PROJECT REPORT
On
Faculty Recruitment Application Portal

Submitted to

COSC

Done By

TEAM - 12

Deevana Mridhula	160119733068	(CSE-1/4)
Namya Reddy	160117733011	(CSE-3/4)
Nikhil Ranga	160118737031	(IT-2/4)
Slesha Adi	160118733138	(CSE-2/4)
Srihitha Reddy	160119733144	(CSE-1/4)
Tarun Kumar	160118737051	(IT-2/4)
Tejaswini Jakka	160118733085	(CSE-2/4)
Velagandula Nanditha	160119733133	(CSE-1/4)

Mentored By

Sri Sai - 160117733049

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)

(Affiliated to Osmania University; Accredited by NBA(AICTE) and NAAC(UGC), ISO Certified 9001:2015)

GANDIPET, HYDERABAD – 500 075

Website: www.cbit.ac.in

June-2020

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)

(Affiliated to Osmania University)

GANDIPET, HYDERABAD – 500 075

CERTIFICATE

This is to certify that the project work entitled "**Faculty Recruitment Application Portal**" submitted to **CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY OPEN SOURCE COMMUNITY,** in partial fulfilment of the requirements for the award of the completion of Summer Internship programme June, 2020, is a record of original work done by **TEAM - 12** in June, 2020, under our supervision and guidance.

Project Guide **Head of the Department**

ACKNOWLEDGEMENTS

We are overwhelmed and extremely grateful to have got this wonderful opportunity to work on this phenomenal project through which we have garnered plentiful knowledge and experience.

We would like to express our deep and sincere gratitude to our project mentor, Mr. SriSai, as well as the various members of the COSC club and the industry partners from Caravel for providing us with the opportunity to work on this project and for their invaluable guidance throughout this project.

Our sincere thanks to the Principal, Dr.P.Ravinder Reddy, as well as the management for giving us a good opportunity to improve our skills with a good learning atmosphere.

We are extremely grateful to our parents for their love, prayers, care and sacrifices for educating and preparing us for our future and for supporting us in the completion of this project.

We would like to thank our friends for their constant help and guidance and for their support in completion of the project.

ABSTRACT

This project report is to provide an extensive insight into the technical and real-time work aspects of the project- Faculty Recruitment Portal. This project focuses on creating an application system for convenient and simple recruitment of faculty for organizations. The process of physical applications and interview rounds is quickly becoming outdated in this age of technology due to the amount of overhead work, resources wasted, inefficiency and inconsistency caused. The Faculty Recruitment Portal is hence, a very appropriate solution.

The Admin side deals with the tasks like uploading vacant roles, viewing the applications and selecting applicants for roles were to be performed. These features were to be provided on a web application. The web application was built using Django. Python is used for providing functionality and ease; HTML, CSS, JavaScript and Bootstrap were used for template creation and design of the web front.

The User side deals with User registration along with their educational qualifications, viewing of all the vacant roles, applying for the roles they choose and checking the status of their applications. These requirements were fulfilled using Android Studio framework as the base for the application development. XML and Java are the programming languages used for developing the activities. XML is Extensible Markup Language, used for creating the design, theme and layout of the activities. Java is used for providing all the functionality to these activities

The two parts of the project are tied together to create one functional system by creating a common database and implementing the concept of RESTFUL APIs. The global database was created on MySQL. Relevant tables were created to ensure the easy connectivity and elimination of redundancy on both sides of the system. The APIs were created using Flask which administers Python as it makes the code simple and easy to debug. The APIs were then deployed using Heroku and they were used accordingly throughout the project wherever the need to interact with the database occurred.

In this project, the gist of the internal workings of the Faculty Recruitment System is stated. Emphasis and detailed description of all the tasks performed and services provided by the project are highlighted. All the techniques/tools used, softwares utilized, methodology applied and functions undertaken and finally, the overall test results of the project are also outlined.

TABLE OF CONTENTS

INTRODUCTION

1.1 Purpose/Objective

This project report is to provide an extensive insight into the technical and real-time work aspects of the project- Faculty Recruitment Portal.

The Objective of this project is to ensure that the use of technology enhances the effective operations of the College Management for faculty recruitment.

Objective 1 - Provide simple, universal access to information and services for all potential faculty and staff.

Objective 2 - Improve the online communications ability for individuals who are interested to apply for faculty positions associated with the College Management.

Objective 3 - Provide advanced web tools for the College Management to enhance its web presence in a way that will reach and attract more potential faculty members and meet the college management community's changing needs.

1.2 Problem Statement

Admin

- Admin Login
- Upload all the roles with vacancies in the college
- Receive all applications from users
- Go through the User’s qualifications and decide to approve or decline
- If approve recruit for only any one post from all his/her applications
- Once a faculty is recruited all the applications with respect to him should be discarded

User

- Registration with uploading all the details about Education qualifications and experience
- Login
- Home page with list of roles that he can apply and status of previous applications
- Limit number of applications to 3
- Apply for a role in college with department details and preferences of subjects
- Receive notifications about the updates regarding applications
- User cannot apply for the same role after getting declined.

1.3 Work Division

A team of eight members was assigned for the Faculty Recruitment Application Portal project. The work was split into three areas: website(Admin side), mobile application(User side),database and API. Three members were assigned for building the website, three members were assigned for building the mobile app and two members were assigned to build all the required APIs which are required to access the database.

The website team had to build a website in such a way that admin can upload all the roles with vacancies in the college, they can also retrieve all the applications from the users and also, they can recruit the users based on their qualifications, research details etc. The mobile application team had to build an app in such a way that user can view all the vacant roles details uploaded by the admin and they can apply for a role and can also retrieve notifications about the updates regarding the application. The API team had to build the APIs for both admin and user in such a way that they can retrieve and upload details to the database whenever required.

Web app Team	:	Velagandula Nanditha-CSE-1/4
		Tarun Kumar-IT-2/4
		Deevana Mridhula-CSE-1/4
Mobile app Team	:	Slesha Adi-CSE-2/4
		Nikhil Ranga-IT-2/4
		Namya Reddy-CSE-3/4
Api & Database Team	:	Tejaswini Jakka-CSE-2/4
		Srihitha Reddy-CSE-1/4

TECHNOLOGY STACK

1.Front-end Design(Web Application): HTML,CSS,Bootstrap,Javascript.

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web.Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML.CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Bootstrap is a free and open-source front-end library for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many web frameworks, it concerns itself with front-end development only.

Back-end(Web Application): Django

Django is a free and open source web application framework written in Python. Django has its own naming system for all functions and components (e.g., HTTP responses are called “views”). The Django framework uses the principles of rapid development, which means developers can do more than one iteration at a time without starting the whole schedule from scratch.With Django, you can tackle projects of any size and capacity, whether it’s a simple website or a high-load web application.

2.Android Application: Java,XML,Android Studio

Java is a general-purpose programming language that is class-based, object-oriented, and designed to have as few implementation dependencies as possible. It is intended to let application developers write once, run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Android relies heavily on these Java fundamentals. The Android SDK includes many standard Java libraries (data structure libraries, math libraries, graphics libraries, networking libraries and everything else you could want) as well as special Android libraries that will help you develop awesome Android applications.

XML stands for Extensible Mark-up Language. XML is a very popular format and commonly used for sharing data. An XML-based layout is a file that defines the different widgets to be used in the UI and the relations between those widgets and their containers. Android treats the layout files as resources. The layout files act as an input to the Android Asset Packaging Tool (AAPT) tool, which creates an R.java file for all the resources.

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA . On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system.
- A fast and feature-rich emulator.
- A unified environment where you can develop for all Android devices.
- Apply Changes to push code and resource changes to your running app without restarting your app.
- Code templates and GitHub integration to help you build common app features and import sample code.
- Extensive testing tools and frameworks.
- Lint tools to catch performance, usability, version compatibility, and other problems.
- C++ and NDK support.
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine.

Android SDK:

A software development kit that enables developers to create applications for the Android platform. The Android SDK includes sample projects with source code, development tools, an emulator, and required libraries to build Android applications.

Gradle:

Gradle is a general-purpose build tool. It makes it easy to build common types of project say Java libraries, by adding a layer of conventions and prebuilt functionality through plugins. You can even create and publish custom plugins to encapsulate your own conventions and build functionality.

Project structure:

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

- Android app modules
- Library modules
- Google App Engine modules

Each app module contains the following folders:

- manifests: Contains the AndroidManifest.xml file.
- java: Contains the Java source code files, including JUnit test code.
- res: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

Layouts:

A layout defines the structure for a user interface in your app, such as in an activity. All elements in the layout are built using a hierarchy of View and ViewGroup objects. A View usually draws something the user can see and interact with. Whereas a ViewGroup is an invisible container that defines the layout structure for View and other ViewGroup objects.

The different types of layout we used in the project are:

- Constraint Layout: ConstraintLayout, which is now the default layout in Android Studio, gives you many ways to place objects. You can constrain them to their container, to each other or to guidelines. This allows you to create large, complex, dynamic and responsive views in a flat hierarchy. It even supports animations!
- Linear Layout: LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally.
- List View: ListView is a view group that displays a list of scrollable items.

Intent:

An intent is to perform an action on the screen. It is mostly used to start an activity, send a broadcast receiver, start services, and send messages between two activities. There are two intents available in android as Implicit Intents and Explicit Intents. Here is a sample example to start a new activity with old activity.

Toasts Overview:

A toast provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message and the current activity remains visible and interactive. Toasts automatically disappear after a timeout.

3.Database: MySQL

MySQL is an Oracle-backed open source relational database management system (RDBMS) based on Structured Query Language. It is based on the structure query language (SQL), which is used for adding, removing, and modifying information in the database. Standard SQL commands, such as ADD, DROP, INSERT, and UPDATE can be used with MySQL.

4.RESTful API: Flask

Flask is a web application framework written in Python. Flask is based on WSGI(Web Server Gateway Interface) toolkit and Jinja2 template engine. WSGI is basically a protocol defined so that Python applications can communicate with a web-server and thus be used as web-application outside of CGI.

SYSTEM REQUIREMENT SPECIFICATION

Software requirements:

- Any Web Browser
- Python 3 with Django Module
- 64 bit operating system.
- Emulator
- Android studio
- pip
- virtualenv
- Virtualenvwrapper
- Flask module

Hardware requirements:

- Modern Operating System:
    - Windows 7 or 10
    - Mac OS X 10.11 or higher, 64-bit
    - Linux: RHEL 6/7, 64-bit (almost all libraries also work in Ubuntu)
- x86 64-bit CPU (Intel / AMD architecture)
- Minimum 4 GB RAM
- Minimum 5 GB free disk space
- A 64 bit environment is required for Android 2.3xGingerbread and higher version.
- Atleast 250GB of free disk space to check out code and an extra 150GB to build it.
- Atleast 8GB of RAM/swap

FEATURES OF THE USER

5.1 Requirements for the Mobile Application:

The user has to register for recruitment by providing necessary personal details and educational qualifications that the Admin can consider while recruiting. After successfully registering, the user will be inserted into the database. The user can now login using the valid Email and password, this data is checked with the database contents- when successful, the user will be directed to the homepage; when the login fails, user is displayed an error message and is expected to retry.

In the homepage, the user can choose if they want to view the list of vacant roles or the status of applications. When the user chooses to view the list of roles, they can view those roles and choose if they want to apply for a role. The user can now apply for the role filling in additional details – department details and preference of subjects relevant to the role. When the user applies for the role with the aforementioned details, the database is checked for two things, if the user was declined for the role previously or if the user has already applied more than thrice; in either case: the user will be sent an error message and the application details won’t be inserted into the database.

When the user chooses to view status of application, the data will be queried using the EmailID of the user. Whenever the Admin changes the status of the application sent, this change would be checked and compared with the previous status and if they aren’t the same, the user will receive a notification.All these factors were to be taken into consideration while creating the app and the rest of the app is to be built around these points. These conditions were to be satisfied to declare the successful completion of this project.

5.2 Workflow of the app:

The Faculty Recruitment Portal App consists of the following pages:

- Main Page.
- Registration Page.
- Login Page.
- Home Page.
- Role List Page.
- Application details Page.
- Application Status Page.

As the Mobile application is for the users, when a user opens the app he/she can see a logo of our application, at the bottom there are two buttons labeled as Register and Login. When the user clicks on the Register button, he/she will be redirected to the Registration page in which the user has to fill his/her basic details which are stored in the database. At the bottom of the Registration Page there are two buttons namely Register and Login Now. After filling the details, the user has to click on the Register button to complete his/her Registration Process. If all the credentials are filled, then a toast message will be displayed saying Registered Successfully.

After registering the user can click on the Login Now button in Registration Page or the Login button in the Main Page to log in to his/her account. When the user clicks on the Login Button, he/she will be redirected to the Login Page where the user must enter his/her Email ID and Password which he/she provided during the Registration process. When the user enters the correct Email ID and Password the user will be redirected to Home Page.

In the Home Page the user can see the two TextView's namely List of Roles and Status of the Applications. When the user clicks on the List of Applications, he/she will be redirected to the role List Page, where the user can see the list of roles available. At the bottom there will be a button named as Apply. When the user clicks on the Apply button, he/she will be redirected to the Application Page.

In the Application Page the user must enter the role ID which he/she wants to apply from the list of vacant roles and enter the preference and research of the subject details and click on the submit button. The details will be sent to the database if the role he/she applied is not declined already and the number of applications he/she applied is less than three.

After applying for the roles, the user can check the status of his/her applications by clicking on the Status of applications TextView in the Home Page. When the user clicks on that TextView he/she will be redirected to the status page in which the user can see the status of the applications he/she applied. The user can use Logout Button in the Home Page to log out of his/her account.

5.3 Class and API Integration Description:

MainActivity.java

This is the Android launcher class for Faculty Recruitment Application Portal. This class creates the first screen of the application which contains the logo of our app and also the user can either select register or login option.

Registration.java

This class creates the screen where user can enter his details along with his qualification and experience details.Here we send all the required parameters and the url with the help of POST method. POST requests supply additional data from the user to the server in the message body. A RESTful API is an application program interface that uses HTTP requests to GET, PUT, POST and DELETE data. So with the help of these user details will be uploaded to the database.Here we need to also verify that whether the emailid entered by the user already exists in the database or not.

Login.java

This class uses EditText to get Email ID and a password protection EditText to get user password. The password protection EditText hides the text while inputting. If the credentials are valid then the user will be redirected to the homepage otherwise he will get an error message. We will send the parameters with the help of the POST method. If the credentials are valid we will receive an access token which should be saved in a string variable so that it can be sent as header whenever we need access to the database after login.

Homepage.java

This class consists of two text views and a logout button. By using those text views we can either view the vacant roles uploaded by the admin in the vacant roles page or we can view the status of the previous applications in the application status page. This class also gives us a notification if there is any update in the status of any application applied by the user.

For the notification part first we use the GET method to retrieve the present status and previous status of all the applications applied by the user ,then we compare both of them,if they are different then a alert box will notify the user of the status update. Then we use POST method to send the application id of the application whose status has been updated so that the previous status will be updated to the present one.

RoleList.java

This class uses GET method to retrieve all the roles uploaded by the admin in the database. This includes role id, role name ,department name and the required qualification. The user needs to select the role which he wants to apply for. Then he will be redirected to the application details page. Here we used recycler view and card view to display the details retrieved from the json response object.

ApplicationDetails.java

This class uses POST method to upload the application details to the database. It includes emailid of the user, role id of the role the user wants to apply for, preference subjects and also his research details. Here the api used will also verify two conditions which are the requirements mentioned in the problem statement like we need to limit the number of applications applied by the user to three and also the user cannot apply for a role for which he was previously declined for. If those conditions are satisfied then the application details will be uploaded to the database otherwise we will get error messages stating the cause of it .

ApplicationStatus.java

This class uses GET method to retrieve the status uploaded by the admin in the database for all the applications applied by the user. It also includes role id, application id, department name,position and status of the application.

FEATURES OF THE API

6.1 Requirements for the API:

If the corresponding API uses a post method, then the administrator/user should insert the values, so that the API can post the details into the database.
If the corresponding API uses a get method, then on receiving the necessary details, the API can get the administrator/user the corresponding and necessary details.

6.2 Description of the files used:

The files that are used are :

1. **app.py**

   In this file, we import all the classes which were created as APIs in different files and create endpoints. We also take care if there is an invalid token error or the missing token error.

2. **db.py**

   In this file, we make a connection with the database tables we have created in the MySQL Workbench.
   For reference:
   Host : skillup-team-12.cxgok3weok8n.ap-south-1.rds.amazonaws.com
   Port : 3306
   User : admin
   Password : coscskillup
   We also convert the decimal data to the string data as json type is not compatible with the decimal datatype.

3. **__init__.py**

   We write nothing in this file. We just name a file this way in a folder, to make it a python package.

4. **admin.py**

   All the APIs used by the admin side are created as classes in this particular file. Then these classes are imported to app.py file so as to make the endpoints for the admin side.

5. **user.py**

   All the APIs used by the Mobile application/user side are created as classes in this particular file. Then these classes are imported to app.py file so as to make the endpoints for the user side.

6.3 APIs for Administrator and User:

There are 15 APIs in total for both the Admin side and the user side.
Here are the functionalities of some of the APIs:
ADMIN SIDE

1. There is a post API where the administrator can post the faculty position vacancies available in the college, which posts into the database.
2. An API which gets the presently available vacancies from the database, so that the admin can view what all vacancies are present.
3. A ‘get’ API through which he can see all the application details filled by the users.
4. A ‘post’ API , which would allow him to post the status of the application into the database.
5. A ‘post’ API which would allow him to post the details of the recruited members into the database.
6. A ‘get’ API which gets the details of the newly recruited faculty from the database.

USER SIDE:

1. There is a post API for the user registration, which posts all the registration details of the user into the database.
2. A post API for user login which takes the Email Id and Password of the user and compares them with the data in database tables, and if it is correctly matched generates a JWT access token and then returns it.
3. A get API for the user to see all the roles present in the database, he can apply for.
4. A post API which posts the application details of the user into the database.
5. A get API which shows the Applications status of that corresponding EmailId of the user.

There are other APIs as well, and their detailed description will be given in the SCREENSHOTS AND TEST RESULTS OF THE APPLICATION part.

6.4 Limiting a User's Application to only 3 :

This part has been taken care in the /appdetails API, through which the user's application forms are posted into the database.
So in the beginning, we take the count of Email Id of a particular user who has filled the application form(s). If the count of his/her Email ID exceeds 3, in that case, we display a message showing that he/she cannot enter the application form more than thrice.

6.5 Not letting user who was declined for a role, apply for the same role again:

This part has been taken care in the /appdetails API, through which the user's application forms are posted into the database.
If a user was initially declined for a role and they apply for that same role again, we check the status of the user and then we display them the message saying he/she cannot apply for that role as he/she was initially declined.

6.6 Deleting all the application details of a recruited user and deleting the Position Role from the Database to which he was recruited:

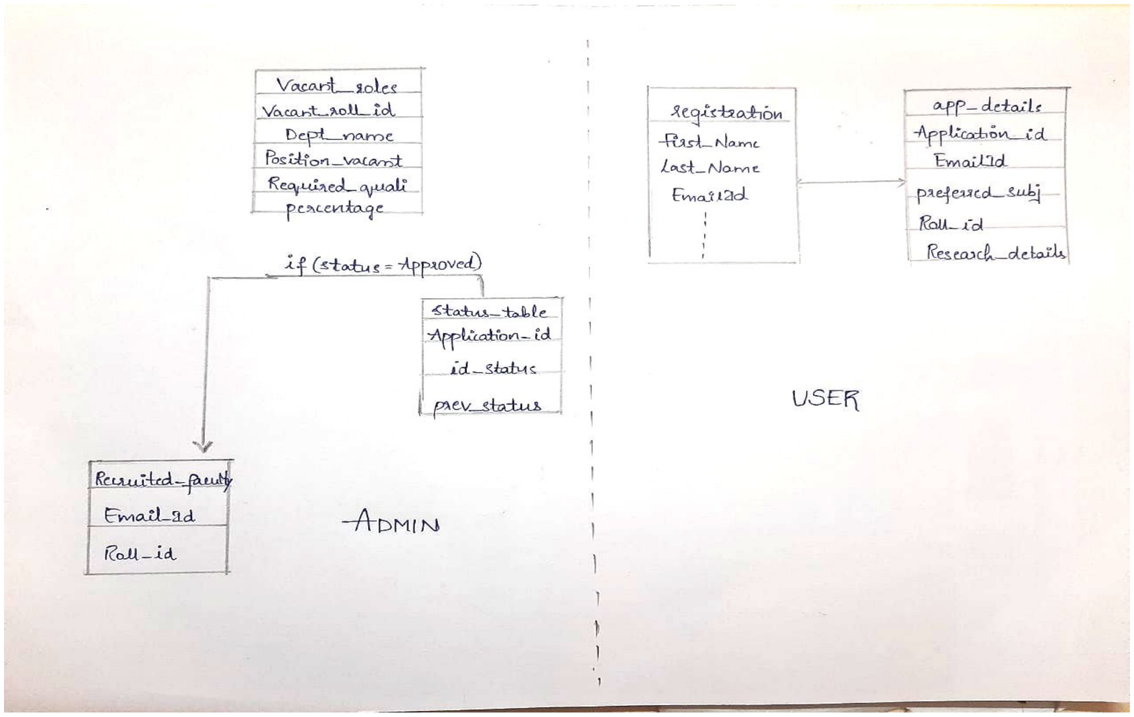
This part has been taken care in the /recruited API, which posts the details of a recruited person into the database.
Once a person is recruited, we delete all his/her application details from the database table using his/her EmailId.
Similarly, we also delete the position to which he/she was recruited from the database, as that position is no more vacant.

DATABASE DESIGN AND TABLES

All of the below tables are created using MySQL and the schema is team12

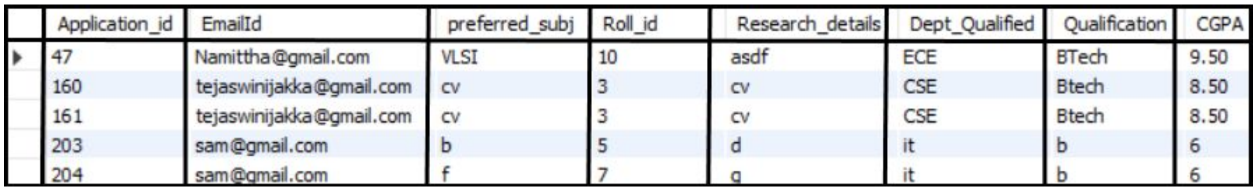
Host: skillup-team-12.cxgok3weok8n.ap-south-1.rds.amazonaws.com

Schema:



Views:

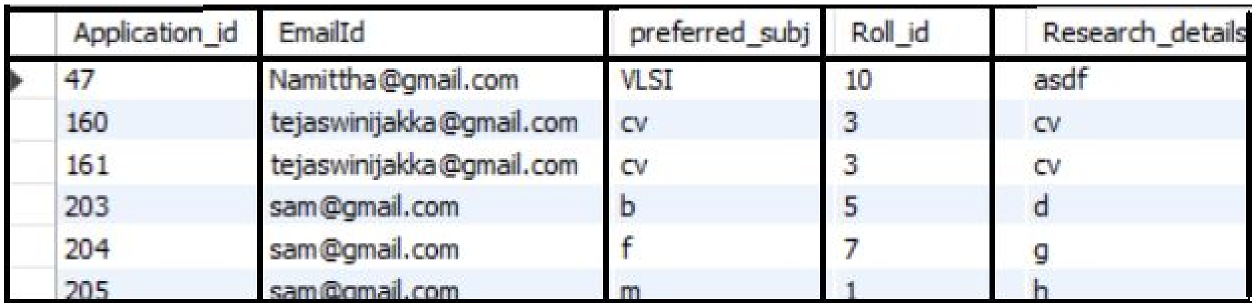
V4:This view we are using to display the status to the user along with all the other fields in v4



	Application_id	EmailId	preferred_subj	Roll_id	Research_details	Dept_Qualified	Qualification	CGPA
▶	47	Namittha@gmail.com	VLSI	10	asdf	ECE	BTech	9.50
	160	tejaswinijakka@gmail.com	cv	3	cv	CSE	Btech	8.50
	161	tejaswinijakka@gmail.com	cv	3	cv	CSE	Btech	8.50
	203	sam@gmail.com	b	5	d	it	b	6
	204	sam@gmail.com	f	7	g	it	b	6

app_details:

```
create table app_details
(
Application_id int Not null AUTO_INCREMENT,
EmailId varchar(50)Not NULL,
preferred_subj varchar(20)NOT null,
Roll_id varchar(10)Not NULL,
Research_details varchar(70) not null,
Primary key(Application_id),
constraint app_1 foreign key(EmailId) references registration(EmailId)
);
```

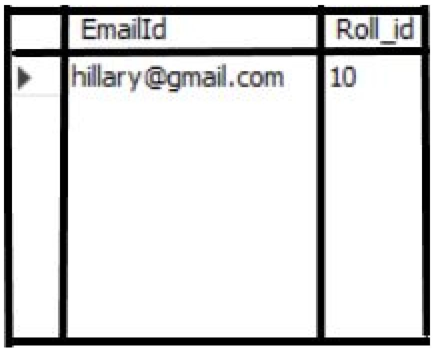


Application_id	EmailId	preferred_subj	Roll_id	Research_details
47	Namittha@gmail.com	VLSI	10	asdf
160	tejaswinijakka@gmail.com	cv	3	cv
161	tejaswinijakka@gmail.com	cv	3	cv
203	sam@gmail.com	b	5	d
204	sam@gmail.com	f	7	g
205	sam@gmail.com	m	1	h

In the table app_details there are Five columns which contains Application-id, EmailId, preferred_subj, Roll_id, Research_details. This table is used to store the details of the applicant so that the admin can see all the Application details of user. Here Application_id is taken as primary key so as to connect with the other Database tables.

recruited_faculty:

```
create table recruited_faculty
(EmailId varchar(50)Not NULL,
Roll_id varchar(10)Not NULL);
```



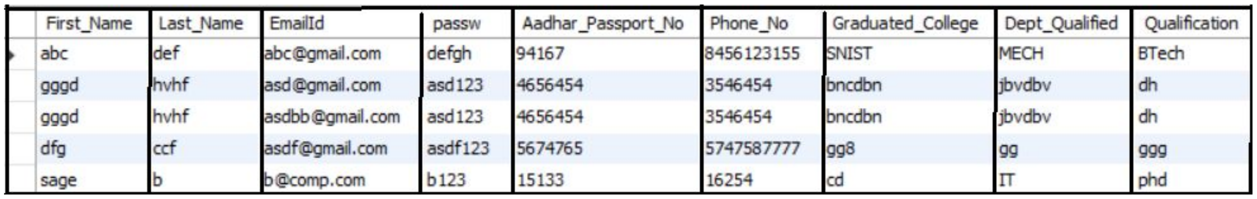
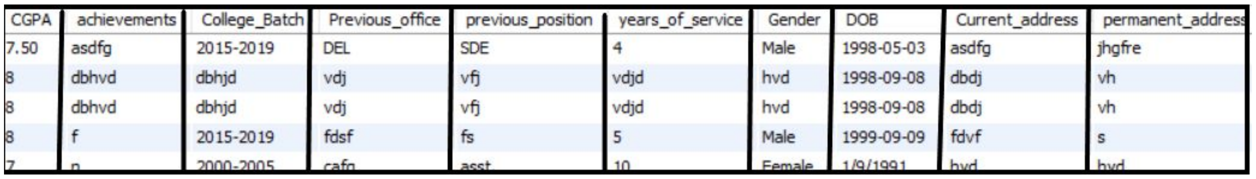
	EmailId	Roll_id
▶	hillary@gmail.com	10

In the table recruted_faculty there are Two columns which are EmailId and Roll_id. This table is used to store the details like Emailid and Rollid of the recruited applicant. Using this table we can see the details of faulty recruited.

Registration:

```
create table registration
(
First_Name varchar(30) Not NULL,
Last_Name varchar(10) Not NULL,
EmailId varchar(50) ,
Passw varchar(15),not null
Aadhar_Passport_No varchar(20) Not NULL,
Phone_No varchar(10) Not Null,
Graduated_College varchar(50) not NULL,
Dept_Qualified varchar(30) not NULL,
Qualification varchar(10) not null,
CGPA varchar(5) not null,
achievements varchar(50) NOT NULL,
College_Batch varchar(15)NOT NULL,
Previous_office varchar(20),
previous_position varchar(20),
```

years_of_service varchar(10),
Gender varchar(10) Not NULL,
DOB varchar(15) not NULL,
Current_address varchar(50) not null,
permanent_address varchar(50),
primary key(EmailId));

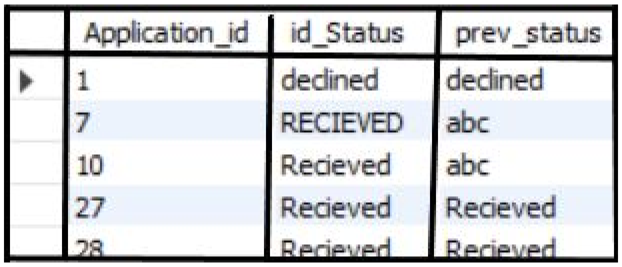



First_Name	Last_Name	EmailId	passw	Aadhar_Passport_No	Phone_No	Graduated_College	Dept_Qualified	Qualification	CGPA	achievements	College_Batch	Previous_office	previous_position	years_of_service	Gender	DOB	Current_address	permanent_address
abc	def	abc@gmail.com	defgh	94167	8456123155	SNIST	MECH	BTech	7.50	asdfg	2015-2019	DEL	SDE	4	Male	1998-05-03	asdfg	jhgfre
gggd	hvhf	asd@gmail.com	asd123	4656454	3546454	bncdbn	jbvdbv	dh	8	dbhvd	dbhjd	vdj	vfj	vdjd	hvd	1998-09-08	dbdj	vh
gggd	hvhf	asdbb@gmail.com	asd123	4656454	3546454	bncdbn	jbvdbv	dh	8	dbhvd	dbhjd	vdj	vfj	vdjd	hvd	1998-09-08	dbdj	vh
dfg	ccf	asdf@gmail.com	asdf123	5674765	5747587777	gg8	gg	ggg	8	f	2015-2019	fdsf	fs	5	Male	1999-09-09	fdvf	s
sage	b	b@comp.com	b123	15133	16254	cd	IT	phd	7	n	2000-2005	cafg	asst	10	Female	1/9/1991	hyd	hyd

In the table registration there are ninteen columns and all of them are required to know the basic details of the applicant as per the requirement. This table stores all the data related to applicant and most of them are given as not null are must to fill and EmailId is taken as primary key.

status_table:

create table status_table
(Application_id int Not null,
id_Status varchar(70) Not Null
prev_status varchar(50) Not Null);

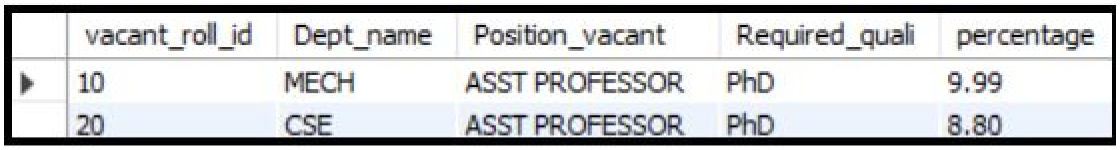


	Application_id	id_Status	prev_status
▶	1	declined	declined
	7	RECIEVED	abc
	10	Recieved	abc
	27	Recieved	Recieved
	28	Recieved	Recieved

The above status_table consists of three columns Application_id, id_status and prev_status. This table is used to store the status or infromation about the applicant whether he/she are selected or declined etc. It also shows the present as well as previous status of the applicant.

vacant_roles:

```
create table vacant_roles
(
vacant_roll_id varchar(10)NOT NULL,
Dept_name varchar(20)NOT NULL,
Position_vacant varchar(30)NOT NULL,
Required_quali varchar(30) not null,
percentage decimal(3,2)NOT NULL,
primary key(vacant_roll_id));
```

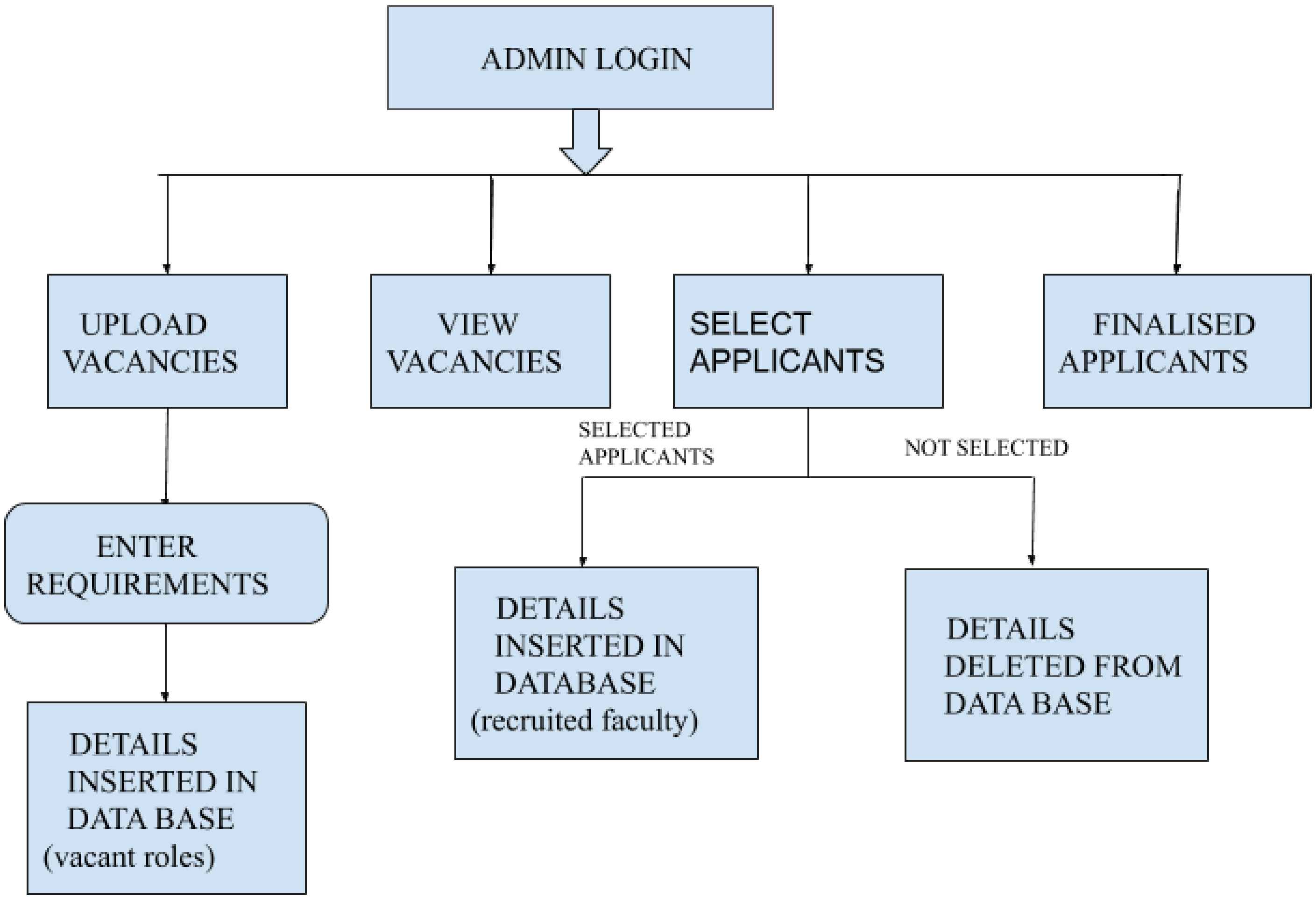


	vacant_roll_id	Dept_name	Position_vacant	Required_quali	percentage
▶	10	MECH	ASST PROFESSOR	PhD	9.99
	20	CSE	ASST PROFESSOR	PhD	8.80

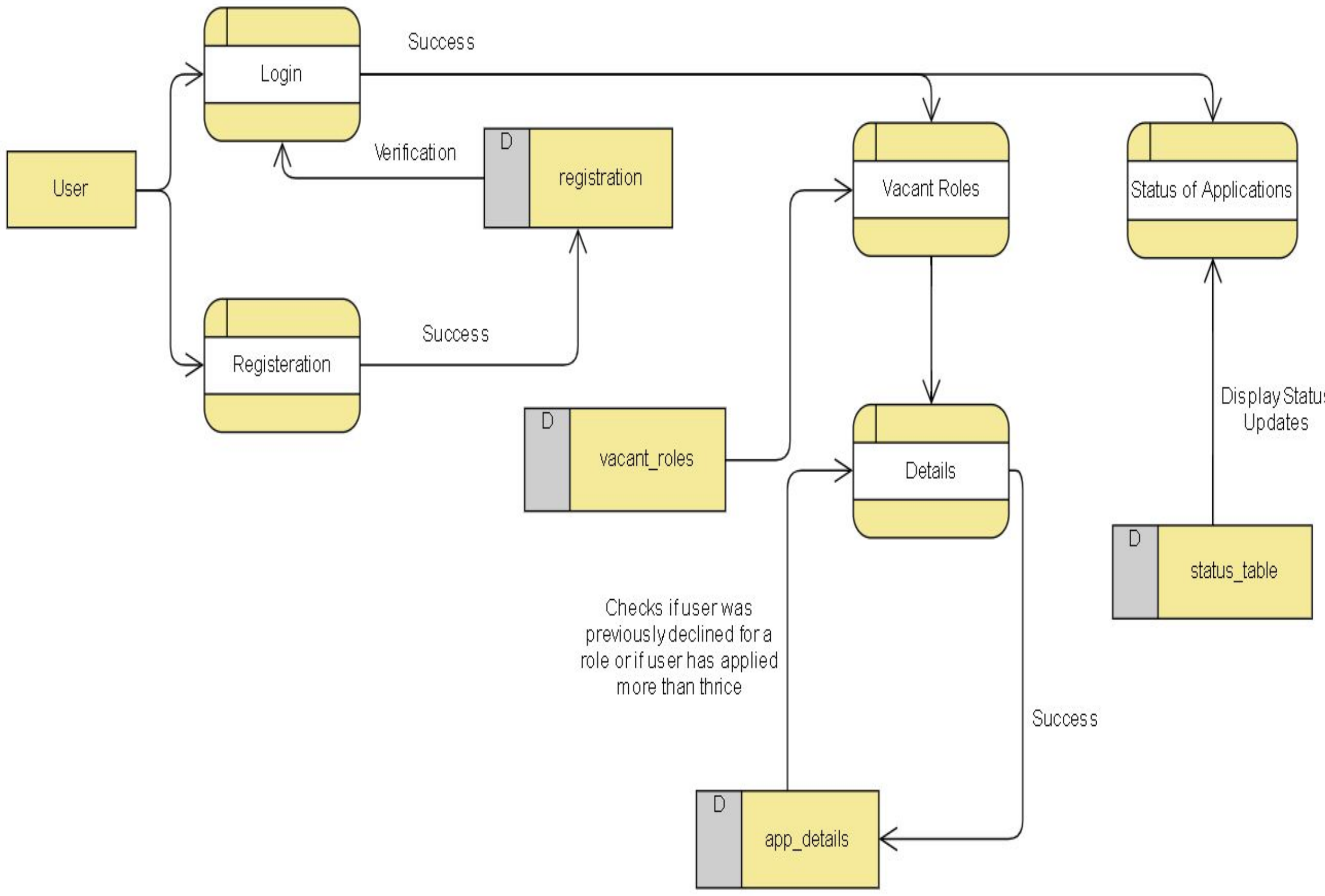
The table vacant_roles contains five columns consisting of vacant_roll_id, Dept_name, Position_vacant, Required_quali, percentage. This table is used to store the details about vacancies and vacant_roll_id is taken as primary key.

UML DIAGRAMS

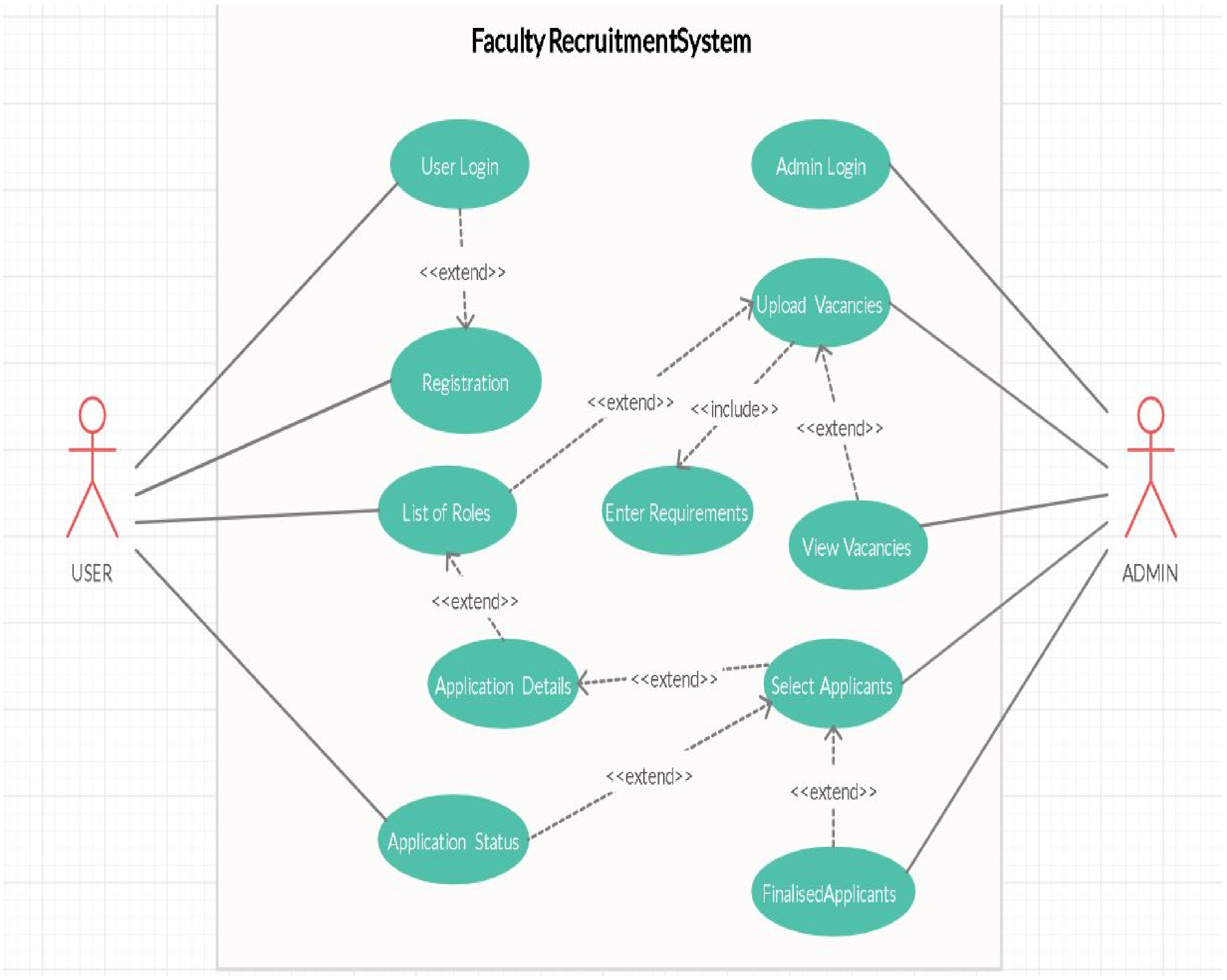
1.Web Application(Flow Design):



2.Android Application(Data Flow Diagram):



3. Use Case Diagram of the project



SCREENSHOTS AND TEST RESULTS OF APPLICATION

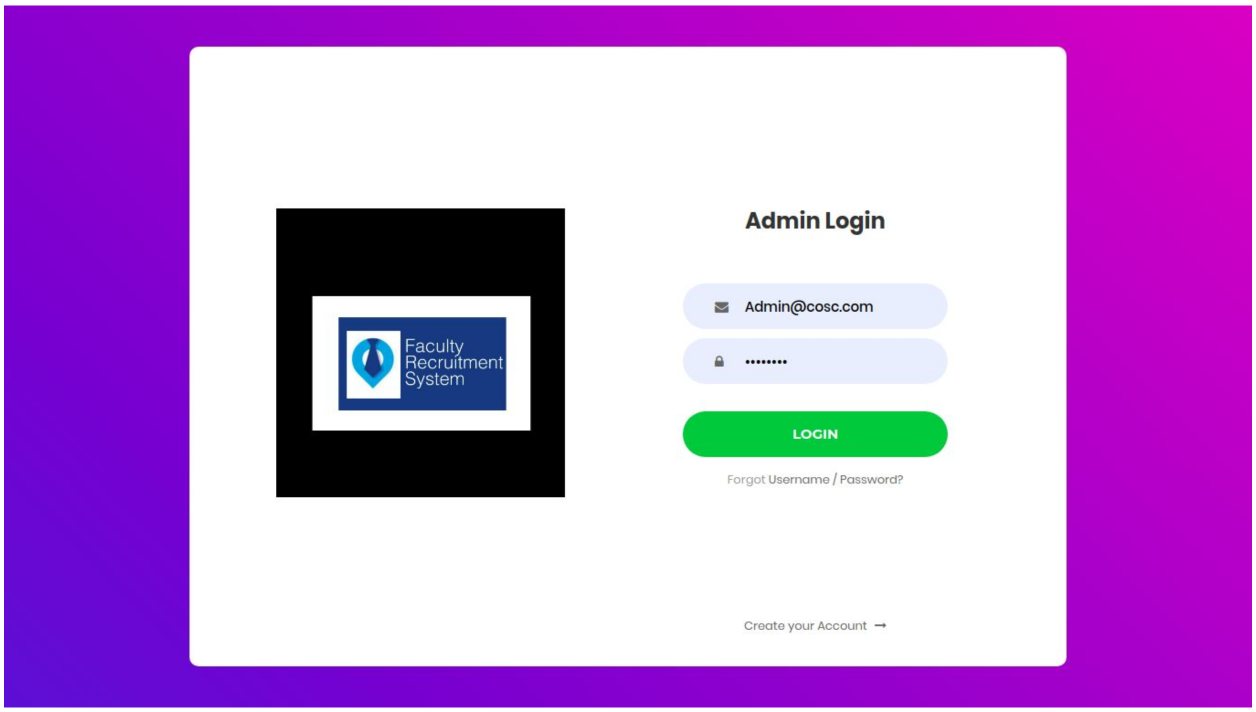
1.Web Application:

These are the required web pages as per the design of UI .

- Admin Login page
- Options page
- Upload Vacancies page
- View Vacancies page
- Select Applicants page
- Recruited faculty page

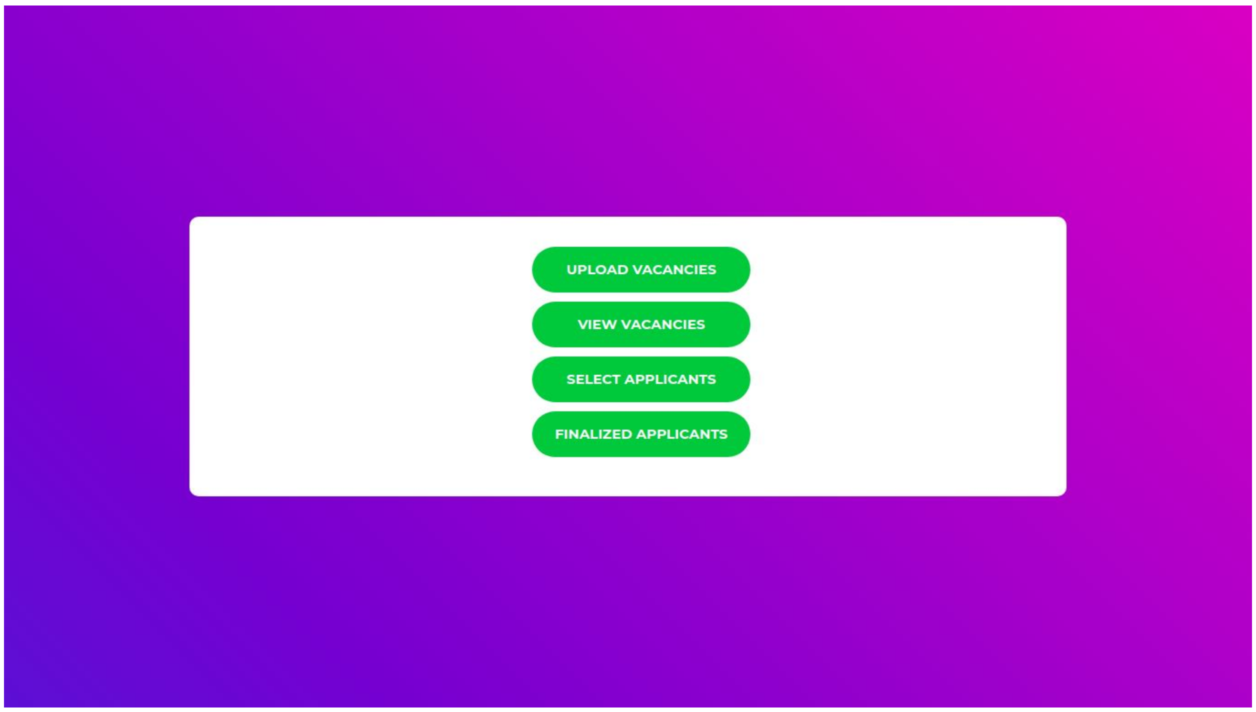
Admin Login Page:

In this web page, admin can login with mail and password.



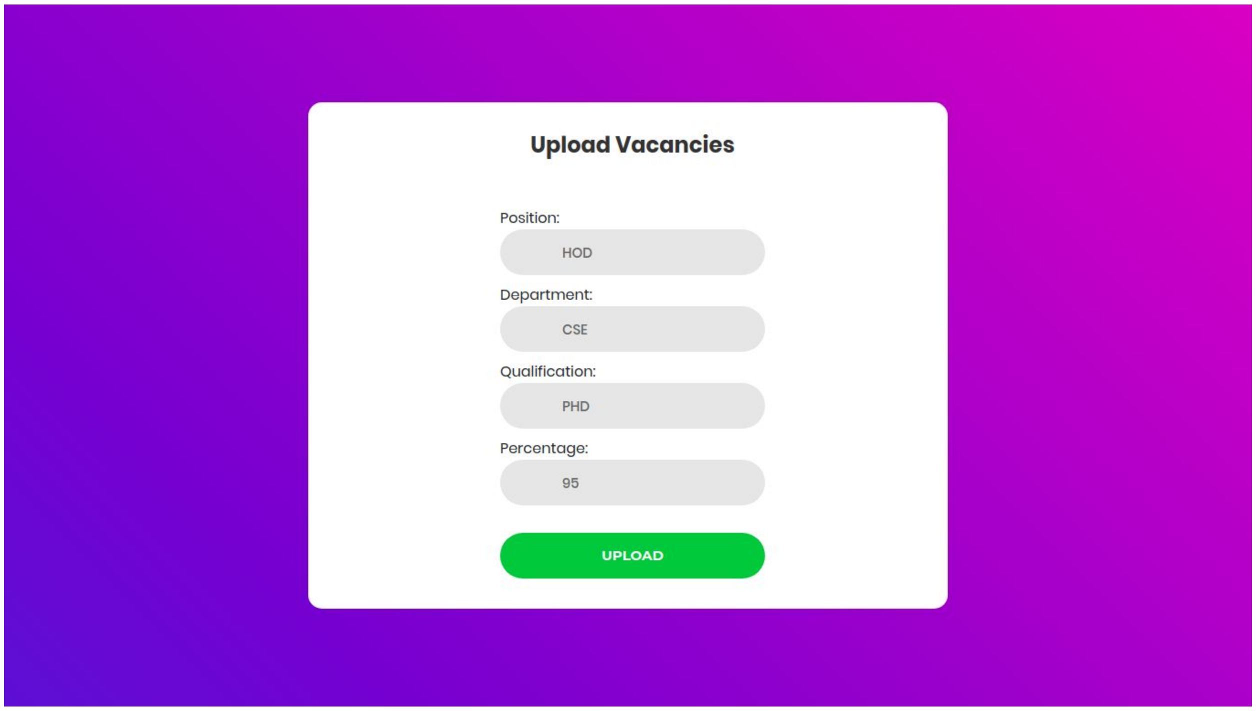
Options Page:

Options page is a navigator to various other web pages like uploading vacancies ,viewing vacancies ,selecting applicants,and recruited applicants.



Uploading Vacancies Page:

In this uploading vacancies page ,admin can upload the information regarding the vacancies like position,department,percentage .


Upload Vacancies

Position:
HOD

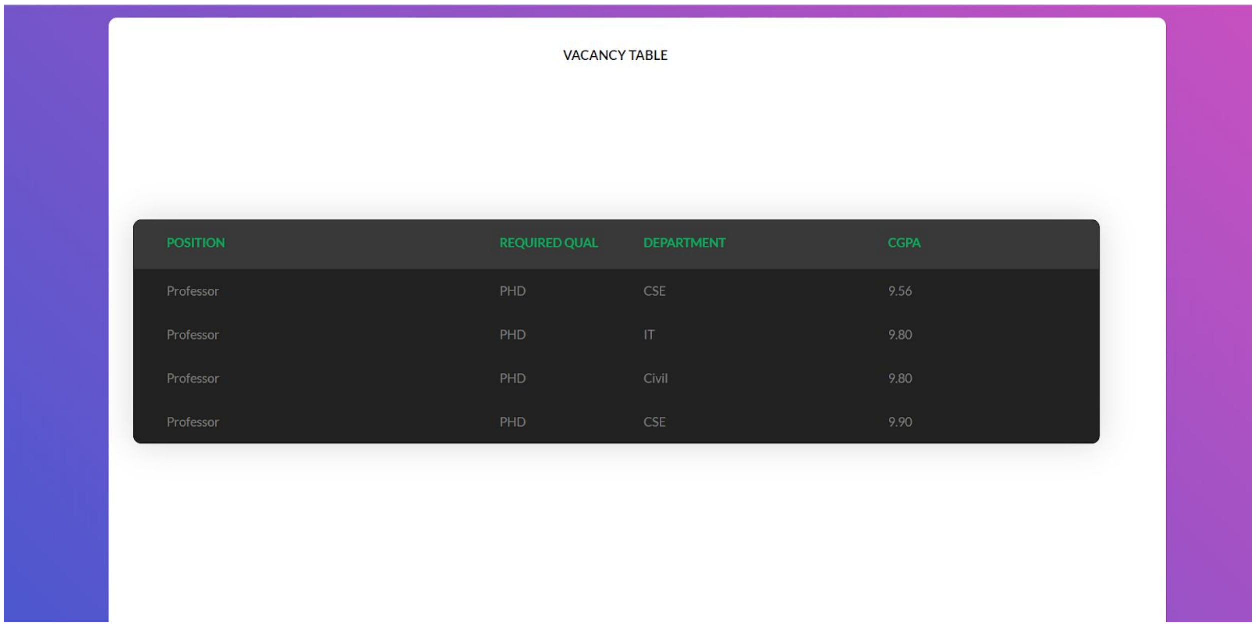
Department:
CSE

Qualification:
PHD

Percentage:
95

Viewing Vacancies Page:

In this page,admin can view the vacancies uploaded but not recruited yet.

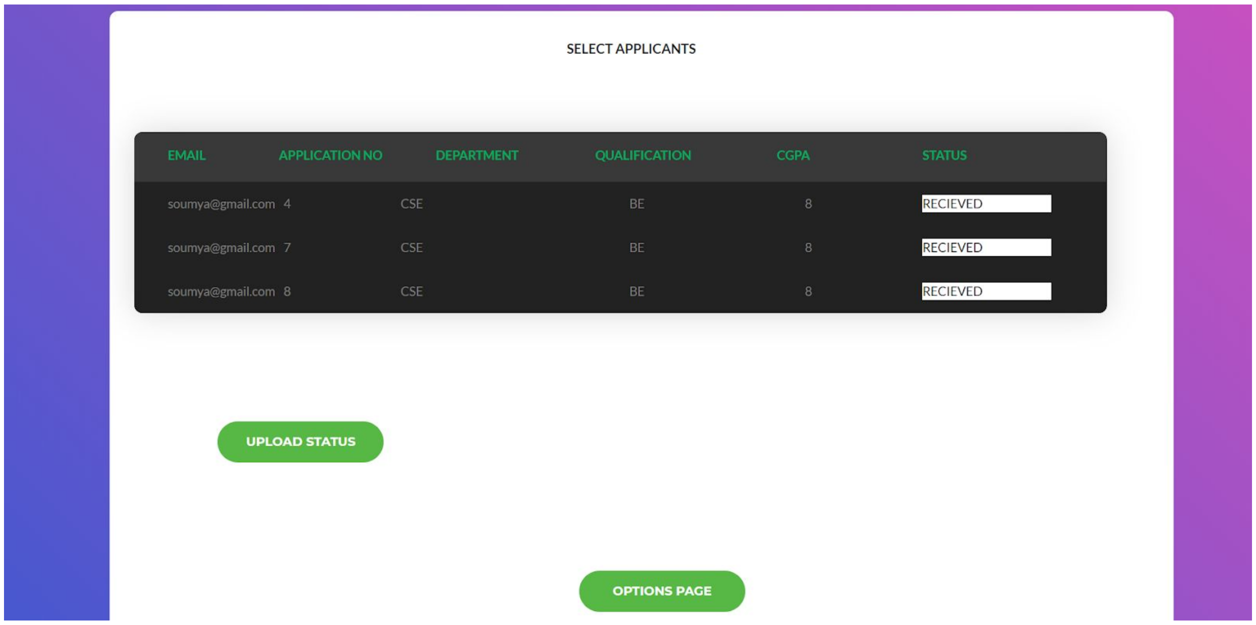


VACANCY TABLE

POSITION	REQUIRED QUAL	DEPARTMENT	CGPA
Professor	PHD	CSE	9.56
Professor	PHD	IT	9.80
Professor	PHD	Civil	9.80
Professor	PHD	CSE	9.90

Select Applicants Page:

In this page,the admin can see the applicants information like qualification and can upload the status whether he is approved or declined.

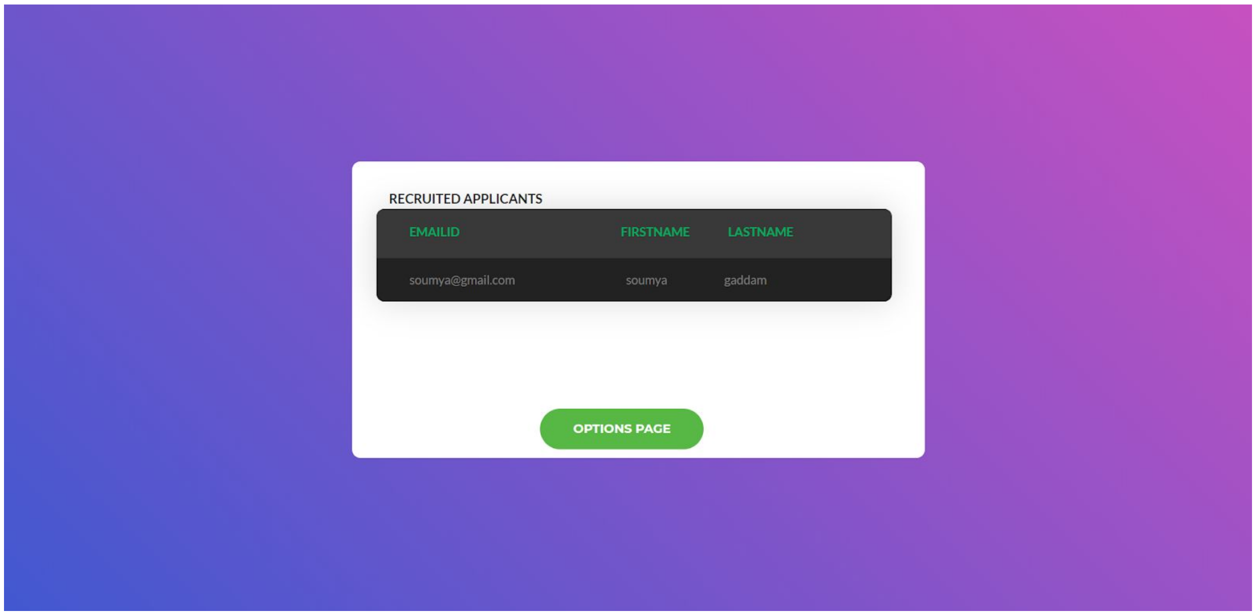


SELECT APPLICANTS

EMAIL	APPLICATION NO	DEPARTMENT	QUALIFICATION	CGPA	STATUS
soumya@gmail.com	4	CSE	BE	8	RECIEVED
soumya@gmail.com	7	CSE	BE	8	RECIEVED
soumya@gmail.com	8	CSE	BE	8	RECIEVED

Recruited Faculty Page:

In this page, admin can see the recruited applicants information.



RECRUITED APPLICANTS

EMAILID	FIRSTNAME	LASTNAME
soumya@gmail.com	soumya	gaddam

2.Android Application:

These are the required app pages as per the design of UI .

- Main page
- User Registration page
- User Login page
- Home page
- Vacancies page
- Application Details page
- Application Status page

Main Page:

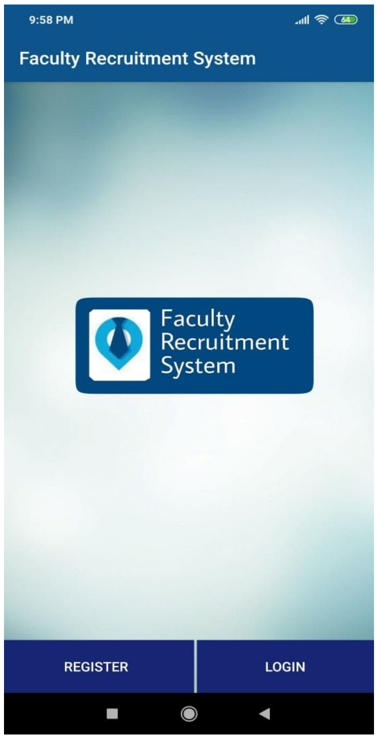
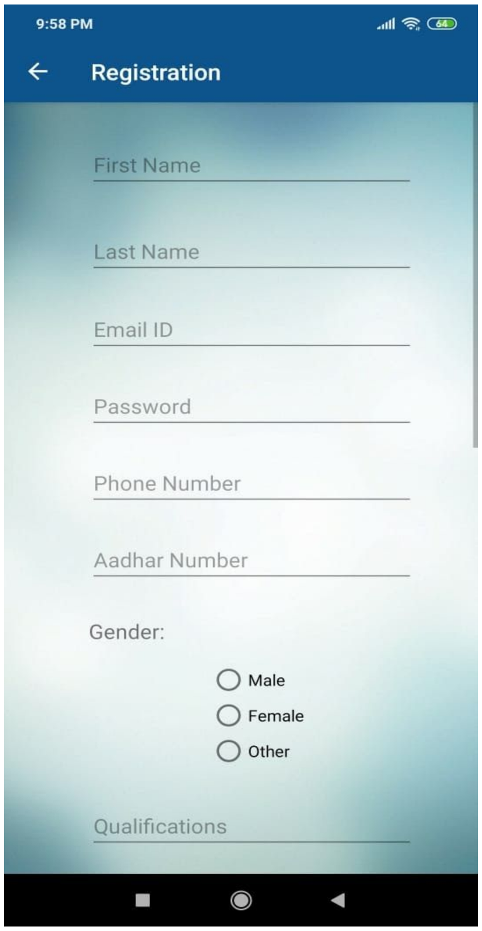


Fig 9.2.1: Main Page view when the user opens the application

Registration Page:

Page before entering details


9:58 PM

Registration

First Name

Last Name

Email ID

Password

Phone Number

Aadhar Number

Gender:

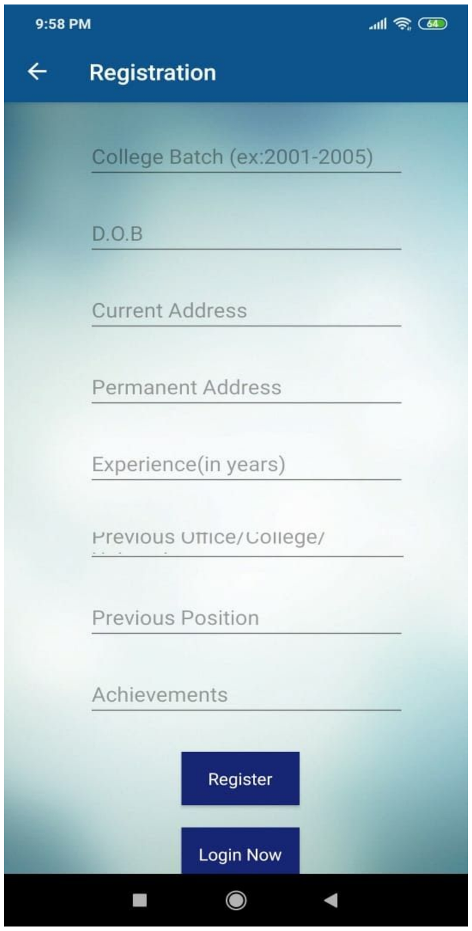
Male

Female

Other

Qualifications

Fig 9.2.2(a): Initial set of fields to be filled in the registration page.


9:58 PM

Registration

College Batch (ex:2001-2005)

D.O.B

Current Address

Permanent Address

Experience(in years)

Previous Office/College/

Previous Position

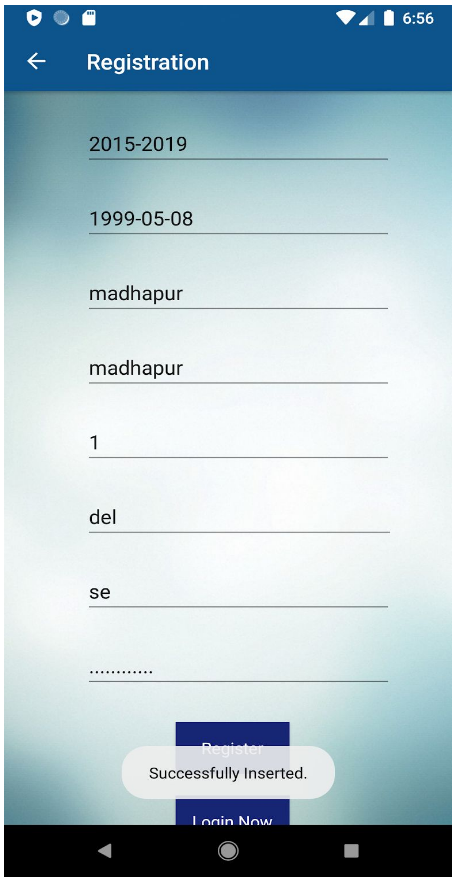
Achievements

Register

Login Now

Fig 9.2.2(b): Final set of fields to be filled in the registration page.

Messages displayed on successful entry and on trying to register using invalid credentials.



2015-2019

1999-05-08

madhapur

madhapur

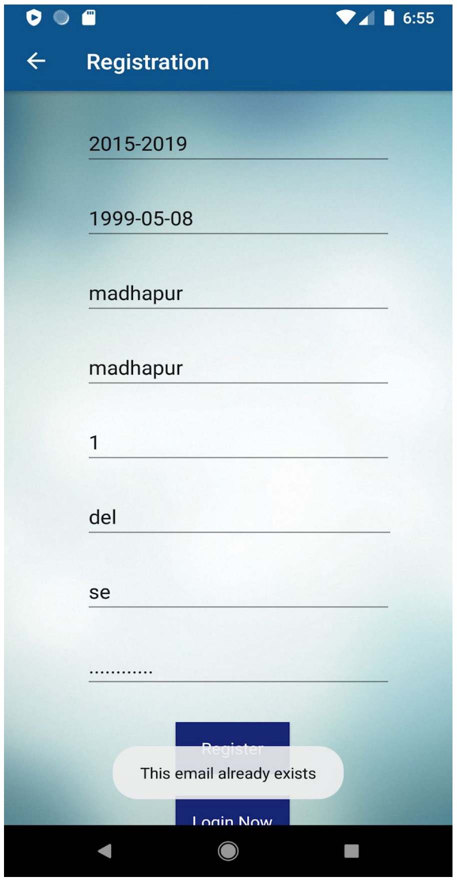
1

del

se

....

Successfully Inserted.



2015-2019

1999-05-08

madhapur

madhapur

1

del

se

....

This email already exists

Fig 9.2.3(a): Success message displayed after user registers using valid credentials.

Fig 9.2.3(b):Error message displayed after user registers with invalid credentials.

Login Page:

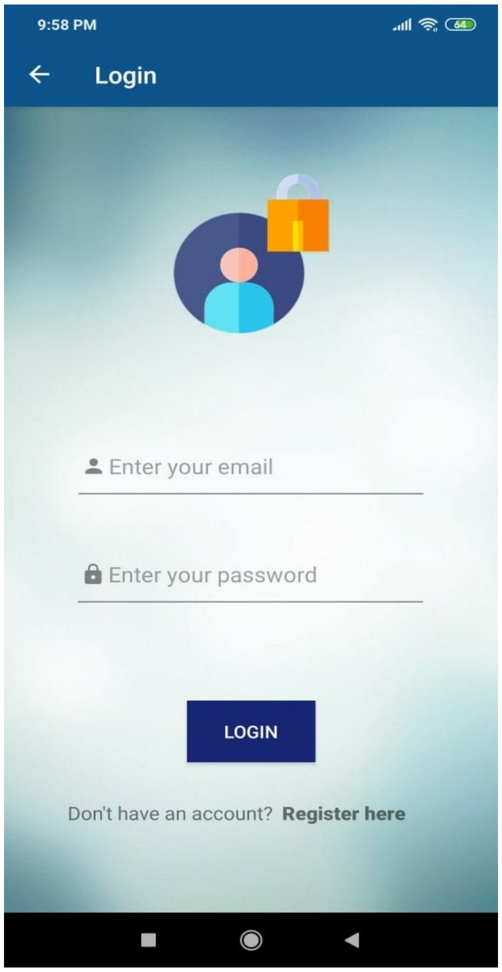


Fig 9.2.4(a): Login page when the user presses login.

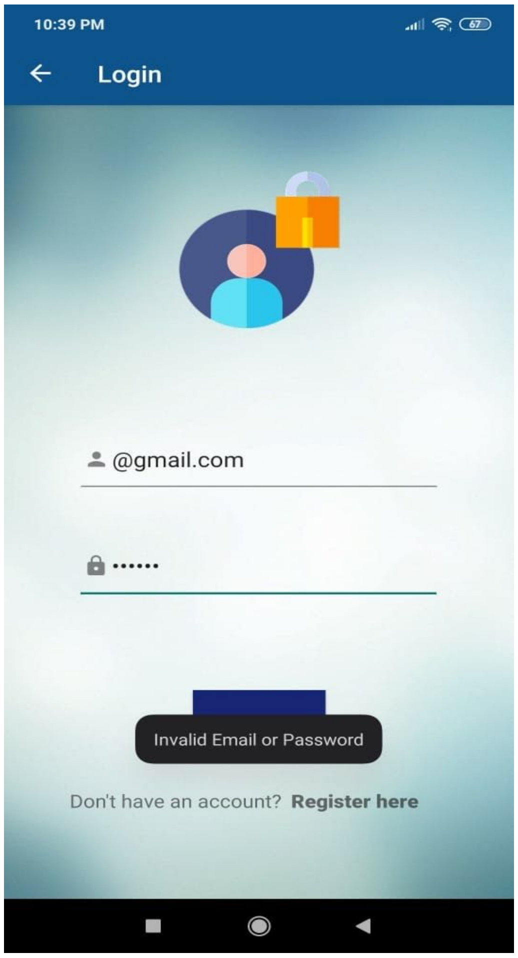

10:39 PM
Login
@gmail.com
••••••
Invalid Email or Password
Don't have an account? **Register here**

Fig 9.2.4(b): When user tries to login using invalid credentials.

Home Page:

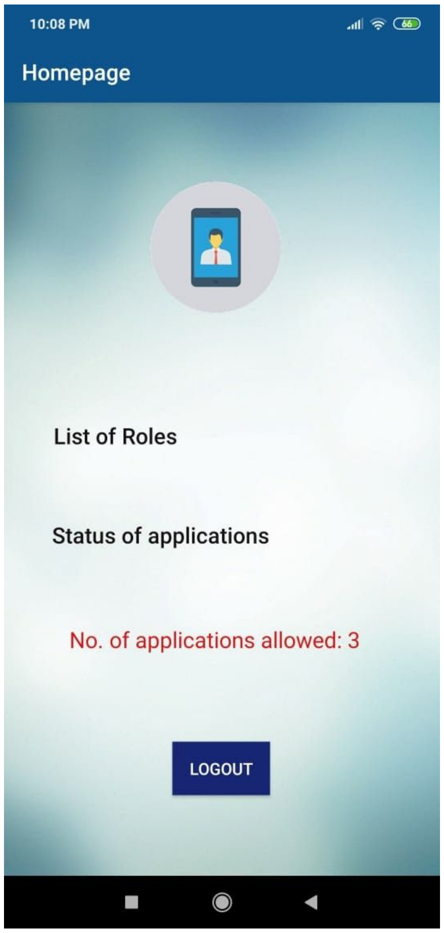
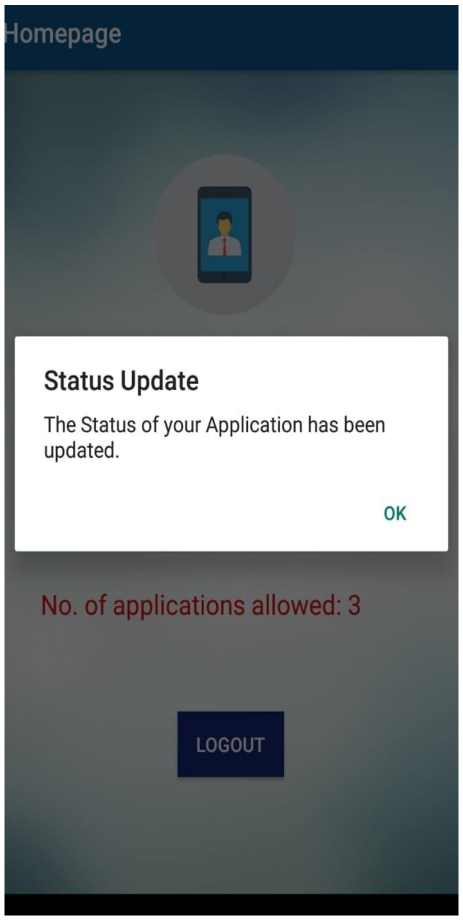


Fig 9.2.5(a): Home Page

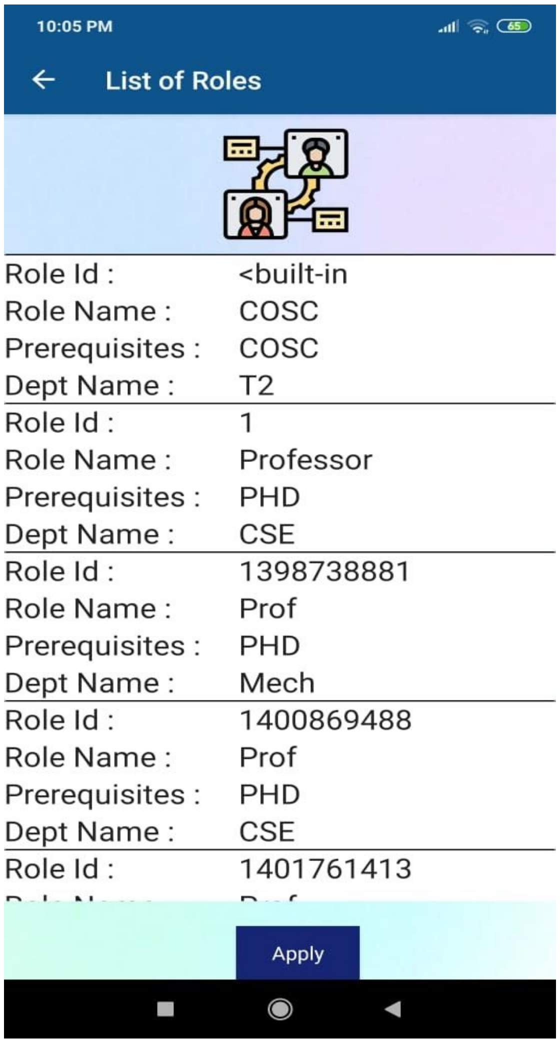


Status Update

The Status of your Application has been updated.

Fig 9.2.5(b): Home Page with the status update prompt

Vacancies Page:


10:05 PM

List of Roles

Role Id :	<built-in
Role Name :	COSC
Prerequisites :	COSC
Dept Name :	T2
Role Id :	1
Role Name :	Professor
Prerequisites :	PHD
Dept Name :	CSE
Role Id :	1398738881
Role Name :	Prof
Prerequisites :	PHD
Dept Name :	Mech
Role Id :	1400869488
Role Name :	Prof
Prerequisites :	PHD
Dept Name :	CSE
Role Id :	1401761413

Fig 9.2.6: List of Roles displayed as entered by the admin

Application Details Page:

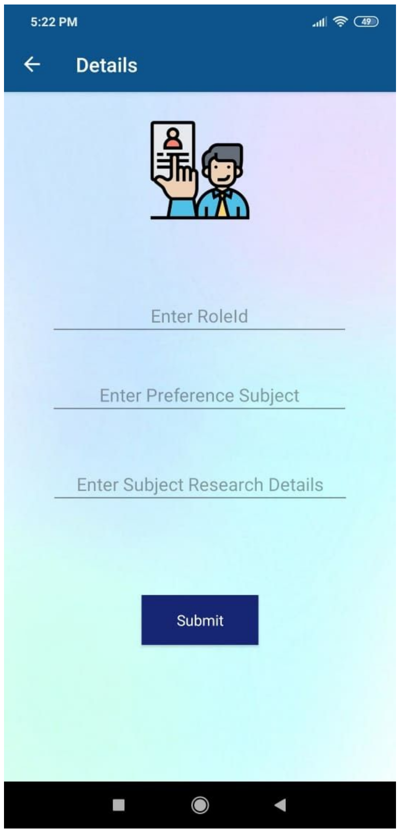


Fig 9.2.7(a): Application Details Page

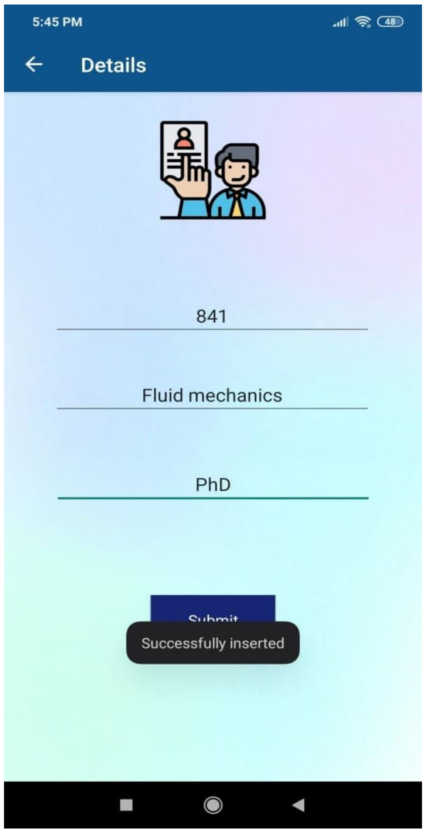
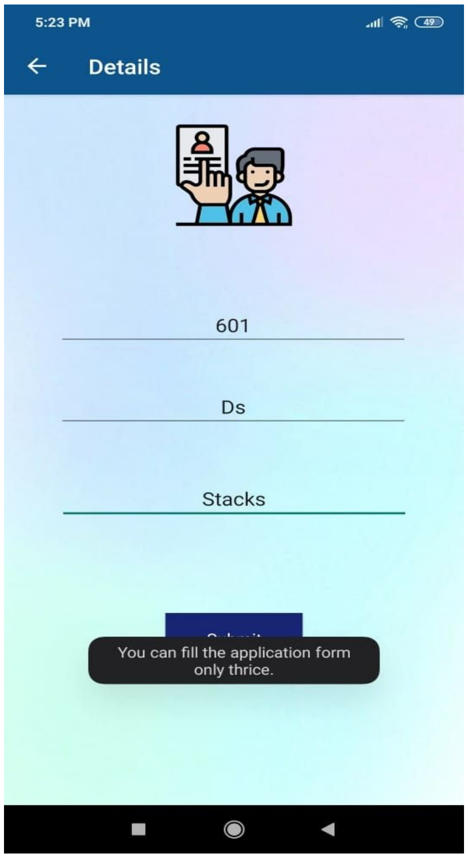
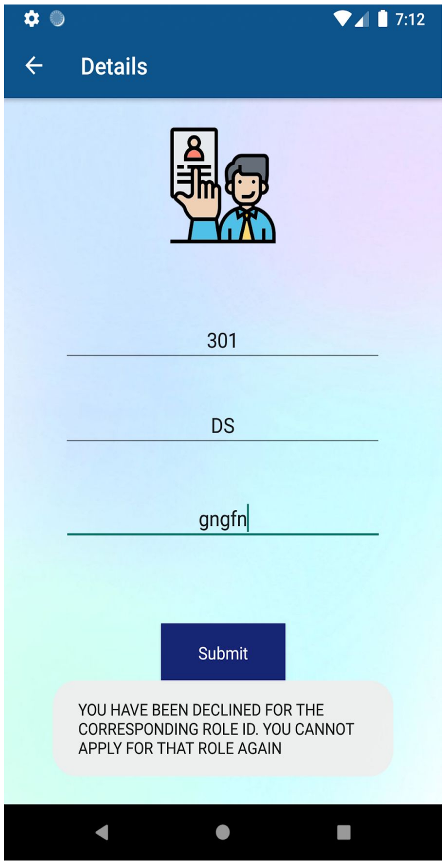


Fig 9.2.7(b): Application Details success message.



You can fill the application form only thrice.

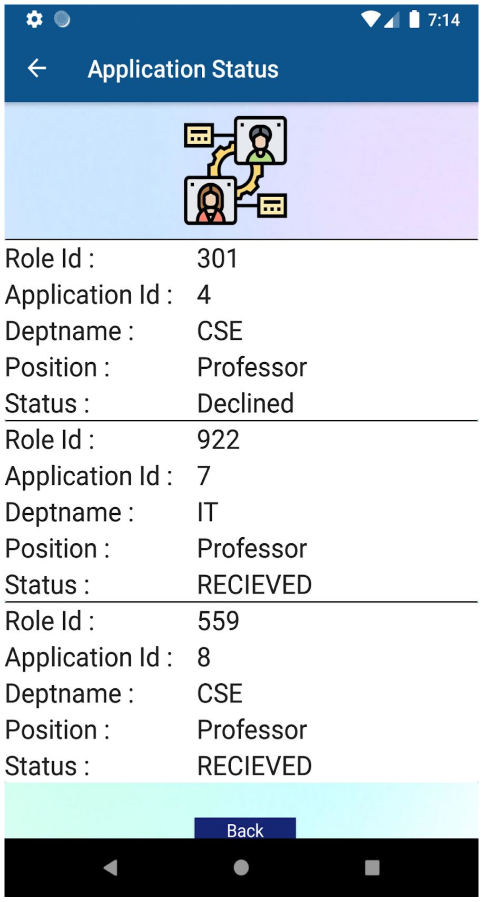
Fig 9.2.7(c):Limited to three message



YOU HAVE BEEN DECLINED FOR THE CORRESPONDING ROLE ID. YOU CANNOT APPLY FOR THAT ROLE AGAIN

Fig 9.2.7(d): Declined message

Application Status Page:



Role Id :	301
Application Id :	4
Deptname :	CSE
Position :	Professor
Status :	Declined
Role Id :	922
Application Id :	7
Deptname :	IT
Position :	Professor
Status :	RECIEVED
Role Id :	559
Application Id :	8
Deptname :	CSE
Position :	Professor
Status :	RECIEVED

Fig 9.2.8: Application Status displayed

3. RESTFUL APIs:

The APIs have been all connected to the database and the same can be witnessed from the db.py file.

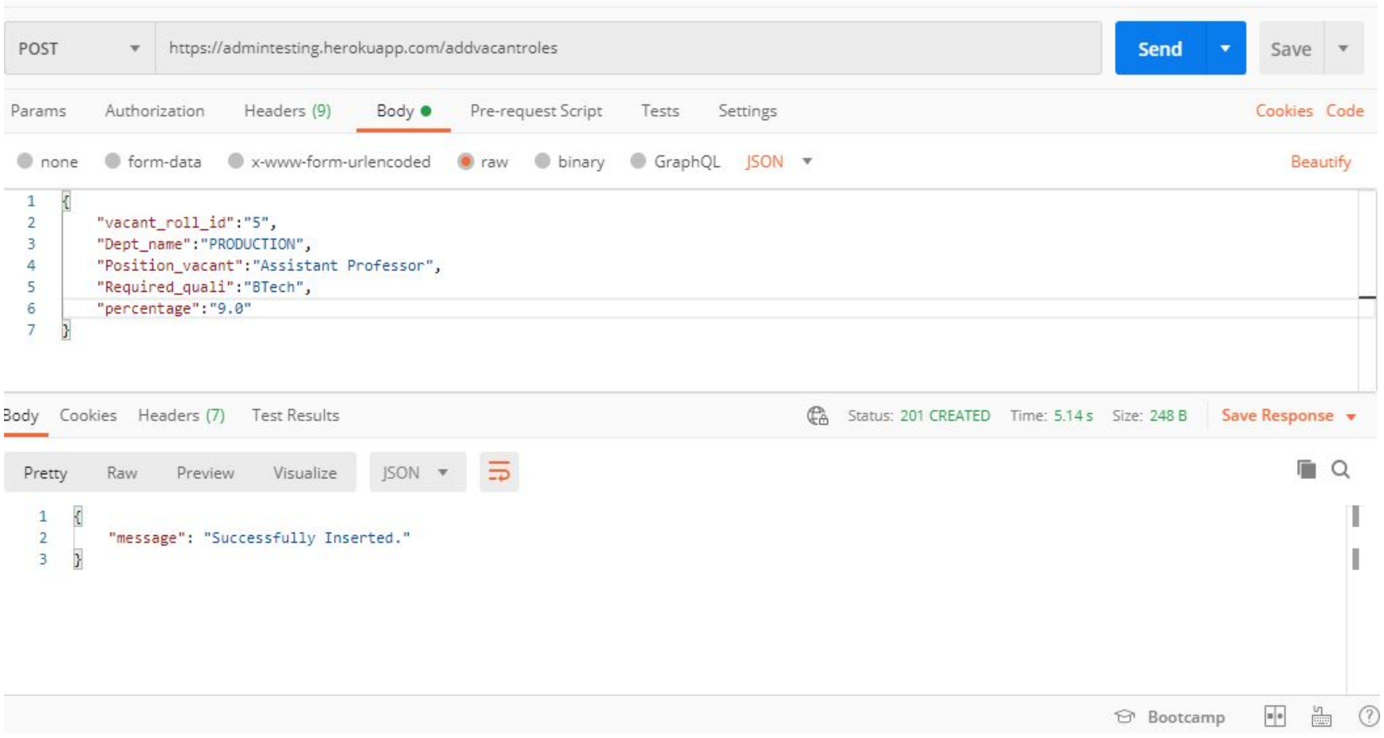
<u>RESTFUL APIs USING FLASK :</u>

- Flask RESTful API for Faculty Recruitment Application.
- All the APIs have been tested by POSTMAN tool successfully.
- This API has been deployed onto Heroku.
- The main purpose of this API is for users to apply for different roles of faculty positions and for admin to add the vacancies and select the recruited people.
- Link : https://admintesting.herokuapp.com/
- As mentioned in the heading, the APIs have been designed using FLASK.
- FLASK was installed using pip command and all the requirements and dependencies were installed.
- The Requirements are:
    - Flask
    - Flask-RESTful
    - flask-jwt-extended
    - pymysql
    - gunicorn
    - flask_cors
    - waitress

Going to all the endpoints created :

ADMIN SIDE :

/addvacantroles : POSTs all the vacant roles of each department of a college/university into the Database by the admin.



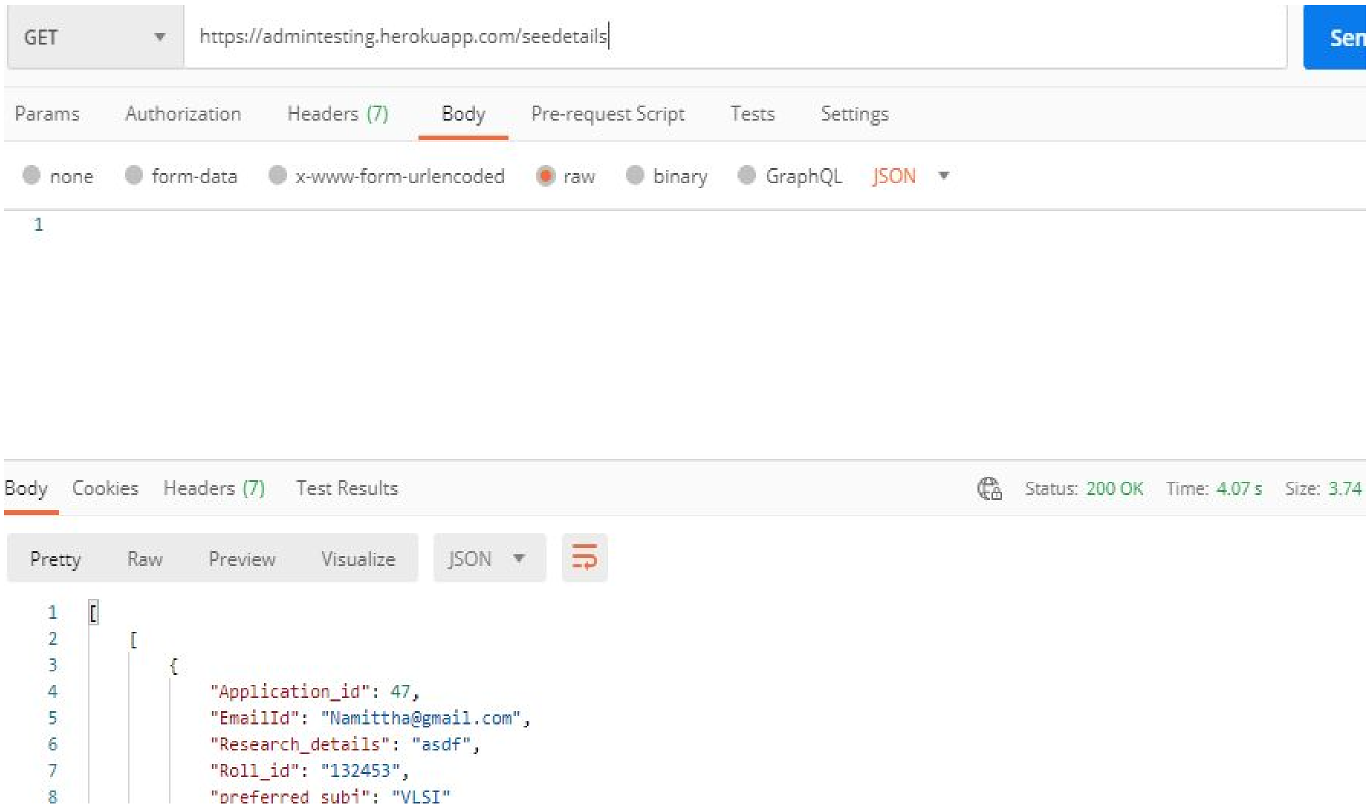
POST https://admintesting.herokuapp.com/addvacantroles

```
{
    "vacant_roll_id":"5",
    "Dept_name":"PRODUCTION",
    "Position_vacant":"Assistant Professor",
    "Required_quali":"BTech",
    "percentage":"9.0"
}
```

Status: 201 CREATED Time: 5.14 s Size: 248 B

```
{
    "message": "Successfully Inserted."
}
```

/seedetails : GETs all the application details and Qualification details filled up by the user, so that the admin can check whether the applied candidate is eligible for the role or not.

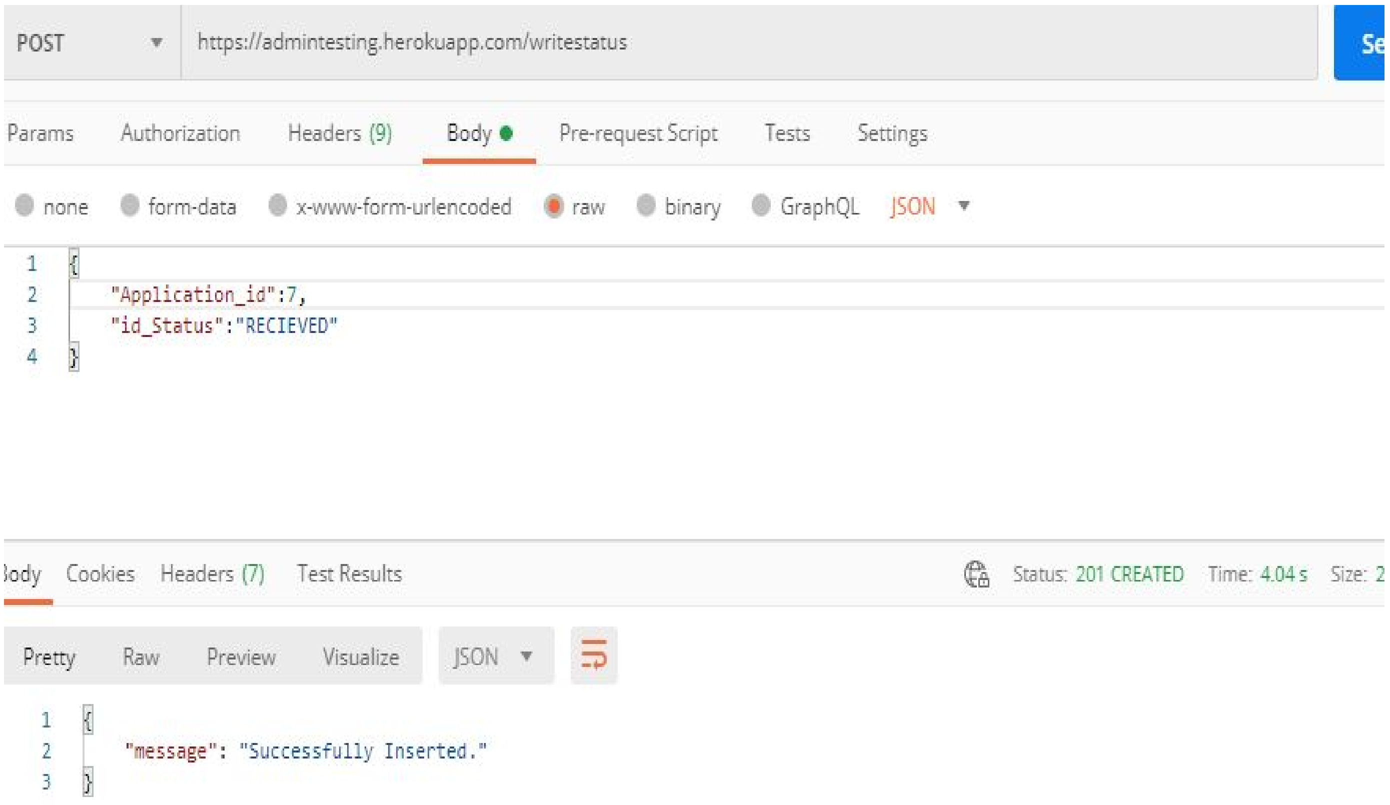


```
https://admintesting.herokuapp.com/seedetails
```

Status: 200 OK Time: 4.07 s Size: 3.74

```
[
    [
        {
            "Application_id": 47,
            "EmailId": "Namittha@gmail.com",
            "Research_details": "asdf",
            "Roll_id": "132453",
            "preferred_subj": "VLSI"
```

/writestatus : POSTs the status written by the admin to a particular application into the Database.



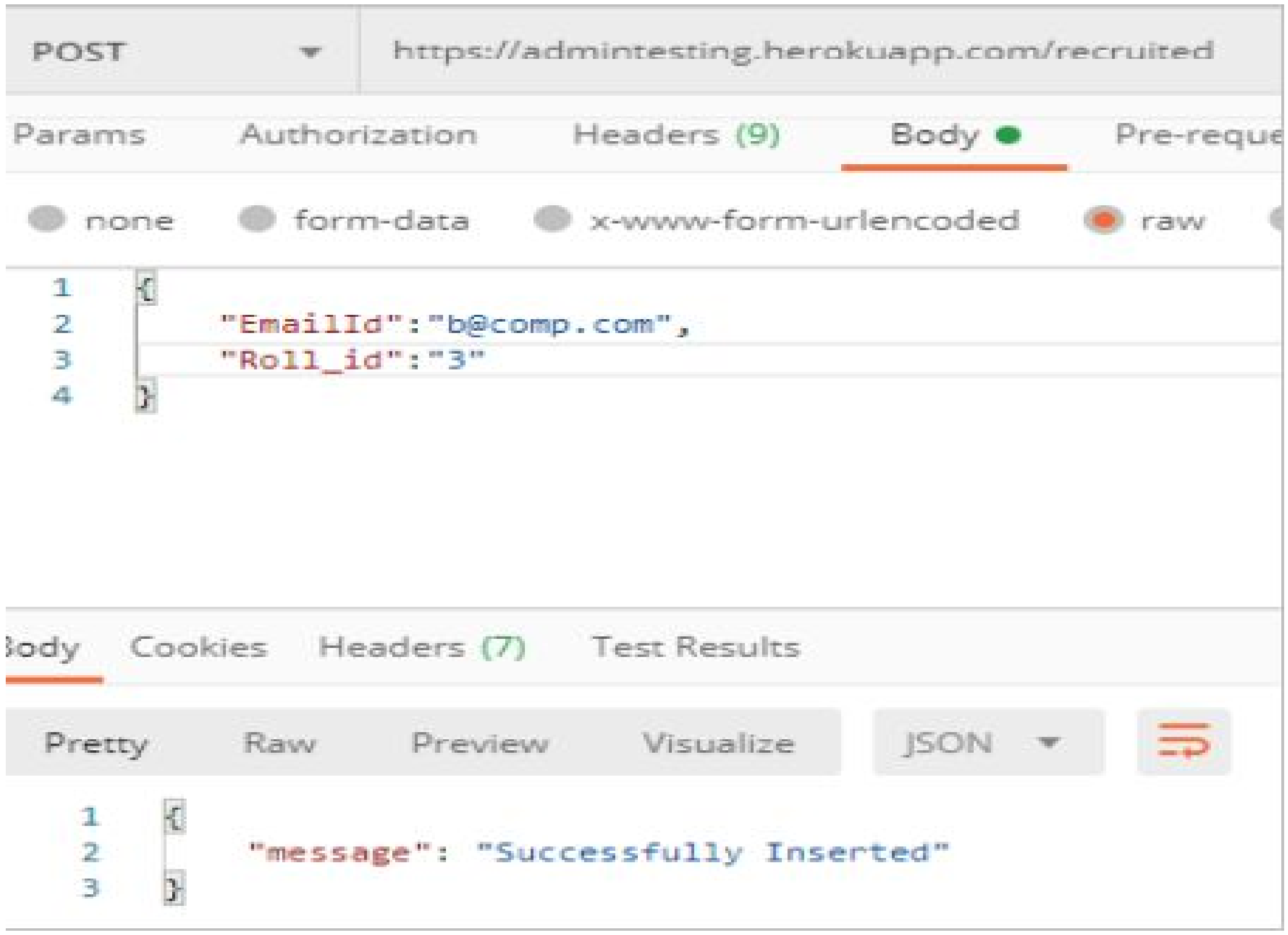
https://admintesting.herokuapp.com/writestatus

```
{
    "Application_id":7,
    "id_Status":"RECIEVED"
}
```

Status: 201 CREATED Time: 4.04 s Size: 2

```
{
    "message": "Successfully Inserted."
}
```

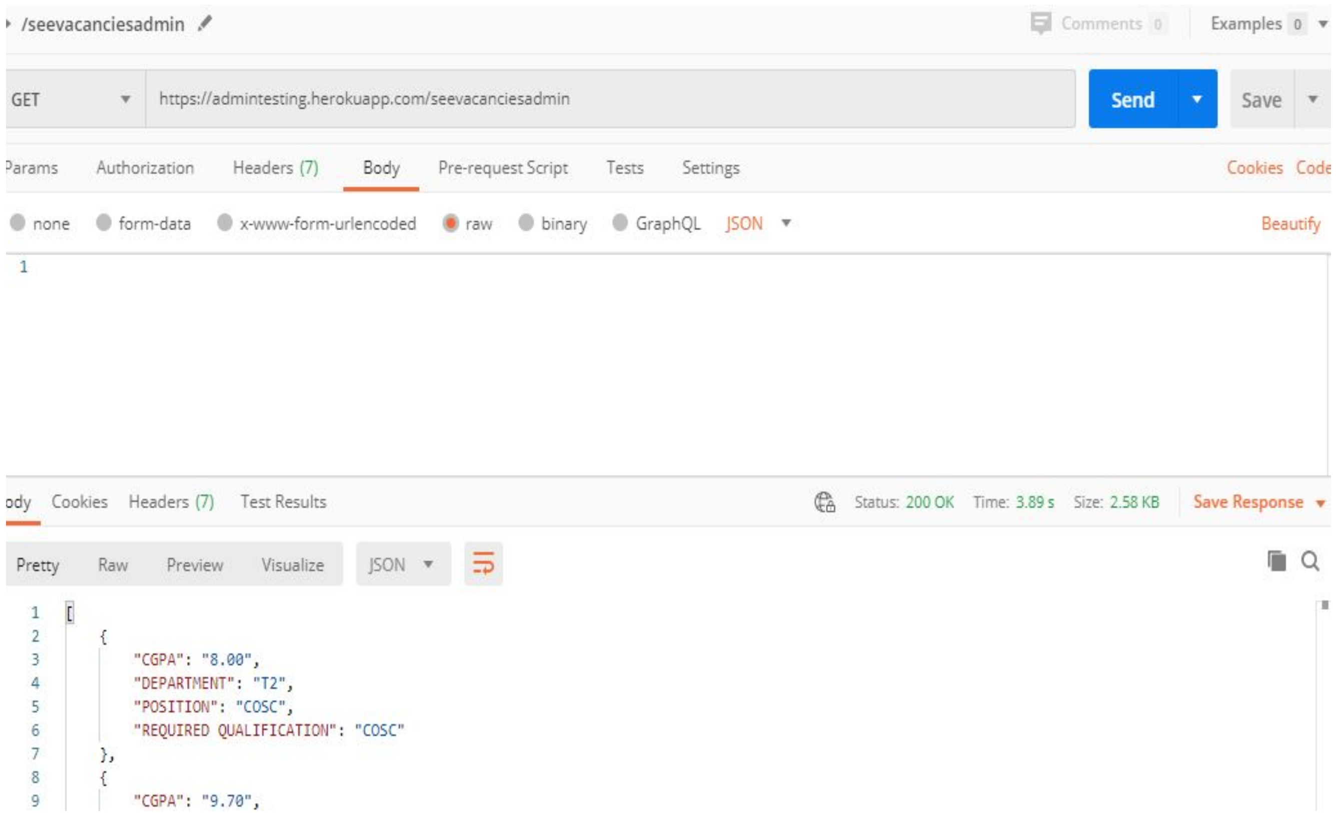
/recruited : POSTs the details of the faculty who are recruited into the Database by admin. This API also deletes all the applications of the person who was recruited, as mentioned in the Problem Statement. It also deletes the vacant Role, vacant department, and its Role ID from the database to which the person was recruited, as it is no longer vacant.



```
{
    "EmailId":"b@comp.com",
    "Roll_id":"3"
}
```

```
{
    "message": "Successfully Inserted"
}
```

/seevacanciesadmin : GETs all the vacancies from the database , so that the admin can refer.

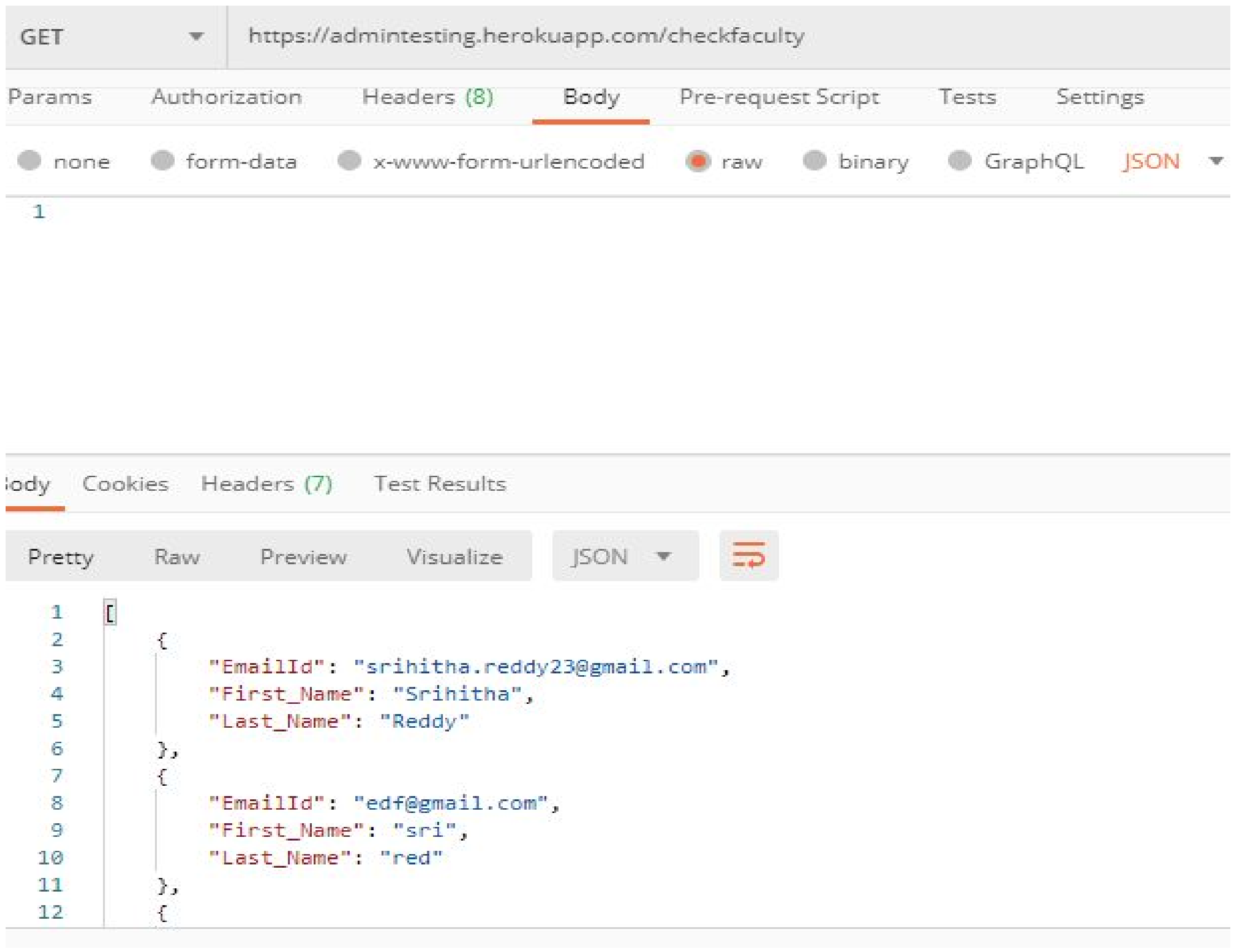


GET https://admintesting.herokuapp.com/seevacanciesadmin

Status: 200 OK Time: 3.89 s Size: 2.58 KB

```
[
    {
        "CGPA": "8.00",
        "DEPARTMENT": "T2",
        "POSITION": "COSC",
        "REQUIRED QUALIFICATION": "COSC"
    },
    {
        "CGPA": "9.70",
```

/checkfaculty : GETs the details of all the recruited faculty from the database for the reference of admin.



https://admintesting.herokuapp.com/checkfaculty

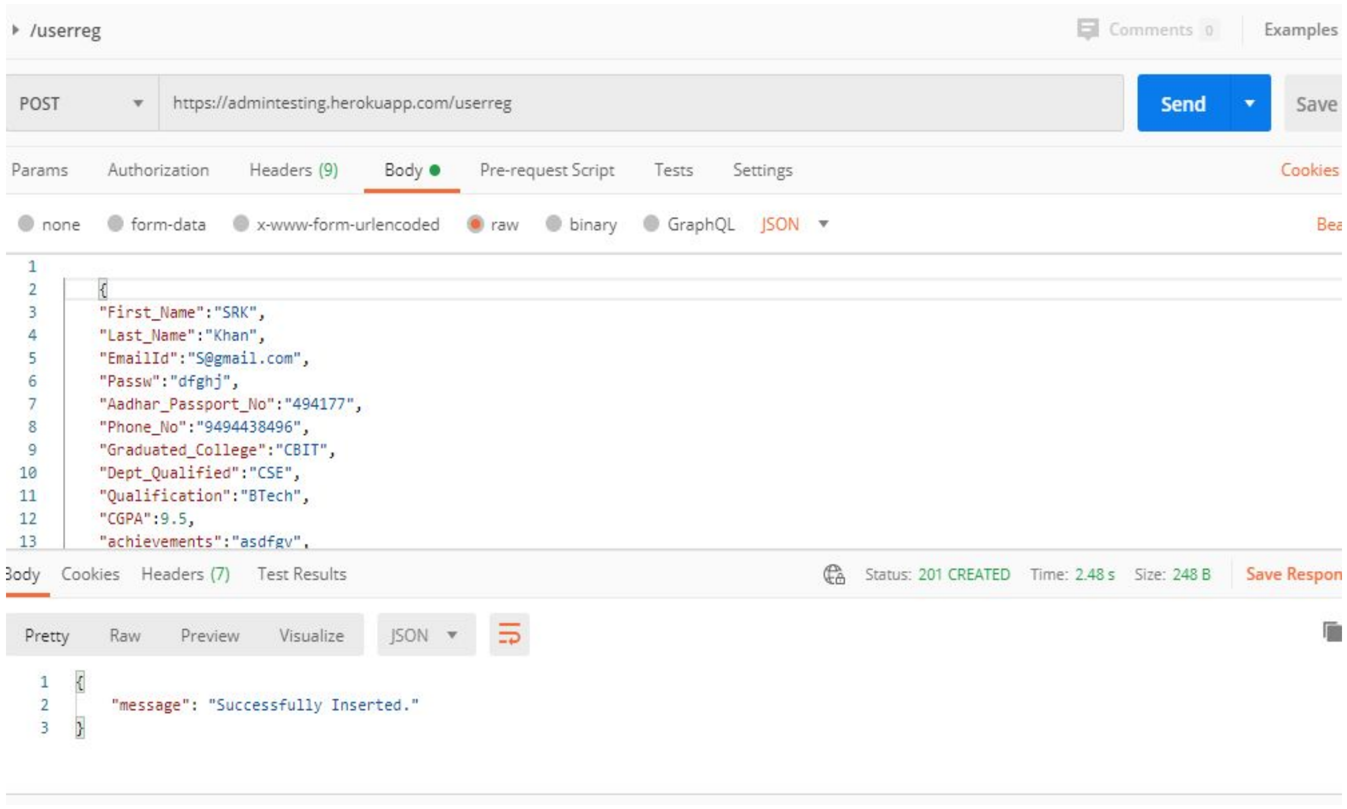
Body: ● raw, JSON

```
[
    {
        "EmailId": "srihitha.reddy23@gmail.com",
        "First_Name": "Srihitha",
        "Last_Name": "Reddy"
    },
    {
        "EmailId": "edf@gmail.com",
        "First_Name": "sri",
        "Last_Name": "red"
    },
    {
```

/enterdeclined : POSTs the Email ID and Role ID of the person into the database, if he was rejected to that Role.

USER SIDE :

/userreg : POSTs all the details of the registration form entered by the user into the Database.



POST https://admintesting.herokuapp.com/userreg

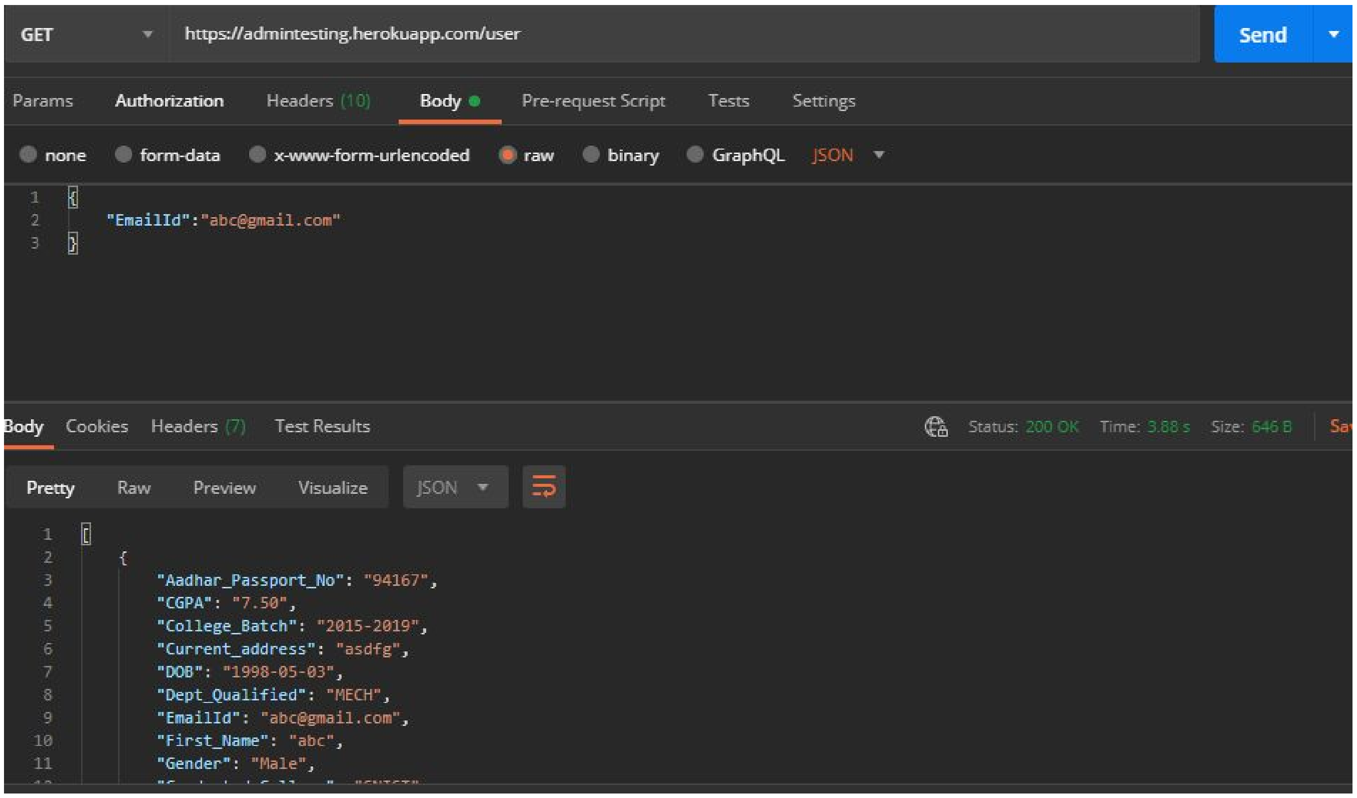
```

{
"First_Name":"SRK",
"Last_Name":"Khan",
"EmailId":"S@gmail.com",
"Passw":"dfghj",
"Aadhar_Passport_No":"494177",
"Phone_No":"9494438496",
"Graduated_College":"CBIT",
"Dept_Qualified":"CSE",
"Qualification":"BTech",
"CGPA":9.5,
"achievements":"asdfgv",
```

Status: 201 CREATED Time: 2.48 s Size: 248 B

JSON

```
{
    "message": "Successfully Inserted."
}
```

/user : GETs all the registration details of the user from the Database, when Email ID is given. (This API was created only for testing purpose).



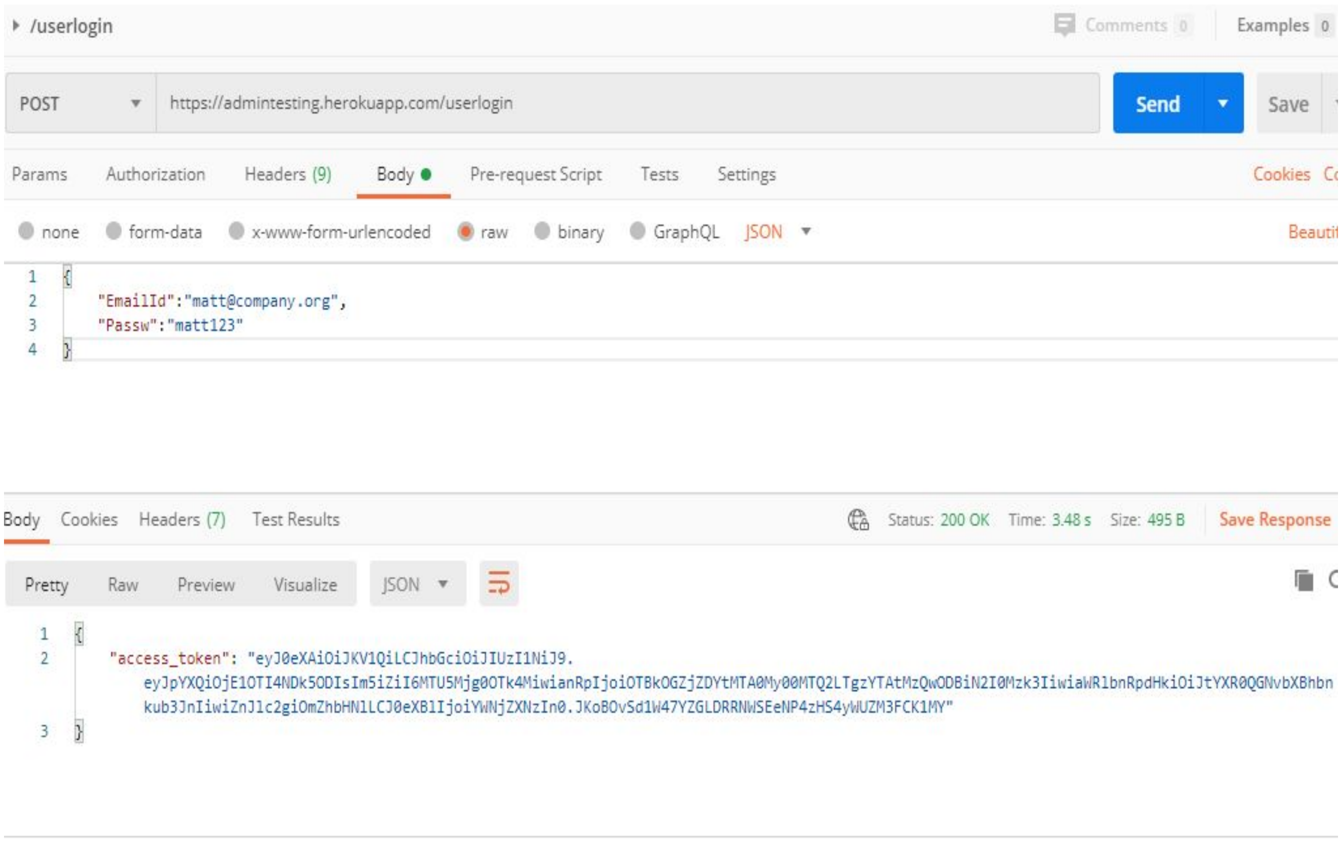
https://admintesting.herokuapp.com/user

```
{
    "EmailId":"abc@gmail.com"
}
```

Status: 200 OK Time: 3.88 s Size: 646 B

```
[
    {
        "Aadhar_Passport_No": "94167",
        "CGPA": "7.50",
        "College_Batch": "2015-2019",
        "Current_address": "asdfg",
        "DOB": "1998-05-03",
        "Dept_Qualified": "MECH",
        "EmailId": "abc@gmail.com",
        "First_Name": "abc",
        "Gender": "Male",
```

/userlogin : Takes a JSON object with 'Email ID' and 'Password' and gives back JWT token if exists in Users table. The JWT shall be used to access all the end points. For all the endpoints an Authorization Header should be included with value 'Bearer ' while testing in POSTMAN.



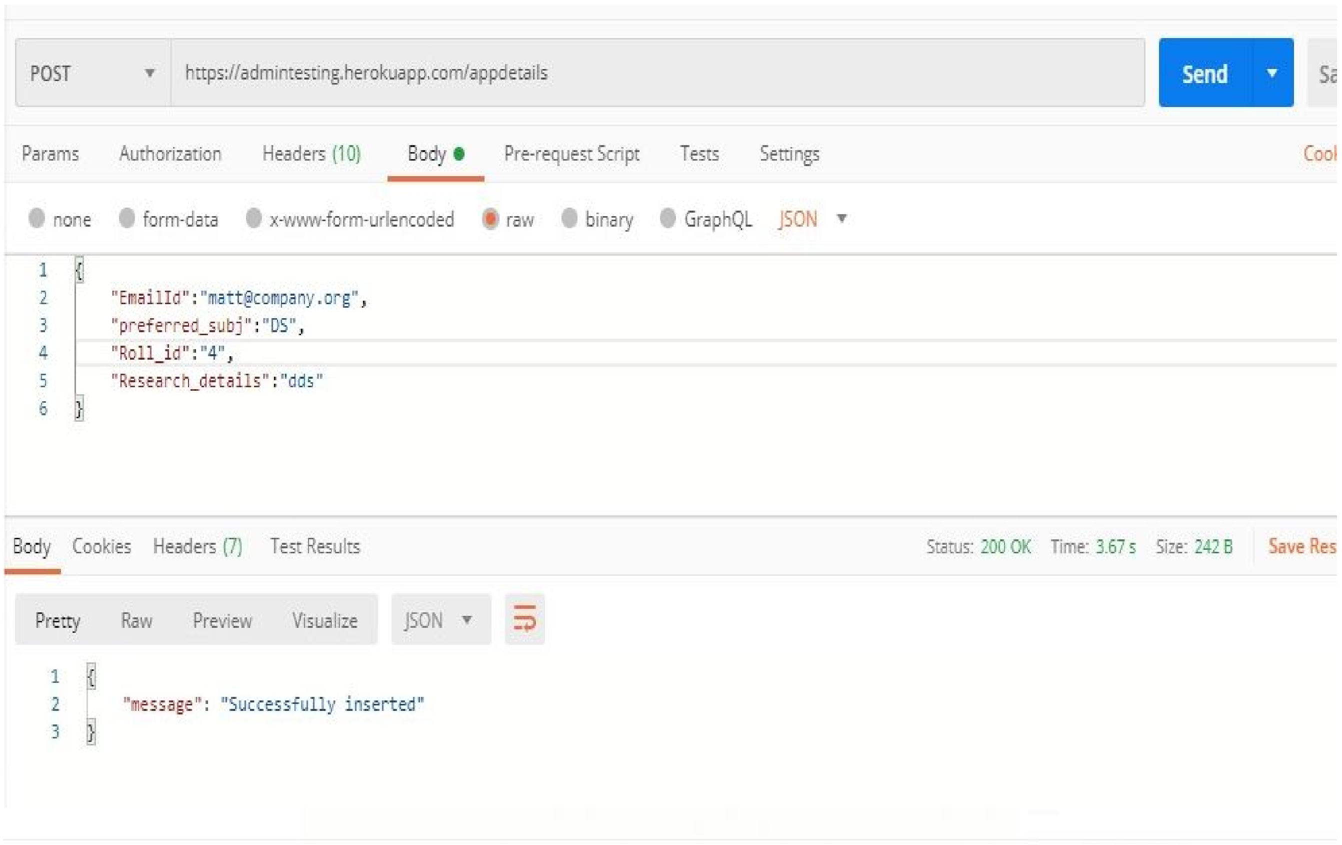
```
{
    "EmailId":"matt@company.org",
    "Passw":"matt123"
}
```

```
{
    "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
        eyJpYXQiOjE1OTI4NDk5ODIsIm5iZiI6MTU5Mjg0OTk4MiwianRpIjoiOTBkOGZjZDYtMTA0My00MTQ2LTgzYTAtMzQwODBiN2I0Mzk3IiwiaWRlbnRpdHkiOiJtYXR0QGNvbXBhbn
        kub3JnIiwiZnJlc2giOmZhbHNlLCJ0eXBlIjoiYWNjZXNzIn0.JKoBOvSd1W47YZGLDRRNWSEeNP4zHS4yWUZM3FCK1MY"
}
```

/appdetails : POSTs the application details of the user into the database. This API also lets the user enter an application form upto three times only.

Also, if any user is declined or rejected for a particular Role and he still applies for the same Role again, this API won't let the user to apply for that same role.

Case 1:



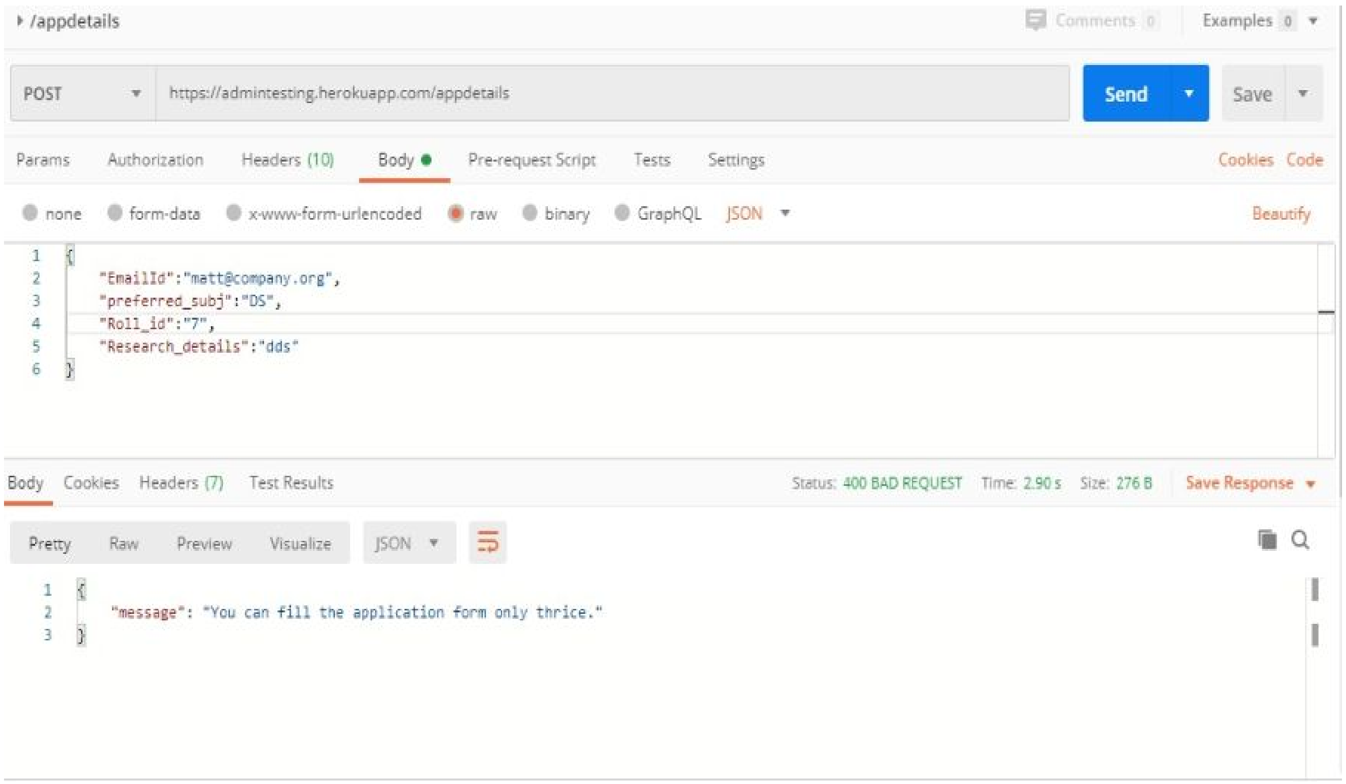
https://admintesting.herokuapp.com/appdetails

```
{
    "EmailId":"matt@company.org",
    "preferred_subj":"DS",
    "Roll_id":"4",
    "Research_details":"dds"
}
```

Status: 200 OK Time: 3.67 s Size: 242 B

```
{
    "message": "Successfully inserted"
}
```

Case 2:



POST https://admintesting.herokuapp.com/appdetails

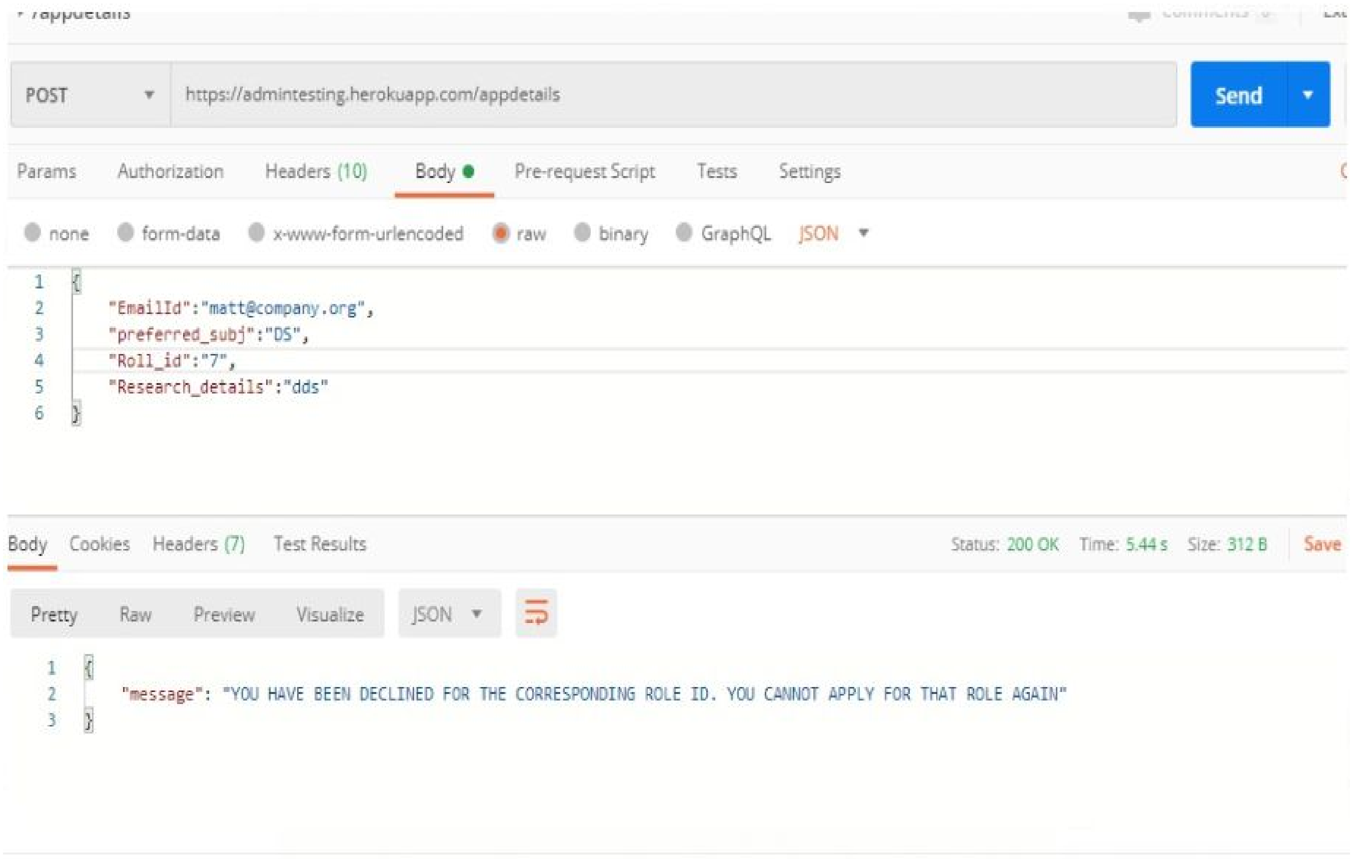
```
{
    "EmailId":"matt@company.org",
    "preferred_subj":"DS",
    "Roll_id":"7",
    "Research_details":"dds"
}
```

Status: 400 BAD REQUEST Time: 2.90 s Size: 276 B

JSON

```
{
    "message": "You can fill the application form only thrice."
}
```

Case 3:



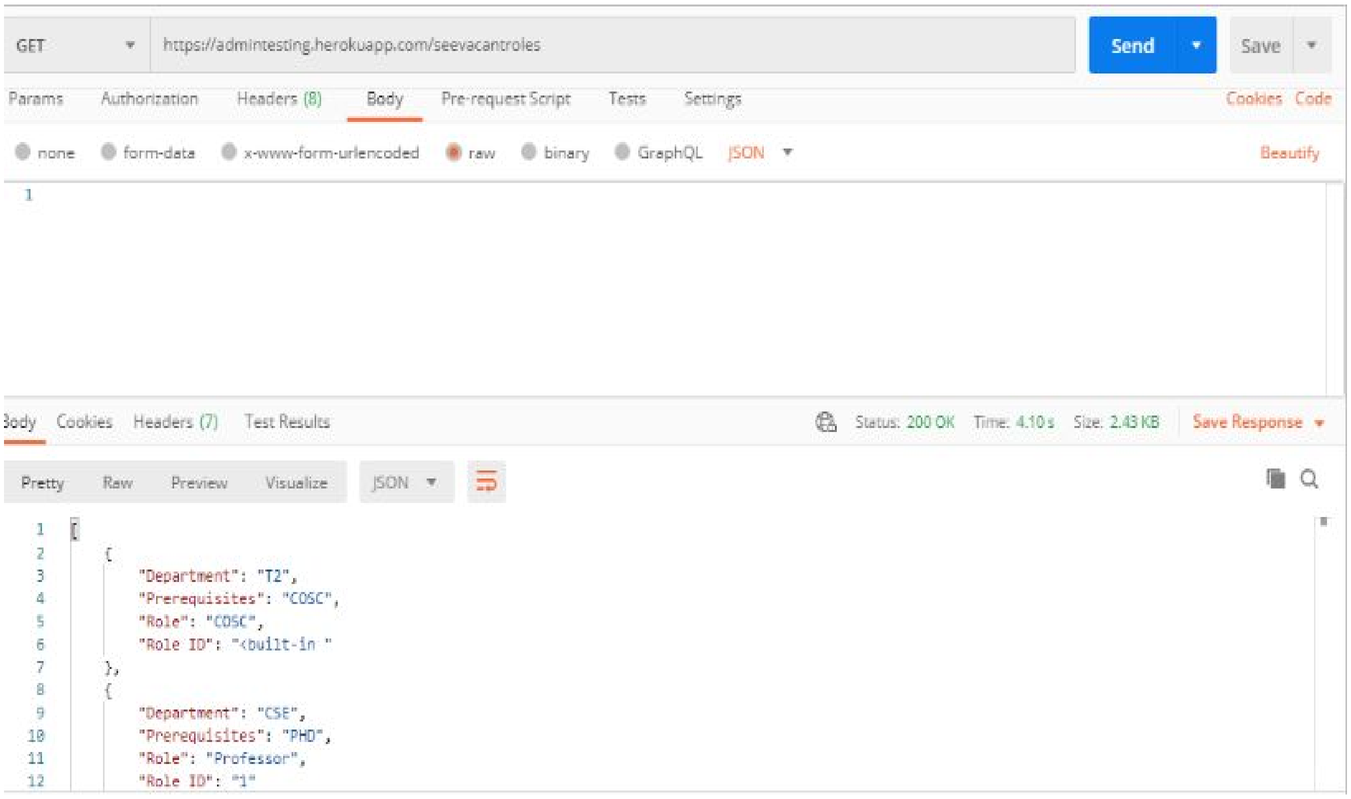
https://admintesting.herokuapp.com/appdetails

```
{
    "EmailId":"matt@company.org",
    "preferred_subj":"DS",
    "Roll_id":"7",
    "Research_details":"dds"
}
```

Status: 200 OK Time: 5.44 s Size: 312 B

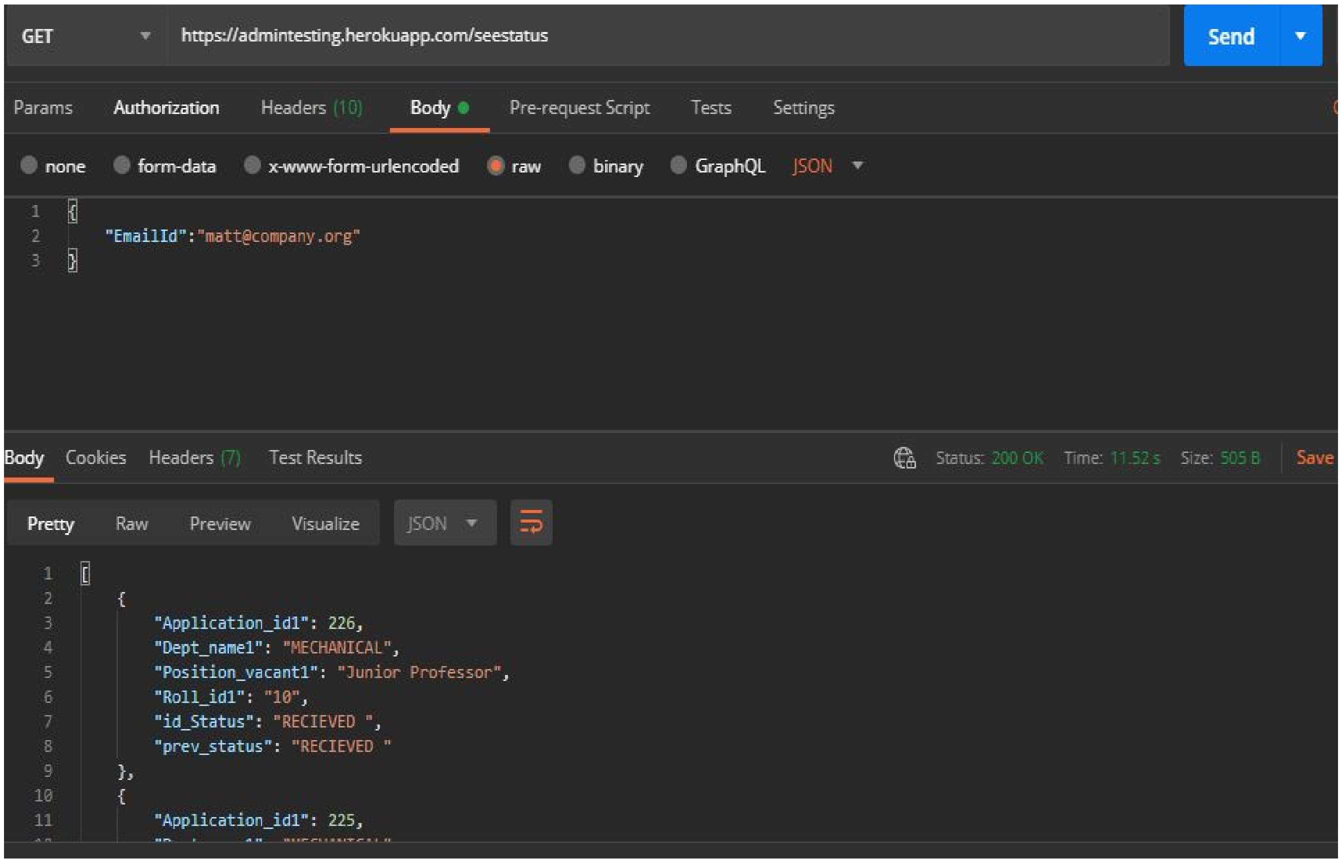
```
{
    "message": "YOU HAVE BEEN DECLINED FOR THE CORRESPONDING ROLE ID. YOU CANNOT APPLY FOR THAT ROLE AGAIN"
}
```

/seevacantroles : GETs all the vacant Role IDs, vacant roles, Required Qualification for that particular roles, the department in which there is Vacant role from the database to the user.



```
[
    {
        "Department": "T2",
        "Prerequisites": "COSC",
        "Role": "COSC",
        "Role ID": "<built-in "
    },
    {
        "Department": "CSE",
        "Prerequisites": "PHD",
        "Role": "Professor",
        "Role ID": "1"
```

/seestatus : GETs the Application ID, Role ID, Department, Position, Status, previous Status corresponding to an Email ID from the Database .



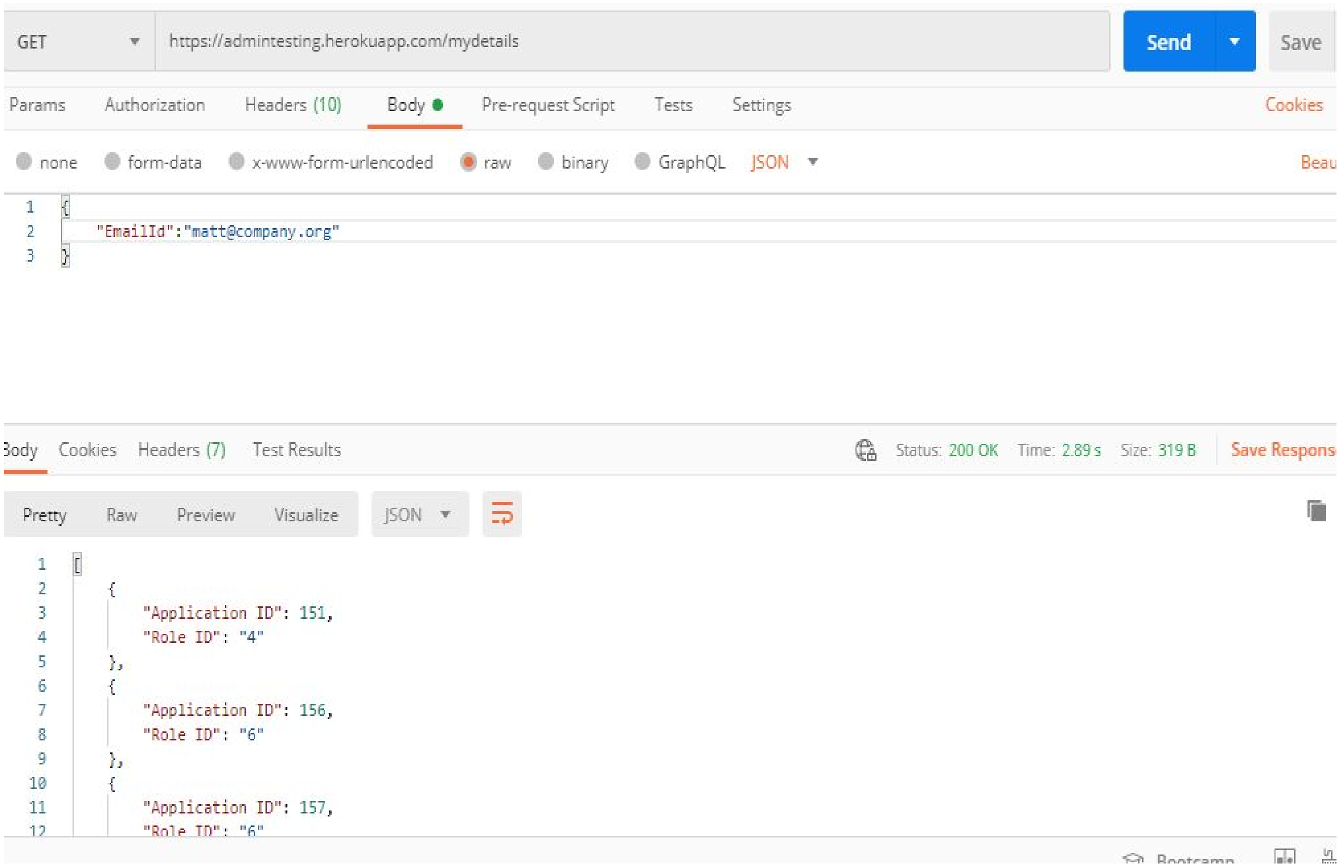
https://admintesting.herokuapp.com/seestatus

```
{
    "EmailId":"matt@company.org"
}
```

Status: 200 OK Time: 11.52 s Size: 505 B

```
[
    {
        "Application_id1": 226,
        "Dept_name1": "MECHANICAL",
        "Position_vacant1": "Junior Professor",
        "Roll_id1": "10",
        "id_Status": "RECIEVED ",
        "prev_status": "RECIEVED "
    },
    {
        "Application_id1": 225,
```

/mydetails : GETs the Application ID(s) and Role ID(s) corresponding to an Email ID from the Database.



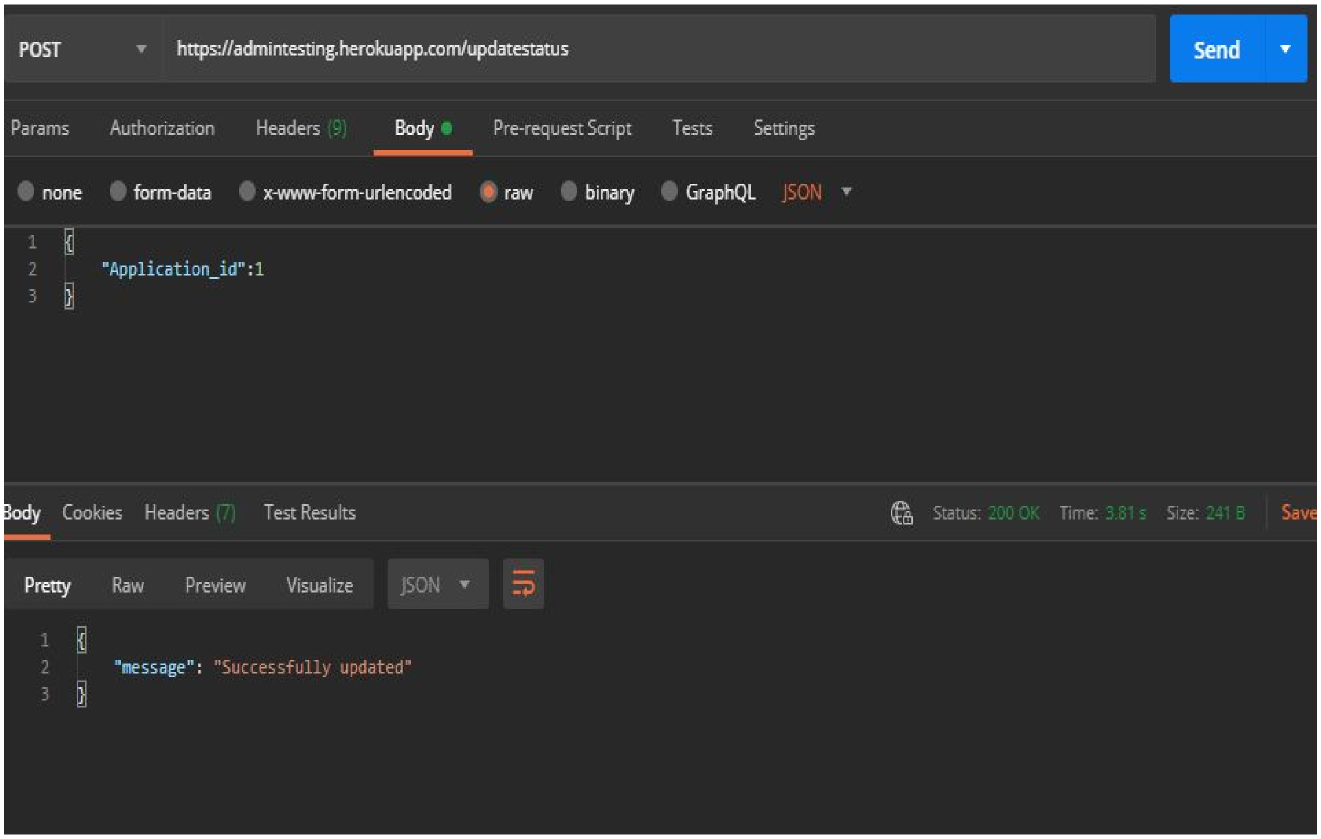
GET https://admintesting.herokuapp.com/mydetails

```
{
    "EmailId":"matt@company.org"
}
```

Status: 200 OK Time: 2.89 s Size: 319 B

```
[
    {
        "Application ID": 151,
        "Role ID": "4"
    },
    {
        "Application ID": 156,
        "Role ID": "6"
    },
    {
        "Application ID": 157,
        "Role ID": "6"
```

/updatestatus : Updates previous status to current status, given an Application ID


POST https://admintesting.herokuapp.com/updatestatus

Send

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings

none form-data x-www-form-urlencoded ● raw binary GraphQL JSON

```
{
    "Application_id":1
}
```

Body Cookies Headers (7) Test Results   Status: 200 OK Time: 3.81 s Size: 241 B Save

Pretty Raw Preview Visualize JSON

```
{
    "message": "Successfully updated"
}
```

FUTURE SCOPE OF WORK

Our application portal has accomplished all the functional requirements.In this project, we have briefly explored how available technology can possibly help how job recruitment and job seeking processes are implemented.Many different adaptations, tests, and experiments could be done on real data. Future work concerns deeper analysis of particular mechanisms, new proposals to try different methods,,experiments with real data,or simply curiosity etc.There can be security lapses,computer glitches etc.and our portal could be made more reliable and secure and user friendly interface.

CONCLUSION

As mentioned elsewhere in this report, the objective of this project was to implement a faculty recruitment application portal.This objective has been accomplished in the course of creating this project. We have used Mysql, Flask for restful APIs, Django, HTML, CSS, Bootstrap, Javascript, Java, XML, Android Studio. We would like to conclude that the project has achieved what it set out to accomplish, even though there will always be areas for potential improvement and enhancement.

REFERENCES

1. Videos of COSC Internship Drive.
2. https://developer.android.com/
3. http://stackoverflow.com/
4. https://www.onlogic.com/company/io-hub/a-simple-blog-with-django/
5. https://dev.mysql.com/doc/refman/8.0/en/sql-syntax.html
6. https://programminghistorian.org/en/lessons/creating-apis-with-python-and-flask
7. https://www.javatpoint.com/java-tutorial
8. https://www.quora.com
9. https://www.javatpoint.com/android-tutorial
10. https://www.tutorialspoint.com/java/index.htm
11. https://www.w3resource.com/java-exercises/
12. https://www.wikipedia.com
13. https://www.w3schools.com/java/
14. https://www.programiz.com/java-programming
15. https://www.tutorialspoint.com/android/index.htm
16. https://developer.android.com/guide
17. https://developer.android.com/training/basics/firstapp
18. https://www.vogella.com/tutorials/android.html
19. https://pymbook.readthedocs.io/en/latest/flask.html