**SOFTWARE REQUIREMENT SPECIFICATIONS**

1.1 Product Overview

1.2 External Interface Requirements

1.2.1 User Interfaces

1.2.2 Hardware Interfaces

1.2.3 Software Interfaces

1.2.4 Communication Interfaces

1.3 Functional Requirements

1.4 Software System Attributes

1.4.1 Reliability

1.4.2 Availability

1.4.3 Security

1.4.4 Portability

1.4.5 Maintainability

1.4.6 Performance

1.5 Performance Requirements

1.6 Database Requirement

1.7 Design Constraints

1.8 Other Requirements

* 1. **Product Overview:**

TecRidge is a platform that enables developers to learn about the projects going on in their surroundings. Also, the developers can add their own ideas into the platform where they can come across the fellow developers, so that they can team up and hence make a better quality product of their idea. Also the developers can find for the suitable guide to guide them in their way to build the product. Also majority of the developers today are cut half their way because of lack of motivation in the area of their interest. Today technology has grown far way long, that all fields of developing can combine to produce great products.

* 1. **External Interface Requirements:**
     1. **User Interfaces**

For User interaction, we have used React-js with Redux-js for a better and faster response. Also it gives better UI for easy usage by the user.

* + 1. **Hardware Interfaces**

The point of interface with the user is either Smartphone of the user or the laptop. On the other end for storage, we use Firebase DB at the back-end.

* + 1. **Software Interfaces**

We are using genetic algorithm for the grading of the projects uploaded in the platform. This is developed using python. Also the platform is web application, which works good in any browser that supports the JavaScript.

* + 1. **Communication Interfaces**

**1.3 Functional Requirements:**

Developers can give their skill set, which they are good at, so that any person can see the profile of the developer before taking them into the team in a project. Also guide can see the skill set of student and vice-versa before approaching.

Developer can search through the projects on the platform so that he/she can join any project on their point of interest. Also they can use this functionality to see the projects done in past to modify and upgrade it.

Also we are using **genetic algorithm** to grade the projects uploaded by the developers. This takes the input from the description given by the developer while uploading the project. The dat given will be stored in Firebase. It is extracted and used for evaluation in GA and the result is stored back to the Firebase.

**1.4 Software System Attributes:**

**1.4.1 Reliability**

The system we use for the UI, React-js is very reliable and many of the web-application today uses the same technology for such purpose. And at the back-end, firebase is a reliable database provided by Google.

**1.4.2 Availability**

The data is hosted on the database that is available 24\*7 and the web application is hosted on the internet. Hence the system is very much available to the user.

**1.4.3 Security**

**1.4.4 Portability**

Since the data is placed in the firebase dataset the web application is very much portable. Also the web application is hosted on the internet and hence it proves the portability of the system.

**1.4.5 Maintainability**

**1.4.6 Performance**

**1.5 Performance Requirement:**

**1.6 Database Requirement:**

**1.7 Data Constraint:**

1)Web app:

* design a website that is extremely easy to navigate, looks good at both orientations and gives the barest minimum content mostly in text format. While this sounds simple, there is a lot of thought that needs to go in the UI and content development.
* unlike print designs, where the viewing area of any design is fixed, web users can (and do) zoom in or out as they interact with a web page, changing the size of text and images. And, by the way, different browsing environments handle zoom differently

2)API (RESTful):

* Uniform Interface
* Stateless
* Cacheable
* Client-Server
* Layered System
* Code on Demand

3)Genetic-Algorithm:

* High Computational Power
* Highest fitness value may not be optimal

4)Native Mobile app

## **Small Screen Sizes and Clunky Controls**: Even as screen sizes become larger and larger, the overall mobile experience is still not an optimal method for reading or accomplishing tasks.

## **Environments Full of Distractions:**Mobile users are much more prone to dropping off than their web counterparts. Phones are often used in non-work settings

## **Broken Data Passing:** Data Passing is pretty complex in mobile apps. No contextual data is stored ,therefore nothing is passed from existing apps to a recently installed app.

**1.8 Other Requirements:**