cruises: cruise_id (PK), cruise_destination, cruise_departure

Dates: date_id (PK), cruise_date_start, cruise_date_end

Passengers: passenger_id (PK), passenger_first_name, passenger_last_name, passenger_phone_number, passenger_email_address, passenger_dob

Rooms: room_id (PK), room_type, room_price

Payments: payment_id (PK), credit_card_num, credit_card_exp

Order_confirmations: order_confirmation_id (PK), order_confirmation_number

Child tables:

Customers_cruises: customer_id (FK) and cruise_id (FK)

Cruises_dates: cruise_id (FK) and date_id (FK)

Cruises_roooms: cruise_id (FK) and room_id (FK)

Payments_order_confirmation: payment_id (FK) and order_confirmation_id (FK)

Customers_passengers: customer_id (FK) and passenger_id (FK)

Also see Implementation section.

Architecture modeling:

The price generator figures out the final quote that the user must pay. The tax function calculates the total tax that will be applied to the final quote. The data analyzer reviews the information given by the user regarding payment (i.e. First name, last name, card info.). The exception reader sees if any text boxes have been left blank by the user. The discount/ look-up part of the system validates user discount codes for respective trips.

4+1 views:

Logical view:

- Concern: Functionally
- Target audience: Developers and users

LOGICAL VIEW:

The logical view is defined as it is connect the functionality of the system to the end-users. State diagrams, class diagram and UML diagrams also represent the logical view.

Process view:

- Concern: Functionally, Performance
- Target audience: Developers and users

PROCESS VIEW:

The process view explains that how the system processes and how they communicate, and how they deal with the system behavior. UML diagram and activity diagram represent the process view.

Development view:

- Concern: Portability
- Target audience: Developers

DEVELOPMENT VIEW:

Basically, development view relates to the programmer's perspective and it is concerned with the software management. All implementation part of the project comes in development view category, UML diagram and package diagrams represent the development view.

Physical view:

- Concern: Quality
- Target audience: Operations

PHYSICAL VIEW:

The physical view is representing the physical data stored and processed in a database. It is also concerned with the properties of software components and make physical connection between them. UML diagram and deployment diagram represent physical view.

Behavioral modeling:

Use Case Diagrams:

Payment:

Rooms:

Destination/Departure:

Dates:

Sequence Diagrams:

Rooms:

Departure/Destination:

Payments:

Dates:

Implementation:

Please see some screenshots in Appendix or GitHub link in Appendix, for full implementation. To run our code please read our readme file which is also located on GitHub. Implementation is still in progress and may not be fully done.

Our IDE is Visual Studio Code, but you can write code, implement, compile, and execute our code in any IDE you like. Back-end framework is using JSON and JavaScript. The front-end is using JavaScript, HTML, and CSS. Our front-end is more than 50% complete. Please see GitHub link in the appendix to run the code. Our back-end and front-end logic is also connected, and that code can also be seen in GitHub.

Also see Database specification and analysis section.

The readme (see screenshot in Appendix or GitHub) has good description, but here are more details. Then open the signup page. You can click on our sign-up page .html file and it will run. Compile and run is the same for the rest of our file till the confirmation page. Also, we have a nav bar on every single page. You can go back and forth to any pages from our nav bar. The .html will run on any IDE regardless of what software you are using. We are using firebase database. You can see the firebase code on all our pages to connect to the database, and for read and write data back and forth. We are using our own domain hosting through GitHub. You can see the GitHub is hosting our Website pages.

Testing:

Implementing Test Cases/Unit Tests:

Test ID	1.1
Purpose of Test	Attempt Sign Up with any valid email with valid password.
Test Environment	I am using JavaScript Jasmine environment to test our signup page. In this page, we are going to inject the file in jasmine. Jasmine has some build in code that will test the functions from our file

Test Steps	describe("A suite is just a function", signup() {
	//calls signup function from signup.html and test the function.
	var a;
	
	it("and so is a spec", function() {
	a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Email and password not currently in Database table linked to a User account.
Expected Result	Successful, then goes to Login Page to re-enter the nearly created Sign Up page.
Likely Problems/Bugs Revealed	The signup button was not working. We need to change some function of the code to make it work.

Test ID	1.2
Purpose of Test	Attempt Sign Up with a current User's email and password
Test Environment	I am using JavaScript Jasmine environment to test our signup page. In this page, we are going to inject the file in jasmine. Jasmine has some build in code that will test the functions from our file

Test Steps	describe("A signup is just a function", signup() {
	//calls signup function from signup.html and test the function.
	var a;
	
	it("and so is a spec", function() {
	a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Email and password that is not valid like a fake email or no email and password.

Expected Result	Unsuccessful, an error message to try again.
Likely Problems/Bugs Revealed	The signup button was not working. We need to change some function of the code to make it work.

Test ID	1.3
Purpose of Test	Attempt Sign Up with any invalid email with invalid password
Test Environment	I am using JavaScript Jasmine environment to test our signup page. In this page, we are going to inject the file in jasmine. Jasmine has some build in code that will test the functions from our file

Test Steps	describe("A signup is just a function", signup() {
	//calls signup function from signup.html and test the function.
	var a;
	
	it("and so is a spec", function() {
	a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Email and password currently in Database table linked to a User account.
Expected Result	Unsuccessful, an error message to try again.

Likely Problems/Bugs Revealed	The signup button was not working. We need to change some function of the code to make it work.

Test ID	2.1
Purpose of Test	Attempt player login with valid user information.
Test Environment	I am using JavaScript Jasmine environment to test our login page. In this page, we are going to inject the file in jasmine. Jasmine has some build in code that will test the functions from our file

Test Steps	describe("A login is just a function", signup() {
	//calls login function from login.html and test the function.
	var a;
	
	it("and so is a spec", function() {
	a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Email and password corresponding to a valid User account. The user account is already being made for login purpose.
Expected Result	Login Successful. User presented with homepage.
Likely Problems/Bugs Revealed	The login button was not working. We need to change some function of the code to make it work.

Test ID	2.2
Purpose of Test	Attempt login with invalid user information.
Test Environment	I am using JavaScript Jasmine environment to test our login page. In this page, we are going to inject the file in jasmine. Jasmine has some build in code that will test the functions from our file

Test Steps	describe("A suite is just a function", login() {
	//calls login function from login.html and test the function to see if the login function is working. var a;
	
	it("and so is a spec", function() {
	a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Email and password corresponding to a valid User account. The user account is already being made for login purpose.
Expected Result	Login Unsuccessful. User presented with homepage.
Likely Problems/Bugs Revealed	Over here we started with invalid information so the bug was already there.

Test ID	3.1
Purpose of Test	Attempt to select an invalid departure and or departure

Test Environment	I am using JavaScript Jasmine environment to test our destination and departure page. In this page, we are going to inject the destination html file in jasmine. Jasmine has some build in code that will test the functions from our departure file.
Test Steps	describe("A suite is just a function", deparature() { var a; it("can use the , function() { expect(this.foo).toEqual(0); this.bar = "Deparature";
Test Input	Two respective destinations/departures
Expected Result	Unsuccessful booking and user gets notified to select a correct destination/departure.
Likely Problems/Bugs Revealed	We are testing for invalid option to see if it is invalid or no.

Test ID	3.2

Purpose of Test	Select two or more departures and or destinations at the same time. We are checking to see if the destination and departure button is working.
Test Environment	I am using JavaScript Jasmine environment to test our destination and departure page. In this page, we are going to inject the destination html file in jasmine. Jasmine has some build in code that will test the functions from our departure file.

Test Steps	describe("A destination is just a function", destination() {
	//calls login function from destination.html and test the all functions. var a;
	
	it("and so is a spec", function()
	{ a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Two respective destinations/departures
Expected Result	User can choose different destination on the page.
Likely Problems/Bugs Revealed	One of the destination was not working. One Function was wrong for one destination.

Test ID	3.3
Purpose of Test	Attempt to select an invalid departure and or departure
Test Environment	I am using JavaScript Jasmine environment to test our destination and departure page. In this page, we are going to inject the destination html file in jasmine. Jasmine has some build in code that will test the functions from our departure file.

Test Steps	describe("A deperature is just a function",
	deparature() { var a;
	
	it("can use the , function() {

	expect(this.foo).toEqual(0); this.bar = "Deparature";
Test Input	Two respective destinations/departures
Expected Result	Unsuccessful booking and user gets notified to select a correct destination/departure.
Likely Problems/Bugs Revealed	We are testing for invalid option to see if it is invalid or no so no bug found.

Test ID	4.1
Purpose of Test	Attempt to select a room that has already been booked
Test Environment	I am using JavaScript Jasmine environment to test our login page. In this page, we are going to inject the file in Jasmine. Jasmine has some build in code that will test the functions from our file.

Test Steps	describe("A suite is just a function", roomSel() {
	//calls room function from room.html and test the function to see if the room selection function is working. var a;
	
	it("and so is a spec", function() { a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Select a room that has just lost vacancy.
Expected Result	System will validate that the room is no longer available
Likely Problems/Bugs Revealed	Over here we started with unavailable room so the bug was going to happen.

Test ID	4.2
Purpose of Test	Attempt to select rooms that are not offered for the specified amount of people
Test Environment	I am using JavaScript Jasmine environment to test our login page. In this page, we are going to inject the file in Jasmine. Jasmine has some build in code that will test the functions from our file.

Test Steps	describe("A suite is just a function", roomSel() {
	//calls room function from room.html and test the function to see if the room selection function is working. var a;
	
	it("and so is a spec", function() { a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Enter a large amount of guest and try to book a room allocated for a small amount of people.
Expected Result	Unsuccessful booking and the system will generate an error exception.
Likely Problems/Bugs Revealed	Over here we started with low amount of rooms so the bug was going to happen.

Test ID	4.3
Purpose of Test	Attempt to skip a room selection.
Test Environment	I am using JavaScript Jasmine environment to test our login page. In this page, we are going to inject the file in Jasmine. Jasmine has some build in code that will test the functions from our file.

Test Steps	describe("A suite is just a function", roomSel() {
	//calls room function from room.html and test the function to see if the room selection function is working. var a;
	
	it("and so is a spec", function() { a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Skip to next page
Expected Result	The database will require a selection before moving on the next page.
Likely Problems/Bugs Revealed	No bugs, since it is impossible to go on to the next page without a room selection.

Test ID	5.1
Purpose of Test	Attempt to select the correct date with the prior cruise departure/destination
Test Environment	I am using JavaScript Jasmine environment to test our login page. In this page, we are going to inject the file in Jasmine. Jasmine has some build in code that will test the functions from our file.

Test Steps	describe("A date is just a function", dateSel() {
	//calls room function from dates.html and test the function to see if the room selection function is working. var a;
	
	it("and so is a spec", function() { a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Cruise departure/destination is assigned with the selected date in database system
Expected Result	System will allow the user to book the departure/destination date

Likely Problems/Bugs Revealed	No bugs, since it is impossible to go on to the next page without a date selection.

Test ID	5.2
Purpose of Test	User select the wrong date to book the cruise, the date is not specified in the database.
Test Environment	I am using JavaScript Jasmine environment to test our login page. In this page, we are going to inject the file in Jasmine. Jasmine has some build in code that will test the functions from our file.

Test Steps	describe("A date is just a function", dateSel() {
	//calls room function from dates.html and test the function to see if the room selection function is working. var a;
	
	it("and so is a spec", function() { a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Select a date which is not allocated in the database for the prior cruise.
Expected Result	The system will prompt a message that prior cruise is not available for the required date.
Likely Problems/Bugs Revealed	No bugs produced, by making dates set to zero to get error.

Test ID	5.3
Purpose of Test	Attempt to select a date for the departure, which is already booked or full.

Test Environment	I am using JavaScript Jasmine environment to test our login page. In this page, we are going to inject the file in Jasmine. Jasmine has some build in code that will test the functions from our file.

Test Steps	describe("A date is just a function", dateSel() {
	//calls room function from dates.html and test the function to see if the room selection function is working. var a;
	
	it("and so is a spec", function() { a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Enter the date, which has no space for more passengers.
Expected Result	The system will prompt a message that prior cruise is not available for the required date.
Likely Problems/Bugs Revealed	No bugs produced, by making dates availability set to zero to get error.

Test ID	6.1
Purpose of Test	User enters their information according to the required fields.
Test Environment	I am using JavaScript Jasmine environment to test our login page. In this page, we are going to inject the file in Jasmine. Jasmine has some build in code that will test the functions from our file.

Test Steps	describe("Database will match the requirements", dataBaseSel() {
	//calls room function from dataBase.html and test the function to see if the database function is working. var a;
	
	it("and so is a spec", function() { a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	By entering all the required fields, system will allow the user to move forward.
Expected Result	It will allow the user to move forward.
Likely Problems/Bugs Revealed	No bug will appear until user entered the information.

Test ID	6.2
Purpose of Test	User leave an empty field while entering their information.
Test Environment	I am using JavaScript Jasmine environment to test our login page. In this page, we are going to inject the file in Jasmine. Jasmine has some build in code that will test the functions from our file.

Test Steps	describe("Database will match the requirements", dataBaseSel() { //calls room function from dataBase.html and test the function to see if the database function is working. var a;
	
	it("and so is a spec", function() { a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Database will not provide positive result until all fields are filled out correctly.
Expected Result	An error message will appear and display to try it again.
Likely Problems/Bugs Revealed	Will get a problem, if all fields were not filled properly.

Test ID	6.3

Purpose of Test	The user is already signed up before and trying to insert his same information again, a message will appear saying that already an existing member.
Test Environment	I am using JavaScript Jasmine environment to test our login page. In this page, we are going to inject the file in Jasmine. Jasmine has some build in code that will test the functions from our file.

Test Steps	describe("Database will match the requirements", dataBaseSel() { //calls room function from dataBase.html and test the function to see if the database function is working. var a;
	
	it("and so is a spec", function() { a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Database check the given information provided by the user.
Expected Result	Existing member tab will appear,
Likely Problems/Bugs Revealed	Will get a bug since it is an existing member.

Test ID	7.1
Purpose of Test	User forgot to fill out the box of security code for his/her credit card.
Test Environment	I am using JavaScript Jasmine environment to test our login page. In this page, we are going to inject the file in Jasmine. Jasmine has some build in code that will test the functions from our file.

Test Steps	describe("User forgot to enter the security code for his/her credit card", paymentSel() {
	//calls room function from payment.html and test the function to see if the database function is working. var a;
	
	it("and so is a spec", function() { a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Each box need fill out for the credit/debit card information.
Expected Result	An error message will be saying the payment cannot be processed.
Likely Problems/Bugs Revealed	Bug will appear since there is no security code entered by the user to process the payment.

Test ID	7.2
Purpose of Test	User put all the information needed from the credit/debit card.
Test Environment	I am using JavaScript Jasmine environment to test our login page. In this page, we are going to inject the file in Jasmine. Jasmine has some build in code that will test the functions from our file.

Test Steps	describe("User entered all the required information", paymentSel() {
	//calls room function from payment.html and test the function to see if the database function is working. var a;
	
	it("and so is a spec", function() { a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Payment information will be checked by the corresponding bank. Bank will check if the user has enough money in the bank account.
Expected Result	A message will say the payment is processed.

Likely Problems/Bugs Revealed	No bug will appear.

Test ID	8.1
Purpose of Test	The payment did not process, and user will not get a confirmation page.
Test Environment	I am using JavaScript Jasmine environment to test our login page. In this page, we are going to inject the file in Jasmine. Jasmine has some build in code that will test the functions from our file.

Test Steps	describe("User leave one box empty", confirmationSel() {
	//calls room function from confirmation.html and test the function to see if the database function is working. var a;
	
	it("and so is a spec", function() { a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Leave one box empty on payment information page.
Expected Result	Error message
Likely Problems/Bugs Revealed	Bug will appear since customer forget to enter their information on one of the payment box.

Test ID	8.2
Purpose of Test	The payment processed, and user will get a confirmation page.
Test Environment	I am using JavaScript Jasmine environment to test our login page. In this page, we are going to inject the file in Jasmine. Jasmine has some build in code that will test the functions from our file.

Test Steps	describe("User will get a confirmation", confirmationSel() {
	//calls room function from confirmation.html and test the function to see if the database function is working. var a;
	
	it("and so is a spec", function() { a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Fill out all the boxes for the credit card information page.
Expected Result	Will get a confirmation.
Likely Problems/Bugs Revealed	No bug will appear

Test ID	9.1
Purpose of Test	Contact page successful message to send to Employee.
Test Environment	I am using JavaScript Jasmine environment to test our login page. In this page, we are going to inject the file in Jasmine. Jasmine has some build in code that will test the functions from our file.

Test Steps	describe("User will get successful message from the employee. ", contactSel() {
	//calls room function from contact.html and test the function to see if the database function is working. var a;
	
	it("and so is a spec", function() { a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Type in Comments on Contact Page
Expected Result	User will get successful message.

Likely Problems/Bugs Revealed	No bug will appear since user is commenting and getting feedback from the employee.

Test ID	9.2
Purpose of Test	Contact page unsuccessful message to send to Employee.
Test Environment	I am using JavaScript Jasmine environment to test our login page. In this page, we are going to inject the file in Jasmine. Jasmine has some build in code that will test the functions from our file.

Test Steps	describe("User will get unsuccessful message from the employee. ", contactSel() {
	//calls room function from contact.html and test the function to see if the database function is working. var a;
	
	it("and so is a spec", function() { a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Type in Comments on Contact Page
Expected Result	User will get unsuccessful message.
Likely Problems/Bugs Revealed	Bug will appear since user was unsuccessful to send the message to employee

Test ID	10.1
Purpose of Test	User clicks on Logout which will be successful
Test Environment	I am using JavaScript Jasmine environment to test our login page. In this page, we are going to inject the file in Jasmine. Jasmine has some build in code that will test the functions from our file.

Test Steps	describe(“User is trying to logout from the website and get successful", logoutSel() {
	//calls room function from logout.html and test the function to see if the database function is working. var a;
	
	it("and so is a spec", function() { a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Click on Logout button
Expected Result	User will be closed out from the website.
Likely Problems/Bugs Revealed	No bug will appear since user is able to logout from the website.

Test ID	10.2
Purpose of Test	User clicks on Logout and nothing happens or error.
Test Environment	I am using JavaScript Jasmine environment to test our login page. In this page, we are going to inject the file in Jasmine. Jasmine has some build in code that will test the functions from our file.

Test Steps	describe(“User is trying to logout from the website and get an error", logoutSel() {
	//calls room function from logout.html and test the function to see if the database function is working. var a;
	
	it("and so is a spec", function() { a = true;
	
	expect(a).toBe(true);
	});
	});

Test Input	Click on Logout button

Expected Result	User will be still be on the current page due to error or invalid logout.
Likely Problems/Bugs Revealed	Bug will appear since user is getting error while logging out from the website.

Test Documentation (bugs found/fixed):

Bugs	The test uncovered bug	Description of bug	Action taken for Bug
Login button was not working when we are testing the login page.	We are testing in the jasmine environment. So, for all the testing like login, signup and destination we first inject the file on a software called karma. Then we write some JavaScript code to test our functions from all those pages.	In the login page login button is somehow not working with the database so when we are testing our test result was showing nothing.	Some problem was within in the function and some was database connection error and some of them was our test code was wrong. So, we manually fix the Database connection, JavaScript function. We also change some of our test code function which one must be true or not to be true.

Sign up button was not working when we're testing	We are testing in the jasmine environment. So, for all the testing like login, signup and destination we first inject the file on a software called karma. Then we write some JavaScript code to test our functions from all those pages.	In the signup page signup button was not working because the function was wrong.	Some problem was within in the function and some was database connection error and some of them was our test code was wrong. So, we manually fix us Database connection, JavaScript function. We also change some of our test code function which one should be true or not to be true.

User could not pick one of the destination from one of the destination page.	We are testing in the jasmine environment. So, for all the testing like login, signup and destination we first inject the file on a software called karma. Then we write some JavaScript code to test our functions from all those pages.	One of the destination was wrong because we still did not put that destination on our database yet.	Some problem was within in the function and some was database connection error and some of them was our test code was wrong. So, we manually fix us Database connection, JavaScript function. We also change some of our test code function which one should be true or not to be true.

User could not pick one of the date from the date page.	We are testing in the jasmine environment. So, for all the testing like login, signup and destination, dates we first inject the file on a software called karma. Then we write some JavaScript code to test our functions from all those pages.	One of the date page code was wrong because our JavaScript file function was not working. Our code was wrong. So, we had to fix this "date" function.	Our code was wrong. So, we had to fix this "date" function. We also change some of our test code function which one should be true or false.

User payment was not processing.	We are testing in the jasmine environment. So, for all the testing like login, signup and destination, dates, payment page. We first inject the file on a software called karma. Then we write some JavaScript code to test our functions from all those pages.	When user put their credit/debit card data. The data was not processing. Our system could not find the data from the card to charge the money from it.	It was our database problem. After writing the code in the payments page the data was being edited to the Firebase data but that data was not being able to read and write. We had to do some research and write separate shit of code. Our test code was same. Function which one must be true or false.

Our database is where we had most of the bugs.	We are testing in the jasmine environment. So, for all the testing like login, signup and destination, dates, payment page, and for our database. We first inject the file on a software called karma. We had to write a separate sheet of code to test the database.	For some pages, it was working. Other pages it was connected but could not read and write the data back and forth.	It was our database problem. After writing the code in the payments page the data was being edited to the Firebase data but that data was not being able to read and write. We had to do a lot of research and write separate sheet of code.

Appendix:

GitHub Project Board:

Test Code:

Screenshot 1.1

Screenshot 1.2

Screenshot 1.3

Screenshot 2.1

```
index-slider.html
index.html
indexold.html
jquery.payform.min.js
login.html
login2.html

19          describe("A login is just a function", signup() {
20              //calls login function from login.html and test the func
21              it("and so is a spec", function() { a = true;
22              expect(a).toBe(true); });
23              });
24
```

Screenshot 2.2

```
index.html
indexold.html
jquery.payform.min.js
login.html
login2.html

25          describe("A suite is just a function", login() {
26              //calls login function from login.html and test the
27              it("and so is a spec", function() { a = true;
28              expect(a).toBe(true); });
29              });
30
```

Screenshot 3.1

```
index-slider.html
index.html
indexold.html
jquery.payform.min.js

31  is just a function", deparature() { var a;
32  e , function() { expect(this.foo).toEqual(0); this.bar = "Deparature";
33
```

Screenshot 3.2

```
index.html
indexold.html
jquery.payform.min.js
login.html
login2.html
main_firebase.js
passenger.html

33  describe("A destination is just a function", destination() {
34  //calls login function from destination.html and test the all functions.
35  it("and so is a spec", function() { a = true;
36  expect(a).toBe(true); });
37  });
```

Screenshot 3.3.

```
index-slider.html
index.html
indexold.html
jquery.payform.min.js

39  describe("A deperature is just a function", deparature() { var a;
40      it("can use the , function() {
41          expect(this.foo).toEqual(0); this.bar = "Deparature";
```

Screenshot 4.1

```
payment.html
payments2.html
README.md
room.html
script.js
signup.html
styles.css

43          describe("A suite is just a function", roomSel() {
44              //calls room function from room.html and test the function t
45
46              it("and so is a spec", function() { a = true;
47
48              expect(a).toBe(true);
49              });
50              });
51
```

Screenshot 4.2

```
payment.html
payments2.html
README.md
room.html
script.js
signup.html
styles.css
Test.js                 9+, U

53          describe("A suite is just a function", roomSel() {
54              //calls room function from room.html and test the functi
55
56              it("and so is a spec", function() { a = true;
57
58              expect(a).toBe(true);
59              });
60              });
61
```

Screenshot 4.3

```
favicon.ico
index-slider.html
index.html
indexold.html
jquery.payform.min.js
login.html
login2.html
main_firebase.js

63  describe("A suite is just a function", roomSel() {
64  //calls room function from room.html and test the function to see if the
65
66  it("and so is a spec", function() { a = true;
67
68  expect(a).toBe(true);
69  });
70  });
71
```

Screenshot 5.1

Screenshot 5.2

Screenshot 5.3

Screenshot 6.1

Screenshot 6.2

Screenshot 6.3

Screenshot 7.1

Screenshot 7.2

Screenshot 8.1

Screenshot 8.2

Screenshot 9.1

Screenshot 9.2

```
payment.html
payments2.html
README.md
room.html
script.js
signup.html
styles.css
Test.js                                   9+, U

179
180         describe("User will get unsuccessful message from the employ
181             //calls room function from contact.html and test the fun
182
183             it("and so is a spec", function() { a = true;
184
185             expect(a).toBe(true);
186             });
187             });
188
```

Screenshot 10.1

```
payment.html
payments2.html
README.md
room.html
script.js
signup.html
styles.css
Test.js                                   9+, U

188
189
190         describe("User is trying to logout from the website and
191             //calls room function from logout.html and test the
192
193             it("and so is a spec", function() { a = true;
194
195             expect(a).toBe(true);
196             });
197             });
198
```

Screenshot 10.2

```
payment.html
payments2.html
README.md
room.html
script.js
signup.html
styles.css
Test.js                                   9+, U

199
200         describe("User is trying to logout from the website
201             //calls room function from logout.html and test
202
203             it("and so is a spec", function() { a = true;
204
205             expect(a).toBe(true);
206             });
207             });
208
```

Firebase (Database):

Readme:

Book a Cruise

GSU CSC Software Engineering Project
Group Name: Friends
Group Number: 1
Project Title: Book a Cruise
Project Webiste Name: Cruising

https://riyazhdholakia.github.io/CSC-SWE-Friends/
Click link to see our website!
Perferred way to run project is to, clone or download as zip and open up index.html file.
Database: Firebase

Contributors:
1. [Riyazh Dholakia](https://github.com/riyazhdholakia)
2. [Sujal Patel](https://github.com/spatel287)
3. [Randall Byrd](https://github.com/Randall97)
4. [Biprojoyti Paul](https://github.com/Bipro723)
5. [Syed Kalam](https://github.com/kalsyed91)

Screenshots of Front-End and Back-End with some code:

Cruising
Home
Book Now
Contact
Log Out
Contact Us
Type a message here, please be detailed...
Submit
Riyazh Dholakia: Full Stack Developer
Biprojoyti Paul: Full Stack Developer
Syed Kalam: Full Stack Developer

Cruising!
Login to Book a Cruise
Email...
Password...
Submit
Sign Up
Use for Login email: riyazhdholakia@gmail.com password: admin123 or Sign Up

Cruising
Home
Book Now
Contact
Log Out
Passenger Info
Passenger 1
First Name
John
Last Name
Smith
DOB
01/05/1978
City
Atlanta
State
GA
Zip Code
30020
Email
hello@gmail.com
Phone Number
404-111-8888
Passenger 2
First Name
John
Last Name
Smith
DOB
01/05/1978
City
Atlanta
State
GA
Zip Code
30020
Zip Code
Email
hello@gmail.com
Phone Number
404-111-8888
Submit form

Cruising
Home
Book Now
Contact
Log Out
Dates
June
Start Date: 6/04/2019
End Date: 6/09/2019
July
Start Date: 7/04/2019
End Date: 7/09/2019
August
Start Date: 8/04/2019
End Date: 8/09/2019
September
Start Date: 9/04/2019
End Date: 9/09/2019
October
Start Date: 10/04/2019
End Date: 10/09/2019
November
Start Date: 11/04/2019
End Date: 11/09/2019

Cruising
Home
Book Now
Contact
Log Out
Suites
Silver Suite 1
Price: $399.99
Gold Suite 1
Price: $499.99
Platinum Suite 1
Price: $599.99
Silver Suite 2
Price: $399.99
Gold Suite 2
Price: $499.99
Platinum Suite 2
Price: $599.99

```html
<!DOCTYPE html>
<html>
<html>
The html element represents the root of an HTML document.
ps://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
<link rel="stylesheet" type="text/css" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.4.0/css/font-awesome.min.css">
<link href="http://fonts.googleapis.com/css?family=Lato:300,400,700,300italic,400italic,700italic" rel="stylesheet" type="text/css">

<link rel="stylesheet" type="text/css" href="appvideo.css">
</head>
<body onload='sendToHomePage()'>

<nav class="navbar navbar-default navbar-fixed-top">
<div class="container">
<div class="navbar-header">
<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">
<span class="sr-only">Toggle navigation</span>
<span class="icon-bar"></span>
<span class="icon-bar"></span>
<span class="icon-bar"></span>
</button>
<a class="navbar-brand" href="index.html">Cruising</a>
</div>
<div id="navbar" class="collapse navbar-collapse">
<ul class="nav navbar-nav">
<li class="active"><a href="index.html">Home</a></li>
<li><a href="destination_departure.html">Book Now</a></li>
<li><a href="contact.html">Contact</a></li>
</ul>
<ul class="nav navbar-nav navbar-right">
<li><a href="login.html" onclick="return logout();">Log Out <i class="fa fa-user"></i></a></li>
</ul>
</div>
</div>
</nav>

<div class="video-container">
<video autoplay loop muted id="video-bg">

<source src="pics/cruisevideo.mp4" type="video/mp4">

</video>
</div>
</div>
</header>

<script src="https://www.gstatic.com/firebasejs/5.8.3/firebase.js"></script>
```

```html
<script src="https://www.gstatic.com/firebasejs/5.8.2/firebase-auth.js"></script>
<script src="https://www.gstatic.com/firebasejs/5.8.2/firebase-database.js"></script>

<script>
function login() {
var userEmail = document.getElementById("email_field").value;
var userPsw = document.getElementById("psw_field").value;

firebase.auth().signInWithEmailAndPassword(userEmail, userPsw).catch(function(error) {
// Handle Errors here.
var errorCode = error.code;
var errorMessage = error.message;
if (error != null) {
window.alert("Error: " + errorMessage);
}
});
sendToHomePage();
}

function sendToHomePage() {
firebase.auth().onAuthStateChanged(function(user) {
if (user) {
// User is signed in.
var displayName = user.displayName;
var email = user.email;
var emailVerified = user.emailVerified;
var photoURL = user.photoURL;
var isAnonymous = user.isAnonymous;
var uid = user.uid;
var providerData = user.providerData;
window.open('index.html', '_self');
// ...
} else {
// User is signed out.
// ...
}
});
}

function signUp() {
window.open('signup.html', '_self');
}
</script>
</body>
</html>
```

```
confirmation.html ×   dates.html
<head>
The head element represents a collection of metadata for the Document.
<head>
  <title>Confirmation</title>
  <link rel="stylesheet" type="text/css" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.
  <link rel="stylesheet" type="text/css" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.4.0/css/font-aw
  <link rel='icon' href='favicon.ico' type='image/x-icon'/>
  <link href="http://fonts.googleapis.com/css?family=Lato:300,400,700,300italic,400italic,700italic" rel="sty
<style>
  body,
html {
    width: 100%;
    height: 100%;
    font-family: "Lato";
    color: white;
    background-image: url(pics/ocean1.jpg);
    background-repeat: no-repeat;
    background-size: cover;
    background-position: center;
}

h1 {
  padding-top: 5%;
  font-weight: 700;
  font-size: 5em;
}

.content{
  text-align: center;
    text-shadow: 0px 4px 3px rgba(0,0,0,0.4),
                 0px 8px 13px rgba(0,0,0,0.1),
                 0px 18px 23px rgba(0,0,0,0.1);
}

p {
  font-size:160%;
}
</style>
```

confirmation.html dates.html

```html
    pointer-events: none;
  }
  </style>
  </head>
<body>

<nav class="navbar navbar-default navbar-fixed-top">
 <div class="container">
   <div class="navbar-header">
     <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria
       <span class="sr-only">Toggle navigation</span>
       <span class="icon-bar"></span>
       <span class="icon-bar"></span>
       <span class="icon-bar"></span>
     </button>
     <a class="navbar-brand" href="index.html">Cruising</a>
   </div>
   <div id="navbar" class="collapse navbar-collapse">
     <ul class="nav navbar-nav">
       <li><a href="index.html">Home</a></li>
       <li class="active"><a href="destination_departure.html">Book Now</a></li>
       <li><a href="contact.html">Contact</a></li>
     </ul>
     <ul class="nav navbar-nav navbar-right">
       <li><a href="login.html" onclick="return logout();">Log Out    <i class="fa fa-user"></i></a></li>
     </ul>
   </div>
 </div>
</nav>

<div class="container">
 <div class="row">
   <div class="col-lg-12">
     <div class="content">
       <h1>Dates</h1>
     </div>
   </div>
 </div>
```

GitHub link: https://github.com/riyazhdholakia/CSC-SWE-Friends