**Software Requirements & Prototype Document**

**Project Title:** Web System 1.0 – A Quiz Creator

**Objective:** Our website will look to let users create, take, and manage quizzes and surveys. Users will be able to easily create quizzes and manage their own, and other users will be able to view these quizzes and take them. Quiz owners will be able to receive answers and see the results of every user.

**User Roles:**

* User (Can take, create, and modify their own Quizzes)
* Administrators (Manage all quizzes)
* Scalable: Can add more specific roles to create exclusive quizzes

1. **Introduction**
   1. **Purpose**

The purpose of this project is to create a centralized quiz website where users can create multiple quizzes and receive results. This can be important for collecting data, or can be for fun.

1. **Requirements Glossary**

User Requirements – Describes user needs.

System Requirements – Describes the functionality of the project that must be present.

Non-Functional Requirements – Describes the hidden aspects of the project, such as performance, security, quality, and modularity.

Technical Requirements – Describes the technologies that will be used.

Project Risks and Constraints – Describes the risks and limitations the project may run into.

Project Management – Describes the deliverables, expectations, and benefits of the project.

1. **Requirements**
   1. **User Requirements**

* Users should be able to create an account and sign in.
* Users should be able to, by default, complete quizzes, create quizzes, modify their own quizzes, and delete their own quizzes.
* Users should be able to view the results of their own quizzes.
* Administrators should be able to view every quiz and have full access to modify or delete them.
* Administrators should be able to view the results of each quiz.
* Administrators should also have access to everything a default user would. This implies they can create and take quizzes too.
* Administrators have access to history logs that contain every change for every quiz.
  1. **System Requirements**
     1. **Quiz Content Management**
* Any changes to quizzes, whether created, modified, or deleted, should be trackable and traced to the creator of said quiz.
* Quiz data will be backed up in the database.
* Only one user can be the creator and manager of a quiz.
* Administrators are the only ones able to view history logs for the website, as well as backup data.
  + 1. **Individual Quiz Functionality Requirements**
* Users will be able to create questions and answers in text only.
* Users can choose what kind of answers they want to accept, such as Multiple Choice, True or False, or Multiple Answers when they create their quiz.
* A quiz can have as many questions as the user would like.
* A quiz can be modified and deleted at any time. This can only be done by administrators and the creator of the quiz.
* A user can see the results of a quiz if the creator allows them to during the setup of a quiz.
* Users who created their quiz will see the results of a quiz in graph form.
  + 1. **Dashboard Requirements**
* All users will be able to see every quiz available to be taken on the dashboard in a list format.
* Users will be able to create their quizzes from here.
* Users will have access to a menu that lets them see their own quizzes.
* Administrators will have special options to modify quiz data on the dashboard.
* Users will be able to search for quizzes based on their name or unique ID.

A diagram of a survey

Description automatically generated

* 1. **Non-Functional Requirements**
     1. **Usability**
* The website should be intuitive, and user-friendly.
* Creating a quiz should be a straightforward and simple process.
* Quizzes should be easy to find and take on the dashboard.
  + 1. **Security**
* All data will be securely stored and protected.
* Authentication information will be encrypted.
* Administrator actions will be logged.
* Quiz modification data will be logged.
  + 1. **Performance**
* The website should be responsive and fast for multiple users.
* The system should be able to handle multiple quizzes being loaded.
* The system should not lag when logging information or querying for quizzes.
  + 1. **Portability**
* The website will be available on all desktop browsers.
* The website will not be compatible with mobile devices.

1. **Technical Requirements**
   1. **Languages and Frameworks**

JavaScript – Used alongside React to develop the front end and control the website with scripts.

HTML – Used to structure the website and load React scripts.

CSS – Used to make the website presentable and intuitive.

React – A powerful framework used to dynamically create quizzes and load data from our database to reduce any latency.

Python – Used alongside Flask to build our API and other miscellaneous conversions.

Flask – A framework used to build our API, creating a transaction bridge between our front end and the database.

SQLite – An embedded database that allows us to store our quizzes in a lightweight, portable manner.

* 1. **Tools**

Visual Studio – The main IDE used to develop the website. Allows us to clone our repository easily, and work with JavaScript and Python at the same time.

Node – Used for Node Package Manager (npm) that allows us to install React and other packages down the line.

Figma – Used for prototyping the website before it is developed.

GitHub – Used to store our repository and serves as the main version control for our website.

1. **Project Risks**

* SQLite is designed to be embedded and lightweight. The website will require an enterprise level database to store quizzes if the website becomes massive.
* Quiz data will need to be engineered in a way to be serialized and recoverable without losing any information.
* Administrators have a lot of power, so the amount of Administrators will be small.

1. **Project Deliverables**

* Project Source Code
* Installation Guide for Developers