ETU student profile system

Phase – 2

Public URL of the system: http://18.142.231.1/

Student name: Rishiharun Balashanmugam

Introduction

ETUX Technical University (ETU) is a newly established university with many degree programs to follow. The university now wants to introduce a system to showcase every final year students' profile regardless of the degree program they're following. The main purpose of this system is that these profiles can be viewed by industry people, and they can hire students if the profile matches their requirements.

About the system

The system typically contains following functionalities:

1) Student signup.
2) Student login.
3) Student profile view/edit.
4) Public student profile view.

Technologies used to develop the system:

Front end

1) HTML
2) CSS
3) JavaScript

Backend

1) Python
2) Flask (python)

Databases

1) Amazon DynamoDB (for storing user details)
2) Amazon S3 bucket (for storing user images)

Hosted in

1) AWS EC2 instance

AWS cloud9 is used as IDE for developing the system. fetch API is also used to communicate between the frontend and the server.

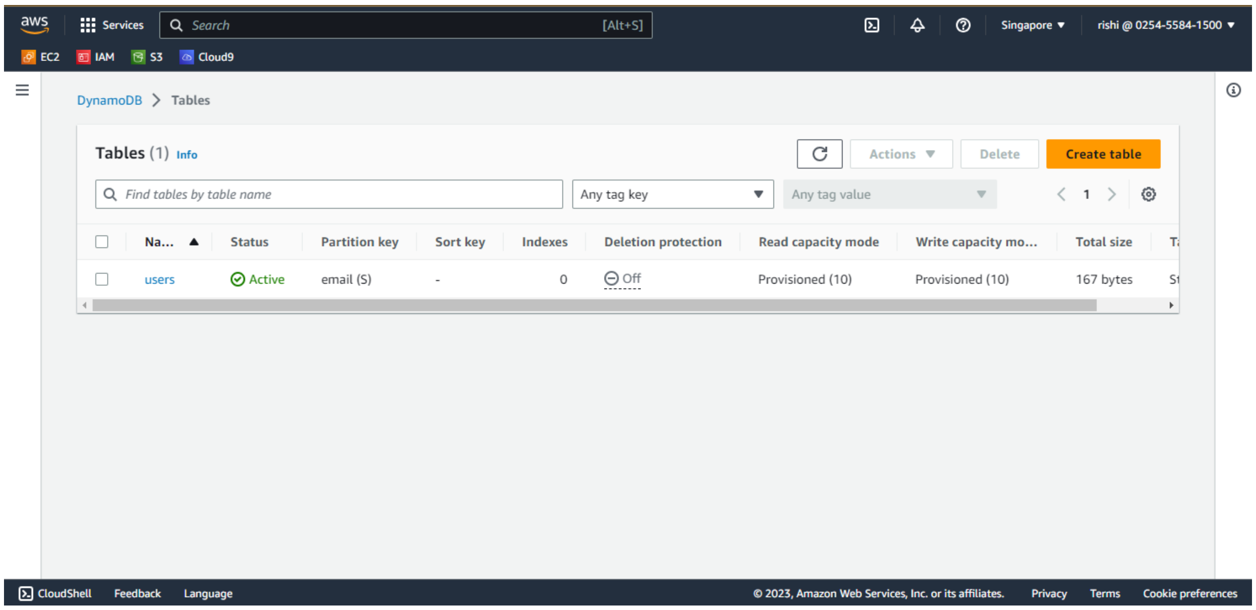
Implementation of the system

1) Setting up the environment

- Before starting to develop the system, first created a cloud9 environment and created a folder called "ETU-student-profile-app" inside this cloud9 environment. All the files related to the app will be in this folder.
- Downloaded the required resources such as boto3 and flask framework in the terminal.
- Created a templates folder to save all the html files here.
- A file named "key_config.py" is created and the account ID and region name is specified here.
- File named "dynamodb_handler.py" for handling database operations.
- File named "app.py" for main server functionalities.
- File named "style.css" inside a folder called "static" for styling purposes.

2) Creating table in database

- A table called "users" was created with email being the partition key.
- To create it the function called "create_user_table" was created in the dynamodb_handler file and called it in the index route.
- Soon after creating the table this code was commented to make sure whenever index route is loaded it doesn't create new tables anymore.



Na...	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mo...	Total size	T
users	Active	email (S)	-	0	Off	Provisioned (10)	Provisioned (10)	167 bytes	S

3) Student signup

Every student must sign up before proceeding further. The system collects following information from every student:

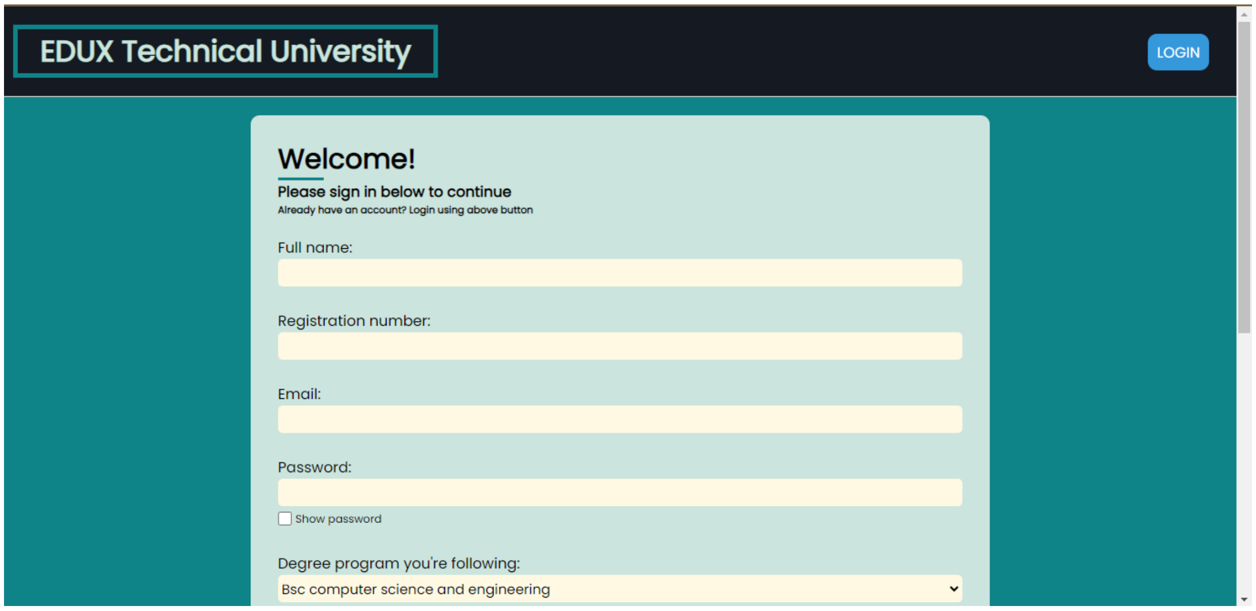
- **Full name:** The full name of the student
- **Registration number:** This is given by the ETU for every student when they register for the degree program. This number must be a 4-digit value. The input field is validated so the user can only enter values ranging from 1000 – 9999.
- **Email:** Email of the user.
- **Password:** The password for the account. The user can check and confirm the password before submitting by **clicking the show password checkbox**.
- **Degree Program:** This is a dropdown menu which has 4 options available to select. (BSc computer science and engineering, BSc mechanical engineering, BSc civil engineering, BSc electrical engineering). The computer science option will be selected as default, but the user can change according to the program they follow.
- **Contact:** The telephone number of the student. Assuming this is a mobile No the input field is validated such the number must start with **07** and have 10 digits in total. Ex: 0719876542
- **Introduction:** The students can give a brief about themselves to the hiring people using this field.
- **Current GPA:** The GPA of the student.
- **Skills:** The students can list their skills to show their proficient areas.

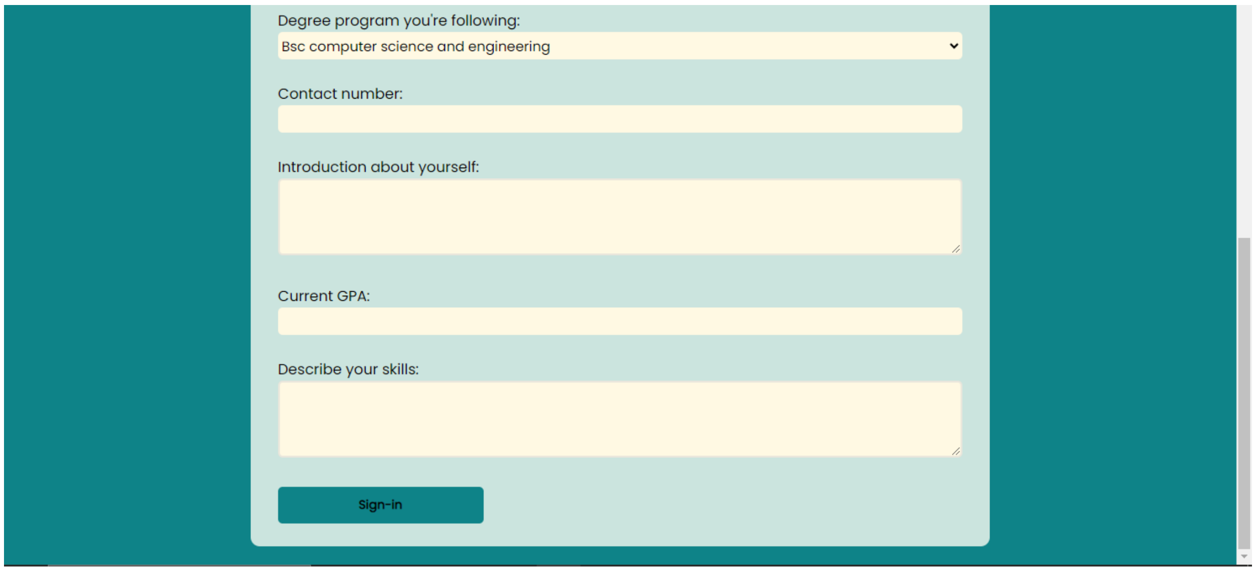
All the fields must be filled out compulsorily and the user can click the signup to register.

Implementation

- Firstly, created a form to enter and register students, named that file as 'sign-in.html' and saved it in the templates folder. Relevant stylings were also done.
- A function to add items to the user tables was created with user parameters and named that function as "add_items_to_user_table".
- Rendered this file in the index route.
- When the user enters the details and clicks sign-in button it'll route to "/signup" and function to add items is called here to save relevant details to the table. This entire process is done using the POST method.
- Then this function will render the login page at the end.

<u>Sign-in page</u>





EDUX Technical University

LOGIN

Welcome!

Please sign in below to continue

Already have an account? Login using above button

Full name:

Registration number:

Email:

Password:

Show password

Degree program you're following: Bsc computer science and engineering

Contact number:

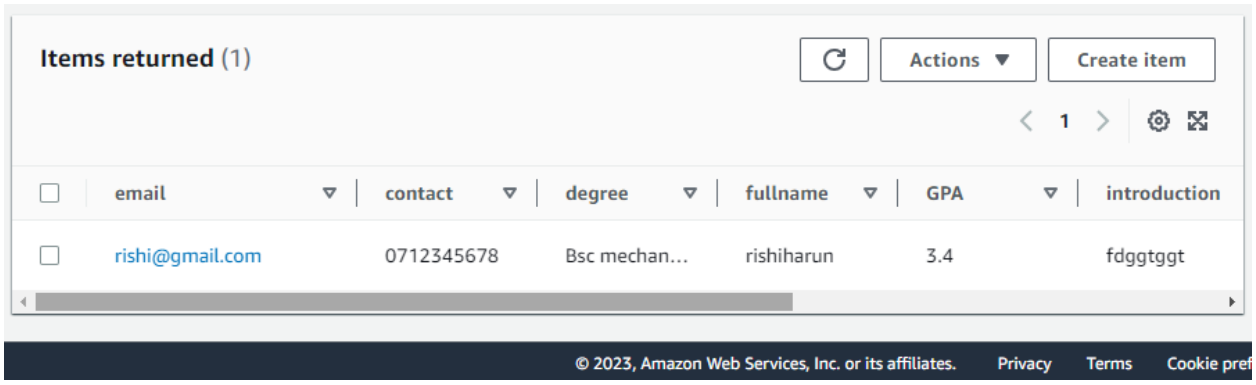
Introduction about yourself:

Current GPA:

Describe your skills:

Sign-in

<u>After submitting the form, the data will be added to the users table.</u>



Items returned (1)

Actions ▼ Create item

	email	contact	degree	fullname	GPA	introduction
☐	rishi@gmail.com	0712345678	Bsc mechan…	rishiharun	3.4	fdggtggt

© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie pref

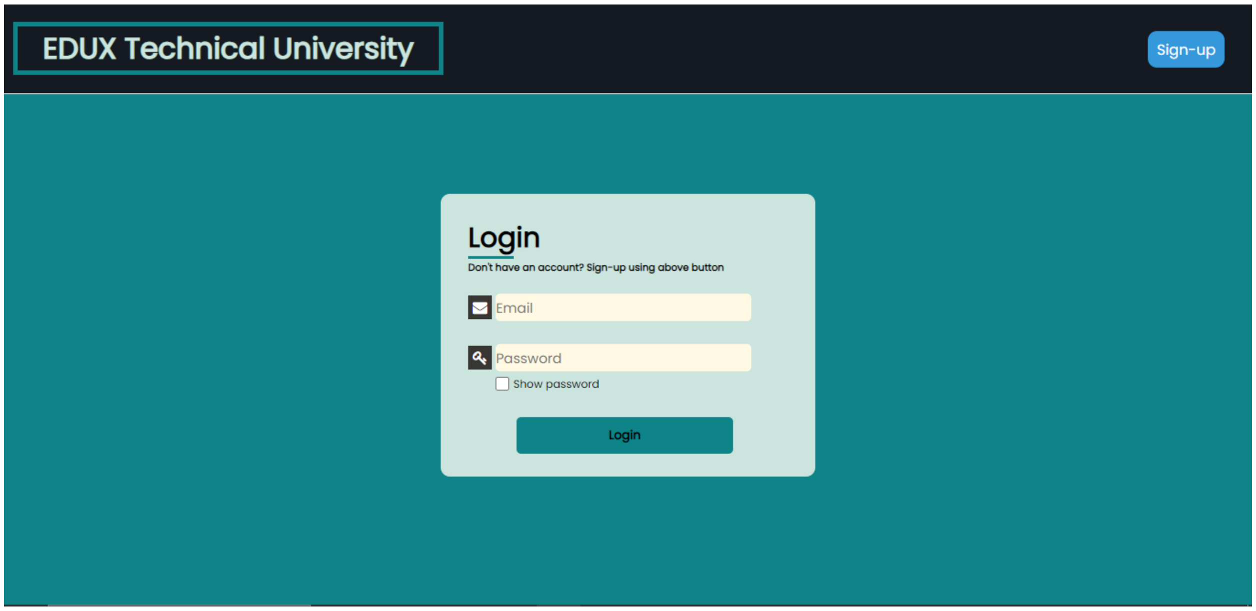
4) Student login

User will land on this page after signing in or when clicking already have account link. In the login page the user will have to enter the email address and password which they provided when signing in. If the credentials are correct, they will be allowed to view/edit their profile.

Implementation

- The login page was created by using HTML and CSS. The page was named "login.html".
- In the backend when the user enters the credentials and click login button it'll be routed to "/check".
- There the credentials will be validated. If the credentials are correct all the user details are loaded and returned along with the profile edit page.

Login page


EDUX Technical University
Sign-up
Login
Don't have an account? Sign-up using above button
Email
Password
Show password
Login

5) a) User profile view/edit

- If the login is successful, the user will be directed to this page where they can view their details and if they wish to change anything it can be done here. Users can edit anything but not the email address.

Implementation

- Created a HTML file called “update.html” and added necessary style to it.
- The user details returned by the “/check” route was used to display relevant details in the “update.html”.
- If the user wants to edit the profile, after entering relevant changes and clicking update button it’ll be routed to “/update” to perform necessary action.
- At the front end the java script code will prevent the default submission and control the actions of the form.
- The user details’ values will be stored in variables and all of them will be stored in a variable called “formdata” as objects.
- Then the “formdata” will be sent to “/update” route using the fetch API as PUT request.
- In the dynamodb_handler file a function to update table item was written and named as “update_user_profile.”
- In the app.py this function is used to update user details which were received from fetch API and send it to the database using PUT method.
- The user will get an alert saying updated successfully after request is sent.

5) b) Profile picture upload

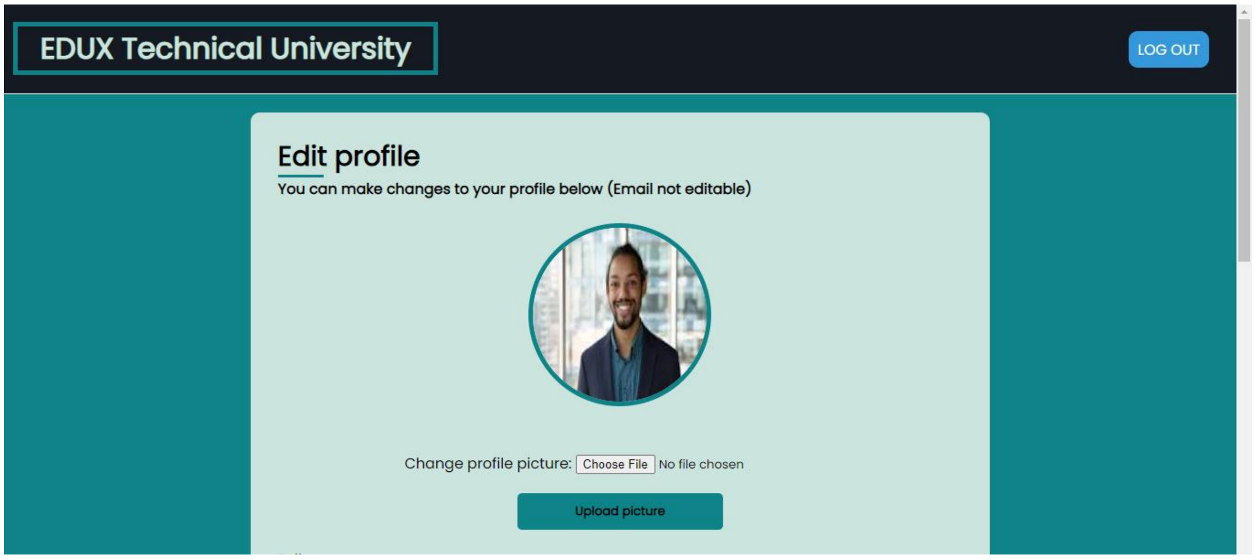
- This is the main feature of phase 2. Previously in the edit profile the users could only edit their details. But now they can upload a picture of themselves.

Implementation

- Created a bucket called profile-image-upload.
- When the user registers, they will be assigned a default picture which they won’t notice. This is because when they login rather the image being empty the default picture will be loaded. So, this default picture was uploaded in the bucket after creating it.

- Adding necessary code to display the picture and allowing the user to change it in the "update.html."
- When the user chooses the picture and click "Upload picture" it'll route to "/upload".
- There the picture is uploaded to the bucket and the object URL is created and stored in the table.
- Then the page will return the new profile picture.

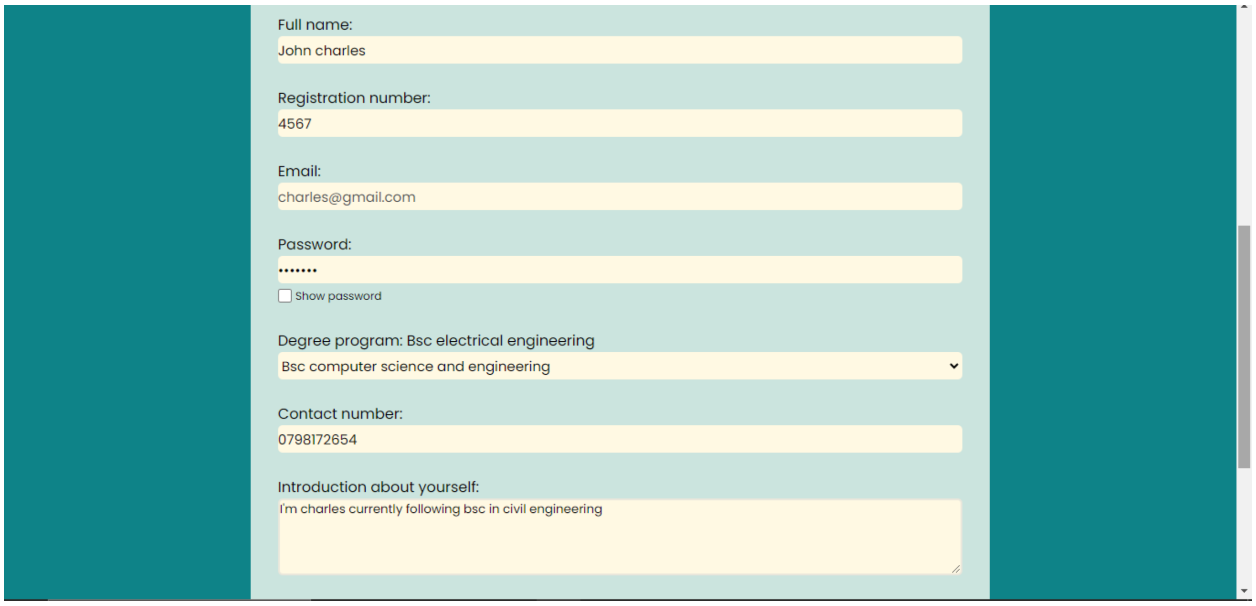
<u>View/edit page.</u>



Edit profile

You can make changes to your profile below (Email not editable)

Change profile picture: Choose File No file chosen


Full name:
John charles

Registration number:
4567

Email:
charles@gmail.com

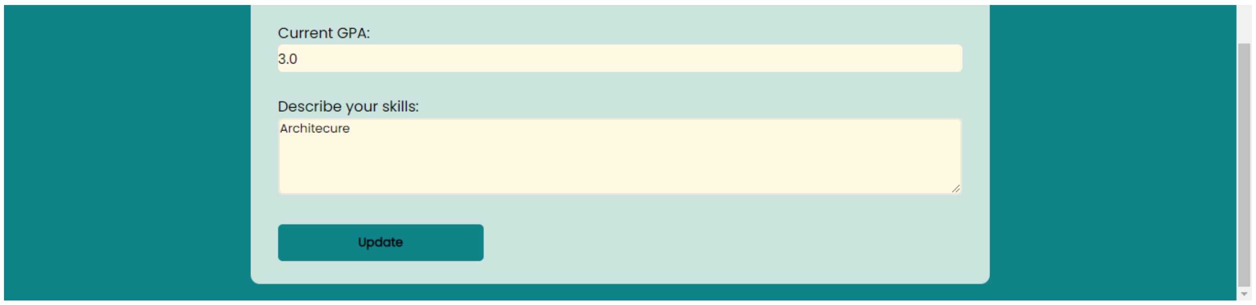
Password:
•••••••

☐ Show password

Degree program: Bsc electrical engineering
Bsc computer science and engineering

Contact number:
0798172654

Introduction about yourself:
I'm charles currently following bsc in civil engineering



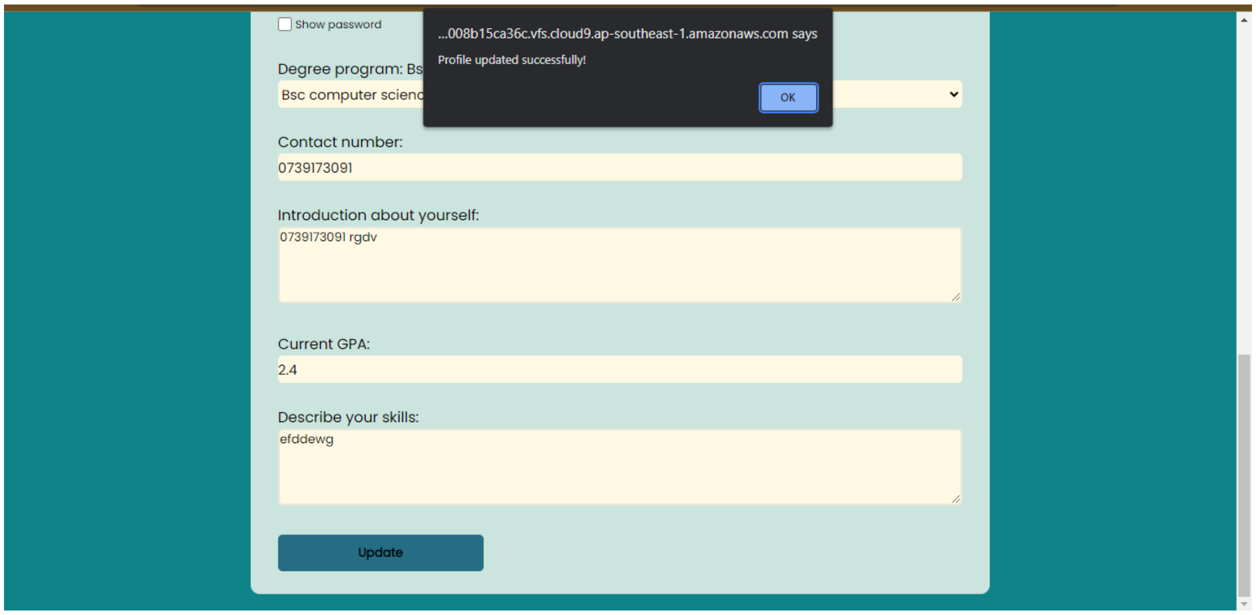
Current GPA:

3.0

Describe your skills:

Architecure

After clicking the update button.



...008b15ca36c.vfs.cloud9.ap-southeast-1.amazonaws.com says

Profile updated successfully!

Degree program: Bs

Bsc computer scienc

Contact number:

0739173091

Introduction about yourself:

0739173091 rgdv

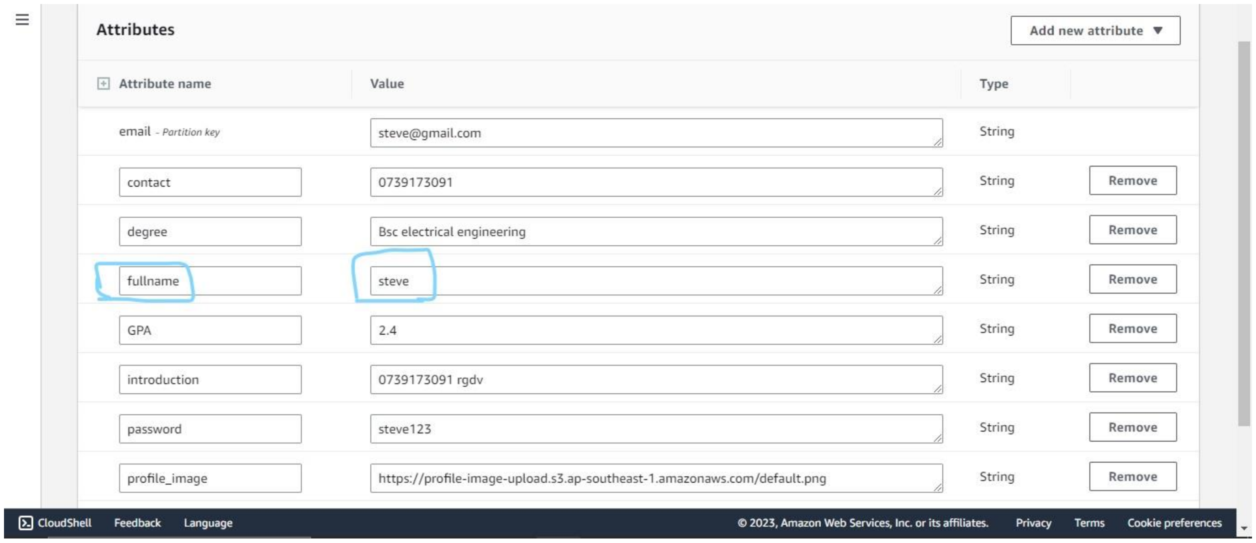
Current GPA:

2.4

Describe your skills:

efddewg

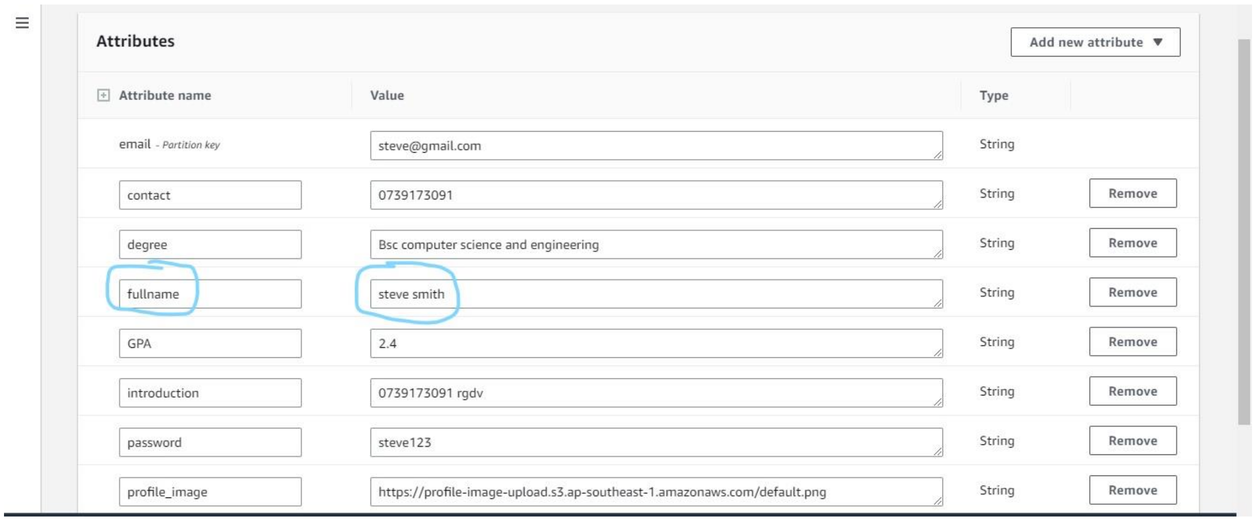
Item before update



Attributes

Attribute name	Value	Type	
email - *Partition key*	steve@gmail.com	String	
contact	0739173091	String	Remove
degree	Bsc electrical engineering	String	Remove
fullname	steve	String	Remove
GPA	2.4	String	Remove
introduction	0739173091 rgdv	String	Remove
password	steve123	String	Remove
profile_image	https://profile-image-upload.s3.ap-southeast-1.amazonaws.com/default.png	String	Remove

<u>After update</u>



Attribute name	Value	Type
email - *Partition key*	steve@gmail.com	String
contact	0739173091	String
degree	Bsc computer science and engineering	String
fullname	steve smith	String
GPA	2.4	String
introduction	0739173091 rgdv	String
password	steve123	String
profile_image	https://profile-image-upload.s3.ap-southeast-1.amazonaws.com/default.png	String

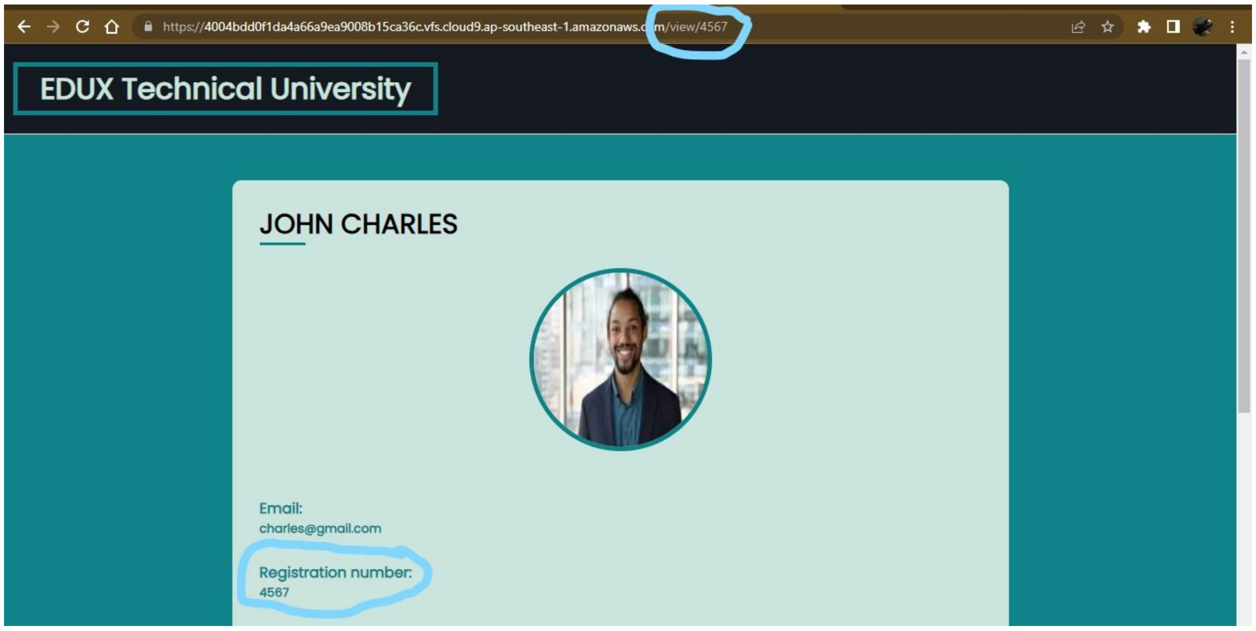
<u>6) Profile public view</u>

The profile of the student can be viewed by using the registration number when requested in the URL. When "/view/ (registration number)" is typed in URL a view of the profile is returned if a profile exists in requested registration number.

<u>Implementation</u>

- Created an HTML file for the public view and added relevant styles to it.
- Created a function called get_user_profile in the dynamodb_handler file.
- In the app.py it'll call the above function and get user information using the GET method and returning the view.html file along with the user details.

Public view page



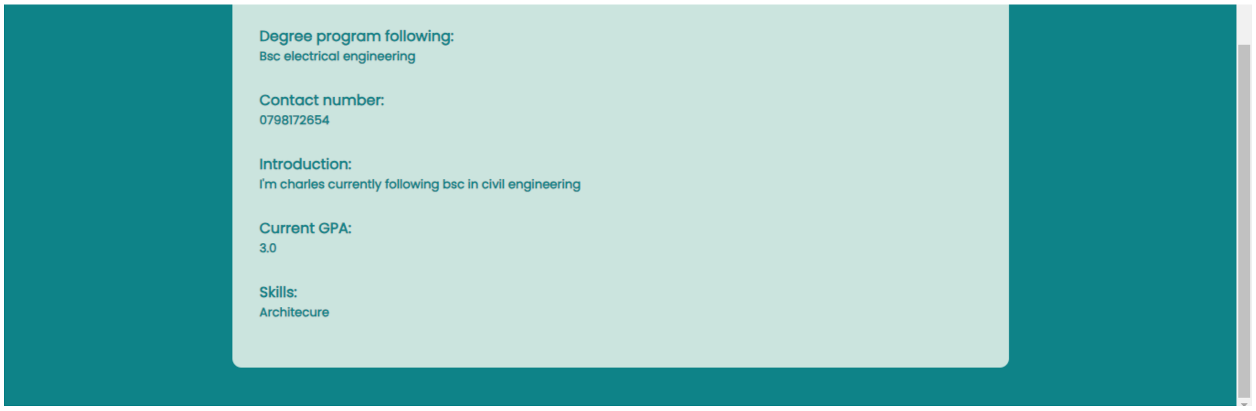
https://4004bdd0f1da4a66a9ea9008b15ca36c.vfs.cloud9.ap-southeast-1.amazonaws.com/view/4567

EDUX Technical University

JOHN CHARLES

Email:
charles@gmail.com

Registration number:
4567



Degree program following:
Bsc electrical engineering

Contact number:
0798172654

Introduction:
I'm charles currently following bsc in civil engineering

Current GPA:
3.0

Skills:
Architecure

7) Deployment of the app

- Created an EC2 instance called flask-server with Ubuntu server.
- Connected that instance and downloaded necessary resources. (Python, flask, boto3, Apache)
- Created a directory to import the source code and linked that directory to "/var/www/html".
- Imported the source code from GitHub using the git clone method.
- Created the flaskapp.wsgi file added relevant implementation to it.
- Added a few code lines to Apache configuration file.
- Finally restarted the server.

Updates on phase 2 than previous phase

- User image upload
- Change in the user interface. (Header, change in interface colors.)
- Notification messages on login page. (When registration is complete a success message will be displayed, also if username or password is incorrect it will also be notified to the user using jinja templates.)