

## Assignment 4 report Explaining the functionality and the implementation of the project:

### MongoDB database:

- For incorporating MongoDB in our web application, we have used “mongoose”, which is a MongoDB object modeling tool designed to work in an asynchronous environment.
- It can be installed by executing the following command in the root directory of our application in the command window.  
`npm install mongoose`
- Connection to database using mongoose has been implemented as shown in the following figure:

```
var dbConfig = require('./db');
var mongoose = require('mongoose');
// Connect to DB
mongoose.connect(dbConfig.url);
```

- Online Database Host



- Online Database

A screenshot of the mLab MongoDB interface. The top navigation bar shows tabs for 'Documents', 'Indexes', 'Stats', and 'Tools'. The 'Documents' tab is selected. Below the tabs, there is a search bar with the placeholder 'Start new search' and a button to 'Delete all documents in collection'. A 'Add document' button is also present. The main area displays the 'All Documents' section with a table header for 'Display mode' (radio buttons for 'list' and 'table') and 'records / page' (dropdown menu). Three documents are listed, each with a preview, edit icon, and delete icon. The first document has the following JSON structure:

```
{ "_id": { "$oid": "59fcfde3903e52a21ec8662" }, "lastName": "Eric", "firstName": "Jim", "email": "admin@csjtu.edu" }
```

The second document has:

```
{ "_id": { "$oid": "59fd09f265d173d121165998" }, "lastName": "Zhai", "firstName": "Yiming", "email": "yiminn@csjtu.edu" }
```

The third document has:

```
{ "_id": { "$oid": "59fd0b6dcae8eee121d48ad0" }, "lastName": "Deng", "firstName": "Zening", "email": "zennnn@csjtu.edu" }
```

On the right side of the interface, there is a sidebar titled 'Documents (aka Objects)' with descriptive text about the 'Documents' tab and a note about unsupported map/reduce queries.

- User Schema: This schema consists of details of a registered user. The user details that are stored are: Username, password, email, first name of user and last name of user.

```
var mongoose = require('mongoose');

module.exports = mongoose.model('User',{
  id: String,
  username: String,
  password: String,
  email: String,
  firstName: String,
  lastName: String
});
```

- Review Schema: This schema has been designed for CRUD operations related to the reviews given by a user for a place he or she visits. It consists of Username (of the user giving the review), review given (the content), creation date (date when a new comment was given by this user), update date (the date on which a user updated one of their previously given comments).

```
var mongoose = require('mongoose');

var UserReview = mongoose.model('UserReview',{
  name: {
    type: String,
    required: true
  },
  content: String,
  create_date: {
    type: Date,
    default: Date.now
  },
  update_date: {
    type: Date,
    default: Date.now
  }
});

module.exports = UserReview;
```

### RESTful API:

- We have used RESTful API in our web application to send requests and receive responses in our web application by using HTTP protocols GET, POST, PUT and DELETE.
- We are using RESTful API for user reviews, i.e., creation, deletion, display and updating of user reviews will be performed by our web application by using RESTful API.
- Performing Create operation using RESTful API and HTTP POST
  - Code for implementing:

```

    router.post('/insert', function(req, res, next) {
      var item = {
        name: req.body.name,
        content: req.body.content
      };

      var data = new UserReview(item);
      data.save();

      res.redirect('/');
    });
  
```

- Screenshots showing implemented functionality:

Before Creating a new review

The screenshot shows a web application interface. At the top, there's a navigation bar with links for Home, About, Contact, and a search bar. The main content area has a form for creating a new review. The form includes fields for Name (with 'Minglei' entered) and Content (with 'This is a new comment!'), and an 'INSERT' button. Below the form is a section titled 'Reviews' containing two comments:

- ID: 59fcf863d3b01d06df3e229b
  - User: Chaya
  - Hired some bikes to cycle to Golden Gate and what a fab end to the holiday it was. I reckon it's the best way to experience the bridge. Had to pinch myself that I was actually cycling across this famous landmark. Took us three hours from Fishermans Warf (total round trip) but took lots of stops for pics. Very easy ride, a few hills but nothing major.
- ID: 59fd0e1d4266f40895721f49
  - User: Yiming
  - Hired some bikes to cycle to Golden Gate and what a fab end to the holiday it was. I reckon it's the best way to experience the bridge. Had to pinch myself that I was actually cycling across this famous landmark. Took us three hours from Fishermans Warf (total round trip) but took lots of stops for pics. Very easy ride, a few hills but nothing major.

## After Creating a new review

The screenshot shows a web application interface. At the top, there is a header bar with a logo, a search bar, and a 'Join us' button. Below the header, there is a navigation menu with links for Home, About, and Contact. The main content area has two sections: 'Content' and 'Name'. There is a large text input field for 'Content' with an 'INSERT' button below it, and a smaller text input field for 'Name' with a 'DELETE' button to its right. Below these fields is another section labeled 'Content' with a text input field and an 'UPDATE' button. In the bottom left corner, there is a heading 'Reviews' followed by a list of three comments, each enclosed in a box:

- ID: 59fcf863d3b01d06df3e229b  
• User: Chaya  
• Hired some bikes to cycle to Golden Gate and what a fab end to the holiday it was. I reckon it's the best way to experience the bridge. Had to pinch myself that I was actually cycling across this famous landmark. Took us three hours from Fishermans Warf (total round trip) but took lots of stops for pics. Very easy ride, a few hills but nothing major.
- ID: 59fd0e1d4266f40895721f49  
• User: Yiming  
• Hired some bikes to cycle to Golden Gate and what a fab end to the holiday it was. I reckon it's the best way to experience the bridge. Had to pinch myself that I was actually cycling across this famous landmark. Took us three hours from Fishermans Warf (total round trip) but took lots of stops for pics. Very easy ride, a few hills but nothing major.
- ID: 59fd49f90ed1063f241cdbfb  
• User: Minglei  
• This is a new comment!

- Performing Read operation using RESTful API and HTTP GET

- Code for implementing:

```
router.get('/display', function(req, res, next) {
  UserReview.find({}, function(err, doc) {
    res.render('display', {items: doc});
  });
});
```

- Screenshots showing implemented functionality:  
Displaying Review

**Reviews**

- ID: 59fcf863d3b01d06df3e229b
- User: Chaya
- Hired some bikes to cycle to Golden Gate and what a fab end to the holiday it was. I reckon it's the best way to experience the bridge. Had to pinch myself that I was actually cycling across this famous landmark. Took us three hours from Fishermans Warf (total round trip) but took lots of stops for pics. Very easy ride, a few hills but nothing major.

- ID: 59fd0e1d4266f40895721f49
- User: Yiming
- Hired some bikes to cycle to Golden Gate and what a fab end to the holiday it was. I reckon it's the best way to experience the bridge. Had to pinch myself that I was actually cycling across this famous landmark. Took us three hours from Fishermans Warf (total round trip) but took lots of stops for pics. Very easy ride, a few hills but nothing major.

- Performing Update operation using RESTful API and HTTP PUT
  - Code for implementing:

```
router.post('/update', function(req, res, next) {
  var id = req.body.id;

  UserReview.findById(id, function(err, doc) {
    if (err) {
      console.error('error, no entry found');
    }
    doc.name = req.body.name;
    doc.content = req.body.content;
    doc.save();
  })
  res.redirect('/');
});
```

- Screenshots showing implemented functionality:  
Before Updating a review

The screenshot shows a web browser window with the URL `localhost:3000/display`. The page title is "Tour Guide for Spartans". The main content area displays a form for updating a review. The form has fields for "Content" (a large text area), "Name" (a text input field containing "Minglei"), and "DELETE" (a button). Below the form is a message: "This comment is already updated!" followed by a green circular icon with a checkmark. There are "INSERT" and "UPDATE" buttons at the bottom. At the top of the page, there is a navigation bar with links for "Home", "About", and "Contact". A search bar with a "Search" button and a "Join us" button are also present.

**Reviews**

- ID: 59fcf863d3b01d06df3e229b
- User: Chaya
- Hired some bikes to cycle to Golden Gate and what a fab end to the holiday it was. I reckon it's the best way to experience the bridge. Had to pinch myself that I was actually cycling across this famous landmark. Took us three hours from Fishermans Warf (total round trip) but took lots of stops for pics. Very easy ride, a few hills but nothing major.

- ID: 59fd0e1d4266f40895721f49
- User: Yiming
- Hired some bikes to cycle to Golden Gate and what a fab end to the holiday it was. I reckon it's the best way to experience the bridge. Had to pinch myself that I was actually cycling across this famous landmark. Took us three hours from Fishermans Warf (total round trip) but took lots of stops for pics. Very easy ride, a few hills but nothing major.

- ID: 59fd49f90ed1063f241cdbfb
- User: Minglei
- This is a new comment!

### After updating a review

This screenshot shows the same web application interface after the review has been updated. The "Content" field is empty. The "Name" field still contains "Minglei". The "DELETE" button is visible. The message "This comment is already updated!" and the green checkmark icon are no longer present. The "UPDATE" button is at the bottom. The rest of the page, including the reviews section and the top navigation, remains the same as in the previous screenshot.

**Reviews**

- ID: 59fcf863d3b01d06df3e229b
- User: Chaya
- Hired some bikes to cycle to Golden Gate and what a fab end to the holiday it was. I reckon it's the best way to experience the bridge. Had to pinch myself that I was actually cycling across this famous landmark. Took us three hours from Fishermans Warf (total round trip) but took lots of stops for pics. Very easy ride, a few hills but nothing major.

- ID: 59fd0e1d4266f40895721f49
- User: Yiming
- Hired some bikes to cycle to Golden Gate and what a fab end to the holiday it was. I reckon it's the best way to experience the bridge. Had to pinch myself that I was actually cycling across this famous landmark. Took us three hours from Fishermans Warf (total round trip) but took lots of stops for pics. Very easy ride, a few hills but nothing major.

- ID: 59fd49f90ed1063f241cdbfb
- User: Minglei
- This is a new comment!

- Performing Delete operation using RESTful API and HTTP DELETE
  - Code for implementing:

```
router.post('/delete', function(req, res, next) {
  var id = req.body.id;
  UserReview.findByIdAndRemove(id).exec();
  res.redirect('/');
});
```

- Screenshots showing implemented functionality:  
Before Deleting a review

The screenshot shows a web application interface titled "Tour Guide for Spartans". At the top, there is a navigation bar with links for Home, About, Contact, and a search bar. Below the navigation, there is a form for managing reviews. The form has fields for Content, Name, and ID, with buttons for INSERT, UPDATE, and DELETE. Below the form, there is a section titled "Reviews" containing three review entries, each enclosed in a box. The first review is from user Chaya, the second from Yiming, and the third from Minglei.

ID	User	Content
59fcf863d3b01d06df3e229b	Chaya	Hired some bikes to cycle to Golden Gate and what a fab end to the holiday it was. I reckon it's the best way to experience the bridge. Had to pinch myself that I was actually cycling across this famous landmark. Took us three hours from Fishermans Warf (total round trip) but took lots of stops for pics. Very easy ride, a few hills but nothing major.
59fd0e1d4266f40895721f49	Yiming	Hired some bikes to cycle to Golden Gate and what a fab end to the holiday it was. I reckon it's the best way to experience the bridge. Had to pinch myself that I was actually cycling across this famous landmark. Took us three hours from Fishermans Warf (total round trip) but took lots of stops for pics. Very easy ride, a few hills but nothing major.
59fd49f90ed1063f241cdbfb	Minglei	This is a new comment!

The screenshot shows the same web application interface after a review has been deleted. The "Reviews" section now only contains the two reviews from Yiming and Minglei, indicating that the review from Chaya has been removed.

ID	User	Content
59fcf863d3b01d06df3e229b	Chaya	Hired some bikes to cycle to Golden Gate and what a fab end to the holiday it was. I reckon it's the best way to experience the bridge. Had to pinch myself that I was actually cycling across this famous landmark. Took us three hours from Fishermans Warf (total round trip) but took lots of stops for pics. Very easy ride, a few hills but nothing major.
59fd0e1d4266f40895721f49	Yiming	Hired some bikes to cycle to Golden Gate and what a fab end to the holiday it was. I reckon it's the best way to experience the bridge. Had to pinch myself that I was actually cycling across this famous landmark. Took us three hours from Fishermans Warf (total round trip) but took lots of stops for pics. Very easy ride, a few hills but nothing major.

## After Deleting a review

### User authentication:

- For user authentication, we have made sure that the already registered users are able to login by providing valid credentials as input.
- The users who try to login and are not yet registered receive an error message on the same page: “User not found”.
- Also, if a user tries to register with a user name that already exists, they can an error message on the screen “User Already Exists”
- The screenshots below demonstrate our “user authentication” functionality and the code written for its implementation.

- Code for implementing user authentication while user registration:

```
function(req, username, password, done) {  
  findOrCreateUser = function(){  
    // find a user in Mongo with provided username  
    User.findOne({ 'username' : username }, function(err, user) {  
      // In case of any error, return using the done method  
      if (err){  
        console.log('Error in SignUp: '+err);  
        return done(err);  
      }  
      // already exists  
      if (user) {  
        console.log('User already exists with username: '+username);  
        return done(null, false, req.flash('message','User Already Exists'));  
      } else {  
        // if there is no user with that email  
        // create the user  
        var newUser = new User();  
  
        // set the user's local credentials  
        newUser.username = username;  
        newUser.password = createHash(password);  
        newUser.email = req.param('email');  
        newUser.firstName = req.param('firstName');  
        newUser.lastName = req.param('lastName');  
  
        // save the user  
        newUser.save(function(err) {  
          if (err){  
            console.log('Error in Saving user: '+err);  
            throw err;  
          }  
          console.log('User Registration succesful');  
          return done(null, newUser);  
        });  
      }  
    };  
    // Delay the execution of findOrCreateUser and execute the method  
    // in the next tick of the event loop  
    process.nextTick(findOrCreateUser);  
  }  
}
```

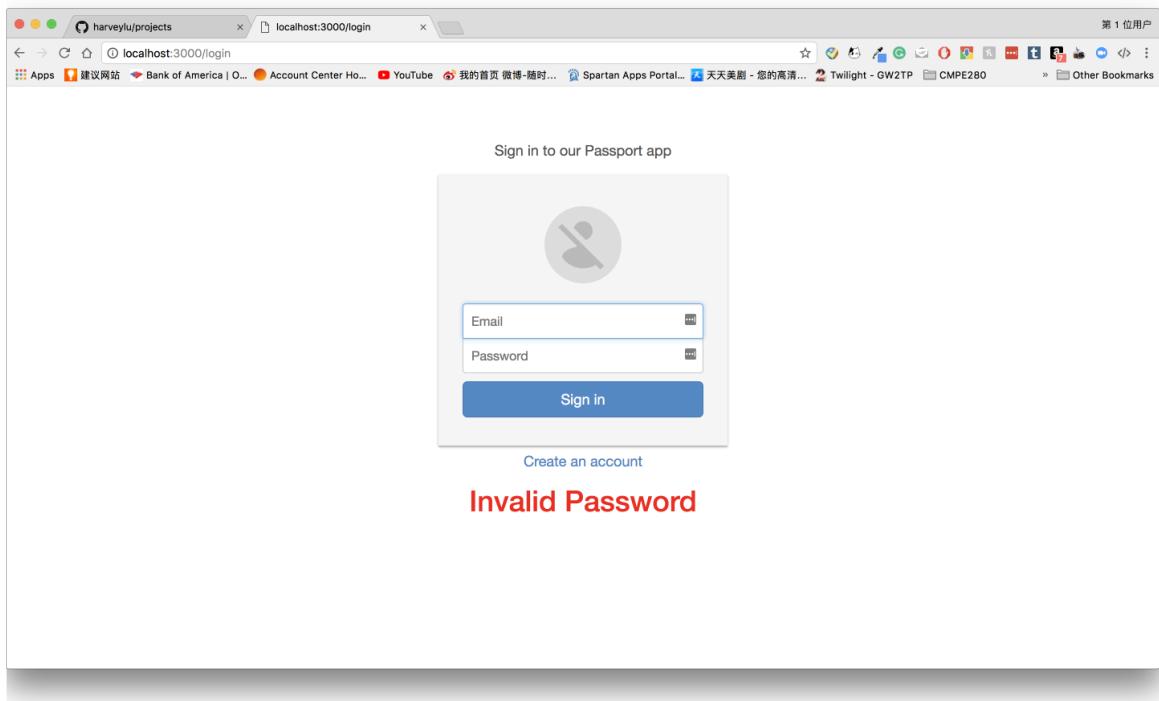
- Code for validating user login credentials or verifying user:

```

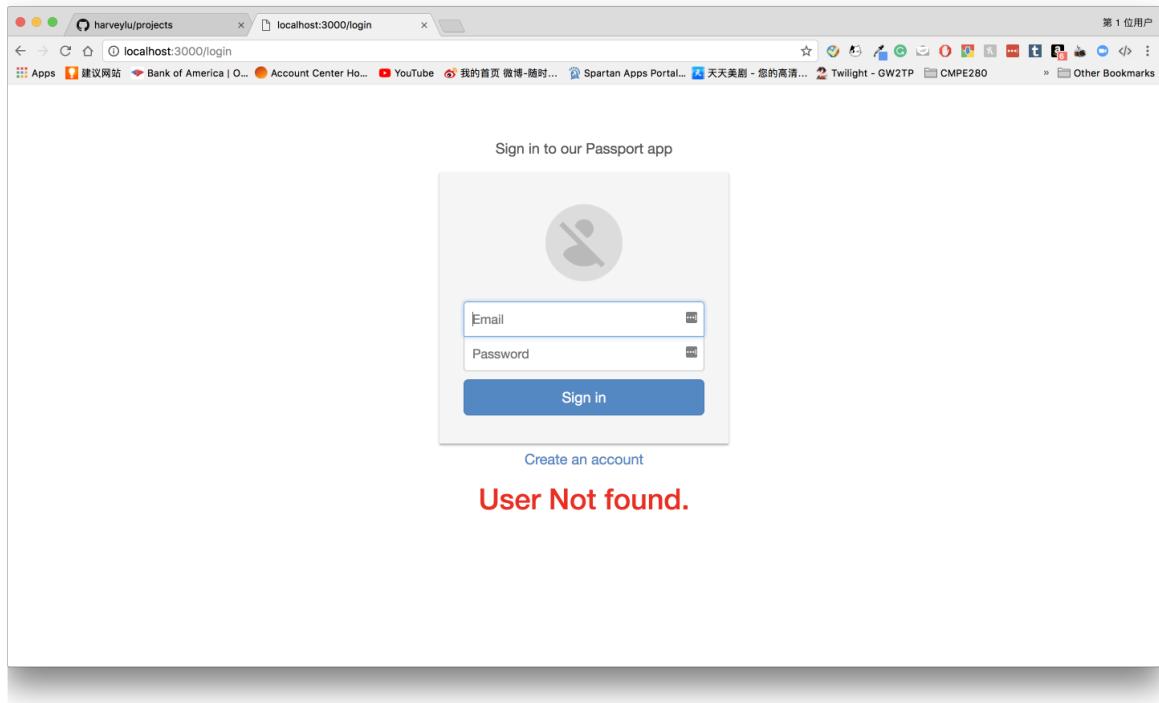
1  var LocalStrategy = require('passport-local').Strategy;
2  var User = require('../models/user');
3  var bcrypt = require('bcrypt-nodejs');
4
5  module.exports = function(passport){
6
7      passport.use('Login', new LocalStrategy({
8          passReqToCallback : true
9      },
10         function(req, username, password, done) {
11             // check in mongo if a user with username exists or not
12             User.findOne({ 'username' : username },
13                 function(err, user) {
14                     // In case of any error, return using the done method
15                     if (err)
16                         return done(err);
17                     // Username does not exist, log the error and redirect back
18                     if (!user){
19                         console.log('User Not Found with username '+username);
20                         return done(null, false, req.flash('message', 'User Not found.'));
21                     }
22                     // User exists but wrong password, log the error
23                     if (!isValidPassword(user, password)){
24                         console.log('Invalid Password');
25                         return done(null, false, req.flash('message', 'Invalid Password'));
26                     }
27                     // User and password both match, return user from done method
28                     // which will be treated like success
29                     return done(null, user);
30                 });
31
32     });
33
34     var isValidPassword = function(user, password){
35         return bcrypt.compareSync(password, user.password);
36     }
37
38 };
39
40 }
41

```

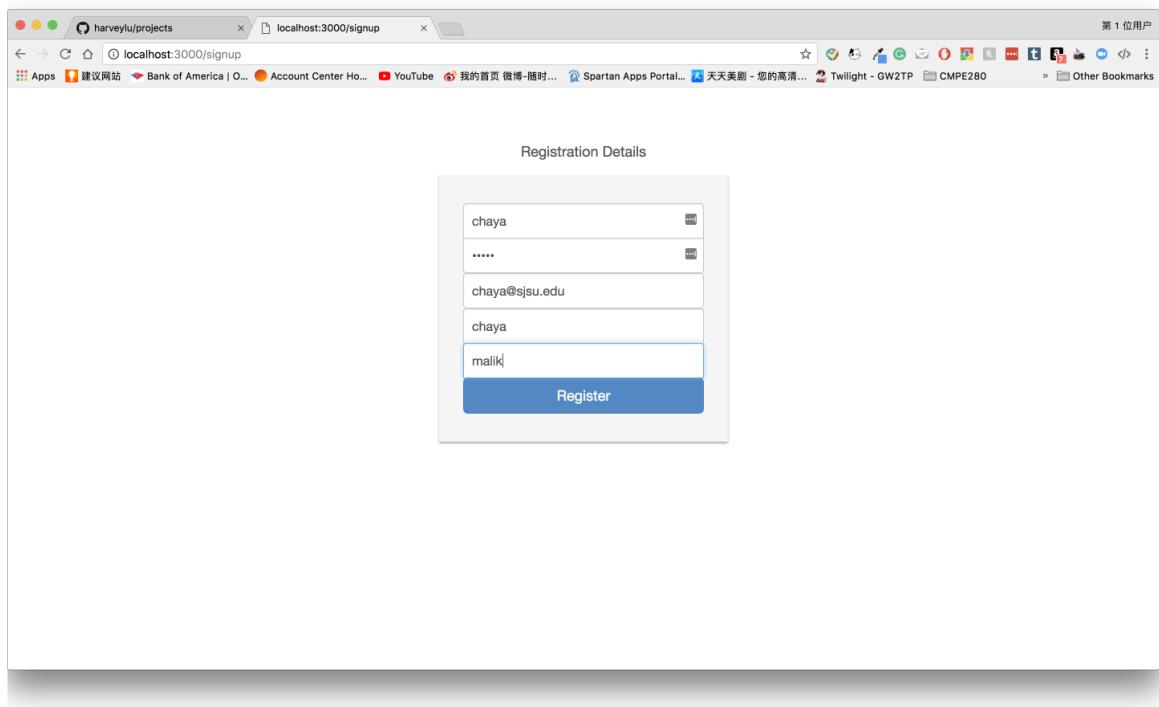
- User Authentication Functionality: “Invalid Password”, a user cannot login if the password provided by them is incorrect.



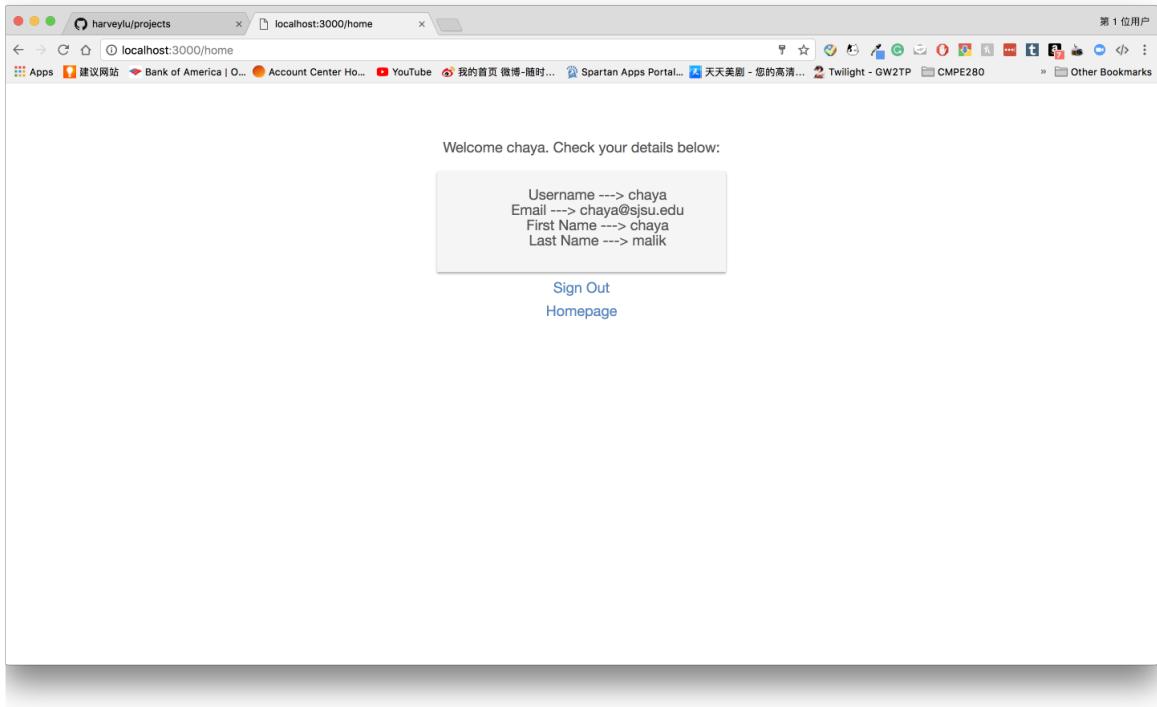
- User Authentication Functionality: “User not found”, a user cannot login using a user name, if they are not registered with the application using that user name.



- User Authentication Functionality: “Registration of a user with the application”

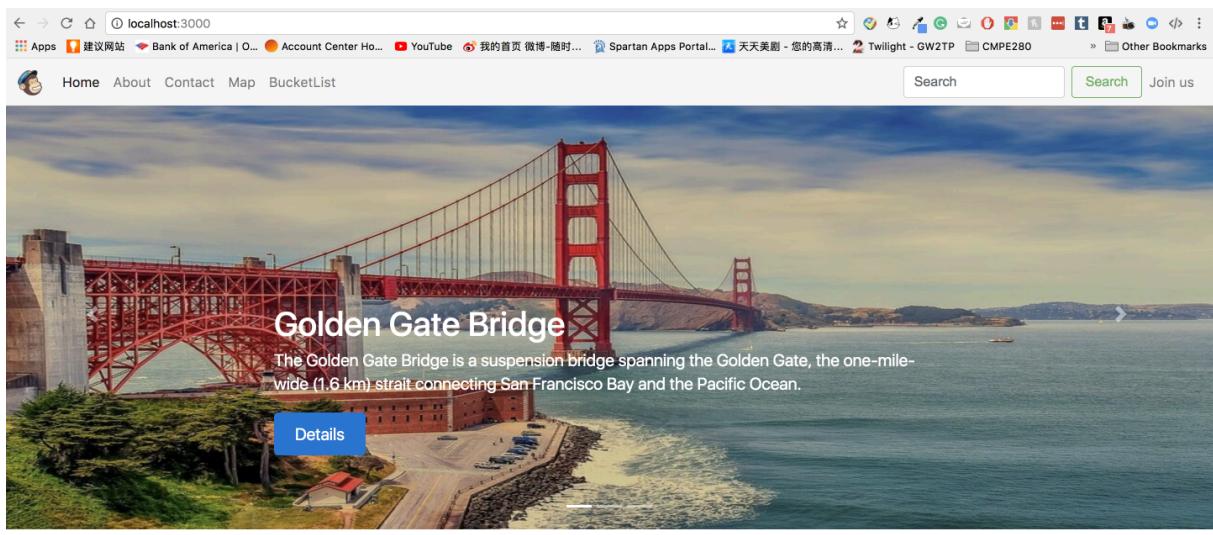


- User Authentication Functionality: “Logged in User”

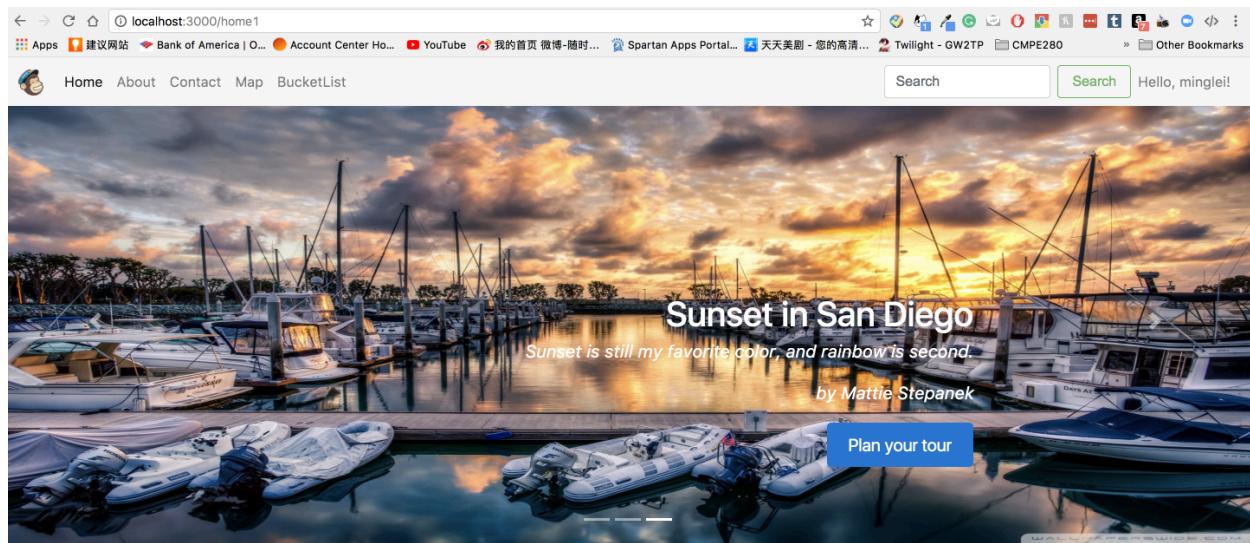


### **Session Management:**

- We have implemented Session Management in our web application to provide a smooth user experience. The user does not have to login every time they access our application.
- By implementing Session Management in our web application, every time a user makes a request our application validates that it is the same user who started the session earlier and does not ask the user for login credentials for every request made.
- The following screenshots show our session management functionality and implementation:
  - Before Login



- After Login (The difference is on the upper-right side, it shows the user's first name)



- Code implementing Session Management in our application
  - Session Management code to check if user is logged in

```
var isAuthenticated = function (req, res, next) {
  // if user is authenticated in the session, call the next() to call the next request handler
  // Passport adds this method to request object. A middleware is allowed to add properties to
  // request and response objects
  if (req.isAuthenticated())
    return next();
  // if the user is not authenticated then redirect him to the login page
  res.redirect('/');
}
```

- Session Management code for page display for an already logged in user (a user who has already initiated the session)

```
/* Handle Login POST */
router.post('/login', passport.authenticate('login', {
  successRedirect: '/home',
  failureRedirect: '/login',
  failureFlash : true
}));

/* GET Home Page */
router.get('/home', isAuthenticated, function(req, res){
  res.render('home', { user: req.user });
});
```