**Name of Project: Book a Cruise**

**Website Name: Cruise Services**

Semester: Spring 2019

Group Number: 1

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

Date of Submission: March 2nd, 2019

**Scheduling and planning:**

**Work Breakdown Structure**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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, and Database | 4 hours | Microsoft Visio Download, Visual Studio Code Download, Use Case(s), Requirements | 3/2/19 | 100% |
| Sujal Patel | spatel287@student.gsu.edu | Problem Statement, Requirements, Test Cases, GitHub Project Board, Readme, Front-End | 4 hours | Microsoft Visio Download, Visual Studio Code Download, Use Case(s), Requirements, | 3/2/19 | 100% |
| Randall Byrd | rbyrd11@student.gsu.edu | Requirements, Use Case Diagram, Test Cases, Architectural modeling, Database | 4 hours | Microsoft Visio Download, Visual Studio Code Download, Use Case(s), Requirement | 3/2/19 | 100% |
| Syed Kalam  **(Coordinator)** | skalam1@student.gsu.edu | Planning Table, Class Diagram, Requirements, Test Cases, Behavioral modeling, Database | 4 hours | Everyone’s work to format project, Microsoft Visio Download, Visual Studio | 3/2/19 | 100% |
| Biprojoyti Paul | bpaul5@student.gsu.edu | Problem Statement, Requirements, Test Cases, Behavioral modeling, Front-End | 4 hours | Microsoft Visio Download, Visual Studio Code Download, Use Case(s), Requirement | 3/2/19 | 100%. |

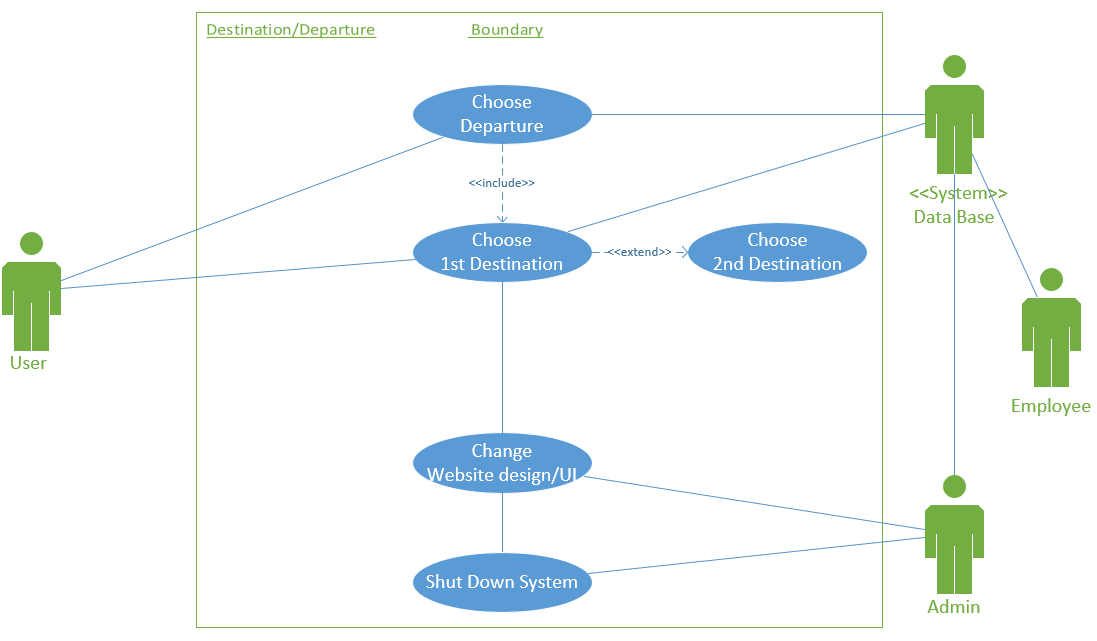
**Problem Statement:**

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 are make 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 mind 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.

**System Modeling:**

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 is available, 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.

**Requirements:**

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: 1.1,1.2,1.3

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: 2.1, 2.2

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 unselect able (grayed out). Checking with Database constantly.

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

Test Cases: 3.1, 3.2, 3.3

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: 4.1, 4.2, 4.3

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: 4.1, 4.2, 4.3

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: 6.1, 6.2, 6.3

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: 7.1, 7.2

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: 8.1, 8.2

Requirement #: 9

Use Case #: 9

Name: Contact

Introduction: Contact page to help User contact us if any issues.

Rationale:

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: 9.1, 9.2

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

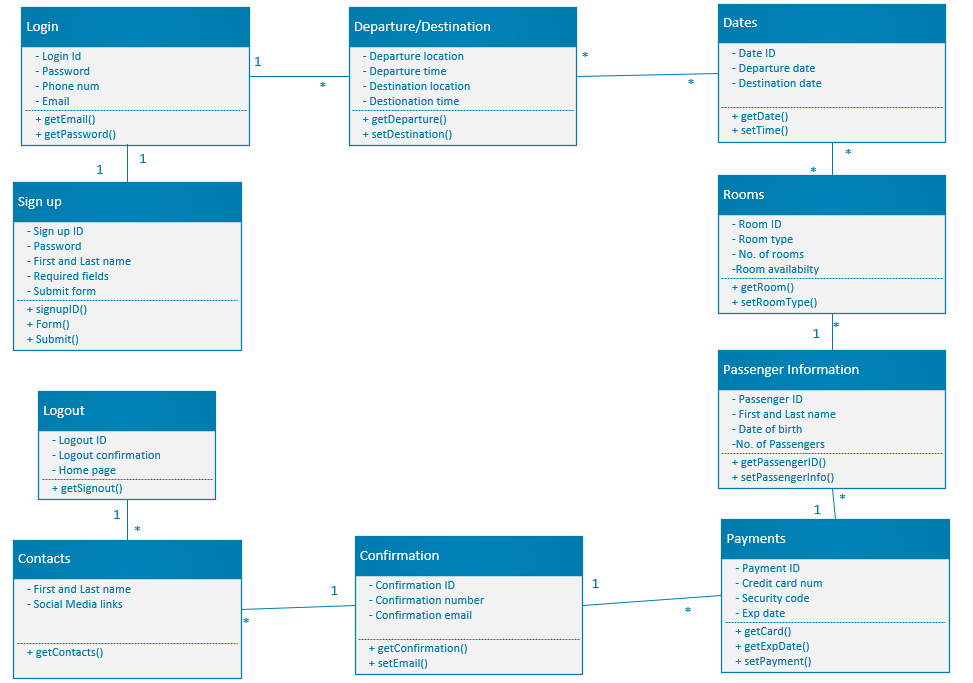
Requirement Description: User should Logout after done with our site.

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

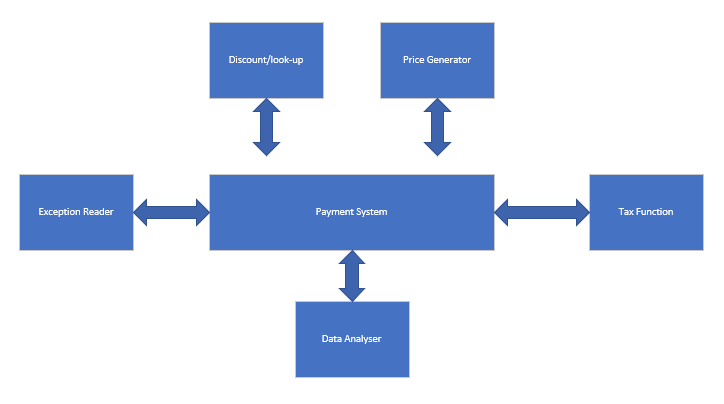
Output: Logged out of the page, back to the login page.

Test Cases: 10.1, 10.2

Class Diagram:

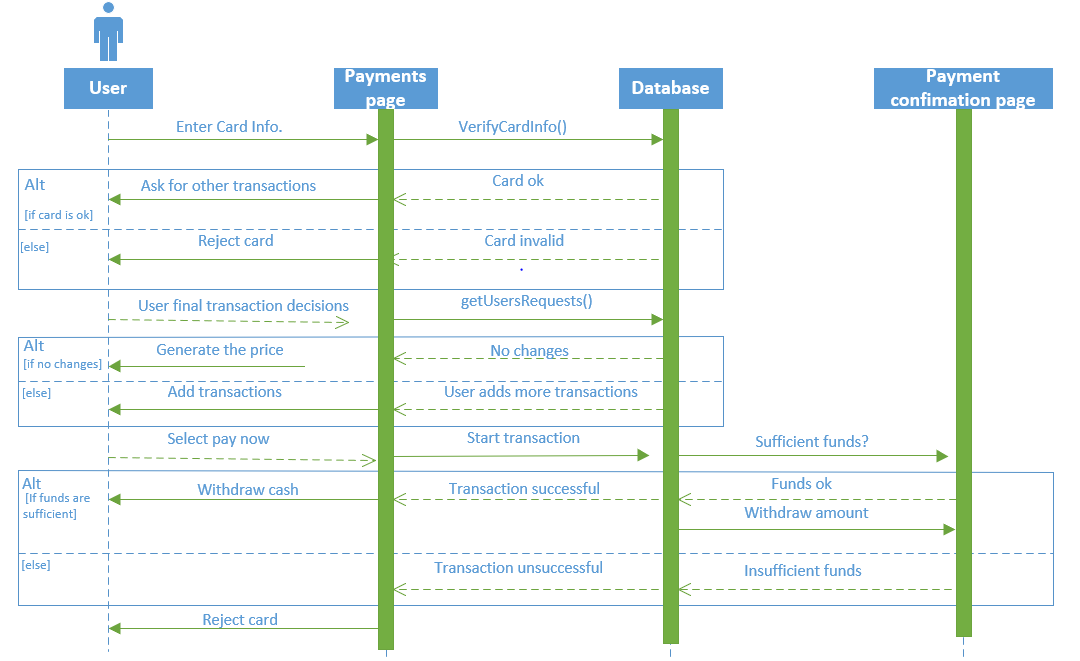
****

Architecture model:



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.

Sequence diagram:



The User enters card information and the database verifies whether the card is valid or not. Once that goes though the user is prompted to add more transactions to their cart. The user will either add more to their cart or move on to pay now. In pay now, database will then start the transaction and it's up to the payments confirmation page to decide whether there are sufficient funds. If there is money on the card the transaction will be successful, and the user will get a confirmation, or else the card is rejected.

**Implementation:**

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. To run our code please read our readme file which is also located on GitHub. The thing to do is clone project and run it. 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.

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)

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.

Please see screenshots and GitHub link in Appendix, for full implementation.

**Testing:**

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.

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.

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.

Identity feature: select (int a[])

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.

Identity feature: select (int a[])

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.

Identity feature: select (int a[])

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 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

Identity feature: next ()

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: Enter the date, which has no space for more passengers.

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.
2. Select the other dates for the departure which has capacity for the passengers.

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.

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.

Identity feature: FillIn (int a)

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.

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.

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.
3. User must hit submit to get unsuccessful message.

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.

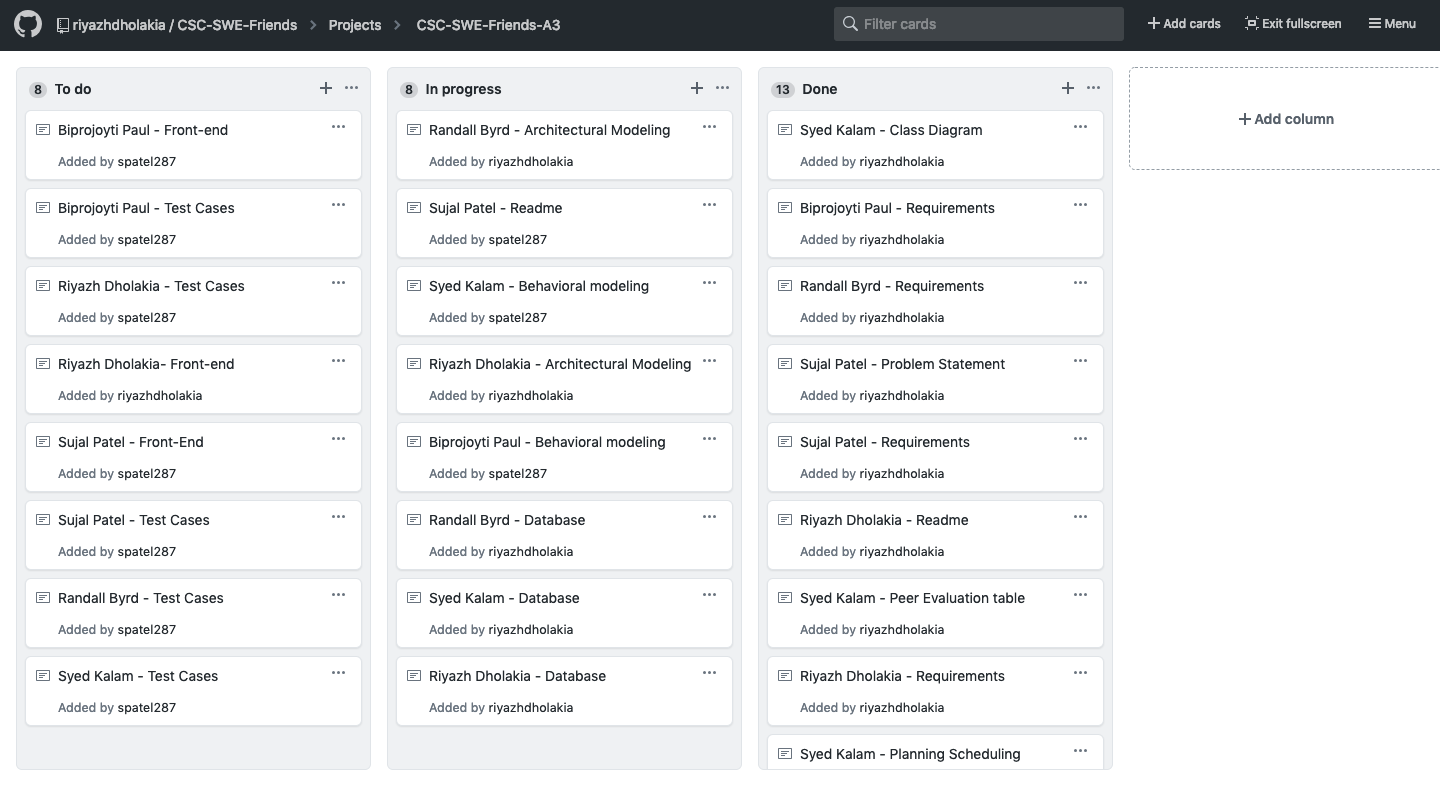
Initialization: none

Test Steps:

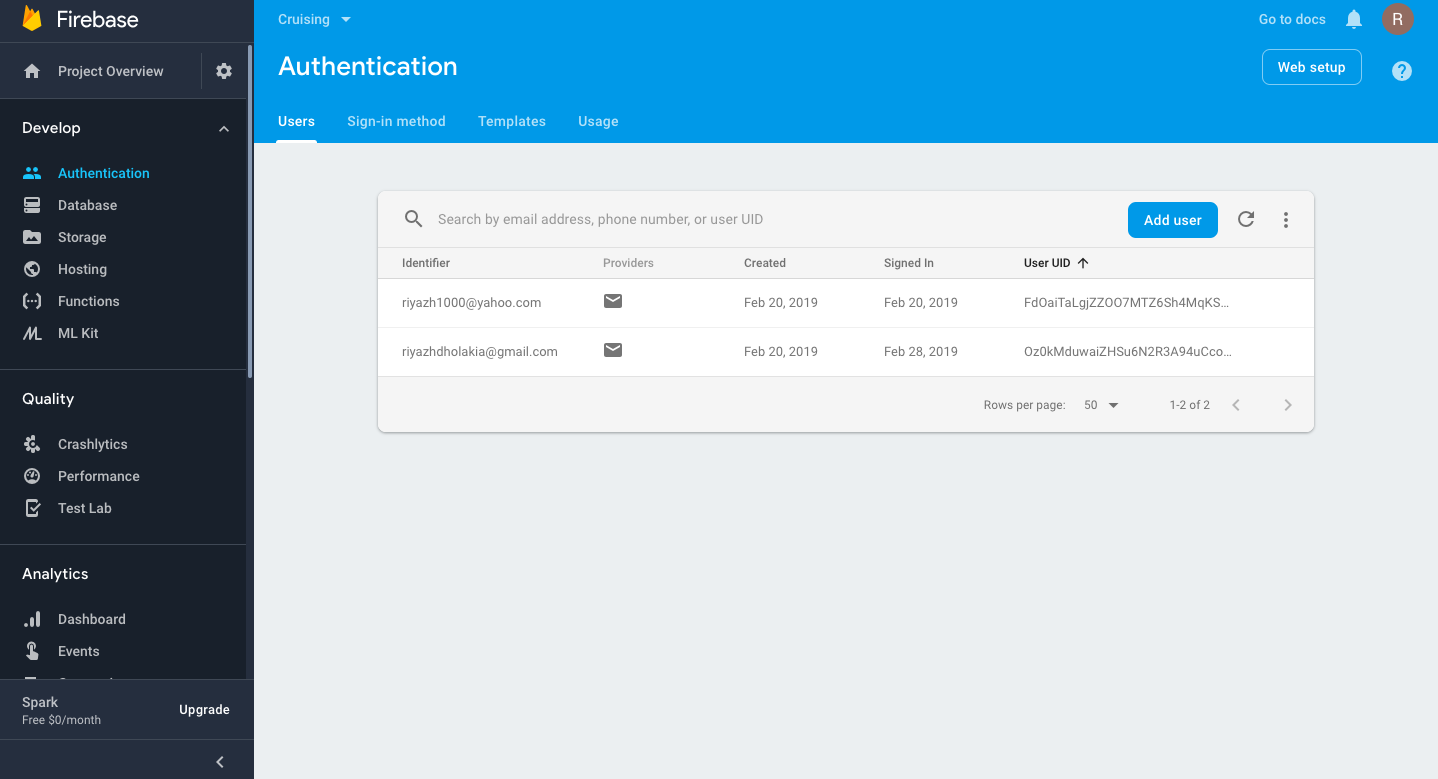
1. It will be just click of button, on the navigation bar.
2. Error message to try again.

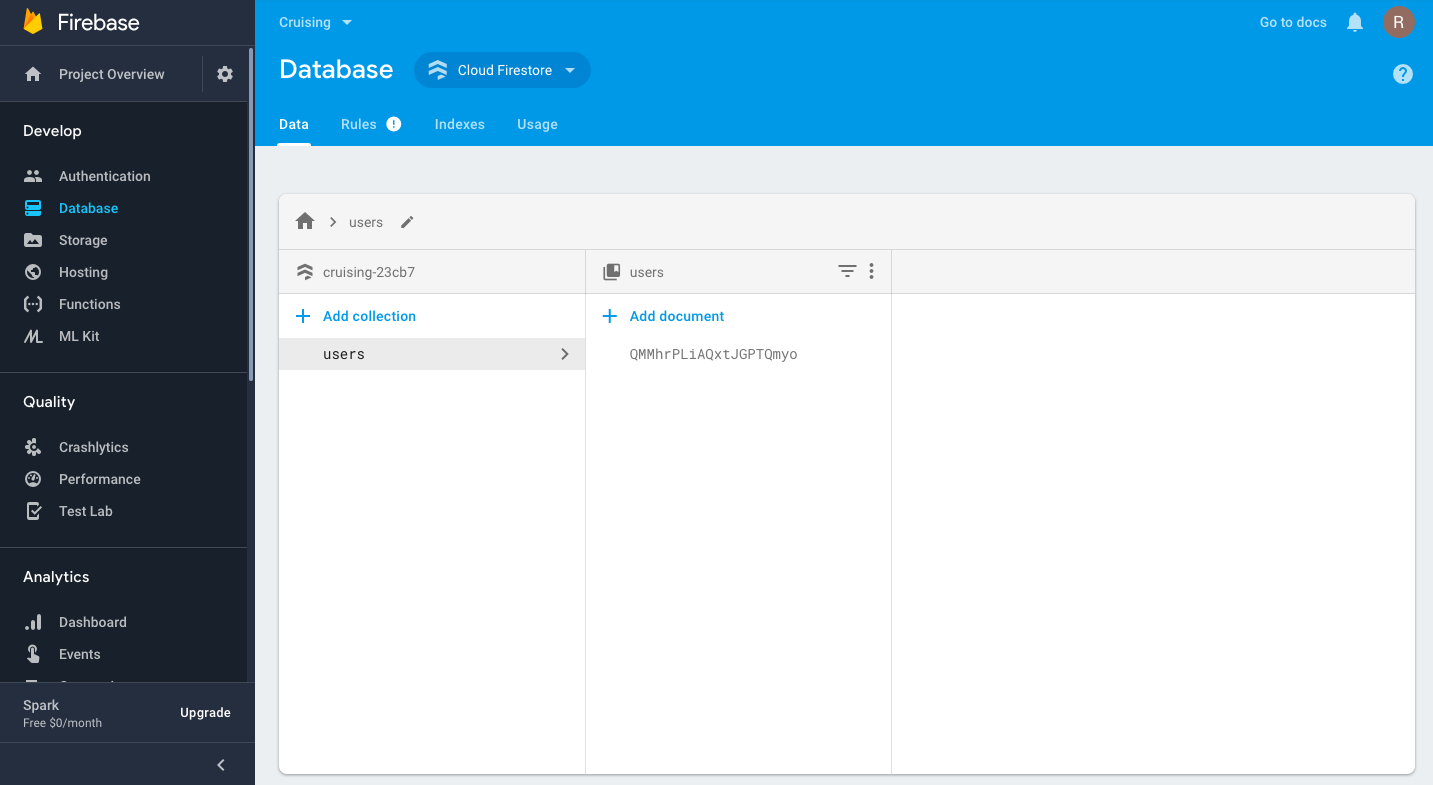
**Appendix:**

**Communication and Collaboration:**

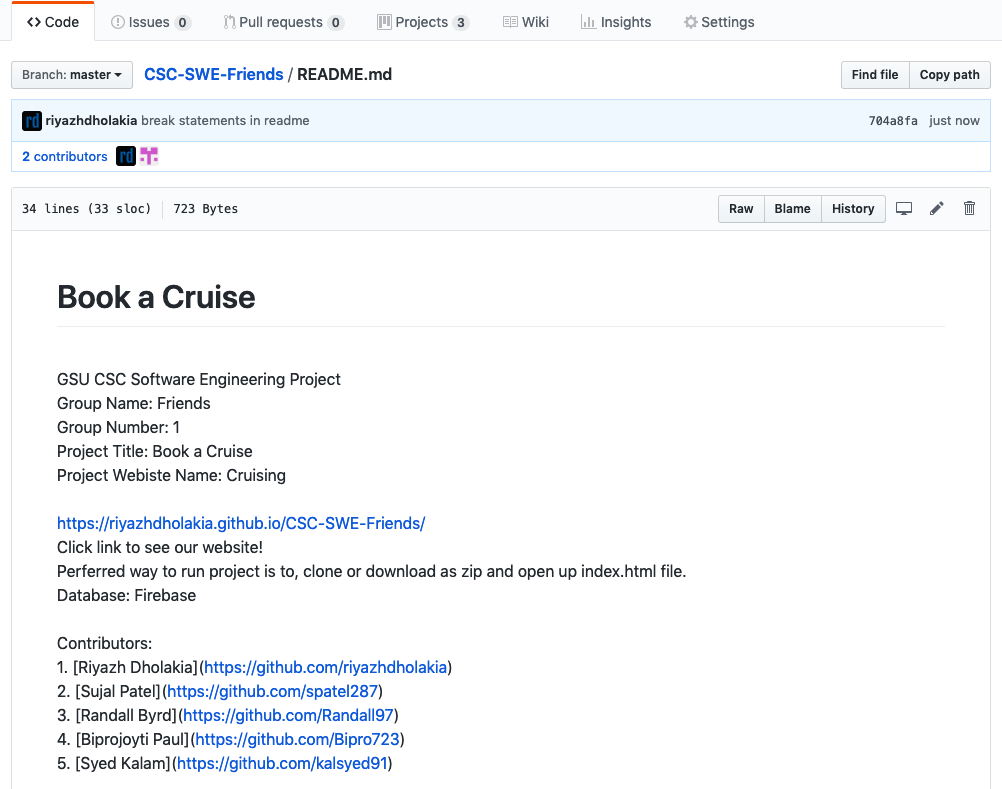


**Firebase:**

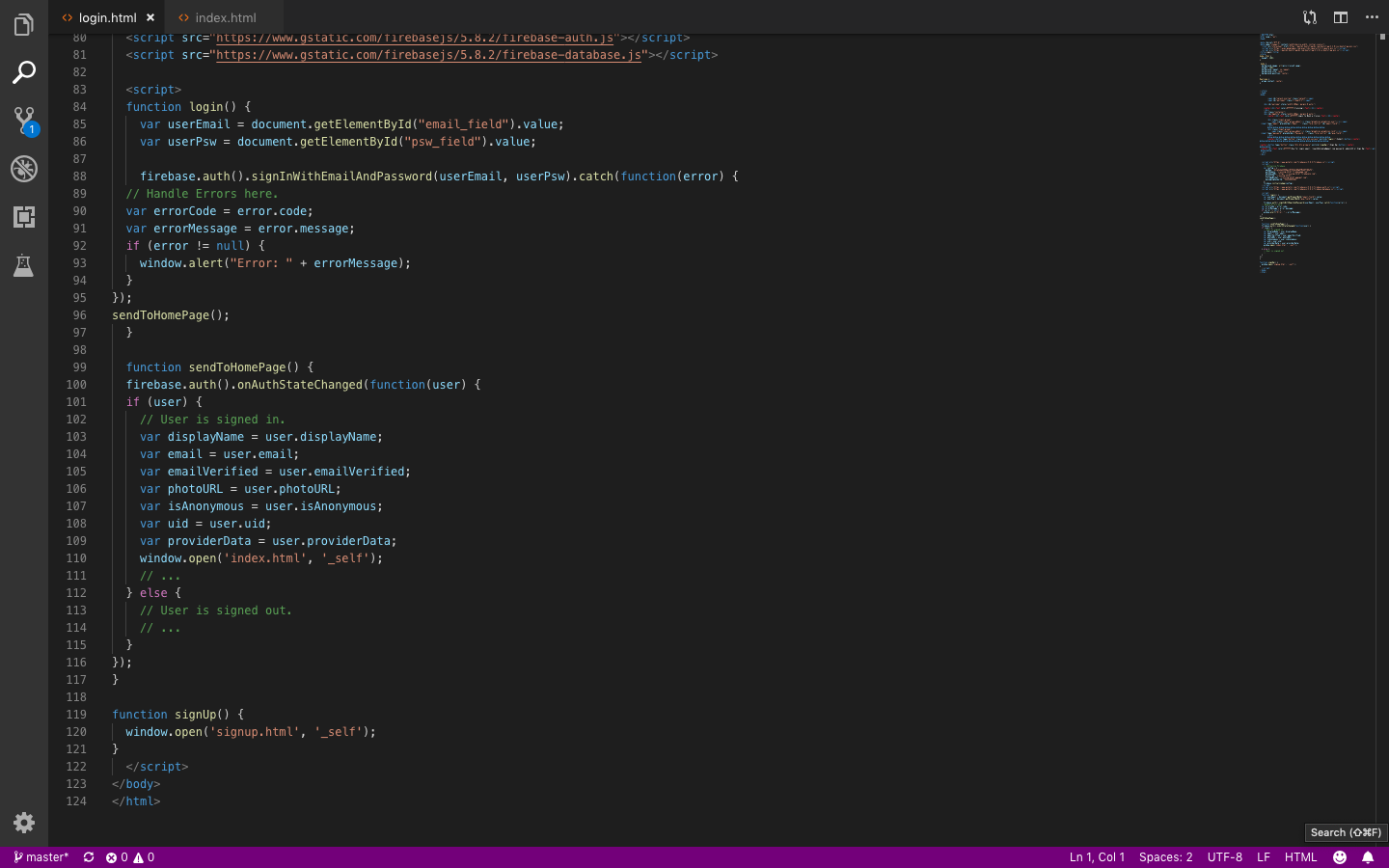
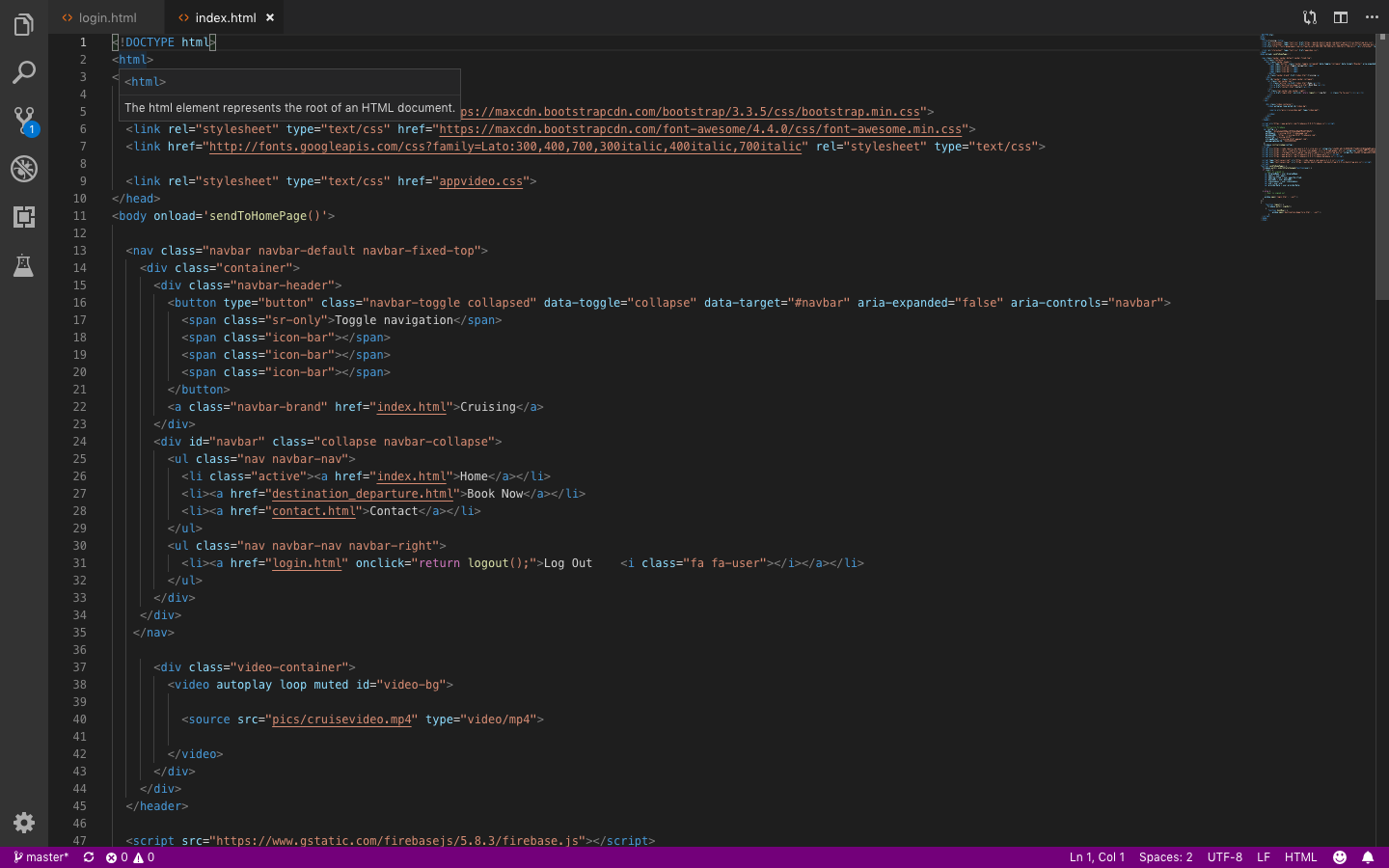
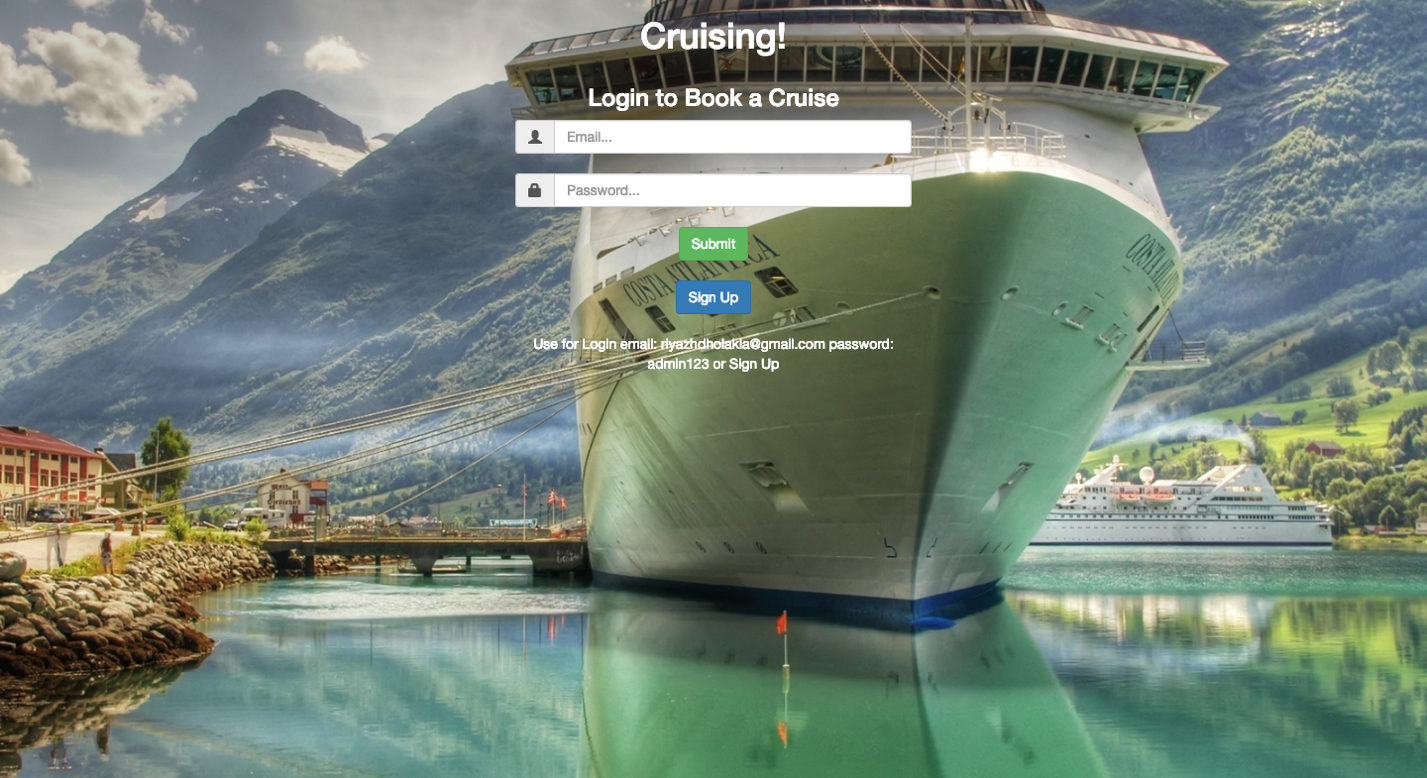




**Readme:**



**Screenshots front-end design, front-end, and database code:**



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