**Software Design Document**

ScholarHive

### Version 1.0

### Prepared by

### Mohammed Shafal Mujeeb

### Nazal Abdul Rahman

### Suryajith K M

### Ebthisam Moideen Koya

### TKM College of Engineering

**3rd April 2023**

**Table of Contents**

1. [Introduction 4](#_TOC_250012)
   1. [Purpose 4](#_TOC_250011)
   2. [Scope 4](#_TOC_250010)
   3. [Intended audience 5](#_TOC_250009)
   4. [References 5](#_TOC_250008)
2. [System Architecture Design 6](#_TOC_250007)
   1. [Description 6](#_TOC_250006)
3. [Application Architecture Design 7](#_TOC_250005)
4. [GUI Design 8](#_TOC_250004)
   1. [User Interface Design 8](#_TOC_250003)
   2. [Visual Design 14](#_TOC_250002)
   3. [Navigation Design 14](#_TOC_250001)
5. [API Design 15](#_TOC_250000)

#### Technological Stacks 16

#### Component Design 17

#### Algorithm Design 18

#### Data Design 20

#### Error Handling Design 22

#### Performance Design 23

#### Security Design 25

#### Testing Design 26

* 1. Testing Approach 26
  2. Testing Scope 26
  3. Test Cases 26
  4. Testing Tools 29

|  |  |  |
| --- | --- | --- |
| 13.3 | Test Environment | 30 |
| 13.3 | Test Data | 30 |
| 13.3 | Acceptance Criteria | 31 |

#### Maintenance Design 32

#### Deployment Design 34

#### Appendix A: Record of Changes 35

# Introduction

This Software Design Document (SDD) is intended to provide a comprehensive overview of the design and architecture of a software system. The document is aimed at developers, architects, and stakeholders involved in the development of the system. It outlines the key design decisions, architectural patterns, and technologies used to build the system.

The SDD covers various aspects of the software design, including system architecture design, application architecture design, GUI design, API design, and technology stack. The document provides detailed descriptions of the design decisions and provides an overview of the system architecture, the application architecture, and the user interface design. Additionally, it provides details about the API design and the technologies used to build the system.

This document aims to serve as a reference guide for the development team throughout the development process. It will help to ensure that the team remains aligned with the design goals and can make informed decisions regarding the implementation of the system.

## Purpose

The purpose of the Software Design Document (SDD) is to provide a comprehensive overview of the design of the software system being developed. It serves as a guide for the developers, testers, and stakeholders involved in the project, and provides a common understanding of the system architecture, application architecture, GUI design, API design, database design, and technology stack. The document outlines the technical aspects of the software and the design decisions that were made during the development process, ensuring that the software meets the speciﬁed requirements and performs as expected. The SDD also helps to identify potential issues and risks that may arise during the development and implementation phases, enabling proactive measures to be taken to mitigate them. Overall, the SDD serves as a critical reference document for the development team throughout the software development lifecycle.

## Scope

The scope of this Software Design Document (SDD) is to provide a comprehensive overview of the design and architecture of the proposed application. It covers the system architecture design, application architecture design, GUI design (mockups), API design, and the technology stack that will be used to develop the application. The SDD aims to provide a clear understanding of the design and architecture of the application to the development team and stakeholders involved in the project. It will also serve as a reference guide for the development team throughout the development process. The SDD is intended to be a living document and will be updated as needed throughout the project lifecycle.

## Intended Audience

The intended audience for this SDD includes software developers, College Professors, and other stakeholders involved in the software development process. This document is intended to provide a detailed design of the system architecture, application architecture, GUI design, API design, and technology stack, as well as any other relevant technical details related to the development of the software application.

The audience can review the overall design and technical description, and focus on speciﬁc components such as the system architecture, application architecture, GUI design, API design, and technology stack. Any concerns or areas of interest that could impact the development or evaluation of the software application can be noted for further discussion and clariﬁcation

## References

* + - IEEE Standards Association. IEEE Std 1016-2009, IEEE Standard for Information Technology--Systems Design--Software Design Descriptions. IEEE, 2009.
    - Martin, Robert C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2017.

# System Architecture Design

## Description

The architecture consists of three main tiers: the Presentation Tier, the Application Tier, and the Data Storage Tier. The Presentation Tier is responsible for the user interface and screens, which are developed using HTML, CSS, and JavaScript.

The Presentation Tier also includes the search algorithm, which allows the user to search for scholarships based on their eligibility criteria.

The Application Tier is responsible for the backend application logic, which is implemented using the Laravel PHP framework. The Application Tier includes the scholarship eligibility engine, which checks the user's eligibility against the scholarship database to determine the scholarships for which they are eligible. The Application Tier also includes the user management system, which allows users to create accounts and manage their scholarship applications

.The Data Storage Tier is responsible for storing and managing the scholarship database, which is implemented using MySQL. The Data Storage Tier also includes the storage for user information, scholarship applications, and other related data.

These three tiers work together to provide the complete functionality of the system, allowing students to create their own dashboard, search for eligible scholarships, and apply for scholarships through the web application.

# Application Architecture Design

The Scholarship Management application architecture will consist of three main components: a frontend built using HTML, CSS, and JavaScript, a backend built using the Laravel web framework, and a database built using MySQL.

The frontend will handle the user interface and the search algorithm processing. The interface will be designed to be user-friendly and easily navigable, allowing students to search for scholarships based on their criteria. The search algorithm will be implemented using JavaScript and will be optimized for efficiency to quickly search through the large scholarship database.

The backend will handle the business logic of the application, processing user input and managing the interactions between the frontend and the database. The backend will also handle the user authentication and authorization process, ensuring that only authorized users can access certain functionalities of the application.

The database will store all scholarship-related information, including scholarship details, student information, and eligibility criteria. MySQL will be used to ensure efficient data storage and retrieval.

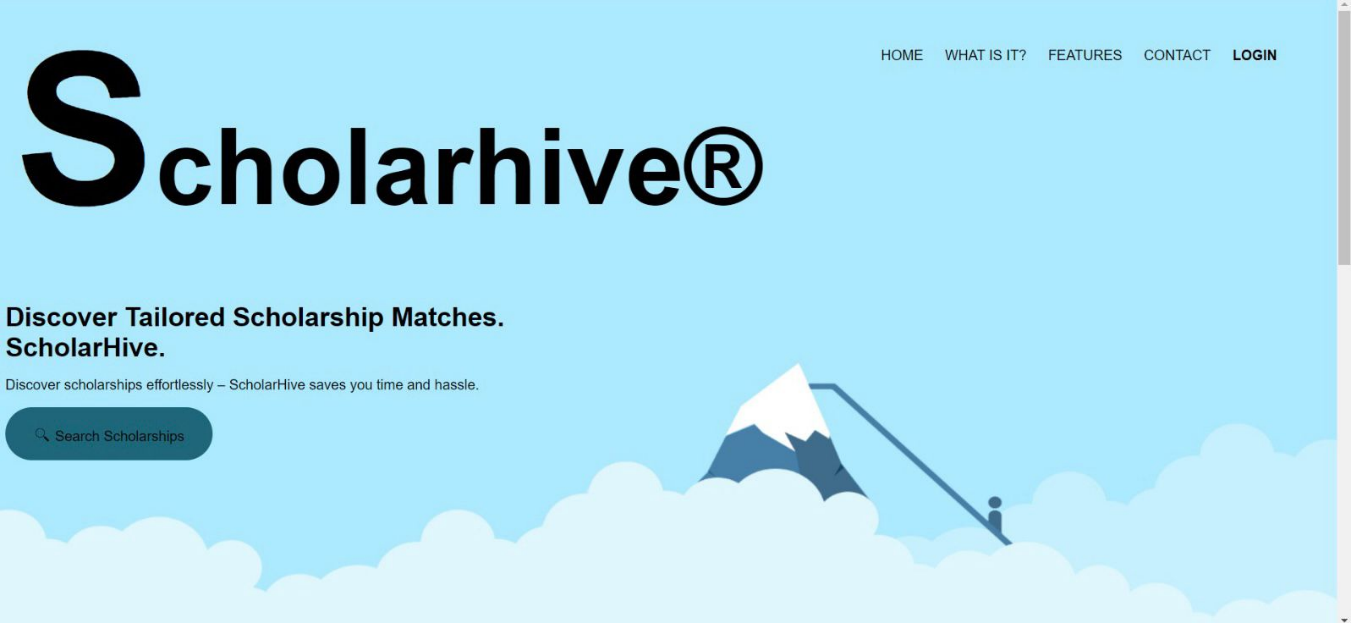
Overall, the application architecture is designed to provide a seamless scholarship management experience for students, utilizing modern technologies for web development and database management.

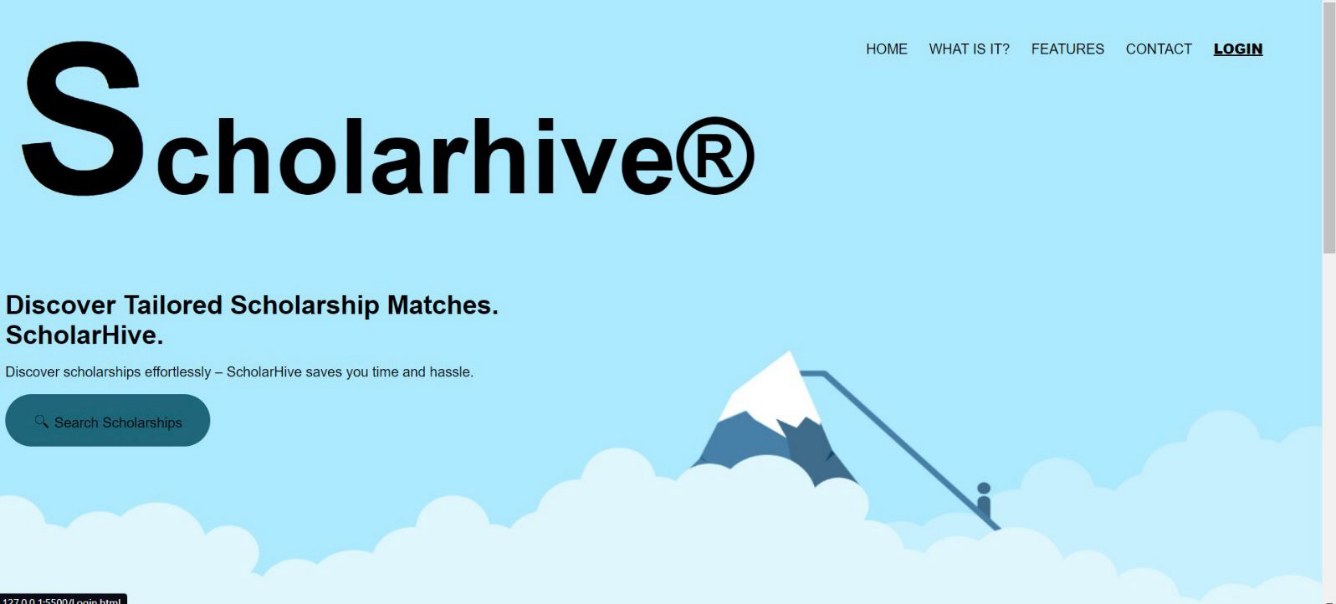
# GUI Design

## User Interface Design

##### Home page

Acts as a navigation page.

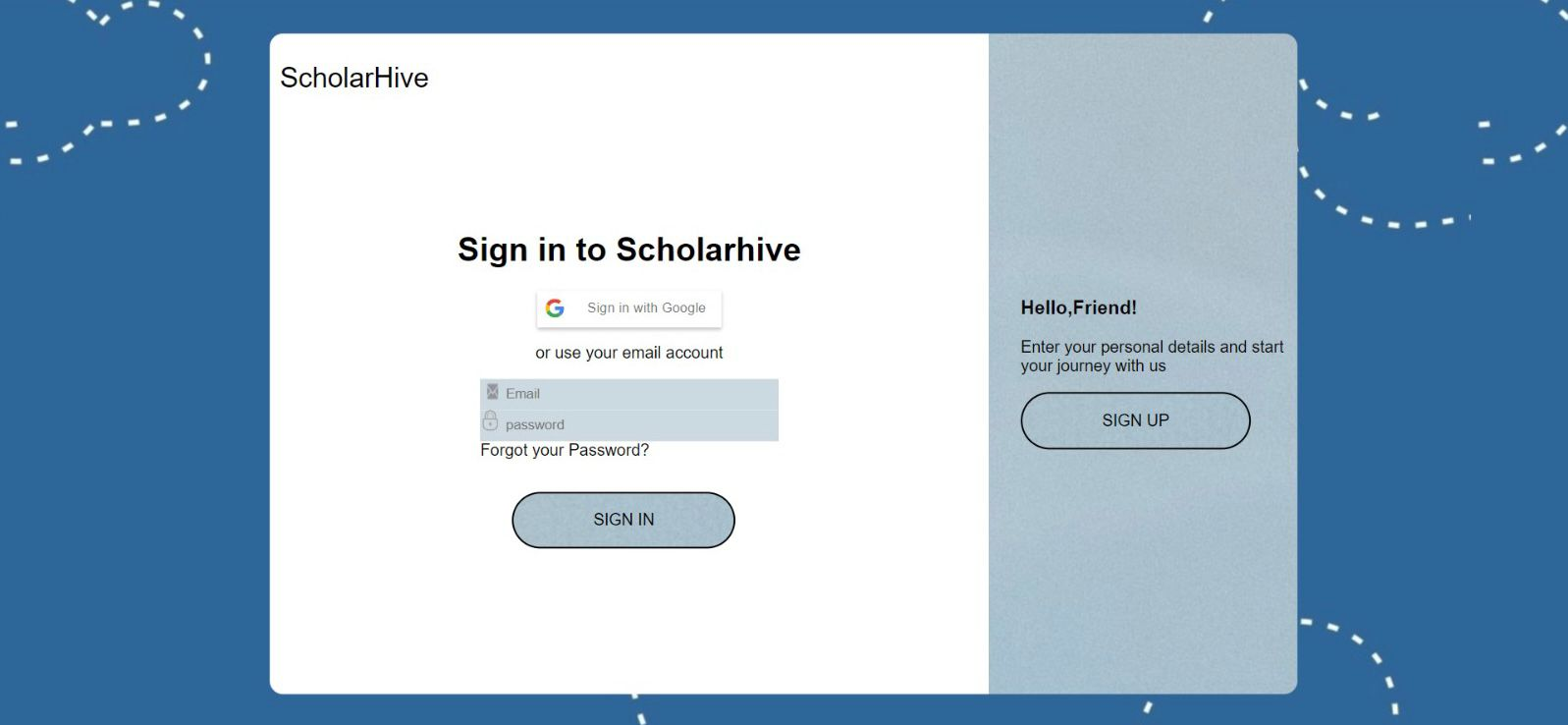




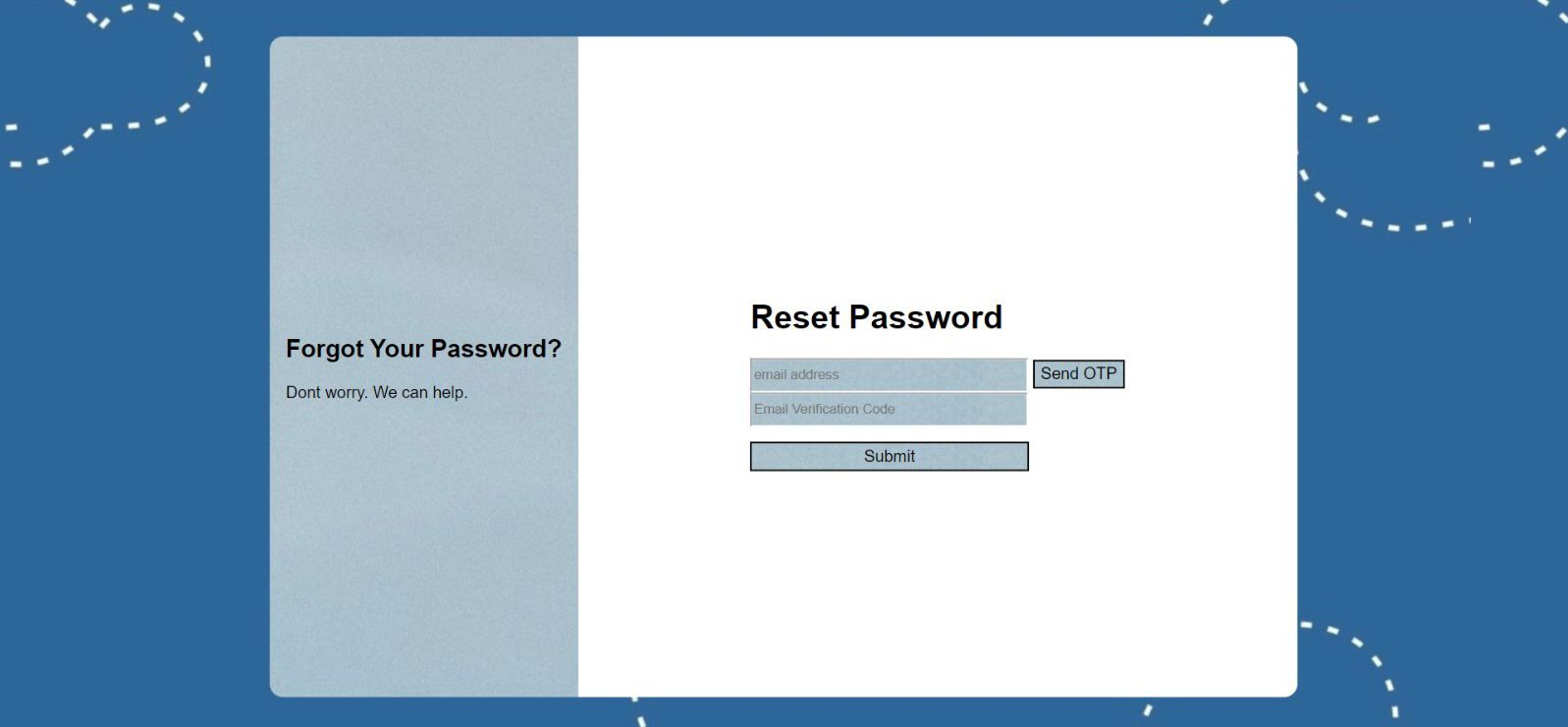
When the cursor is hovered over the navigation bar, cursor changes.

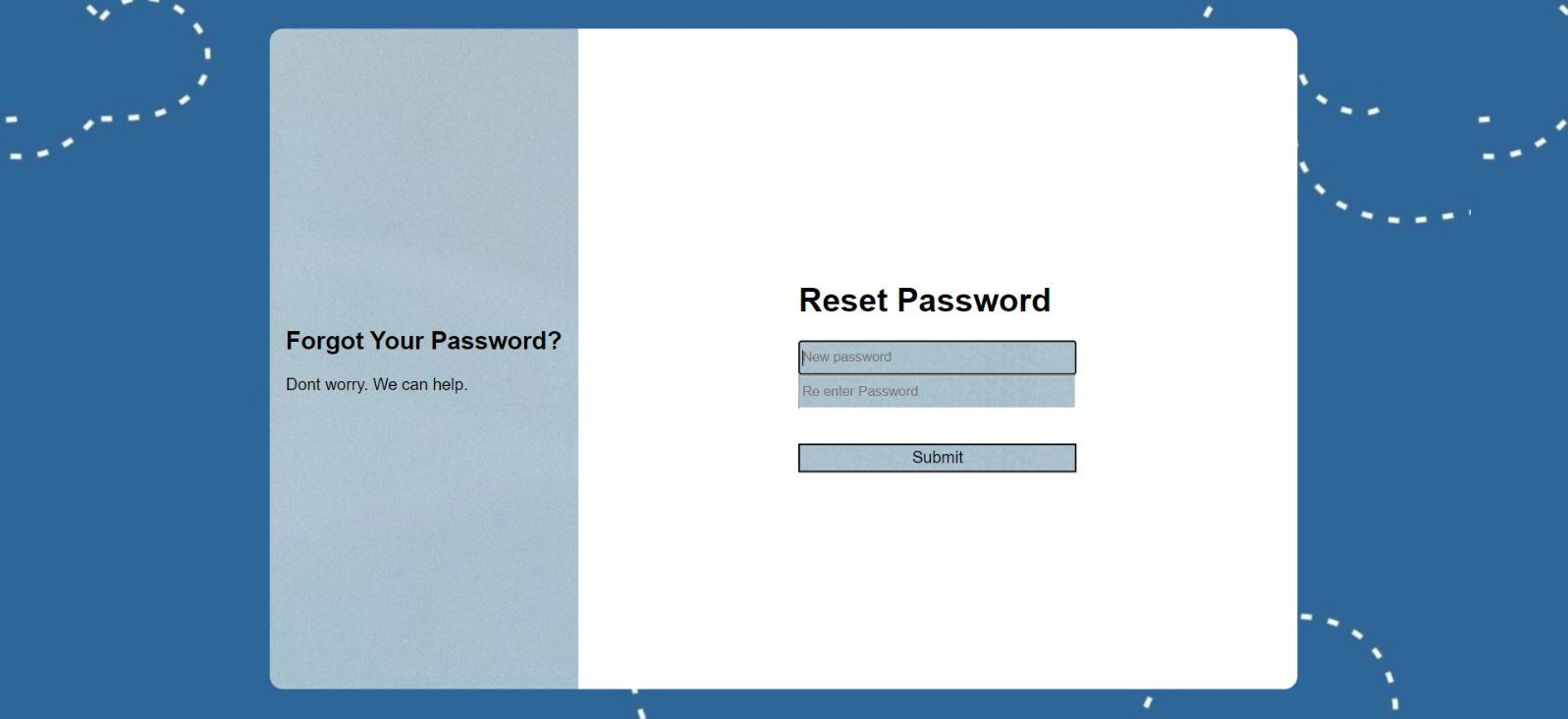
##### Login Page

The user can login using email id and password. There is an option of forgot password where the user can login using entering the user’s email id.



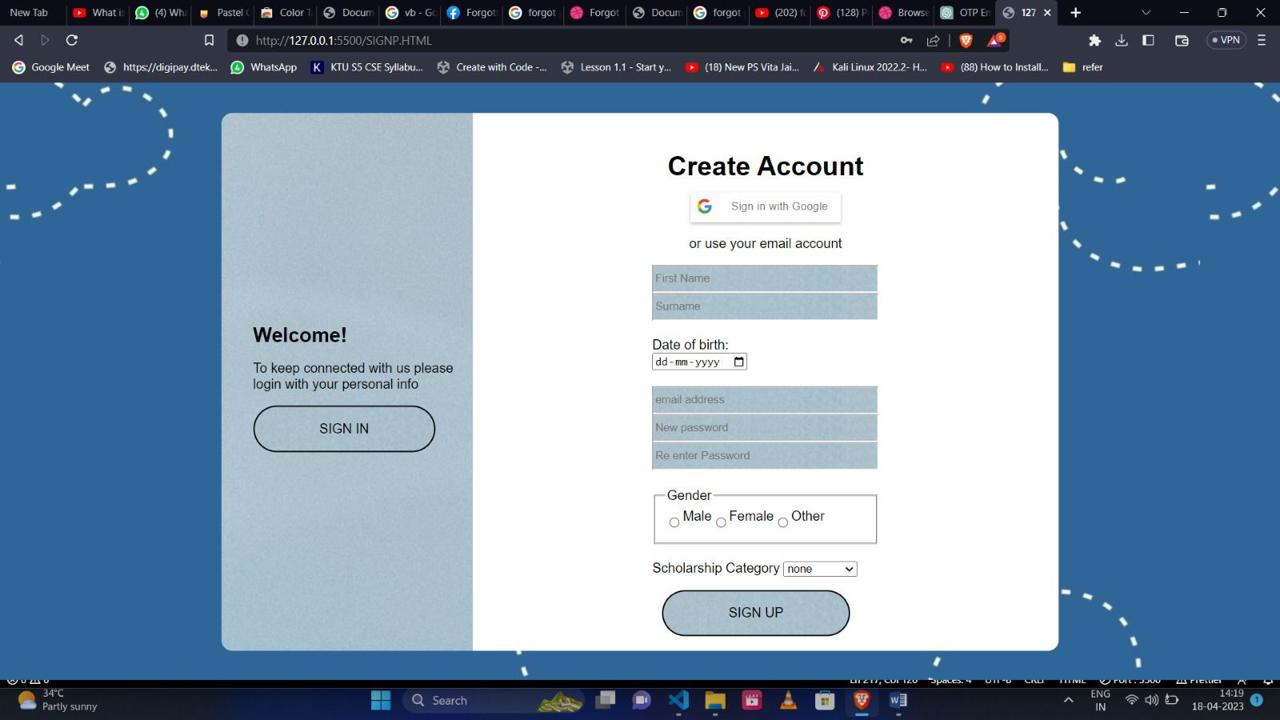
The user can reset the password if the user has forgotten the password, which will ask to enter the email id and the user will have to enter the otp sent to that email id.

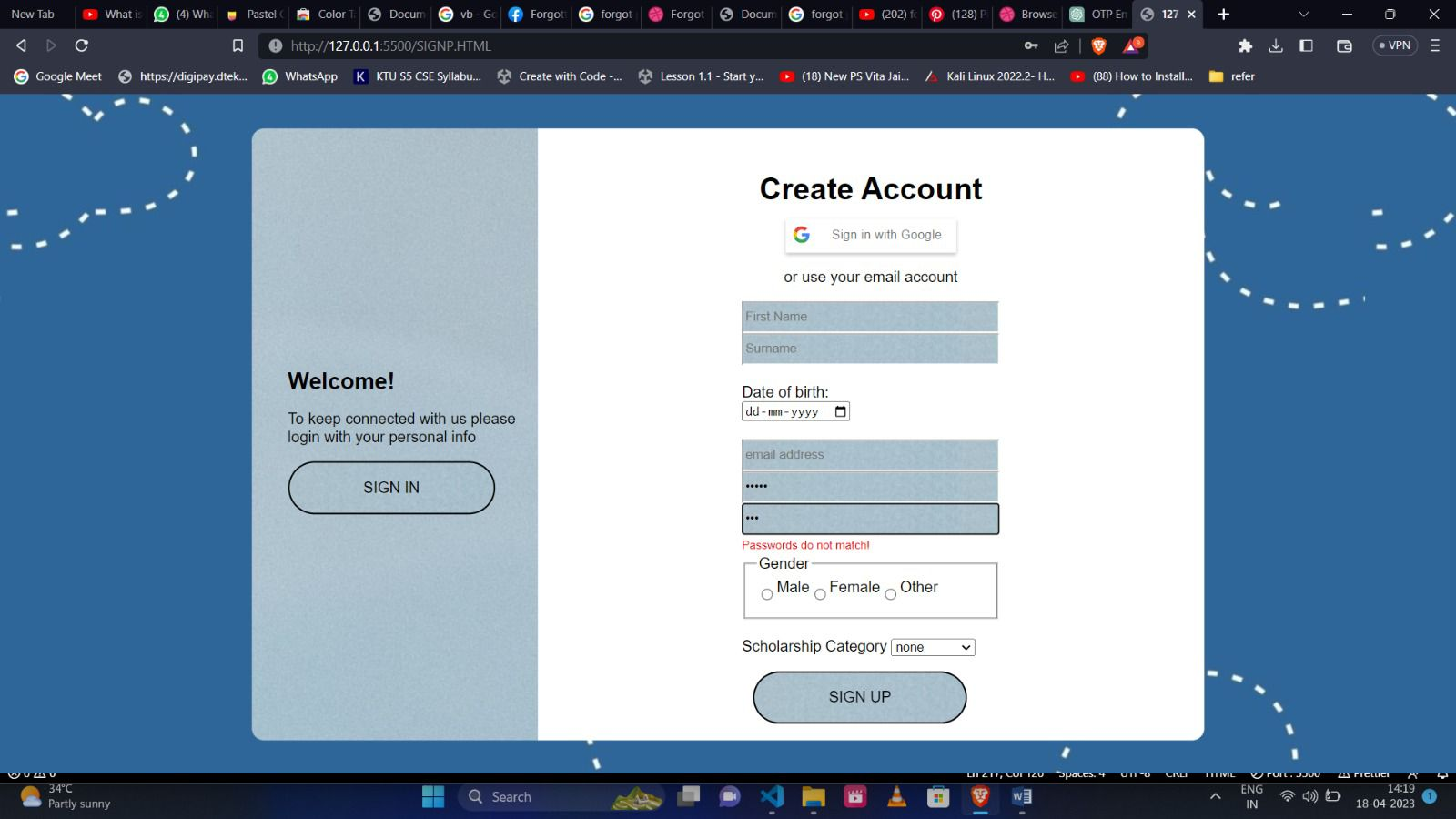
j****

****Now the user can reset the password.

**c.Sign up page**

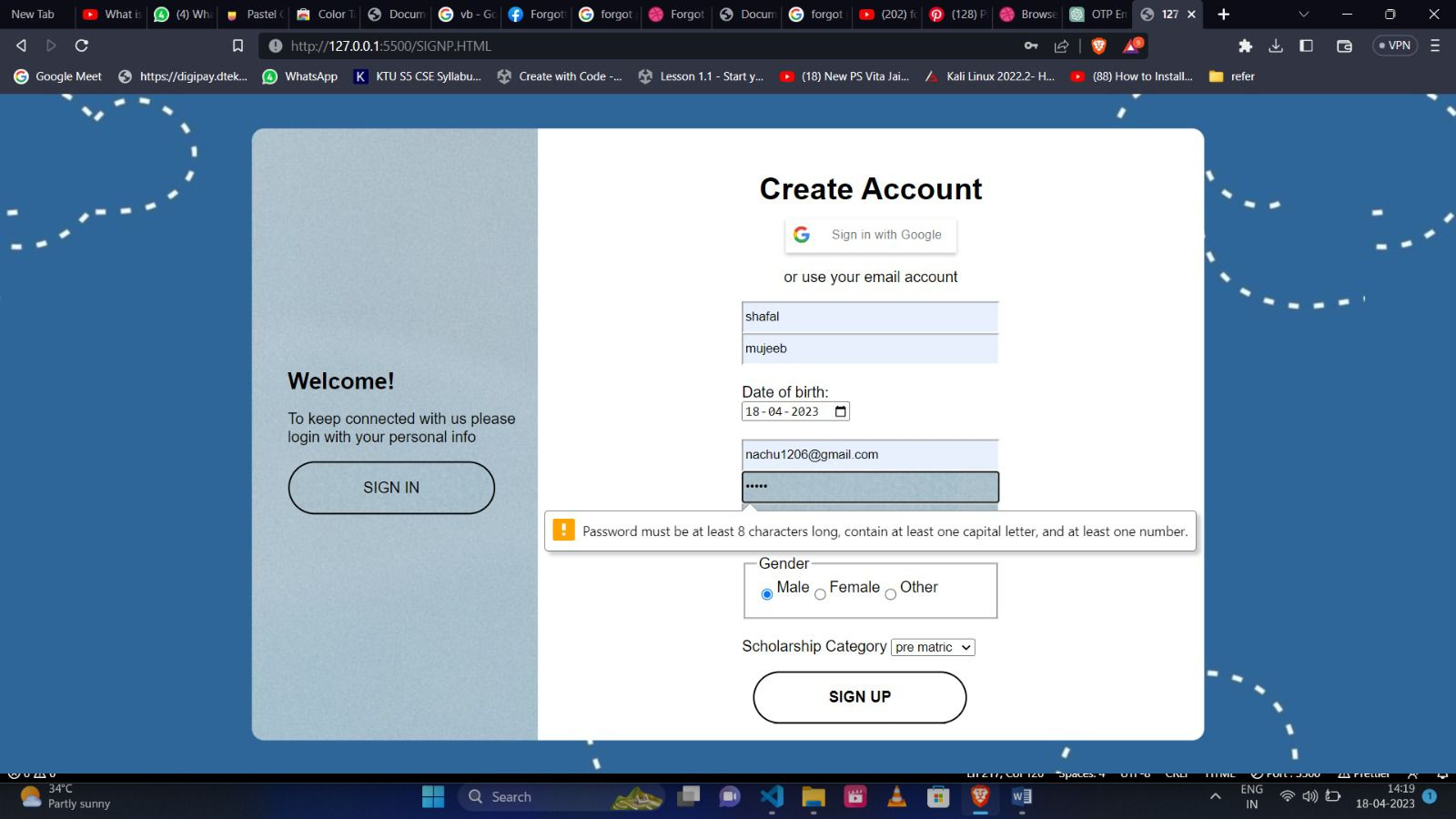
* The home screen includes three buttons, sign in, sign up, sign in with google which can be used to get to the user’s account.
* The user can sign up using first name, last name, date of birth, gender, email id, category of scholarship and also create a new password.
* You can also go back to the homepage.

****



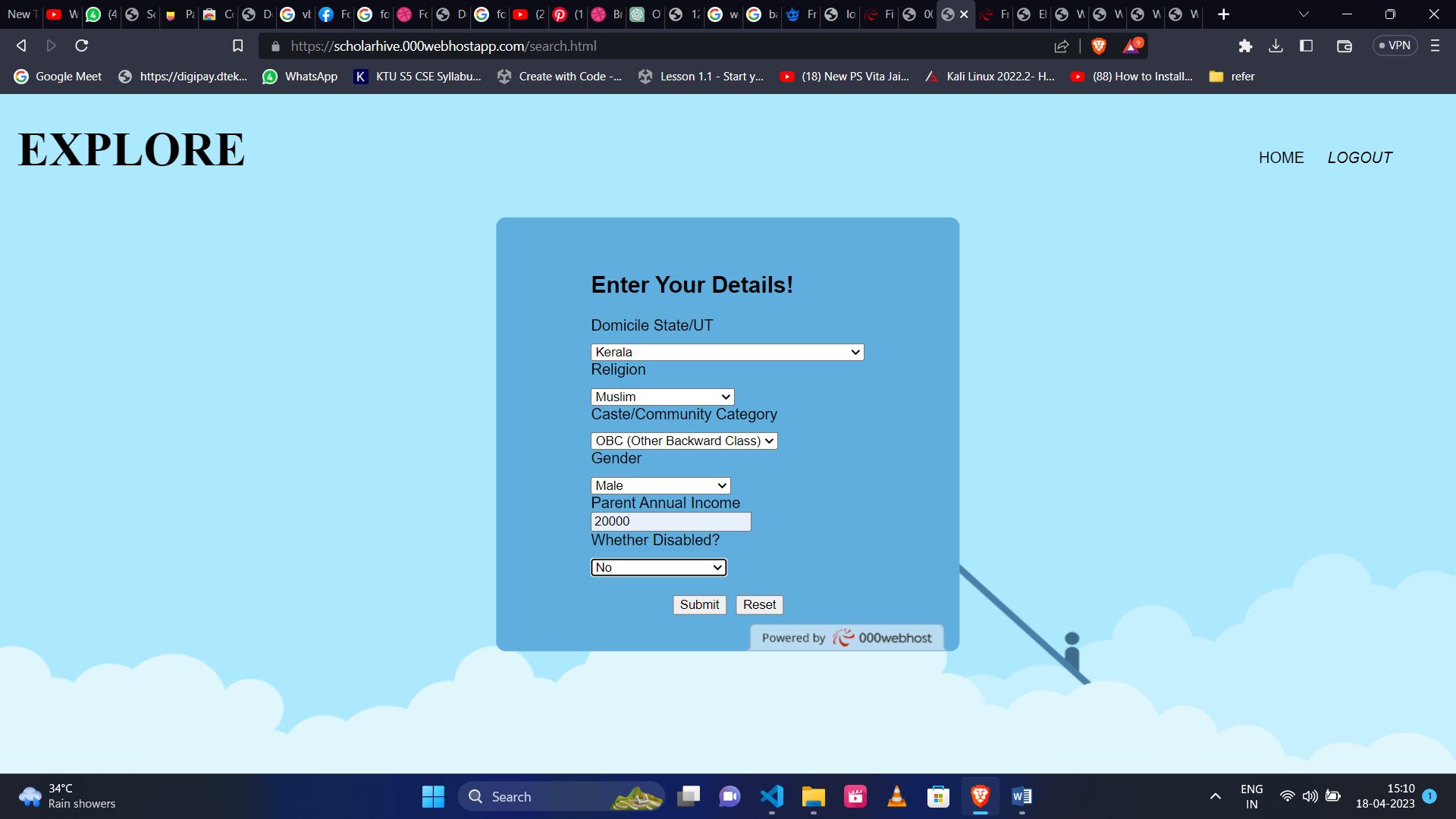
When you create a new password, it will check whether the two entered passwords are same or not.

The password should be a minimum of 8 characters.

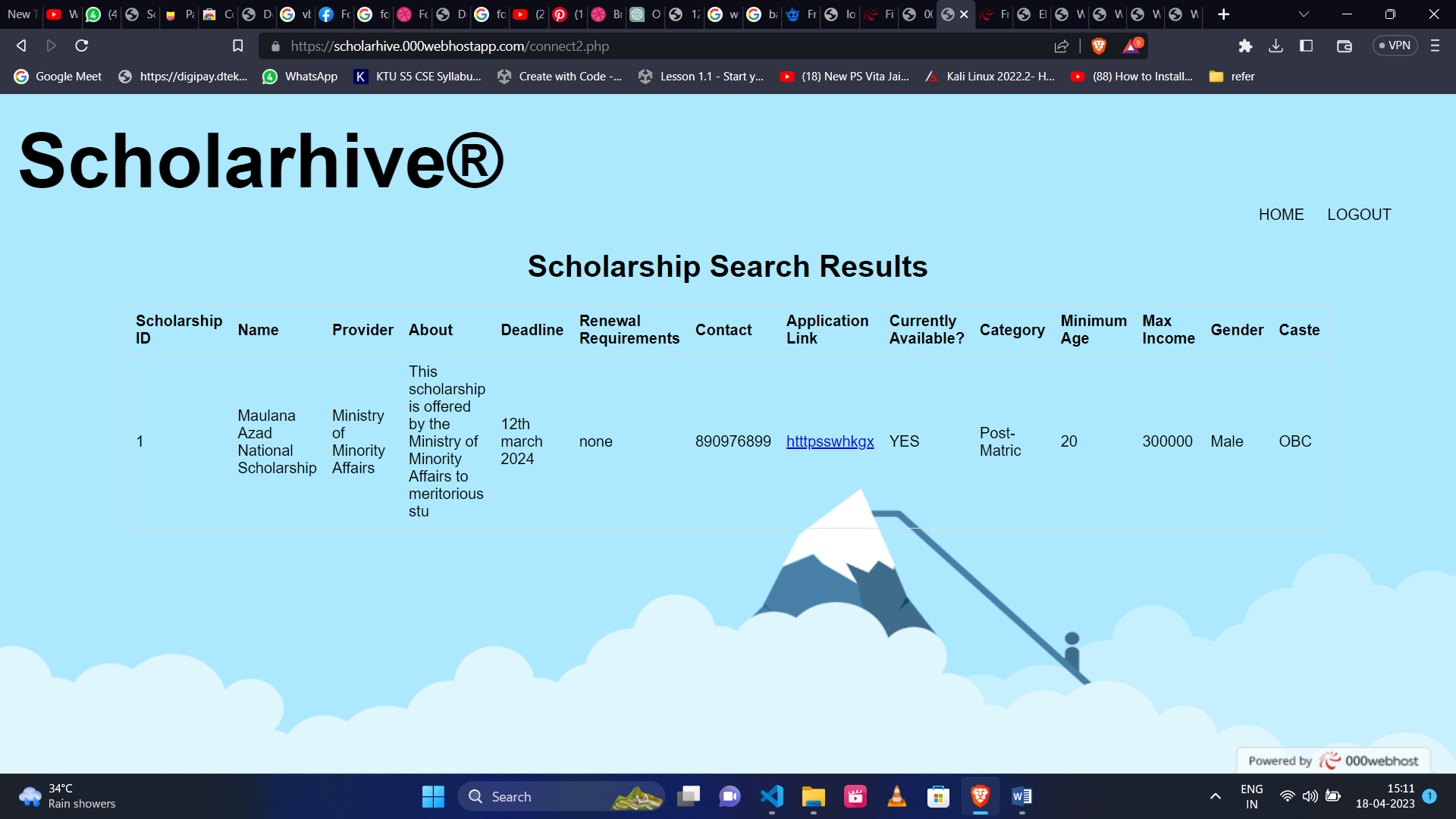


1. **Search Page**

The user can search scholarships using this page.



The sorted scholarships are listed in a new page according to the details entered by the user.



## Visual Design

* The GUI design will use high-contrast colors and larger font sizes for better visibility and readability.
* Icons and buttons will have descriptive labels in addition to their visual representation.
* Animations are used to make the page more aesthetically appealing.

## Navigation Design

* The application will use a consistent navigation structure throughout
* Navigation will be designed with clear and descriptive labels to aid users with limited vision.

# API Design

APIs, or Application Programming Interfaces, are a set of protocols and tools that allow different software applications to communicate with each other. In the context of web development, APIs enable client-side applications to interact with server-side applications and databases, allowing for the exchange of data and functionality. In your project, you are using two main APIs: PHPMyAdmin and VSCode. PHPMyAdmin is a web-based interface used to manage MySQL databases, while VSCode is a code editor that provides extensions for web development, including debugging tools, syntax highlighting, and code completion.

**5.1. PHP MyAdmin API**

* The PHPMyAdmin API will be used to manage the scholarship data in the system's database. This API will consist of the following endpoints:
* Authentication endpoint: Used to authenticate users and grant access to the scholarship database.
* Scholarship retrieval endpoint: Used to retrieve scholarship data from the database.
* Scholarship creation endpoint: Used to create new scholarship entries in the database.
* Scholarship update endpoint: Used to update existing scholarship entries in the database.
* Scholarship deletion endpoint: Used to delete scholarship entries from the database.

**5.2. VS Code API**

* The VS Code API will be used to manage the codebase of the scholarship management system. This API will consist of the following endpoints:
* Authentication endpoint: Used to authenticate users and grant access to the codebase.
* Code retrieval endpoint: Used to retrieve code from the code repository.
* Code update endpoint: Used to update the codebase with new features, bug fixes, and other changes.
* Code version control endpoint: Used to manage the different versions of the codebase.

The scholarship management system will also use other APIs such as the email API for communication with scholarship applicants and external stakeholders, and the payment API to process scholarship disbursements.

**6.Technology Stack:**

The technology stack for the Scholarship Management System (SMS) can vary based on the specific requirements and preferences of the development team. However, here is the potential technology stack for the project:

Front-end:

* + HTML, CSS, JavaScript for creating the user interface
  + React or Angular as a front-end framework for building the web application
  + Redux or other state management library for managing application state

Back-end:

* + A server-side programming language such as Python or PHP for implementing the application logic
  + A web application framework such as Django or Laravel for building the application
  + An API for integrating with external systems and services

Database:

* A relational database management system such as MySQL or PostgreSQL for storing data related to the SMS
* An object-relational mapping (ORM) tool such as SQLAlchemy or Django ORM for interacting with the database

Deployment:

* + A cloud service such as AWS, Azure or Google Cloud for hosting the application
  + Docker for containerization and deployment of the application
  + Kubernetes or other orchestration tool for managing containers and scaling the application

1. **Component Design**

* User Authentication Component:

This component is responsible for verifying the identity of the user and authenticating their login credentials. It includes features such as user login, password reset, and user profile management.

* Scholarship Management Component:

This component is responsible for managing scholarships, including creating new scholarships, updating existing scholarships, and deleting scholarships. It includes features such as scholarship search, application submission, and status tracking.

* Scholarship Application Component:

This component is responsible for managing scholarship applications, including receiving and reviewing scholarship applications, accepting or rejecting applications, and notifying students of their application status.

* Scholarship Recommendation Component:

This component is responsible for generating scholarship recommendations for students based on their academic achievements, financial needs, and other relevant factors.

* Student Profile Component:

This component is responsible for managing student profiles, including collecting and storing student information, managing academic records, and tracking scholarship application status.

1. **Algorithm Design**
2. Start the program and prompt the user to enter their search criteria.
3. Receive the user input and validate it.
4. Connect to the scholarship database using the appropriate API.
5. Retrieve the scholarships that match the user's search criteria from the database.
6. Display the list of scholarships to the user.
7. Prompt the user to select a scholarship from the list.
8. Receive the user's scholarship selection.
9. Retrieve the details of the selected scholarship from the database.
10. Display the details of the selected scholarship to the user.
11. If the user is interested in the scholarship, prompt them to apply.
12. Receive the user's application information.
13. Validate the user's application information.
14. If the application is valid, save the application to the database.
15. Display a message to the user indicating that their application has been submitted.

The algorithm design in pseudocode:

startProgram():

searchCriteria = getUserInput()

scholarships = searchDatabase(searchCriteria)

displayScholarships(scholarships)

selectedScholarship = getUserSelection()

scholarshipDetails = getScholarshipDetails(selectedScholarship)

displayScholarshipDetails(scholarshipDetails)

if userInterested:

applicationInfo = getUserApplicationInfo()

if validateApplication(applicationInfo):

saveApplication(applicationInfo)

displayMessage(applicationSubmitted\_

**9.Data Design**

**1. postmatric\_government table:**

* Scholarship ID (primary key)
* Scholarship Name
* Scholarship Description
* Scholarship Amount
* Eligibility Criteria
* Application Deadline
* Scholarship Status

The postmatric\_government table stores information about post-matriculation government-funded scholarships. Each scholarship has a unique ID, a name, a description, an amount, eligibility criteria, an application deadline, and a status.

**2.prematric\_government table:**

* Scholarship ID (primary key)
* Scholarship Name
* Scholarship Description
* Scholarship Amount
* Eligibility Criteria
* Application Deadline
* Scholarship Status

The prematric\_government table stores information about pre-matriculation government-funded scholarships. Each scholarship has a unique ID, a name, a description, an amount, eligibility criteria, an application deadline, and a status.

**3.postmatric\_private table:**

* Scholarship ID (primary key)
* Scholarship Name
* Scholarship Description
* Scholarship Amount
* Eligibility Criteria
* Application Deadline
* Scholarship Status

The postmatric\_private table stores information about post-matriculation privately-funded scholarships. Each scholarship has a unique ID, a name, a description, an amount, eligibility criteria, an application deadline, and a status.

**4.prematric\_private table:**

* Scholarship ID (primary key)
* Scholarship Name
* Scholarship Description
* Scholarship Amount
* Eligibility Criteria
* Application Deadline
* Scholarship Status

The prematric\_private table stores information about pre-matriculation privately-funded scholarships. Each scholarship has a unique ID, a name, a description, an amount, eligibility criteria, an application deadline, and a status.

In each of these tables, the Scholarship ID field serves as the primary key, which uniquely identifies each scholarship record in the table. The other fields store information about the scholarship, such as the scholarship name, description, amount, eligibility criteria, application deadline, and status.

1. **Error Handling Design**

* Network errors: In case of a slow or unstable internet connection, the system will display an appropriate error message to the user, informing them about the issue and suggesting that they try again later. If possible, the system will automatically retry the action that caused the error once the network connection is stable. For example, if a user is trying to submit an application and the network connection is lost, the system will automatically save the application as a draft so the user can complete it later.
* Error reporting: Whenever an error or crash occurs, the system will automatically report the issue to the server. The system will also provide an error message to the user, explaining what went wrong and offering suggestions for how to resolve the issue.
* Hardware error checking: The system will have a feature that allows users to test their hardware (e.g. microphone, webcam, etc.) before using the application. If there is an issue with the hardware, the system will display an error message and suggest troubleshooting steps to the user.
* Clear and concise error messages: Error messages will be clear, concise, and easy to understand. They will be spoken aloud by the system in a way that is easy to comprehend for users with varying levels of education and language proficiency.
* Handling of special characters and slang: When users input special characters or slang that the system cannot recognize, the system will prompt the user to input the word or phrase character by character. The system will also have a built-in database of common slang terms and abbreviations to help identify and translate these terms correctly.

By implementing these error handling strategies, the scholarship management system will provide a more seamless and user-friendly experience for its users.

1. **Performance Design**

The performance of the website is critical to ensure a smooth and eﬃcient user experience. The following performance design considerations have been taken into account during the development of the website:

● Data Retrieval and Storage:

The performance of data retrieval and storage is critical to ensure that the scholarship management system can handle a large volume of scholarship applications and data. We will be using a high-performance database to store scholarship applications and data. We will also be optimizing the database queries to ensure that they are fast and efficient.

● Backend Framework:

A robust backend framework will be selected for the scholarship management system to ensure fast and reliable performance. The selected framework will be optimized for better performance by using caching techniques and optimizing the database queries. We will also implement load balancing to distribute the workload among servers to handle a high volume of requests.

● Frontend Framework:

A high-performance frontend framework will be selected to ensure that the scholarship management system is lightweight and fast. We will optimize the application's UI to ensure that it is responsive and provides a smooth user experience. Additionally, we will be using code splitting and lazy loading to optimize the application's performance by loading only the required components.

● Caching:

Caching is a crucial technique to improve application performance. We will use caching to store frequently accessed data, such as scholarship details and user data, to reduce the number of database queries and improve application performance. This will ensure that the scholarship management system is fast and responsive.

● Load Testing and Optimization:

We will be conducting regular load testing to ensure that the scholarship management system can handle a large volume of scholarship applications and data. Any performance issues that are identified during testing will be addressed immediately to ensure a smooth and efficient user experience. We will also be optimizing the application to ensure that it is as fast and efficient as possible.

● Automated Monitoring and Alerting:

Automated monitoring and alerting will be implemented to identify and resolve any performance issues in real-time. This will help to ensure that the scholarship management system is always performing optimally and is available to users without any performance issues.

1. **Security Design**

* Secure authentication: The scholarship management system will use secure authentication mechanisms to ensure that only authorized users can access the system. This might include password-based authentication, two-factor authentication, or biometric authentication, depending on the level of security required.
* Role-based access control: Access to different parts of the scholarship management system will be restricted based on the user's role. For example, a student will have access only to their scholarship application and status, while an administrator will have access to all the scholarship applications.
* Storage security: The scholarship management system will ensure that all sensitive data is stored securely. This might include encryption of data at rest and in transit, secure backup mechanisms, and access controls to ensure that only authorized users can access the data.
* Compliance with data privacy regulations: The scholarship management system will comply with all relevant data privacy regulations, such as GDPR, to ensure that user data is collected, processed, and stored in a secure and compliant manner.
* Scam protection: The scholarship management system will incorporate features to protect against scams and fraud. For example, the system might provide guidance on how to identify and avoid scholarship scams, and it might have mechanisms in place to detect and block fraudulent scholarship applications.
* Regular security audits: The scholarship management system will undergo regular security audits to identify any vulnerabilities and ensure that the system remains secure. Any identified vulnerabilities will be addressed promptly to ensure the ongoing security of the system.

1. **Testing Design** 
   1. **Testing Approach**

The overall approach for testing the application will be a combination of manual and automated testing. Manual testing will be used to verify the user interface and user experience, while automated testing will be used to verify functionality, integration, and performance. Each feature of the application will be tested separately before testing the application as a whole.

* 1. **Testing Scope**

The scope of testing for this application will include the following types of testing:

* + 1. Functional testing: To ensure that the application meets its functional requirements
    2. Integration testing: To ensure that all components of the application work together seamlessly.
    3. Performance testing: To ensure that the application performs within acceptable limits and responds to user input within 2 seconds and loads images and data within 5 seconds
    4. Security testing: To verify that the application is secure and complies with GDPR and other privacy regulations
    5. Usability testing: To test the application's ease of use and accessibility for blind and visually impaired users.
  1. **Test Cases**

We will use the following test cases to verify that the application meets its requirements:

##### Functional Testing

a. Test Case 1 - Scholarship Application Expected Result: Users should be able to submit scholarship applications and provide all required information, including personal information, academic information, and essays. The system should validate the information and provide clear feedback to users about any missing or incorrect information.

b. Test Case 2 - Scholarship Review and Approval Expected Result: The system should allow scholarship committee members to review and approve scholarship applications based on predefined criteria. Committee members should be able to access all required information, including application details and supporting documents, and provide feedback to applicants if necessary.

c. Test Case 3 - Scholarship Renewal Expected Result: The system should allow scholarship recipients to renew their scholarships and provide updated information as necessary. The system should also validate the information and provide clear feedback to recipients about any missing or incorrect information.

**2. Integration Testing**

a. Test Case 1 - Student Information System Integration Expected Result: The scholarship management system should be able to integrate with the student information system to validate academic information and eligibility criteria.

b. Test Case 2 - Email Notification Integration Expected Result: The scholarship management system should be able to integrate with email notification systems to notify applicants and recipients of the scholarship award decision and any required actions.

c. Test Case 3 - Payment System Integration Expected Result: The scholarship management system should be able to integrate with payment systems to process scholarship awards and disburse funds to recipients.

**3.Performance Testing**

a. Test Case 1 - Application Submission Speed Expected Result: The system should be able to process scholarship applications within an acceptable time frame, even during peak application periods.

b. Test Case 2 - Scholarship Award Processing Time Expected Result: The system should be able to process scholarship awards and disburse funds to recipients within an acceptable time frame.

c. Test Case 3 - Concurrent Users Expected Result: The system should be able to handle multiple users simultaneously without a significant decrease in performance.

d. Test Case 4 - Load Testing Expected Result: The system should be able to handle a high volume of scholarship applications and awards without a significant decrease in performance.

**4.Security Testing**

a. Test Case 1 - Authentication Expected Result: The system should require users to authenticate with a strong password or other secure authentication methods before accessing scholarship information or performing any actions.

b. Test Case 2 - Access Control Expected Result: The system should enforce access controls to ensure that only authorized users can access and modify scholarship information.

c. Test Case 3 - Data Encryption Expected Result: The system should encrypt all sensitive scholarship information, including personal and academic data, to protect against unauthorized access or interception.

d. Test Case 4 - Error Handling Expected Result: The system should handle errors and exceptions securely to prevent information disclosure or other security vulnerabilities.

**5.Usability Testing**

a. Test Case 1 - Application Submission and Review Expected Result: Users should be able to submit scholarship applications and committee members should be able to review applications without confusion or difficulty.

b. Test Case 2 - Feedback on User Actions Expected Result: The system should provide clear and timely feedback to users when an action is performed, such as confirming the application submission or providing feedback on missing information.

c. Test Case 3 - Help and Documentation Expected Result: The system should provide clear and comprehensive help and documentation to guide users through the scholarship application and award process.

**13.1 Testing Tools**

We will use the following testing tools to automate and facilitate testing:

* Test Management Tool: A test management tool such as TestRail or Zephyr can be used to manage test cases, execute test runs, and track results.
* Load Testing Tool: A load testing tool such as JMeter or Gatling can be used to simulate a high number of concurrent users accessing the scholarship management system and verify its performance under heavy load.
* Security Testing Tool: A security testing tool such as OWASP ZAP can be used to perform security testing on the scholarship management system and identify potential vulnerabilities, such as injection attacks, cross-site scripting, and broken authentication.
* Web Accessibility Testing Tool: A web accessibility testing tool such as Axe or WAVE can be used to check the scholarship management system for accessibility compliance, including color contrast, keyboard navigation, and screen reader compatibility.
* Automation Testing Tool: An automation testing tool such as Selenium or Cypress can be used to automate repetitive testing tasks, such as login, user registration, and form submissions, to speed up the testing process and improve test coverage.
* Code Coverage Tool: A code coverage tool such as JaCoCo or Cobertura can be used to measure the amount of code that is covered by automated tests, helping to identify areas of the scholarship management system that require more testing.
  1. **Test Environment**
* Operating System: Windows, MacOS, Linux
* Web Browsers: Chrome, Firefox, Safari, Edge
* Database: MySQL, PostgreSQL, Oracle
* Web Server: Apache, Nginx
* Development Framework: Ruby on Rails, Django, Spring
* Test Automation Tools: Selenium WebDriver, JUnit, TestNG
  1. **Test data**
* Test student data: This can include mock student records with varying levels of scholarship eligibility and application statuses (e.g. approved, denied, pending).
* Test scholarship data: This can include mock scholarship records with varying criteria and application requirements.
* Test financial data: This can include mock financial records to test the system's ability to accurately calculate scholarship amounts based on financial need.
* Test user data: This can include mock user records with different roles and permissions (e.g.

admin, faculty, student).

* Test email templates and messages: This can include mock email templates and messages to test the system's ability to send and receive email notifications.
  1. **Acceptance Criteria**

The following acceptance criteria must be met for the application to be considered ready for release:

* All test cases have been executed and passed
* Performance benchmarks have been met, including page load times and response times
* Security requirements have been met, including data encryption and user authentication
* Usability metrics have been met, including user satisfaction and ease of use.
* Compatibility with different operating systems and devices
* Compatibility with different email providers (e.g., Gmail, Outlook, Yahoo)
* Compliance with accessibility guidelines
* Compliance with industry standards and regulations
* Integration with other assistive technologies

1. **Maintenance Design**

The following strategies will be implemented to ensure effective maintenance:

* 1. Regular updates: The application will be updated regularly to ensure that it remains compatible with the latest screen reader software and email providers. This will also include fixing any bugs that may arise and addressing user feedback.
  2. Backward compatibility: The application will be designed to maintain backward compatibility with previous versions of the application to ensure that users who have not upgraded to the latest version can continue to use the application without interruption.
  3. User feedback: A mechanism will be put in place to gather user feedback, which will be used to inform future updates and improvements to the application.
  4. Data backup and recovery: Regular backups of user data will be performed to ensure that

user data is not lost in the event of a system failure. A recovery plan will also be in place to restore user data in the event of a catastrophic failure.

* 1. Security updates: The application will be monitored regularly for security vulnerabilities, and updates will be made as necessary to ensure that the application remains secure and protects user data.
  2. User training: As the application is updated and new features are added, user training materials will be updated to ensure that users are aware of these changes and know how to use the application effectively.

1. **Deployment Design**

For the scholarship management project, the deployment design will involve several steps to ensure that the application can be easily accessed and used by its intended users. The deployment process will involve the following steps:

1. Hosting: The scholarship management application will be hosted on a cloud server such as Amazon Web Services (AWS) or Microsoft Azure. This will ensure that the application is accessible to users at any time and from any location.
2. Platform Compatibility: The application will be designed to work on both web and mobile platforms. The web version of the application will be accessible through a web browser on a desktop or laptop computer. The mobile version of the application will be available for download on the Google Play Store and Apple App Store.
3. Database: The scholarship management application will use a cloud-based database such as Amazon RDS or Azure SQL Database. This will ensure that the application data is secure and can be accessed from anywhere with an internet connection.
4. Security: The application will employ industry-standard security practices to protect user data and prevent unauthorized access. This will include secure storage and transmission of user data, as well as strict access control measures for the application's backend services.
5. Scalability: The application will be designed to work with a scalable cloud infrastructure. This will allow the application to handle increasing numbers of users and requests without compromising performance or availability.
6. Monitoring: The application's cloud infrastructure will be monitored regularly to ensure that it is operating optimally. This will involve monitoring the application's performance, availability, and user feedback to identify and resolve any issues that arise.

**Appendix A: Record of Changes**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Description of change** |
| 1.0 | 06/04/2023 | Suryajith | Added System and Application Architecture |
| 1.0 | 04/04/2023 | Nazal | Added Data and Algorithm Design |
| 1.0 | 02/04/2023 | Ebthisam | Started GUI Design and Testing Design |
| 1.0 | 17/04/2023 | Shafal | Edited Component and Error Handling Design |
| 1.0 | 12/04/2023 | Ebthisam | Updated Maintenance Design |
| 1.0 | 08/04/2023 | Nazal | Edited Performance Design part |
| 1.0 | 03/04/2023 | Suryajith | Added Security Design |