PES University, Bengaluru

(Established under Karnataka Act 16 of 2013)

Department of Computer Science & Engineering

Session: Jan - May 2022

Object Oriented Analysis and Design with Java - Laboratory

UE19CS353

Mini Project

Report on

SMART CITY

By:

Guhan K– PES1UG19CS171

Hanuraag Baskaran – PES1UG19CS177

Hariharasudan S A– PES1UG19CS178

Harshita Vidapanakal – PES1UG19CS185

6th Semester - 'C'

1. Project Description

Our project, which is a smart city portal, enables you to find all the requisite information for different cities in one simple portal. We have implemented this for the following 4 cities:

1. Bangalore
2. Chennai
3. Hyderabad
4. Mangalore

The portal has the following sections:

1. Places to visit- A list of the tourist attractions in the city
2. Lodging- List of places for accomodation
3. Education-List of educational institute/schools
4. Map- Redirects to Google Maps of the city
5. Jobs- Redirects to naukri.com in the given city
6. Restaurants- a list of the most popular dining attractions in the area

We have an administrator, who has global access to the entire system, and oversees the registration and the verification of the users upon sign in. We have three predefined user roles, which determines which sections of the portal they can access, based on their needs. We have split the roles into the following:
1)Student
2)Tourist
3)Job Seeker

A student has access to the following sections of the portal:

1) Education
2) Map

A tourist has access to the following sections of the portal:

1) Places to visit
2) Lodging
3) Map
4) Restaurants

A job seeker has access to the following sections of the portal

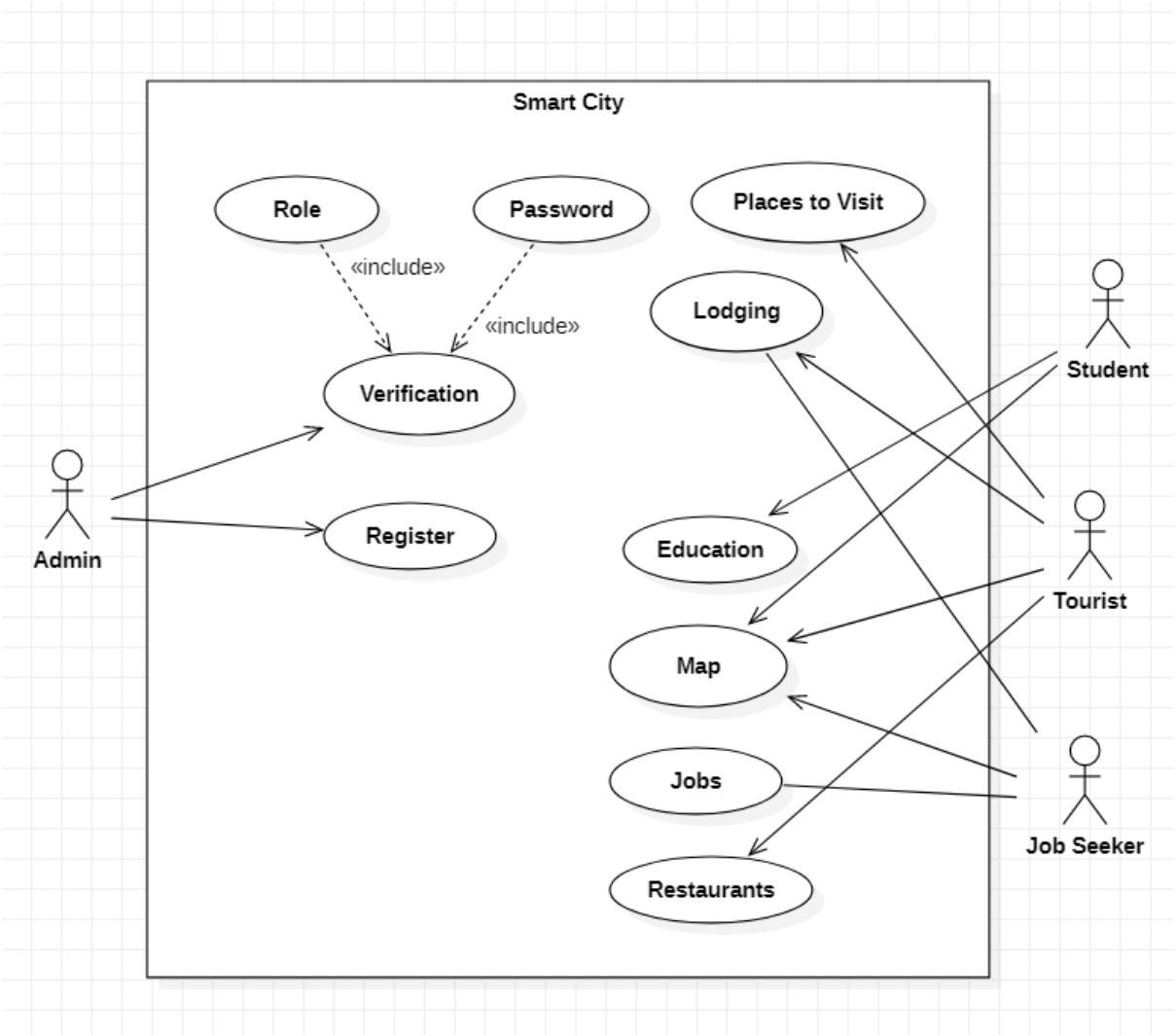
1) Lodging
2) Maps
3) Jobs

We will store the login details in a MySQL database. The schema of the database is defined below. We use JDBC to connect the database to the frontend to retrieve or verify the requisite information.

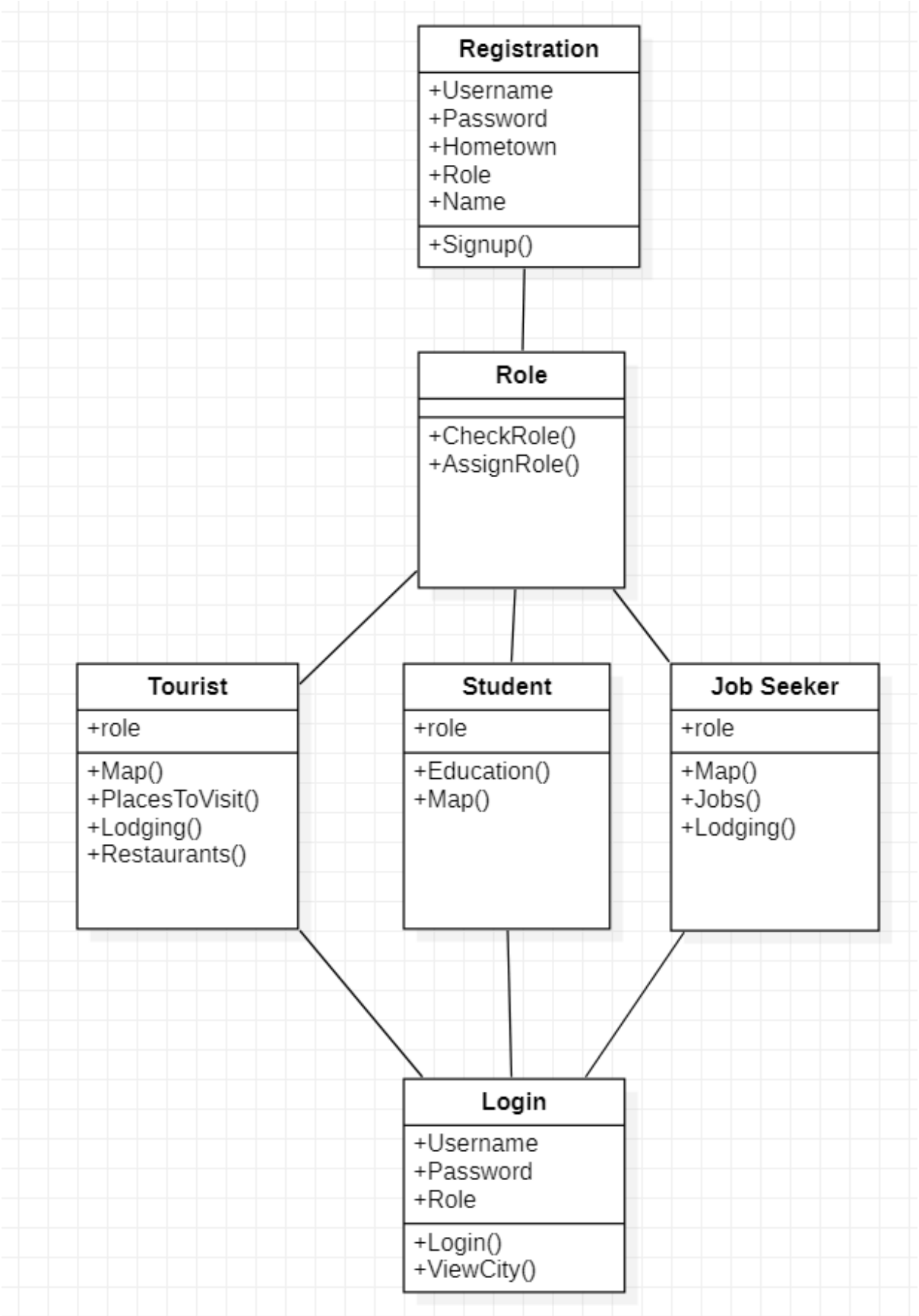
[Hari3008/Smart City-OOAD(github.com)]

2. Analysis and Design Models

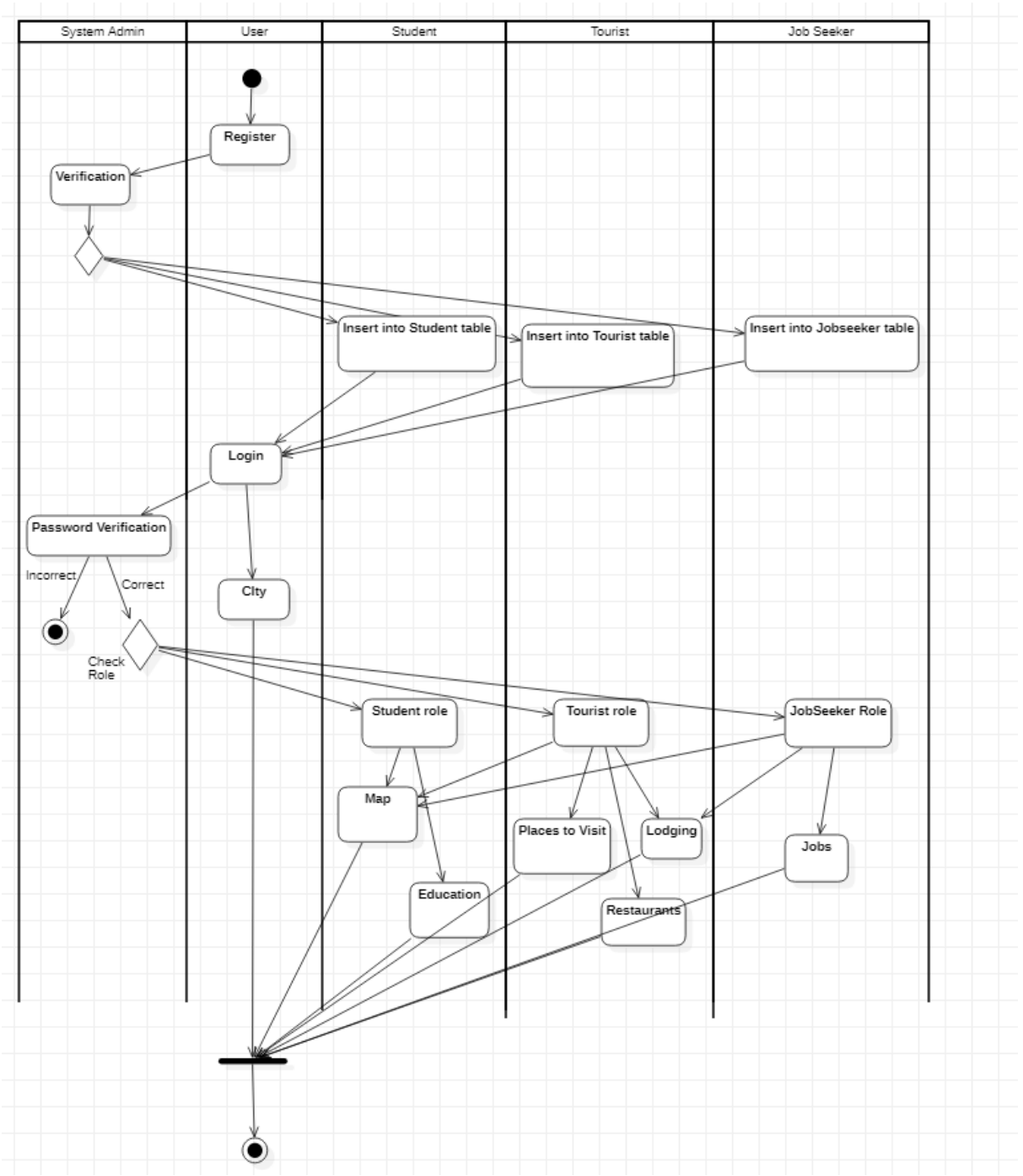
Use Case



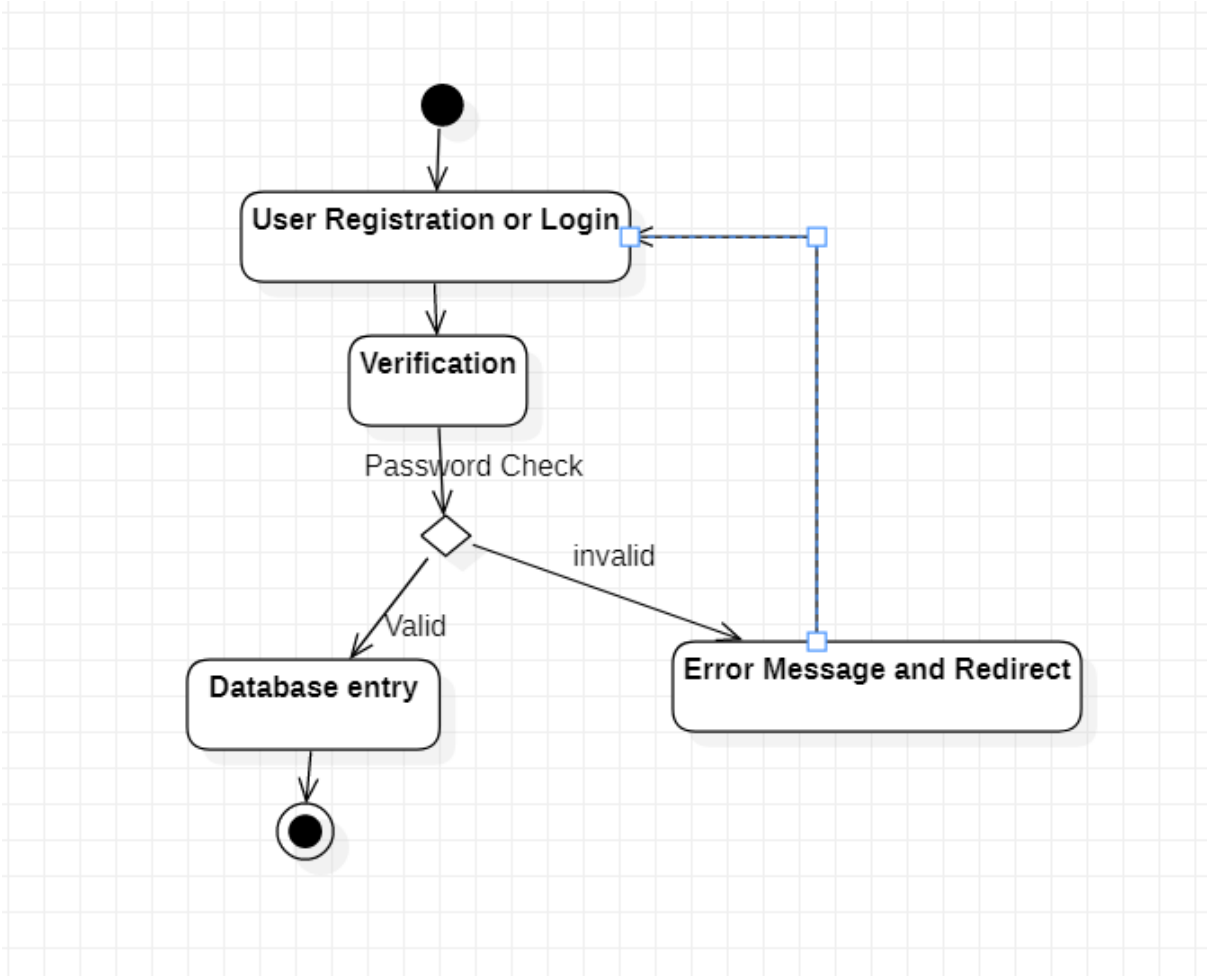
Class Diagram


Registration
+Username
+Password
+Hometown
+Role
+Name
+Signup()
Role
+CheckRole()
+AssignRole()
Tourist
+role
+Map()
+PlacesToVisit()
+Lodging()
+Restaurants()
Student
+role
+Education()
+Map()
Job Seeker
+role
+Map()
+Jobs()
+Lodging()
Login
+Username
+Password
+Role
+Login()
+ViewCity()

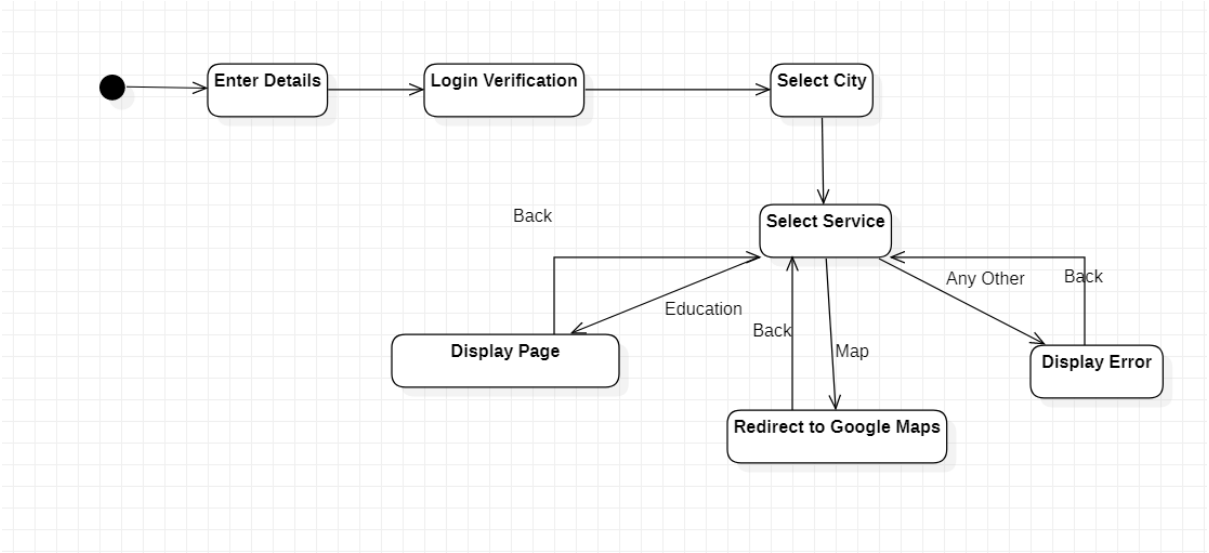
Activity Diagram :



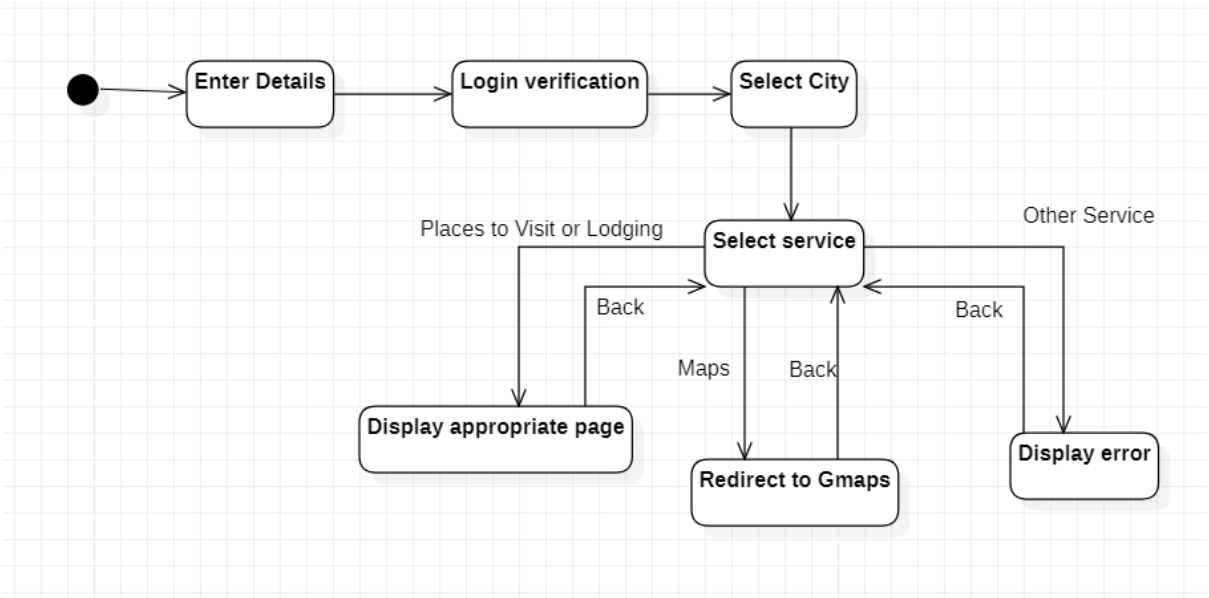
User Registration/Login State Diagram:



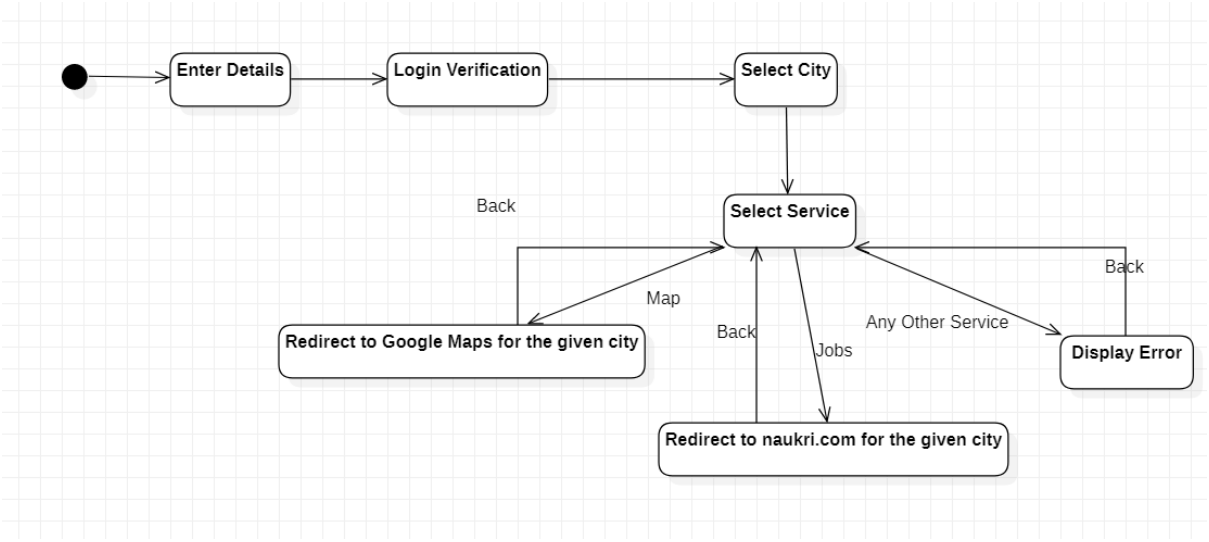
Student Role State Diagram



Tourist Role State Diagram



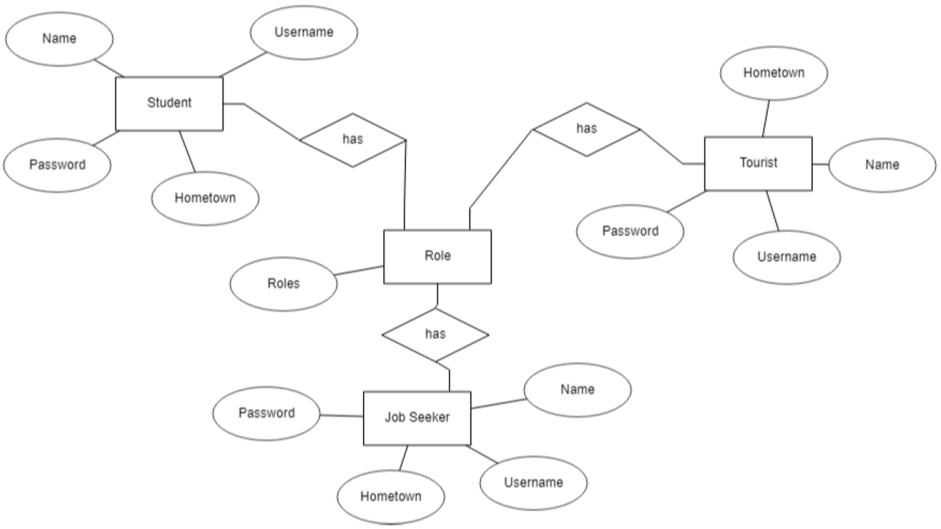
Job Seeker State Diagram



3. Tools and Frameworks Used:

- Eclipse IDE for Java Developers
- MySQL Database
- Java Swing
- Java AWT
- Java Animation Class

ER diagram:



Relational Schema:

Student Table:

Name	Varchar(30)
Username	Varchar(30)
Role	Varchar(30)
Password	Varchar(30)
Hometown	Varchar(30)

Tourist Table:

Name	Varchar(30)
Username	Varchar(30)
Role	Varchar(30)
Password	Varchar(30)
Hometown	Varchar(30)

Job Seeker:

Name	Varchar(30)
Username	Varchar(30)

Role	Varchar(30)
Password	Varchar(30)
Hometown	Varchar(30)

4. Design Principles and Design Patterns Applied

Design Principles used:

1. **Open Close Principle:**
   The abstract class Imageshow.java is open for extension but closed for modification. It has methods which decide the orientation of the image which is used by almost all classes in the project. Any change in the view or orientation of the image requires an extension by adding a method in this class and the changes will be reflected appropriately by the classes that derive this class.

```
package smartcitymini;

import java.awt.Image;

abstract class Imageshow{
    public void SetImageSizeCity(JLabel back)
    {
        ImageIcon icon=new ImageIcon(".//img//bg13.jpg");
        Image img=icon.getImage();
        Image newImg=img.getScaledInstance(back.getWidth(), back.getHeight(), Image.SCALE_SMOOTH);
        //Image newImg=img.getScaledInstance(500, 500, Image.SCALE_SMOOTH);
        ImageIcon newImc=new ImageIcon(newImg);
        back.setIcon(newImc);

    }
    public void SetImageSizeForCity(JLabel back,ImageIcon im)
    {
        ImageIcon icon=im;
        Image img=icon.getImage();
        Image newImg=img.getScaledInstance(back.getWidth(), back.getHeight(), Image.SCALE_SMOOTH);
        //Image newImg=img.getScaledInstance(500, 500, Image.SCALE_SMOOTH);
        ImageIcon newImc=new ImageIcon(newImg);
        back.setIcon(newImc);

    }

}
```

2. **Single Responsibility Principle:**
   Each class has a single responsibility like Jobs.java redirects a particular city's job page in the browser, Places.java creates a page for Places to visit for any city, Education.java to display educational institutions for any city and so on.

```java
package smartcitymini;


import java.awt.Desktop;

public class Jobs {
    public Jobs(String url, String cityname) {
        if (Desktop.isDesktopSupported() && Desktop.getDesktop().isSupported(Desktop.Action.BROWSE)) {
            try {
                Desktop.getDesktop().browse(new URI(url));
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (URISyntaxException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

2 Design Patterns used:

- **Singleton -**

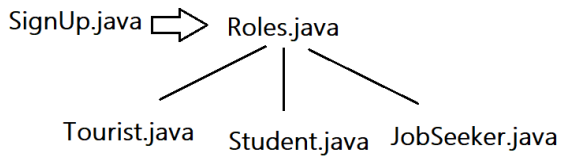
  Need for singleton: A single instance to the database has to be created initially and this instance has to be referred from then.

  Implemented in the file: Login.java

```java
    private static DbConnect db;
    /**
     * Create the frame.
     */
    public login() {
        setTitle("Login");
        if(db==null) {
            DbConnect db=new DbConnect();
        }
        connect=db.dataConnector();
```

- **Factory -**

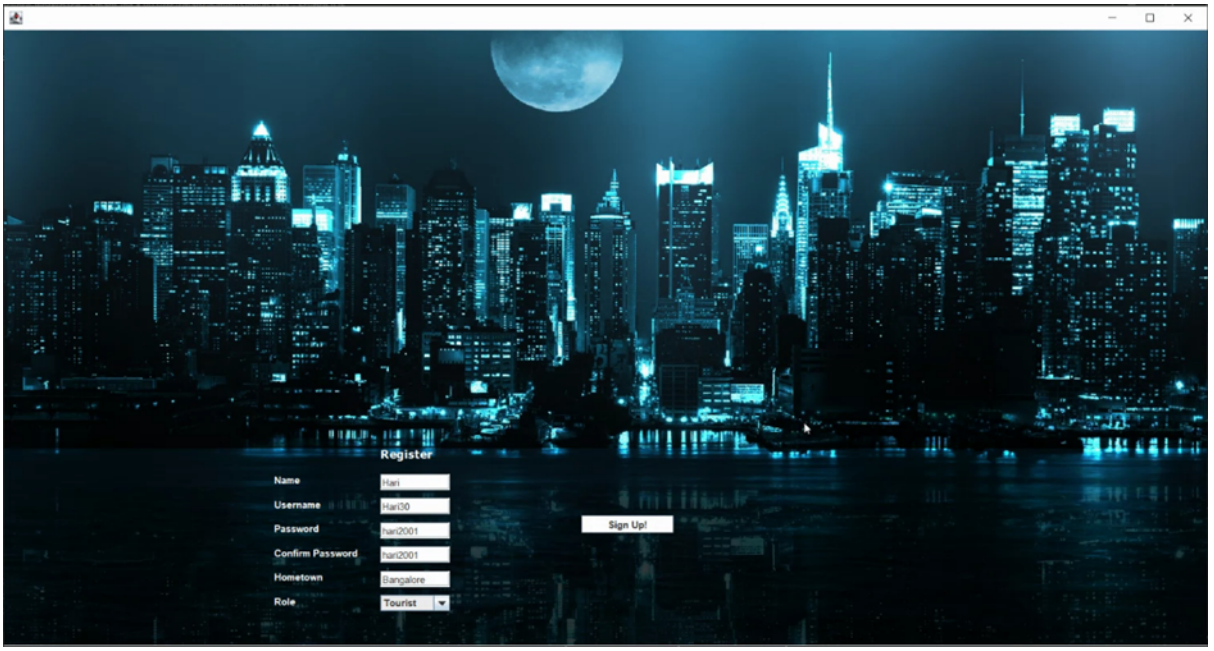
Need for Role: The class SignUp calls the class Role which decides which class to call out of Student(), Tourist(), JobSeeker() based on the parameter passed to Role dynamically.
Flow of control in files:



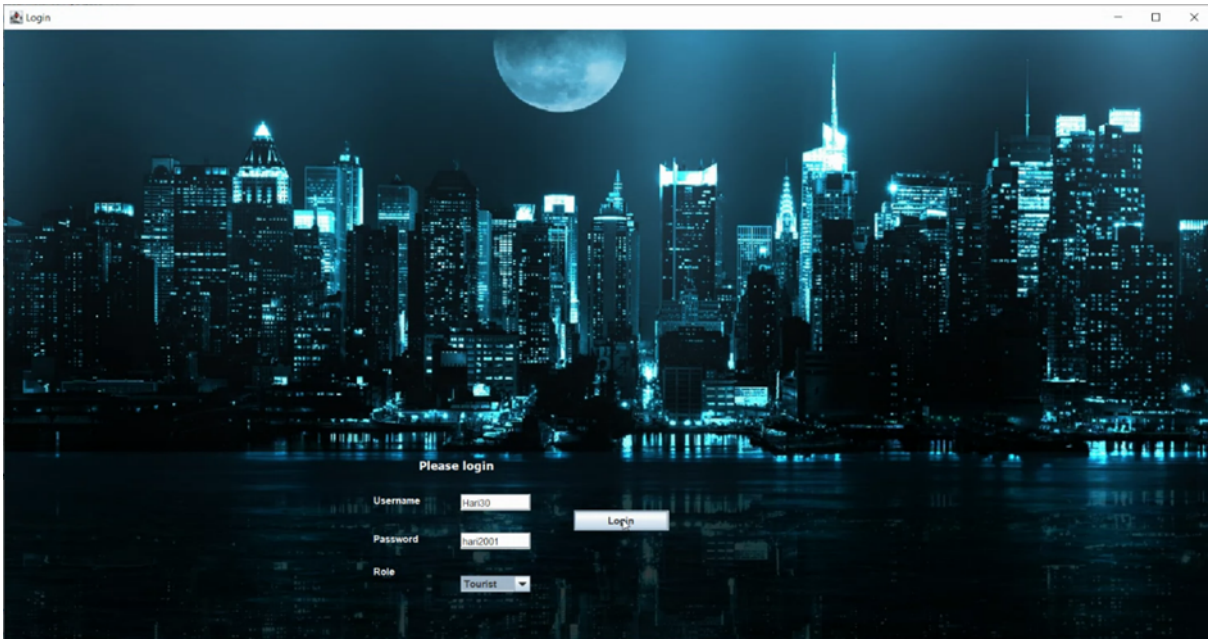
```java
package smartcitymini;

public class Roles {
    Roles(String Name, String Username, String Password,String HomeTown, String Role ){
        if(Role == "Job Seeker"){
            JobSeeker jb = new JobSeeker(Name, Username, Password, HomeTown);
        }
        if(Role == "Student"){
                Student st = new Student(Name, Username, Password, HomeTown);
        }
        if(Role == "Tourist"){
            Tourist to = new Tourist(Name, Username, Password, HomeTown);
        }
    }
}
```

5. Application Screenshots (3-4 important pages)

Screenshot 1 - Registration Page

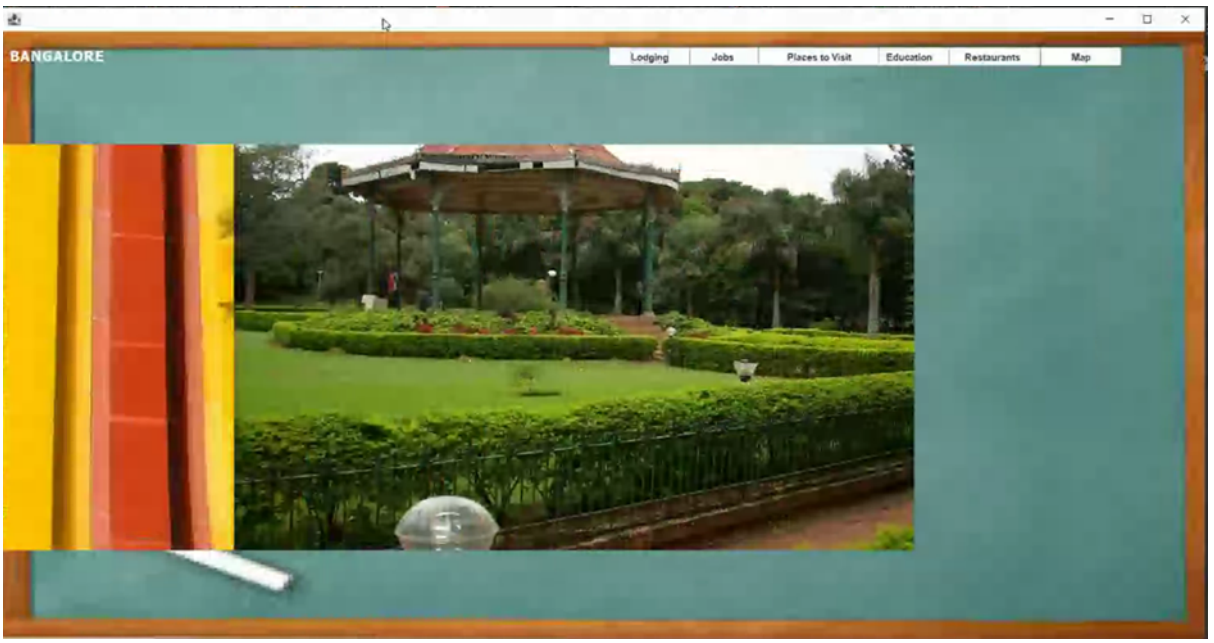


Screenshot 2 - Login Page

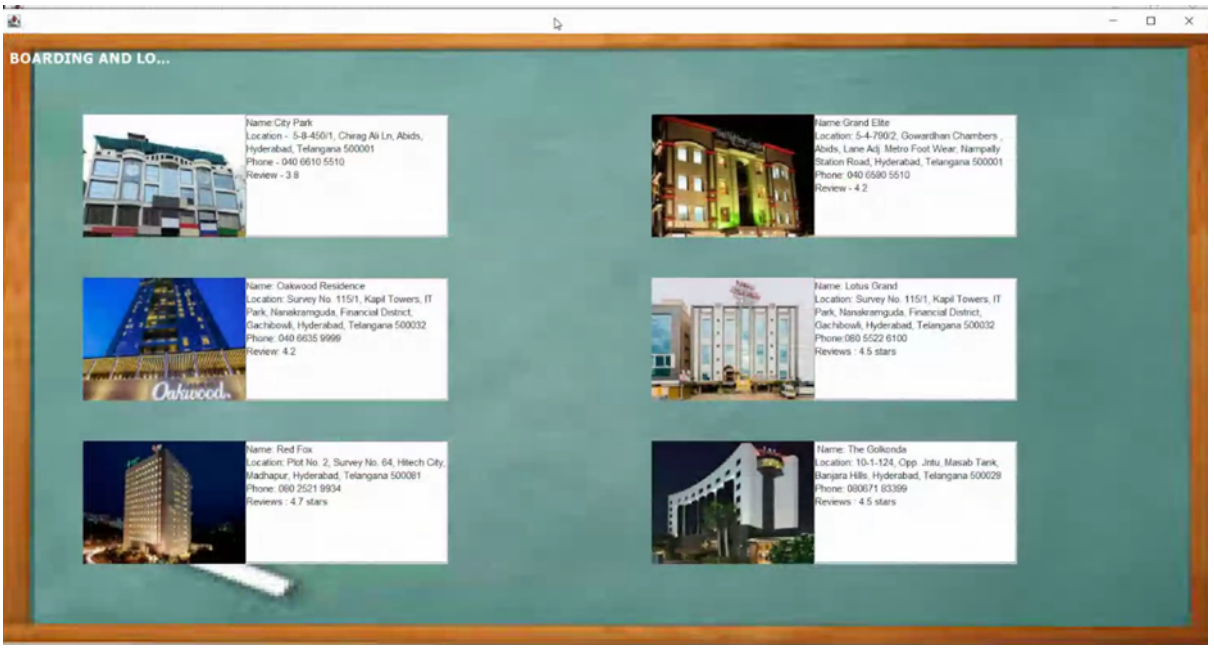


Screenshot 3a - Page to select a city

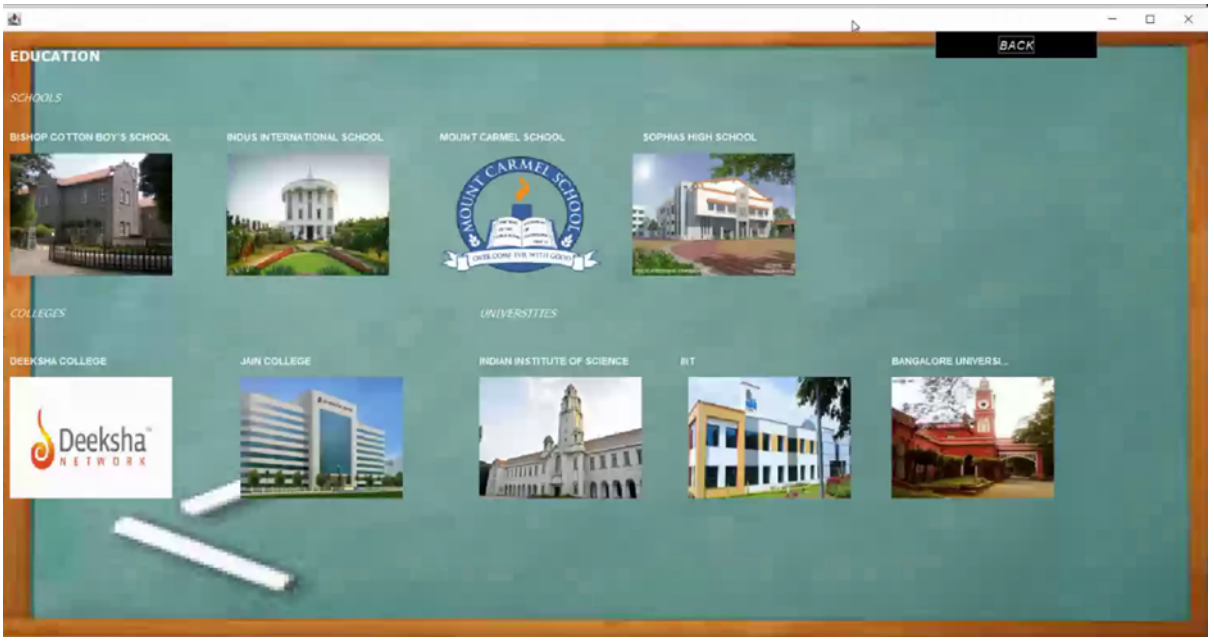
Screenshot 3b - Each city’s page



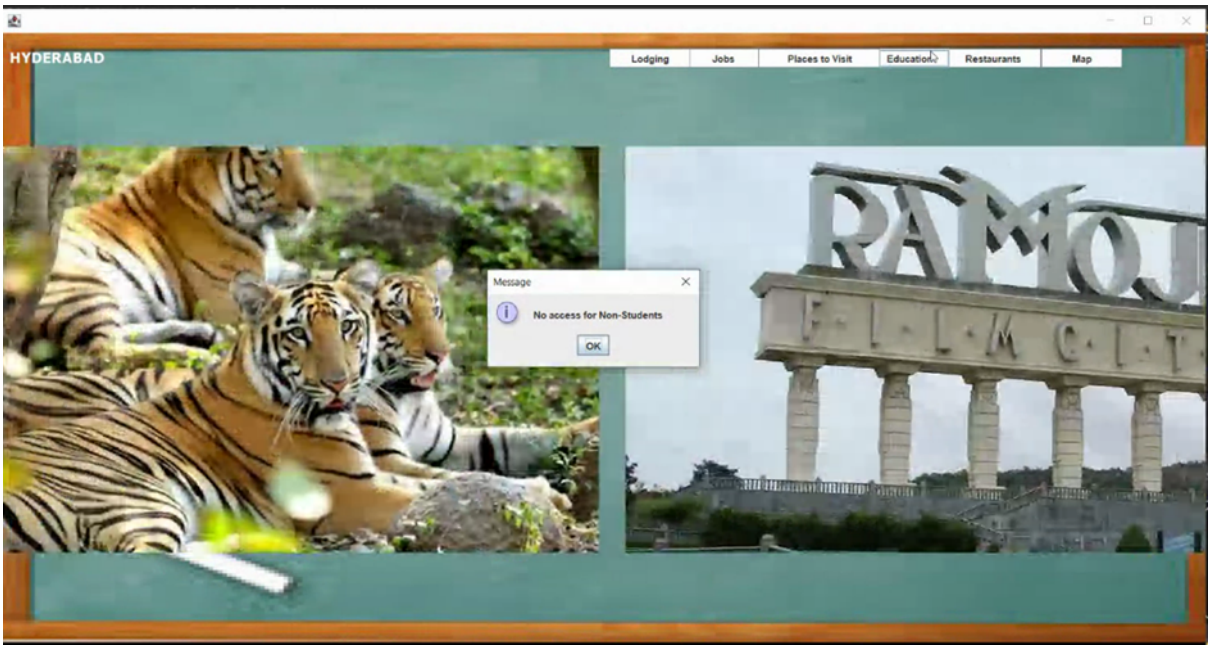
Screenshot 4 - Lodging(access granted)



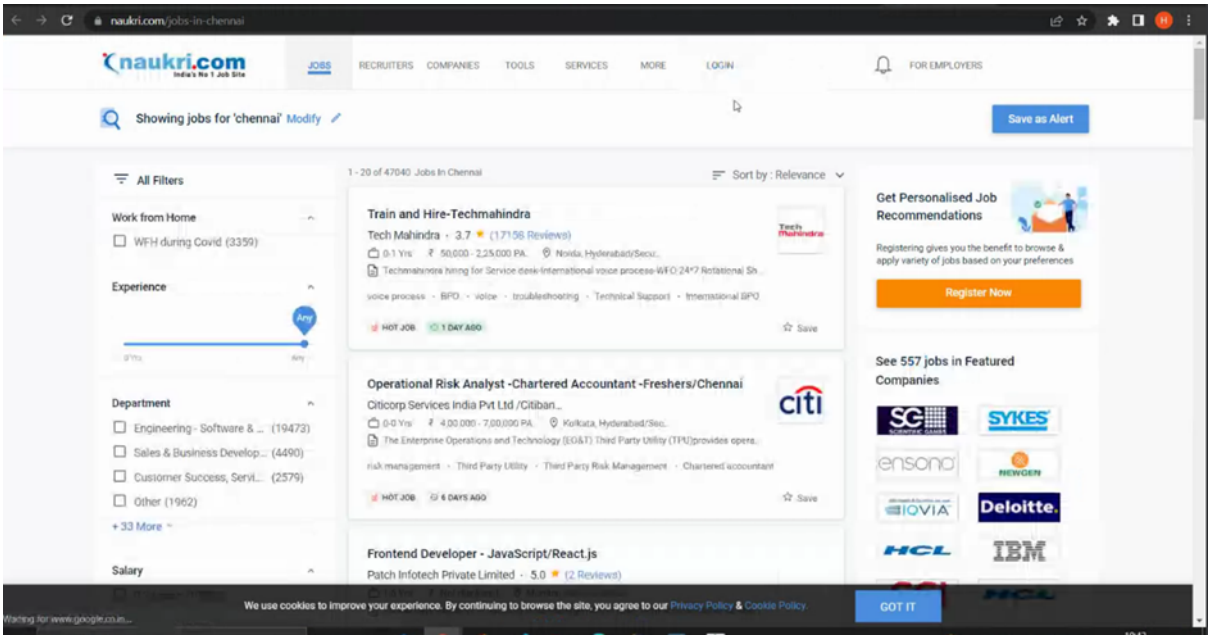
Screenshot 5a - Education(access granted)



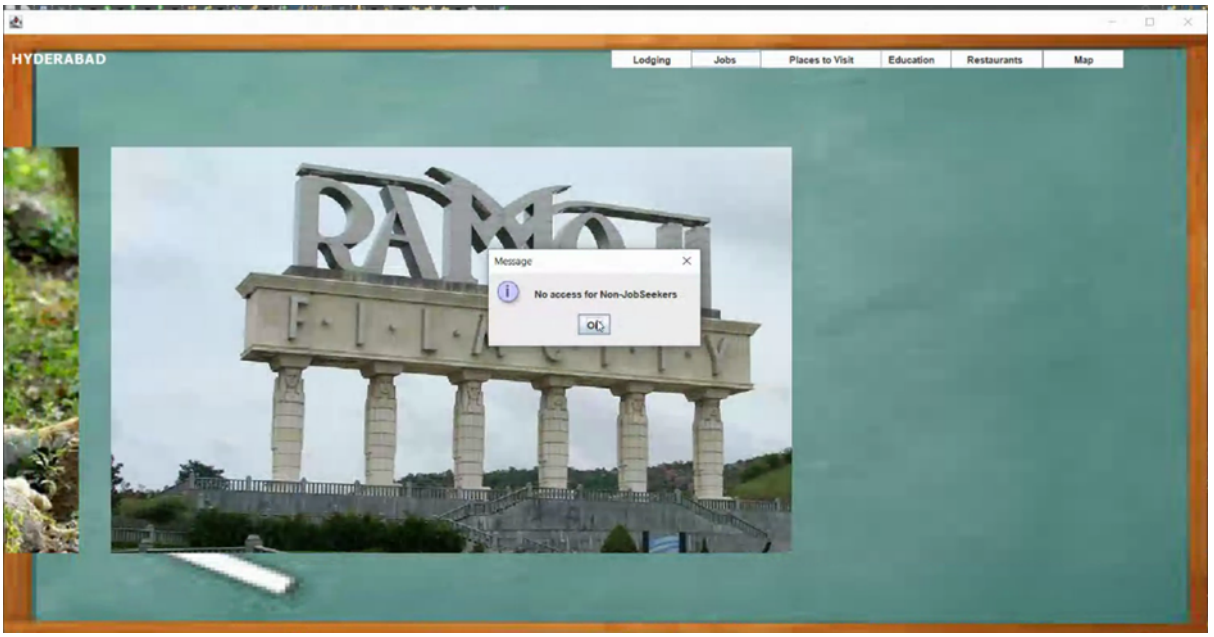
Screenshot 5b - Education(access denied)



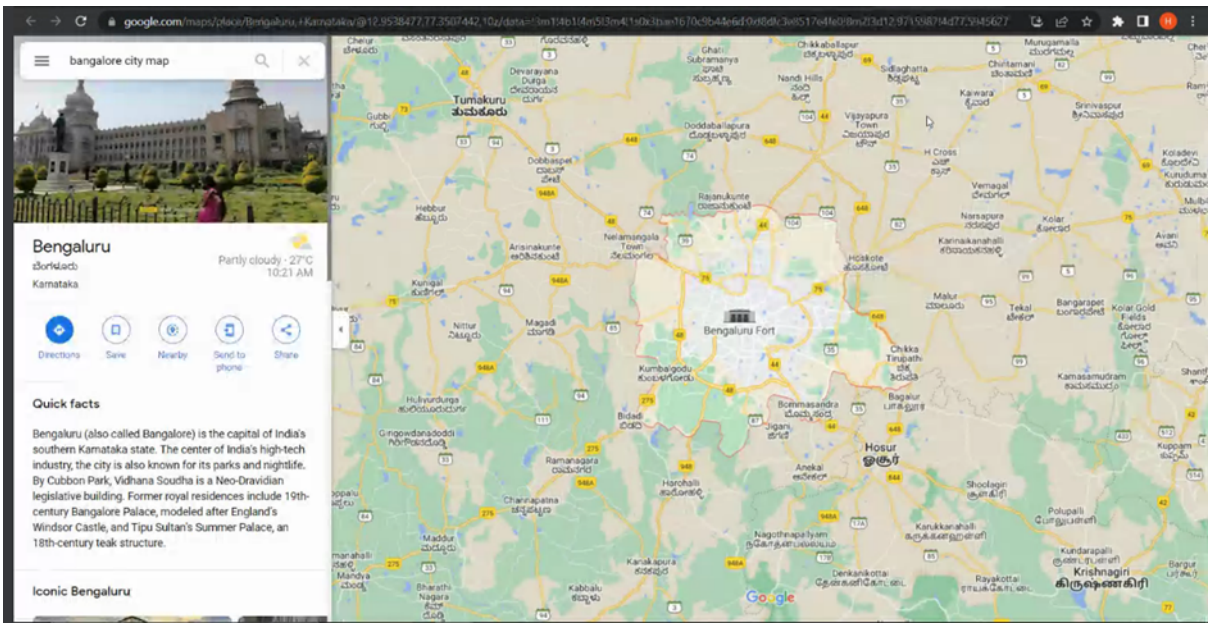
Screenshot 6a - Jobs(access granted)



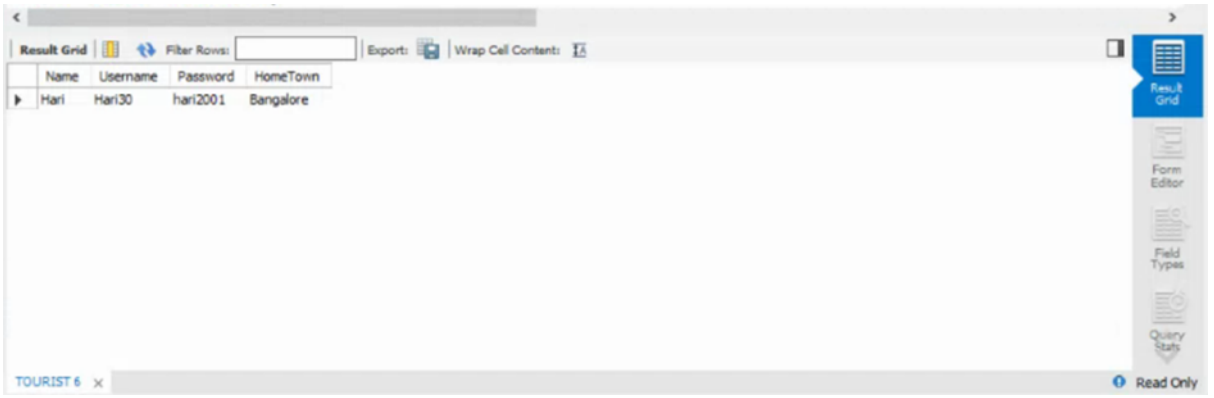
Screenshot 6b - Jobs(access denied)



Screenshot 7 - Maps



Screenshot 8 - Database



6. Team member contributions

Guhan K	City Pages, Report
Hanuraag Baskaran	Database Creation, Report
Hariharasudan S A	Login, 3 Use cases, Report
Harshita Vidapanakal	Signup, 3 Use cases, Report