

A Project Report On
Unsung Angels - A Web Application For
Specially Abled Children

By

Payal Save (2021510060)
Vyankatesh Wable(2021510067)

Under the guidance of
Internal Supervisor

Prof. Pallavi Thakur



Department of Master Of Computer Application
Sardar Patel Institute of Technology
Autonomous Institute Affiliated to Mumbai University
2020-21

CERTIFICATE OF APPROVAL

This is to certify that the following students

Payal Save (2021510060)
Vyankatesh Wable (2021510067)

Have satisfactorily carried out work on the project
entitled

**“Unsung Angels- A Web Application
For Specially Abled Children”**

Towards the fulfilment of project, as laid down
by
Sardar Patel Institute of Technology
during year
2021-22.

Project Guide:
Prof. Pallavi Thakur

PROJECT APPROVAL CERTIFICATE

This is to certify that the following students

**Payal Save (2021510060)
Vyankatesh Wable (2021510067)**

Have successfully completed the Project report on

**“Unsung Angels - A Web Application For
Specially Abled Children”,**

which is found to be satisfactory and is approved

at

SARDAR PATEL INSTITUTE OF TECHNOLOGY,
ANDHERI (W), MUMBAI

INTERNAL EXAMINER

EXTERNAL EXAMINER

HEAD OF DEPARTMENT

PRINCIPAL

Contents

| | |
|--|----|
| Abstract | i |
| Objectives | i |
| List Of Figures | ii |
| List Of Tables | ii |
| 1 Introduction | 1 |
| 1.1 Problem Definition | 1 |
| 1.2 Objectives and Scope | 1 |
| 1.2.1 Objectives | 1 |
| 1.2.2 Scope | 1 |
| 1.3 Existing System | 2 |
| 1.4 Proposed System | 2 |
| 1.5 System Requirements | 4 |
| 2 Software Requirement Specification (SRS) and Design | 5 |
| 2.1 Purpose | 5 |
| 2.2 Definition | 5 |
| 2.3 Overall Description | 5 |
| 2.3.1 Product Functions | 5 |
| 2.3.2 User Characteristics | 6 |
| 3 Project Analysis and Design | 7 |
| 3.1 Methodologies Adapted | 7 |
| 3.2 Modules | 8 |
| 3.2.1 Activity diagram | 8 |
| 3.2.2 Communication Diagram | 9 |
| 3.2.3 Work Breakdown Structure | 9 |
| 3.2.4 PERT Chart | 10 |
| 3.2.5 Gantt Chart | 10 |
| 3.2.6 Use-Case | 11 |
| 4 Project Implementation and Testing | 17 |
| 4.1 Login and Register | 17 |
| 4.2 Home - Landing View | 19 |
| 4.3 Create New Profile | 20 |
| 4.4 Fill Profile details | 21 |
| 4.5 View Profile | 22 |
| 4.6 Apply for Scholarships | 23 |
| 4.7 Scholarship Details | 24 |
| 4.8 Personal Details Form to Apply for Scholarship | 25 |
| 4.9 Acknowledgement Page | 26 |
| 4.10 Email gratification | 27 |

| | | |
|----------|-----------------------------------|-----------|
| 4.11 | Search For Scholarships | 28 |
| 4.12 | View Profiles | 29 |
| 4.13 | Feedback form | 30 |
| 4.14 | App.js | 31 |
| 4.15 | App.js | 31 |
| 4.16 | Firebase | 32 |
| 4.17 | Index.js | 32 |
| 4.18 | Header | 33 |
| 4.19 | Packages Installed | 33 |
| 4.20 | Navigation Bar | 34 |
| 4.21 | SignIn Page | 34 |
| 4.22 | Sign Up Page | 35 |
| 4.23 | Contact Us Page | 35 |
| 4.24 | Scholarships Details | 36 |
| 4.25 | Search for Scholarships | 36 |
| 5 | Test Cases | 37 |
| 6 | Limitations | 38 |
| 7 | Future Enhancements | 38 |
| 8 | User Manual | 39 |
| 9 | Bibliography | 40 |
| 9.1 | Web References | 40 |

Abstract

The Unsung Angels Web Application is a web application which caters to the needs of the students with special needs by providing them a platform to apply for various scholarship schemes.

It facilitates them with different scholarships schemes specially curated for them which in turn saves them from the hassle of finding authentic sites for scholarships.

With this web app, the users can efficiently create their profiles, search for particular scholarships, view the profile details, send feedback messages, apply for a specific scholarship, etc.

Objectives

The Web based application 'Unsung Angels Web App' is

- To provide the students with special needs a user friendly platform for the scholarship process.
- To provide a convenient and easy platform for the students with sound features.
- To provide a facility to create their profiles with the required details meant for the scholarships.
- To provide a platform to make them aware of the various scholarship schemes curated for them.

List of Figures

| | |
|---|----|
| 3.1.1Diagrammatic Representation of Waterfall Model | 7 |
| 3.2.1Activity Diagram | 8 |
| 3.2.2Communication Diagram | 9 |
| 3.2.3Work Breakdown Structure | 9 |
| 3.2.4PERT Chart | 10 |
| 3.2.5Gantt Chart | 10 |
| 3.2.6Use-Case Diagram | 11 |
| 4.1.1 Login | 17 |
| 4.1.2 Register | 18 |
| 4.2.1 Home View | 19 |
| 4.3.1Create New Profile | 20 |
| 4.4.1Profile Form | 21 |
| 4.5.1Profile View | 22 |
| 4.6.1Apply for Scholarships Page | 23 |
| 4.7.1Scholarship Details View | 24 |
| 4.8.1Personal Details Form to Avail Scholarship | 25 |
| 4.9.1Acknowledgement Page | 26 |
| 4.10.Email gratification Page | 27 |
| 4.11.Search For Various Scholarships Using Search Bar | 28 |
| 4.12.View Profile Page | 29 |
| 4.13.Feedback Page to get in touch with us | 30 |

List of Tables

| | |
|---|----|
| 1.5.1 Hardware Requirements on Server Side | 4 |
| 1.5.2 Hardware Requirements on Client Side | 4 |
| 1.5.3 Software Requirements on Server Side | 4 |
| 1.5.3 Software Requirements on Client Side | 4 |
| 4.2.1 Use Case Table - Register | 12 |
| 4.2.2 Use Case Table - Login | 12 |
| 4.2.3 Use Case Table - Create New Profile | 13 |
| 4.2.4 Use Case Table - Edit Profile | 13 |
| 4.2.5 Use Case Table - View Profile | 13 |
| 4.2.6 Use Case Table - Apply For Scholarships | 14 |
| 4.2.7 Use Case Table - View scholarships | 14 |
| 4.2.8 Use Case Table - Search for scholarships | 14 |
| 4.2.9 Use Case Table - Fill personal details form | 15 |
| 4.2.10 Use Case Table - Get Acknowledgement | 15 |
| 4.2.11 Use Case Table - Get email gratification | 15 |
| 4.2.12 Use Case Table - Give feedback | 16 |
| 5.1 Test Case - Login and Register | 37 |
| 5.2 Test Case - Others | 37 |

1 Introduction

1.1 Problem Definition

To provide ease and convenience to the students with special needs to apply for various scholarship schemes curated for them thereby saving them from the hassle of visiting unauthentic sites regarding scholarships.

1.2 Objectives and Scope

1.2.1 Objectives

The Web based application 'Unsung Angels Web App' is

- To provide the students with special needs a user friendly platform for the scholarship process.
- To provide a convenient and easy platform for the students with sound features.
- To provide a facility to create their profiles with the required details meant for the scholarships.
- To provide a platform to make them aware of the various scholarship schemes curated for them.

1.2.2 Scope

The students with disabilities can provide his/her details required to create student profiles on the application for further process.

Through this application , the students can view various authentic scholarships schemes and search for their desirable one.

Our system is made to reduce their hassles and aims to provide them a user - friendly environment through which they can easily go through various scholarship schemes curated for them. It also provides them with instant gratifications through emails regarding the scholarships once they successfully apply for one.

1.3 Existing System

There are various sites on the internet that aim to offer scholarships for the students . However , there exists hardly any platform for the students with spe- cial needs regarding scholarships for their betterment. Each time they want to apply for a specific scholarship especially established for them , they have to go through a great deal of sites which might in some cases could be compromising. So , our system aims to eradicate these limitations.

Some of the disadvantages of existing system are as follows :

- Time consuming
Going through a lot of sites on the internet can consume ample amount of the time of the students.
- Redundancy
The students might land up on some sites which may provide them redundant information.
- Unauthenticity/Originality
As there hardly exists any platform for the such scholarships , the students might end up dealing with some unauthentic sites which can prove to be compromisng.

1.4 Proposed System

The user is the student with special needs looking for new scholarship schemes to apply through this web application. The student can thereby provide his/her details to create new user profile on the system. The system will be equipped with all the authentic scholarship schemes especially curated for them. The user can search for a specific scholarship scheme by typing the specific keywords. The user can create their profiles and then apply for a specific scholarship by viewing its details. The user can get instant gratifications through emails once they are done with successfully applying for the scholarship.

The user initially has to register on the system by providing some details such as name , email , phone number , password , etc which all are mandatory fields. Then the user can easily login with the existing email and password to enter the further process. On successful login , the user would be taken to the create and view profile page where the name of the user would be mentioned signalling his/her login.

The user can view his profile details once done with filling the form and can edit some fields if he/she wants to. He/she can also delete the created profile if need arises. The user can view other user's profiles as well.

Some of the advantages of our system are as follows :

- User Friendly

It provides attractive user interface for the students to efficiently interact with the system by providing easy navigations , pleasing colors , etc.

- Special scholarships under one roof

Users can view and apply for various scholarships which are specifically curated for the students with special needs thereby providing them a secured platform where they get to view all the authentic scholarship schemes under one roof.

1.5 System Requirements

- Hardware Requirements on Server Side

Table 1.5.1: Hardware Requirements on Server Side

| | |
|-----------|--|
| Processor | Dual Core Processor or Above |
| RAM | Minimum 4 GB RAM |
| Storage | Minimum 10 GB Hard Disk Space for smooth run |

- Hardware Requirements on Client Side

Table 1.5.2: Hardware Requirements on Client Side

| | |
|-----------|---|
| Device | Android Device with Touch Screen minimum 5" inch Display / Computer system / Laptop |
| Processor | Dual Core Processor or Above |
| RAM | Minimum 2 GB RAM |
| Storage | Minimum 250 MB Storage Space |

- Software Requirements on Server Side

Table 1.5.3: Software Requirements on Server Side

| | |
|------------------|----------------|
| Operating System | OS Independent |
| Database | Firestore |

- Software Requirements on Client Side

Table 1.5.3: Software Requirements on Client Side

| | |
|------------------|---|
| Operating System | Android/IOS Smartphone / Computer system / Laptop |
| Server | Not Required |

2 Software Requirement Specification (SRS) and Design

2.1 Purpose

The purpose of our project is to develop a web application that can help the students with special needs to apply for various scholarship schemes specifically curated for them by providing them a sound platform for scholarships.

This can save them from the hassle of visiting a great deal of sites over the internet which might not provide them with authentic information. The project serves the purpose of providing all the scholarships under one roof which can prove to be highly beneficial for them. The project also aims to provide user-friendliness , ease and convenience to the user.

2.2 Definition

To build an application for the students with special needs to apply for scholarships specifically curated for them.

2.3 Overall Description

2.3.1 Product Functions

The product function includes:

1. Authentication: Users are required to register and login using their credentials. Failing to provide proper credentials during login will not let the user avail the further process.
2. Profile: The users can create their student profile by entering the required details asked in the form.
3. Apply: The users can apply for scholarships after creating the profile.
4. View Scholarships : The user can view different scholarships which the users can apply for.
5. Search Scholarships : The user can search for a particular scholarship by entering the familiar
6. Fill Details : Once the scholarship is chosen , the user is required to fill a small form for getting gratification through his/her email.
7. Send feedback message : The user can provide his/her valuable feedback messages.

2.3.2 User Characteristics

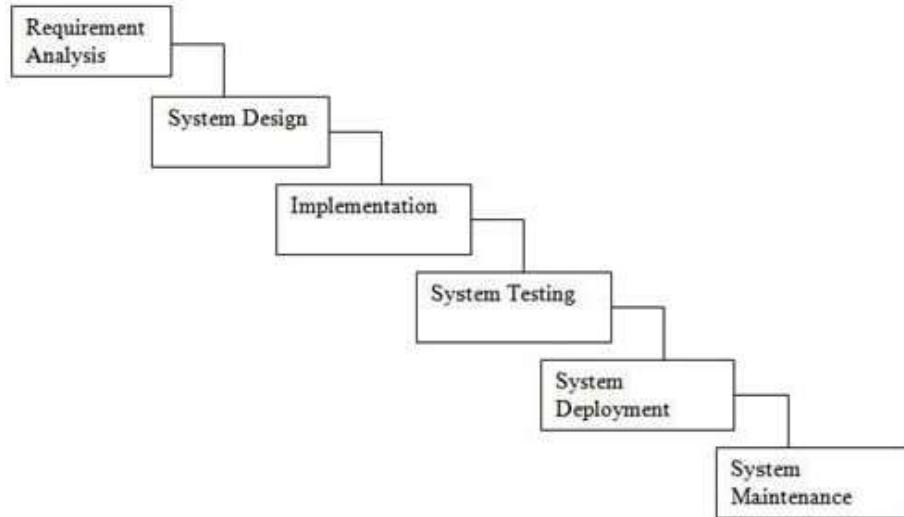
- Students with disabilities : The users of the platform are the students with disabilities who can easily apply for scholarships through this platform.
- Parents of students with disabilities : The users of the platform can be the parents of the students with disabilities who can apply for such scholarships on their behalf.
- Special school admins : The users of the platform can also be the admins of special schools who can apply for such scholarships on their behalf.

3 Project Analysis and Design

3.1 Methodologies Adapted

In Waterfall model, very less customer interaction is involved during the development of the product. Once the product is ready then only it can be demonstrated to the end users.

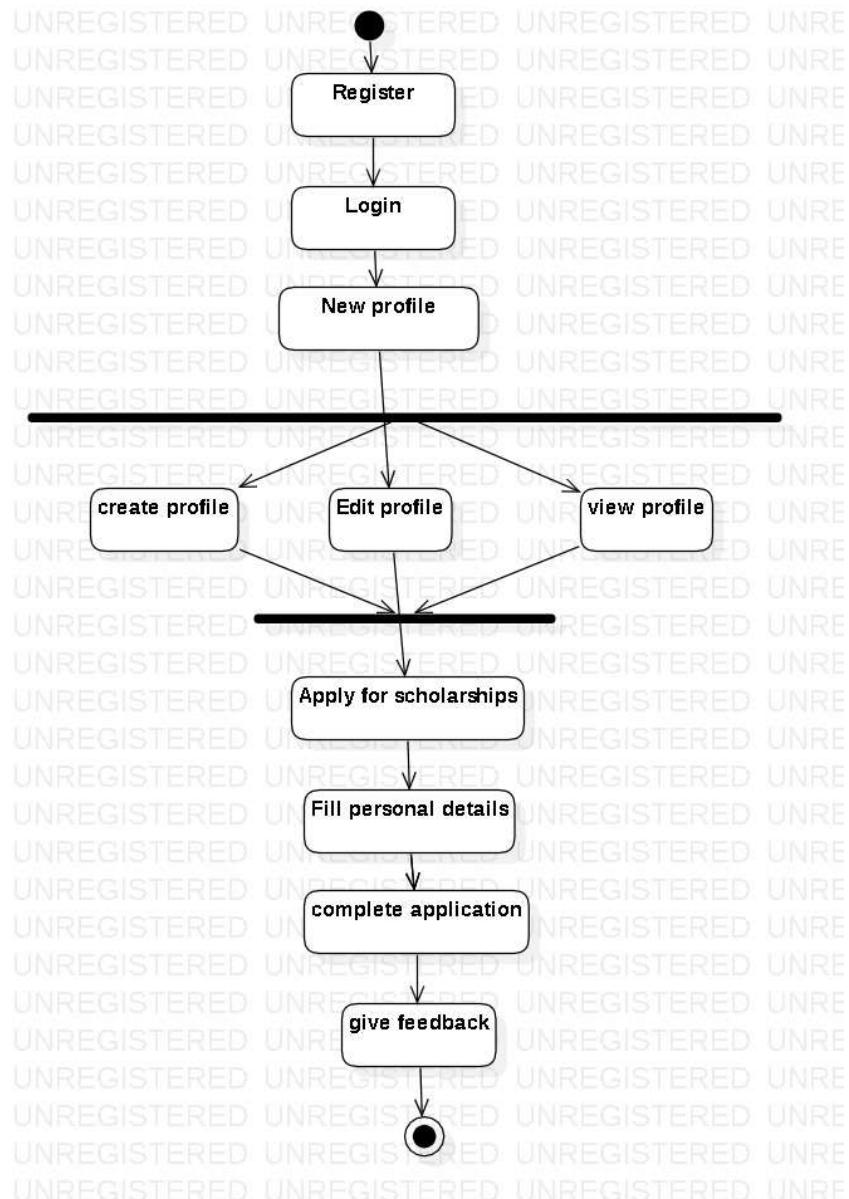
Once the product is developed and if any failure occurs then the cost of such issues is very high, because we need to update everything from document till the logic.



3.1.1: Diagrammatic Representation of Waterfall Model

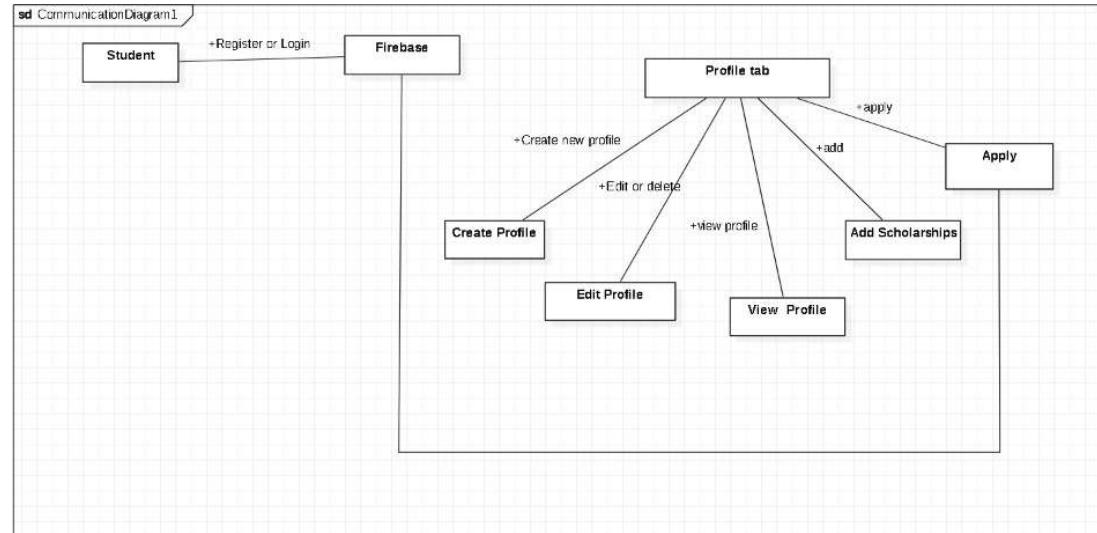
3.2 Modules

3.2.1 Activity diagram



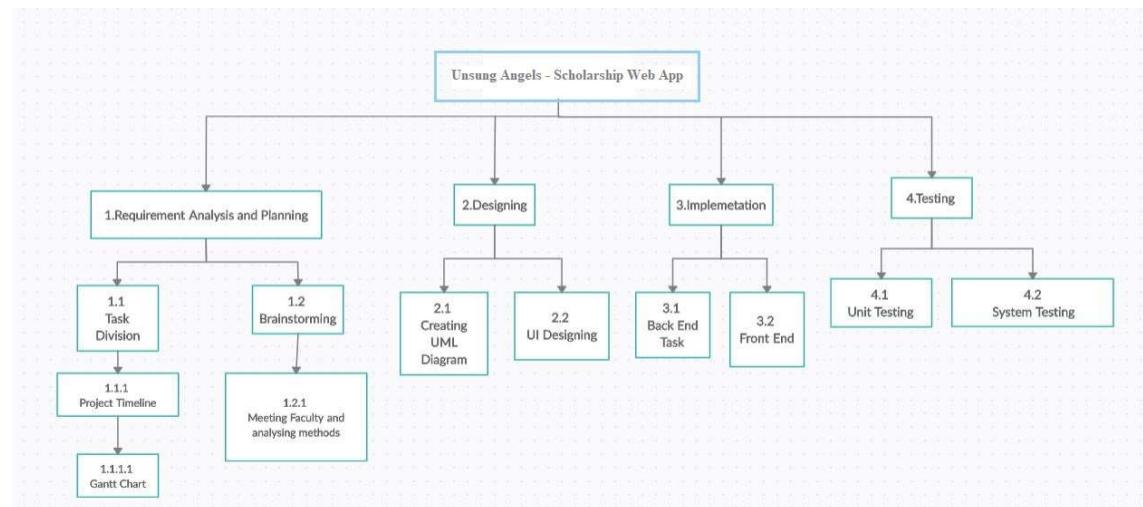
3.2.1: Activity Diagram

3.2.2 Communication Diagram



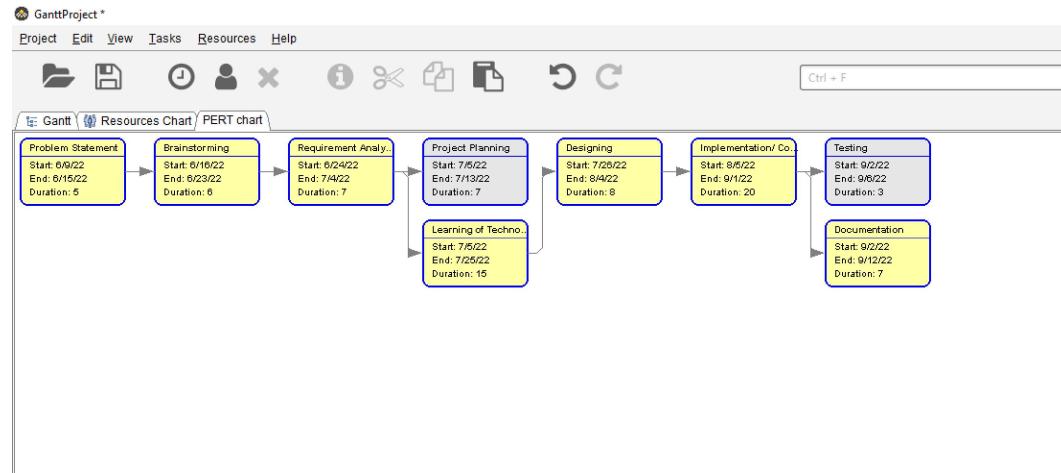
3.2.2: Communication Diagram

3.2.3 Work Breakdown Structure



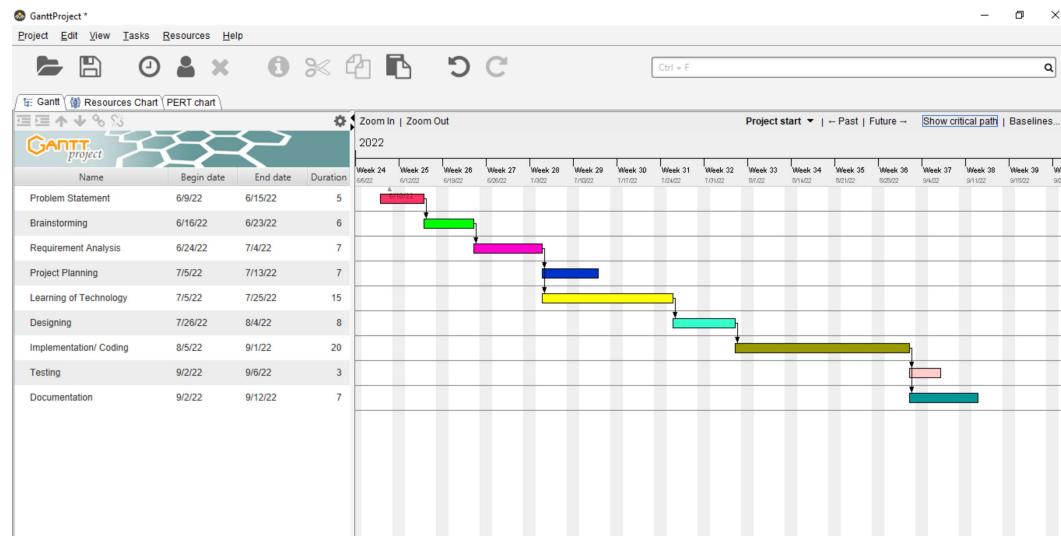
3.2.3: Work Breakdown Structure

3.2.4 PERT Chart



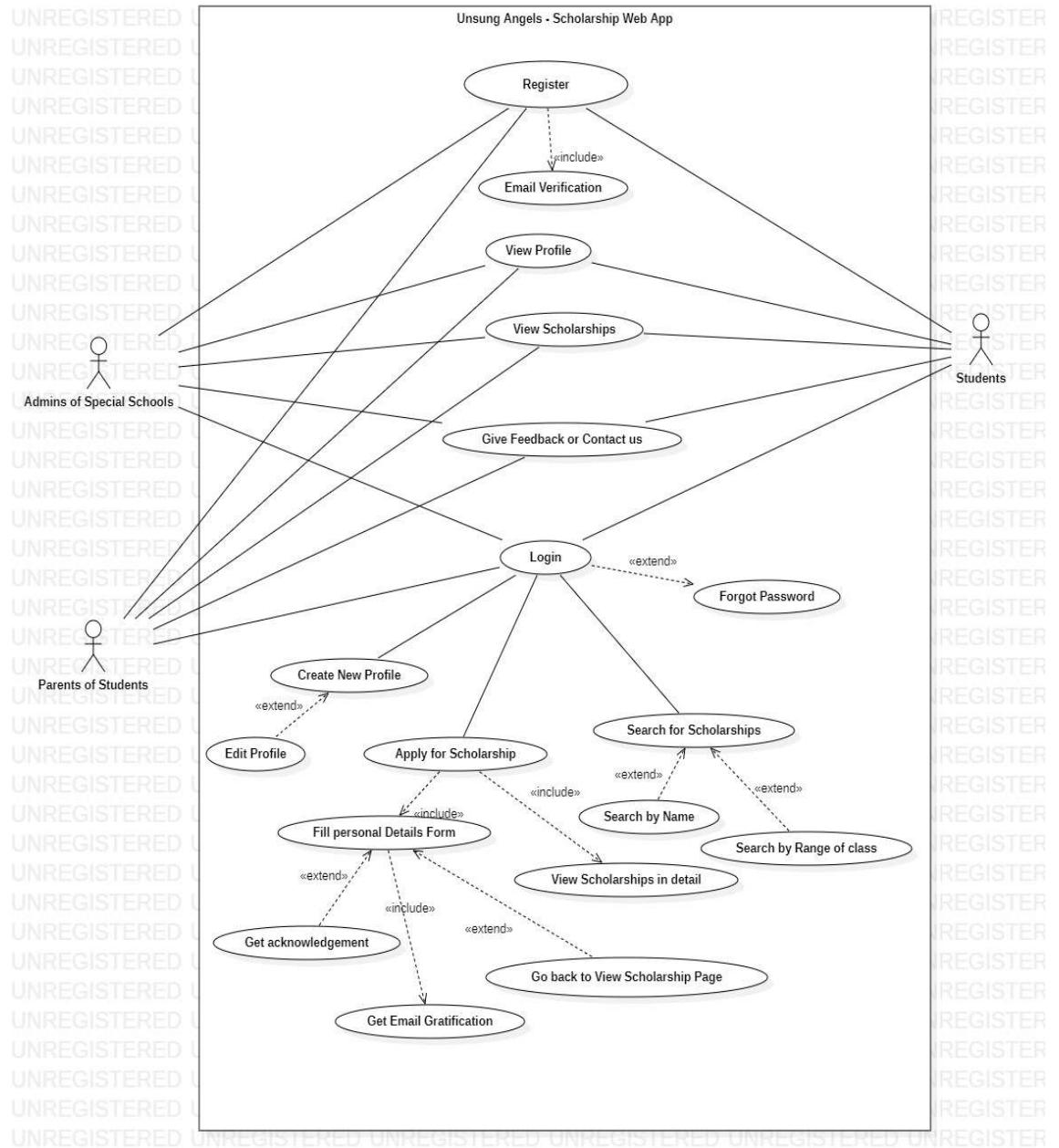
3.2.4: PERT Chart

3.2.5 Gantt Chart



3.2.5: Gantt Chart

3.2.6 Use-Case



3.2.6: Use-Case Diagram

Use Cases:

1. Register
2. Login
3. Create New Profile
4. Edit Profile
5. View Profile
6. Apply for Scholarship
7. View Scholarships
8. Search for Scholarships
9. Fill Personal Details Form
10. Get Acknowledgement
11. Get Email gratification
12. Give Feedback

Table 4.2.1: Use Case Table - Register

| | |
|----------------|---|
| Use Case ID | 1 |
| Use Case Name | Register |
| Actor | Admin, Students and Parents |
| Pre-Condition | They must register themselves first |
| Post-Condition | User can login |
| Flow of events | Login,Register or Edit Profile and View Scholarships and Apply for the same |

Table 4.2.2: Use Case Table - Login

| | |
|----------------|--|
| Use Case ID | 2 |
| Use Case Name | Login |
| Actor | Admin, Students and Parents |
| Pre-Condition | They must register themselves first |
| Post-Condition | User can view its profile and Scholarships detail and can apply for Scholarships |
| Flow of events | Login,Register or Edit profile, Apply for Scholarships and view profile and scholarships |

Table 4.2.3: Use Case Table - Create New Profile

| | |
|----------------|--|
| Use Case ID | 3 |
| Use Case Name | Create New Profile |
| Actor | Admin, Students and Parents |
| Pre-Condition | They must register and login themselves. |
| Post-Condition | Users can create new profile with required details and can view his/her profile there. |
| Flow of events | Register , login, create new profile or edit profile, view scholarships, apply for the same. |

Table 4.2.4: Use Case Table - Edit Profile

| | |
|----------------|--|
| Use Case ID | 4 |
| Use Case Name | Edit Profile |
| Actor | Admin, Students and Parents |
| Pre-Condition | They must register and login themselves. They must create a new profile as well. |
| Post-Condition | Users can edit their existing profile if they want to make some changes. |
| Flow of events | Register , login , create new profile or edit profile |

Table 4.2.5: Use Case Table - View Profile

| | |
|----------------|--|
| Use Case ID | 5 |
| Use Case Name | View Profile |
| Actor | Admin, Students and Parents |
| Pre-Condition | They must register and login themselves. They must create or edit their profile. |
| Post-Condition | Users can view their created or edited profile there. |
| Flow of events | Register, login, create new profile or edit profile, view profile |

Table 4.2.6: Use Case Table - Apply For Scholarships

| | |
|----------------|---|
| Use Case ID | 6 |
| Use Case Name | Apply For Scholarships |
| Actor | Admin, Students and Parents |
| Pre-Condition | They must register and login themselves. They must create or edit their profile. They can view their profile. |
| Post-Condition | Users can apply for scholarships by clicking on the button on the profiles created. |
| Flow of events | Register, login, create new profile or edit profile, view profile,apply |

Table 4.2.7: Use Case Table - View scholarships

| | |
|----------------|---|
| Use Case ID | 7 |
| Use Case Name | View scholarships |
| Actor | Admin, Students and Parents |
| Pre-Condition | They need to click on the view scholarships tab |
| Post-Condition | Users can view different scholarship schemes curated for them |
| Flow of events | Open the web application and click on the view scholarships tab |

Table 4.2.8: Use Case Table - Search for scholarships

| | |
|----------------|---|
| Use Case ID | 8 |
| Use Case Name | Search for scholarships |
| Actor | Admin, Students and Parents |
| Pre-Condition | They need to visit the web application and click on view scholarships tab |
| Post-Condition | Users can search for a particular scholarship by entering a specific keyword in the search bar. |
| Flow of events | Open the web application and click on the view scholarships tab , search for scholarship |

Table 4.2.9: Use Case Table - Fill personal details form

| | |
|----------------|---|
| Use Case ID | 9 |
| Use Case Name | Fill personal details form |
| Actor | Admin, Students and Parents |
| Pre-Condition | Register, login themselves. They must create new profile or edit profile, view profile, apply for scholarships |
| Post-Condition | Users can fill out the personal details for completion of the application. |
| Flow of events | Register, login, create new profile or edit profile, view profile, apply for scholarships,fill personal details |

Table 4.2.10: Use Case Table - Get Acknowledgement

| | |
|----------------|---|
| Use Case ID | 10 |
| Use Case Name | Get Acknowledgement |
| Actor | Admin, Students and Parents |
| Pre-Condition | Register, login themselves. They must create new profile or edit profile, view profile, apply for scholarships, fill personal details, click on submit. |
| Post-Condition | Users can view the acknowledgement page once they have successfully completed the application. |
| Flow of events | Register, login, create new profile or edit profile, view profile, apply for scholarships, fill personal details, click on submit |

Table 4.2.11: Use Case Table - Get email gratification

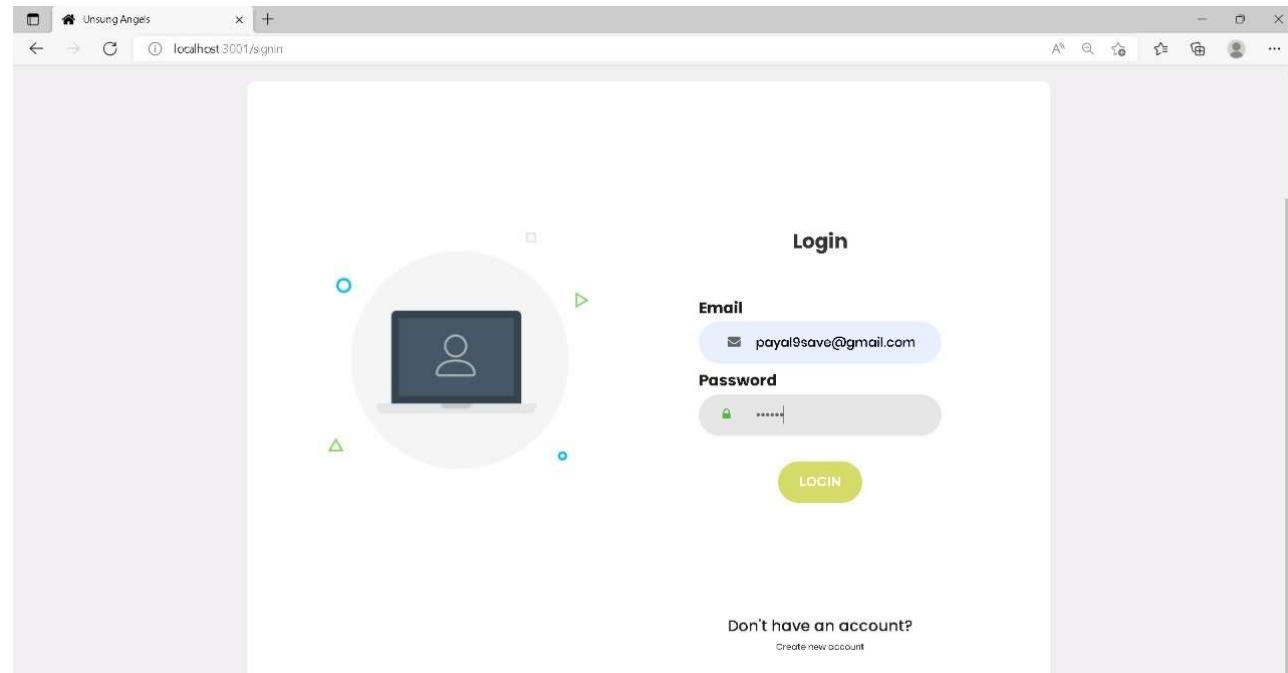
| | |
|----------------|---|
| Use Case ID | 11 |
| Use Case Name | Get email gratification |
| Actor | Admin, Students and Parents |
| Pre-Condition | They must register and login themselves. They must create new profile or edit profile, view profile, apply for scholarships, fill personal details, click on submit |
| Post-Condition | Users can receive instant email gratification on the email id specified by them regarding their information which they have filled on the web application. |
| Flow of events | Register, login, create new profile or edit profile, view profile, apply for scholarships, fill personal details, click on submit |

Table 4.2.12: Use Case Table - Give feedback

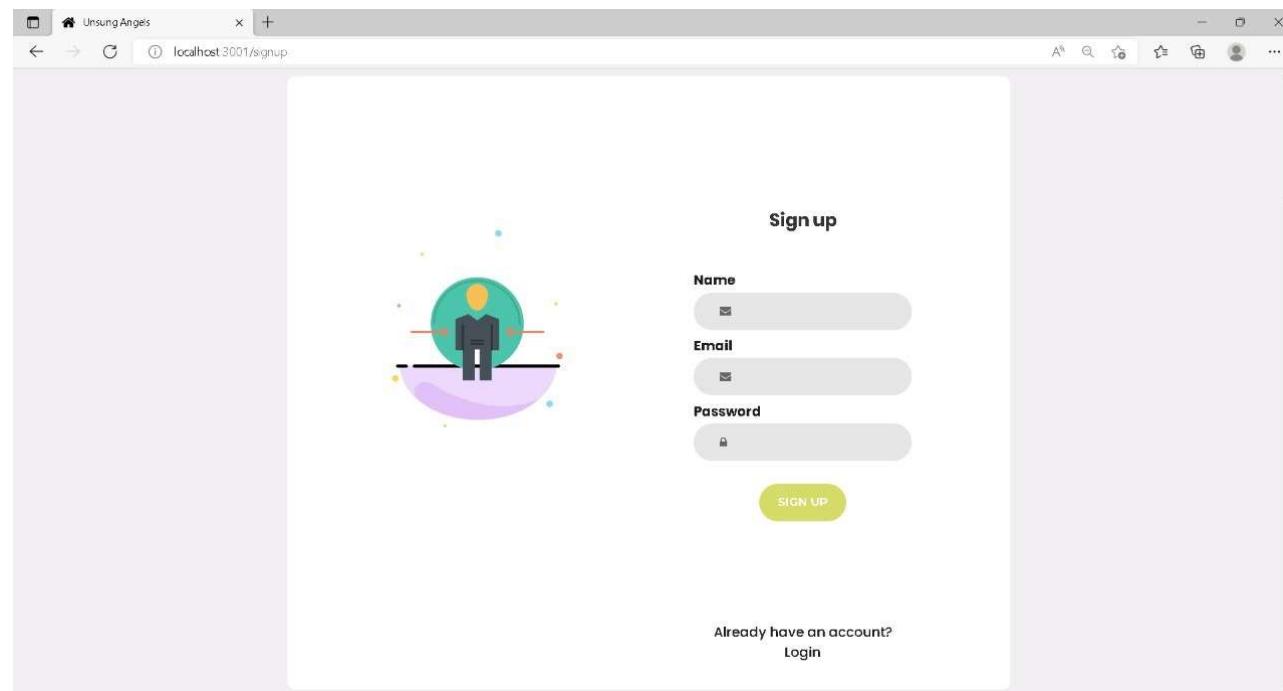
| | |
|----------------|---|
| Use Case ID | 12 |
| Use Case Name | Give feedback |
| Actor | Admin, Students and Parents |
| Pre-Condition | They must visit the web application and click on contact us tab |
| Post-Condition | Users can give their feedback or send messages regarding scholarships. |
| Flow of events | Open the web application , click on contact us tab , fill the feedback form , click on submit |

4 Project Implementation and Testing

4.1 Login and Register

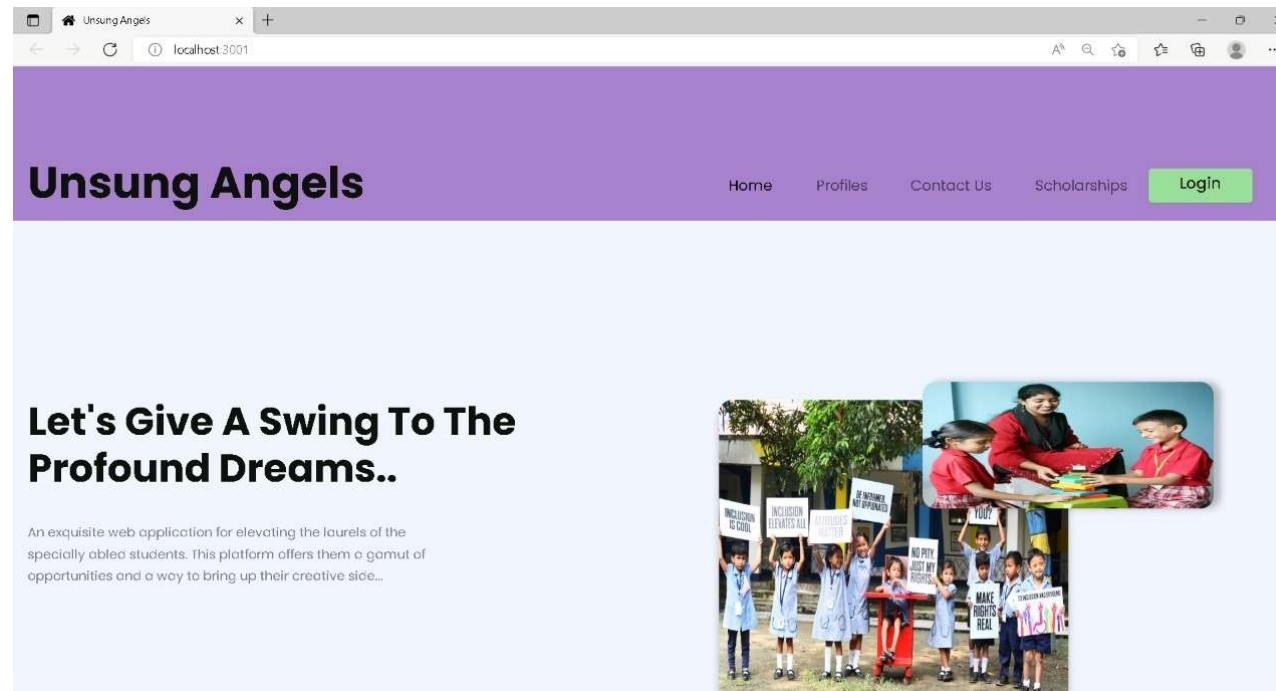


4.1.1: Login



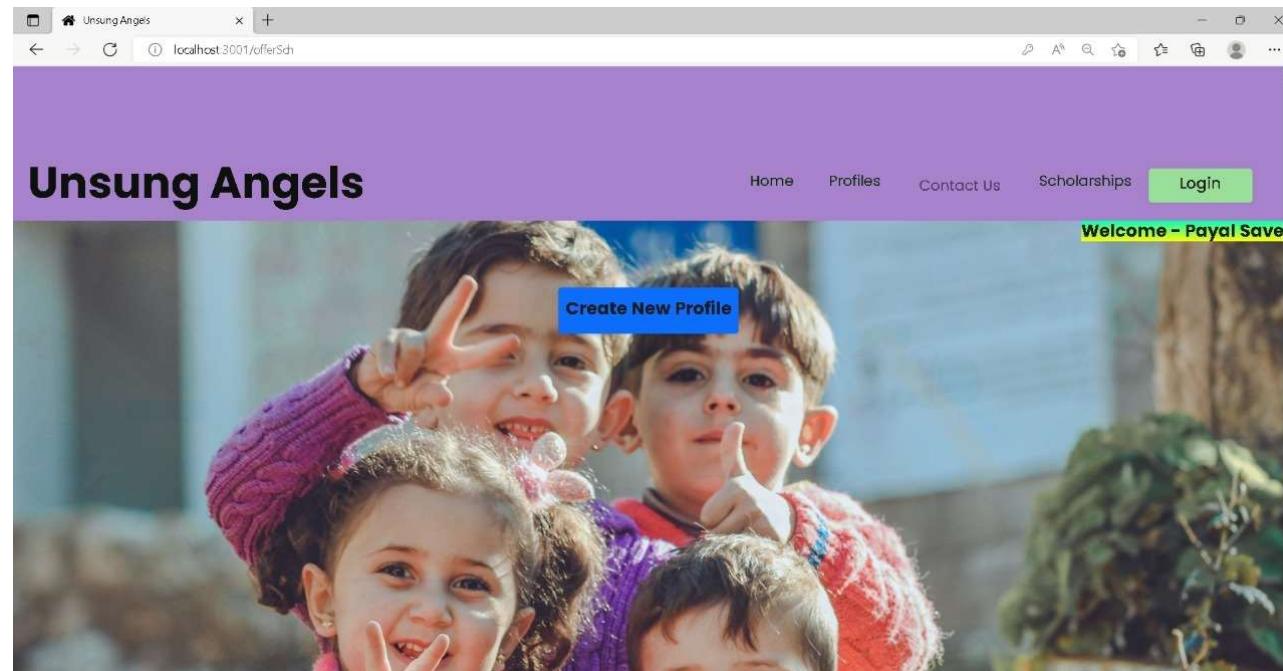
4.1.2: Register

4.2 Home - Landing View



4.2.1: Home View

4.3 Create New Profile



4.3.1: Create New Profile

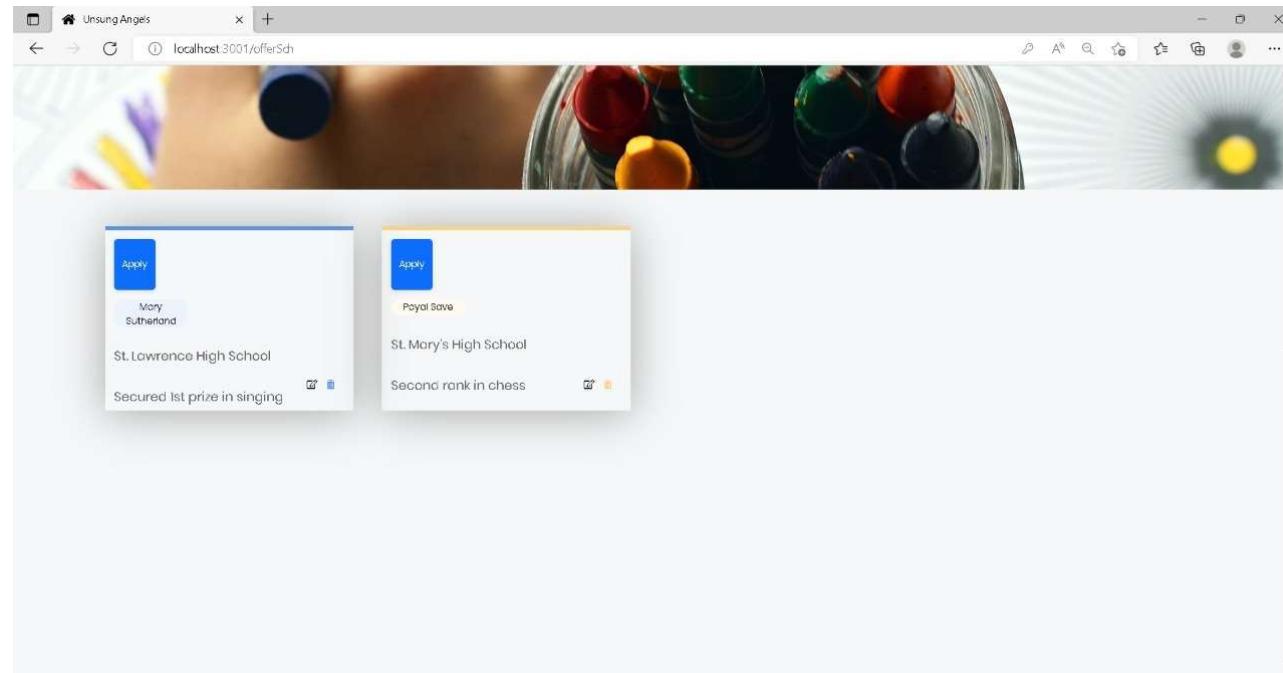
4.4 Fill Profile details

The screenshot shows a web browser window with the title 'Unsung Angels' and the URL 'localhost:3001/offersch'. The main page has a purple header with the text 'Unsung Angels' and a 'Login' button. Below the header, there is a dark background image of a person's face. On the right side of the screen, there is a green banner with the text 'Welcome - Payal Save'. A modal dialog box is open in the center, titled 'Profile'. It contains the following fields:

| | |
|----------------|--|
| Student Name | Payal Save |
| Grade | 3rd |
| Date Of Birth | 08/08/2015 |
| School | St. Mary's High School |
| School Contact | 9887657512 |
| School Email | payal@save@gmail.com |
| Image | Choose File <input type="file" value="pp1.jpg"/> |
| Document | |

4.4.1: Profile Form

4.5 View Profile



4.5.1: Profile View

4.6 Apply for Scholarships

The screenshot shows a web browser window for 'Unsung Angels' at 'localhost:3001/scholar9'. The page has a purple header with the title 'Unsung Angels' and navigation links for Home, Profiles, Contact Us, Scholarships, and a green 'Login' button. Below the header, there are several cards for different scholarship programs:

- Bharti Infratel Scholarship Program** (FOR PERSONS WITH DISABILITIES FOR UNDERGRADUATE / PROFESSIONAL / POSTGRADUATE COURSES): Includes a logo for NIEPM and a blue 'Apply here' button.
- Astha Special Scholarship** (FOR DIFFERENTLY ABLED): Includes a logo for Astha and a blue 'Apply here' button.
- Divyangan- For person with disability**: Includes a logo for the Department of Empowerment of Persons with Disabilities (Divyangjan) and a blue 'Apply here' button.
- Pragati & Saksham Scholarship Schemes**: Includes a logo for Pragati and a blue 'Apply here' button.
- Saksham Scholarship Scheme for specially abled children**: Includes a yellow logo and a blue 'Apply here' button.
- SuccessCDs Top Class Education Scholarship for Students with Disabilities**: Includes a blue logo and a blue 'Apply here' button.
- National Scholarship Portal** (Ministry Of Electronics & Information Technology, Government of India): Includes a blue logo and a blue 'Apply here' button.

4.6.1: Apply for Scholarships Page

4.7 Scholarship Details

The screenshot shows a web browser window for 'Unsung Angels' at 'localhost:3001/Details/4'. The page has a purple header with the title 'Unsung Angels' and navigation links for Home, Profiles, Contact Us, Scholarships, and a green 'Login' button. Below the header is a section titled 'PRAGATI & SAKSHAM Scholarship Schemes' featuring two small images: one of a person in a graduation cap and another of a girl. Text next to the images says 'Pragati Scholarship scheme aims at providing assistance for advancement of girls pursuing technical education.' Below this is another section for 'Saksham Scholarship scheme' with images of children and text about providing encouragement and support to specially abled children. The main content area displays the 'Pragati - Progressive Scholarship' details, stating it is a progressive portal for empowering students, especially girls with special needs. It also mentions 'Class 5-8' and 'Pragati - Progressive Scholarship' again.

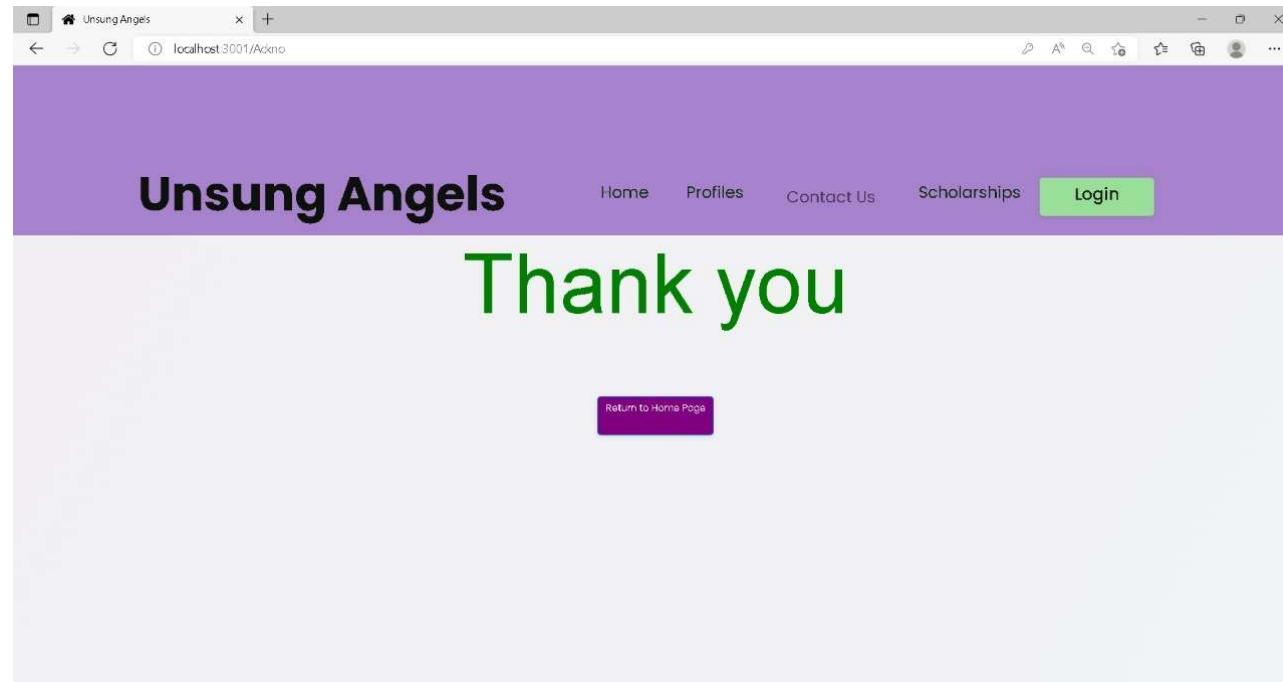
4.7.1: Scholarship Details View

4.8 Personal Details Form to Apply for Scholarship

The screenshot shows a web browser window titled "Unsung Angels" with the URL "localhost:3001/Details/4". The page is titled "Class 5-8 Pragati - Progressive Scholarship" and has a sub-instruction "Enter details". It contains four input fields: "Full Name" (Payal Save), "City" (Dahanu), "Disability Type" (Down's Syndrome), and "Email ID" (payalsave@gmail.com). Below these fields are two buttons: a blue "SUBMIT" button and a green "FINISH" button.

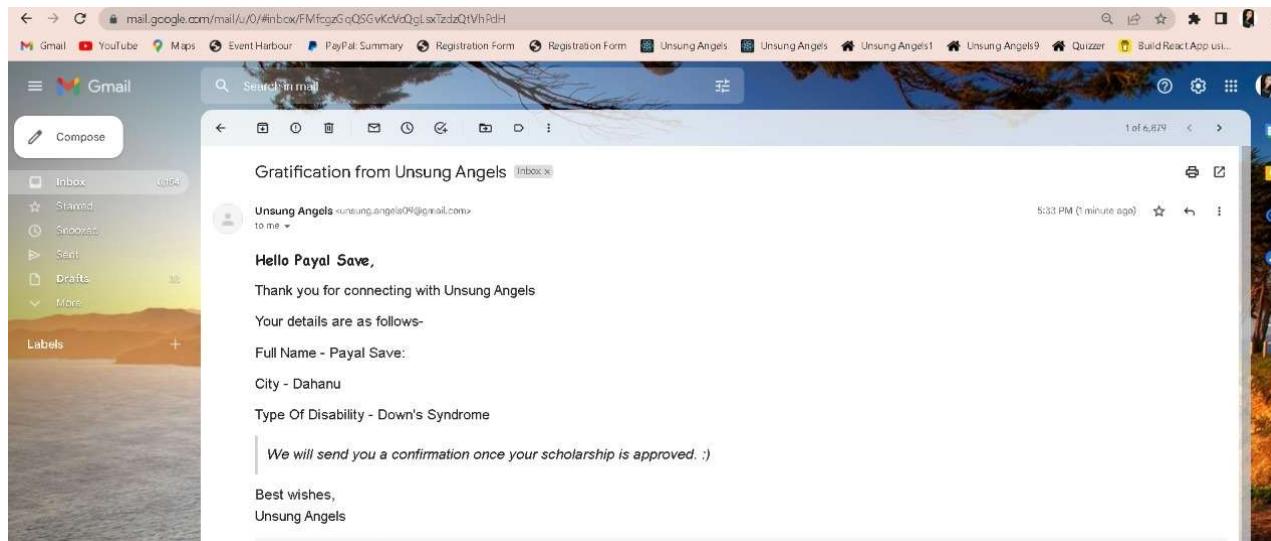
4.8.1: Personal Details Form to Avail Scholarship

4.9 Acknowledgement Page



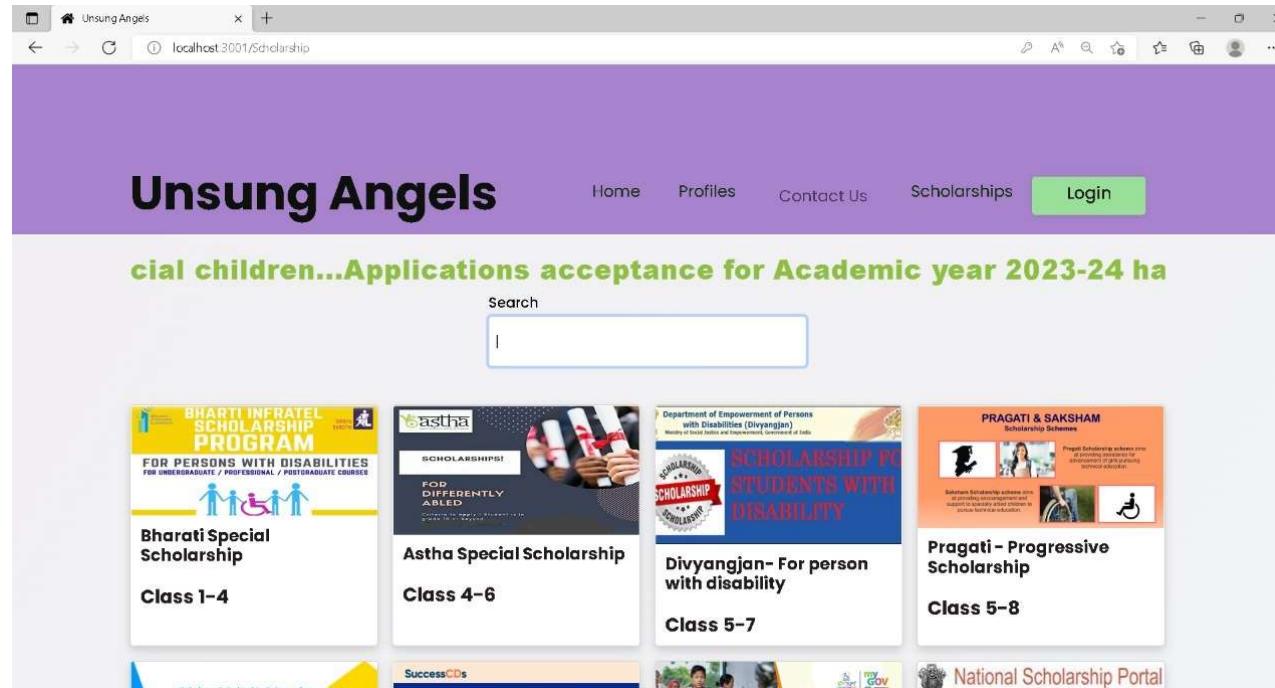
4.9.1: Acknowledgement Page

4.10 Email gratification



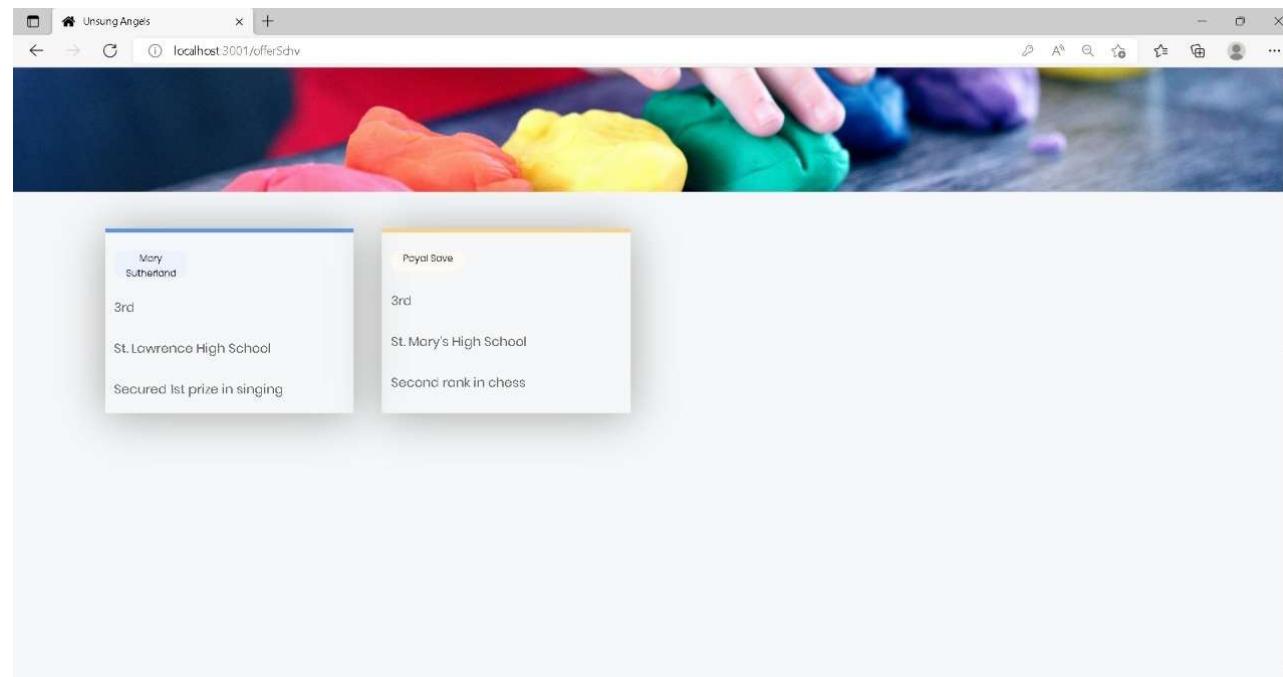
4.10.1: Email gratification Page

4.11 Search For Scholarships



4.11.1: Search For Various Scholarships Using Search Bar

4.12 View Profiles



4.12.1: View Profile Page

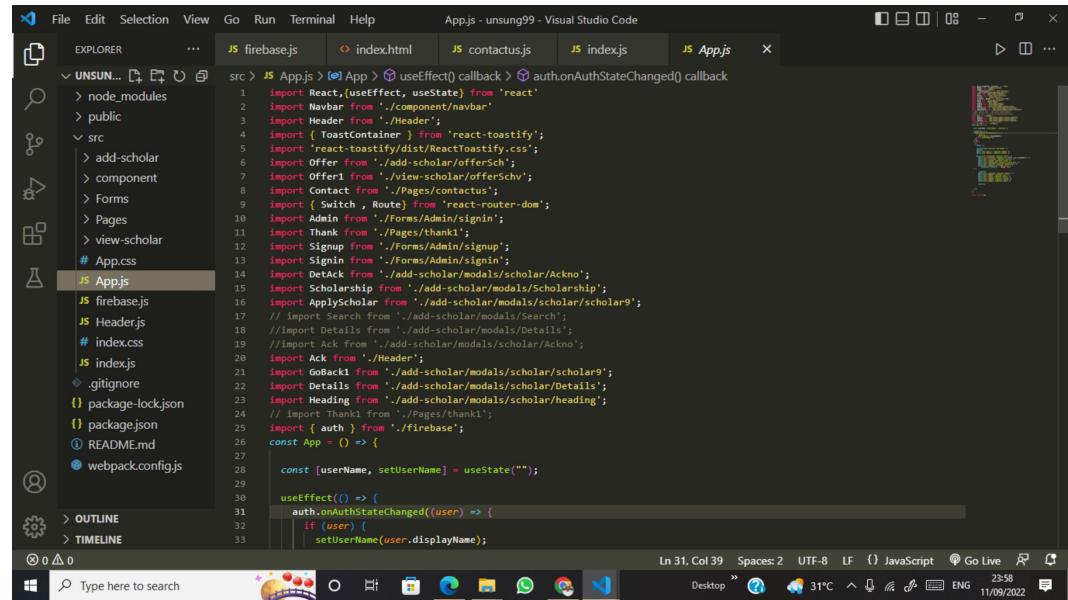
4.13 Feedback form

The screenshot shows a web browser window with the title bar "Unsung Angels" and the URL "localhost:3001/contactus". The main content area displays a form titled "SEND US A MESSAGE". The form has three input fields: "Name" containing "Payal Save", "Email" containing "payal@save@gmail.com", and "Message" containing "good service". Below the message field is a large blue "Submit" button.

4.13.1: Feedback Page to get in touch with us

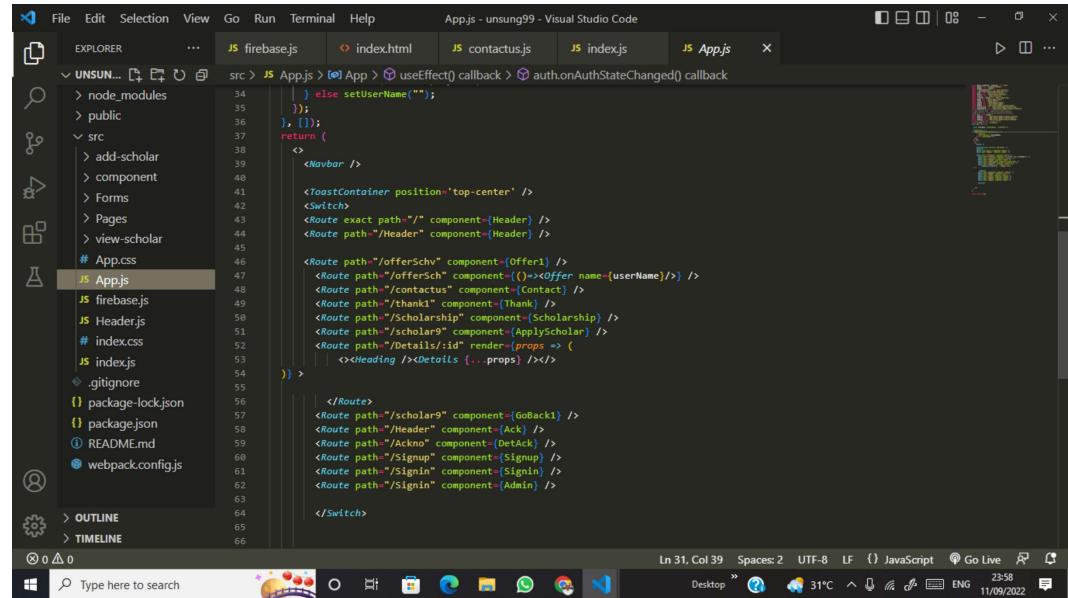
Payal Save (2021510060)
Unsung Angels - A Web App Vyankatesh Wable (2021510067)

4.14 App.js



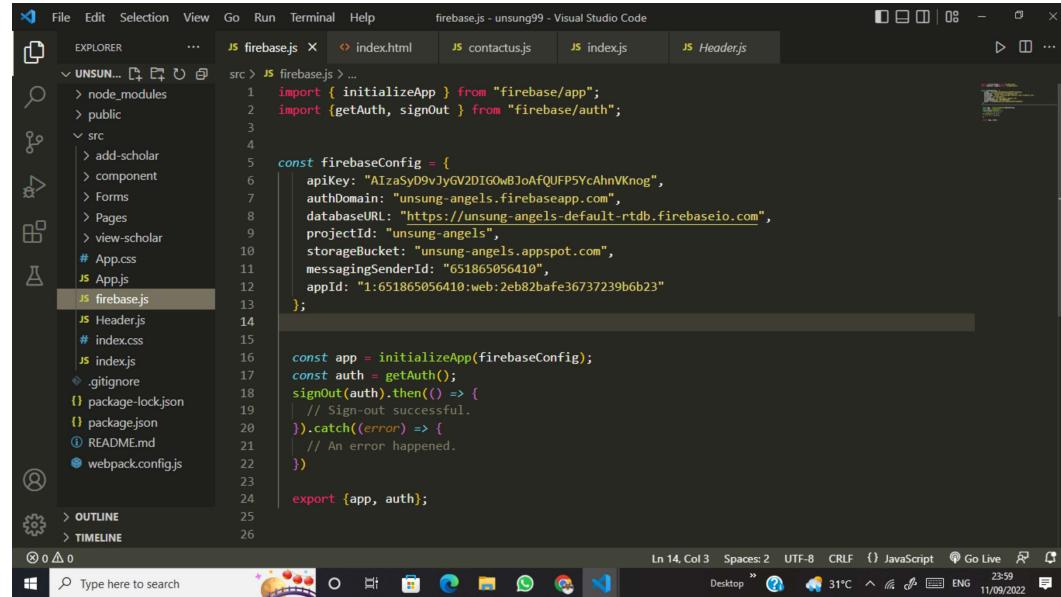
```
File Edit Selection View Go Run Terminal Help App.js - unsung99 - Visual Studio Code
src > JS App.js > (o) App > useEffect() callback > auth.onAuthStateChanged() callback
1 import React,{useEffect, useState} from 'react'
2 import Navbar from './component/navbar'
3 import Header from './Header';
4 import { ToastContainer } from 'react-toastify';
5 import 'react-toastify/dist/ReactToastify.css';
6 import Offer from './add-scholar/offerSch';
7 import Offer1 from './view-scholar/offerSchv';
8 import Contact from './Pages/contactus';
9 import { Switch , Route } from 'react-router-dom';
10 import Admin from './Forms/admin/signin';
11 import Thank from './Pages/thank1';
12 import Signup from './Forms/Admin/signup';
13 import Signin from './Forms/Admin/signin';
14 import Details from './add-scholar/modals/scholar/Ackno';
15 import Scholarship from './add-scholar/modals/Scholarship';
16 import Applyscholar from './add-scholar/modals/scholar/scholar9';
17 // import Search from './add-scholar/modals/Search';
18 //import Details from './add-scholar/modals/Details';
19 //import Ack from './add-scholar/modals/scholar/Ackno';
20 import Ack from './Header';
21 import GoBack1 from './add-scholar/modals/scholar/scholar9';
22 import Details from './add-scholar/modals/scholar/Details';
23 import Heading from './add-scholar/modals/scholar/heading';
24 // import Thank1 from './Pages/thank1';
25 import { auth } from './firebase';
26 const App = () => {
27
28   const [userName, setUserName] = useState("");
29
30   useEffect(() => {
31     auth.onAuthStateChanged((user) => {
32       if (user) {
33         setUserName(user.displayName);
34       } else setUserName("");
35     });
36   }, []);
37
38   return (
39     <Navbar />
40     <ToastContainer position="top-center" />
41     <Switch>
42       <Route exact path="/" component={Header} />
43       <Route path="/Header" component={Header} />
44
45       <Route path="/offersch" component={Offer} />
46       <Route path="/offersch" component={Offer1} />
47       <Route path="/contactus" component={Contact} />
48       <Route path="/thank1" component={Thank} />
49       <Route path="/scholarship" component={Scholarship} />
50       <Route path="/scholar9" component={ApplyScholar} />
51       <Route path="/Details/:id" render={props => (
52         <Heading /><Details {...props} />/)
53       )}>
54
55       <Route>
56         <Route path="/scholar9" component={GoBack1} />
57         <Route path="/Header" component={Ack} />
58         <Route path="/Ackno" component={Details} />
59         <Route path="/Signup" component={Signup} />
60         <Route path="/Signin" component={Signin} />
61         <Route path="/Signin" component={Admin} />
62
63       </Route>
64     </Switch>
65   );
66 }
```

4.15 App.js



```
File Edit Selection View Go Run Terminal Help App.js - unsung99 - Visual Studio Code
src > JS App.js > (o) App > useEffect() callback > auth.onAuthStateChanged() callback
1 import React,{useEffect, useState} from 'react'
2 import Navbar from './component/navbar'
3 import Header from './Header';
4 import { ToastContainer } from 'react-toastify';
5 import 'react-toastify/dist/ReactToastify.css';
6 import Offer from './add-scholar/offerSch';
7 import Offer1 from './view-scholar/offerSchv';
8 import Contact from './Pages/contactus';
9 import { Switch , Route } from 'react-router-dom';
10 import Admin from './Forms/admin/signin';
11 import Thank from './Pages/thank1';
12 import Signup from './Forms/Admin/signup';
13 import Signin from './Forms/Admin/signin';
14 import Details from './add-scholar/modals/scholar/Ackno';
15 import Scholarship from './add-scholar/modals/Scholarship';
16 import Applyscholar from './add-scholar/modals/scholar/scholar9';
17 // import Search from './add-scholar/modals/Search';
18 //import Details from './add-scholar/modals/Details';
19 //import Ack from './add-scholar/modals/scholar/Ackno';
20 import Ack from './Header';
21 import GoBack1 from './add-scholar/modals/scholar/scholar9';
22 import Details from './add-scholar/modals/scholar/Details';
23 import Heading from './add-scholar/modals/scholar/heading';
24 // import Thank1 from './Pages/thank1';
25 import { auth } from './firebase';
26 const App = () => {
27
28   const [userName, setUserName] = useState("");
29
30   useEffect(() => {
31     auth.onAuthStateChanged((user) => {
32       if (user) {
33         setUserName(user.displayName);
34       } else setUserName("");
35     });
36   }, []);
37
38   return (
39     <Navbar />
40     <ToastContainer position="top-center" />
41     <Switch>
42       <Route exact path="/" component={Header} />
43       <Route path="/Header" component={Header} />
44
45       <Route path="/offersch" component={Offer} />
46       <Route path="/offersch" component={Offer1} />
47       <Route path="/contactus" component={Contact} />
48       <Route path="/thank1" component={Thank} />
49       <Route path="/scholarship" component={Scholarship} />
50       <Route path="/scholar9" component={ApplyScholar} />
51       <Route path="/Details/:id" render={props => (
52         <Heading /><Details {...props} />/)
53       )}>
54
55       <Route>
56         <Route path="/scholar9" component={GoBack1} />
57         <Route path="/Header" component={Ack} />
58         <Route path="/Ackno" component={Details} />
59         <Route path="/Signup" component={Signup} />
60         <Route path="/Signin" component={Signin} />
61         <Route path="/Signin" component={Admin} />
62
63       </Route>
64     </Switch>
65   );
66 }
```

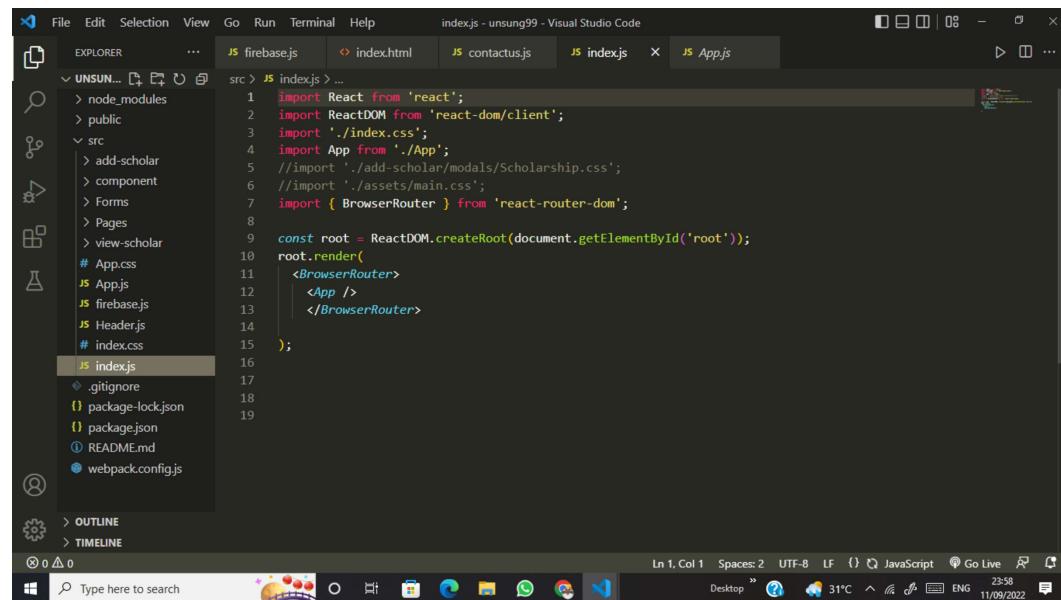
4.16 Firebase



The screenshot shows the Visual Studio Code interface with the file 'firebase.js' open. The code initializes a Firebase app with specific configuration details, including the API key, auth domain, database URL, project ID, storage bucket, messagingSenderId, and appId.

```
src > JS firebase.js ...
1 import { initializeApp } from "firebase/app";
2 import { getAuth, signOut } from "firebase/auth";
3
4 const firebaseConfig = {
5   apiKey: "AIzaSyD9vJyGV2DIG0wBJoAfQUFP5YcAhnVKnog",
6   authDomain: "unsung-angels.firebaseioapp.com",
7   databaseURL: "https://unsung-angels-default.firebaseio.com",
8   projectId: "unsung-angels",
9   storageBucket: "unsung-angels.appspot.com",
10  messagingSenderId: "651865056410",
11  appId: "1:651865056410:web:2eb82bafe36737239b6b23"
12};
13
14
15 const app = initializeApp(firebaseConfig);
16 const auth = getAuth();
17 signOut(auth).then(() => {
18   // Sign-out successful.
19 }).catch((error) => {
20   // An error happened.
21 });
22
23
24 export { app, auth };
25
26
```

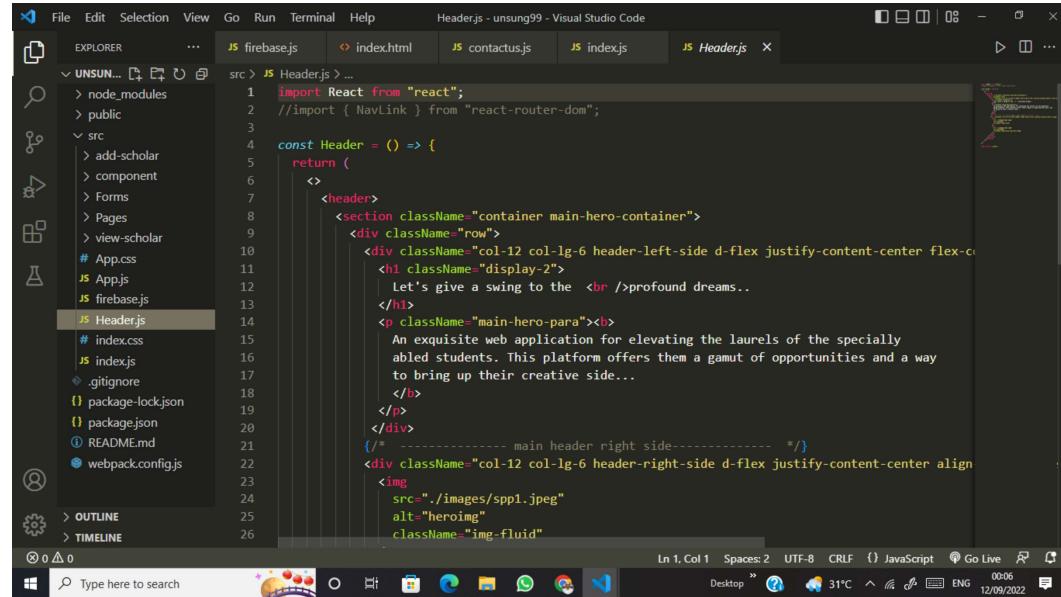
4.17 Index.js



The screenshot shows the Visual Studio Code interface with the file 'index.js' open. The code imports React, ReactDOM, and other components, then creates a root element and renders the App component within a BrowserRouter.

```
src > JS index.js ...
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5 //import './add-scholar/modals/Scholarship.css';
6 //import './assets/main.css';
7 import { BrowserRouter } from 'react-router-dom';
8
9 const root = ReactDOM.createRoot(document.getElementById('root'));
10 root.render(
11   <BrowserRouter>
12     <App />
13   </BrowserRouter>
14 );
15
```

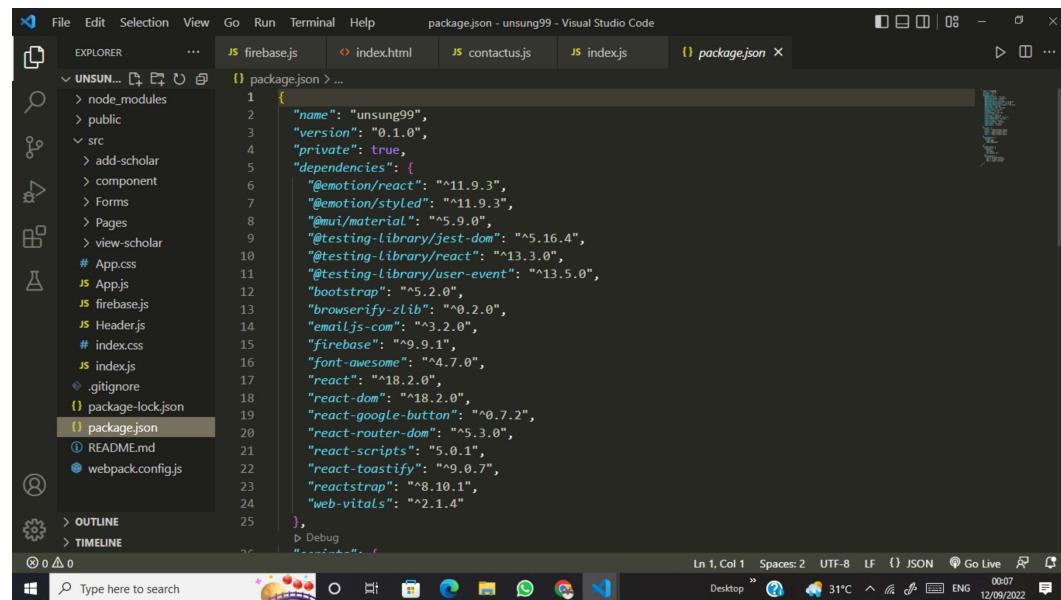
4.18 Header



The screenshot shows the Visual Studio Code interface with the Header.js file open. The code defines a functional component named Header that returns a JSX structure. It includes a main hero container with a heading and a paragraph, and a right side section featuring an image of a superhero.

```
import React from "react";
//import { NavLink } from "react-router-dom";
const Header = () => {
  return (
    <>
      <header>
        <section className="container main-hero-container">
          <div className="row">
            <div className="col-12 col-lg-6 header-left-side d-flex justify-content-center flex-grow-1">
              <h1>display-2</h1>
              Let's give a swing to the <br />profound dreams..
            </div>
            <div className="col-12 col-lg-6 header-right-side d-flex justify-content-center align-items-center">
              An exquisite web application for elevating the laurels of the specially abled students. This platform offers them a gamut of opportunities and a way to bring up their creative side...
            </div>
          </div>
        </section>
        <div className="img-fluid" alt="heroimg" style={{ height: '100%' }}>
          
        </div>
      </header>
    </>
  );
}
```

4.19 Packages Installed



The screenshot shows the Visual Studio Code interface with the package.json file open. The JSON object contains details about the project, including its name, version, private status, and dependencies.

```
{
  "name": "unsung99",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@emotion/react": "^11.9.3",
    "@emotion/styled": "^11.9.3",
    "@mui/material": "5.9.0",
    "@testing-library/jest-dom": "^5.16.4",
    "@testing-library/react": "^13.3.0",
    "@testing-library/user-event": "13.5.0",
    "bootstrap": "5.2.0",
    "browserify-zlib": "0.2.0",
    "emailjs-com": "3.2.0",
    "firebase": "9.9.1",
    "font-awesome": "4.7.0",
    "react": "18.2.0",
    "react-dom": "18.2.0",
    "react-google-button": "0.7.2",
    "react-router-dom": "5.3.0",
    "react-scripts": "5.0.1",
    "react-toastify": "9.0.7",
    "reactstrap": "8.10.1",
    "web-vitals": "2.1.4"
  }
}
```

4.20 Navigation Bar

```
src > component > JS navbar.js > ...
1 // Import React from 'react';
2 import React, { useState } from "react";
3 // Import Navbar from 'react-router-dom';
4 import './navbar.css';
5 // Import React from "react";
6
7 const Navbar = () => {
8   const [show, setShow] = useState(false);
9
10   return (
11     <>
12       <section className="navbar-bg">
13         <nav className="navbar navbar-expand-lg navbar-light">
14           <div className="container">
15             <a href="#" className="navbar-brand" style={{fontSize: "4em",fontWeight:"bold"}}>Unsung Angels</a>
16           </div>
17           <button
18             className="navbar-toggler"
19             type="button"
20             data-bs-toggle="collapse"
21             data-bs-target="#navbarSupportedContent"
22             aria-controls="navbarSupportedContent"
23             aria-expanded="false"
24             aria-label="Toggle navigation"
25             onClick={() => setShow(show)}
26           <span className="navbar-toggler-icon"></span>
27         </button>
28         <div className="collapse navbar-collapse" show={show}>
29           <ul className="navbar-nav ms-auto mb-2 mb-lg-0">
30             <li className="nav-item">
31               <a href="#" className="nav-link active" aria-current="page" to="/Header">
32                 <Home/>
33               </a>
34             </li>
35             <li className="nav-item">
36               <a href="#" className="nav-link" to="/offerSchw">
37                 <OfferSchw/>
38               </a>
39             </li>
40           </ul>
41         </div>
42       </nav>
43     </section>
44   );
45 }
46
47 export default Navbar;
```

4.21 SignIn Page



```
src > Forms > Admin > JS signIn.js > ...
1 import React, { useState } from 'react';
2 import { NavLink, useHistory } from 'react-router-dom';
3 import { signInWithEmailAndPassword } from 'firebase/auth';
4 import { auth } from './../../../firebase';
5 //import './formInput.css';
6 import GoogleButton from 'react-google-button';
7
8 const SignIn = () => {
9
10   let history = useHistory();
11   const [values, setValues] = useState({
12     email: '',
13     pass: ''
14   });
15   const [errorMsg, setErrorMsg] = useState("");
16   const [submitButtonDisabled, setSubmitButtonDisabled] = useState(false);
17
18   const handleSubmission = () => {
19     if (!values.email || !values.pass) {
20       setErrorMsg("fill all fields");
21       return;
22     }
23     setErrorMsg("");
24
25     setSubmitButtonDisabled(true);
26     signInWithEmailAndPassword(auth, values.email, values.pass)
27       .then(async (res) => {
28         setSubmitButtonDisabled(false);
29         history.push("/offerSch");
30       })
31       .catch((err) => {
32         setSubmitButtonDisabled(false);
33         setErrorMsg(err.message);
34       });
35   };
36
37   return (
38     <div className="limiter">
39       <div style={{ display: "flex", alignContent: "center" }>
40         <div style={{ width: "40%" }>
41           <h3>ĐĂNG NHẬP</h3>
42           <FormInput type="text" value={values.email} onChange={(e) => setValues({ ...values, email: e.target.value })} placeholder="Email" />
43           <FormInput type="password" value={values.pass} onChange={(e) => setValues({ ...values, pass: e.target.value })} placeholder="Mật khẩu" />
44           <GoogleButton onClick={handleSubmission} />
45         </div>
46         <div style={{ width: "50%", text-align: "center" }>
47           
48           <p>ĐĂNG NHẬP</p>
49           <p>Nhập tên đăng nhập và mật khẩu để đăng nhập vào hệ thống. Nếu bạn chưa có tài khoản, vui lòng đăng ký.</p>
50           <a href="#">Quên mật khẩu?</a>
51         </div>
52       </div>
53     </div>
54   );
55 }
```

4.22 Sign Up Page

```
src > Forms > Admin > JS signup.js > ...
14  const Signup = () => {
15    let history = useHistory();
16
17    const [values, setValues] = useState({
18      name: '',
19      email: '',
20      pass: ''
21    });
22
23    const [errorMsg, setErrorMsg] = useState("");
24    const [submitButtonDisabled, setSubmitButtonDisabled] = useState(false);
25
26    const handleSubmission = () => {
27
28      if (!values.name || !values.email || !values.pass) {
29        setErrorMsg("Please fill out all the fields");
30        return;
31      }
32
33      setErrorMsg("");
34      setSubmitButtonDisabled(true);
35      createWithEmailAndPassword(auth, values.email, values.pass).then(async (res) => {
36        const user = res.user;
37
38        await updateProfile(user, {
39          displayName: values.name,
40        });
41        history.push("/signin");
42
43      })
44
45      .catch((err) => {
46        setSubmitButtonDisabled(false);
47        setErrorMsg(err.message);
48      });
49
50    };
51  };

const Signup = () => {
  let history = useHistory();

  const [values, setValues] = useState({
    name: '',
    email: '',
    pass: ''
  });

  const [errorMsg, setErrorMsg] = useState("");
  const [submitButtonDisabled, setSubmitButtonDisabled] = useState(false);

  const handleSubmission = () => {

    if (!values.name || !values.email || !values.pass) {
      setErrorMsg("Please fill out all the fields");
      return;
    }

    setErrorMsg("");
    setSubmitButtonDisabled(true);
    createWithEmailAndPassword(auth, values.email, values.pass).then(async (res) => {
      const user = res.user;

      await updateProfile(user, {
        displayName: values.name,
      });
      history.push("/signin");

    })

    .catch((err) => {
      setSubmitButtonDisabled(false);
      setErrorMsg(err.message);
    });
  };
};


```

4.23 Contact Us Page

```
File Edit Selection View Go Run Terminal Help contactus.js - unsung99 - Visual Studio Code

EXPLORER ... JS firebase.js JS signin.js JS signup.js index.html JS contactus.js X JS index.js

src > Pages > JS contactus.js > [o] Contactus > [o] postData > [o] res
1 import React, {useState} from 'react';
2
3
4 const Contactus = () => {
5
6   const [user, setUser]=useState({
7     name:"",
8     email:"",
9     message:"",
10    });
11
12   let name, value;
13   const getUserData =(event)=>{
14     name = event.target.name;
15     value = event.target.value;
16
17     setUser({ ...user, [name]: value });
18   };
19
20   const postData = async (e) => {
21     e.preventDefault();
22
23     const {name, email, message} = user;
24
25     if(name && email && message){
26
27       const res = await fetch(
28         "https://unsung-angels-default.firebaseio.com/contactusform.json",
29         {
30           method : "POST",
31           headers: {
32             "content-type": "application/json",
33           },
34           body: JSON.stringify({
35             name,
36             email,
37             message,
38           })
39         }
40       );
41     }
42   };
43
44   return (
45     <div>
46       <h1>Contact Us</h1>
47       <form>
48         <input type="text" name="name" placeholder="Name" />
49         <input type="text" name="email" placeholder="Email" />
50         <input type="text" name="message" placeholder="Message" />
51         <button type="button" onClick={postData}>Submit</button>
52       </form>
53     </div>
54   );
55 }

const Contactus = () => {
56
57   const [user, setUser]=useState({
58     name:"",
59     email:"",
60     message:"",
61    });
62
63   let name, value;
64   const getUserData =(event)=>{
65     name = event.target.name;
66     value = event.target.value;
67
68     setUser({ ...user, [name]: value });
69   };
70
71   const postData = async (e) => {
72     e.preventDefault();
73
74     const {name, email, message} = user;
75
76     if(name && email && message){
77
78       const res = await fetch(
79         "https://unsung-angels-default.firebaseio.com/contactusform.json",
80         {
81           method : "POST",
82           headers: {
83             "content-type": "application/json",
84           },
85           body: JSON.stringify({
86             name,
87             email,
88             message,
89           })
90         }
91       );
92     }
93   };
94
95   return (
96     <div>
97       <h1>Contact Us</h1>
98       <form>
99         <input type="text" name="name" placeholder="Name" />
100        <input type="text" name="email" placeholder="Email" />
101        <input type="text" name="message" placeholder="Message" />
102        <button type="button" onClick={postData}>Submit</button>
103      </form>
104    </div>
105  );
106}

const Contactus = () => {
107
108   const [user, setUser]=useState({
109     name:"",
110     email:"",
111     message:"",
112    });
113
114   let name, value;
115   const getUserData =(event)=>{
116     name = event.target.name;
117     value = event.target.value;
118
119     setUser({ ...user, [name]: value });
120   };
121
122   const postData = async (e) => {
123     e.preventDefault();
124
125     const {name, email, message} = user;
126
127     if(name && email && message){
128
129       const res = await fetch(
130         "https://unsung-angels-default.firebaseio.com/contactusform.json",
131         {
132           method : "POST",
133           headers: {
134             "content-type": "application/json",
135           },
136           body: JSON.stringify({
137             name,
138             email,
139             message,
140           })
141         }
142       );
143     }
144   };
145
146   return (
147     <div>
148       <h1>Contact Us</h1>
149       <form>
150         <input type="text" name="name" placeholder="Name" />
151         <input type="text" name="email" placeholder="Email" />
152         <input type="text" name="message" placeholder="Message" />
153         <button type="button" onClick={postData}>Submit</button>
154       </form>
155     </div>
156   );
157}

const Contactus = () => {
158
159   const [user, setUser]=useState({
160     name:"",
161     email:"",
162     message:"",
163    });
164
165   let name, value;
166   const getUserData =(event)=>{
167     name = event.target.name;
168     value = event.target.value;
169
170     setUser({ ...user, [name]: value });
171   };
172
173   const postData = async (e) => {
174     e.preventDefault();
175
176     const {name, email, message} = user;
177
178     if(name && email && message){
179
180       const res = await fetch(
181         "https://unsung-angels-default.firebaseio.com/contactusform.json",
182         {
183           method : "POST",
184           headers: {
185             "content-type": "application/json",
186           },
187           body: JSON.stringify({
188             name,
189             email,
190             message,
191           })
192         }
193       );
194     }
195   };
196
197   return (
198     <div>
199       <h1>Contact Us</h1>
200       <form>
201         <input type="text" name="name" placeholder="Name" />
202         <input type="text" name="email" placeholder="Email" />
203         <input type="text" name="message" placeholder="Message" />
204         <button type="button" onClick={postData}>Submit</button>
205       </form>
206     </div>
207   );
208}

const Contactus = () => {
209
210   const [user, setUser]=useState({
211     name:"",
212     email:"",
213     message:"",
214    });
215
216   let name, value;
217   const getUserData =(event)=>{
218     name = event.target.name;
219     value = event.target.value;
220
221     setUser({ ...user, [name]: value });
222   };
223
224   const postData = async (e) => {
225     e.preventDefault();
226
227     const {name, email, message} = user;
228
229     if(name && email && message){
230
231       const res = await fetch(
232         "https://unsung-angels-default.firebaseio.com/contactusform.json",
233         {
234           method : "POST",
235           headers: {
236             "content-type": "application/json",
237           },
238           body: JSON.stringify({
239             name,
240             email,
241             message,
242           })
243         }
244       );
245     }
246   };
247
248   return (
249     <div>
250       <h1>Contact Us</h1>
251       <form>
252         <input type="text" name="name" placeholder="Name" />
253         <input type="text" name="email" placeholder="Email" />
254         <input type="text" name="message" placeholder="Message" />
255         <button type="button" onClick={postData}>Submit</button>
256       </form>
257     </div>
258   );
259}

const Contactus = () => {
260
261   const [user, setUser]=useState({
262     name:"",
263     email:"",
264     message:"",
265    });
266
267   let name, value;
268   const getUserData =(event)=>{
269     name = event.target.name;
270     value = event.target.value;
271
272     setUser({ ...user, [name]: value });
273   };
274
275   const postData = async (e) => {
276     e.preventDefault();
277
278     const {name, email, message} = user;
279
280     if(name && email && message){
281
282       const res = await fetch(
283         "https://unsung-angels-default.firebaseio.com/contactusform.json",
284         {
285           method : "POST",
286           headers: {
287             "content-type": "application/json",
288           },
289           body: JSON.stringify({
290             name,
291             email,
292             message,
293           })
294         }
295       );
296     }
297   };
298
299   return (
300     <div>
301       <h1>Contact Us</h1>
302       <form>
303         <input type="text" name="name" placeholder="Name" />
304         <input type="text" name="email" placeholder="Email" />
305         <input type="text" name="message" placeholder="Message" />
306         <button type="button" onClick={postData}>Submit</button>
307       </form>
308     </div>
309   );
310}

const Contactus = () => {
311
312   const [user, setUser]=useState({
313     name:"",
314     email:"",
315     message:"",
316    });
317
318   let name, value;
319   const getUserData =(event)=>{
320     name = event.target.name;
321     value = event.target.value;
322
323     setUser({ ...user, [name]: value });
324   };
325
326   const postData = async (e) => {
327     e.preventDefault();
328
329     const {name, email, message} = user;
330
331     if(name && email && message){
332
333       const res = await fetch(
334         "https://unsung-angels-default.firebaseio.com/contactusform.json",
335         {
336           method : "POST",
337           headers: {
338             "content-type": "application/json",
339           },
340           body: JSON.stringify({
341             name,
342             email,
343             message,
344           })
345         }
346       );
347     }
348   };
349
350   return (
351     <div>
352       <h1>Contact Us</h1>
353       <form>
354         <input type="text" name="name" placeholder="Name" />
355         <input type="text" name="email" placeholder="Email" />
356         <input type="text" name="message" placeholder="Message" />
357         <button type="button" onClick={postData}>Submit</button>
358       </form>
359     </div>
360   );
361}

const Contactus = () => {
362
363   const [user, setUser]=useState({
364     name:"",
365     email:"",
366     message:"",
367    });
368
369   let name, value;
370   const getUserData =(event)=>{
371     name = event.target.name;
372     value = event.target.value;
373
374     setUser({ ...user, [name]: value });
375   };
376
377   const postData = async (e) => {
378     e.preventDefault();
379
380     const {name, email, message} = user;
381
382     if(name && email && message){
383
384       const res = await fetch(
385         "https://unsung-angels-default.firebaseio.com/contactusform.json",
386         {
387           method : "POST",
388           headers: {
389             "content-type": "application/json",
390           },
391           body: JSON.stringify({
392             name,
393             email,
394             message,
395           })
396         }
397       );
398     }
399   };
400
401   return (
402     <div>
403       <h1>Contact Us</h1>
404       <form>
405         <input type="text" name="name" placeholder="Name" />
406         <input type="text" name="email" placeholder="Email" />
407         <input type="text" name="message" placeholder="Message" />
408         <button type="button" onClick={postData}>Submit</button>
409       </form>
410     </div>
411   );
412}

const Contactus = () => {
413
414   const [user, setUser]=useState({
415     name:"",
416     email:"",
417     message:"",
418    });
419
420   let name, value;
421   const getUserData =(event)=>{
422     name = event.target.name;
423     value = event.target.value;
424
425     setUser({ ...user, [name]: value });
426   };
427
428   const postData = async (e) => {
429     e.preventDefault();
430
431     const {name, email, message} = user;
432
433     if(name && email && message){
434
435       const res = await fetch(
436         "https://unsung-angels-default.firebaseio.com/contactusform.json",
437         {
438           method : "POST",
439           headers: {
440             "content-type": "application/json",
441           },
442           body: JSON.stringify({
443             name,
444             email,
445             message,
446           })
447         }
448       );
449     }
450   };
451
452   return (
453     <div>
454       <h1>Contact Us</h1>
455       <form>
456         <input type="text" name="name" placeholder="Name" />
457         <input type="text" name="email" placeholder="Email" />
458         <input type="text" name="message" placeholder="Message" />
459         <button type="button" onClick={postData}>Submit</button>
460       </form>
461     </div>
462   );
463}

const Contactus = () => {
464
465   const [user, setUser]=useState({
466     name:"",
467     email:"",
468     message:"",
469    });
470
471   let name, value;
472   const getUserData =(event)=>{
473     name = event.target.name;
474     value = event.target.value;
475
476     setUser({ ...user, [name]: value });
477   };
478
479   const postData = async (e) => {
480     e.preventDefault();
481
482     const {name, email, message} = user;
483
484     if(name && email && message){
485
486       const res = await fetch(
487         "https://unsung-angels-default.firebaseio.com/contactusform.json",
488         {
489           method : "POST",
490           headers: {
491             "content-type": "application/json",
492           },
493           body: JSON.stringify({
494             name,
495             email,
496             message,
497           })
498         }
499       );
500     }
501   };
502
503   return (
504     <div>
505       <h1>Contact Us</h1>
506       <form>
507         <input type="text" name="name" placeholder="Name" />
508         <input type="text" name="email" placeholder="Email" />
509         <input type="text" name="message" placeholder="Message" />
510         <button type="button" onClick={postData}>Submit</button>
511       </form>
512     </div>
513   );
514}

const Contactus = () => {
515
516   const [user, setUser]=useState({
517     name:"",
518     email:"",
519     message:"",
520    });
521
522   let name, value;
523   const getUserData =(event)=>{
524     name = event.target.name;
525     value = event.target.value;
526
527     setUser({ ...user, [name]: value });
528   };
529
530   const postData = async (e) => {
531     e.preventDefault();
532
533     const {name, email, message} = user;
534
535     if(name && email && message){
536
537       const res = await fetch(
538         "https://unsung-angels-default.firebaseio.com/contactusform.json",
539         {
540           method : "POST",
541           headers: {
542             "content-type": "application/json",
543           },
544           body: JSON.stringify({
545             name,
546             email,
547             message,
548           })
549         }
550       );
551     }
552   };
553
554   return (
555     <div>
556       <h1>Contact Us</h1>
557       <form>
558         <input type="text" name="name" placeholder="Name" />
559         <input type="text" name="email" placeholder="Email" />
560         <input type="text" name="message" placeholder="Message" />
561         <button type="button" onClick={postData}>Submit</button>
562       </form>
563     </div>
564   );
565}

const Contactus = () => {
566
567   const [user, setUser]=useState({
568     name:"",
569     email:"",
570     message:"",
571    });
572
573   let name, value;
574   const getUserData =(event)=>{
575     name = event.target.name;
576     value = event.target.value;
577
578     setUser({ ...user, [name]: value });
579   };
580
581   const postData = async (e) => {
582     e.preventDefault();
583
584     const {name, email, message} = user;
585
586     if(name && email && message){
587
588       const res = await fetch(
589         "https://unsung-angels-default.firebaseio.com/contactusform.json",
590         {
591           method : "POST",
592           headers: {
593             "content-type": "application/json",
594           },
595           body: JSON.stringify({
596             name,
597             email,
598             message,
599           })
600         }
601       );
602     }
603   };
604
605   return (
606     <div>
607       <h1>Contact Us</h1>
608       <form>
609         <input type="text" name="name" placeholder="Name" />
610         <input type="text" name="email" placeholder="Email" />
611         <input type="text" name="message" placeholder="Message" />
612         <button type="button" onClick={postData}>Submit</button>
613       </form>
614     </div>
615   );
616}

const Contactus = () => {
617
618   const [user, setUser]=useState({
619     name:"",
620     email:"",
621     message:"",
622    });
623
624   let name, value;
625   const getUserData =(event)=>{
626     name = event.target.name;
627     value = event.target.value;
628
629     setUser({ ...user, [name]: value });
630   };
631
632   const postData = async (e) => {
633     e.preventDefault();
634
635     const {name, email, message} = user;
636
637     if(name && email && message){
638
639       const res = await fetch(
640         "https://unsung-angels-default.firebaseio.com/contactusform.json",
641         {
642           method : "POST",
643           headers: {
644             "content-type": "application/json",
645           },
646           body: JSON.stringify({
647             name,
648             email,
649             message,
650           })
651         }
652       );
653     }
654   };
655
656   return (
657     <div>
658       <h1>Contact Us</h1>
659       <form>
660         <input type="text" name="name" placeholder="Name" />
661         <input type="text" name="email" placeholder="Email" />
662         <input type="text" name="message" placeholder="Message" />
663         <button type="button" onClick={postData}>Submit</button>
664       </form>
665     </div>
666   );
667}

const Contactus = () => {
668
669   const [user, setUser]=useState({
670     name:"",
671     email:"",
672     message:"",
673    });
674
675   let name, value;
676   const getUserData =(event)=>{
677     name = event.target.name;
678     value = event.target.value;
679
680     setUser({ ...user, [name]: value });
681   };
682
683   const postData = async (e) => {
684     e.preventDefault();
685
686     const {name, email, message} = user;
687
688     if(name && email && message){
689
690       const res = await fetch(
691         "https://unsung-angels-default.firebaseio.com/contactusform.json",
692         {
693           method : "POST",
694           headers: {
695             "content-type": "application/json",
696           },
697           body: JSON.stringify({
698             name,
699             email,
700             message,
701           })
702         }
703       );
704     }
705   };
706
707   return (
708     <div>
709       <h1>Contact Us</h1>
710       <form>
711         <input type="text" name="name" placeholder="Name" />
712         <input type="text" name="email" placeholder="Email" />
713         <input type="text" name="message" placeholder="Message" />
714         <button type="button" onClick={postData}>Submit</button>
715       </form>
716     </div>
717   );
718}

const Contactus = () => {
719
720   const [user, setUser]=useState({
721     name:"",
722     email:"",
723     message:"",
724    });
725
726   let name, value;
727   const getUserData =(event)=>{
728     name = event.target.name;
729     value = event.target.value;
730
731     setUser({ ...user, [name]: value });
732   };
733
734   const postData = async (e) => {
735     e.preventDefault();
736
737     const {name, email, message} = user;
738
739     if(name && email && message){
740
741       const res = await fetch(
742         "https://unsung-angels-default.firebaseio.com/contactusform.json",
743         {
744           method : "POST",
745           headers: {
746             "content-type": "application/json",
747           },
748           body: JSON.stringify({
749             name,
750             email,
751             message,
752           })
753         }
754       );
755     }
756   };
757
758   return (
759     <div>
760       <h1>Contact Us</h1>
761       <form>
762         <input type="text" name="name" placeholder="Name" />
763         <input type="text" name="email" placeholder="Email" />
764         <input type="text" name="message" placeholder="Message" />
765         <button type="button" onClick={postData}>Submit</button>
766       </form>
767     </div>
768   );
769}

const Contactus = () => {
770
771   const [user, setUser]=useState({
772     name:"",
773     email:"",
774     message:"",
775    });
776
777   let name, value;
778   const getUserData =(event)=>{
779     name = event.target.name;
780     value = event.target.value;
781
782     setUser({ ...user, [name]: value });
783   };
784
785   const postData = async (e) => {
786     e.preventDefault();
787
788     const {name, email, message} = user;
789
790     if(name && email && message){
791
792       const res = await fetch(
793         "https://unsung-angels-default.firebaseio.com/contactusform.json",
794         {
795           method : "POST",
796           headers: {
797             "content-type": "application/json",
798           },
799           body: JSON.stringify({
800             name,
801             email,
802             message,
803           })
804         }
805       );
806     }
807   };
808
809   return (
810     <div>
811       <h1>Contact Us</h1>
812       <form>
813         <input type="text" name="name" placeholder="Name" />
814         <input type="text" name="email" placeholder="Email" />
815         <input type="text" name="message" placeholder="Message" />
816         <button type="button" onClick={postData}>Submit</button>
817       </form>
818     </div>
819   );
820}

const Contactus = () => {
821
822   const [user, setUser]=useState({
823     name:"",
824     email:"",
825     message:"",
826    });
827
828   let name, value;
829   const getUserData =(event)=>{
830     name = event.target.name;
831     value = event.target.value;
832
833     setUser({ ...user, [name]: value });
834   };
835
836   const postData = async (e) => {
837     e.preventDefault();
838
839     const {name, email, message} = user;
840
841     if(name && email && message){
842
843       const res = await fetch(
844         "https://unsung-angels-default.firebaseio.com/contactusform.json",
845         {
846           method : "POST",
847           headers: {
848             "content-type": "application/json",
849           },
850           body: JSON.stringify({
851             name,
852             email,
853             message,
854           })
855         }
856       );
857     }
858   };
859
860   return (
861     <div>
862       <h1>Contact Us</h1>
863       <form>
864         <input type="text" name="name" placeholder="Name" />
865         <input type="text" name="email" placeholder="Email" />
866         <input type="text" name="message" placeholder="Message" />
867         <button type="button" onClick={postData}>Submit</button>
868       </form>
869     </div>
870   );
871}

const Contactus = () => {
872
873   const [user, setUser]=useState({
874     name:"",
875     email:"",
876     message:"",
877    });
878
879   let name, value;
880   const getUserData =(event)=>{
881     name = event.target.name;
882     value = event.target.value;
883
884     setUser({ ...user, [name]: value });
885   };
886
887   const postData = async (e) => {
888     e.preventDefault();
889
890     const {name, email, message} = user;
891
892     if(name && email && message){
893
894       const res = await fetch(
895         "https://unsung-angels-default.firebaseio.com/contactusform.json",
896         {
897           method : "POST",
898           headers: {
899             "content-type": "application/json",
900           },
901           body: JSON.stringify({
902             name,
903             email,
904             message,
905           })
906         }
907       );
908     }
909   };
910
911   return (
912     <div>
913       <h1>Contact Us</h1>
914       <form>
915         <input type="text" name="name" placeholder="Name" />
916         <input type="text" name="email" placeholder="Email" />
917         <input type="text" name="message" placeholder="Message" />
918         <button type="button" onClick={postData}>Submit</button>
919       </form>
920     </div>
921   );
922}

const Contactus = () => {
923
924   const [user, setUser]=useState({
925     name:"",
926     email:"",
927     message:"",
928    });
929
930   let name, value;
931   const getUserData =(event)=>{
932     name = event.target.name;
933     value = event.target.value;
934
935     setUser({ ...user, [name]: value });
936   };
937
938   const postData = async (e) => {
939     e.preventDefault();
940
941     const {name, email, message} = user;
942
943     if(name && email && message){
944
945       const res = await fetch(
946         "https://unsung-angels-default.firebaseio.com/contactusform.json",
947         {
948           method : "POST",
949           headers: {
950             "content-type": "application/json",
951           },
952           body: JSON.stringify({
953             name,
954             email,
955             message,
956           })
957         }
958       );
959     }
960   };
961
962   return (
963     <div>
964       <h1>Contact Us</h1>
965       <form>
966         <input type="text" name="name" placeholder="Name" />
967         <input type="text" name="email" placeholder="Email" />
968         <input type="text" name="message" placeholder="Message" />
969         <button type="button" onClick={postData}>Submit</button>
970       </form>
971     </div>
972   );
973}

const Contactus = () => {
974
975   const [user, setUser]=useState({
976     name:"",
977     email:"",
978     message:"",
979    });
980
981   let name, value;
982   const getUserData =(event)=>{
983     name = event.target.name;
984     value = event.target.value;
985
986     setUser({ ...user, [name]: value });
987   };
988
989   const postData = async (e) => {
990     e.preventDefault();
991
992     const {name, email, message} = user;
993
994     if(name && email && message){
995
996       const res = await fetch(
997         "https://unsung-angels-default.firebaseio.com/contactusform.json",
998         {
999           method : "POST",
1000           headers: {
1001             "content-type": "application/json",
1002           },
1003           body: JSON.stringify({
1004             name,
1005             email,
1006             message,
1007           })
1008         }
1009       );
1010     }
1011   };
1012
1013   return (
1014     <div>
1015       <h1>Contact Us</h1>
1016       <form>
1017         <input type="text" name="name" placeholder="Name" />
1018         <input type="text" name="email" placeholder="Email" />
1019         <input type="text" name="message" placeholder="Message" />
1020         <button type="button" onClick={postData}>Submit</button>
1021       </form>
1022     </div>
1023   );
1024}

const Contactus = () => {
1025
1026   const [user, setUser]=useState({
1027     name:"",
1028     email:"",
1029     message:"",
1030    });
1031
1032   let name, value;
1033   const getUserData =(event)=>{
1034     name = event.target.name;
1035     value = event.target.value;
1036
1037     setUser({ ...user, [name]: value });
1038   };
1039
1040   const postData = async (e) => {
1041     e.preventDefault();
1042
1043     const {name, email, message} = user;
1044
1045     if(name && email && message){
1046
1047       const res = await fetch(
1048         "https://unsung-angels-default.firebaseio.com/contactusform.json",
1049         {
1050           method : "POST",
1051           headers: {
1052             "content-type": "application/json",
1053           },
1054           body: JSON.stringify({
1055             name,
1056             email,
1057             message,
1058           })
1059         }
1060       );
1061     }
1062   };
1063
1064   return (
1065     <div>
1066       <h1>Contact Us</h1>
1067       <form>
1068         <input type="text" name="name" placeholder="Name" />
1069         <input type="text" name="email" placeholder="Email" />
1070         <input type="text" name="message" placeholder="Message" />
1071         <button type="button" onClick={postData}>Submit</button>
1072       </form>
1073     </div>
1074   );
1075}

const Contactus = () => {
1076
1077   const [user, setUser]=useState({
1078     name:"",
1079     email:"",
1080     message:"",
1081    });
1082
1083   let name, value;
1084   const getUserData =(event)=>{
1085     name = event.target.name;
1086     value = event.target.value;
1087
1088     setUser({ ...user, [name]: value });
1089   };
1090
1091   const postData = async (e) => {
1092     e.preventDefault();
1093
1094     const {name, email, message} = user;
1095
1096     if(name && email && message){
1097
1098       const res = await fetch(
1099         "https://unsung-angels-default.firebaseio.com/contactusform.json",
1100         {
1101           method : "POST",
1102           headers: {
1103             "content-type": "application/json",
1104           },
1105           body: JSON.stringify({
1106             name,
1107             email,
1108             message,
1109           })
1110         }
1111       );
1112     }
1113   };
1114
1115   return (
1116     <div>
1117       <h1>Contact Us</h1>
1118       <form>
1119         <input type="text" name="name" placeholder="Name" />
1120         <input type="text" name="email" placeholder="Email" />
1121         <input type="text" name="message" placeholder="Message" />
1122         <button type="button" onClick={postData}>Submit</button>
1123       </form>
1124     </div>
1125   );
1126}

const Contactus = () => {
1127
1128   const [user, setUser]=useState({
1129     name:"",
1130     email:"",
1131     message:"",
1132    });
1133
1134   let name, value;
1135   const getUserData =(event)=>{
1136     name = event.target.name;
1137     value = event.target.value;
1138
1139     setUser({ ...user, [name]: value });
1140   };
1141
1142   const postData = async (e) => {
1143     e.preventDefault();
1144
1145     const {name, email, message} = user;
1146
1147     if(name && email && message){
1148
1149       const res = await fetch(
1150         "https://unsung-angels-default.firebaseio.com/contactusform.json",
1151         {
1152           method : "POST",
1153           headers: {
1154             "content-type": "application/json",
1155           },
1156           body: JSON.stringify({
1157             name,
1158             email,
1159             message,
1160           })
1161         }
1162       );
1163     }
1164   };
1165
1166   return (
1167     <div>
1168       <h1>Contact Us</h1>
1169       <form>
1170         <input type="text" name="name" placeholder="Name" />
1171         <input type="text" name="email" placeholder="Email" />
1172         <input type="text" name="message" placeholder="Message" />
1173         <button type="button" onClick={postData}>Submit</button>
1174       </form>
1175     </div>
1176   );
1177}

const Contactus = () => {
1178
1179   const [user, setUser]=useState({
1180     name:"",
1181     email:"",
1182     message:"",
1183    });
1184
1185   let name, value;
1186   const getUserData =(event)=>{
1187     name = event.target.name;
1188     value = event.target.value;
1189
1190     setUser({ ...user, [name]: value });
1191   };
1192
1193   const postData = async (e) => {
1194     e.preventDefault();
1195
1196     const {name, email, message} = user;
1197
1198     if(name && email && message){
1199
1200       const res = await fetch(
1201         "https://unsung-angels-default.firebaseio.com/contactusform.json",
1202         {
1203           method : "POST",
1204           headers: {
1205             "content-type": "application/json",
1206           },
1207           body: JSON.stringify({
1208             name,
1209             email,
1210             message,
1211           })
1212         }
1213       );
1214     }
1215   };
1216
1217   return (
1218     <div>
1219       <h1>Contact Us</h1>
1220       <form>
1221         <input type="text" name="name" placeholder="Name" />
1222         <input type="text" name="email" placeholder="Email" />
1223         <input type="text" name="message" placeholder="Message" />
1224         <button type="button" onClick={postData}>Submit</button>
1225       </form>
1226     </div>
1227   );
1228}

const Contactus = () => {
1229
1230   const [user, setUser]=useState({
1231     name:"",
1232     email:"",
1233     message:"",
1234    });
1235
1236   let name, value;
1237   const getUserData =(event)=>{
1238     name = event.target.name;
1239     value = event.target.value;
1240
1241     setUser({ ...user, [name]: value });
1242   };
1243
1244   const postData = async (e) => {
1245     e.preventDefault();
1246
1247     const {name, email, message} = user;
1248
1249     if(name && email && message){
1250
1251       const res = await fetch(
1252         "https://unsung-angels-default.firebaseio.com/contactusform.json",
1253         {
1254           method : "POST",
1255           headers: {
1256             "content-type": "application/json",
1257           },
1258           body: JSON.stringify({
1259             name,
1260             email,
1261             message,
1262           })
1263         }
1264       );
1265     }
1266   };
1267
1268   return (
1269     <div>
1270       <h1>Contact Us</h1>
1271       <form>
1272         <input type="text" name="
```

4.24 Scholarships Details

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows a project structure with files like `node_modules`, `public`, and `src`. The `src` folder contains `add-scholar`, `components`, `modals`, and several `js` files including `Card.js`, `TodoList.js`, `Details.js`, `Ackno.js`, `blog9.js`, `card9.js`, `data11.js`, and `dump`.
- Code Editor (Center):** The active file is `Details.js`, which imports React, useState, useEffect, and other components from react-router-dom, and emailjs from emailjs.com. It defines a class-based component `Details` that fetches data by ID and sends an email using emailjs.
- Status Bar (Bottom):** Shows the current file as `Details.js - unsung9 - Visual Studio Code`, line 1, column 1, with tabs for `index.html`, `contactus.js`, `index.js`, and `Details.js`. It also displays file statistics (1.0 MB), code analysis results (0 errors, 0 warnings), and system information (Windows 10, Node.js 14.17.0).

4.25 Search for Scholarships

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** On the left, it lists files and folders related to the project, including `data11.js`, `Details.css`, `Details.js`, `dump`, `heading.js`, `niepmrd-logo.png`, `scholar9.js`, `signlang.jpeg`, `create-task.css`, `CreateTask.js`, `data.js`, `EditTask.js`, `firebase.js`, `Scholarship.css`, `Scholarship.js`, `Search.css`, and the current file, `Search.js`.
- Code Editor:** The main area displays the `Search.js` file. The code defines a functional component `Search` that uses the `useState` hook to manage a search filter. It filters an array of data based on the input value and returns a JSX structure with a `marquee` element for a scholarship message and a form input field.
- Status Bar:** At the bottom, it shows "Line 1, Col 1", "Spaces: 4", "UTF-8", "CRLF", "JavaScript", "Go Live", and a timestamp "00:26 12/09/2022".
- Bottom Bar:** It includes icons for file operations like Open, Save, and Close, as well as other developer tools.

5 Test Cases

Table 5.1: Test Case - Login and Register

| Test Case ID | Test Case Name | Test Data | Expected Output | Actual Output | Result |
|--------------|---------------------------------|--|-------------------------|---------------|--------|
| 1 | User enter user id and password | Enters the correct user id and password | Log in Successful | Home Page | Pass |
| 2 | User enter user id and password | Enters the user id and password | Prompt error | Prompt error | Pass |
| 3 | User enter user id and password | Valid user id and password which doesn't exist in Database | Registered Successfully | Login Page | Pass |
| 4 | User enter user id and password | Invalid user id and password which contains in Database | Prompt error | Prompt error | Pass |

Table 5.2: Test Case - Others

| Test Case ID | Test Case Name | Test Data | Expected Output | Actual Output | Result |
|--------------|---|--------------------------|---|----------------------------|--------|
| 1 | User does not enter the required credentials in the Contact Us form | All fields are mandatory | Error message showing all fields required | Message Indicating Success | Fail |
| 2 | User enters all the required fields in the Contact Us form | All fields are mandatory | Successful completion of Contact Us form | Message Indicating Success | Pass |

6 Limitations

- It needs internet to be accessed.
- It does not have a direct link to the scholarship sites.
- User cannot edit all the details filled in the profile form.
- It does not have a feature to apply for actual scholarships present.
- PDF's cannot be created regarding the details of the systems.

7 Future Enhancements

- Automatic PDF generation of the details of the users.
- Feature of applying for scholarships directly from the system.
- Users can get notifications regarding the status of their application whether approved or rejected.

8 User Manual

Part 1 – Register

Upon visiting the platform , the first thing the user needs to do is register on the web application by providing his/her credentials.The user can now proceed to login.

Part 2 – Login

Once registered successfully , the user can login to the system by his/her email id and password which should be correct .Failing to which would not let you to proceed further.

Part 3 – Create Profile

Once the user has successfully signed in to the system , the user can now create his/her profile by filling in the required details.

Part 4 – Apply for Scholarships

After filling the details for profile creation , the user can now apply for scholarships and see which all options are available for him/her.

Part 5 – Filling the details form

After selecting a particular scholarship , the user can click on it to view it in detail and can now fill the details which will provide them with a gratification email instantly.

Part 6 – Filling the contact form

The users can provide their valuable feedback through this feature.

Part 7 – Search for a particular scholarship

The users can search for a particular scholarship by putting the familiar keywords in the search bar.

9 Bibliography

9.1 Web References

- [1.] <https://reactjs.org/>
- [2.] <https://firebase.google.com/docs>
- [3.] <https://www.youtube.com/user/Firebase>
- [4.] <https://stackoverflow.com/>
- [5.] <https://react.school/material-ui>
- [6.] <https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/>
- [7.] <https://www.geeksforgeeks.org/>
- [8.] <https://www.w3schools.com/>
- [9.] <https://www.draw.io/connect/office365/index.html>