**CHAPTER 1**

**INTRODUCTION**

* 1. **PROBLEM DEFINITION**

Free code camp is open source community that helps you learn to code and you can work through self-paced coding challenges, build projects and earn certificates.Freecodecamp.com is a website is developed by ‘Quincy Larison’ to help others to learn.

GitHub is a web-based [Git](https://en.wikipedia.org/wiki/Git) or [version control repository](https://en.wikipedia.org/wiki/Repository_(version_control)) and [Internet hosting service](https://en.wikipedia.org/wiki/Internet_hosting_service). It provides [access control](https://en.wikipedia.org/wiki/Access_control) and several collaboration features such as [bug tracking](https://en.wikipedia.org/wiki/Bug_tracking_system), [feature requests](https://en.wikipedia.org/wiki/Software_feature), [task management](https://en.wikipedia.org/wiki/Task_management), and [wikis](https://en.wikipedia.org/wiki/Wiki) for every project.

Camperbot for KGISL meets up as per existing system discussed earlier gives only the names and points. Problem is tracking the specific set of user’s activity from the gitter room.

Problem here is monthly data, yearly data, highest rank, total number of challenges solved, overall statistics, maximum problem solver is not displayed, and excluder option is not available. Only limited information is provided in the existing system. This project has got these problems to overcome and make it more informative one.

**1.2 OBJECTIVE OF THE PROJECT**

The main objective of the project is to track the student activity based on the scores. This will help the students also understand their statistics among all, which in turn help the students improve their Coding skills. It also provides useful (URL) links to open free code camp directly from their gitter accounts without having to open a new window or tab.

The graphical description is the most effective and understandable form of presentation. Any person with or without the knowledge of codes will understand it quickly. Also it’s very simple, responsive and elegant. We will also be able to track the students who are inactive for a very long time in the camp. Ranks based on the score are also being provided which will bring lots of interest and enthusiasm while working with the Free Code Camp.

**1.3 SIGNIFICANCE OF THE PROJECT**

The project has its particular significance; the pictorial representation of the ranks of the campers will be posted in a very elegant way for the people who wish to view it.

It highly authenticated and access is given only for the requested users who hold a GitHub account. Also there is place given to add and remove users who are inactive.

**1.4 OUTLINE OF THE PROJECT**

The project is a web application which gives an overall statics of the problems solved by the campers (Students) through FreeCodeCamp. The other such modules which give detailed information about the camper’s. This application’s main outline is to track the camper’s activity based on the problem count.

**CHAPTER 2**

**LITERATURE REVIEW**

**2.1 FREECODECAMP WEBSITE**

Free code camp is open source community that helps you learn to code and you can work through self-paced coding challenges, build projects and earn certificates.**Freecodecamp.com** is a website is developed by **‘Quincy Larison’** to help others to learn. This helps to the non-profit organizations through developing projects for the organization by the campers. Organizationcan built a community around Freecodecamp to help students to learn more about web technologies.

Beginning with tutorials that introduce students to HTML, CSS, and JavaScript, students’ progress to project assignments that they must complete either alone or in pairs. Upon completion of all project tasks, students are partnered with other nonprofits to build web applications, giving the students practical development experience.

Free code camp has their own API for sharing their details. It is integrated in the gitter.im chat application. Gitter is where developers come to talk .It is an instant messaging and chat room syatem for developers and users of github repositories.

**2.2 GITHUB WEBSITE:**

**GitHub** is a web-based [Git](https://en.wikipedia.org/wiki/Git) or [version control repository](https://en.wikipedia.org/wiki/Repository_(version_control)) and [Internet hosting service](https://en.wikipedia.org/wiki/Internet_hosting_service). It offers all of the [distributed version control](https://en.wikipedia.org/wiki/Distributed_version_control) and [source code management](https://en.wikipedia.org/wiki/Source_code_management) (SCM) functionality of Git as well as adding its own features. It provides [access control](https://en.wikipedia.org/wiki/Access_control) and several collaboration features such as [bug tracking](https://en.wikipedia.org/wiki/Bug_tracking_system), [feature requests](https://en.wikipedia.org/wiki/Software_feature), [task management](https://en.wikipedia.org/wiki/Task_management), and [wikis](https://en.wikipedia.org/wiki/Wiki) for every project. Github offers both plans for private and free [repositories](https://en.wikipedia.org/wiki/Repository_(version_control)) on the same account which are commonly used to host [open-source](https://en.wikipedia.org/wiki/Open-source) software projects.

Projects on Github can be accessed and manipulated using the standard Git command-line interface and all of the standard Git commands work with it. Github also allows registered and non-registered users to browse public repositories on the site. Multiple desktop clients and Git [plugins](https://en.wikipedia.org/wiki/Plug-in_(computing)) have also been created by Github and other third parties that integrate with the platform.

The site provides [social networking](https://en.wikipedia.org/wiki/Social_networking)-like functions such as feeds, followers, wikis (using [wiki software](https://en.wikipedia.org/wiki/Wiki_software) called [Gollum](https://en.wikipedia.org/wiki/Gollum_Wiki)) and a [social network graph](https://en.wikipedia.org/wiki/Collaboration_graph) to display how developers work on their versions ("[forks](https://en.wikipedia.org/wiki/Fork_(software_development))") of a repository and what fork (and branch within that fork) is newest.

A user must create an account in order to contribute content to the site, but public repositories can be browsed and downloaded by anyone. With a registered user account, users are able to discuss, manage, create repositories, submit contributions to others' repositories, and [review changes to code](https://en.wikipedia.org/wiki/Code_review).

**Gitter** is an [instant messaging](https://en.wikipedia.org/wiki/Instant_messaging) and chat room system for developers and users of [GitHub](https://en.wikipedia.org/wiki/GitHub) repositories. Gitter is provided as [software-as-a-service](https://en.wikipedia.org/wiki/Software-as-a-service), with a free option providing all basic features and the ability to create a single private chat room, and paid subscription options for individuals and organisations, which allows them to create arbitrary numbers of private chat rooms.

**CHAPTER 3**

**SYSTEM SPECIFICATION**

**3.1 HARDWARE REQUIREMENTS**

The components of the computer such as electrical, electronic and mechanical units are known as the hardware of computer. This input, output unit and central processing unit (CPU), are called as hardware. Thus hardware is the equipment involved in the functioning of a computer.

Processor : Intel Core Duo 2.0 Ghz or more

RAM : 512 MB or More

Harddisk : 40GB or More

Monitor : 15’CRT or LCD monitor

Keyboard : Normal or Multimedia

Mouse : Compatible mouse

**3.2 SOFTWARE REQUIREMENTS**

It deals with defining software resource requirements and prerequisits that needed to be installed on a computer to provide optimal functioning of an application. These requirements are prerequisits are generally not included in the softwear installation of package and needed to be installed seppartaely before the softwear is installed.

Programming Language : C#.net

Operating System : Any windows versions release after

windows XP ,Ubuntu ,MAC OS

Front-end : HTML, Bootstrap, CSS, JQuery and

JavaScript

Back-end : MySQL

Server used : Apache Server,IIS server

**3.2.1 FRONT END**

The front end is designed using C#.net framework which includes other Web Technologies like HTML, Bootstrap, CSS and JavaScript for effective designing.

**3.2.1.1 C#.NET**

C# is a simple, modern, general-purpose, object-oriented programming language developed by Microsoft within its .NET framework. Microsoft's aim is to facilitate the exchange of information and services over the Web, and to enable developers to build highly [portable](http://searchstorage.techtarget.com/definition/portability) applications. C# simplifies programming through its use of Extensible Markup Language ([XML](http://searchsoa.techtarget.com/definition/XML)) and Simple Object Access Protocol ([SOAP](http://searchsoa.techtarget.com/definition/SOAP)) which allow access to a programming [object](http://searchsoa.techtarget.com/definition/object) or [method](http://searchcio-midmarket.techtarget.com/definition/method) without requiring the programmer to write additional code for each step. Because programmers can build on existing code, rather than repeatedly duplicating it, C# is expected to make it faster and less expensive to get new products and services to market.

A programming infrastructure created by Microsoft for building, deploying, and running applications and services that use .**NET** technologies, such as desktop applications and Web services. The .**NET Framework** contains three major parts: the Common Language Runtime (**CLR**), the **Framework** Class Library and ASP.NET.

Microsoft .NET is a platform for developing “managed” software that runs primarily on Microsoft Windows. Programmers produce software by combining their own source code with .**NET** Framework and other libraries. . **NET** Framework is intended to be used by most new applications created for the Windows platform. Microsoft also produces an integrated development environment largely for .**NET** software called Visual Studio.

**3.2.1.2 HTML**

HTML is a computer language devised to allow website creation. These websites can then be viewed by anyone else connected to the Internet.

The definition of HTML is Hypertext Markup Language.

* Hypertext is the method by which you move around on the web — by clicking on special text called hyperlinks which bring you to the next page. The fact that it is hyper just means it is not linear — i.e. you can go to any place on the Internet whenever you want by clicking on links — there is no set order to do things in.
* Mark up is what HTML tags do to the text inside them. They mark it as a certain type of text HTML is a Language, as it has code-words and syntax like any other language.

The advantages are

* HTML is highly flexible
* HTML is supported on almost browser
* HTML is user friendly
* HTML is an open technology
* HTML is consistent and efficient
* HTML bids the superlative medium for its users
* HTML is easily understandable and does not require any training
* HTML is designed with a feature of interaction between the WebPages, which makes it effective
* HTML provides search engine compatible pages
* HTML is easier to maintain and update any site
* HTML does not involve strain on the servers
* For HTML web pages, it takes less time to load the web pages
* HTML validation is another important key factor which increases the web accessibility
* HTML webpage’s look and feel attracts larger masses to visit the website.

**3.2.1.3 JAVASCRIPT**

**JavaScript** is a dynamic computer programming language that is run by most modern browsers. It supports object-oriented programming and procedural programming. It can be used to control web pages on the client side of the browser, server-side programs, and even mobile applications. This language is used commonly in combination with **HTML**, **CSS**, and **AJAX**.

The benefits of Javascript are

* JavaScript is executed on the client side. This means that the code is executed on the user's processor instead of the web server thus saving bandwidth and strain on the web server.
* The JavaScript language is relatively easy to learn and comprises of syntax that is close to English. It uses the DOM model that provides plenty of prewritten functionality to the various objects on pages making it a breeze to develop a script to solve a custom purpose.
* The code is executed on the user's computer, results and processing is completed almost instantly depending on the task (tasks in JavaScript on web pages are usually simple so as to prevent being a memory hog) as it does not need to be processed in the site's web server and sent back to the user consuming local as well as server bandwidth.
* Third party add-ons like Grease monkey enable JavaScript developers to write snippets of JavaScript which can execute on desired web pages to extend its functionality. If you use a website and require a certain feature to be included, you can write it yourself and use an add-on like Grease monkey to implement it on the web page.

**3.2.2 BACK END**

The back end is designed using MySQL, whose primary function is to store data securely and retrieve it later, as requested by other software applications.

**3.2.2.1 MYSQL**

MySQL is the most widely used relational database management system (RDBMS) and is a centre component of the **LAMP** open source web application software stack. LAMP is an acronym for Linux, Apache, MySQL and Perl/Python/PHP. It is commonly used open source **RDBMS**, for proprietary used, several paid editions are available and offer additional functionality.

MySQL is an open-source database so you don't have to pay a single penny to use it. MySQL is a very powerful program so it can handle a large set of functionality of the most expensive and powerful database packages.

MySQL is customizable because it is an open source database and the open-source GPL license facilitates programmers to modify the SQL software according to their own specific environment. MySQL is quicker than other databases so it can work well even with the large data set.

MySQL supports many operating systems with many languages like PHP, PERL, C, C++, JAVA, etc. MySQL uses a standard form of the well-known SQL data language. MySQL is very friendly with PHP, the most popular language for web development.

MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).

The features of mysql are

The following list shows the most important properties of MySQL. This section is directed to the reader who already has some knowledge of relational databases.

In Relational Database Management System (RDBMS), MySQL is a relational database management system.

EASY TO USE: MySQL is easy to use. You have to get only the basic knowledge of SQL. You can build and interact with MySQL with only a few simple SQL statements.MYSQL is a open source

IT IS SECURE: MySQL consist of a solid data security layer that protects sensitive data from intruders. Passwords are encrypted in MySQL.

CLIENT/ SERVER ARCHITECTURE: MySQL follows a client /server architecture. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they query data, save changes, etc.

FREE TO DOWNLOAD: MySQL is free to use and you can download it from MySQL official website.

IT IS SCALABLE: MySQL can handle almost any amount of data, up to as much as 50 million rows or more. The default file size limit is about 4 GB. However, you can increase this number to a theoretical limit of 8 TB of data.

COMPATIBLE ON MANY OPERATING SYSTEMS: MySQL is compatible to run on many operating systems, like Novell NetWare, Windows\* Linux\*, many varieties of UNIX\* (such as Sun\* Solaris\*, AIX, and DEC\* UNIX), OS/2, FreeBSD\*, and others. MySQL also provides a facility that the clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).

ALLOWS ROLL-BACK: MySQL allows transactions to be rolled back, commit and crash recovery.

HIGH PERFORMANCE: MySQL is faster, more reliable and cheaper because of its unique storage engine architecture.

HIGH FLEXIBILITY: MySQL supports a large number of embedded applications which makes MySQL very flexible.

HIGH PRODUCTIVITY: MySQL uses Triggers, Stored procedures and views which allows the developer to give a higher productivity.

HIGH PERFORMANCE: MySQL database gives maximum performance while fetching the data from any of the web application.

MySQL supports as its database language -- as its name suggests – SQL (Structured Query Language). SQL is a standardized language for querying and updating data and for the administration of a database. There are several SQL dialects (about as many as there are database systems). MySQL adheres to the current SQL standard (at the moment SQL:2003), although with significant restrictions and a large number of extensions.

**CHAPTER 4**

**SYSTEM ANALYSIS**

System analysis is a problem solving technique that decomposes a system into component pieces of purpose of studying how well those component parts work and interact to accomplish their purpose the following chapter provides the detail description of the existing system. It also provides an overview of the proposed system and feasibility of the FCC status viewer.

**4.1 EXISTING SYSTEM**

The existing system has no status viewer for free code camp.Only it helps to get the brownie points of all users and display their avatar profile. We Can’t get minimum problem solver list, maximum problem solver. And there is no options for comparing the user progress based on Day, Month, Year.Since the existing system is not user friendly.

So we introduce an interactive and Responsive FCC status viewer. Existing system is <http://kgashok.github.io/elm-simple-json-decoding/freecodecamp.html>.

**4.2 PROPOSED SYSTEM**

Camperbot FCC status viewer is an open source web application which is developed to track the students by university. It motivates the students to improve themselves in web technologies. To overcome the disadvantages of the existing system such as individual camper report, user excluder option, reports generated based on month and year. It is a user friendly application. One who uses this application finds it easy to understand what it is all about.

**4.2.1 ADVANTAGES**

* It helps to find the count of the campers.
* It helps to find the total problem solved by the campers.
* It helps to find the rank of the campers.
* Daily activities report of the camper/community.
* It helps to find the maximum problem solver and minimum problem solver.
* It helps to exclude the user.ie user with higher rank such as Mrs.Ashok, Quincy larison, Ms.Ramya
* GitHub Authentication is provided to increase better security performance.
* Overall problems solved by all campers will be displayed on a regular basis.

**4.3 FEASIBILITY STUDY**

Feasibility study is a high level capsule version of the entire process intended to answer a number of questions like: What is the problem? Is there any feasible solution to the given problem? Is the problem even worth solving? Feasibility study is conducted once the problem clearly understood. Feasibility study is necessary to determine that the proposed system is feasible by considering the technical, operational, and economical factors. By having a detailed feasibility study the management will have a clear-cut view of the proposed system.

Feasibility study encompasses the following things

* Economical Feasibility
* Operational feasibility
* Technical Feasibility

In this phase, we study the feasibility of all proposed systems, and pick the best feasible solution for the problem. The feasibility is studied based on three main factors as follows.

**4.3.1 ECONOMICAL FEASIBILITY**

In this step, we verify which proposal is more economical. We compare the financial benefits of the new system with the investment. The new system is economically feasible only when the financial benefits are more than the investments an expenditure. Economical Feasibility determines whether the project goal can be within the resource limits allocated to it or not. It must determine whether it is worthwhile to process with the entire project or whether the benefits obtained from the new system are not worth the costs. Financial benefits must be equal or exceed the costs. In this issue, we should consider,

* The cost to conduct a full system investigation.
* The cost of h/w and s/w for the class of application being considered.
* The development tool.
* The cost of maintenance etc.,

Our project is economically feasible because the cost of development is very minimal when compared to financial benefits of the application.

**4.3.2 OPERATION FEASIBILITY**

In this step, we verify different operational factors of the proposed systems like man-power, time etc., whichever solution uses less operational resources, is the best operationally feasible solution. The solution should also be operationally satisfied user objectives could be fitted into the current system operation.

* The methods of processing and presentation are completely accepted but the clients since they can meet all user requirements.
* The clients have been involved in the planning and development of the system.
* The proposed system will not cause any problem under any circumstances.

Our project is operationally feasible because the requirements and personnel requirements are satisfied. We are a team of 4 members and we worked on this project for three working months.

**4.3.3 TECHNICAL FEASIBILITY**

In this step, we verify whether the proposed systems are technically feasible or not, i.e., all the technologies required to develop the system are available readily or not.

Technical Feasibility determines whether the organization has the technology and skills necessary to carry out the project and how this should be obtained. The system can be feasible because of the following grounds.

* All necessary technology exists to develop the system.
* This system is too flexible and it can be expanded further.
* This system can give guarantees of accuracy, ease of use, reliability and the data security.
* This system can give instant response to inquire.

Our project is technically feasible because, all the technology needed for our project is readily available.

**CHAPTER 5**

**PROJECT DESCRIPTION**

**5.1 OVERVIEW OF THE PROJECT**

FCC Status Viewer is a web application which is designed to track the camper’s activities. This application displays the student activities graphically along with user’s details and their points.

**5.2 MODULE DESCRIPTION**

Problem is tracking the specific set of users activity from the gitter room. This data will stored in a specific database to generate reports.

**5.2.1 USER RANK VIEWER**

This module used to display all the campers Based on their brownie points with their avatars and the link will be displayed. That link redirects the user to the specific users profile page.

**5.2.2 USER PROFILE VIEWER**

This module displays the each user profile and activity in the free code camp. This module will have the graphs and charts which display the user Activities.

**5.2.3 USER EXCLUDER**

This module used to exclude the users those who flagged as dishonesty and inactive campers from the all counts. This module is allowed for some specific user and this will hardcoded in the webpage.

**5.2.4 TIME BASED ACTIVITY VIEWER**

This module used to display the campers activity based on the time interval. User can select the time interval for the graph. The graph is generated based on the data.

**5.2.5 LIVE UPDATES**

It will update the user name and their scores. This module is Like a Live cricket score update.

**5.3 DATA FLOW DIAGRAM**

Data flow diagram is the 2-D diagram that explains how data is processed and transferred in a system. The graphical depiction defines each source of data and how it interacts with the other sources to reach common output.

Activity and the title for the activity should be placed inside the rectangle. Data flow Diagram (DFD) is an important technique for modeling a systems high level detail by showing how input data is transformed to output results through a sequence of functional transformations. DFDs reveal relationships among and between the various components in a program or system. DFDs consist of four major components: processes, data stores and data flow.

**Figure 5.1 Data Flow Diagram Level 0**

**Figure 5.2 Data Flow Diagram level 1**

**5.4 ER DIAGRAM**

ER Diagram is called as an entity-relationship (ER) diagram, is a graphical representation of entities and their relationship to each other, typically used in computing in regard to the organization of data within databases or information systems. An entity is a piece of data object or concept about which data is stored.

**Figure 5.3 ER Diagram**

**5.5 DATABASE DESIGN**

USER TABLE**:** User table contains whole details of user such as user Id , user name, date of join, user excluder, user excluder date, avatar url. User Id is a unique id and it is the primary key. Username is the name of user. Date of join field contains the joining date information of each user. Avatar url is the link of user profile photo. User excluder date field contain the date when the user is excluded.

**Table 5.3.1 User Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **field Name** | **Type** | **Size** | **Description** | **Constraint** |
| Id | Varchar | 50 | User ID | Primary Key |
| Name | Varchar | 100 | Display Name | Not Null |
| Doj | Date |  | Date of Join | Not Null |
| Username | Varchar | 50 | User | Not Null |
| url | Varchar | 50 | Image URL | Not Null |
| Excluder | Varchar | 1 | User excluding |  |
| Exclude\_date | Date |  | Excluding date |  |

DAILY UPDATE TABLE: User details will be updated daily inorder to display the user information in graph. User id is used as a foreign key which refer the user id in user table.

**Table 5.3.2 Daily Update Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **field Name** | **Type** | **Size** | **Description** | **Constraint** |
| Date | Date |  | Record created date | Not Null |
| user\_ID | Varchar | 50 | User ID from user table | Foreign Key |
| Points | Integer |  | Brownie Points | Not Null |
| Rank | Integer |  | Rank of the user | Not Null |

DAILY COUNT TABLE: This table is maintained to calculate the user points individually and also for displaying the total count of all user. This information is displayed in the dashboard.

**Table 5.3.3 Daily Count Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **field Name** | **Type** | **Size** | **Description** | **Constraint** |
| Date | Date |  | Record Date | Primary Key, Not Null |
| points\_Count | Integer |  | Total points | Not Null |
| user\_count | Integer |  | Total campers | Not Null |

**5.6 INPUT DESIGN**

Input design is a part of overall system design. It is the process of converting an external user oriented format. It is an input system into a machine oriented format. It is a part of overall system design, which requires very careful attention. In forms there is a variety of controls for designing user data entry screens. Data block controls provide easy access to records stored in the backend database.

The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To achieve the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

In this project, there are five pages. There are

* User Excluder
* User Profile
* Time Activity
* Rank Viewer
* Message Sender

**5.7 OUTPUT DESIGN**

Outputs from computer systems are required primarily to communicate the result of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs in general are:

* External Output, whose destination is outside the organization,
* Internal Output whose destination is within organization and they are the user’s main interface with the computer.
* Operational output whose use is purely within the computer department
* Interface outputs, which involve the user in communicating directly.

In this project, there are three output designs. There are

* Rank Report
* Profile Report
* Maximum Problem Solver

**CHAPTER 6**

**SYSTEM TESTING**

**FCC status viewer** was developed and the entire system was tested to make sure that the system works as per the requirements as stated in the high level specification. Verification and validation for the system where performed to make sure the system does not contain any bugs.

**6.1. TESTING METHODS**

The entire system was divided into separate or individual modules or sub modules (units). Using various levels of testing such as unit test, integration test, output test, acceptance test the system was completely tested and was found bug free. Different testing techniques such as **black box testing, white box testing,functional and load testing** was also carried out to make sure the system functions as per user requirements.

**6.2 TYPES OF TESTING**

There are numerous types of system testing that can be done as part of the system testing and deliver process. The lists of test types are explained below:

**6.2.1 UNIT TESTING**

Unit testing focuses verification effort on the smallest unit of software design i.e. the module. In this project I have used index page for unit testing. Testing has been done to verify the maximum problem solver and first three rankers from the campsite room are displayed in the carousel or not.

**6.2.2 INTEGRATION TESTING**

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

Integration testing identifies problems that occur when units are combined. By using a test plan that requires you to test each unit and ensure the viability of each before combining units, you know that any errors discovered when combining units are likely related to the interface between units. This method reduces the number of possibilities to a far simpler level of analysis. Integration testing follows two approaches known as ‘Top-Down’ approach and ‘Bottom-Up’ approach.

**6.2.3 FUNCTIONAL TESTING**

Functional Testing is a testing technique that is used to test the features/functionality of the system or Software, should cover all the scenarios including failure paths and boundary cases. Functions are tested by feeding them input and examining the output. Functional testing usually describes what the system does.

Functional testing typically involves six steps

* The identification of functions that the software is expected to perform
* The creation of input data based on the function's specifications
* The determination of output based on the function's specifications
* The execution of the test case
* The comparison of actual and expected outputs
* To check whether the application works as per the customer need.

**6.2.4 STRESS TESTING**

A testing technique which evaluates a system or component at or beyond the limits of its specified requirements. This testing process is usually conducted by the performance engineer.

**6.2.5 ACCEPTANCE TESTING**

Acceptance testing is a testing technique performed to determine whether or not the software system has met the requirement specifications. The main purpose of this test is to evaluate the system's compliance with the business requirements and verify if it is has met the required criteria for delivery to end users. The acceptance test cases are executed against the test data or using an acceptance test script and then the results are compared with the expected ones.

**6.2.6 WHITE BOX TESTING**

White box testing is the detailed investigation of internal logic and structure of the code. White box testing is also a Glass box testing or open box testing. In this testing, by knowing the specific demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing is

* Flow graph notation
* Cyclometric complexity
* Deriving test cases
* Graph matrices control

**6.2.7 BLACK BOX TESTING**

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operations performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are

* Graph based testing methods
* Equivalence partitioning
* Boundary value analysis
* Comparison testing

**6.3 TESTING STRATEGY**

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically.

For software testing is a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics.

Testing begins at the module level and works “outward” towards the integration of the entire computer based system. Different testing techniques are appropriate at different points in time. Developer of the software and an independent test group conducts testing.

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically.

**6.4 TEST CASE**

A test case is a set of conditions or variables under which is testing will determine if a requirement upon an application is partially or fully satisfied. The sample test cases for the project are as follows.

`

**Table 6.1 Test Case**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test\_case\_Id** | **Description** | **Test Data** | **Expected Result** | **Actual Result** | **Result** |
| test\_index\_1 | Verify the correct data fetch from the dqatabase |  | index page should contain text "Total problem and total user count" | index page contains text "Total problem and total user count" | Pass |
|  | Verify the correct data fetch from the database | **33641** | total problem count should equivalent to 33641 | total problem count is equivalent to 33641 | Pass |
|  | Verify the correct data fetch from the database | **396** | total user count should equivalent to 396 | total user count is equivalent to 396 | Pass |
| test\_index\_2 | verify the maximum problem solver ist | **Dyson** | It should display the first Ranker as Dyson | Displays the first Ranker as Dyson | Pass |
|  | verify the maximum problem solver ist | **352** | It should display the first Ranker count as 352 | Displays the first Ranker count as 352 | Pass |
|  | verify the maximum problem solver ist |  | It should display the dyson's avatar from github | Displays the dyson's avatar from github | Pass |

**CHAPTER 7**

**SYSTEM IMPLEMENTATION**

Implementation is the stage of the project where the theoretical design is turned into a working system. At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts of the user department. If the implementation is not carefully planned a controlled it can cause chaos and confusion.

Implementation includes all those activities that take place to convert from the old system to the new one. The new system may be totally new, replacing an existing manual or automated system or it may be a major modification to an existing system. Proper implementation is essential to provide a reliable system to meet the organization requirements. Successful implementation may not guarantee improvement in the organization using the new system, but improper installation will prevent it.

The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after thorough testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system.

The most crucial stage is achieving a new successful system and giving confidence on the new system for the user that it will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover. The more complex the system being implemented, the more involved will be the system analysis and the design effort required just for implementation. The system implementation has three main aspects. They are education and training, system testing and changeover.

The implementation stage involves following tasks.

* Careful planning.
* Investigation of system and constraints.
* Design of methods to achieve the changeover.
* Training of the staff in the changeover phase.
* Evaluation of the changeover method.

The method of implementation and the time scale to be adopted are found out initially. Next the system is tested properly and the same time users are trained in the new procedures.

The most crucial stage is achieving a new successful system and giving confidence on the new system for the user that it will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover. The more complex the system being implemented, the more involved will be the system analysis and the design effort required just for implementation. The system implementation has three main aspects. They are education and training, system testing and changeover.

The implementation stage involves following tasks.

* Careful planning.
* Investigation of system and constraints.
* Design of methods to achieve the changeover.
* Training of the staff in the changeover phase.
* Evaluation of the changeover method.

The method of implementation and the time scale to be adopted are found out initially. Next the system is tested properly and the same time users are trained in the new procedures.

**7.1 IMPLEMENTATION PROCEDURES**

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended users and the operation of the system. In many organizations some one who will not be operating it, will commission the software development project. In the initial stage, they doubt about the software but we have to ensure that the resistance does not build up as one has to make sure that

* The active user must be aware of the benefits of using the system.
* Their confidence in the software is built up.
* Proper guidance is imparted to the user so that he is comfortable in using the application.

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not up running on the server, the actual processes will not take place.

**CHAPTER 8**

**CONCLUSION & FUTURE ENHANCEMENTS**

**8.1 CONCLUSION:**

The project has been appreciated by all the users in the organization. It is easy to use. User friendly screens are provided for faster access. UI is designed in an efficient way that it can be used in mobile phones also. It is compatible with all working devices with the use of Bootstrap Framework. The usage of software increases the efficiency, decreases the effort of viewing the details of every camper. This application gives every details of the camper present in KGiSL/campsite gitter room. It has been thoroughly tested and implemented.

**8.2 FUTURE ENHANCEMENT:**

The future enhancements are as follows

* Separate login will be provided for users and admin.
* The current version is for admin only.
* This version has got user excluder option.
* In the future versions even end users will be able to see their details.
* End users will not have user excluder option.

**CHAPTER 9**

**APPENDIX**

**9.1 SOURCE CODE**

**Dashboard controller code:**

using Octokit;

using System;

using System.Collections.Generic;

using System.Threading.Tasks;

using System.Web;

using System.Web.Mvc;

using System.Web.Security;

namespace CamperBot\_FCC\_Status\_Viewer.Controllers

{

public class GeneralController : Controller

{

const string clientId = "28bfffe90ee14b851674";

private const string clientSecret = "049d738c88dc3bb0a352eff6663a7c03b44311b0";

readonly GitHubClient client =

new GitHubClient(new ProductHeaderValue("Camperbot-FCC-Status-Viewer"));

// GET: General/Dashboard

// List the user current month and some other status

public async Task<ActionResult> Dashboard()

{

var accessToken = Session["OAuthToken"] as string;

if (accessToken != null)

{

// This allows the client to make requests to the GitHub API on the user's behalf

// without ever having the user's OAuth credentials.

client.Credentials = new Credentials(accessToken);

}

try

{

var user = await client.User.Current();

ViewBag.AvatarUrl = user.AvatarUrl;

ViewBag.UserName = user.Name;

List<Models.current\_month\_update\_list> currentMonthUpdateList = CamperBot\_FCC\_Status\_Viewer.Models.DashboardChartModel.FetchCurrentMonthStatus();

return View(currentMonthUpdateList);

}

catch (AuthorizationException)

{

// Either the accessToken is null or it's invalid. This redirects

// to the GitHub OAuth login page. That page will redirect back to the

// Authorize action.

return Redirect(GetOauthLoginUrl());

}

}

// GET: General/UserRankViewer

// List the user in rank order

public async Task<ActionResult> UserRankViewer()

{

var accessToken = Session["OAuthToken"] as string;

if (accessToken != null)

{

// This allows the client to make requests to the GitHub API on the user's behalf

// without ever having the user's OAuth credentials.

client.Credentials = new Credentials(accessToken);

}

try

{

var user = await client.User.Current();

ViewBag.AvatarUrl = user.AvatarUrl;

ViewBag.UserName = user.Name;

//CamperBot\_FCC\_Status\_Viewer.Models = namespace

//UserRankViewerModel = class name

//DatabaseConnection() = method in that class

List<Models.rank\_list> camperRankList = CamperBot\_FCC\_Status\_Viewer.Models.UserRankViewerModel.GenerateRankList();

return View(camperRankList);

}

catch (AuthorizationException)

{

// Either the accessToken is null or it's invalid. This redirects

// to the GitHub OAuth login page. That page will redirect back to the

// Authorize action.

return Redirect(GetOauthLoginUrl());

}

}

// GET: General/AllUserDetailsDatabase

// Fetch all data from user table and generate a list

public async Task<ActionResult> AllUserDetailsDatabase()

{

var accessToken = Session["OAuthToken"] as string;

if (accessToken != null)

{

// This allows the client to make requests to the GitHub API on the user's behalf

// without ever having the user's OAuth credentials.

client.Credentials = new Credentials(accessToken);

}

try

{

var user = await client.User.Current();

ViewBag.AvatarUrl = user.AvatarUrl;

ViewBag.UserName = user.Name;

List<Models.database\_list> allUserDetailList = CamperBot\_FCC\_Status\_Viewer.Models.FetchUserDetailsTableModel.FetchUserDetailsTable();

return View(allUserDetailList);

}

catch (AuthorizationException)

{

// Either the accessToken is null or it's invalid. This redirects

// to the GitHub OAuth login page. That page will redirect back to the

// Authorize action.

return Redirect(GetOauthLoginUrl());

}

}

// GET: General/UserProfileViewer

// Fetch the complete data of particular user and generate a list

public async Task<ActionResult> UserProfileViewer(string userId)

{

var accessToken = Session["OAuthToken"] as string;

if (accessToken != null)

{

// This allows the client to make requests to the GitHub API on the user's behalf

// without ever having the user's OAuth credentials.

client.Credentials = new Credentials(accessToken);

}

try

{

var user = await client.User.Current();

ViewBag.AvatarUrl = user.AvatarUrl;

ViewBag.UserName = user.Name;

List<Models.user\_profile\_details> UserProfileDetails = CamperBot\_FCC\_Status\_Viewer.Models.UserProfileModel.FetchUserProfileDetails(userId);

return View(UserProfileDetails);

}

catch (AuthorizationException)

{

// Either the accessToken is null or it's invalid. This redirects

// to the GitHub OAuth login page. That page will redirect back to the

// Authorize action.

return Redirect(GetOauthLoginUrl());

}

}

// This is the Callback URL that the GitHub OAuth Login page will redirect back to.

public async Task<ActionResult> Authorize(string code, string state)

{

if (!String.IsNullOrEmpty(code))

{

var expectedState = Session["CSRF:State"] as string;

if (state != expectedState) throw new InvalidOperationException("SECURITY FAIL!");

Session["CSRF:State"] = null;

var token = await client.Oauth.CreateAccessToken(

new OauthTokenRequest(clientId, clientSecret, code)

{

RedirectUri = new Uri("http://localhost:50568/general/authorize")

});

Session["OAuthToken"] = token.AccessToken;

}

return RedirectToAction("dashboard");

}

private string GetOauthLoginUrl()

{

string csrf = Membership.GeneratePassword(24, 1);

Session["CSRF:State"] = csrf;

// 1. Redirect users to request GitHub access

var request = new OauthLoginRequest(clientId)

{

Scopes = { "user", "notifications" },

State = csrf

};

var oauthLoginUrl = client.Oauth.GetGitHubLoginUrl(request);

return oauthLoginUrl.ToString();

}

}

}

**MODEL:**

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace CamperBot\_FCC\_Status\_Viewer.Models

{

public class DashboardChartModel

{

public static List<current\_month\_update\_list> FetchCurrentMonthStatus()

{

List<current\_month\_update\_list> tempList = new List<current\_month\_update\_list>();

// Database Connection

string connectionString = "server=localhost;user=root;database=camperbot;port=3306;password=;";

MySqlConnection connection = new MySqlConnection(connectionString);

try

{

connection.Open();

// Select Total points from daily update table

string sqlQuery = "SELECT u\_date, pts\_count, u\_count FROM daily\_count";

MySqlCommand cmd = new MySqlCommand(sqlQuery, connection);

// Generate the list with fetched data

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

current\_month\_update\_list temp\_data = new current\_month\_update\_list()

{

u\_date = rdr["u\_date"].ToString(),

pts\_count = rdr["pts\_count"].ToString(),

u\_count = rdr["u\_count"].ToString()

};

tempList.Add(temp\_data);

}

rdr.Close();

//Find the Inactive user and append to same list

// Find count of Inactive user

string sqlQuery1 = "SELECT COUNT(\*) i\_user FROM user\_rank\_list WHERE points = 0 ;";

MySqlCommand cmd1 = new MySqlCommand(sqlQuery1, connection);

// Generate the list with fetched data

MySqlDataReader rdr1 = cmd1.ExecuteReader();

while (rdr1.Read())

{

current\_month\_update\_list temp\_data = new current\_month\_update\_list()

{

u\_date = "InactiveUser",

pts\_count = "InactiveUser",

u\_count = rdr1["i\_user"].ToString()

};

tempList.Add(temp\_data);

}

rdr1.Close();

//end find inactive user

}

catch (Exception ex)

{

}

connection.Close();

return (tempList);

}

}

// Template for list

public class current\_month\_update\_list

{

public string u\_date { get; set; }

public string pts\_count { get; set; }

public string u\_count { get; set; }

}

**Rank module code:**

**MODEL:**

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace CamperBot\_FCC\_Status\_Viewer.Models

{

public class UserRankViewerModel

{

public static List<rank\_list> GenerateRankList()

{

List<rank\_list> tempList = new List<rank\_list>();

// Database Connection

string connectionString = "server=localhost;user=root;database=camperbot;port=3306;password=;";

MySqlConnection connection = new MySqlConnection(connectionString);

try

{

connection.Open();

// Select user based on rank order

string sqlQuery = "SELECT user.uid, user.name, user.uname, user.url, user\_rank\_list.points FROM user, user\_rank\_list WHERE user.uid = user\_rank\_list.uid order by user\_rank\_list.points DESC;";

MySqlCommand cmd = new MySqlCommand(sqlQuery, connection);

// Generate the list with fetched data

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

rank\_list temp\_data = new rank\_list()

{

uid = rdr["uid"].ToString(),

name = rdr["name"].ToString(),

uname = rdr["uname"].ToString(),

url = rdr["url"].ToString(),

points = rdr["points"].ToString()

};

tempList.Add(temp\_data);

}

rdr.Close();

}

catch (Exception ex)

{

}

connection.Close();

return (tempList);

}

}

// Template for list

public class rank\_list

{

public string uid { get; set; }

public string name { get; set; }

public string uname { get; set; }

public string url { get; set; }

public string points { get; set; }

}

}

**VIEW:**

@model List<CamperBot\_FCC\_Status\_Viewer.Models.rank\_list>

@{

Layout = "~/Views/Shared/\_Layout.cshtml";

var rank = 0;

var temp\_points = "";

var flag = 1;

}

<link rel="stylesheet" href="~/Custom/css/user-profiles-list-small.css">

<!-- User rank viewer -->

<div class="right\_col" role="main">

<div class="">

<div class="page-title">

<div class="title\_left">

<h3>CamperBot Leader Board</h3>

</div>

</div>

<div class="clearfix"></div>

<div class="col-md-12 col-sm-12 col-xs-12">

<div class="x\_panel">

<div class="x\_title">

<h2>User Rank List<small>campers</small></h2>

<div class="clearfix"></div>

</div>

<div class="x\_content">

<!--List the user based on rank-->

<ul id="rankList" class="user-profiles-list-small">

<!--template-->

<!--

<li>

<div class="user-avatar">

<a href="#">

<img src="~/Images/profile.jpg" width="48" alt="Profile of Mark Smith Peterson" />

</a>

</div>

<p class="user-name">

<a href="">Mark Smith Peterson</a>

<span>Rank : #1 | Point : 350</span>

</p>

<a class="delete" href="#"><i class="fa fa-plus"></i></a>

</li>

-->

<!--/template-->

@foreach (var item in Model)

{

if (@item.points != "0")

{

if (flag == 1)

{

temp\_points = @item.points;

flag = 2;

rank++;

}

else if (flag == 2)

{

if (temp\_points != @item.points)

{

temp\_points = @item.points;

rank++;

}

}

}

else

{

rank = 0;

}

<li>

<div class="user-avatar">

<a href = "UserProfileViewer?userId=@item.uid">

<img src="@item.url" width="48" alt="Profile Picture Or Avatar of Camper" />

</a>

</div>

<p class="user-name">

<a href="UserProfileViewer?userId=@item.uid">@item.name</a>

<span>Rank : #@rank | Point : @item.points</span>

</p>

<a class="delete" href="UserProfileViewer?userId=@item.uid"><i class="fa fa-plus"></i></a>

</li>

}

</ul>

<!--/List the user based on rank-->

</div>

</div>

</div>

</div>

</div>

<!-- /User rank viewer -->

**Live update code:**

var key = "ae28f23f134c4364ad45e7b7355cfa91c92038bb";

var arr = [];

var points = 0;

var html = '<table class="table" id="data"><thead><tr><th class="text-center">Avatar</th><th class="text-center">Name</th><th class="text-center">User Name</th><th class="text-center">Points</th></tr></thead><tbody>';

var sum = 0;

$(document).ready(function() {

var url = 'https://api.gitter.im/v1/rooms?access\_token=' + key;

var roomId = "";

var noOfUsers = 0;

$.ajax({

type: 'GET', url: url,

//data:data,

async: false,

dataType: 'json',

success: function(data) {

//Do stuff with the JSON data

for (var i = 0; i < data.length; i++) { //data.length

if (data[i]["name"] == 'kgisl/campsite') {

roomId = data[i]["id"];

noOfUsers = data[i]["userCount"];

break;

}

}

},

error: function(xhr, textStatus, errorThrown) {

points = 0;

}

});

var jsonData = [];

for (var i = 0; i < noOfUsers; i += 100) { //30){//noOfUsers

$.ajax({

type: 'GET',

url: 'https://api.gitter.im/v1/rooms/' + roomId + '/users?access\_token=' + key + '&skip=' + i + '&limit=100',

//data:data,

async: false,

dataType: 'json',

success: function(data) {

$.merge(jsonData, data);

//alert(jsonData);

},

error: function(xhr, textStatus, errorThrown) {

points = 0;

}

});

}

getData(jsonData);

});

$(document).ajaxStop(function() {

console.log('triggered');

sortTable();

});

function getData(jsonData) {

//alert(json["array"].length

var len = jsonData.length;

for (var i = 0; i < len; i++) { //len

if (jsonData[i]["username"] !== 'QuincyLarson') {

browniePointsFetcher(jsonData[i]["username"]);

arr.push({avatar: jsonData[i]["avatarUrlSmall"], avatar2: jsonData[i]["avatarUrlMedium"], name: jsonData[i]["displayName"], uname: jsonData[i]["username"], points: points});

}

}

var j = 0;

html += arr.map(function(a) {

j++;

return '<tr>' + dataFormatter(a.avatar, a.name, a.uname, a.points, a.avatar2) + '</tr>';

}).join('');

html += '</tbody></table>';

$("#data").html(html);

var a = $("#data").html();

$("#campers").html('<h2><span class="label label-info btn-success">Total Campers:- ' + j + '</span></h2>');

$("#totalProblems").html('<h2><span class="label label-info btn-success">Total Problems:- ' + sum + '</span></h2>');

}

function browniePointsFetcher(uname) {

var points = 0;

var url = 'https://www.freecodecamp.com/api/users/about?username=' + uname.toLowerCase();

$.ajax({

type: 'GET', url: url,

//data:data,

async: true,

dataType: 'json',

success: function(data) {

//Do stuff with the JSON data

points = data["about"]["browniePoints"];

$('#' + uname).html('<h2 class="pts">' + points + '</h2>');

sum += points;

$("#totalProblems").html('<h2><span class="label label-info">Total Problems:- ' + sum + '</span></h2>');

},

error: function(xhr, textStatus, errorThrown) {

points = 0;

$('#' + uname).html('<h2 class="pts">0</h2>');

}

});

}

function dataFormatter(image, name, uname, points, urlmedium) {

//alert(urlmedium);

var temp\_html = '<td>';

temp\_html += '<img src=' + image + ' class="img-thumbnail" width="100px" ></img></td>';

temp\_html += '<td>';

temp\_html += '<h3>' + name + '</h3></td>';

temp\_html += '<td>';

temp\_html += '<h3><a href="http://freecodecamp.com/' + uname + '" target="\_blank">' + uname + '</a></h3></td>';

temp\_html += '<td id=' + uname + ' class="points"></td>';

return temp\_html;

}

function sortTable() {

var rows = $('table tbody tr').get();

rows.sort(function(a, b) {

var x = parseInt($(a).children('td').eq(3).children().text());

var y = parseInt($(b).children('td').eq(3).children().text());

if (x < y)

return 1;

if (x > y)

return -1;

return 0;

});

$.each(rows, function(index, row) {

$('table').children('tbody').append(row);

});

}

**VIEW:**

@{

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<!-- Live update -->

<link rel="stylesheet" href="~/Custom/live\_resource/css/live.css">

<div class="right\_col" role="main">

<div class="">

<div class="page-title">

<div class="title\_left">

<h3>Live Update</h3>

</div>

</div>

<div class="clearfix"></div>

<div class="row">

<div class="col-md-12">

<div class="x\_panel">

<div class="x\_content">

<div class="row">

<div class="col-lg-4 col-md-6 col-sm-6 col-xs-12 profile\_details">

</div>

<div class="clearfix"></div>

<!--live page content-->

<div class="container-fluid well">

<div class="row text-center">

<div class="col-xs-12">

<h1><span class="label label-primary">Free Code Camp User Report</span></h1>

</div>

</div>

<div class="row text-center">

<div class="col-xs-6" id="campers"></div>

<div class="col-xs-6" id="totalProblems"></div>

</div>

<div class="row text-center">

<div class="col-xs-12" id="data"></div>

</div>

</div>

<!--/live page content-->

</div>

</div>

</div>

</div>

</div>

</div>

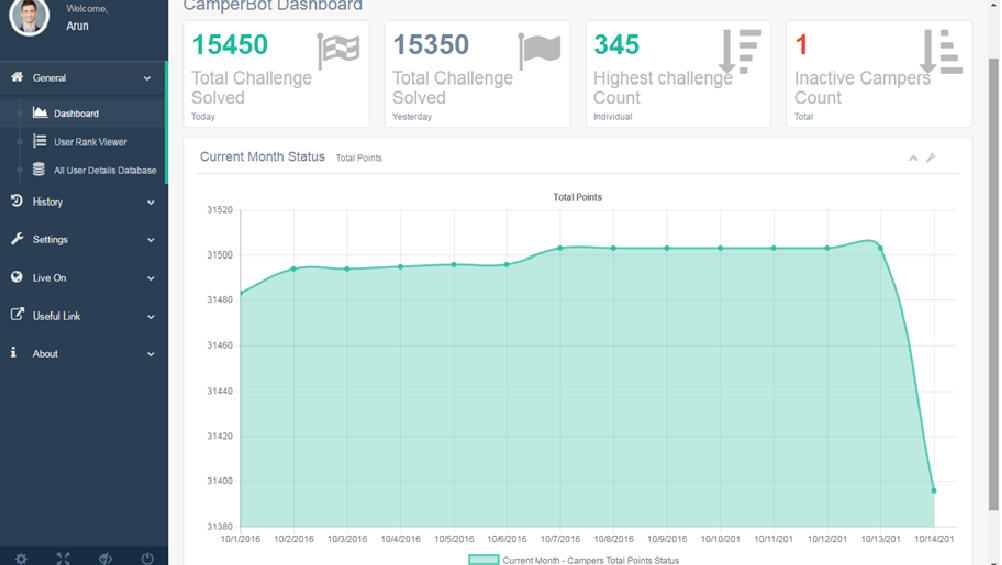
</div>

<script type="text/javascript" src="~/Custom/live\_resource/js/fccLiveUpdateBot.js"></script>

**9.2 SCREEN SHOTS**

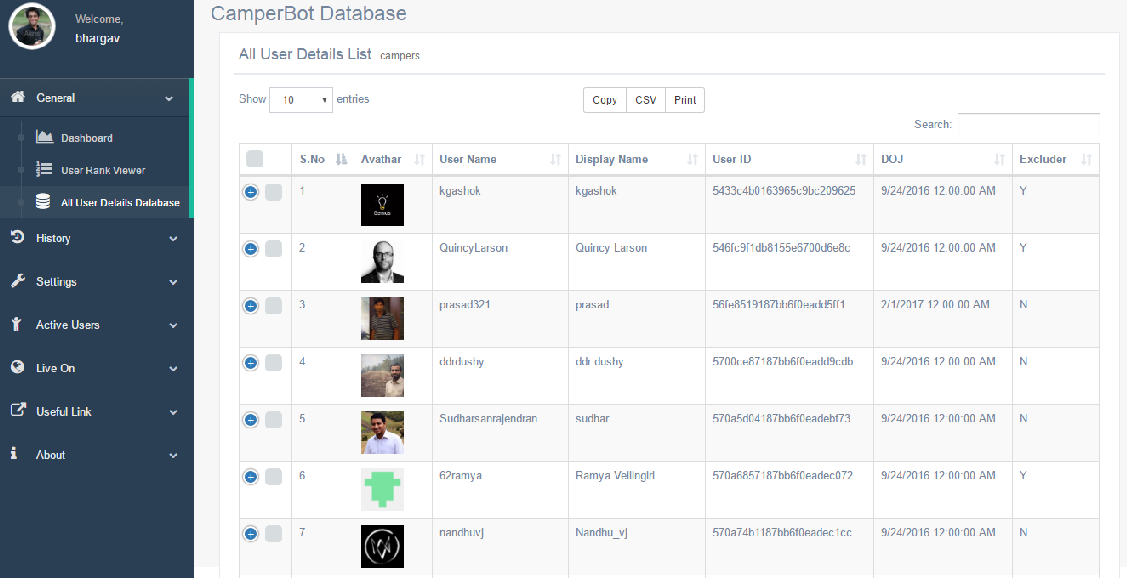
**9.2.1 DASHBOARD**

Dashboard is the first module, it mainly covers the application design .It includes the information such as, Total challenge solved by the individual user (includes current date and yesterday date ), Displays highest count solved by the user, Displays inactive user count , The current month status describes graphically total points acquired by the user.

****

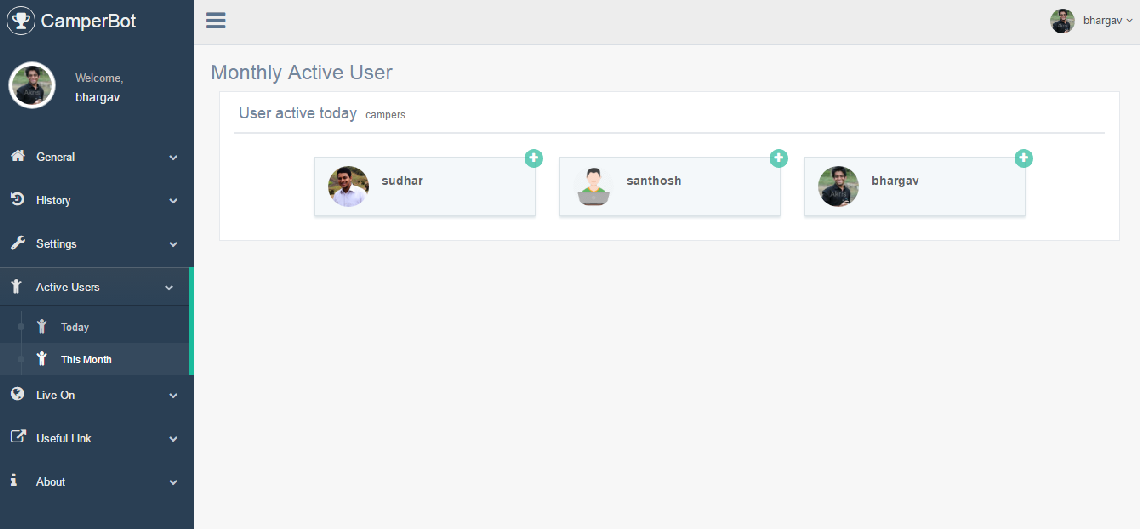
**9.2.2 DATABASE**

Database holds the whole information of user such as Avatar URL, User Name, User ID, Display Name, date of join, User Excluder.

****

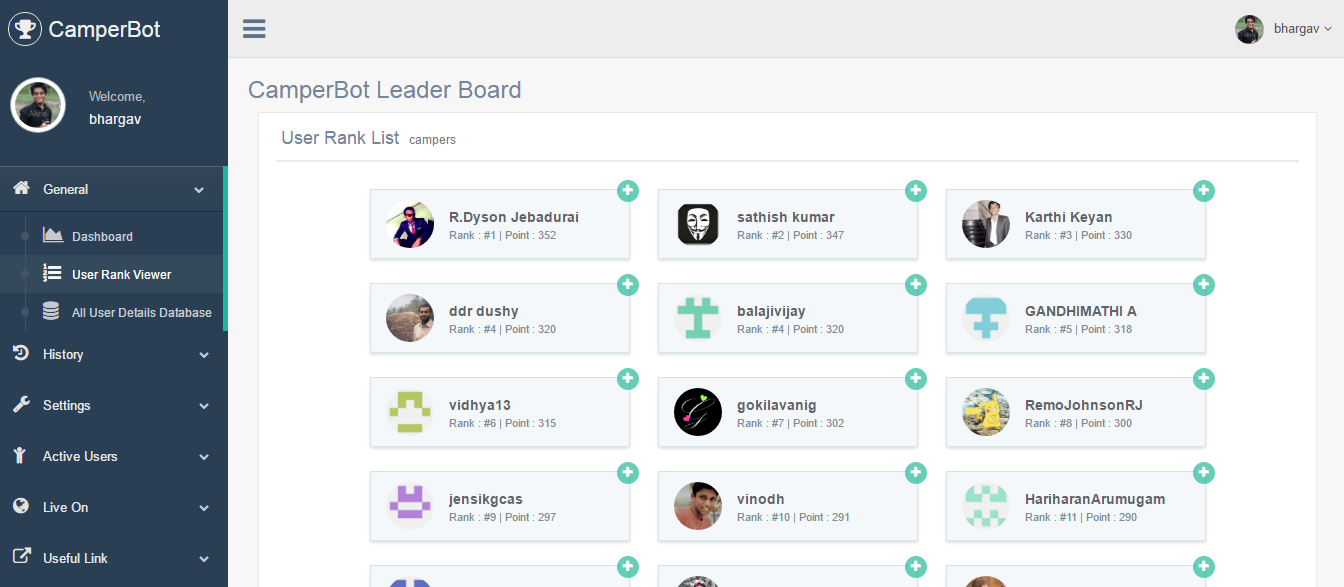
**9.2.3 ACTIVE USER:**

It displays the user those who are Active for a long period (monthly wise). We can also view their whole information by clicking the plus sign which is placed in the right corner of each user activity.

****

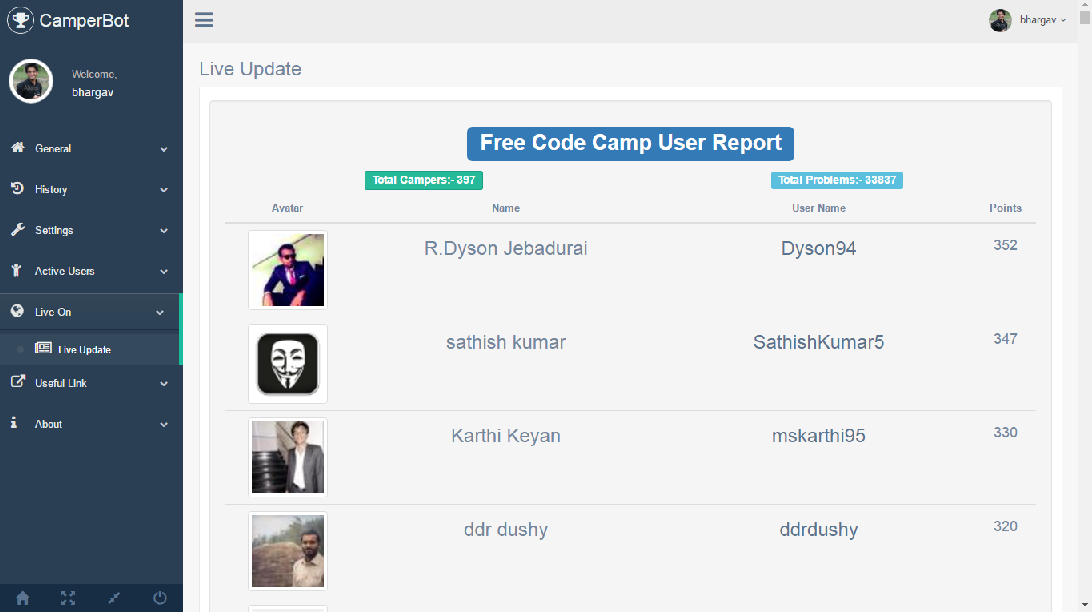
**9.2.4 USER RANK:**

User list will be displayed based on their rank. That is the campers with highest rank will be displayed in a descending order.

****

**9.2.5 LIVE UPDATE:**

Live update displays the information like user name, brownie points, user profile picture. The data is directly fetched from the API for every 10 minutes.

****

**REFERENCES**

* 1. <https://developer.gitter.im/docs/welcome>
  2. <https://developer.github.com/v3/oauth/>
  3. <https://github.com/freeCodeCamp/open-api>