

Gutech Jobs

(Web Design Project 2)

Supervised by

Dr Nabil

Done by:

Hajid Alkindi

Table of contents

Project Overview.....	4
Back-end.....	5
SQL.....	5
phpmyadmin.....	5
Firebase storage.....	6
Email Configuration.....	7
API-Endpoints.....	8
Environmental variables.....	8
List of endpoints.....	9
POST /sendemail.....	9
POST /admin-signup.....	10
POST /signup.....	11
POST /login.....	11
GET /jobs.....	12
PUT /updatejob/:jobId.....	13
DELETE /deletejob/:jobId.....	13
PUT /update-application-status/:applicationId/:newStatus.....	13
GET /user-applications.....	13
POST /apply.....	14
POST /add-job.....	14
GET /skills.....	15
GET /admin/jobs.....	15
GET /job/:jobId.....	16
GET /job-applications.....	16
DELETE /delete-application/:applicationId.....	16
Front-End.....	16
React.js.....	16
React-Bootstrap.....	16
Screenshots Demonstration.....	18
Login (Admin & User).....	18
Signup.....	18
User & Admin Sign Up.....	18
Homepage.....	19
User View.....	20
Navigation menu.....	20
View Jobs.....	20
Filter function.....	20
Apply for Job.....	21
My applications.....	21
Contact us.....	22
Admin View.....	23

Navigation menu.....	23
View Jobs.....	24
Add Jobs.....	24
Admin section.....	26
My Jobs.....	26
Delete Job Function.....	26
Update Job Function.....	27
View job Applications.....	28
Conclusion.....	31
References.....	32

Project Overview

The project's goal is to develop a web application that simplifies the process of job searching and job hiring. It gives users a place to look through a variety of job postings, apply for jobs, and monitor the progress of their applications. The application provides administrators with strong functionalities to handle job listings, handle applications, and carry out specific actions like approval, rejection, or deletion.

This application makes use of two famous JavaScript frameworks: Express.js and Node.js for the back end and React.js for the front end. Using the knowledge I gained from GUtech's "Database Management" course, I have created a database design that is effective and low in redundancy. The project also uses email integration, Firebase storage for effective data management, and a variety of API requests for handling dynamic data.

In summary, The project has aided me to understand many concepts and possibilities in full stack development, Moving forward, I am excited to apply my knowledge to real world applications.

Back-end

SQL

Relational database systems are the most commonly used type of database systems, as it's very easy to create, and work with, however my reasoning towards choosing to use a relational database system was mainly due to scalability, for a project like this, it is expected to have many job listings and applicants and administrators working on the application and a NOSQL solution would have made the application much more complicated, therefore I've used a SQL database system.

phpmyadmin

Table	Action	Rows	Type	Collation	Size	Overhead
applications	☆ - Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_general_ci	48.0 KiB	-
jobs	☆ - Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_general_ci	32.0 KiB	-
job_skills	☆ - Browse Structure Search Insert Empty Drop	7	InnoDB	utf8mb4_general_ci	32.0 KiB	-
skills	☆ - Browse Structure Search Insert Empty Drop	49	InnoDB	utf8mb4_general_ci	16.0 KiB	-
users	☆ - Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_general_ci	16.0 KiB	-
5 tables	Sum	65	InnoDB	utf8mb4_general_ci	144.0 KiB	0 B

Firebase storage

The project involved the process of storing images, and pdf files, such as personal photos and resumes of applicants, and for simplicity sake, i have decided to go with firebase storage, as it provided a free plan for storage with 1 free GB for usage, in addition it also provided excellent Javascript SDK to integrate with React.

Firebase SDK

SDK setup and configuration

npm CDN Config

If you're already using [NPM](#) and a module bundler such as [webpack](#) or [Rollup](#), you can run the following command to install the latest SDK ([Learn more](#)):

```
$ npm install firebase
```

Then, initialise Firebase and begin using the SDKs for the products that you'd like to use.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "[REDACTED]",
  authDomain: "gutechjobs.firebaseio.com",
  projectId: "gutechjobs",
  storageBucket: "gutechjobs.appspot.com",
  messagingSenderId: "[REDACTED]",
  appId: "[REDACTED]",
  measurementId: "[REDACTED]"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
```

Note: This option uses the [modular JavaScript SDK](#), which provides a reduced SDK size.

Learn more about Firebase for web: [Get started](#), [Web SDK API Reference](#), [Samples](#)

Firebase SDK Initialization

```
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";

const firebaseConfig = {
  apiKey: "XXXXXXXXXXXXXX",
  authDomain: "gutechjobs.firebaseio.com",
  projectId: "gutechjobs",
  storageBucket: "gutechjobs.appspot.com",
  messagingSenderId: "XXXXXXXXXXXXXX",
  appId: "XXXXXXXXXXXXXX",
  measurementId: "XXXXXXXXXXXXXX"
};

// Initialize Firebase
const firebaseapp = initializeApp(firebaseConfig);
const analytics = getAnalytics(firebaseapp);

export default firebaseapp;
```

Firebase.js

Email Configuration

The Project had few cases where emails were sent to administrators or applicants, in which I've utilised SendGrid free plan that offers free 100 emails sending per day, and I'm connecting to this service using an api key provided by sendgrid.

NAME	API KEY	ACTION
GutechJobs API Key ID: y21dH7QuSA6wzE8wuxta5A	copy · refresh	trash

API-Endpoints

One of the biggest challenges with SQL databases, is the speed it takes to perform read and write operations between client side and server side, in which api end-points were the solution to this issue, Define APIs

Environmental variables

Environmental variables are variables that stored in .env file for security purposes, and those variables in my case include the following:

1. JWT Secret Key
2. SendGRID Api key
3. Sender Email
4. Database Host
5. Database username
6. Database Password
7. Database name

```
JWT_SECRET=  
SENDGRID_API_KEY=  
SENDER_EMAIL=21-0099@student.gutech.edu.om  
  
DB_HOST=localhost  
DB_USER=Hajid  
DB_PASSWORD=  
DB_NAME=reactproject
```

That later can be utilised in express js endpoints by

```
require("dotenv").config();
```

And using the variables as following:

```
sgMail.setApiKey(process.env.SENDGRID_API_KEY);  
  
const db = mysql.createConnection({  
    host: process.env.DB_HOST,  
    user: process.env.DB_USER,  
    password: process.env.DB_PASSWORD,  
    database: process.env.DB_NAME  
});
```

List of endpoints

POST /sendemail

```
app.post('/sendemail', async (req, res) => {
  const { to, subject, text } = req.body;

  const msg = {
    to: to,
    from: process.env.SENDER_EMAIL, r
    subject: subject,
    text: text,
  };

  try {
    await sgMail.send(msg);
    res.status(200).send('Email sent successfully');
  } catch (error) {
    console.error(error);

    if (error.response) {
      console.error(error.response.body)
    }
    res.status(500).send('Error in sending email');
  }
});
```

Send an email using SendGrid. Requires recipient details in the request body.

POST /admin-signup

```
const saltRounds = 10;

app.post('/admin-signup', (req, res) => {
    const { username, password, email } = req.body;

    db.query('SELECT * FROM users WHERE username = ? OR email = ?', [username, email], (err, results) => {
        if (err) {
            console.error(err);
            return res.status(500).json({ error: 'Error checking for existing user' });
        }

        if (results.length > 0) {
            // Username or email already exists
            return res.status(409).json({ error: 'Username or email already exists' });
        }

        // Hashing the password
        bcrypt.hash(password, saltRounds, (hashErr, hash) => {
            if (hashErr) {
                console.error(hashErr);
                return res.status(500).json({ error: 'Error hashing password' });
            }

            // Store admin user with hashed password
            db.query('INSERT INTO users (username, password, email, user_type) VALUES (?, ?, ?, "admin")',
                [username, hash, email],
                (insertErr, insertResults) => {
                    if (insertErr) {
                        console.error(insertErr);
                        return res.status(500).json({ error: 'Error signing up' });
                    }
                    res.status(200).json({ message: 'Admin signup successful' });
                });
        });
    });
});
```

Allows admin users to sign up. Checks for existing users and stores hashed passwords.

POST /signup

```
app.post('/signup', (req, res) => {
  const { username, password, email } = req.body;

  // First, check if the username or email already exists
  db.query('SELECT * FROM users WHERE username = ? OR email = ?', [username, email], (err, results) => {
    if (err) {
      console.error(err);
      return res.status(500).json({ error: 'Error checking for existing user' });
    }

    if (results.length > 0) {
      // Username or email already exists
      return res.status(409).json({ error: 'Username or email already exists' });
    }

    // Hash the password
    bcrypt.hash(password, saltRounds, (hashErr, hash) => {
      if (hashErr) {
        console.error(hashErr);
        return res.status(500).json({ error: 'Error hashing password' });
      }

      // Store user with hashed password
      db.query('INSERT INTO users (username, password, email, user_type) VALUES (?, ?, ?, "normal")',
        [username, hash, email],
        (insertErr, insertResults) => {
          if (insertErr) {
            console.error(insertErr);
            return res.status(500).json({ error: 'Error signing up' });
          }
          res.status(200).json({ message: 'Signup successful' });
        });
    });
  });
});
```

Allows normal users to sign up. Similar to admin signup but assigns a different user type.

POST /login

```
app.post('/login', (req, res) => {
  const { username, password } = req.body;

  db.query('SELECT * FROM users WHERE username = ?', [username], (err, results) => {
    if (err) {
      console.error(err);
      return res.status(500).json({ error: 'Error logging in' });
    }

    if (results.length > 0) {
      const user = results[0];

      bcrypt.compare(password, user.password, (error, isMatch) => {
        if (error) {
          console.error(error);
          return res.status(500).json({ error: 'Error checking password' });
        }

        if (isMatch) {
          // Passwords match, create a token
        }
      });
    }
  });
});
```

```

        const token = jwt.sign(
          { id: user.id, username: user.username },
          process.env.JWT_SECRET,
          { expiresIn: '24h' } // Token expires in 24 hours
        );

        // Remove the password from the user object before sending
        const { password, ...userWithoutPassword } = user;

        res.status(200).json({
          message: 'Login successful',
          token,
          user: userWithoutPassword
        });
      } else {
        // Passwords do not match
        res.status(400).json({ message: 'Username or password is incorrect' });
      }
    );
  } else {
    // User not found
    res.status(400).json({ message: 'Username or password is incorrect' });
  }
});
);
}
);
```

```

Authenticates a user and provides a JWT token for session management.

GET /jobs

```

pp.get('/jobs', verifyToken, (req, res) => {
 const query = `
 SELECT j.job_id, j.title, j.type, j.salary, j.description,
 j.picture, j.created_at, j.updated_at,
 GROUP_CONCAT(s.name SEPARATOR ', ') AS skills
 FROM jobs j
 LEFT JOIN job_skills js ON j.job_id = js.job_id
 LEFT JOIN skills s ON js.skill_id = s.skill_id
 GROUP BY j.job_id
 `;

 db.query(query, (err, results) => {
 if (err) throw err;
 res.json(results);
 });
});
```

```

Returns list of jobs posted on the site

PUT /updatejob/:jobId

Updates the details of a job specified by jobId. Handles associated skills.

DELETE /deletejob/:jobId

Deletes a job and its associated skills from the database.

PUT /update-application-status/:applicationId/:newStatus

Updates the status of a job application to either 'rejected' or 'accepted'.

GET /user-applications

```
app.get('/user-applications', (req, res) => {
  const userId = req.query.userId;
  console.log(userId);
  db.query('SELECT * FROM applications WHERE user_id = ?', [userId],
  (err, results) => {
    if (err) {
      console.error(err);
      return res.status(500).json({ error: 'Error fetching
applications' });
    }
    res.json(results);
  });
});
```

Fetches applications for a specific user, based on userId.

POST /apply

```
app.post('/apply', (req, res) => {
  console.log(req.body);
  const { userId, jobId, name, email, personalPhoto, cv } = req.body;
  const query = "INSERT INTO applications (user_id, job_id, name,
email, personal_photo, cv) VALUES (?, ?, ?, ?, ?, ?)";
  db.query(query, [userId, jobId, name, email, personalPhoto, cv],
(err, results) => {
  if (err) {
    console.error(err);
    return res.status(500).json({ error: 'Error submitting
application' });
  }
  res.status(200).json({ message: 'Application submitted
successfully' });
});
});
```

Allows a user to apply for a job. Submits application details to the database.

POST /add-job

```
app.post('/add-job', (req, res) => {
  const { user_id, title, type, salary, description, picture, skills } = req.body;
  const jobQuery = "INSERT INTO jobs (user_id, title, type, salary, description, picture) VALUES (?, ?, ?, ?, ?, ?)";
  // First, insert the job into the jobs table
  db.query(jobQuery, [user_id, title, type, salary, description, picture], (err, results) => {
    if (err) {
      console.error(err);
      return res.status(500).json({ error: 'Error adding job' });
    }
    const jobId = results.insertId;
    // Now, insert skills associated with the job
    if (skills && skills.length) {
      const skillQuery = "INSERT INTO job_skills (job_id, skill_id) VALUES ?";
      const skillValues = skills.map(skillId => [jobId, skillId]);
      db.query(skillQuery, [skillValues], (err, results) => {
        if (err) {
          console.error(err);
          return res.status(500).json({ error: 'Error adding job skills' });
        }
        res.status(200).json({ message: 'Job and job skills added successfully' });
      });
    } else {
      res.status(200).json({ message: 'Job added successfully, no skills to add' });
    }
  });
});
```

Allows an admin to add a new job listing, including associated skills.

GET /skills

Retrieves a list of all available skills from the database.

GET /admin/jobs

```
app.get('/admin/jobs', (req, res) => {
  const adminId = req.query.adminId;

  db.query('SELECT * FROM jobs WHERE user_id = ?', [adminId], (err, jobs) => {
    if (err) {
      console.error(err);
      return res.status(500).json({ error: 'Error fetching jobs' });
    }

    const jobsWithSkillsPromises = jobs.map(job => {
      return new Promise((resolve, reject) => {
        db.query('SELECT skills.* FROM job_skills JOIN skills ON job_skills.skill_id = skills.skill_id WHERE job_skills.job_id = ?', [job.job_id], (err, skills) => {
          if (err) {
            return reject(err);
          }
          job.skills = skills.map(skill => ({ value: skill.skill_id, label: skill.name }));
          resolve(job);
        });
      });
    });

    Promise.all(jobsWithSkillsPromises)
      .then(jobsWithSkills => res.json(jobsWithSkills))
      .catch(err => {
        console.error(err);
        res.status(500).json({ error: 'Error fetching job skills' });
      });
  });
});
```

Fetches all jobs posted by a specific admin user.

GET /job/:jobId

Fetches details for a specific job based on jobId.

GET /job-applications

Retrieves applications for a specific job, based on jobId.

DELETE /delete-application/:applicationId

Deletes a specific job application based on applicationId.

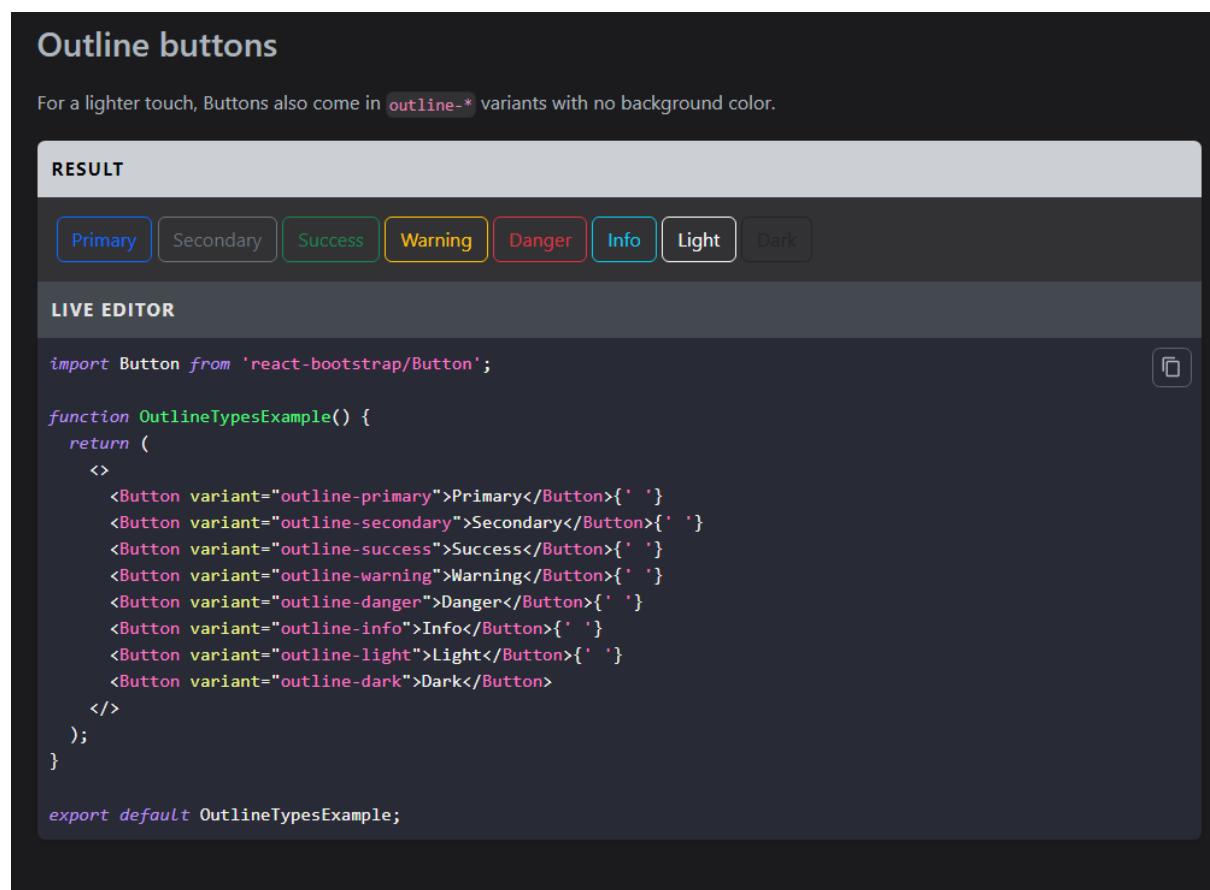
Front-End

React.js

React js is a javascript library used for building user interfaces, react is a easy to use framework that has excellent documentation that is very easy for intermediate programmers to use.

React-Bootstrap

React-bootstrap is a UI library that provides easy to use components such as buttons, cards, navigation menu components that a user can easily integrate in their code and utilise to produce amazing looking user interfaces, in addition react bootstrap provides very good documentation on how to implement those components and initialise react-bootstrap in the project.



The screenshot shows a dark-themed interface for a live code editor. At the top, there's a header with the title "Outline buttons". Below the header, a sub-header says "For a lighter touch, Buttons also come in `outline-*` variants with no background color." Underneath this, there's a "RESULT" section containing a row of eight buttons labeled "Primary", "Secondary", "Success", "Warning", "Danger", "Info", "Light", and "Dark". Below the result section is a "LIVE EDITOR" section containing the following code:

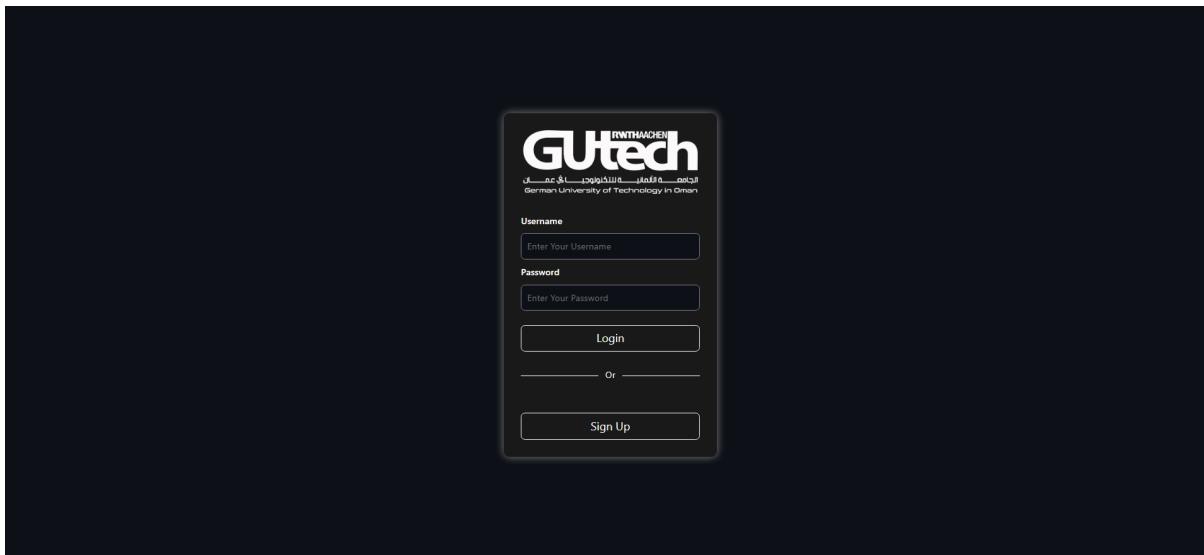
```
import Button from 'react-bootstrap/Button';

function OutlineTypesExample() {
  return (
    <>
      <Button variant="outline-primary">Primary</Button>{' '}
      <Button variant="outline-secondary">Secondary</Button>{' '}
      <Button variant="outline-success">Success</Button>{' '}
      <Button variant="outline-warning">Warning</Button>{' '}
      <Button variant="outline-danger">Danger</Button>{' '}
      <Button variant="outline-info">Info</Button>{' '}
      <Button variant="outline-light">Light</Button>{' '}
      <Button variant="outline-dark">Dark</Button>
    </>
  );
}

export default OutlineTypesExample;
```

Screenshots Demonstration

Login (Admin & User)

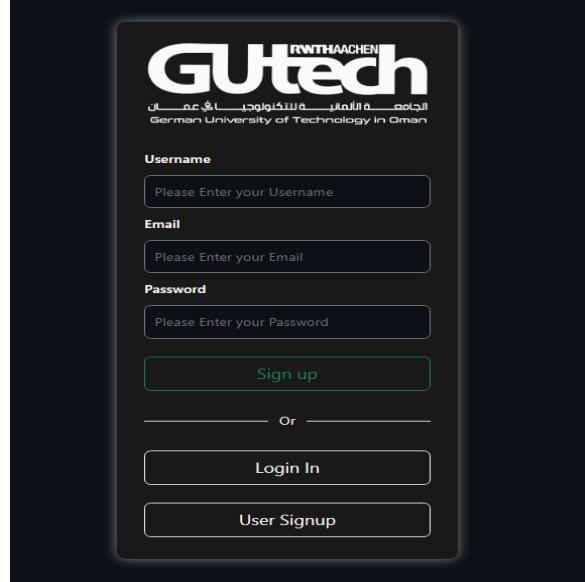
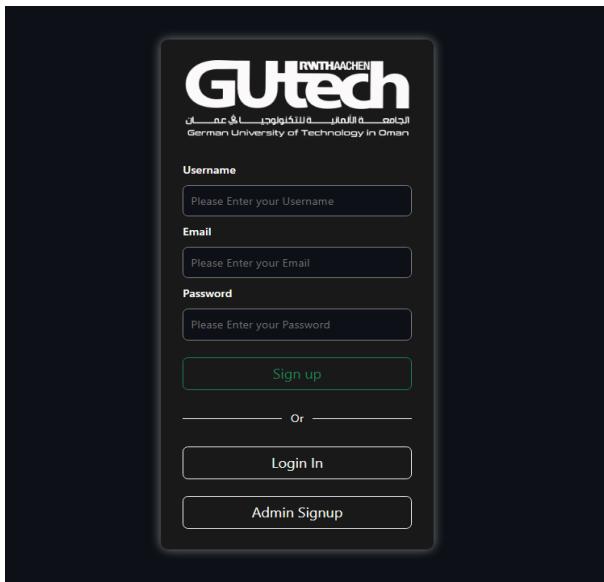


The Program is able to determine the user type either admin or user and redirect them to the corresponding navigation view, if the user doesn't have an account they can click on the sign up screen which will redirect them to sign up.

Signup

There are two sign up screens, user and admin signup and can be toggled through through user signup button and admin signup button.

User & Admin Sign Up



Homepage

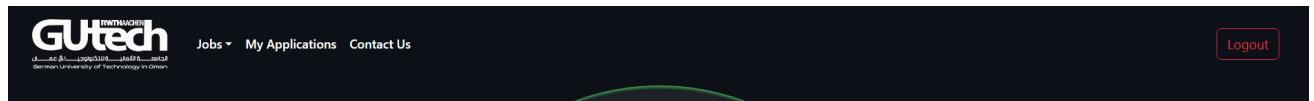
This is the homepage view that is accessible to all user types, the get started button redirects you to /jobs and in addition the home page displays the team.

The screenshot shows the Gutech Jobs homepage with a dark background and a large green circular graphic element. At the top, there is a navigation bar with the Gutech logo, 'Jobs', 'Admin', and 'Logout' buttons. The main heading 'Gutech Jobs' is displayed prominently in white, followed by the tagline 'Your Gateway to Career Success - Apply with Gutech Jobs Today!'. A 'Get Started' button is located below the tagline. The overall design is professional and modern.

The screenshot shows the Gutech Team page with a dark background. The title 'TEAM' is centered at the top. Below it, a paragraph introduces the dynamic team: 'Meet our dynamic team! Comprised of creative minds and dedicated professionals, our team members are the driving force behind our success. Each brings unique skills and a shared commitment to excellence. Get to know the people making a difference in our company.' The page is divided into four quadrants, each featuring a team member's profile: Hajid Alkindi (Chief Executive Officer), Mike Hanna (Full-Stack Developer), Abdullah Alriyami (AI Specialist), and Sameer Khalil (Cyber Security Specialist). Each profile includes a circular photo, the team member's name, their title, and a brief quote or description.

User View

Navigation menu



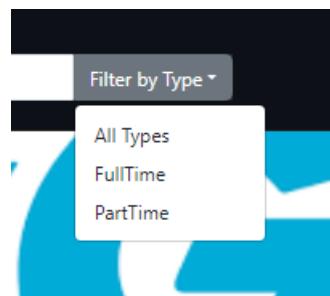
The user menu allows user to browse through, homepage, jobs view, contact us and my application pages

View Jobs

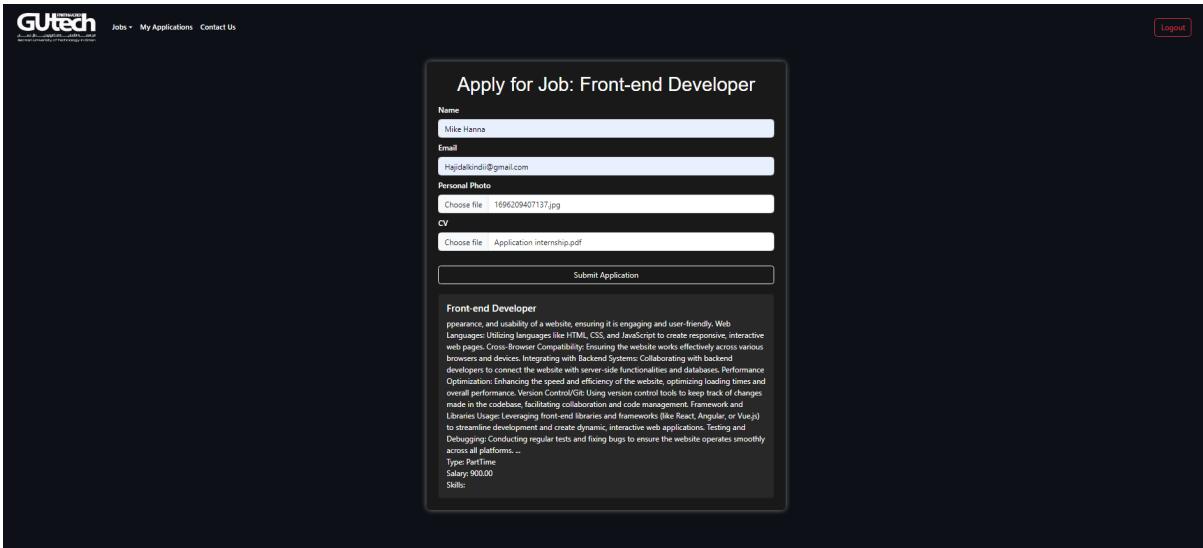
Job Title	Description	Type	Salary	Skills
Data scientist	Data Collection and Cleaning: Gathering data from various sources and ensuring its accuracy and integrity.	PartTime	600.00	Python, SQL, Communication, Teamwork
Back End Developer	Server, Database, and Application Logic: Developing and maintaining the core functional logic and op...	FullTime	600.00	Python, SQL, Azure, Communication, Problem-solving
Front-end Developer	Appearance, and usability of a website, ensuring it is engaging and user-friendly. Web Languages: Util...	PartTime	900.00	Python, JavaScript, HTML, CSS, React, Communication

The user is able to search for jobs by name and also is able to filter that based on part time jobs and full time jobs, and he is able to click apply on a job and be redirected to the next page.

Filter function



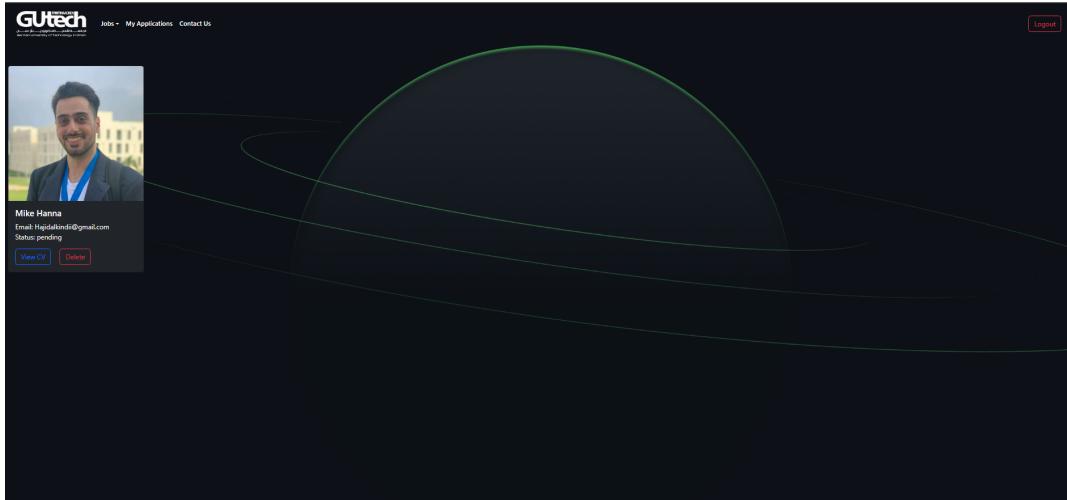
Apply for Job



The screenshot shows a job application form titled "Apply for Job: Front-end Developer". The form includes fields for Name (Mike Hanna), Email (Haydakindii@gmail.com), Personal Photo (choose file: 16946209407137.jpg), CV (choose file: Application internship.pdf), and a "Submit Application" button. Below the form is a detailed job description for a Front-end Developer, listing responsibilities such as creating responsive web pages, integrating with backend systems, and ensuring performance. It also mentions version control, testing, and deployment. At the bottom, it specifies part-time status, a salary of 900.00, and lists skills.

Users will fill in application details, such as name, email and upload personal photo and CV which is stored in firebase storage and the link to the firebase storage is stored in the SQL database for reference.

My applications



My applications page will display the applications that were submitted by the current logged in user, and show their status(pending, rejected or approved), when a user gets their application updated, they will receive this email:

Successful Application Inbox x

 21-0099@student.gutech.edu.om via sendgrid.net
to Hajidalkindii ▾

Dear Mr/Mrs Mike Hanna,

You have successfully applied for the job: Front-end Developer.

Best regards,
Gutech Jobs

↶ Reply ↷ Forward

Contact us

Jobs - My Applications Contact Us Logout

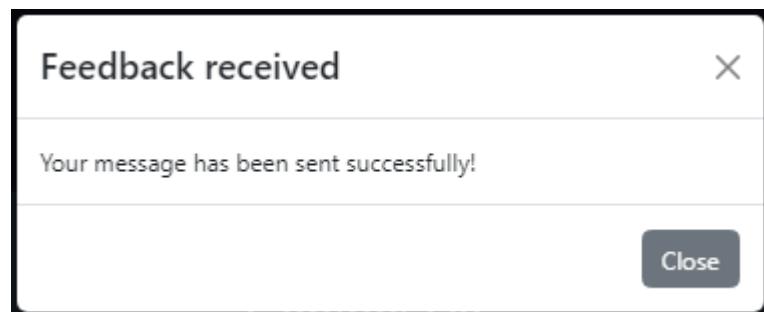
Contact Us

Name: Sameer Khalil Email: SameerKhalil@gmail.com

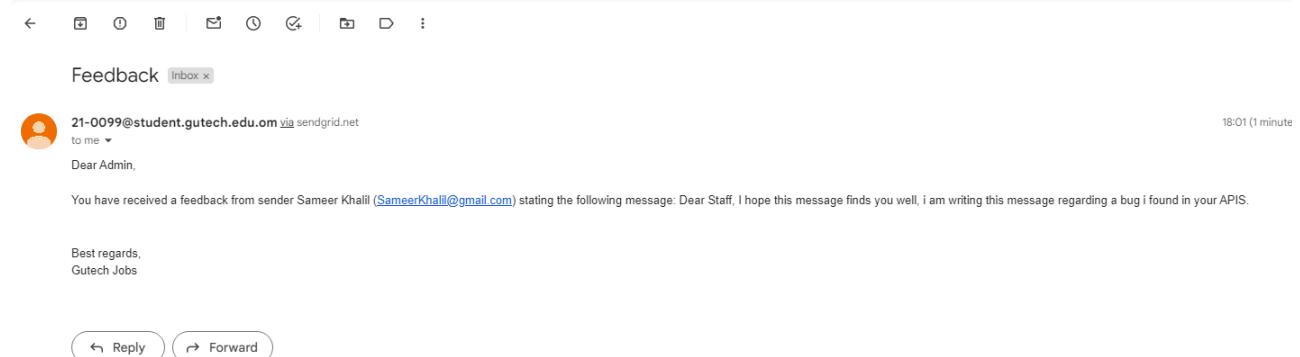
Subject: Bug found

Message:
Dear Staff, I hope this message finds you well, i am writing this message regarding a bug i found in your APIS.

Send Message



This page provides a contact us form, which will send an email feedback to the admin, like this:



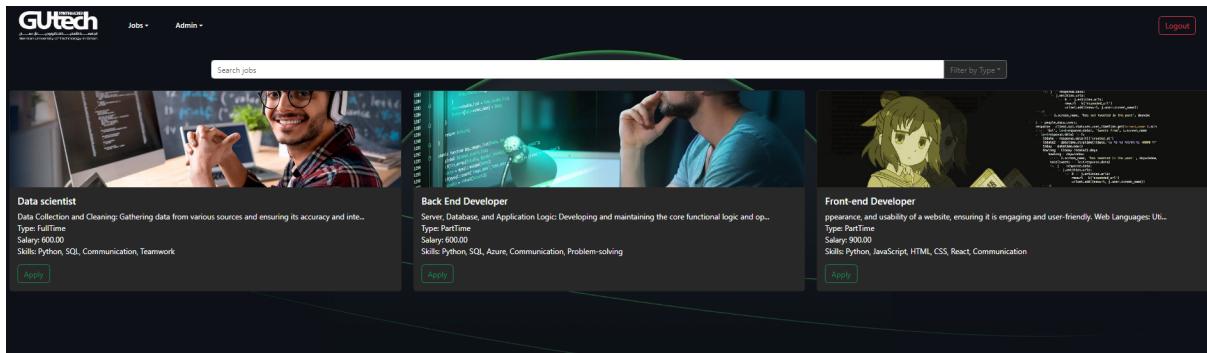
Admin View

Navigation menu

The screenshot shows the GUTech admin dashboard. At the top, there is a navigation bar with the GUTech logo, 'Jobs', 'Admin', and a 'Logout' button. Below the navigation bar, there is a large green circular graphic. The main area has a dark background with white text. It features the GUTech logo again, followed by 'Jobs' and 'Admin' dropdown menus. A central button labeled 'My Jobs' is visible. To the right, there is another section with 'View' and 'Add' buttons.

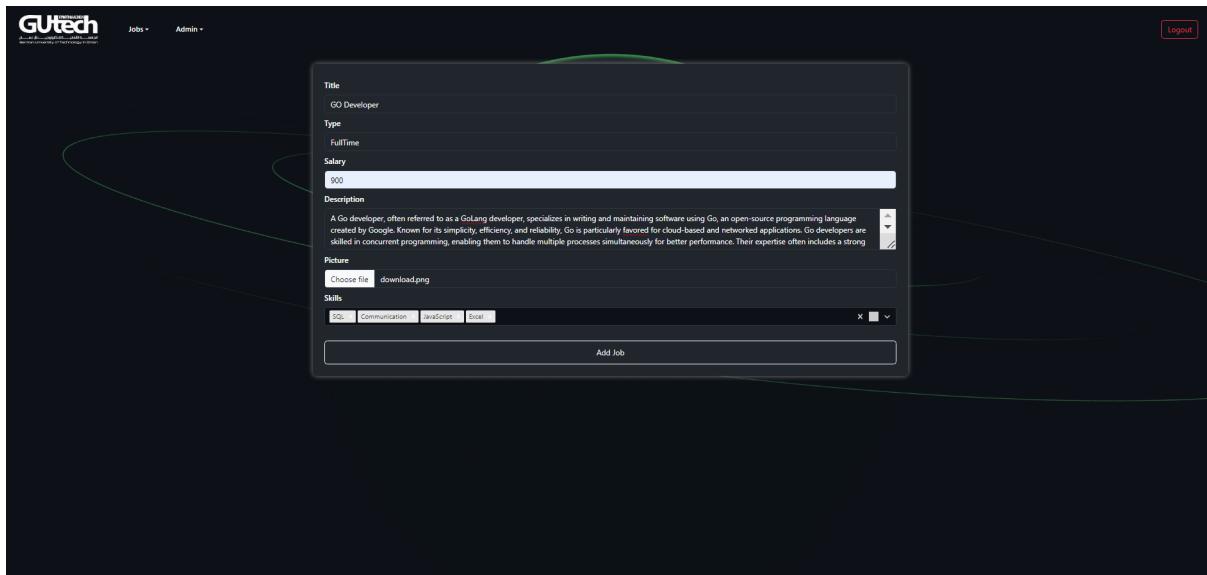
The admin menu allows user to browse through, homepage, jobs view & add pages and admin jobs (my jobs)

View Jobs

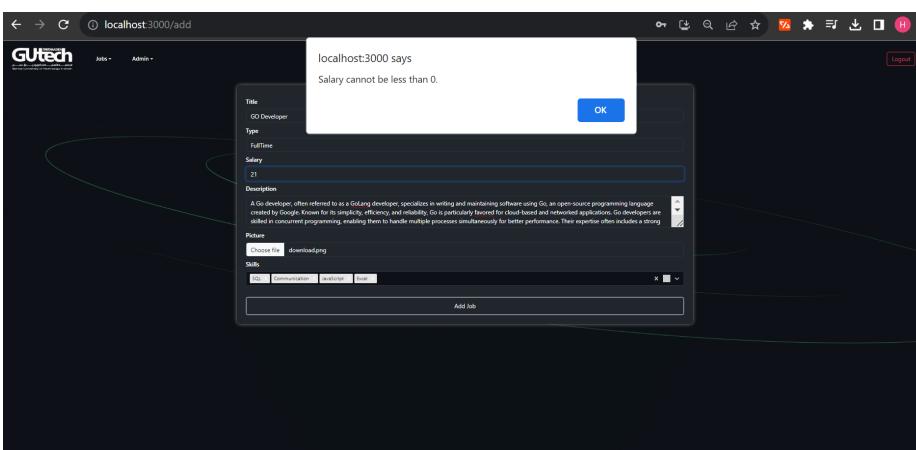
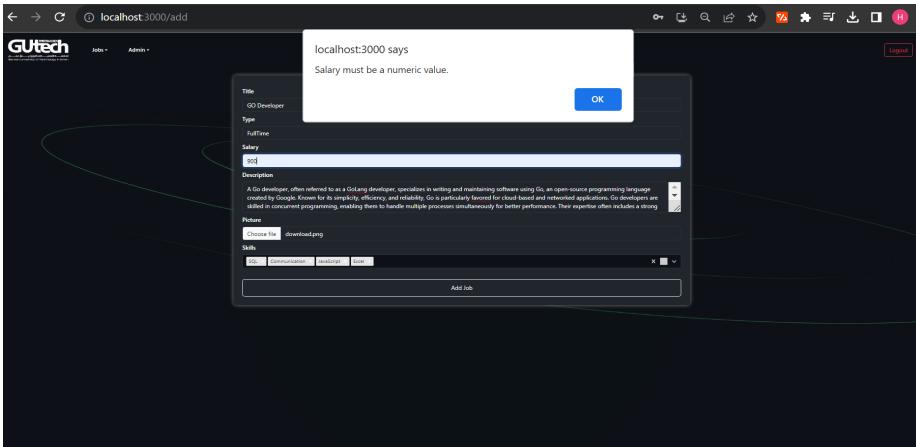


The view is very simple, pretty much the same as the user page that displays the available jobs with the filter logic.

Add Jobs



This page will allow an admin to post a job listing, and fill out details such as salary, title, description, choose set of skills and the form has input checks such that for example salary cannot be less than 0 and cannot be letters:



Additionally when admin added a job they will receive this email:

 21-0099@student.gutech.edu.om via sendgrid.net
 to Hajdalkindii ▾
 18:11 (2 minutes ago) ⌂ ⌃ ⌄

Dear Admin, You have successfully created the following job:

Job Title: GO Developer

Job Description: A Go developer, often referred to as a GoLang developer, specializes in writing and maintaining software using Go, an open-source programming language created by Google. Known for its simplicity, efficiency, and reliability, Go is particularly favored for cloud-based and networked applications. Go developers are skilled in concurrent programming, enabling them to handle multiple processes simultaneously for better performance. Their expertise often includes a strong understanding of system programming, microservices architectures, and containerization technologies like Docker and Kubernetes. They are adept at troubleshooting and optimizing code for scalable and high-performing applications, making them valuable in fast-paced and technologically evolving environments.

Job Type: FullTime

Job Salary: 900

Best regards,
Gutech Jobs

[Reply](#) [Forward](#)

Admin section

My Jobs

The screenshot shows the 'My Jobs' section of the GUTech admin interface. It displays four job postings with their respective descriptions and application buttons:

- Data scientist**: Describes the role of collecting and cleaning data, using statistical techniques to identify trends or patterns, building predictive models, and applying data-driven approaches to solve complex business problems.
- Back End Developer**: Describes the role of developing and maintaining core functional logic and operations of software applications, creating APIs, and integrating with databases.
- Front-end Developer**: Describes the role of ensuring website appearance and user-friendliness, using languages like HTML, CSS, and JavaScript.
- GO Developer**: Describes the role of writing code that runs on servers, often using Go, Python, Ruby, or PHP.

This page will allow an admin to view their posted jobs, and be able to create changes to the job postings such as, update, delete and additionally they can also view the applications submitted by the users for a job

Delete Job Function

The screenshot shows the 'Delete Job Function' feature. A modal dialog box titled 'Confirm Deletion' asks 'Are you sure you want to delete this job?'. The background shows the same job listing structure as the previous screenshot, with the GO Developer job visible.

The screenshot shows the GUTech Admin interface at localhost:3000/admin/jobs. It displays three job listings:

- Back End Developer**: Descriptions include Server, Database, and Application Logic; API Creation and Management; Building and maintaining APIs (Application Programming Interfaces) that allow the system to communicate with other software; Database Management; Creating, managing, and optimizing databases to store and retrieve data efficiently; Server-Side Scripting; Writing code that runs on the server, often using languages like Java, Python, Ruby, or PHP.
- Front-end Developer**: Descriptions include Appearance, and Usability of a website, ensuring it is engaging and user-friendly; Web Languages: Utilizing languages like HTML, CSS, and JavaScript to create responsive, interactive web pages; Cross-Browser Compatibility: Ensuring the website works effectively across different browsers and devices; Integrating with Backend Systems: Ensuring the website has a solid foundation to connect the user interface to the back-end functionalities and databases; Performance Optimization: Enhancing the speed and efficiency of the website, optimizing loading times and overall performance.
- GO Developer**: Descriptions include A Go developer, often referred to as a GoLang developer, specializes in writing and maintaining software using Go, an open-source programming language created by Google. Known for its simplicity, efficiency, and reliability, Go is particularly favored for cloud-based and networked applications. Go developers are skilled in concurrent programming, enabling them to handle multiple processes simultaneously for better performance. Their expertise often includes a strong understanding of system programming, microservices architectures, and containerization technologies like Docker and Kubernetes. They are adept at troubleshooting and optimizing code for scalable and high-performing applications, making them valuable in fast-paced and technologically evolving environments.

Each listing includes "View Applications", "Delete Job", and "Update" buttons.

Admin is able to delete a job listing, and when deleted all the applicants associated with that job will also be deleted from the database.

Update Job Function

Admin is able to update job details, such as changing the title, description , set of skills and so on.

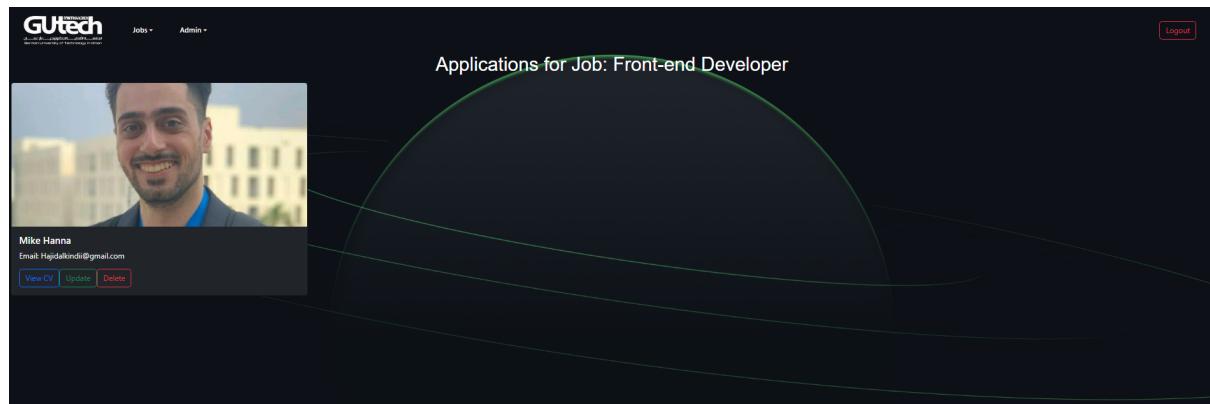
The screenshot shows the GUTech Admin interface at localhost:3000/admin/jobs. A modal window titled "Update Job" is open for the "GO Developer" position. The modal fields are as follows:

Title	GO Developer
Type	FullTime
Salary	900.00
Description	A Go developer, often referred to as a GoLang developer, specializes in writing and maintaining software using Go, an open-source programming language created by Google. Known for its simplicity, efficiency, and reliability, Go is particularly favored for cloud-based and networked applications. Go developers are skilled in concurrent programming, enabling them to handle multiple processes simultaneously for better performance. Their expertise often includes a strong understanding of system programming, microservices architectures, and containerization technologies like Docker and Kubernetes. They are adept at troubleshooting and optimizing code for scalable and high-performing applications, making them valuable in fast-paced and technologically evolving environments.
Skills	JavaScript, SQL, Node.js, Communication

The modal has a "Update" button at the bottom. The background shows the same job listings as the previous screenshot.

Updating job...

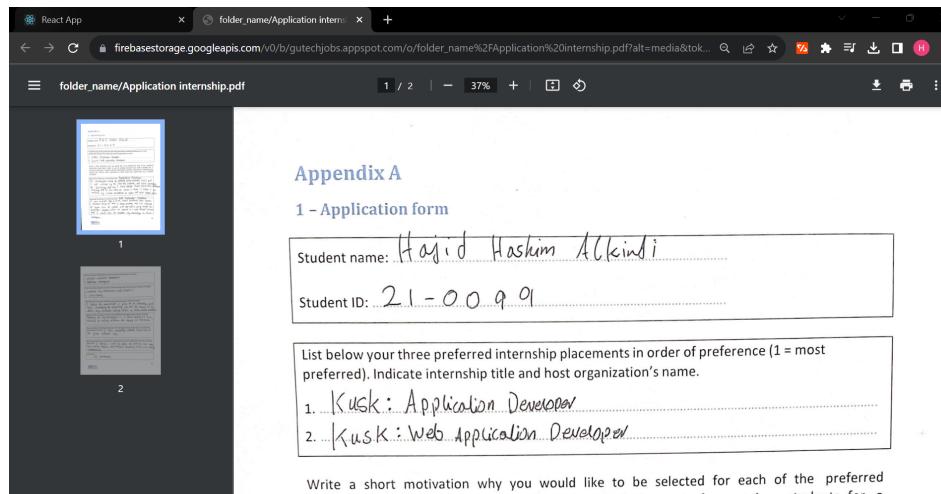
View job Applications



The page will show the submitted applications that the users submitted for a job.

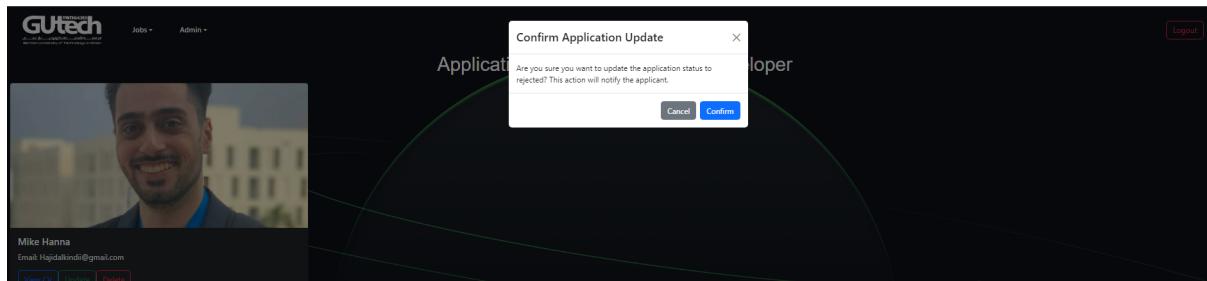
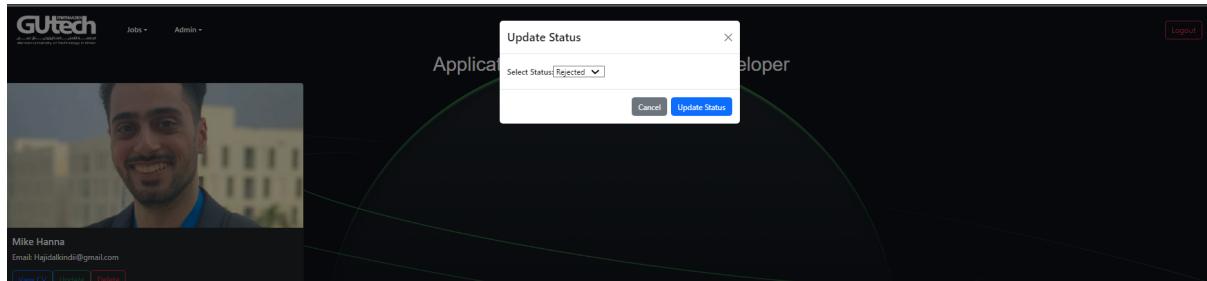
The admin then can:

1. View CV



View the CV file stored in firebase storage.

2. Update Application status



Update an application status, to either pending, rejected or approved and when updated the associated applicant that status was changed will receive an email like this:

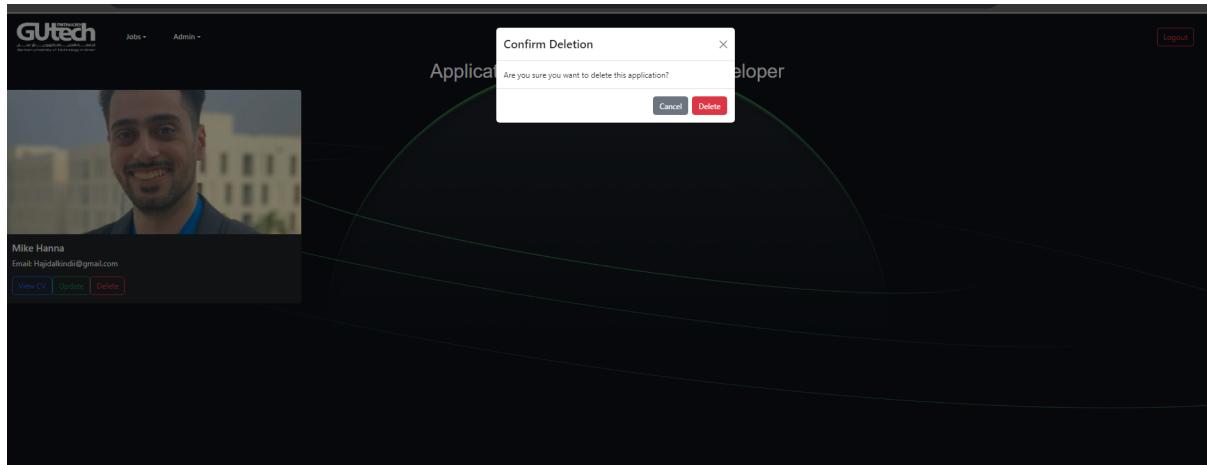
 21-0099@student.gutech.edu.om via sendgrid.net
to Hajidalkindi ▾

Dear User, your application has been rejected for the job title: Front-end Developer.

Best regards,
Gutech Jobs

3. Delete Application



Delete a user's application if the admin wishes to.

Conclusion

As we approach the conclusion of this endeavour, it's important for me to pause and reflect on the journey - the highs and lows encountered. Merging the elements of front-end and back-end tech to create an integrated, functional tool has been a truly eye-opening experience.

References

<https://react.dev/>

<https://www.w3schools.com/REACT/DEFAULT.ASP>

<https://www.youtube.com/watch?v=QrVYLLpoyMw>

<https://react-bootstrap.netlify.app/>