GPS Trails

Prepared by
Alekh Meka, Ayaan Siddiqui, Karan Vishwakarma, Maseeh Khan
University of Illinois Chicago

April 2021

Table of Contents

List of Figures

List of Tables

Project Description

Project Overview

GPS Trails is a user-centralized web-application that is designed to foster a new community of nature enthusiasts. Some of its features include allowing users to input their own trial discoveries and reviewing previous trails already in the database. GPS Trails is implemented using 3 stand-alone components: Angular for the client facing front-end, NestJS as the backend API server that talks to a MySQL database to store and fetch trail and user related data. Additionally, we make use of the Mapbox API in order to serve a custom interactable map interface.

Project Domain

The domain of this project covers the need for an effective social platform that addresses the needs of our consumers. Due to this demand, user security and confidentiality are highly regarded. This product is expected to have a variety of users, with varying interest in the hiking community.

Relationship to Other Documents

This document only uses the original GPS Trails development project document as it's only source.

Naming Conventions and Definitions

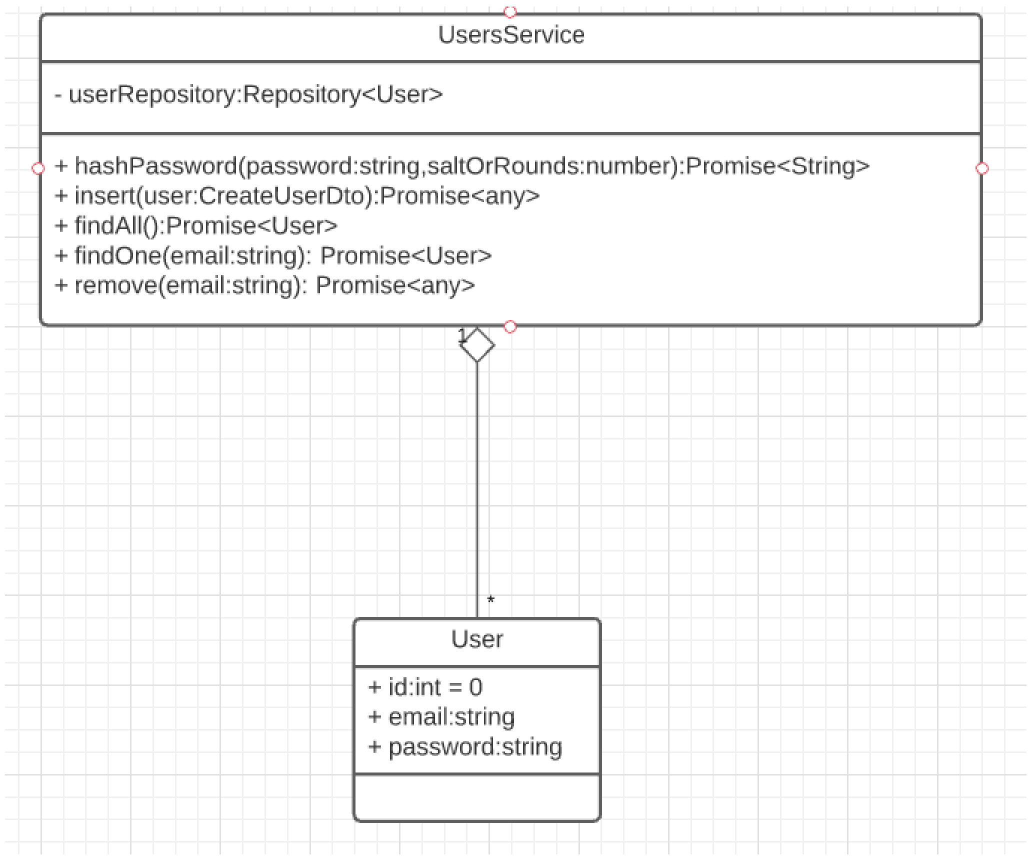
Definitions of Key Terms

Since this project based on the implementation of the original GPS Trails development project, all key terms are also valid in this document. However, there are two terms that we would like to clear.

- **Park:** Used to describe a national parks/preservation. (Ex. Yosemite Nation Park)
- **Trail:** Used to describe the different paths or routes present within a specific park.

UML and Other Notation Used in This Document

Content


UsersService
- userRepository:Repository<User>
+ hashPassword(password:string,saltOrRounds:number):Promise<String>
+ insert(user:CreateUserDto):Promise<any>
+ findAll():Promise<User>
+ findOne(email:string): Promise<User>
+ remove(email:string): Promise<any>
1
*
User
+ id:int = 0
+ email:string
+ password:string

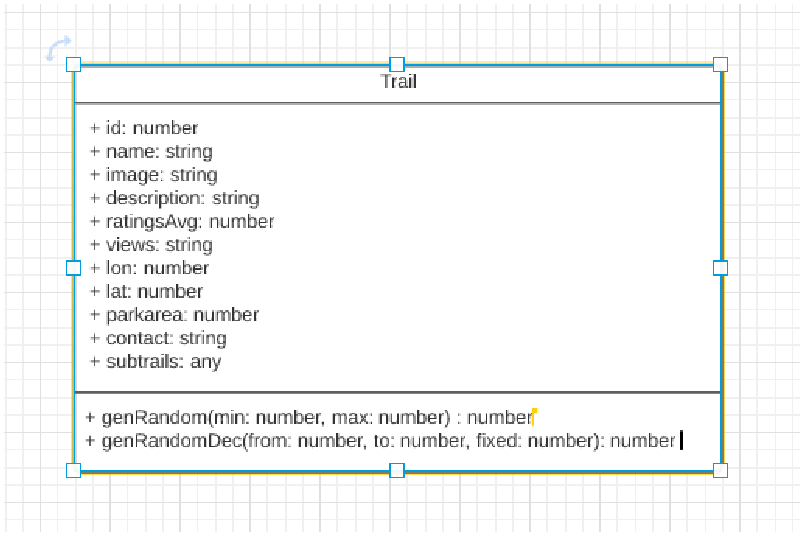
Motivation

The User object describes the model for the database ORM. This model describes the user characteristics as having a unique numeric ID, a unique email address, and a password. The UsersService class allows us to operate on the data given by the database. The UsersService class also provides methods that can be used to handle, create and manage the User model.

Consideration

The User model object exists outside of the UsersService class and has a one-to-many relationship with the User model. Additionally, the Repository<User> is an Array based data-structure that holds elements of type User model.

Content



Motivation

The Trail object describes the model for all the trails. All trails must use this object to create an instance and use this structure everywhere inside the application for consistent data.

Data Dictionary for Any Included Models

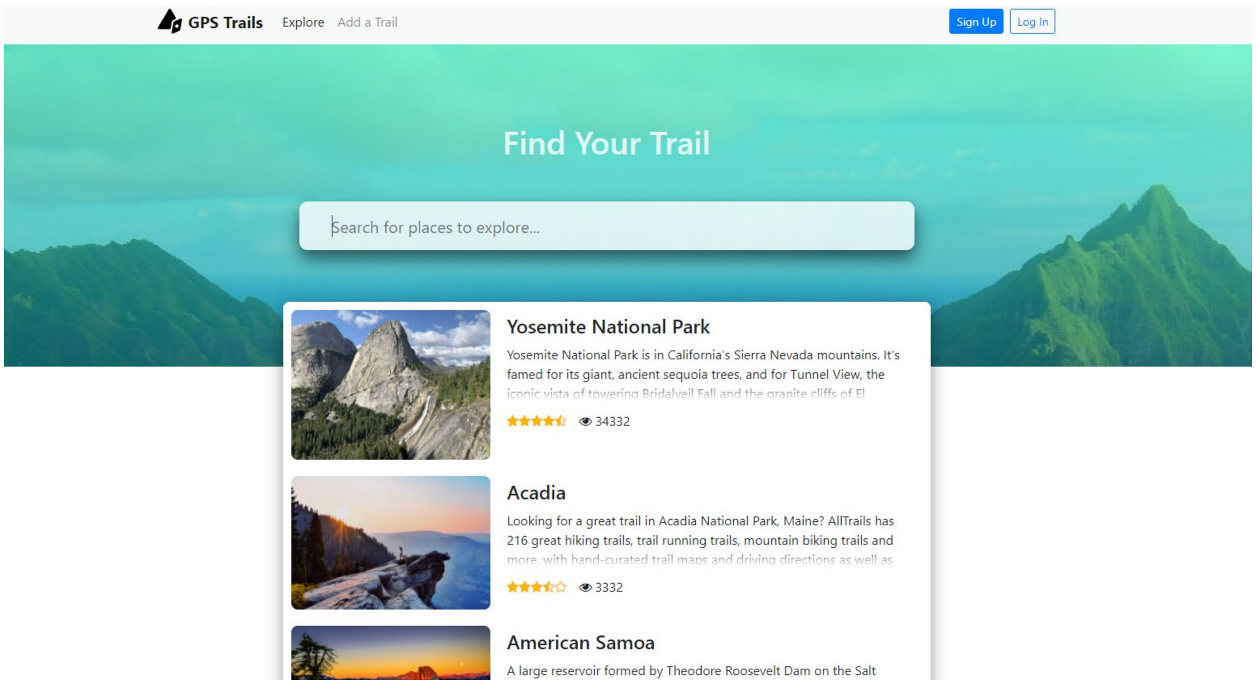
Due to the use of a MySQL database we did not have a need to use any of the more complex data structures like Hash tables or AVL Trees. All data that was needed was provided in a simple Array data by the database ORM.

Project Deliverables

Throughout our releases, we implemented functionality to our project incrementally by addressing several major components for each release. However, we covered several components on both the front end and the back end for each release so that we could generate tangible changes between each release.

First Release

During our first release we presented basic functionality for some of our components and verified that we could connect our Angular front end with our Nest JS back end. Through this release, we demonstrated the Mapbox API and basic trails available in our .json file. In this release, we created the generic ways the user could leave a review and launched our prototype that encompassed most of our initial use cases.



Yosemite National Park

Yosemite National Park is in California's Sierra Nevada mountains. It's famed for its giant, ancient sequoia trees, and for Tunnel View, the iconic vista of towering Bridalveil Fall and the granite cliffs of El

34332

Acadia

Looking for a great trail in Acadia National Park, Maine? AllTrails has 216 great hiking trails, trail running trails, mountain biking trails and more, with hand-curated trail maps and driving directions as well as

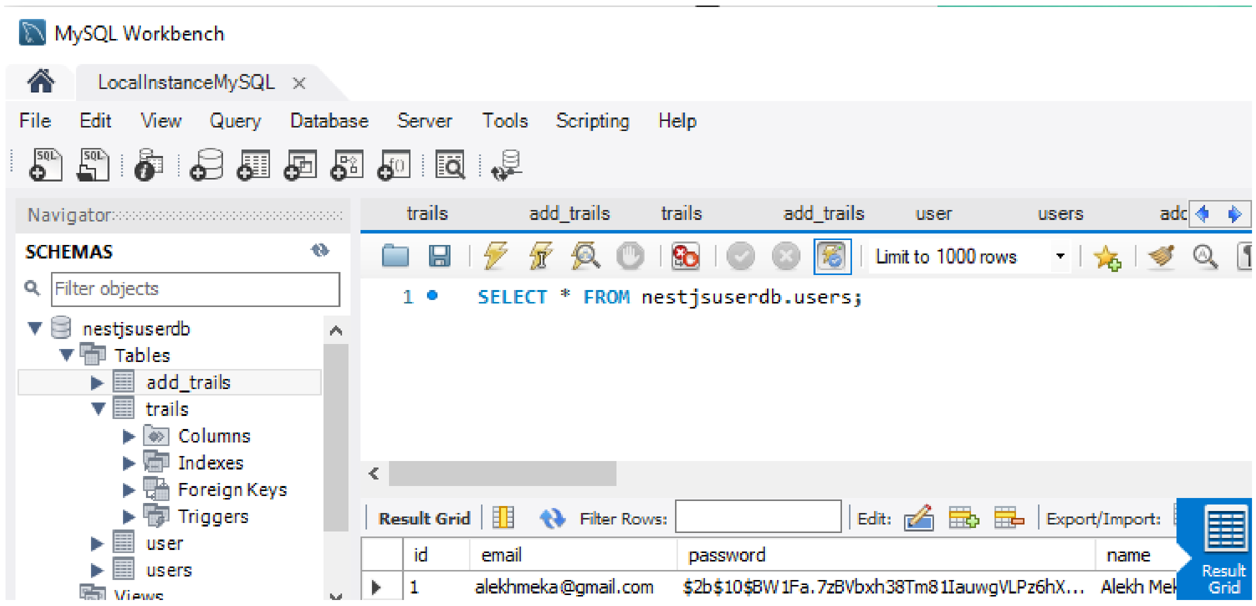
3332

American Samoa

A large reservoir formed by Theodore Roosevelt Dam on the Salt

Second Release

Our second release demonstrated our completed functionality with our databases and user authentication by storing all relevant information in our MySQL tables. In this release, we focused on implementing a safe cryptographic way of storing passwords and improved the functionality of previous components. We had also added our Add-A-Trail component and included a Sub-Trails section for each trail in our database.



```
SELECT * FROM nestjsuserdb.users;
```

id	email	password	name
1	alekhmeka@gmail.com	$2b$10$BW1Fa.7zBVbxh38Tm81IauwgVLPz6hX...	Alekh Mel

Comparison with Original Project Design Document

The previous group had developed the concept of GPS Trails as a mobile application for users. Our group had implemented it as a web application so that it could be used on both mobile and desktop platforms. The development group planned to launch this application on the Google Play Store and Apple Store. Our group has built a working prototype that can also be rendered into a mobile application in the future.

Testing

Items to be Tested

ID #1 – Login

ID #2 – Register

ID #3 – AddTrail

ID #4 – Home

ID #5 – Trail.genRandom

ID #6 – Trail.genRandomDec

Test Specifications

ID# 1 - Login

Description: The login credentials are supplied by the users for access to the trails.

Items covered by this test: The login components of GPS Trails

Requirements addressed by this test:

- Unique identification for all users so that they can save their sessions.
- Checking the connectivity of database with the server.

Environmental needs: The user needs to be connected to the internet and must have registered with a username and password to start the account.

Intercase Dependencies: Following test should precede before the Login test.

- Testing of the credentials of the user in the registration components. The credentials of the user in the registration components.
- Verification for the Connectivity to the internet.

- Verification Connectivity to the internet.

Test Procedures: The user has to register in the application with the details which will be utilized for the login components. Once the registration of the user is successful, the user will be directed to the login component. After the redirection, the information such as the username and password is supplied to the Api for testing purposes.

Input Specification: The inputs include the username and the password which are in the form of text, numbers and might also include alphanumeric characters.

Output Specifications: If the user is successfully logged in with the information, then the home component is displayed. On the other hand, if the user fails to provide the correct credentials, then an error message is sent on the front page.

	Correct credentials	**Incorrect credentials**
Inputs	Username and password	Incorrect inputs (username and password)
Outputs	Home Component	Incorrect session page

Pass/Fail Criteria: The passing criteria is the availability of the features which are locked for the guest users or the external people who are not logged inside the GPS Trails application

<u>ID# 2 - Register</u>

Description: Tests to verify that the Register page creates UI elements on our webpage in accordance to their relative position in our prototype.

Items covered by this test: Register Component - Frontend

Requirements addressed by this test: This test verifies that the list view of elements appear in the right location on the user's screen. Also checks to see if the input boxes become populated by the HTTP requests when the user clicks the register button.

Environmental needs: N/A

Intercase Dependencies: Login Component

Test Procedures: The test checks each item in our list of fields obtained from the user and verifies that they are in the right position in the browser. After doing so, it verifies that the hardcoded input provided by the test case to the HTML elements, is the same input that gets populated into our data structure before being sent to our API server.

Input Specification: Check Position Of: Email, Full Name, Password. Verify population of: test@test.com, Test Case, test123

Output Specifications: Position Order: Email(top), Full Name(middle), Password(bottom)

RegisterComponent.email = test@test.com; RegisterComponent.fullname = Test Case; RegisterComponent.password = test123

Pass/Fail Criteria: In order to pass the test, the UI elements must appear in the proper locations, and our List of user input items should be populated according to our test input.

ID# 3 - AddTrail

Description: Tests to verify that the Add-A-Trail page creates UI elements on our webpage in accordance to their relative position in our prototype. Also checks to see if HTTP GET requests are populating our data structure accordingly.

Items covered by this test: AddTrails Component - Frontend

Requirements addressed by this test: This test verifies that the list view of elements appear in the right location on the user's screen. Also checks to see if the input boxes become populated by the HTTP requests when the user clicks the add-a-trail button.

Environmental needs: N/A

Intercase Dependencies: Trails Component

Test Procedures: The test checks each item in our list of fields obtained from the user and verifies that they are in the right position in the browser. After doing so, it verifies that the hardcoded input provided by the test case to the HTML elements, is the same input that gets populated into our data structure before being sent to our API server.

Input Specification: Check Position Of: Email, Name, ParkName, TrailName, Experience. Verify population of: test@email.com, test, TestPark, Trail, Happy

Output Specifications: Position Order: Email(1st), Name(2nd), ParkName(3rd), TrialName(4th), Experience(5th)

AddTrailComponent.email = test@email.com; AddTrailComponent.name = test; AddTrailComponent.parkname = TestPark; AddTrailComponent.trailname = Trail; AddTrailComponent.experience = Happy;

Pass/Fail Criteria: In order to pass the test, the UI elements must appear in the proper locations, and our List of user input items should be populated according to our test input.

ID# 4 - Home

Description: Tests to verify that the Home page creates UI elements on our webpage in accordance to their relative position in our prototype. Also tests to verify that the fields are being populated according to our database.

Items covered by this test: Home Component - Frontend

Requirements addressed by this test: This test verifies that the MapBox API is appearing in the right location on the web browser with the search bar right above it. This API and information is populated from our MySQL DB.

Environmental needs: N/A

Intercase Dependencies: Interaction with Trails Database in Backend

Test Procedures: The test populates our MapBox API with results from our MySQL Database as normal. First, we verify that all the view elements are in the positions designated by the relative layout. From there, we check to see if all our database entries appear in our webpage.

Input Specification: trails; seachList; searchresult

Output Specifications:

homeComponent.trials == MySQLDB.Trails.entries

homeComponent.searchList == MySQLDB.Trails.entries

homeComponent.searchresult == “”

Pass/Fail Criteria: In order to pass the test, the UI elements must appear in the proper locations, and our homeComponent object must be populated with all the corresponding tables in our database.

ID# 5 - Trail.genRandom

Description: Verify that the random numbers generated by the genRandom function of the Trails component differ when generating a random integer of different bounds.

Items covered by this test: Trails Component - Backend

Requirements addressed by this test: This test verifies that the random number generator is working properly for the Trails component. If the numbers generated are equal or fall within the same equivalence class, this test fails its assertions.

Environmental needs: N/A

Intercase Dependencies: Add-A-Trail Component

Test Procedures: The test verifies that each random number generated falls within the bounds in lieu of overflow/underflow errors with error bounds. This is repeated 100,000 times for all three types of comparisons.

Input Specification: The test checks to see if a number obtained from two different equivalence classes is not the same. Then it checks to see if new numbers obtained from two different equivalence classes have the proper less than or greater than relationship.

Output Specifications:

Trail.genRandom(0, 10) == Trail.genRandom(11, 20) -> should return false

Trail.genRandom(-54154,1541) < Trail.genRandom(1542, 45135) -> should return true

Trail.genRandom(54654,456546) > Trail.genRandom(0, 41561) -> should return true

Pass/Fail Criteria: In order to pass the test, the random number generator must be pulling valid numbers from each equivalence class. These three tests are run in a for-loop 100,000 times so that there is a high probability of catching an error if it occurs.

ID# 6 - Trail.genRandomDec

Description: Verify that the random numbers generated by the genRandomDec function of the Trails component differ when generating a random float value of different bounds.

Items covered by this test: Trails Component - Backend

Requirements addressed by this test: This test verifies that the random number generator is working properly for the Trails component. If the numbers generated are equal or fall within the same equivalence class, this test fails its assertions.

Environmental needs: N/A

Intercase Dependencies: Add-A-Trail Component

Test Procedures: The test verifies that each random number generated falls within the bounds in lieu of overflow/underflow errors with error bounds. This is repeated 100,000 times for all three types of comparisons.

Input Specification: The test checks to see if a number obtained from two different equivalence classes is not the same. Then it checks to see if new numbers obtained from two different equivalence classes have the proper less than or greater than relationship.

Output Specifications:

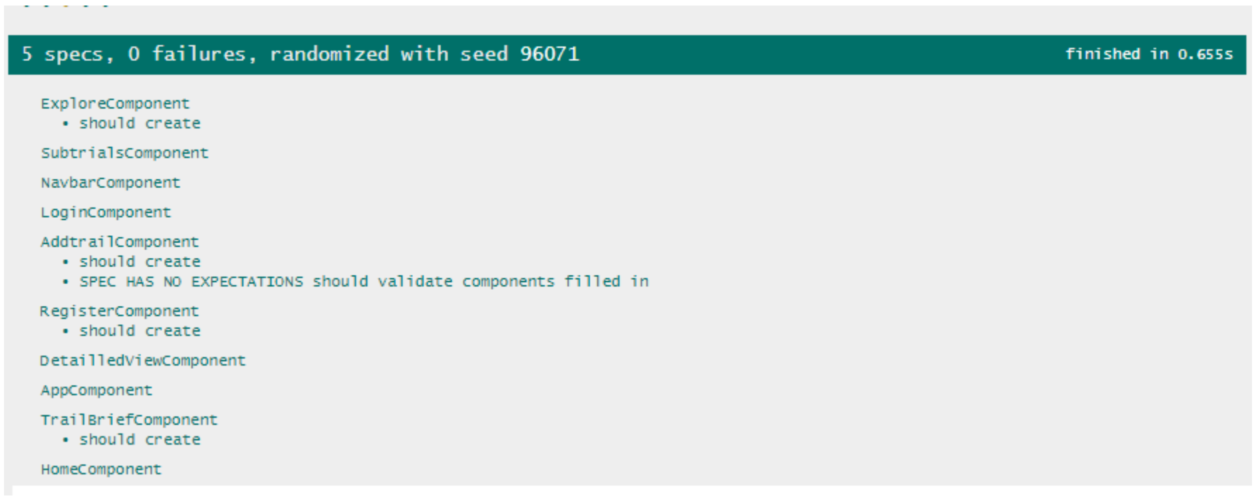
Trail.genRandomDec.(0, 10, 2) == Trail.genRandomDec(11, 20, 2) -> should return false

Trail.genRandomDec(-3451, 20, 2) < Trail.genRandomDec(21, 5125, 3) -> should return true

Trail.genRandomDec(4848, 74515, 3) > Trial.genRandomDec(-15162, 4847, 3) -> should return true

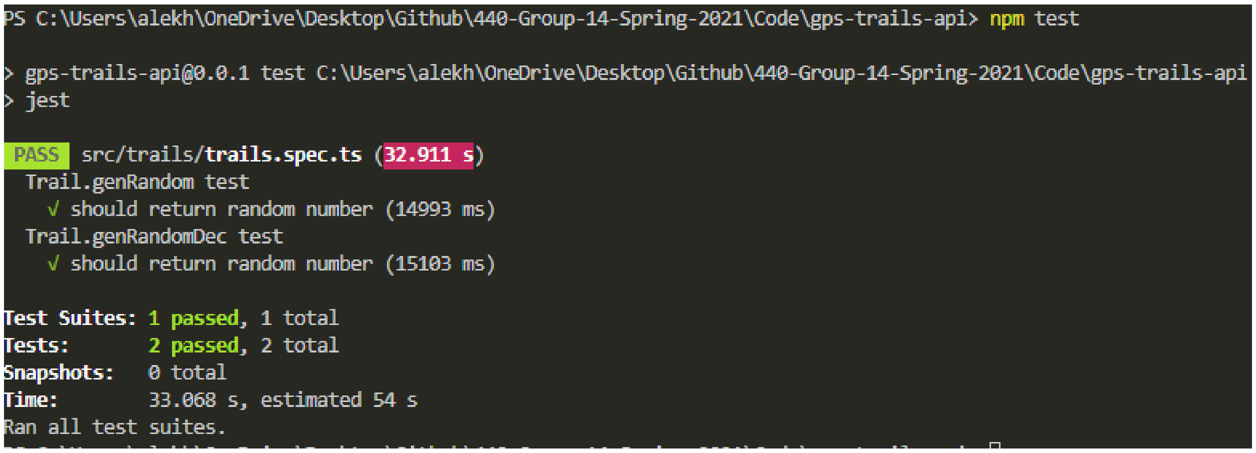
Pass/Fail Criteria: In order to pass the test, the random number generator must be pulling valid floats from each equivalence class. These three tests are run in a for-loop 100,000 times so that there is a high probability of catching an error if it occurs.

Test Results



```
5 specs, 0 failures, randomized with seed 96071                    finished in 0.655s

  ExploreComponent
    • should create
  SubtrialsComponent
  NavbarComponent
  LoginComponent
  AddtrailComponent
    • should create
    • SPEC HAS NO EXPECTATIONS should validate components filled in
  RegisterComponent
    • should create
  DetailledViewComponent
  AppComponent
  TrailBriefComponent
    • should create
  HomeComponent
```



```
PS C:\Users\alekh\OneDrive\Desktop\Github\440-Group-14-Spring-2021\Code\gps-trails-api> npm test

> gps-trails-api@0.0.1 test C:\Users\alekh\OneDrive\Desktop\Github\440-Group-14-Spring-2021\Code\gps-trails-api
> jest

 PASS  src/trails/trails.spec.ts (32.911 s)
  Trail.genRandom test
    √ should return random number (14993 ms)
  Trail.genRandomDec test
    √ should return random number (15103 ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        33.068 s, estimated 54 s
Ran all test suites.
```

<u>**ID# 1 - Login**</u>

Date(s) of Execution: Generated this test prior to Release 2 – Tested the login component thoroughly. After Release 2 – improved functionality with password hashing test. Tested again before creating this document.

Staff conducting tests: Ayaan

Expected Results: test@email.com password123

Actual Results: test@email.com password123

Test Status: PASS

<u>**ID# 2 - Register**</u>

Date(s) of Execution: Generated this test prior to Release 1. Tested again before creating this document.

Staff conducting tests: Maseeh

Expected Results: UI Elements in proper loctions

Actual Results: UI elements in correct locations on multiple browsers

Test Status: PASS

<u>**ID# 3 - AddTrail**</u>

Date(s) of Execution: Generated this test prior to Release 2 – Tested the AddTrail component thoroughly. After Release 2 – improved functionality with password hashing test. Tested again before creating this document.

Staff conducting tests: Karan

Expected Results: UI elements appear in proper location and

AddTrailComponent.email = test@email.com; AddTrailComponent.name = test; AddTrailComponent.parkname = TestPark; AddTrailComponent.trailname = Trail; AddTrailComponent.experience = Happy;

Actual Results: Same as above

Test Status: PASS

ID# 4 - Home

Date(s) of Execution: Generated prior to Release 1 – Improved to test database prior to Release 2. Tested again before creating this document.

Staff conducting tests: Ayaan and Karan

Expected Results: UI elements appear in proper location and populated trails/search list contains all MySQL Trail's Table entries

Actual Results: same as above

Test Status: PASS

ID# 5 – Trail.genRandom

Date(s) of Execution: Generated this test prior to Release 2 – Tested the random number generator for our database. Tested again before creating this document.

Staff conducting tests: Alekh

Expected Results: false, true, true -> 100,000 iterations

Actual Results: same as above

Test Status: PASS

ID# 6 – Trail.genRandomDec

Date(s) of Execution: Generated this test prior to Release 2 – Tested the random float number generator for our database. Tested again before creating this document.

Staff conducting tests: Alekh

Expected Results: false, true, true -> 100,000 iterations

Actual Results: same as above

Test Status: PASS

Regression Testing

Throughout the semester our group had implemented regression testing by running all our previous test cases for our UI components whenever a change was made to our front-end. Prior to our releases, we performed regression testing by running all test cases in the system. Initially before release 2, we were failing several assertions in 3 - AddTrail regarding our connection with our MySQL database, but through debugging and inspection, we had resolved that issue before our demo.

Inspection

Items to be Inspected

*F*ollowing components were inspected during the process.

Addtrails component: This component is responsible for storing the user responses off the different trails.

Home and Detailed Component: The connectivity and data transfer between the Angular framework while ensuring the data abstraction properties of the whole application.

Inspection Procedures

Addtrails component

The Addtrails inspection was performed after the component was fulfilling its basic use cases and delivering the functional requirements for the user.

Kick **off meeting:** Alekh and Maseeh integrated the Addtrails section and implemented the component. A detailed summary of the code with source code of the different branches in GitHub was available for inspection.

During the individual preparation, Ayaan and Karan performed a detailed evaluation of the source code from different branches for the add trails component. The Add trails section incorporated a lot of user forms which are added to the data bases as field. After the inspection, the forms were perfectly placed and had the relevant placeholders for making the usability of GPS Trails effective. The component had a separate styling file which helped in mapping the stylings corresponding to the element in the Document object model.

Home and Detailed Component

The Home component and detailed component delivers the park information that needs to be displayed on the detailed section. This helps the user to get an overview of geographical aspects of the park.

Kick **off meeting:** Karan and Ayaan integrated the detailed section and implemented the components of home component. A detailed summary of the code with source code of the different branches in GitHub was available for inspection.

During the individual meeting, Alekh and Maseeh documented the various reports of the code structures and usability of the GPS Trails application from the perspective of the user. The components were encapsulating data into classes for better code readability and optimizing the data exchange between components. The components were nested inside each other to reuse the coding data from one component to another hence integrating the composite design pattern.

Inspection Results

The Add trails component inspection the usability of components, structure of the code, nesting of components, was evaluated during the inspection. Karan and Ayaan did the inspection for this component. The inspection helped to highlight the interaction between the components and the redundant associations due to the initial overview.

The Detailed component was inspected by Alekh and Maseeh which helped in optimizing the structure of the components. The inspection report helped to detect the edge cases which could have crashed the application.

Recommendations and Conclusions

After completing this project, our group had realized the necessity of designing unit tests incrementally as each component is created for an application. As we designed more test cases, debugging became an easier, natural process where we could clearly identify areas of risk. Integration tests are also a necessity when designing applications for the web. By developing proper integration tests, developers can adequately assess the issues between the API and user interface.

Project Issues

Open Issues

Content

With the development of distributed system, the platform must rely on different factors which might change with time. These issues can mitigate against the working of the use cases of the application.

Motivation

GPS Trails provides a platform for efficient and easy way of accessing the different trails inside the parks. The national park authorities manage the different vegetation and animals which are a park of the park. As the animals are becoming extinct with the changes in the habitats and national parks are becoming stricter in terms of rules and regulations. These might impact the trailing system utilized by these parks. Many people must get special permission for entering some regions of the park which might become cumbersome and difficult to detect which parks are open for all people.

Examples

Regions of the park where the endangered species are found are generally prohibited and must be ensured that the common people and community have no access to that area of the park. This creates issues as the rules and regulations might change with different situation and these issues can conflict with the details of the parks published on the website by people who visited the same area of the park.

Considerations

It is important to address these issues for the long release as these might hinder with the functional requirements of GPS Trails.

Waiting Room

Content

GPS Trails provides a variety of services to provide the users with the optimum functionality through its user interface. Requirements are changing with evolving period and new functionalities are a need of the hour. These required need to be addressed in the future releases to provide the users the features.

Motivation

To keep the product compatible with the needs of the users, the releases should address the functionalities and ensure that no requirement is lost. These requirements need to be taken into account and addressed in the upcoming releases. Currently, the release focus on requirements which will help to implement maximum use cases with minimum costs for the product.

Consideration

Some requirements need to be shifted to the future releases to for providing a waiting room for the other requirements and taking the future requirements seriously. Some of these requirements are.

- Virtual trails: The users should be able to see the trails virtually on their devices to get a better understanding of the terrain of the trails.

- Online refreshments and services: In remote area of the parks, it is important to cater to the needs of the user. As a result, a 24*7 connection is required with the users.

Ideas for Solutions

Content

GPS Trails provides a backend server to support the needs of the user and implement a distributed environment where multiple users can pair up with the website and fulfill their needs. The platform provides an architecture which ensures concurrent execution of the services without any latency in the features provided to the users.

Motivation

The platform is built on a client-server model which serves multiple clients through a common point. The server is backed up by the NEST API which enables object relational mapping with the Node JS libraries. This is helpful and compatible with variety of frameworks and frontend libraries. Implementing the backend with these techniques will help to make a platform independent server which can be utilized for cross-platform development in mobile applications as well.

The front end of the website provides a variety of use cases which will be interacting with each other from time to time. To reduce the coupling between the parts, the components of Angular framework can be used to ensure abstraction and data hiding in the platform. Furthermore, the data exchanged between the components can be filtered and secured from unknown party applications.

The architecture is built on MVC architecture where the model contains the data for the various trails and parks. The View constitutes the front-end Angular components of the website. The controller helps to facilitate the interaction between these two components.

Considerations

The ideas involved during requirement gathering provide a basic overview of the system which might help to solve the complexity of the platform. These ideas might include the design patterns, architecture of the platform, components for the use cases etc. These constitute the first ideas of the platform and software to be developed.

Project Retrospective

Content

The GPS Trails application was segregated into different components to ensure the decoupling of the different use cases of the system and at the same time delivering the functionality of the user. All the use cases were tested, and a thorough inspection was performed to ensure that the website is compatible with the different inputs from the user.

The login component was able to authenticate the user successfully to ensure that only valid users are allowed to access the information about the various information about national parks.

The explore component was able to navigate from one national park to another very efficiently without any latency.

The detailed section was successfully displaying the information about the national parks and locating the latitude and longitude coordinates of the different parks on the Map Box API.

The add trails section connected the front-end Angular framework and saved the user feedback in the database efficiently.

Motivation

The sub trails section was able to point the location of the trails, but more optimization is needed for accurate location of the geographical locations.

Considerations

The components do not function efficiently can be made a priority in the next releases to ensure the use cases are tackled with seriousness. This will reduce the testing and inspection steps and make the system more robust.

Glossary

Trails: national parks have walking area integrated inside the parks for the people to explore the different places inside the parks safely.

MVC: stands for Model View controller. It is a type of architecture utilized by platforms for distributed systems.

AddTrails: name of one of the components of GPS Trails

MySQL: A type of database management system for storing and managing large scale data.

Mapbox: A map generating application for locating the national parks and trails inside these. parks.

References / Bibliography

[1] GPS Trails Final Report, Chris Lenell, Deonvell Shed, Luke Wittenkeller, and Fade Abdeljaber

[1] J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012.

[2] A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013.

[3] Robertson and Robertson, Mastering the Requirements Process.

[4] M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.

Index

No index entries found.