Applied Project 2 (3002)

National School of Business Management

Team Members :

Manathunga Pubuditha Chanakya Manathunga (4610087)

Purusoth Shanmugarajah (4610078)

A.V Chamodh Abhisheke (4610107)

P.P.Tilan Jayaweera (4610073)

L M Basura(4610082)

September 10, 2020

Contents

Chapter 1

Implementation

1.1 Introduction

Here we are going to discuss about the steps taken to bring our conceptual idea into reality. We are going to go over the major moving pieces of the system, how they are configured and how they work together with each other. We are going to go over the following topics in detail.

- Environment - This is the development environment that we have used when creating the system. This needs to be the same among all the participants of the group or else it can cause many problems. The synchronization of the software system is more imperative than the hardware requirements.
- Development tools and Technologies - These are the different software components that were used inorder to implement the proposed system.
- Application Architecture - This Area goes over the arhitecture of the system.
- Reference Material - These are the different sources of knowledge we used how we could implement our system
- Implementation - Here we discuss about what we actually did durint implementation. We go over how we set up the database, the test server and what we did softwarewise.
- Major Code Segments -This section goes over what we think to be important segments of code.

1.2 Implementation Environment

Since we are building a website that people can access, we needed a way to host the website. We thought of using a service caleed "Heroku" inorder to do so. Heroku is a cloud Platform as a Service (PaaS). Using heroku makes it easier for people to deploy, manage and scale apps.

There are several reasons we chose to use Heroku :

- Heroku Supports Python(the programing language that was chosen) and has documentation and tutorials on how to get The Django Framework Up and running.

- Using Heroku meant that we did not have to wory about the Software and Hardware Environment being used. Since docker uses containers we are able to customize the environment to suit the needs to of the software.
- We can easily communicate with the heroku servers using a very user friendly CLI(Command Line Interface) . We prefer the CLI over the the tradional because , personally we find working with CLI to be faster.
- Deployment is taken care of. This is done using git commands. For example, after creating a heroku project we can simple upload the code and get deployed using :

```
git push heroku master
```

1.3 Development tools and Technologies

1.3.1 Development Tools

Various Development tools were used when creating this. Let us go over them.

- PyCharm - PyCharm is an IDE(Integrated Development Environemnt) that is used in development of applications, especially using Python. PyCharm makes it easier to write and maintain large scale Python Projects. Since we have html,css and other non Python Code in this, it also helps having that we have PyCharm. Because Pycharm not only supports writing of python code, but also helps with others as well. Git Integration is also another reason that Pycharm became useful.
- Vim - Vim is text editor, which is a sucessor of the program vi. It is very popular among linux users because vim/vi comes preinstalled in most linux distro. It is lightweight and highly customizable.
- Firefox - Firefox is an Open Soure Webbrowser from Mozzilla. This is was used as the primary web browser for testing the site.
- Chrome - Chrome is the other Webbrowser we have used. It is the most popular Web browser and as such, we need to test the webstire on chrome as well.
- Virtual Environment (venv) - This is an important tool that allowed us to create software environment to use our application. Using Virtual enviroments meant that we did not need install packages globally.
- Git - Git is version control system. Using using git all of the team members were able to alter the same code base without having to worry about what changes another person has done.Also it allows us to track changes . It gives us the opportunity to potentially backtrack to an older version if the current version is potentially buggy.

1.3.2 Development Technologies

Having the Software Development environment constant was necessary inorder for everyone to sucessfully contribute. This meant that we had to be mindful about the software versions that we were using. Below are the software packages/libraries that were used and their various versions.

- python version 3.8.3 - High Level , Interpreted general purpose programming language
- Django version 2.2.13 - A framework for building websites with mvc architecture in python
- djongo version 1.3.1 - Connects django orm with MongoDB
- dnspython version 1.16.0 - This is a DNS Tool kit For Python
- bson version 0.5.8 - An Implementation of BSON format for Python

Using Virtal Environments allowed us to make sure that everyone had these exact versions. Virtual Environment is a python environment such that Python interpreter , libraries and scripts installed into it are isolated from the default libraries installed in the person's own computer.

So even though every one did not have the same libraries and python version preinstalled in their computer, we were able to maintain a constant enviroment using Virtual Environments. This way of seperating the different projects made it easier for people to have the same packages install with the same versions without running into many errors and hardships.

We have choosen to use NoSQL for this. NoSQL is storing data in a non relational format. NoSQL allows us to have unstructured data. There are several reasons for using NoSQL Databases :

- It is scalable. So if required using NoSQL we can cater to increase database requirements without much headache.
- It is schemaless so if required we have the freedom to store unstructured data.

We have used MongoDB. MongoDB is NOSQL database, specifically a document database. MongoDB was very easy to get setup. Using "MongoDB Atlas" we were able to host our database in the cloud. This makes it possible to anyone to easily access the data through the web, this makes it ideal for out situation.

1.3.3 Existing Reusable Components

Django - Django comes with several pre implemented features that a developer can use directly or customize as required . This includes :

- User Model - Django has it's own User Model. The user Model has the attributes such as :
  - username - The Unique Identifier of the user.
  - first_name

  - last_name
  - email - Used for things like password reset and so on.
  - password
  - user_permission - The User permissions that have and been given to the person
  - is_staff - Is the person an admin
  - and etc ..

  Having the basics all setup makes our jobs easier. We had to alter this model a bit to include some more attributes like rank,score , submissions. This will be explained later.

- Authentication - Django Provides us with forms and logic to perform basic authentication and other related actions when it comes managing user accounts. This includes the following :
  - Logging in and Logging out
  - Password Reset
  - Password Change
  - Registration

  Since these are setup with industry standards we, we do not need to put much effort into it. We just simply to minor tweaks to make it compatible with our scenario and that is it.

- Django ORM - Django provides a fully functional ORM right out of the box. An ORM allows us to use Database Objects as we were using Objects when it comes to object oriented programming. We can do the basic CRUD operations using an object oriented paradigm . This means that we do not need to mess around with writing Database Code such as SQL.

  The ORM makes it easier for us to make changes to the database structure because it too is handles automatically. Also this abstraction allows us to use change the database we are using without needing to redo the whole system's programmingturnover .

- Bootstrap - Bootstrap is css framework that helps in creating beautiful looking UI which are responsive. Why usee Bootstrap?
  - BootStrap has premade designs. It thus will save us a lot of time, which usually make lots of times to create
  - It is easy to use
  - BootStrap can be very easily customized to fit out UI needs
  - Using Bootstrap is free
  - Bootstrap is popular. So there are so many resources to learn from and things that people have already pre built.

1.4 Application Architecture

The MVC architecture was used when creating this. Django uses this model when it comes to building the software. Thus we too are incentivised to use this model. The MVC architecuture consists of three components :

- Model - The Model holds the any of the code related to interacting with the database. When it comes to django they refer to model using the same name.
- View - This is the component used for the UI and other UI related parts of the website. When it comes to django they call this the template.
- Controller - This is the part that handles the business logic of the application. It coordinates the flow of information between the model and the view . Confusingly django calls this the View.

Why Should We Use Use the MVC architecture ? Well these are the main reasons why we actually chose it :

- Using MVC makes the process of building the site less complex. Since the parts are not connected, we can develop them seperately without having to parrallely develop them, if it was done in one file
- Since the different parts can be developed seperately, the different parts can be easily assigned to different people. This makes it easier for people to collaborate in a project.
- Using MVC makes it much easier to modify the code and also maintain it.
- Debugging of the application much easier if it uses the MVC architecture.

So even though Django uses the MVC architecture, their naming is different, instead they have Models,Templates and Views.

1.5 Reference Material

We have used several resources to assist us in the development of the software. Based on the order of importance they are listed below :

- django official Documentation - This is one of the most thorough documentation that we have ever come across.
- stackoverflow - This is a site where programmers post questions/problems , and others answer them
- youtube - was very useful in understanding the concepts and nuances of web developoment
- djongo documentation - Documentation for using the MongoDB as the database for the website

1.6 Implementation

Lets us go into detail of what we did inorder to develop the website.

1.6.1 Database Setup

We used two databases when building the website. One is the local test database while the other is the production database. Both of them are mongodb. To setup mongoDB locally , simply installed mongodb on our computers. After it is installed we can use it .

Initally we turn on the mongo daemon using the following code (linux).

```
sudo systemctl start mongod
```

1.6.2 Accessing Software

Inorder to use the website, only a webbrowser is required. The website supports both Mobile and Desktop Environments.

1.7 Major Code Segments

Let us now see some of the important and noteworthy snippets of code.

1.7.1 Database Connection

We used a package called djongo inorder to make the connection to the mongoDB database. Connecting to NOSQL databases does not official support by Django.

Inorder to make this work we had to do the following steps .

1. Installing djongo using pip
2. Adding the following configuration to the application. We have replaced sensitive information with 4x.

```
DATABASES = {
    'default': {
        'ENGINE': 'djongo',
        'NAME': 'xxxx',
         'CLIENT':{
            'host':'mongodb+srv://xxxx:xxxx@cluster0-bvrc6.
               ↪ mongodb.net/xxxx?retryWrites=true&w=majority',
            'username':'xxxxx',
            'password':'xxxxx',
            'port':27017,
        }

    }
}
```

- ENGINE : is for the type of database we are using. Usually we specify the database like MYSQL or Postgres. But since Django doesnt officially support django we have to specify it as such.
- NAME : Name is the name of the collection (equivalent to the database) we are using.
- CLIENT : This describes the various things that are required inorder to be authenticated to connect to the database,that is hosted.
  - host : the connection string. Has username and password in it.

3. Now we can use Django ORM as if we were using a regular relational Database

1.7.2 Custom User Model

Django is very convinient when it comes to getting a fully functional website up and running fast. But sometimes the default configurations do not fit what we are looking for . The Default User Model is a great example of this. Using the Default User we can easily get a authentication system working. But it is lacking.

Using the django documentation we see that it is quite easy to modify the default User Model.Let us see what changes were done inorder to make this suitable for use.

1. Created a app called myuser using the code

```
django startapp myuser
```

2. Then in models.py of the app myuser(we created above) the following code was added .

```

from djongo import models
from django.contrib.auth.models import AbstractUser
from submission.models import Submission


class MyUser(AbstractUser):
    # This is is to store the date of birth
    date_of_birth=models.DateField(null=True,blank=True)

    #Based on the work done a person is provided a rank
    rank=models.IntegerField(default=1)

    # This is an array of submissions. These are all the
        ↪ submissions by a Specific user.
    submissions = models.ArrayField(model_container=Submission,
        ↪ default=[])

    # This is what counts a person's score
    score=models.IntegerField(default=0)

    def increment_score(self,project_level,rank) :
        if(rank == 0 or rank >3 ):
            return
        threshold = self.get_threshold()
        self.score += (4-rank) * project_level
        if(self.score > threshold) :
            self.rank += 1

    def get_threshold(self):
        threshold = (2 ** (self.rank-1) )* 10
        return threshold
```

We import AbstractUser. We Create a Class MyUser that inherits from AbstractUser. AbstractUser is a class in django and it is what django inherit from to create it's User Model. We add new Attributes like :

- date_of_birth - The person's Birthday(A Date)
- rank - The Person's ranking(A Number)
- submissions - The different submissions they have created (An Array of Submissions). The Default value is an empty list
- score- The Person's Score which determines the person's rank (A Number)

  Notice that a person is using an Array Field. This is not something that can be done in Relational Databases. We would instead use another table and use joins when displaying them. Since NOSQL is schemaless and actively discorages joins and instead uses redundancy and data duplication, we have gone with this way instead.

  Then there is funcition called *increment_score*. This function is what gives progress. When a person has a rank that is not zero, his score will increase relative to how hard the project is (based on project level).

3. Finally We add the following line to *settings.py* in the project folder.

```
AUTH_USER_MODEL='myuser.MyUser'
```

Now we can use the built in Authentication Views to with our new model

1.7.3 Authentication

Using Django makes the process of Authentication very easy to setup. Lets go over how we got this to happen.

1. The Easiest way to make the Authentication Available is by adding the routes to the views of the authentication to the main urls.py file.

```
urlpatterns = [
 ...
   path('accounts/',include('django.contrib.auth.urls')),
]
```

Now you can access the different views that come with this. These include :

- accounts/login/ [name='login'] - For login
- accounts/logout/ [name='logout'] - For Logout
- accounts/password_change/ [name='password_change'] - To CHange Password
- accounts/password_change/done/ [name='password_change_done'] - After Password CHange is done
- accounts/password_reset/ [name='password_reset'] - Password Reset
- accounts/password_reset/done/ [name='password_reset_done'] - After Password Reset
- accounts/reset/¡uidb64¿/¡token¿/ [name='password_reset_confirm'] - Confirm Password Reset
- accounts/reset/done/ [name='password_reset_complete'] - After Password Reset Complete

2. We can customize some of these by giving our own custom template. We wanted to have a custom Login template. It is very easy when it comes to django. Create a folder called registration and add our custom template into it. We created a folder called registration under the myuser's template folder. This is the structure of the html file. Notice that inputs are provided from django's forms and we did not have to define it explicitly.

```
{% extends "my/base.html" %}
{% block content %}
<h2>Login Page</h2>
{% if form.errors %}
<p> The username and password did not match </p>
{% endif %}

{% if next %}
       <p>Please Login To View this Page </p>
{% endif %}
<form method ="POST" action = "{% url 'login' %}">
       {% csrf_token %}

               <label >Username</label>
                <div class="form-control"> {{ form.username}}</
                   ↪ div>

        <label >Password</label>
            <div class="form-control"> {{ form.password  }} </
               ↪ div>


       <p> <input class="btn btn-primary" type="submit" value="
           ↪ Login"> </p>
        <input  type="hidden" name = "next" value = "{{ next }}">
</form>

<p> <a href="{% url 'password_reset' %}"> Lost Password ? </a>
    ↪ </p>
{% endblock content %}
```

1.7.4 Registration

Django helps us in the registration proccess as well. The UserCreationForm is a form that basically takes care of creating a user. Since our user Model is different and we require more information form the user we can extend this UserCreationForm to suit our needs.

1. Initially we create our custom Form. This form inherits from the UserCreationForm .

```
from django.contrib.auth.forms import UserCreationForm
from django import forms
from .models import MyUser
class RegistrationForm(UserCreationForm) :
    email = forms.EmailField(required=True)

    class Meta :
        model = MyUser
        fields = (
        'username',
        'first_name',
        'last_name',
        'email',
        'date_of_birth',
        'password1',
        'password2',
        )

    def save(self,commit=True) :
        user = super(RegistrationForm,self).save(commit=False)
        user.first_name = self.cleaned_data['first_name']
        user.last_name = self.cleaned_data['last_name']
        user.email = self.cleaned_data['email']
        user.date_of_birth = self.cleaned_data['date_of_birth']

        if commit :
            user.save()
            return user
```

We need added the field date_of_birth as well, even though it is not in the typical user model. The fields password1 and password2 need to be named like that. They are the two password inputs. Then in the save function we describe how we take the the data and save it to the model.

2. The we connect the form with a view. Below you can see that if the request is a get request the form is loaded. If it is a post requst then the request is processed. If It is valid the user is created. If not the form is loaded again.

```

def register(request) :
    if request.method == "POST" :
        form = RegistrationForm(request.POST)
        if form.is_valid() :
            user=form.save()
            #user= authenticate(username = username, password =
                ↪ password)
            login(request,user)
            return redirect("projects:feed")
            #return HttpResponseRedirect(reverse("projects:feed
                ↪ ",))

    else :
        form = RegistrationForm()

    context = { 'form':form}
    return render(request,'my/register.html',context)
```

3. The Only thing left is to add the view to the urls.py so that it can be navigated into.

1.7.5 Project Model

People post their projects into our site to get them done. The Project model is one of the more important models in our site. Let us see how this model was created and what other parts are related to this.

Below is the code for the project model.

```
from djongo import models
from submission.models import Submission

class Project(models.Model):
    # This is the price the creator of the project is willing to give
    amount = models.DecimalField("Offered Price", max_digits=6,
        ↪ decimal_places=2)

    # The username of the creator of the project
    creator = models.CharField(max_length=200)

    # This is an array of submissions. The submissions contain the
        ↪ creator,thesubmitter and the files
    submissions = models.ArrayField(model_container=Submission ,
        ↪ default=[])

    # The date it needs to be submitted
    deadline = models.DateField()
    # #tags=models.ArrayField(model_container=models.CharField(
        ↪ max_length=40))

    # The Name of the project
    name = models.CharField(max_length=200)

    # A Description of the project
    description = models.CharField(max_length=5000)

    # If the user has given the ranks
    ranked = models.BooleanField()

    # Date Posted

    date_posted = models.DateField()

    # Project Level

    level= models.IntegerField(default=1)
    # What happens when we type print(project)
    def __str__(self):
        return self.name
    # A Function to get the number of submissions made
    def get_no_of_submissions(self):
        return len(self.submissions)
```

There is an attribute called ranked. This is a flag to check whether the project creator has ranked the submissions that people have given. Again this model has used an Array of submission. The Submission model will be discused later. The project has a level. This level determines the number of points a person can get from doing the project.

1.7.6 Submission

The Submission is another important Model. The submission class is an abstract model. Usually a transfers to a document in MongoDB. But since the Submission does not exist alone and only exists in a Project, or for a person it is cracterised as an abstract model.

Below is the code for the Submission model :

```
from django.db import models



class Submission(models.Model):
```

```
class Meta:
# This means that submissions is not created in the database
   abstract = True

# The Submitter information is given as username
submitter = models.CharField(max_length=200)

#The files given by the submitter
files = models.URLField()

# The Date and time the submission was made
dateTimeField = models.DateTimeField()

# The project to which the submission was made
project = models.IntegerField()

# The rank of the submission given by the creator of the project
rank = models.IntegerField()
```

1.7.7 Project Feed

The Project Feed is where we display the projects that have been created. Each element in the project feed goes to a details page, where a person can view more details of the project and add submissions. Inorder to make this work we have done the following :

- Created a view Inheriting from Django's ListView

```
class project_list_view(ListView):
   model=Project
   template_name='projects/home.html'
   context_object_name='projects'
   ordering=['-date_posted']
```

  Here we set the model we are using as well as other features like how we want to order the list. We have created a template .

- A template was created. Below is the code for the template that was created.

```
{% extends "projects/base.html" %}
{% block content %}
       {% for project in projects %}
                      <article class="media content-section">
         <div class="media-body">
           <div class="article-metadata">
             <a class="mr-2" href="#">{{ project.creator }}</a>
```

```
                <small class="text-muted">| Dead Line : {{ project.
                    ↪ deadline|date:"D d M Y" }} </small>


              </div>
              <h2><a class="article-title" href="{% url 'projects:
                  ↪ details' project.id %}">{{ project.name }}</a
                  ↪ ></h2>
              <p class="article-content">{{ project.description
                  ↪ }}</p>

            <div >
                <small class="text-muted">Amount : {{ project.
                    ↪ amount}}$ |</small>
                <small class="text-muted">Tag {{ project.tag}} | </
                    ↪ small>
                    <small class="text-muted">Submissions {{ project
                        ↪ .get_no_of_submissions }}</small>

              </div>
            </div>
          </article>
          {% endfor %}
{% endblock content %}
```

We have used bootstrap inorder to style this.

1.7.8 Ranking People

Ranking allows people to see how well their submission is and learn from them. Ranking incentivises people and the rate at which people rank helps in balancing efforts.

Let me explain. A novice will choose to do easier projects. They will try and some may win, there score will increase. But there is also a chance for a professional to keep on doing novice projects. This is bad , since novice people do have the chance to win, while the professional/skilled people do not get challenging experience to learn from. So it is imperative that we try to make skilled people do tthe more difficult ones rather than the easier ones.

So inorder to make sure that this is the case we have come up with the following way to rank a person.

```
def increment_score(self,project_level,rank) :
        if(rank == 0):
            return
        self.score += (4-rank) * project_level
        threshold = (2 ** (self.rank-1) )* 10
        if(self.score > threshold) :
            self.rank += 1
```

There are 2 Key features here :

- The score a person gets per project depends on how well they did as well the project level. So a person can get more marks by doing a project with a higher level
- Every time the rank increases the score needed to go to the next rank doubles.

This helps with the following :

- In the beginning hook people in as they see quick progress
- disincentives people with higher ranks from doing lower level projects because their level is not getting increased.
- People with higher skills can very easily advance their skills because they can do projects that more difficult and earn more points.

1.8 Deployment

1.8.1 Deploying site on Heroku

The Deployment was done on Heroku. Heroku is a Cloud Platform as a Service. Using heroku we can very easily get the site live with a public domain.

Below we will go over the steps that were taken inorder to deploy the project.

- Installing the Heroku cli and logging in
- Create a heroku project at the project root using :

  ```
  heroku create
  ```

  Heroku will create a remote repository. We add the created repository using the following code. The xxxx is the name of the remote repository

  ```
  heroku git:remote -a xxxx
  ```

- Installing gunicorn server. We used the regular development server when testing our project. Now we need to use a development server software. We will use gunicorn.

  ```
  pip install gnuicorn
  ```

- Create the Procfile. The Profile tells heroku what to do run over python server. Below is the code for the Procfile.

  ```
  web: sh -c 'cd ./cerfify/ && gunicorn cerfify.wsgi'
  ```

- Get the requirements of the projects into a text file called *requirements.txt* in the root folder of the project.

  ```
  pip freeze > requirements.txt
  ```

- Now add the new created files and commit the changes.
- Finally say :

  ```
  git push heroku master
  ```

  This will send the project to the repository and run it. The Website will be hosted.

Currently the website link for the site is :

```
https://fierce-escarpment-10409.herokuapp.com
```

1.8.2 Working With Static Files

Heroku discourages saving static files in their website and urges developers to use CDN(Content Delivery Networks) to actaully host the varies static files. Since we do not have many static files storing them would be easier. Let us see how we do so using django.

- We first set the following in the *settings.py*

  ```
  BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(
      ↪ __file__)))

  # Static files (CSS, JavaScript, Images)
  # https://docs.djangoproject.com/en/1.9/howto/static-files/
  STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
  STATIC_URL = '/static/'

  # Extra places for collectstatic to find static files.
  STATICFILES_DIRS = (
      os.path.join(BASE_DIR, 'static'),
  )
  ```

- Then we create a folder called static and place a file within it. It does not matter what it is. The file is placed inorder for git to track that the folder was added.
- Django does not allow the serving of static files during production. We use library called WhiteNoise to make this possible. We first install it and add it to the *MIDDLEWARE* in the project settings.

  ```
  pip install whitenoise
  ```

  ```
  MIDDLEWARE = [
                  ...
      'whitenoise.middleware.WhiteNoiseMiddleware',
      ]
  ```

- Finally we commit the changes and push it to heroku.

1.9 Security

Security is a concern when it comes to any software. Lets go over what comes out of the box when we use django

- Using Django template systems help us decrease the chances of XSS - Cross Site Scripting. Django template systems by default escapes variables, unless they are explicitly marked as safe.
- We can turn on CSFR protection. This guarentees that POST requests are sent from your site.
- SQL Injection Protection - When we use the database we use Django's built in ORM . Thus there is no easy way for hackers to get into to the database.
- Django can detect if content is being requested from an unauthorized iframe. This helps prevent Clickjacking.
- The Default Password Hashing algorithm is PBKDF2, or another is bcrypt. Both very good hashing algorithms, and stops brute force password attacks

Chapter 2

Testing

The Main Objectives of doing the tests are :

- to make certain that the product we create matches the customer requirements and also is bug free.
- to find out errors that were done during the implementation phase.
- to ensure the product quality.
- to make sure that system does not break

2.1 Test Plan

2.1.1 Objectives

The Testing Plan gives an overview of the different aspects for the testing efforts being taken. Writing this document means means that :

- anyone who is new can easily get up to speed with the testing practices by picking up this document.
- the aspects of the software that needs to be tested are identified
- a high level strategy of how testing is to be conducted is created.
- the resources required are understood.
- the scope of testing being done is clear.

2.1.2 Scope

Features to be tested

These will be elaborated futhur on the testcases

- User Management
  - Registering - User Creation
  - Login - Loging in to the Website

  - Logout - Logging out of the the system
  - User Profile - Increasing/Display of rank.
  - LeaderBoard - Ranking among the different people

- Project Management
  - Project Creation
  - Project Closure
  - Project Feed

- Submission Managment
  - Adding Submission
  - Viewing Files - Permissions for who can view what files

Features not being Tested

We are not going to test the following :

- Performance
- Security
- Hardware

2.1.3 Types of Testing

- Unit Testing - The Units/Components of the Software is tested. This is the easiest to check and fix
- Integration Testing - Several Units/Compnents are tested to together, particulary when a scenerio is being tested
- System Testing - Overall System is being tested

2.1.4 Environment Requirements

We are going to check the sytem on several systems

- Chrome Web (Latest Version) browser on :
  - Windows
  - Linux
  - IOS
  - Android
  - Mac

- Firefox Web (Latest Version) browser on :
  - Windows

- Linux
- IOS
- Android
- Mac

2.1.5 Roles and Responsibilites

- Project Manager - Overall Guidance and management
- Test Lead - Creates the Test Plan and Test Cases
- Implementation Team - Implements the software and performs unit Tests
- Testing Team - Performs integration and System Tests

2.2 Test Report

Test Case ID	**Test Scenario**	**Test Case**	**Pre Conditions**	**Test Steps**	**Test Data**	**Expected Results**	**Actual Result**	**Status**
CL_01	Check Login functionality	Check Response when valid username and password is provided	Must have gone to the Login page And not be logged in	1.Go to the Website 2.Click On Login in the NavBar 3.Enter a Valid Username 4.Enter a Valid Password 5.Click Login	Username : palm Password: kg%XF642	Go to the Home Page	Opened the Home page	Pass

CL_02	Check Login functionality	Check Response when valid username and invalid password is provided	Must have gone to the Login page And not be logged in	1.Go to the Website 2.Click On Login in the NavBar 3.Enter a Valid Username 4.Enter an Inalid Password 5.Click Login	Username : palm Password: abcdefghi	Error Message The username and password did not match	Got An Error Message The username and password did not match	Pass
CL_03	Check Login functionality	Check Response when invalid username and valid password is provided	Must have gone to the Login page And not be logged in	1.Go to the Website 2.Click On Login in the NavBar 3.Enter an Invalid Username 4.Enter a Valid Password 5.Click Login	Username : palm_1 Password: kg%XF642	Error Message The username and password did not match	Got An Error Message The username and password did not match	Pass
CL_04	Check Login functionality	Check Response when invalid username and invalid password is provided	Must have gone to the Login page And not be logged in	1.Go to the Website 2.Click On Login in the NavBar 3.Enter an Invalid Username 4.Enter an Invalid Password 5.Click Login	Username : palm_1 Password: abcdegh	Error Message The username and password did not match	Got An Error Message The username and password did not match	Pass

CR_01	Check Register Functionality	Check Response when valid data is provided for registration	Must have gone to the Register Page, not be logged in and not have the username already registered	1.Go to the Website 2.Click On Regsiter in the NavBar 3.Enter a Valid Username 4.Enter First Name 5.Enter Last Name 6.Enter a valid Email 7.Enter a valid Date OF Birth 8.Enter a valid Password 9.Confirm the password by retyping 10.Click Register	Username : palm Email : palm@gmail.com Password: kg%XF642 Date Of Birth : 12/13/1998 FirstName : Puduitha LastName : Manathunga	Go to the Home Page	Opened the Home page	Pass

CR_02	Check Register Functionality	Check Response when valid data , with empty confim Password	Must have gone to the Register Page, not be logged in and not have the username already registered	1.Go to the Website 2.Click On Regsiter in the NavBar 3.Enter a Valid Username 4.Enter First Name 5.Enter Last Name 6.Enter a valid Email 7.Enter a valid Date OF Birth 8.Enter a valid Password 9.Keep Confirm password Empty 10.Click Register	Username : palm Email : palm@gmail.com Password: kg%XF642 Date Of Birth : 12/13/1998 FirstName : Puburitha LastName : Manathunga	Show Error saying this field is required	Shows Error Saying This Field is required	Pass

CP_01	Check Project Creation Functionality	Check Response when valid data is provided for Project Creation	Must have gone to project creation page and need to be logged in	1.Go to the Website 2.Click On Get Your Project Done in the NavBar 3.Enter a Project Name 4.Enter Project Description 5.Give Price Money value 6.Give Project Level 7.Enter a valid Date 8.Click Create Project	Project Name : Test Project Name Project Description : Test Project Description Price : 0 Project Level : 1 Date : 10/1/2020	Go to Project Detail page	Go to Project Detail Page	Pass

CP_02	Check Project Creation Functionality	Check Response when valid data is provided for Project Creation, But with invalid date	Must have gone to project creation page and need to be logged in	1.Go to the Website 2.Click On Get Your Project Done in the NavBar 3.Enter a Project Name 4.Enter Project Description 5.Give Price Money value 6.Give Project Level 7.Enter an invalid Date 8.Click Create Project	Project Name : Test Project Name Project Description : Test Project Description Price : 0 Project Level : 1 Date : 10/1/0000	Show Error Saying enter valid date	Shows an error saying invalid date	Pass

CP_03	Check Project Creation Functionality	Check Response when valid data is provided for Project Creation, But with a level number greater than 7	Must have gone to project creation page and need to be logged in	1.Go to the Website 2.Click On Get Your Project Done in the NavBar 3.Enter a Project Name 4.Enter Project Description 5.Give Price Money value 6.Give an Invalid Project level 7.Enter a valid Date 8.Click Create Project	Project Name : Test Project Name Project Description : Test Project Description Price : 0 Project Level : 100 Date : 10/1/2020	Show Error Saying Enter number no more than 7	Shows Error Saying Enter number no more than 7	Pass

CP_04	Check Project Creation Functionality	Check Response when valid data is provided for Project Creation, But with a level number greater than 8	Must have gone to project creation page and need to be logged in	1.Go to the Website 2.Click On Get Your Project Done in the NavBar 3.Enter a Project Name 4.Enter Project Description 5.Give invalid Money value 6.Give an valid Project level 7.Enter a valid Date 8.Click Create Project	Project Name : Test Project Name Project Description : Test Project Description Price : asjdkasd Project Level : 1 Date : 10/1/2020	Show Error Saying Please Enter a Number	Shows Error Saying Please Enter a Number	Pass

CV_01	Check View Project Details Functionality	Check What Happens when the creator of project views details	User needs to be logged in and in the details Page that was created by the same user . The Project still needs to be open	1.Go to Details Page of a project created by the user	No Data	Add Submission Section Should not be found	Add Submission section not found	Pass
CV_02	Check View Project Details Functionality	Check What happens when a user creates a submission	1. User needs to be logged in 2. There needs to be an open project to submit to	1.Go to Details Page of a project created by the user from the Project Feed 2.If the project	File Link: https://www.github.com/	The page should reload with the submission having been added	The Page Reload with the submission being added	Pass

CSR_01	Check Set Rank Functionality	Check What happens when the user gives valid rankings to the user submissions	1.User Needs to be logged in 2.User Needs to have created a project and it should still be open 3.There needs to be submissions made for the said project	1.Go to Details Page of a project created by the user 2.Ranks to the submissions are set with valid values 3.Set Rank Button is clicked	Ranks : 1,2,3	Page Reloads with the Ranks set	The Page Reloads with the ranks set	Pass