

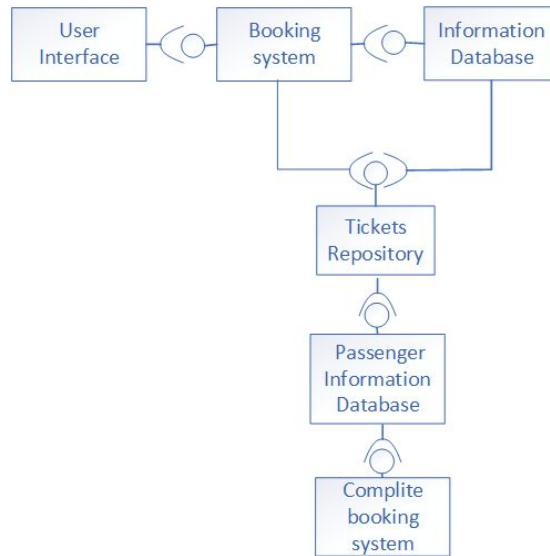
Tickets Reservation – Delta, Greyhound, Amtrak
Terminal
Spring 2019
SSJKV
Group 2
Kamal Rimal, Sarah Swilley, Saleh Alhassan, Juan
Martinez, Viktoriya Rasuli
4/14/2019

Work Breakdown Structure

Assignee Name	Email	Task	Duration (hours)	Dependency	Due date	Evaluation
Saleh Alhassan (coordinator)	salhassan1@student.gsu.edu	Implement the Database Design. Implement the Test case.	8 hours	All parts of the report should be ready.	4/13/2019	100% Should be ready within 24 hours before a deadline.
Juan Martinez	Jmartinez41@student.gsu.edu	Test Case Script. Test Documentation and create new Test cases.	6 hours	Implement The test case should be done first.	4/13/2019	100% Should be ready within 24 hours before a deadline Work with Kamal on test case script.
Sarah Swilley	sswilley1@student.gsu.edu	Implement the Class Diagram Design (Frontend and Logic). Put report together.	8 hours	Implement the Database Design. This should be done first.	4/14/2019	100% Work with Viktoriya on Implement the Class Diagram Design. Report should be ready within 7 hours before a deadline.
Kamal Rimal	krimall@student.gsu.edu	Do Revise and Refine System. Test Case Script.	6 hours	None	4/12/2019	100% Should be ready within 48 hours before a deadline. Work with Juanon test case script.
Viktoriya Rasuli	Vrasuli1@student.gsu.edu	Creates new To do; In Progress and Done in the GitHub as described in an assignment. Implement the Class Diagram Design (Frontend and Logic). Architecture modeling.	7 hours	Implement the Database Design. This should be done first.	4/12/2019	100% Behavioral should be ready within 48 hours before a deadline. Work with Sarah on Implement the Class Diagram Design.

Revise and Refine your System

Based on the feedback that was provided to us in the A4, we redid Logical View look.



Requirements

Requirement: Log In

Introduction: We will have a login page that will ask for users username and password.

Rationale: Make it easier to track transactions and know who is buying the tickets.

Input: User's username and password

Requirements Instructions:

1. The user must have completed registration in order to have a login.
2. The system must have the information from the username and password stored and ready to be checked

Output: Profile

Test Cases:

Test Case ID: 1.1

Description: Attempt login on the main page of the website.

Test Inputs: Email and password corresponding to a valid user account.

Expected Results: Login Successful. User is then presented with another page that will show the bus information.

Dependencies: None

Initialization: Users account must already have been created.

Test Steps:

1. Submit valid username and password on a login page.

Test Case ID: 1.2

Description: Attempt to login without incorrect user information.

Test Inputs: Email and password are not correct.

Expected Results: Login Unsuccessful. The user will be notified that the user name in password doesn't match.

Dependencies: None

Initialization: Database information that matches the correct user information.

Test Steps:

1. Submit invalid username and password on a login page.

Requirement: Change password

Introduction: We will have a user page that will ask for user his old and a new password.

Rationale: A user must be easily able to change password.

Input: User's old password, and new password

Requirements Instructions:

1. The user must already have created account, be logged in and entered profile page in order to change password.
2. The system should add a replace an old password with new one in the database.

Output: User profile page

Test Cases:

Test Case ID: 2.1

Description: User attempts to change the password on his account.

Test Inputs: User enters an old password and new password into assigned fields.

Expected Results: Change of password is successful. The user redirects to a user profile page.

Dependencies: A user should log in first before be able to change the old password.

Initialization: Database with information about user password and login name.

Test Steps:

1. Submit valid old password and then new password into the provided fields.
2. Click button saves changes that locate at the center-bottom page.

Test Case ID: 2.2

Description: User attempts to change the password on his account.

Test Inputs: User enters an invalid old password and then new password into assigned fields.

Expected Results: Change of password is unsuccessful. The user stays on the same page and sees a notification that the old password is not matching with entered one.

Dependencies: A user should log in first before be able to change the old password.

Initialization: None

Test Steps:

1. Submit invalid old password and then new password into provided fields.
2. Click button saves changes that locate at the center-bottom page.

Requirement: View bus info

Introduction: The clear and colorful page which will provide a list of specific and clear understandable information about each bus' trip.

Rationale: A user must be able to easily find a trip to their destination by bus.

Input: Click the bus button.

Requirements Instructions:

1. User have to clicks at the button named bus that locates at the center of the main page.
2. The system must be able to display a list of busses with the their information from the database.

Output: The page has a list of available buses with information for each bus such as place of departure and arrival, date of departure and return, and time.

Test Cases:

Test Case ID: 3.1

Description: Attempt of the user views bus information.

Test Inputs: Click the bus button that at the center of the main page.

Expected Results: User provides a dashboard with bus information.

Dependencies: User should first visit the main page before to view the bus information.

Initialization: Database with information about place departure, place arrival and date of departure and return.

Test Steps:

1. Click button that at the center of the main page.

Test Case ID: 3.2

Description: Attempt of the user views bus information.

Test Inputs: Click the bus button that at the center of the main page.

Expected Results: Access of the page is unsuccessful.

Dependencies: User should first visit the main page before to view the bus information.

Initialization: None

Test Steps:

1. Click button that at the center of the main page.
2. Refresh the page and try to click at the button again.

Requirement: Check Bus availability

Introduction: The page must be able to have a clear way of navigation to the previous page. There should also be a list of available busses from the database that pulls it up with the bus information.

Rationale: Make it easy for the user to know what busses are available.

Input: Click the bus button that at the center of the main page.

Requirements Instructions:

1. User clicks at the bus button that at the center of the main page.
2. The system must be able to display a list of available busses from the database.

Output: A list of available bus with their information.

Test Cases:

Test Case ID: 4.1

Description: User attempts to check a bus availability.

Test Inputs: User clicks at the bus button.

Expected Results: On the next page displays all available buses with their information.

Dependencies: A user should be on the home page.

Initialization: Database information with date, time, total number of seat in the bus, locations of departure and arrival.

Test Steps:

1. Click on the bus button that locates at the center of the main page.

Test Case ID: 4.2

Description: User attempts to check a bus availability.

Test Inputs: User clicks at the bus button.

Expected Results: A user stays at the same page and can't check the bus availability.

Dependencies: A user should be on the home page.

Initialization: Database accesses bus information, and then cannot disconnect from it.

Test Steps:

1. Click on the bus button that locates at the center of the main page.
2. Refresh page and try again by clicking the same button.

Requirement: Book Bus ticket

Introduction: The user confirms the bus trip which they have selected.

Rationale: To make it easy for the user to know what bus trip they have selected and clear as to what they will be getting.

Input: The bus trip from the database with time and city. Also, click the icon of the bus or on its name.

Requirements Instructions:

1. Receive information from the database.
2. Show the user the time and date selected.
3. User should click the icon of the bus or on its name.

Output: User will be transferred to the next page to fill personal information for the ticket reservation.

Test Cases:

Test Case ID: 5.1

Description: User attempts to book a bus ticket.

Test Inputs: User clicks at the icon of the bus or on its name.

Expected Results: User transfers to the next page to fill passenger information and choose seat.

Dependencies: User should be login.

Initialization: None

Test Steps:

1. Clicks at the chosen icon of the bus or on its name.

Test Case ID: 5.2

Description: User attempts to book a bus ticket.

Test Inputs: User clicks at the icon of the bus or on its name.

Expected Results: Procession of the booking ticket is unsuccessful.

Dependencies: User should be login

Initialization: None

Test Steps:

1. Click at the chosen icon of the bus or on its name.
2. Refresh page and try again by clicking the icon or bus name.
3. Go back to the main page and try access this page again.

Requirement: Receives Orders

Introduction: The system will receive an order for what bus trip the user has selected

Rationale: To start the authentication process and give the user a confirmation code.

Input: The bus trip that was selected by the user.

Requirements Instructions:

1. User should complete filling personal information and chose the seat.
2. System collects information provided by the user and matching with database.

Output: Selected bus trip.

Test Cases:

Test Case ID: 6.1

Description: Authentication received order.

Test Inputs: JWT token.

Expected Results: Authentication receives order and process it.

Dependencies: User should be log in first and complete all needed section to reserve a ticket.

Initialization: Database information with date, time, location of departure and arrival, total number of seat in the bus, chosen seat.

Test Steps:

1. Click button Submit at the center-bottom of a page with choice seat and filling passenger information.

Test Case ID: 6.2

Description: Authentication received order.

Test Inputs: JWT token.

Expected Results: Authentication failed receiving the order.

Dependencies: User should be log in first and complete all needed section to reserve a ticket.

Initialization: Database accessed information provided by user but can't match it with an existing one on database.

Test Steps:

1. Click button Submit at the center-bottom of a page with choice seat and filling passenger information.

2. Try again by entering information once again and then click submit button again.

Requirement: View Orders

Introduction: The system will check the database for any conflict with the information given by the user.

Rationale: To make sure that the ticket that the user has selected is available.

Input: The information selected by the user for the bus trip selected.

Requirements Instructions:

1. User should provide information required for booking ticket.

2. System compares data selected with the database and look for conflict.

Output: Ticket information.

Test Cases:

Test Case ID: 7.1

Description: Authentication attempts to check received order.

Test Inputs: JWT token.

Expected Results: Authentication checked the received order.

Dependencies: A user should be log in to the website and choose bus with time and destination, also filled personal information and choose the seat.

Initialization: Database matching user information with information which is already there.

Test Steps:

1. Click button Submit on passenger information page.

Test Case ID: 7.2

Description: Authentication attempts to check received order.

Test Inputs: JWT token.

Expected Results: Authentication can't checked the received order and can't process it.

Dependencies: A user should be log in to the website and choose bus with time and destination, also filled personal information and choose the seat.

Initialization: None

Test Steps:

1. Click button Submit on passenger information page.

2. User should try to click button submit again after re entering passenger information.

Requirement: Provides Ticket

Introduction: After confirming that there is no conflict with the system will put all the information together on the ticket.

Rationale: Make and deliver a ticket for the user

Input: Information received from the database that was selected by the user.

Requirements Instructions:

1. Receive information from the database.
2. Show the user the time and date selected in a ticket.

Output: Send notification to the confirmation page that reservation of ticket was success.

Test Cases:

Test Case ID: 8.1

Description: Authentication after checking information provides user with a ticket.

Test Inputs: JWT token.

Expected Results: Authentication process order successful; user will see the confirmation page.

Dependencies: A user should log in first and provide information for reserving ticket.

Initialization: Database finds information about availability seats and updates it with a new.

Test Steps:

1. Click button Submit on passenger information with availability seat page.

Test Case ID: 8.2

Description: Authentication after checking information provides user with a ticket.

Test Inputs: JWT token.

Expected Results: Authentication process order unsuccessful; user will stay at the same page.

Dependencies: A user should log in first and provide information for reserving ticket.

Initialization: None

Test Steps:

1. Click button Submit on passenger information with availability seat page.
2. User should try to click button submit again after re entering passenger information.

Requirement: Choose seats

Introduction: The user will be able to select a seat from available seats on a bus.

Rationale: Organizing and knowing who is where and to keep a tally of available seats.

Input: Click at the checkbox and then at the button choose.

Requirements Instructions:

1. A user has to choose the seat from the provided available one by clicking the button choose.
2. The system temporarily secure the seat to the user, until it finishing providing personal information.

Output: The user will see a window that is going to ask fill personal information of passenger.

Test Cases:

Test Case ID: 9.1

Description: User attempts to choose seats.

Test Inputs: User must be already logged in.

Expected Results: Successful. A user has chosen his seat. User moves to the next page.

Dependencies: A user should log in and already choose the trip before to choose seats on the bus.

Initialization: Database number and letter of seat and row of available seats.

Test Steps:

1. Click at the Drop-down list and choose a seat from available.
2. Click at button submit that at the center-bottom page.

Test Case ID: 9.2

Description: User attempts to choose seats.

Test Inputs: User must be already logged in.

Expected Results: Unsuccess. A user did not choose a seat. User stays at the same page.

Dependencies: A user should log in and already choose the trip before to choose seats on the bus.

Initialization: None

Test Steps:

1. Don't click at the Drop-down list and don't choose a seat.
2. Click at button submit that at the center-bottom page.

Requirement: Enter passenger info

Introduction: The user will be taken to a new page where they can give us their personal information and the people that they are traveling with.

Rationale: To know which person is seating in what seat and to keep the inventory of available seats

Input: Name, address, phone, email.

Requirements Instructions:

1. The user must know all the information and must submit it to continue.
2. The system will add the information and put it on the ticket.

Output: The information that was given.

Test Cases:

Test Case ID: 10.1

Description: User attempts to enter valid passenger information to continue booking tickets.

Test Inputs: Name, phone number, and email.

Expected Results: All fields filled successfully. User will move to the next page.

Dependencies: A user should log in first, choose the trip.

Initialization: None

Test Steps:

1. Enter the passenger information into the provided fields.
2. Click button submit at the center-bottom page.

Test Case ID: 10.2

Description: User attempts to enter invalid passenger information (enters the phone number in the name field or name in the phone number) to continue booking tickets.

Test Inputs: Name, phone number, and email.

Expected Results: Booking of the ticket unsuccessful.

Dependencies: A user should log in first, choose the trip.

Initialization: Catch try system should be already implemented to check the fields.

Test Steps:

1. Enter the wrong passenger information in the provided fields.
2. Click button submit at the center-bottom page.

Requirement: Complete booking

Introduction: The user will select to complete the booking and there will be a confirmation number given to them.

Rationale: Confirming that the user has selected the right bus trip and for the user to keep track of their upcoming trip.

Input: 'Complete booking' icon.

Requirements Instructions:

1. The user will click to confirm the trip
2. The system will provide a confirmation number.

Output: Confirmation number.

Test Cases:**Test Case ID: 11.1**

Description: A user sees a confirmation page with all provided information.

Test Inputs: User must be already complete reservation tickets.

Expected Results: Confirmation page with correct information on it.

Dependencies: User should log in first and finish ticket reservation.

Initialization: Database shows information provided by a user.

Test Steps:

1. Click at button submit that at the center-bottom page.

Test Case ID: 11.2

Description: A user sees a confirmation page with all provided information.

Test Inputs: User must be already complete reservation tickets.

Expected Results: Confirmation page with incorrect information on it.

Dependencies: User should log in first and finish ticket reservation.

Initialization: None

Test Steps:

1. Click at button submit that at the center-bottom page.
2. Return to the previous page and try again by clicking the button.

Requirement: Print ticket

Introduction: The user will be able to print a ticket or send it on their email.

Rationale: To authenticate the transaction.

Input: The ticket info from the previous screen.

Requirements Instructions:

1. The system will give them a ticket to print out or to send.
2. The user will be able to print or save their ticket.

Output: Ticket.

Test Cases:**Test Case ID: 12.1**

Description: The user after finishing tickets reservation trying to print tickets.

Test Inputs: User has to complete a reservation ticket.

Expected Results: User receives the ticket and prints it successfully.

Dependencies: User should log in first and finish ticket reservation.

Initialization: None

Test Steps:

1. Click at button print the ticket that at the center-bottom page.
2. Click at button send the ticket that at the center-bottom page

Test Case ID: 12.2

Description: The user after finishing tickets reservation trying to print tickets.

Test Inputs: User has to complete a reservation ticket.

Expected Results: User receives the ticket and can't print it.

Dependencies: User should log in first and finish ticket reservation.

Initialization: None

Test Steps:

1. Click at button print the ticket that at the center-bottom page.
2. Click at button send the ticket that at the center-bottom page
3. Refresh page and then try again by clicking the same button.

Requirement: Logout

Introduction: The user will be able to save and exit out of the website

Rationale: To keep track of people that are using and booking tickets

Input: Click the logout button

Requirements Instructions:

1. The user needs to be signed
2. The system will log the user out after 30 minutes of no use

Output: Back to Home screen

Test Cases:

Test Case ID: 13.1

Description: User attempts to logout after the ticket is booked.

Test Inputs: User must be already logged in.

Expected Results: Logout successful.

Dependencies: Must be logged in. Must have a logout button.

Initialization: Database accesses user information, and then disconnects from it.

Test Steps:

1. Click the logout button at the top right-hand side of the page.

Test Case ID: 13.2

Description: User attempts to logout after the ticket is booked.

Test Inputs: User must be already logged in.

Expected Results: Logout unsuccessful.

Dependencies: Must be logged in. Must have a logout button.

Initialization: Database accesses user information, and then cannot disconnect from it.

Test Steps:

1. Click the logout button at the top right-hand side of the page.
2. Go back to the main page and try clicking the button again.

System Modeling (Analysis)

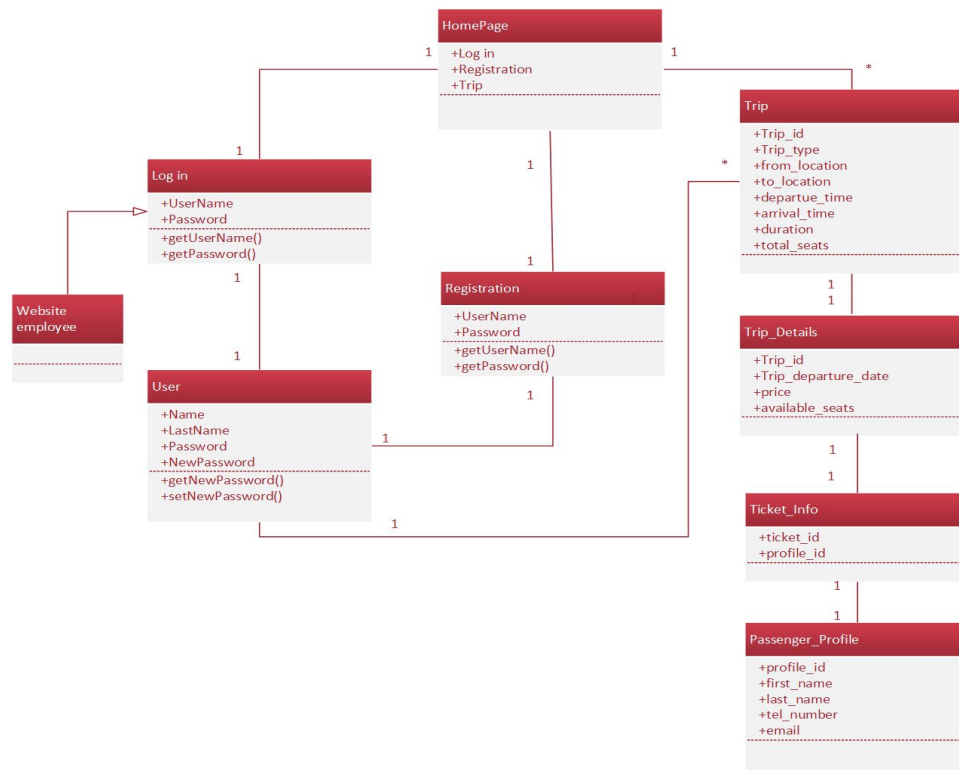
Class Diagram:

Our class diagram has nine objects such as HomePage, Log in, Trips, Registration, Trip_Details, Ticket_info, Passengare_Profile, User, Website Employee. All of the objects in our diagram have an association with other objects except for Website Employee object that has a directed association with Log in object. The multiplicity almost between every object is the same and equal to one to one. Only in two cases, we have different multiplicity that is not equal one to one, it between objects HomePage and Trips as also between objects User and Trips. In both cases, it equals one to many.

All object have several attributes in our class diagram, except one which is Website Employee, because it object playing the role of the person. The attributes for HomePage object are log in, registration and trips. In the Log in and Registration object, attributes that we are going to use are username and password. In the User object, we will use name, lastname, password and newpassword attributes. The Trip object will have attributes such as trip_id, trip_type, from_location, to_location, departure_time, arrival_time, duration, and total_seats. In the Trip_Details object, we will have four attributes trip_id, trip_departure_data, price and available_seats. The Ticket_Info object has attributes ticket_id

and profile_id. The last object is Passenger_Profile object. It will have attributes such as profile_id, first_name, last_name, tel_number and email.

We decided in our class diagram that only three objects out of nine will have operations. These three objects are Log in, User and Registration where Log in and Registration have identical operations which are getUserName() and getPassword(). The User object has getNewPassword() and setNewPassword().



Database Specification and Analysis:

We will use a MongoDB No-SQL database, and as such, the structure of the database and terminology used will be different from a normal SQL database. This means that there will be no “tables” like there would be if we were using a MySQL database. Instead, there will be data collections which are comprised of multiple documents. We will have two main data collections: user and trip collections. The user collection will have user documents with the following data attributes and type pair: (user_id: ID type), (user_name: string), (password: string), and a nested user_tickets document having: (ticket_id: ID type), (trip_id: ID type). The trip documents’ pair will be: (trip_id: ID type), (trip_type: string), (location_to: string), (location_from: string), (day: string), (departure_time: string), (arrival time: string), (total_seats: integer), and (price: integer). It is important to note that if this was a MySQL database, there would have been four tables: User, Trip, Ticket, and Location tables. But again, since we are using MongoDB, we have a different database structure. The primary keys for the user, ticket, and trip documents are user_id, ticket_id, and trip_id respectively. Below are two images illustrating the structure of the two database documents as discussed above.

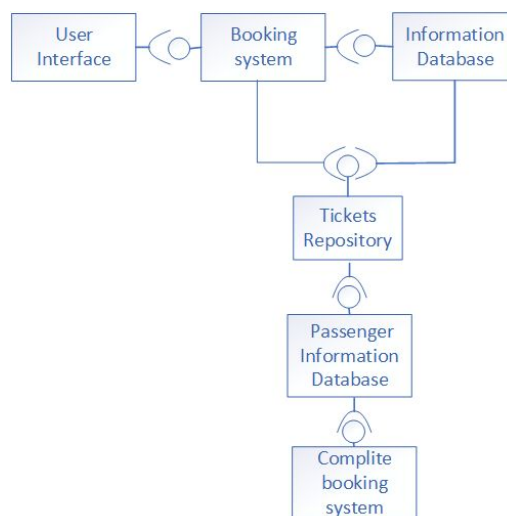
```

_id: ObjectId("5c74c5ba02932c20b0c256f8")
name: "Newer User"
password: "password"
__v: 0
✓ tickets: Object
  ticket_id: ObjectId("5c7a0c0b7d203a0000e66ad0")
  trip_id: ObjectId("5c75c1fe22b7e431643bc2a7")
  _id: ObjectId("5c75c1fe22b7e431643bc2a7")
  trip_type: "Bus"
  location_to: "Boston"
  location_from: "Atlanta"
  day: "Monday"
  departure_time: "1:00 pm"
  arrival_time: "6:00 pm"
  total_seats: 37
  price: 500
  __v: 0

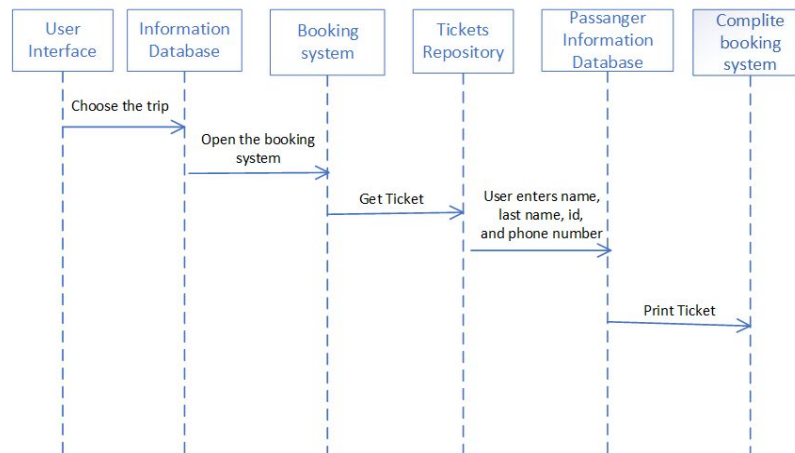
```

Architecture modeling:

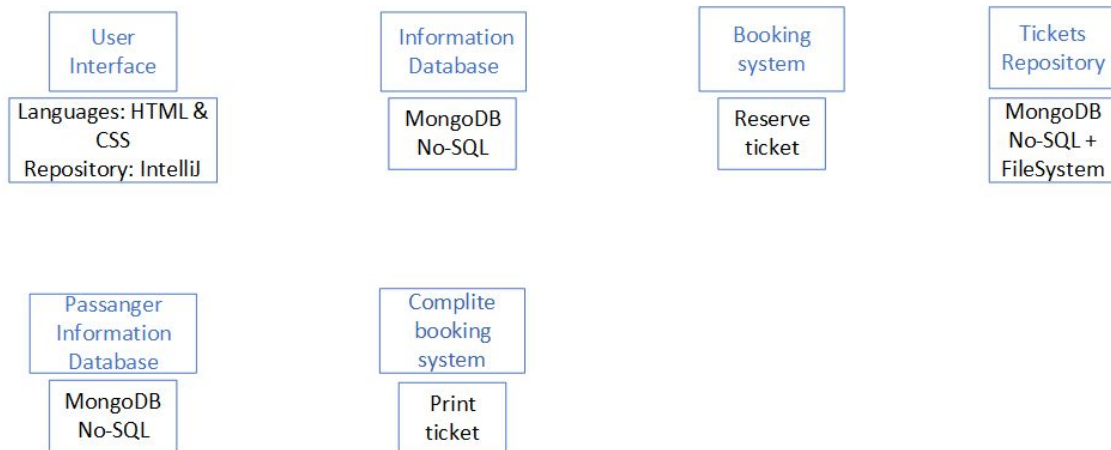
Logical View



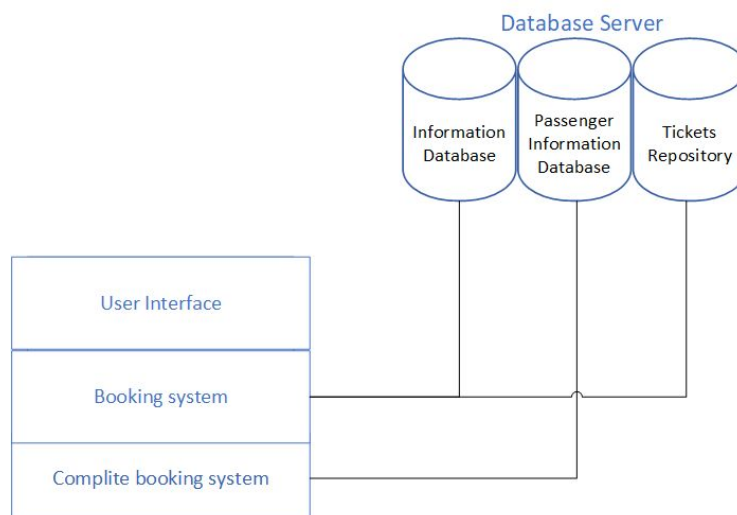
Process View



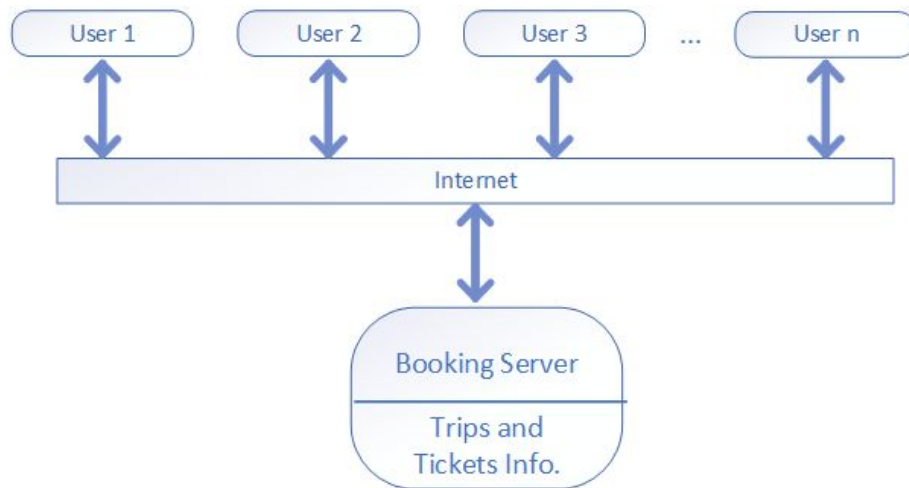
Development View



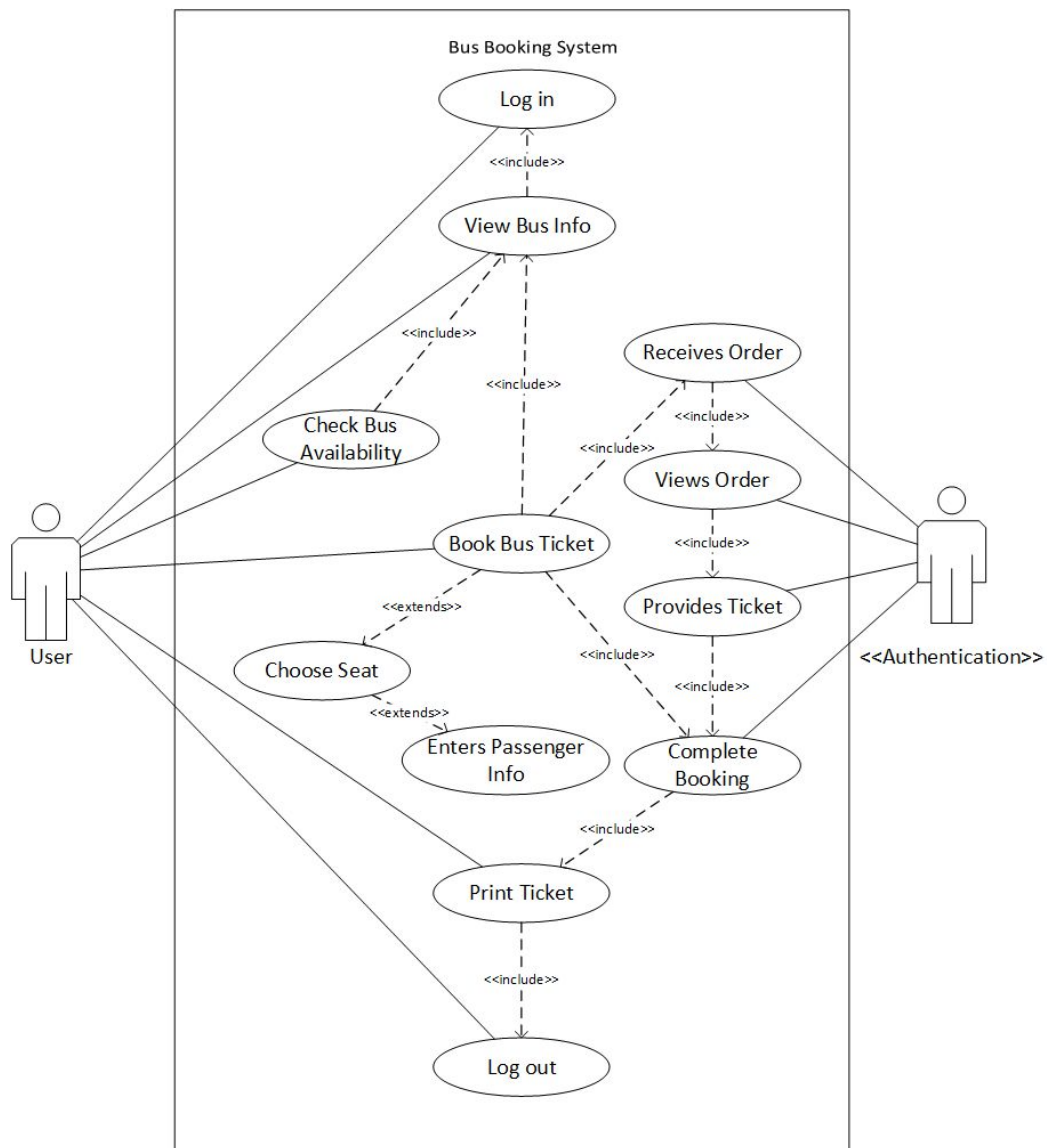
Physical View



Client-Server Architecture

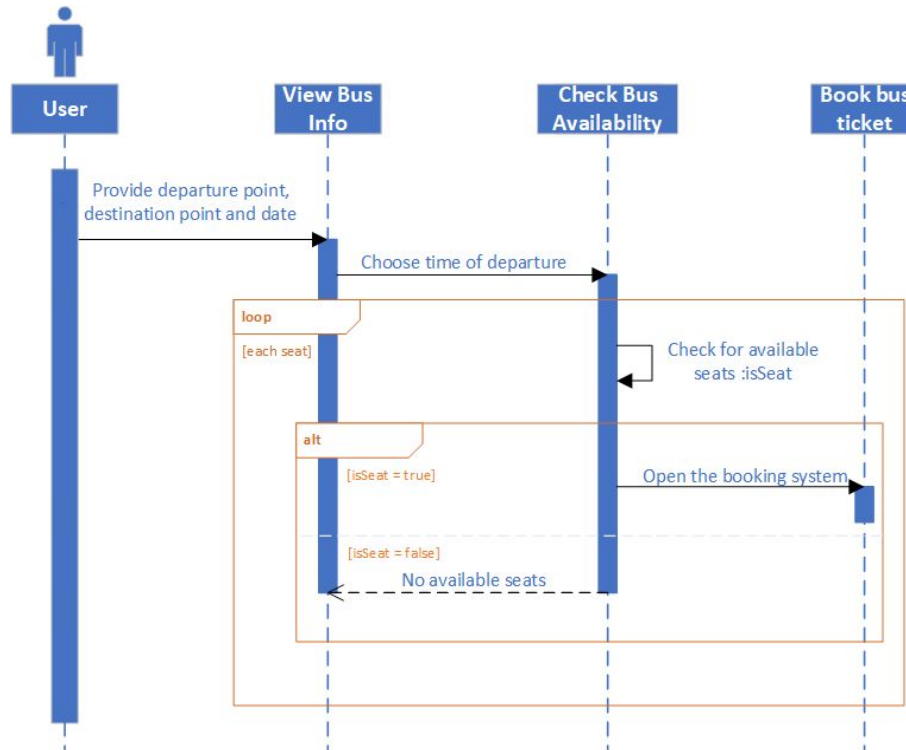


Use Case Diagram:

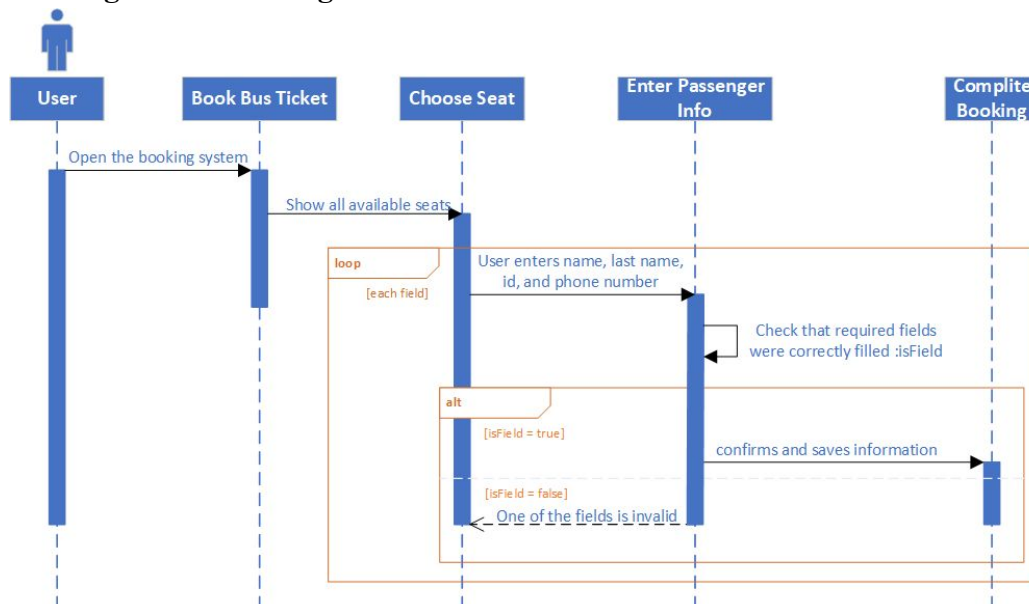


Behavior modeling:

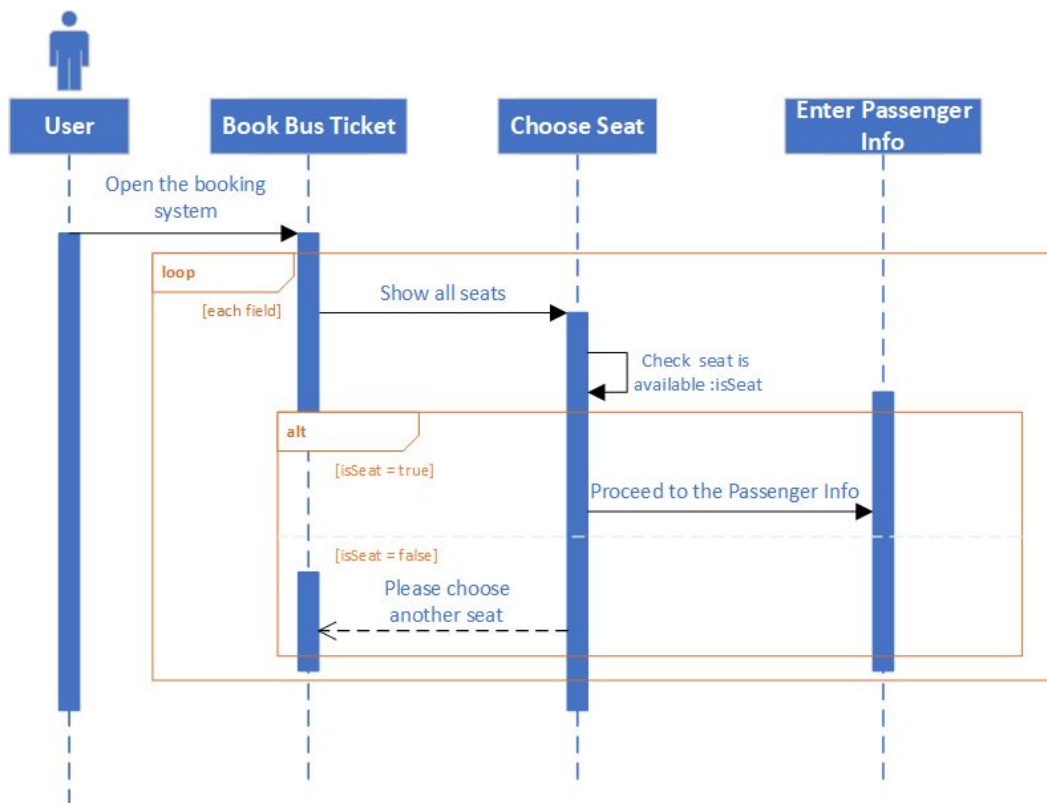
Sequence diagram of Check Bus Availability



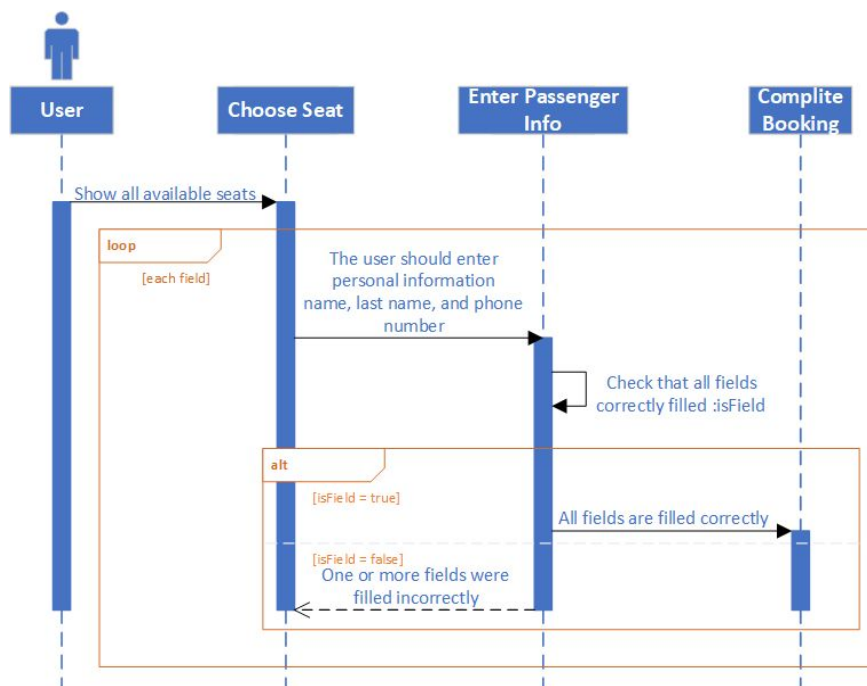
Sequence diagram of Booking Tickets



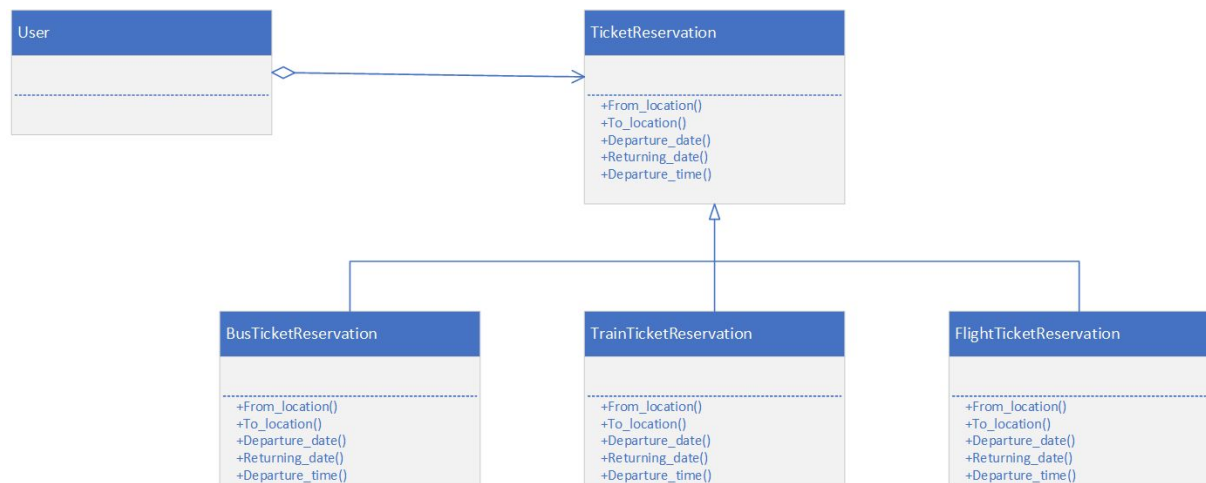
Sequence diagram of Choose Seat



Sequence diagram of Entering Passenger Information



Strategy Pattern



Implementation

Implement the Database Design (Tables, Backend):

For the backend, we used JavaScript to create an API that communicates with the MongoDB database we created. As it stands, the API can connect to 10 endpoints dealing with getting, posting, patching, and deleting a user and trip data. The backend currently allows for the implementation for all the use cases. Node.js was used as the run time environment to execute the developed JavaScript code.

Implement the Class Diagram Design (Frontend and Logic):

For the frontend of the project, I have chosen to use IntelliJ. Because it is an actual IDE, I have capabilities that I wouldn't with a common text editor. I have also decided to code this project using HTML, CSS, and JavaScript because all browsers support it, there is a multitude of web development tools, it's search engine friendly, and it's not overwhelmingly complicated. HTML also has the bootstrap library which helps create a more aesthetically pleasing website.

Testing

Creating Testing environment:

Installed javascript testing frameworks jasmine and karma through npm install. The tests cases were written in JavaScript.

Implementing the Test case:

```
describe('signinController', function() {
  beforeEach(module('myAngApp'));

  var $controller, loggedInOnService;

  beforeEach(inject(function(_$controller_, _loggedInOnService_) {
    $controller = _$controller_;
    loggedInOnService = _loggedInOnService_;
  }));

  describe('$scope.signin', function() {

    var $scope, controller, $cookies;

    beforeEach(function() {
      $scope = {};
      $cookies = {};
      controller = $controller('signinController', { $scope: $scope, $cookies: $cookies});
    });

    it('does not allow the the user to sign in', function() {
      $scope.userEmail = 'notrightemailformat';
      $scope.userPassword = 'shouldnotmatter';
      $scope.signin();
      expect($cookies['uName']).not.toBeDefined();
    });
    it('allows the the user to sign in', function() {
      $scope.userEmail = 'salhassan1@student.gsu.edu';
      $scope.userPassword = 'pass';
      $scope.signin();
      expect(loggedInOnService.signedIn).toBeDefined();
    });
  });
});
```

```

describe('passwordController', function() {
  beforeEach(module('myAngApp'));

  var $controller;

  beforeEach(inject(function(_$controller_) {
    $controller = _$controller_;
  }));

  describe('$scope.chgpass', function() {

    var $scope, controller;

    beforeEach(function() {
      $scope = {};
      controller = $controller('passwordController', { $scope: $scope });
    });

    it('ensure correct current user password is entered', function() {
      $scope.userPassword = 'shouldnotmatter';
      expect($scope.userPassword).toBe('shouldnotmatter');
    });
    it('ensure the current user password is changed', function() {
      $scope.newpass = 'pass';
      expect($scope.newpass).toBe('pass');
    });
  });
});

```

Above are four tests cases relating to sign-in and change-password.

```

describe('mainController', function() {
  beforeEach(module('myAngApp'));

  var $controller, loggedOnService;

  beforeEach(inject(function(_$controller_, _loggedOnService_) {
    $controller = _$controller_;
    loggedOnService = _loggedOnService_;
  }));

  describe('sign out function', function() {

    var $scope, controller;

    beforeEach(function() {
      $scope = {};

      controller = $controller('mainController', { $scope: $scope });
    });

    it('allows the user to sign out', function() {

      loggedOnService.addUser("Saleh")
      $scope.signOut();
      expect(loggedOnService.getUser()).toBe('Sign In')

    });

    it('does not allow the user to sign out', function() {

      loggedOnService.addUser("Saleh")
      expect(loggedOnService.getUser()).not.toBe('Sign In')

    });

  });
});

```



```

describe('mainController', function() {
  beforeEach(module('myAngApp'));

  var $controller, $location;

  beforeEach(inject(function(_$controller_, _$location_) {
    $controller = _$controller_;
    $location = _$location_
  }));

  describe('view bus info', function() {

    var $scope, controller;

    beforeEach(function() {
      $scope = {};
      controller = $controller('mainController', { $scope: $scope });

      spyOn($location, 'path').and.callThrough();
    });

    it('allows the user to navigate to and view bus info page', function() {

      $location.path('/bus');
      expect($location.path()).toBe('/bus')

    });

    it('does not allow the user to navigate to and view bus info page', function() {

      $location.path('/');
      expect($location.path()).not.toBe('/bus')

    });

  });
});

```

```

describe('enterInfoController', function() {
  beforeEach(module('myAngApp'));

  var $controller, loggedOnService;

  beforeEach(inject(function(_$controller_, _loggedOnService_) {
    $controller = _$controller_;
    loggedOnService = _loggedOnService_;
  }));

  describe('personal info page', function() {

    var $scope, controller;

    beforeEach(function() {
      $scope = {};

      controller = $controller('enterInfoController', { $scope: $scope });
    });

    it('allows the user to enter personal info', function() {

      $scope.userTicketName = "Vicky"
      $scope.openTicket();
      expect($scope.userTicketName).toBe('Vicky')

    });

    it('does not allow the user to open ticket after entering personal info', function() {

      $scope.userTicketName = "Test"
      expect($scope.check('form')).not.toBeDefined()

    });

  });
});

```



```

it('should receive order', inject(function($http, $httpBackend) {

    var $scope = {};
    var order = {
        trip_id: '1111'
    }

    $http.post('http://localhost/orders', order)
        .then(function(data, status, headers, config) {
            $scope.ticket = data.data
        })
        .catch(function(data, status, headers, config) {
            $scope.valid = false;
        });

    $httpBackend
        .when('POST', 'http://localhost/orders', order)
        .respond(201, { trip_id: '1111' });

    // $httpBackend.flush();

    expect($httpBackend.flush).not.toThrow();

}));

it('should not receive order', inject(function($http, $httpBackend) {

    var $scope = {};
    var order = {
        trip_id: '1111'
    }

    $http.post('http://localhost/orders', order)
        .then(function(data, status, headers, config) {
            $scope.ticket = data.data
        })
        .catch(function(data, status, headers, config) {
            $scope.valid = false;
        });

    $httpBackend
        .when('POST', 'http://localhost/orders', order)
        .respond(201, { trip_id: '1111' });

    $httpBackend.flush();

    expect($httpBackend.flush).toThrow();

}));

```

```

it('should view order', inject(function($http, $httpBackend) {

    var $scope = {};
    var order = {
        trip_id: '1111'
    }

    $http.post('http://localhost/orders', order)
        .then(function(data, status, headers, config) {
            $scope.ticket = data.data
        })
        .catch(function(data, status, headers, config) {
            $scope.valid = false;
        });

    $httpBackend
        .when('POST', 'http://localhost/orders', order)
        .respond(201, { trip_id: '1111' });

    $httpBackend.flush();

    expect($scope.ticket).toEqual({ trip_id: '1111' });

})));

it('should not view order', inject(function($http, $httpBackend) {

    var $scope = {};
    var order = {
        trip_id: '1111'
    }

    $http.post('http://localhost/orders', order)
        .then(function(data, status, headers, config) {
            $scope.ticket = data.data
        })
        .catch(function(data, status, headers, config) {
            $scope.valid = false;
        });

    $httpBackend
        .when('POST', 'http://localhost/orders', order)
        .respond(201, { trip_id: '1111' });

    // $httpBackend.flush();

    expect($scope.ticket).not.toBeDefined();

})));

```

```

it('should provide reserved ticket', inject(function($http, $httpBackend) {

    var $scope = {};

    $http.get('http://localhost/orders')
        .then(function(data, status, headers, config) {
            $scope.valid = true;
            $scope.response = data.data;
        })
        .catch(function(data, status, headers, config) {
            $scope.valid = false;
        });

    $httpBackend
        .when('GET', 'http://localhost/orders')
        .respond(200, { trip_id: '12345' });

    $httpBackend.flush();

    expect($scope.valid).toBe(true);
    expect($scope.response).toEqual({ trip_id: '12345' });

}));

it('should not provide reserved ticket', inject(function($http, $httpBackend) {

    var $scope = {};

    $http.get('http://localhost/orders')
        .then(function(data, status, headers, config) {
            $scope.valid = true;
            $scope.response = data.data;
        })
        .catch(function(data, status, headers, config) {
            $scope.valid = false;
        });

    $httpBackend
        .when('GET', 'http://localhost/orders')
        .respond(200, { trip_id: '12345' });

    $httpBackend.flush();

    expect($scope.valid).toBe(true);
    expect($scope.response).toEqual({ trip_id: '12345' });

}));

```

```

describe('busController', function() {
  beforeEach(module('myAngApp'));

  var $controller, $location;

  beforeEach(inject(function(_$controller_, _$location_) {
    $controller = _$controller_;
    $location = _$location_
  }));

  describe('view bus info', function() {

    var $scope, controller;

    beforeEach(function() {
      $scope = {};
      controller = $controller('busController', { $scope: $scope });

      spyOn($location, 'path').and.callThrough();
    });

    it('allows the user to navigate to and view bus info page', function() {

      trip = {
        _id: '1234'
      }
      $scope.bookBusTrip(trip);
      expect($location.path()).toBe('/sign-in')

    });

    it('does not allow the user to navigate to and view bus info page', function() {

      expect($location.path()).not.toBe('/sign-in')

    });

  });
});

```

```

describe('mainController', function() {
  beforeEach(module('myAngApp'));

  var $controller, $location;

  beforeEach(inject(function(_$controller_, _$location_) {
    $controller = _$controller_;
    $location = _$location_
  }));

  describe('view all bus availability', function() {

    var $scope, controller;

    beforeEach(function() {
      $scope = {};
      controller = $controller('mainController', { $scope: $scope });

      spyOn($location, 'path').and.callThrough();
    });

    it('allows the user to navigate to and view all bus availabilities', function() {

      $location.path('/bus');
      expect($location.path()).toBe('/bus')

    });

    it('does not allow the user to navigate to and view all bus availabilities', function() {

      $location.path('/');
      expect($location.path()).not.toBe('/bus')

    });
  });
});

```



```

describe('enterInfoController', function() {
  beforeEach(module('myAngApp'));

  var $controller, loggedOnService;

  beforeEach(inject(function(_$controller_, _loggedOnService_) {
    $controller = _$controller_;
    loggedOnService = _loggedOnService_;
  }));

  describe('choose available seats', function() {

    var $scope, controller;

    beforeEach(function() {
      $scope = {};

      controller = $controller('enterInfoController', { $scope: $scope });
    });

    it('allows the user to choose available seats', function() {

      expect($scope.tripSeats[0]).toBe('Row 8 Seat 8C')

    });
    it('does not allow the user to open ticket after entering personal info', function() {

      expect($scope.tripSeats[0]).not.toBe('Not An Available Seat')

    });
  });
});

```

```

describe('modalController', function() {
  beforeEach(module('myAngApp'));

  var $controller, $location;

  beforeEach(inject(function(_$controller_, _$location_) {
    $controller = _$controller_;
    $location = _$location_;
  }));

  describe('check conformation page', function() {

    var $scope, controller, modalInstance;

    beforeEach(function() {

      modalInstance = {
        close: jasmine.createSpy('uibModalInstance.close'),
        dismiss: jasmine.createSpy('uibModalInstance.dismiss'),
        result: {
          then: jasmine.createSpy('uibModalInstance.result.then')
        }
      }

      // modalInstance = jasmine.createSpyObj('modalInstance', ['close', 'dismiss', 'result.then']);

      $scope = {};

      controller = $controller('modalController', { $scope: $scope, $uibModalInstance: modalInstance });

      spyOn($location, 'path').and.callThrough();

    });

    it('ensures the user sees the conformation page', function() {

      expect($scope.ticketNumber).toBeDefined()

    });

    it('ensures the user does not see the conformation page', function() {

      expect($location.path()).not.toBe('/enter-info')

    });

  });
});

```

```

describe('modalController', function() {
  beforeEach(module('myAngApp'));

  var $controller, $location;

  beforeEach(inject(function(_$controller_, _$location_) {
    $controller = _$controller_;
    $location = _$location_;
  }));

  describe('print function', function() {

    var $scope, controller, modalInstance;

    beforeEach(function() {

      modalInstance = {
        close: jasmine.createSpy('uibModalInstance.close'),
        dismiss: jasmine.createSpy('uibModalInstance.dismiss'),
        result: {
          then: jasmine.createSpy('uibModalInstance.result.then')
        }
      }

      // modalInstance = jasmine.createSpyObj('modalInstance', ['close', 'dismiss', 'result.then']);

      $scope = {};

      controller = $controller('modalController', { $scope: $scope, $uibModalInstance: modalInstance });

      spyOn($location, 'path').and.callThrough();

    });

    it('should close the modal when the user prints', function() {

      $scope.sendTicket();
      expect(modalInstance.close).toHaveBeenCalled();

    });

    it('should not close the modal if the user has not printed or been sent email ', function() {

      expect(modalInstance.close).not.toHaveBeenCalled();

    });

  });
});

```

Test Case Scrip:

<u>Test ID</u>	1.1
<u>Purpose of Test</u>	Make sure that the user can log in on the website with valid information.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop.

	No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	In the log in page user should do: User clicks with the mouse on the field that says Email and enters the valid Email address into the field. User clicks with the mouse on the field that says Password and enters the valid Password into the field. User click button log in.
<u>Test Inputs</u>	Valid user email and password into assigned fields.
<u>Expected Result</u>	Login Successful. User returning to the main page where he/she can choose for which transport will book a ticket.
<u>Likely Problem/Bugs Revealed</u>	None

<u>Test ID</u>	1.2
<u>Purpose of Test</u>	Make sure that the user can't log in on the website with invalid information.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	In the log in page user should do: User clicks with the mouse on the field that says Email and enters the valid Email address into the field. User clicks with the mouse on the field that says Password and enters the valid Password into the field. User click button log in.
<u>Test Inputs</u>	Wrong user email and password into assigned fields.

<u>Expected Result</u>	Login Unsuccessful. The user stays at the same page and sees notification that Email and/or Password is incorrect.
<u>Likely Problem/Bugs Revealed</u>	None

<u>Test ID</u>	2.1
<u>Purpose of Test</u>	Make sure that the user can change his/her password when is it needed.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	In the user, profile page tester should : Clicks with the mouse on the field that says Enter an old password and enters the valid old password there. Clicks with the mouse on the field that says Enter an old password and enters the new password there. Click button save changes that locates at the center-bottom page.
<u>Test Inputs</u>	User enters old password and new password into assigned fields.
<u>Expected Result</u>	Change of password is successful. The user redirects to user profile page.
<u>Likely Problem/Bugs Revealed</u>	None

<u>Test ID</u>	2.2
<u>Purpose of Test</u>	Make sure that the user can change his/her password when is it needed. Also, the system can recognize when the wrong old password was entered.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	In the user, profile page tester should :

	<p>Clicks with the mouse on the field that says Enter an old password and enters the invalid old password there.</p> <p>Clicks with the mouse on the field that says Enter an old password and enters the new password there.</p> <p>Click button save changes that locates at the center-bottom page.</p>
<u>Test Inputs</u>	<p>User enters a wrong old password into assigned field.</p> <p>User enters new password into assigned field.</p>
<u>Expected Result</u>	Change of password is unsuccessful. The user stays on the same page and sees a notification that the old password is not matching with entered one.
<u>Likely Problem/Bugs Revealed</u>	None

<u>Test ID</u>	3.1
<u>Purpose of Test</u>	Make sure that users can view the bus information after search.
<u>Test Environment</u>	<p>Ures hardware : Asus F556U laptop.</p> <p>No other application was running during the testing on a laptop.</p> <p>Operation system: Windows 10</p>
<u>Test Steps</u>	Click button bus that locates at the center of the main page.
<u>Test Inputs</u>	User must be already logged in.
<u>Expected Result</u>	A user sees a dashboard with a list of different bus trips and their information.
<u>Likely Problem/Bugs Revealed</u>	A location.path not recognized as a function.

<u>Test ID</u>	3.2
<u>Purpose of Test</u>	Make sure that users can view the bus information after search.

<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	Click button bus that locates at the center of the main page. Refresh the page and try to click at the button again.
<u>Test Inputs</u>	User must be already logged in
<u>Expected Result</u>	Access of the page is unsuccessful.
<u>Likely Problem/Bugs Revealed</u>	None

<u>Test ID</u>	4.1
<u>Purpose of Test</u>	Make sure that the user can check the availability of the bus.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	Click button bus that locates at the center of the main page.
<u>Test Inputs</u>	User clicks at the bus button.
<u>Expected Result</u>	A user moves to the next page where is display all available busses with their information.
<u>Likely Problem/Bugs Revealed</u>	None

<u>Test ID</u>	4.2
<u>Purpose of Test</u>	Make sure that the user can check the availability of the bus.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	Click button bus that locates at the center of the main page.

	Refresh page and try again by clicking the same button.
<u>Test Inputs</u>	User clicks at the bus button.
<u>Expected Result</u>	User stays at the same page.
<u>Likely Problem/Bugs Revealed</u>	None

<u>Test ID</u>	5.1
<u>Purpose of Test</u>	Make sure that the user can book the ticket.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	Click at the chosen icon of the bus or on its name.
<u>Test Inputs</u>	User must be already logged in.
<u>Expected Result</u>	User will move to the next page of the website where can continue to reserve the ticket.
<u>Likely Problem/Bugs Revealed</u>	None

<u>Test ID</u>	5.2
<u>Purpose of Test</u>	Make sure that user can book a ticket.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	Click at the chosen icon of the bus or on its name.
<u>Test Inputs</u>	User must be already logged in.
<u>Expected Result</u>	User stays on the same page.
<u>Likely Problem/Bugs Revealed</u>	None

<u>Test ID</u>	6.1
-----------------------	-----

<u>Purpose of Test</u>	Authentication received user order.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	Click button Submit at the center-bottom of a page with choice seat and filling passenger information.
<u>Test Inputs</u>	JWT token
<u>Expected Result</u>	Authentication receives order and process it.
<u>Likely Problem/Bugs Revealed</u>	None

<u>Test ID</u>	6.2
<u>Purpose of Test</u>	Authentication received user order.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	Click button Submit at the center-bottom of a page with choice seat and filling passenger information. Try again by entering information once again and then click submit button again.
<u>Test Inputs</u>	JWT token
<u>Expected Result</u>	Authentication failed receiving the order.
<u>Likely Problem/Bugs Revealed</u>	None

<u>Test ID</u>	7.1
<u>Purpose of Test</u>	Make sure that Authentication can check received order.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10

<u>Test Steps</u>	Click button Submit at the center-bottom of a page with choice seat and filling passenger information.
<u>Test Inputs</u>	JWT token
<u>Expected Result</u>	Authentication views the receives order and process it.
<u>Likely Problem/Bugs Revealed</u>	None

<u>Test ID</u>	7.2
<u>Purpose of Test</u>	Make sure that Authentication can check received order.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	Click button Submit at the center-bottom of a page with choice seat and filling passenger information. Try again by entering information once again and then click submit button again.
<u>Test Inputs</u>	JWT token
<u>Expected Result</u>	Authentication can't views the received order and can't process it.
<u>Likely Problem/Bugs Revealed</u>	None

<u>Test ID</u>	8.1
<u>Purpose of Test</u>	Make sure that the authentication provides user a reserved ticket.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	Click button Submit at the center-bottom of a page with choice seat and filling passenger information.
<u>Test Inputs</u>	JWT token

<u>Expected Result</u>	Authentication process order successful. User will see the confirmation page.
<u>Likely Problem/Bugs Revealed</u>	None

<u>Test ID</u>	8.2
<u>Purpose of Test</u>	Make sure that the authentication provides user with a right reserved ticket and correct information on it.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	Click button Submit at the center-bottom of a page with choice seat and filling passenger information. Try again by entering information once again and then click submit button again.
<u>Test Inputs</u>	JWT token
<u>Expected Result</u>	Authentication process order unsuccessful; user will stay at the same page.
<u>Likely Problem/Bugs Revealed</u>	None

<u>Test ID</u>	9.1
<u>Purpose of Test</u>	Make sure that the user can choose the seat.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	Click at the Drop-down list. Choose a seat. Click at button submit that at the center-bottom page.
<u>Test Inputs</u>	A chosen seat from the Drop-down list. Click on button submit.
<u>Expected Result</u>	User moves to the next page.

<u>Likely Problem/Bugs Revealed</u>	None
--	------

<u>Test ID</u>	9.2
<u>Purpose of Test</u>	Make sure that the user can choose the seat.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	Click at button submit that at the center-bottom page.
<u>Test Inputs</u>	Click on button submit.
<u>Expected Result</u>	User stays at the same page.
<u>Likely Problem/Bugs Revealed</u>	None

<u>Test ID</u>	10.1
<u>Purpose of Test</u>	Make sure that the user can enter personal information to reserve a ticket.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	Clicks with the mouse on the field that says Name and enters the valid name there. Clicks with the mouse on the field that says Phone number and enters phone number. Clicks with the mouse on the field that says Email and enters email information.. Click button submit that locates at the center-bottom page.
<u>Test Inputs</u>	User enters name into assigned field. User enters phone number into assigned field. User enters email into assigned field.
<u>Expected Result</u>	A user sees a confirmation page that ticket was booked.
<u>Likely Problem/Bugs Revealed</u>	A phone number field accepting letters.

<u>Test ID</u>	10.2
<u>Purpose of Test</u>	Make sure that the user can't reserve a ticket with provided incorrect information.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	Clicks with the mouse on the field that says Name and enters the numbers. Clicks with the mouse on the field that says Phone number and enters letters.. Clicks with the mouse on the field that says Email and enters email information.. Click button submit that locates at the center-bottom page.
<u>Test Inputs</u>	User enters number instead of name into assigned field. User enters letters instead of phone number into assigned field. User enters email into assigned field.
<u>Expected Result</u>	Booking of the ticket unsuccessful user stays at the same page.
<u>Likely Problem/Bugs Revealed</u>	None

<u>Test ID</u>	11.1
<u>Purpose of Test</u>	Make sure that the user can see a confirmation page with correct information on it.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	Click at button submit that at the center-bottom page.
<u>Test Inputs</u>	User must be already complete reservation tickets.

<u>Expected Result</u>	Confirmation page with correct information on it.
<u>Likely Problem/Bugs Revealed</u>	Ticket id not showing in confirmation page.

<u>Test ID</u>	11.2
<u>Purpose of Test</u>	Make sure that the user can see a confirmation page with correct information on it.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	Click at button submit that at the center-bottom page. Return to the previous page and try again by clicking the button.
<u>Test Inputs</u>	User must be already complete reservation tickets.
<u>Expected Result</u>	Confirmation page with incorrect information on it.
<u>Likely Problem/Bugs Revealed</u>	None

<u>Test ID</u>	12.1
<u>Purpose of Test</u>	Make sure that user is able to print ticket
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	Click at button send the ticket that at the center-bottom page. Click at button print the ticket that at the center-bottom page.
<u>Test Inputs</u>	Click on button print the ticket. Click on button send the ticket.
<u>Expected Result</u>	User successfully printed/dended the ticket.
<u>Likely Problem/Bugs Revealed</u>	User can't print or send the ticket.

<u>Test ID</u>	12.2
<u>Purpose of Test</u>	Make sure that user is able to print ticket
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	Click at button print the ticket that at the center-bottom page. Click at button send the ticket that at the center-bottom page. Refresh page. Try again by clicking the same button.
<u>Test Inputs</u>	Click on button print the ticket. Click on button send the ticket.
<u>Expected Result</u>	User unsuccessfully printed/sended the ticket.
<u>Likely Problem/Bugs Revealed</u>	None

<u>Test ID</u>	13.1
<u>Purpose of Test</u>	Make sure that the user can log out from the website after finishing his booking tickets and his personal information can't be accessed after log out.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	Click the logout button at the top right-hand side of the page.
<u>Test Inputs</u>	User must be already logged in.
<u>Expected Result</u>	Logout successful.
<u>Likely Problem/Bugs Revealed</u>	None

<u>Test ID</u>	13.2
-----------------------	------

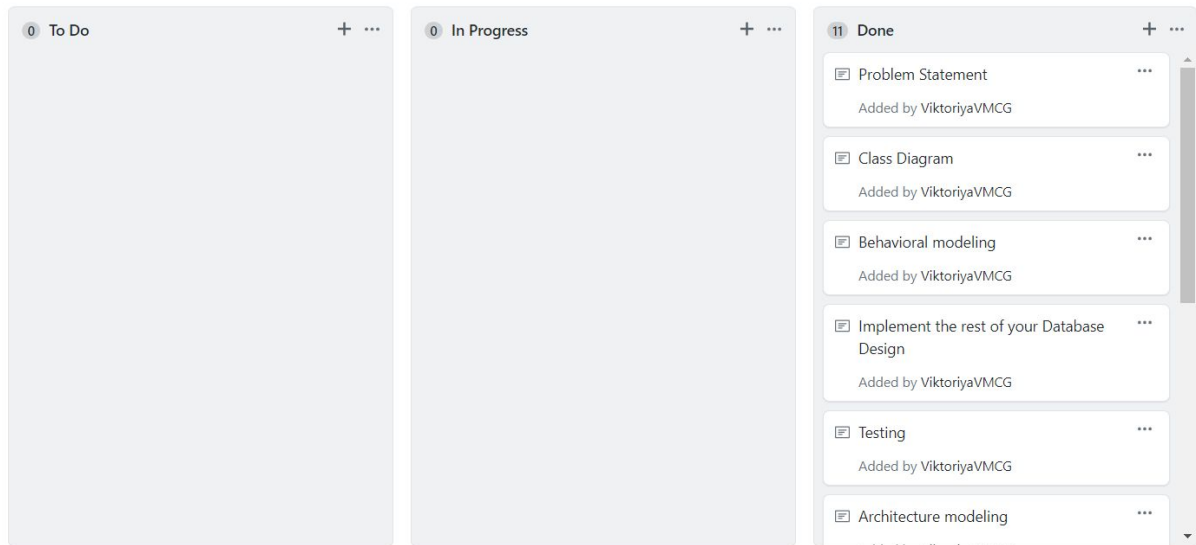
<u>Purpose of Test</u>	Make sure that the user can log out from the website after finishing his booking tickets and his personal information can't be accessed after log out.
<u>Test Environment</u>	Ures hardware : Asus F556U laptop. No other application was running during the testing on a laptop. Operation system: Windows 10
<u>Test Steps</u>	Click the logout button at the top right-hand side of the page. Go back to the main page and try clicking the button again.
<u>Test Inputs</u>	User must be already logged in.
<u>Expected Result</u>	Logout unsuccessful.
<u>Likely Problem/Bugs Revealed</u>	None

Test Documentation:

Bug	The test uncovered the bug	Description of the bug	An action was taken to fix the bug
1	3.1	A location.path not recognized as a function.	An injected \$location to the main program scope.
2	10.1	The phone number field was able to accept any characters including letters.	Was used a try-catch exception to make sure that only numbers are passing.
3	11.1	Ticket id not showing in confirmation page.	Binded the ticket id to the website scope.
4	12.1	The user attempts to print ticket but it not printing out.	We tried our best, but couldn't fix it.
5	12.1	The user attempts to send a ticket to the user email but not receiving it.	We tried our best, but couldn't fix it.

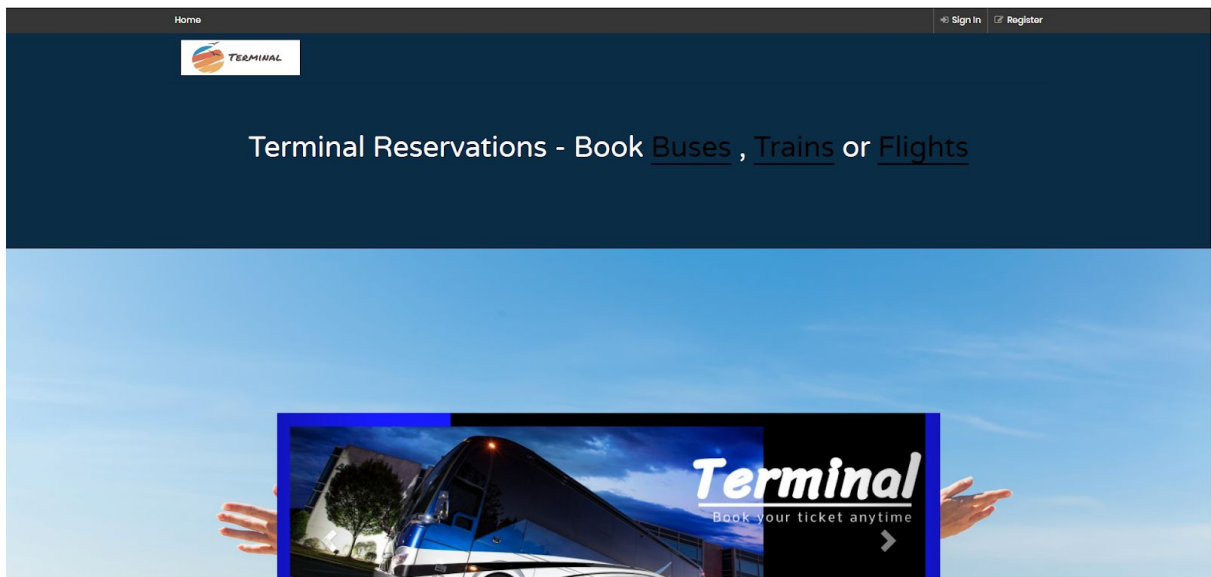
Appendix

<https://github.com/5aleh/book-on-time/projects/6>

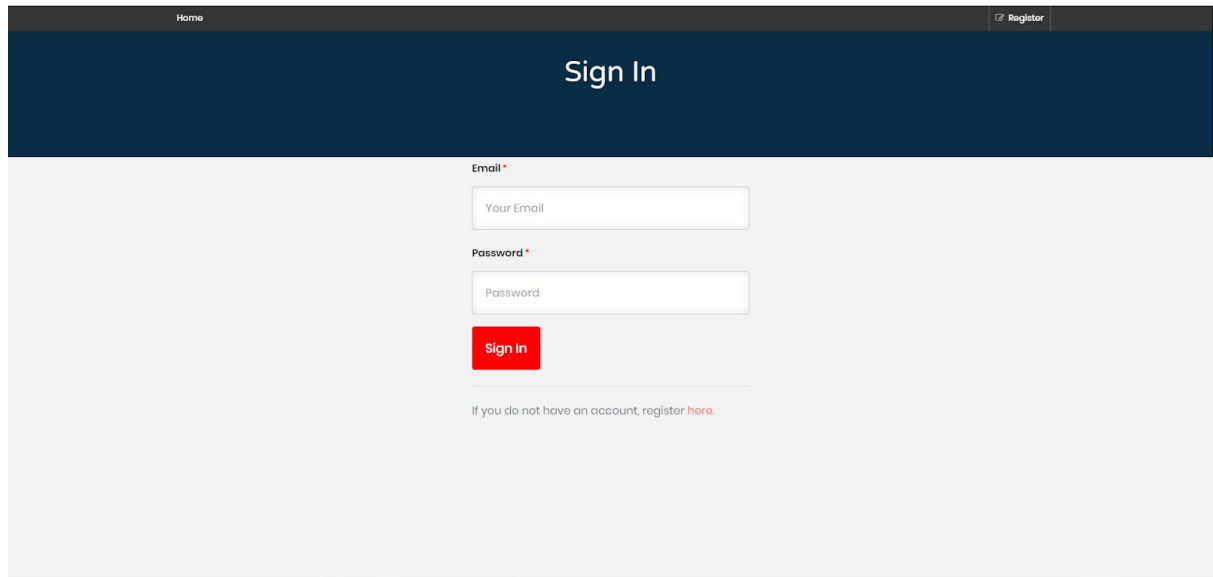


Implemented Frontend

Home Page



Sign in Page



The Sign In page features a dark blue header with a 'Home' link on the left and a 'Register' link on the right. The main title 'Sign In' is centered in the header. Below the header, the page has a light gray background. The form is centered and includes an 'Email' field with a red asterisk, a 'Password' field with a red asterisk, and a red 'Sign In' button. A link to the registration page is provided at the bottom.

Home Register

Sign In

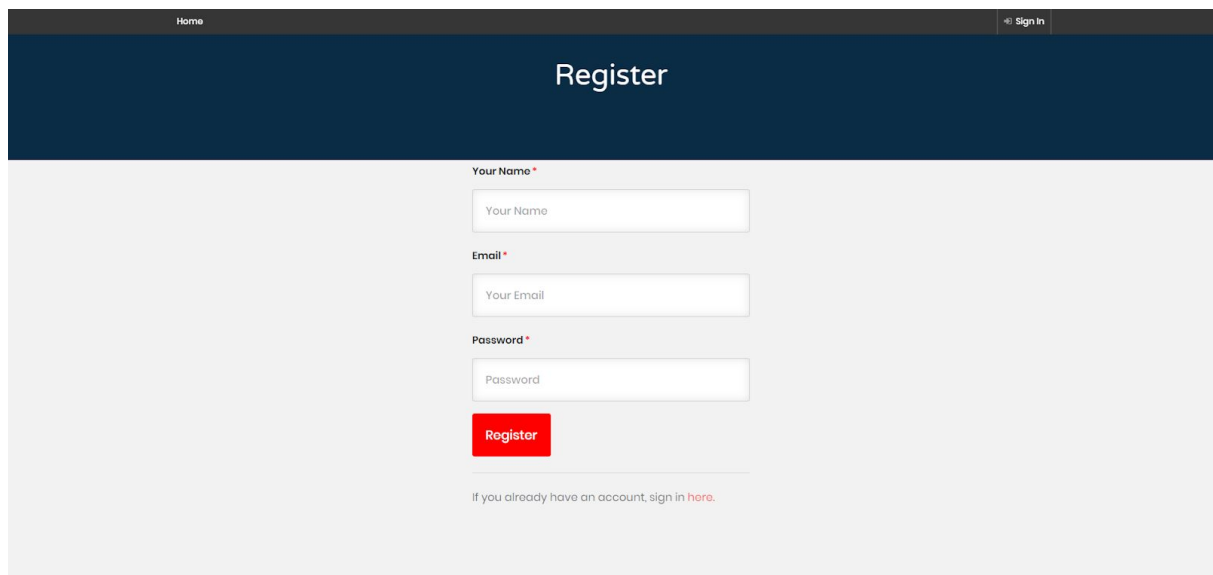
Email *

Password *

Sign In

If you do not have an account, register [here](#).

Register page



The Register page features a dark blue header with a 'Home' link on the left and a 'Sign In' link on the right. The main title 'Register' is centered in the header. Below the header, the page has a light gray background. The form is centered and includes a 'Your Name' field with a red asterisk, an 'Email' field with a red asterisk, a 'Password' field with a red asterisk, and a red 'Register' button. A link to the sign-in page is provided at the bottom.

Home Sign In

Register

Your Name *

Email *

Password *


Register

If you already have an account, sign in [here](#).


Bus page

[Home](#)

[Sign In](#)[Register](#)



Book Buses



Bus

From: Atlanta

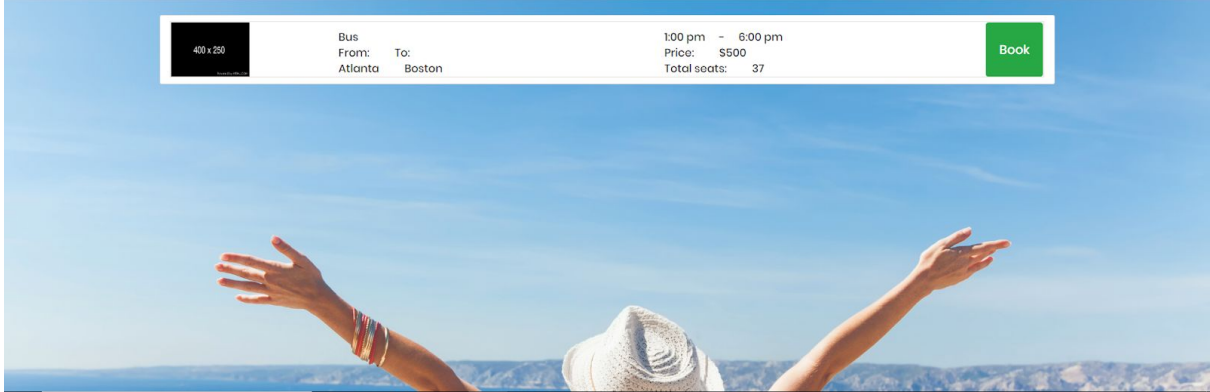
To: Boston

1:00 pm - 6:00 pm

Price: \$500

Total seats: 37


Book




Train page

[Home](#)

[Sign In](#)[Register](#)



Book Trains



Train

From: Chicago

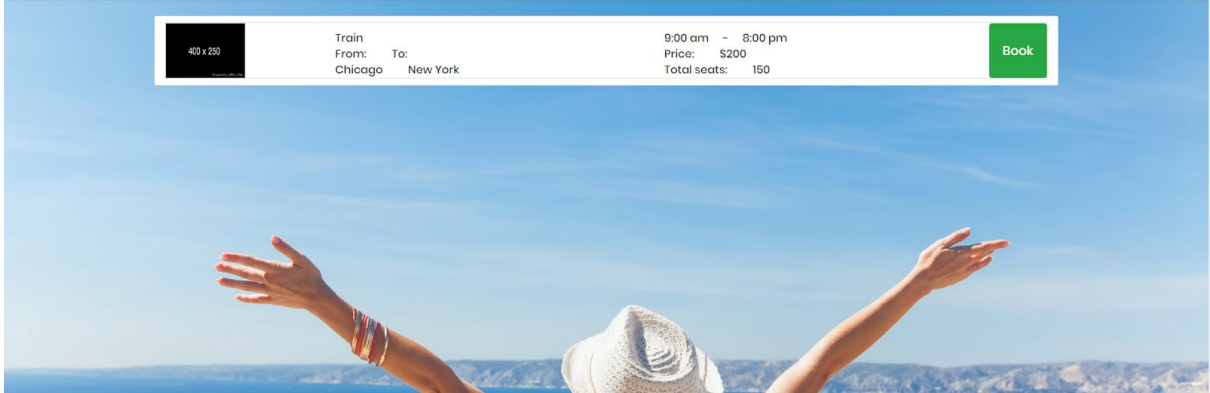
To: New York

9:00 am - 8:00 pm

Price: \$200


Total seats: 150

Book





Flight page

[Home](#)[Sign In](#)[Register](#)



Book Flights



Flight

From: To:


San Francisco Paris

7:15 pm - 7:45 am

Price: \$1200

Total seats: 216

Book



Flight

From: To:

Atlanta Japan

5:00 pm - 5:00 am


Price: \$1500

Total seats: 200

Book

Profile page

[Home](#)[Practice Test](#)[Sign Out](#)



My Profile

[Change Password](#)

Personal Information

Your Name *

Practice Test

Contact

Phone

Your Phone

Email

Change password page

The screenshot shows the 'Change Password' page. At the top, there is a dark blue header with the 'TERMINAL' logo on the left and navigation links 'Home', 'Practice Test', and 'Sign Out' on the right. The main title 'Change Password' is centered in the header. Below the header, on the left, is a 'My Profile' link. The main content area contains three password input fields: 'Current Password', 'New Password', and 'Repeat Password'. Each field has a red asterisk indicating it is required. A red 'Change Password' button is positioned below the 'Repeat Password' field.

Home Practice Test Sign Out

TERMINAL

Change Password

My Profile

Current Password *

Current Password

New Password *

New Password

Repeat Password *

Repeat New Password

Change Password

Choose seat and enter personal information page

The screenshot shows the 'Choose Seat And Enter Personal Information' page. The header is identical to the previous page. The main title 'Choose Seat And Enter Personal Information' is centered. Below the header, the page is divided into two sections: 'Choose your seat' and 'Enter your personal information'. The 'Choose your seat' section has a dropdown menu labeled 'Choose Seat' and a note 'Please choose one of the following available seats.' The 'Enter your personal information' section has a note 'Please complete the following fields.' and three input fields for 'Name:', 'Phone number:', and 'Email:'. A green 'Submit' button is at the bottom center.

Home Practice Test Sign Out Profile

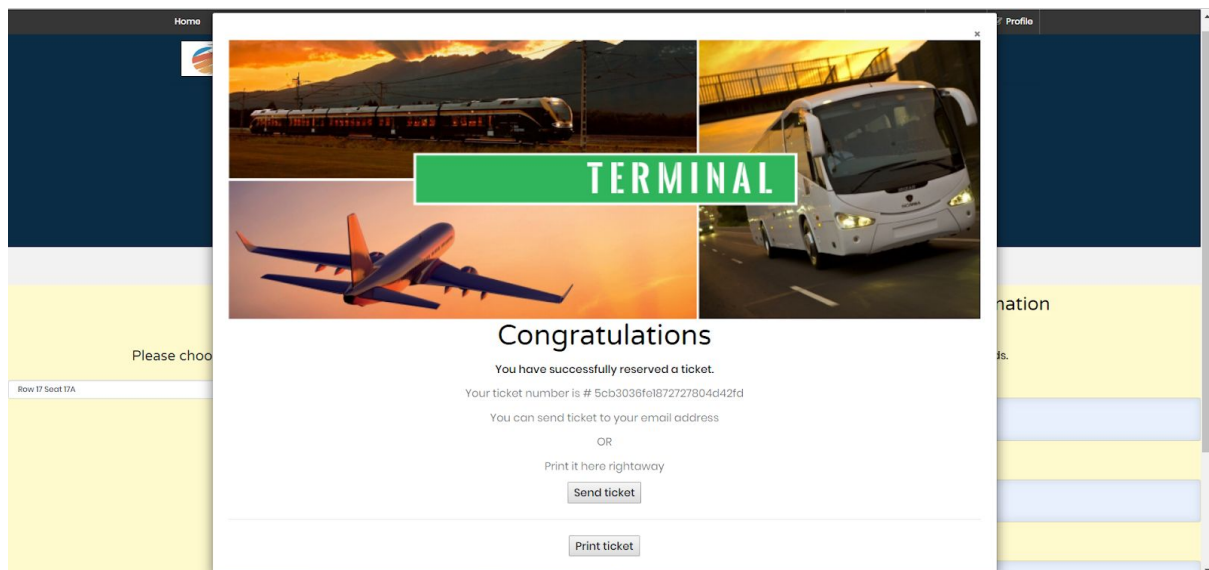
TERMINAL

Choose Seat And Enter Personal Information

Ticket reservation

Choose your seat	Enter your personal information
<p>Please choose one of the following available seats.</p> <p>Choose Seat</p>	<p>Please complete the following fields.</p> <p>Name:</p> <p>Phone number:</p> <p>Email:</p>
<p>Submit</p>	

Conformation modal page



Implemented Backend:

Below is the structure of the Users document model with some user examples

```
_id: ObjectId("5c99816d03695e00e8b5ec0d")
name: "student"
password: "$2b$10$JnGg6kc.w/hMxyNov4bvWu/uaHDs6XOVh2YQXoVDRsrVno9T6Dsre"
email: "student@test.com"
✓ tickets: Array
  ✓ 0: Object
    _id: ObjectId("5cb26c20edba313b88d2c924")
    trip_id: ObjectId("5c76f9177f6d1c4ea425e7b5")
  > 1: Object
  > 2: Object
  __v: 0
```

```
_id: ObjectId("5c997ee803695e00e8b5ec0c")
name: "new test"
password: "$2b$10$moU/fM/SMTyjrKNKdyPziejJVqfuPBXqwE/HtPt17BB0FCqm0H4/S"
email: "test2@test.com"
> tickets: Array
__v: 0
```

```
_id: ObjectId("5c99816d03695e00e8b5ec0d")
name: "student"
password: "$2b$10$JnGg6kc.w/hMxyNov4bvWu/uaHDs6XOVh2YQXoVDRsrVno9T6Dsre"
email: "student@test.com"
✓ tickets: Array
  > 0: Object
  > 1: Object
  > 2: Object
__v: 0
```

```
_id: ObjectId("5c9a7c5777edb129cc8fd6f6")
name: "test"
password: "$2b$10$sSGDoHvLbiPOXeRmN/Jj304TT1IFzrDW.HsaTB.OwJsCFkrm/x3W2"
email: "test@test.com"
> tickets: Array
__v: 0
```

```
_id: ObjectId("5cb3015ee1872727804d42fc")
name: "Practice Test"
password: "$2b$10$StNYeb/sDB9U1W44HL8pV./vZZDAxLCS1vrplAlEz2MNB/RpQ6F62"
email: "practice@test.com"
> tickets: Array
__v: 0
```

Below is the structure of the trips document model

```
_id: ObjectId("5c75c1fe22b7e431643bc2a7")
trip_type: "Bus"
location_to: "Boston"
location_from: "Atlanta"
day: "Monday"
departure_time: "1:00 pm"
arrival_time: "6:00 pm"
total_seats: 37
price: 500
__v: 0
```

```

_id: ObjectId("5c9e4c971c9d440000d3cb67")
trip_type: "Flight"
location_to: "Japan"
location_from: "Atlanta"
date: 2019-05-13T09:00:00.000+00:00
departure_time: "5:00 pm"
arrival_time: "5:00 am"
total_seats: 200
price: 1500
__v: 0

_id: ObjectId("5c9e4d1c1c9d440000d3cb68")
trip_type: "Train"
location_to: "New York"
location_from: "Chicago"
date: 2019-04-30T03:15:00.000+00:00
departure_time: "9:00 am"
arrival_time: "8:00 pm"
total_seats: 150
price: 200
__v: 0

```

Perspective of the database. Two collections: trips and users

1 DATABASES 2 COLLECTIONS

+ Create Database

Q NAMESPACES

▼ terminal-reservation-db

trips

users