TERMINAL

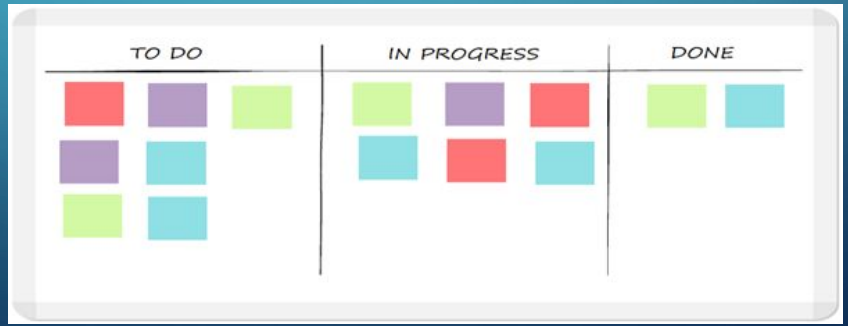
SSJKV
GROUP 2

Kamal Rimal, Sarah Swilley, Saleh Alhassan, Juan Martinez, Viktoriya Rasuli

PROBLEM

Plan/Schedule

- Hard to separate work among team members.
  - Some people did have the required skills.
  - Some team members didn't want to take work and learn new skills.
- Schedule deadline to submit the work of each team member.
  - Some team members always missed the submission deadline work and required to extend.



PROBLEM

Collaboration: GitHub/Slack

- It is hard to schedule an online meeting in Slack.
  - People at work, busy, have the stuff to do on their own.
- Bad team communication.
  - Some people not communicating with other team members.
  - Some people not responding to the team chat.
  - Some team members response to messages after several hours.
- Find a time for a collective meeting.
  - Busy schedule and work.

PROBLEM

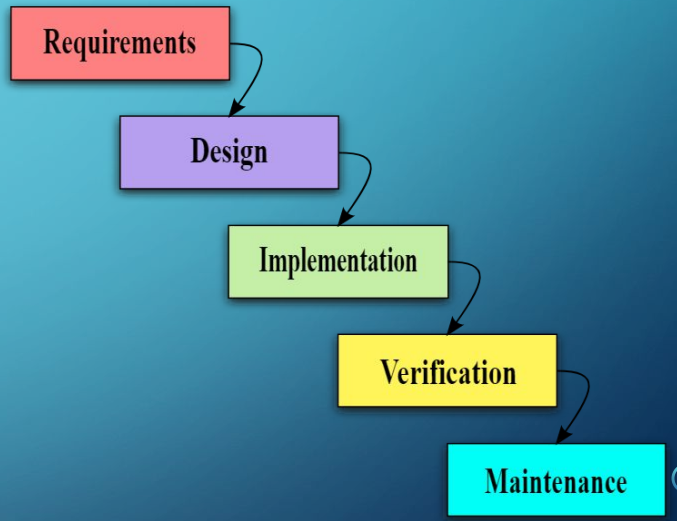
Development Process

- Main problems with the development were figuring out the layout -- the linear path taken through the websites
  - Trying to figure out a good entry point
  - Transition pages
  - Having conflicting ideas about format and necessary webpages
  - Having database and front-end working separately
    - miscommunication about how data is being pulled from the database and how it should look on the front-end side

Development Process Waterfall

Why did we used it?

- Easy to flaw.
- It can be implemented for any size of project.
- Testing is done at every stage.



IMPLEMENTATION

- Languages used
  - HTML, CSS, and Javascript.
  - AngularJS, Bootstrap CSS
- Database used
  - NoSQL > MongoDB
- API
  - Created our own
  - Node.js, Express.js, and Mongoose


```
  _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
```

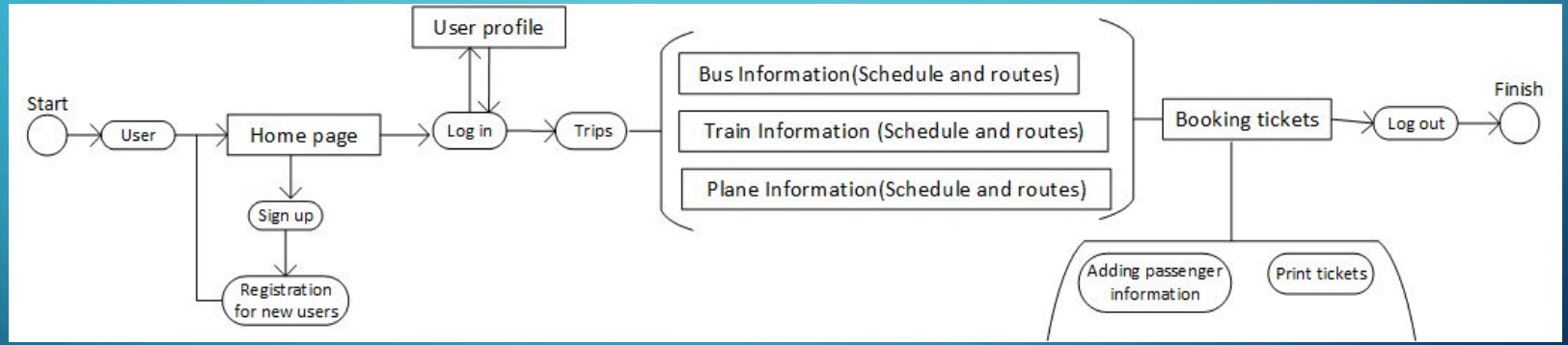
REQUIREMENT

Elicitation

- Visualization - As a group we spent a long time building up to our project by drawing diagrams and deciding how to proceed visually from point to point.
- Collecting data - For the front-end we spent many hours looking at other websites and navigating through them to see how we should proceed with our own website.
- Deciding the appropriate tools to use for this particular project was hard because we all come from different programming language backgrounds (except java), however, building a website solely with java can be cumbersome.

REQUIREMENT

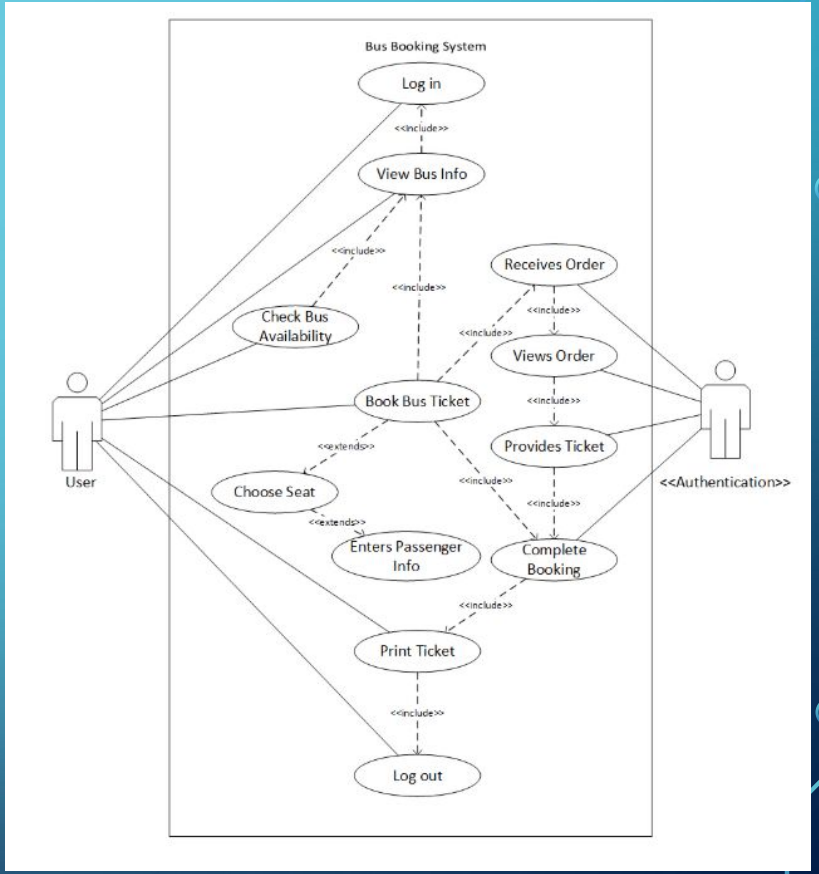
Context diagram/Activities diagram



REQUIREMENT

Use Cases and Use cases diagram

- Actors: User & Authentication
  - User: Log in, View Bus Info, Check Bus Availability, Book Bus Ticket, Choose Seat, Enter Passenger Info, Complete Booking, Print Ticket, Log out.
  - Authentication: Receives Order, Views Order, Provides Ticket, Complete Booking



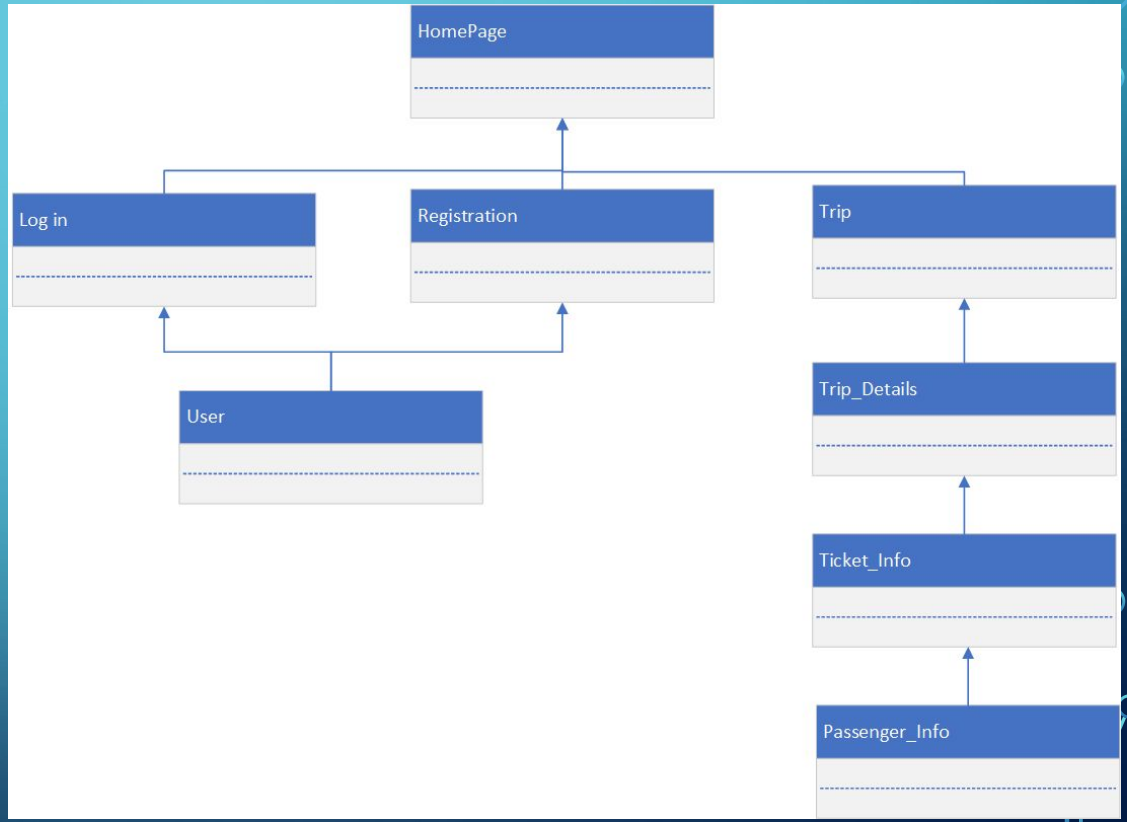
Use cases diagram of Bus Booking System

CLASS DIAGRAM

Design

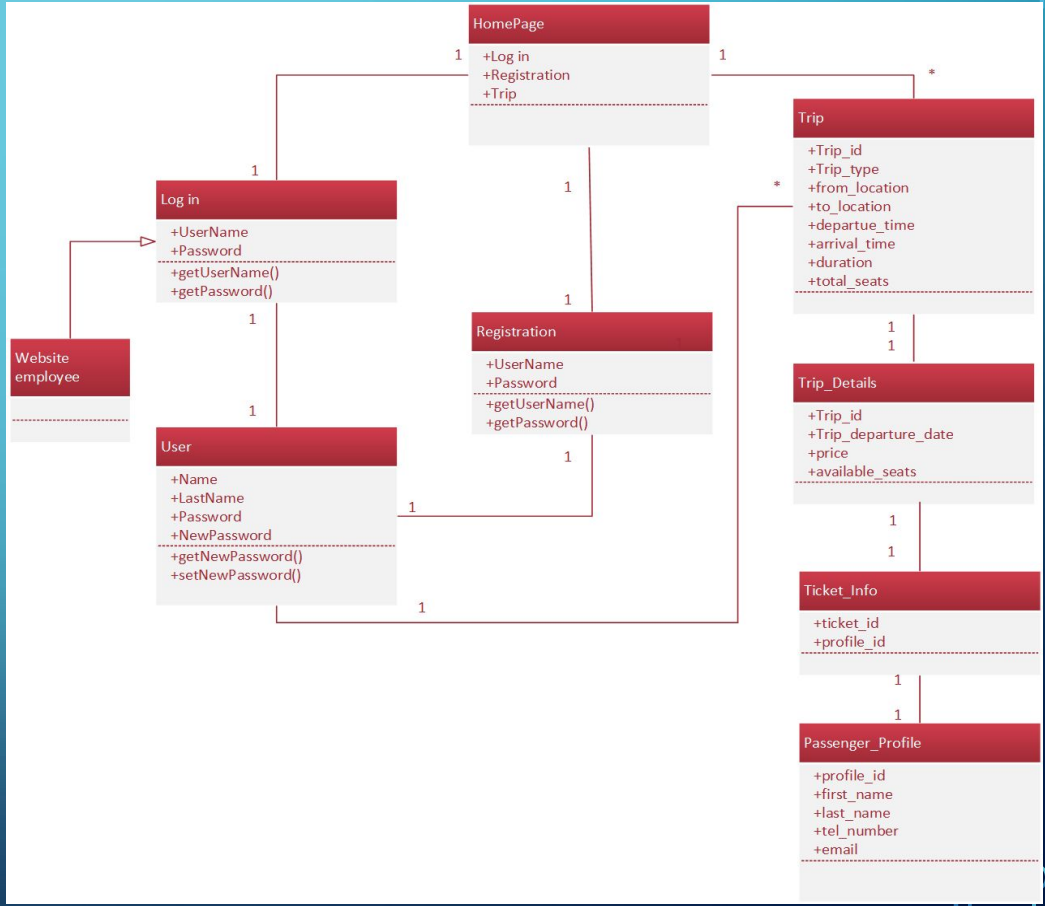
Classes: 6

- HomePage
- Log In
- Registration
- User(User Profile)
- Trip
- Trip_details
- Ticket_Info
- Passenger_Info



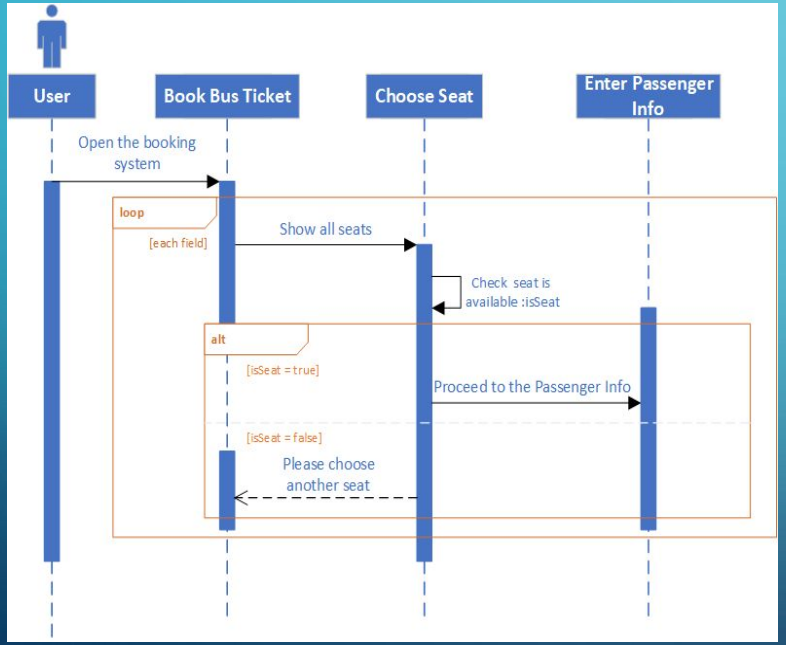
CLASS DIAGRAM

- Relationship type between objects is Association.
- Directed Association between Website employee & Log in.
- Multiplicity almost between every object is equal to one to one.
- HomePage and Trips & User and Trips have one to many.

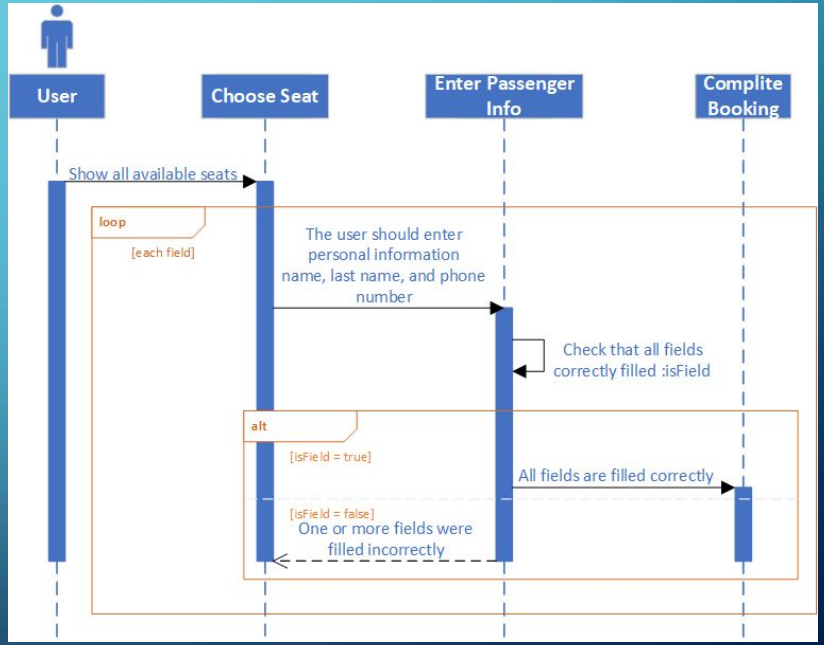


SEQUENCE DIAGRAM

Choose seat

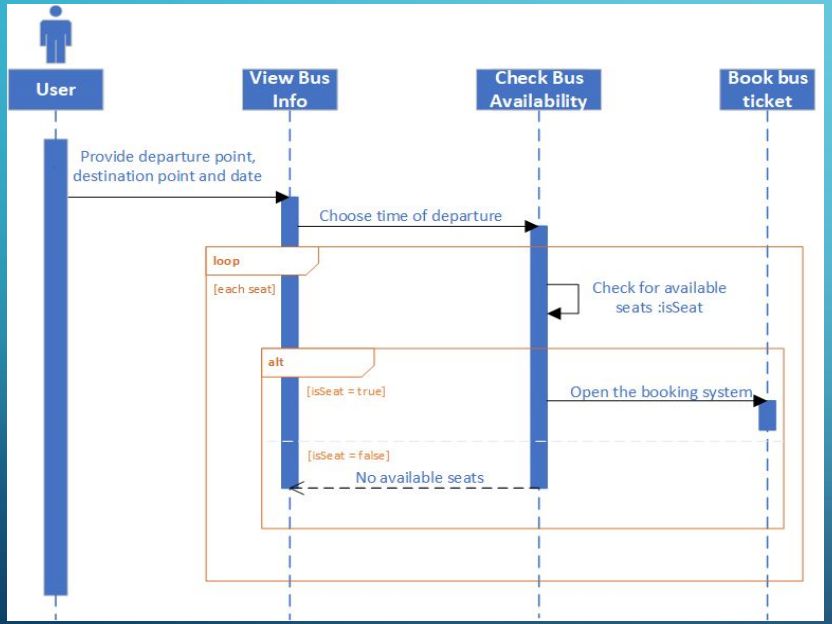


Entering Passenger Information

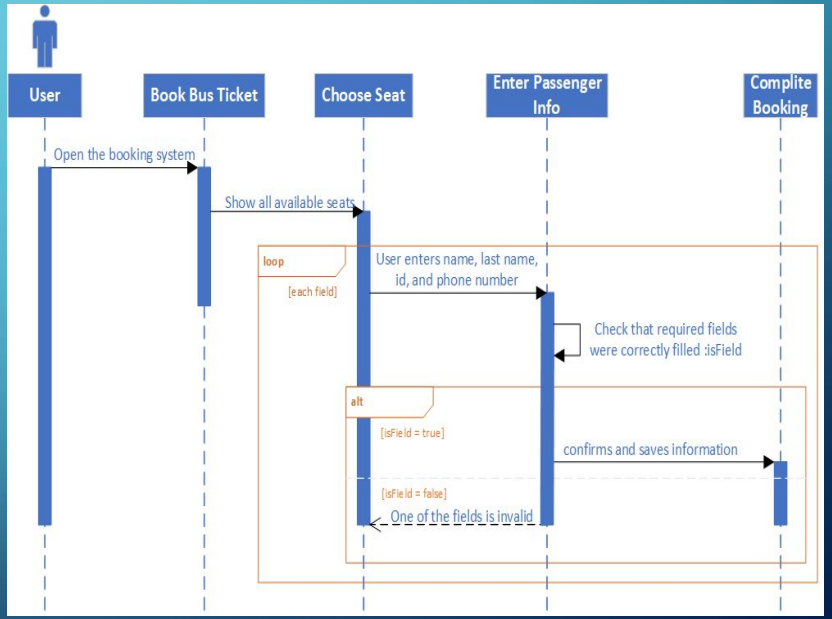


SEQUENCE DIAGRAM

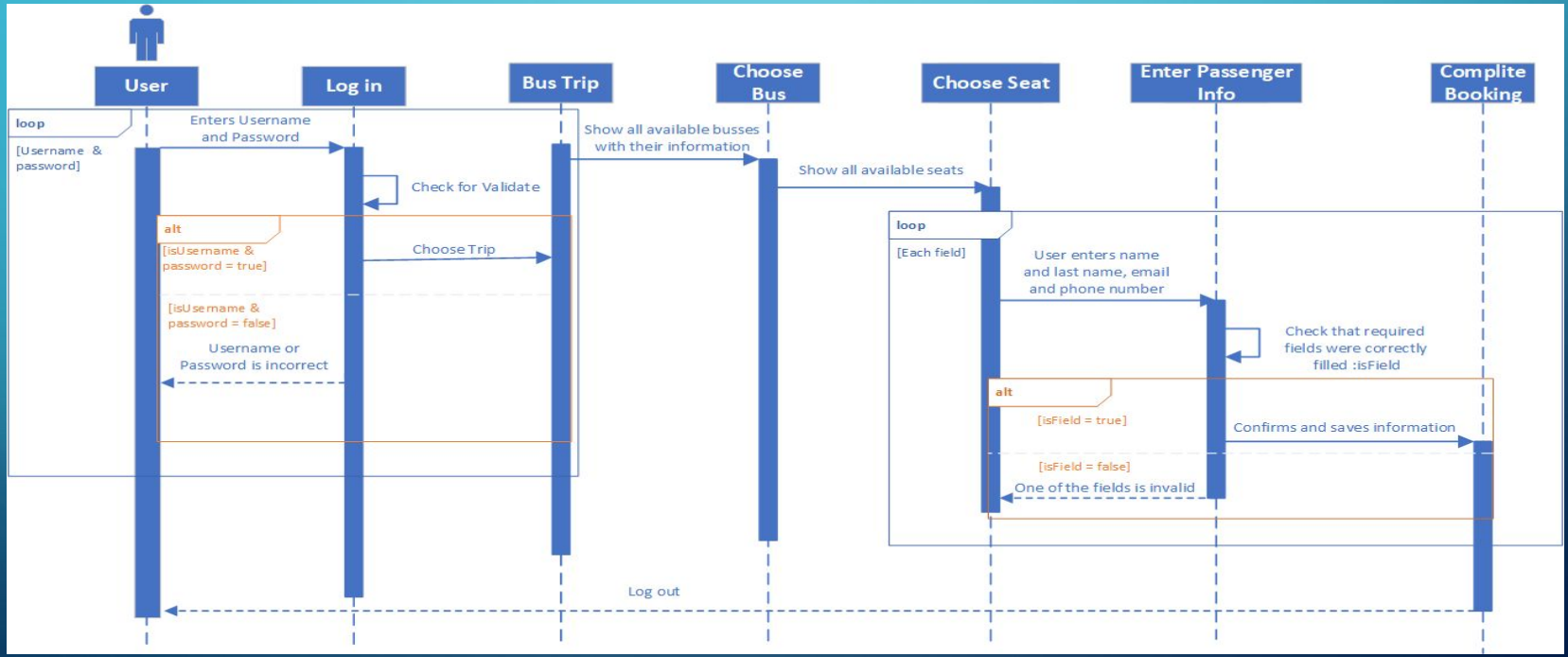
Check Bus Availability



Booking Tickets

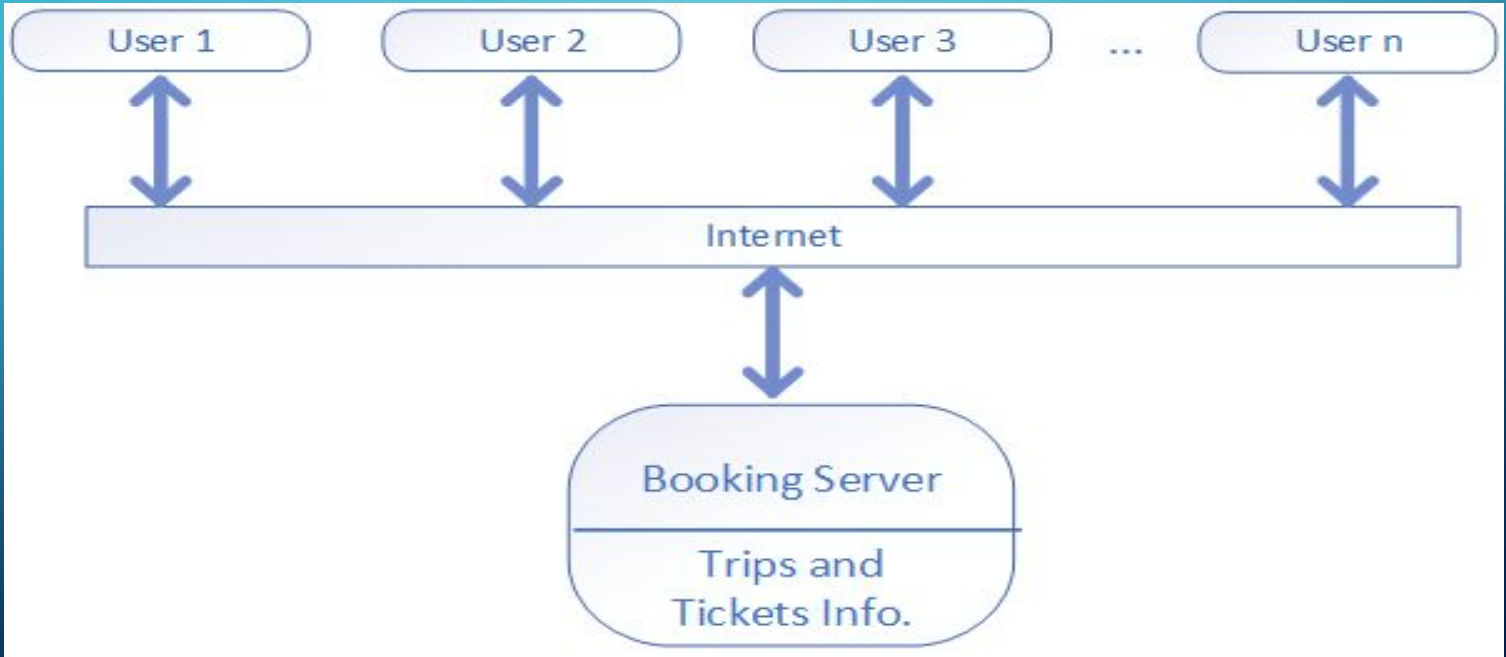


SEQUENCE DIAGRAM



ARCHITECTURE

Client-Server Architecture



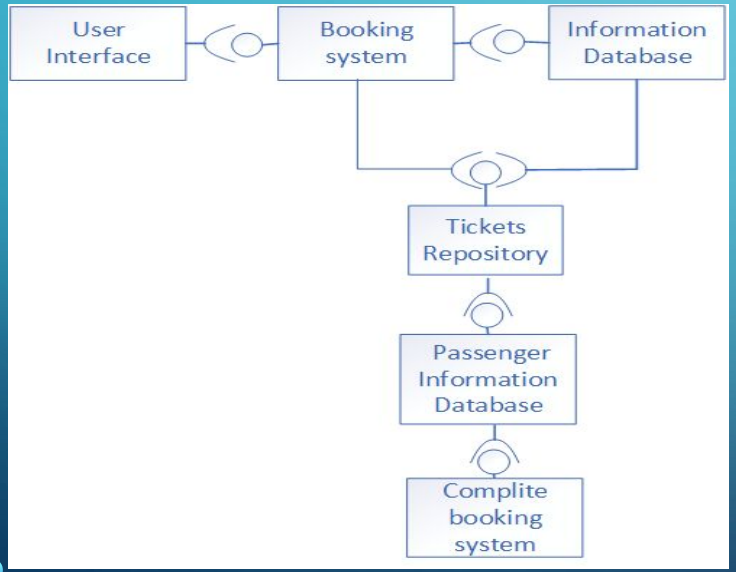
Client-Server Architecture

Why did we chose it?

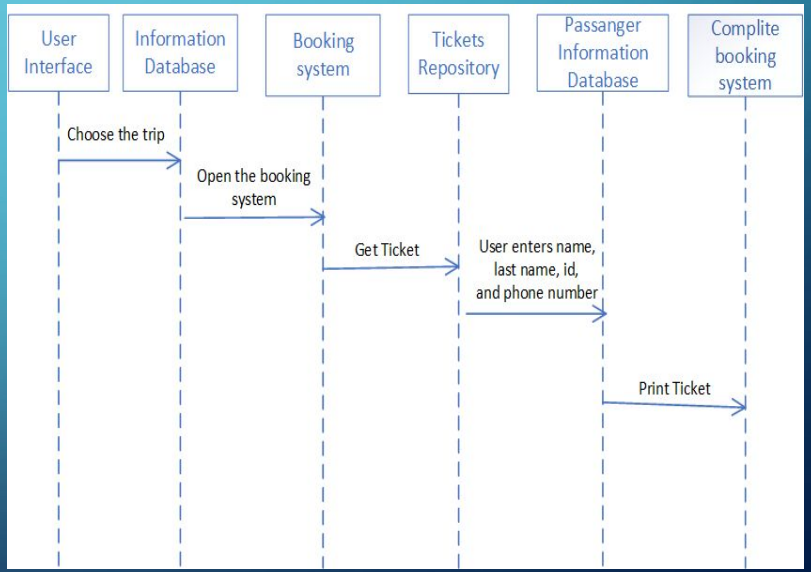
- It has a simple architecture
- It allows clients to access the website whenever a person needs it.
- This architecture was designed to not have difficulty providing needed service to clients.

ARCHITECTURE

Logical View

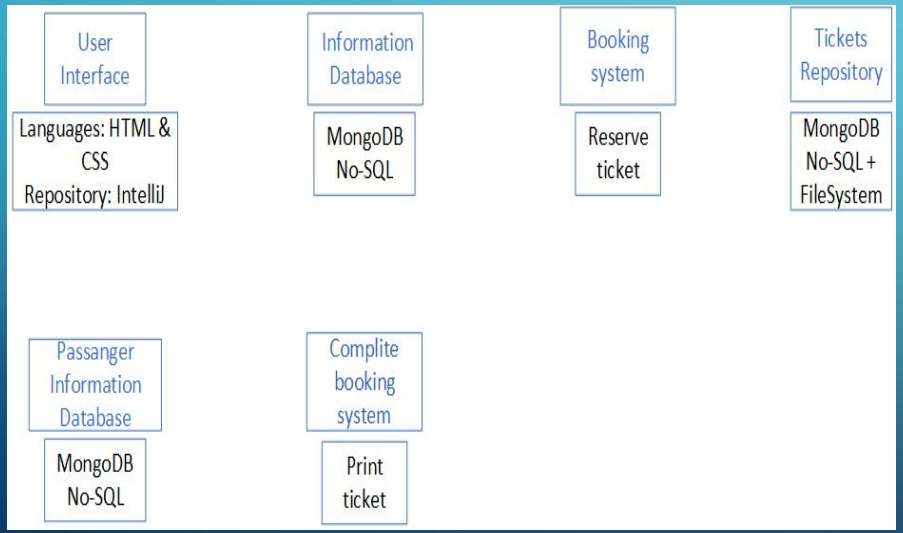


Process View

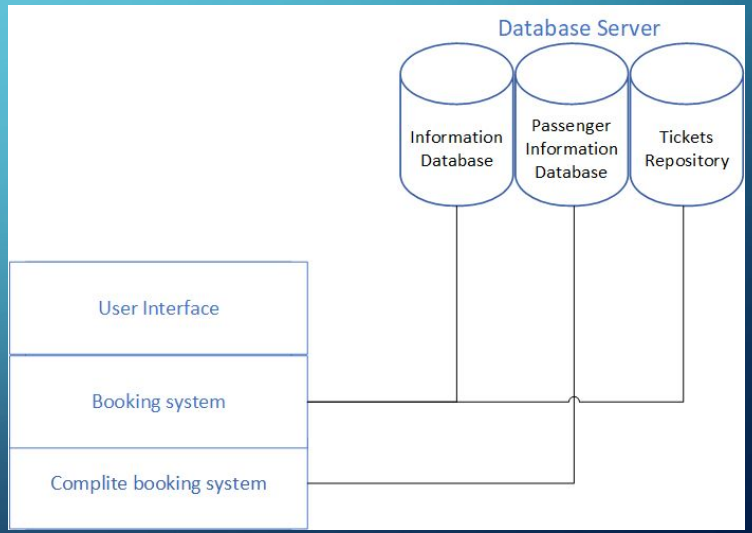


ARCHITECTURE

Development View

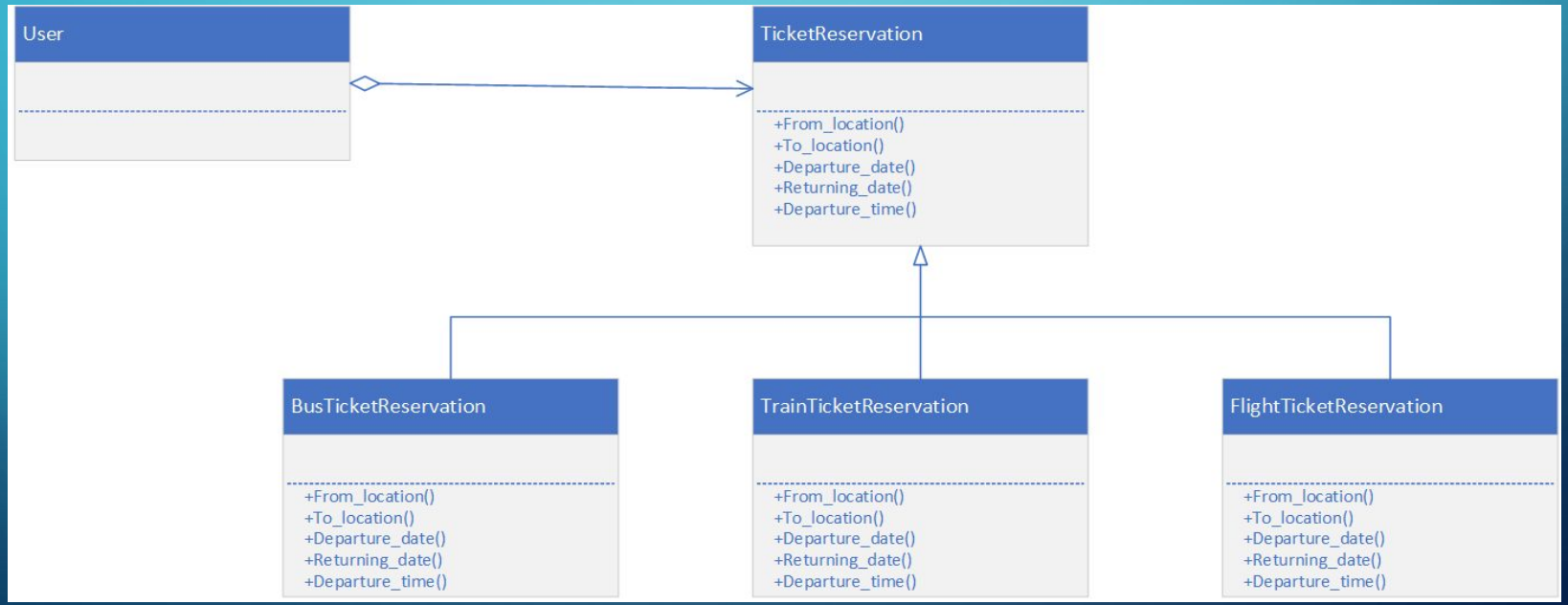


Physical View



DESIGN PATTERNS

Strategy Pattern



TESTING

Functional

```
beforeEach(function() {
  $scope = {};
  // localStorage = {};
  controller = $controller('signinController', { $scope: $scope});
  // myData = _myData_;
  store = {};
  localStorage = window.localStorage;
  spyOn(localStorage, 'getItem').and.callFake(function (key) {
    return key in store ? store[key] : null;
  });
  spyOn(localStorage, 'setItem').and.callFake(function (key, value) {
    return store[key] = value + '';
  });
  spyOn(localStorage, 'clear').and.callFake(function () {
    store = {};
  });
});

it('should not allow the the user to sign in', function() {
  $scope.userEmail = 'notrightemailformat';
  $scope.userPassword = 'shouldnotmatter';
  expect(loggedOnService.signedIn).toBe(false)
});
it('should allow the user to sign in', function() {
  $scope.userEmail = 'salhassan1@student.gsu.edu';
  $scope.userPassword = 'pass';
  $scope.signin();
  expect(loggedOnService.signedIn).toBe(true)
});
```

White Box

```
beforeEach(function() {
  $scope = {};
  controller = $controller('passwordController', { $scope: $scope });
  store = {};
  localStorage = window.localStorage;
  spyOn(localStorage, 'getItem').and.callFake(function (key) {
    return key in store ? store[key] : null;
  });
  spyOn(localStorage, 'setItem').and.callFake(function (key, value) {
    return store[key] = value + '';
  });
  spyOn(localStorage, 'clear').and.callFake(function () {
    store = {};
  });
  $scope.userPassword = 'shouldnotmatter';
  localStorage.setItem('uPassword', $scope.userPassword)
});

it('ensure correct current user password is entered', function() {


  // $scope.chgpass();
  expect(localStorage.getItem('uPassword')).toBe('shouldnotmatter');
});
it('ensure the current user password is changed', function() {
  $scope.newpass = 'pass';
  localStorage.setItem('uPassword', $scope.newpass)
  expect(localStorage.getItem('uPassword')).toBe('pass');
});
```

DEMONSTRATION

https://5aleh.github.io