**CHAPTER-1**

**ABOUT THE COMPANY**

* 1. **Company Profile**

**Bharat Electronics Limited (BEL)** is an Indian Government-owned aerospace and defence electronics company. It primarily manufactures advanced electronic products for ground and aerospace applications. BEL is one of nine PSUs under the Ministry of Defence of India. It has been granted “*Navratna”* status by the Government of India.

* 1. **Vision and Mission**
* **Vision :** To be a world-class enterprise in professional electronics.
* **Mission :** To be a customer focused, globally competitive company in defence electronics and in other chosen areas of professional electronics, through quality, technology and innovation.
  1. **Products and Services**

**1.3.1 Products**

BEL designs, develops and manufactures a range of products in the following fields:

1. Electronic voting machines
2. Missiles
3. Semi-conductors
4. Radars
5. Traffic lights

**1.3.2 Services**

BEL produces a wide range of equipment for areas such as defence communication, radars, naval systems, c4i systems, weapon systems, homeland security, telecom & broadcast systems, electronic warfare, tank electronics, electro-optics, professional electronic components and solar photovoltaic systems.

BEL also aims to increase its non-defence share in the overall business. Some of the areas the company is focusing on, include solutions for the Civil Aviation sector, Anti Drone Systems, Satellite Assembly & Integration, Solar, Railway & Metro solutions, Network & Cyber Security solutions, Energy Storage products for Electric Vehicles, Homeland Security & Smart Cities, Medical Electronics, Artificial Intelligence, etc.

It manufactures state-of-the-art electronic products and systems for the Army, Navy and the Air Force.

* 1. **Awards and Accomplishments**
* IETE Corporate Award- for performance in Development of Software (Large Enterprises) -2021. Two BEL scientists received Young Scientist awards.
* 12 Public Relations Council of India (PRCI) awards (2 Gold, 4 Silver, 3 Bronze and 3 Consolation Awards) - Best Corporate Event, Health Care Communication Films, Government Communication Films, Motivational Films, House Journal – Print (English), Annual Report, Best Corona Awareness Programme, Corporate Film, Human Resource Management during Covid-19, Covid Management in Public Sector, Best Use of Media Relations, Best Use of Content.
* Product Development & Innovation Center (PDIC) at Bengaluru has been certified for implementation of Information Security Management System in accordance with ISO IEC 2001:2013 by TUV SUD for the Scope of Design & Development of Products, Systems and Sub-systems for various Technologies and support functions.
* International Aerospace & Defence Awards - BEL won Outstanding Contribution to the Aerospace Industry Award.
* PSE Excellence Award 2019 for Corporate Governance awarded by the Indian Chamber of Commerce (ICC).

**CHAPTER – 2**

**INTRODUCTION**

**2.1 Internship Title**

“Public grievance redressal module for e-Governance ”.

**2.2 Category**

WEB Based Software module.

**2.3 Overview**

The Public Grievance Redressal Portal is an online platform designed to facilitate the submission, processing, and resolution of public complaints. The portal provides a user-friendly interface for individuals to register, login, and report grievances they encounter in their daily lives. The system incorporates three key roles: users, Grievance Routing Officers (ROs), and Grievance Redressal Officers (GROs), each with distinct responsibilities and access levels.

**2.4 Objectives of Internship**

**2.5 Purpose of Internship**

**2.6 Scope of Internship**

**CHAPTER-3**

**SOFTWARE REQUIREMENT SPECIFICATION**

**3.1 Product Perspective**

1. **User Perspective**: The portal offers an intuitive interface for users to easily register, log in, and submit complaints. It enables users to track the progress of their complaints, ensuring transparency and keeping them informed throughout the resolution process.
2. **Grievance Routing Officer (RO) Perspective**: The portal provides ROs with a dedicated platform to efficiently review and process assigned complaints. It helps them manage and prioritize complaints, communicate with users, and take necessary actions for resolution.
3. **Grievance Redressal Officer (GRO) Perspective**: GROs have a comprehensive view of all complaints and user details. They can monitor the entire grievance redressal process, update complaint statuses, and take appropriate actions to resolve grievances promptly.
4. **Administrative Perspective**: The portal offers administrative functionalities for managing user accounts, roles, and system configuration. Administrators can generate reports and analytics for monitoring performance, data integrity, and system maintenance.
5. **Organizational Perspective**: The portal enables organizations to demonstrate their commitment to addressing public grievances effectively. It facilitates transparent and accountable complaint management, helping build trust, enhance reputation, and strengthen public confidence.

**3.2 Product Functions**

Product Functions for the Public Grievance Redressal Portal:

**User Functions**:

* User Registration: Users can create an account by providing their email, password, and other necessary details.
* User Login: Registered users can securely log in to access the portal.
* Complaint Submission: Users can submit complaints by providing details such as description, date, images (if applicable), and location.
* Complaint Tracking: Users can track the status and progress of their submitted complaints.
* Communication: Users can communicate with Grievance Routing Officers (ROs) regarding their complaints and receive updates.

**Grievance Routing Officer (RO) Functions:**

* Complaint Assignment: ROs can review and assign complaints to themselves for further processing.
* Complaint Processing: ROs can manage and process assigned complaints, including reviewing details, gathering additional information if needed, and taking appropriate actions.
* Communication: ROs can communicate with users to seek clarifications, provide updates, or request additional information related to the complaints.
* Complaint Escalation: If a complaint requires the attention of a higher authority, ROs can escalate it to the Grievance Redressal Officer (GRO).

**Grievance Redressal Officer (GRO) Functions:**

* Complaint Monitoring: GROs have access to all complaints and their statuses in the system, allowing them to monitor the overall progress.
* Complaint Resolution: GROs can review and take necessary actions to resolve complaints, ensuring prompt redressal.
* Communication: GROs can communicate with both users and ROs to provide instructions, updates, or resolutions related to the complaints.
* Status Update: GROs have the authority to update the status of complaints as they progress through the resolution process.

**Administrator Functions:**

* User Management: Administrators can manage user accounts, including creation, modification, and deletion.
* Role Management: Administrators can assign roles (e.g., user, RO, GRO) to registered users based on their responsibilities.
* System Configuration: Administrators can configure system settings, such as email notifications, complaint categories, or priority levels.
* Reporting and Analytics: Administrators can generate reports and analytics to gain insights into complaint trends, system performance, and user engagement.

**Organizational Functions:**

* Reputation Management: The portal helps organizations demonstrate their commitment to resolving public grievances promptly and efficiently, enhancing their reputation.
* Performance Evaluation: The system provides data and metrics for organizations to evaluate their performance in handling and resolving complaints.
* Compliance and Accountability: The portal promotes transparency and accountability in the grievance redressal process, ensuring adherence to regulatory requirements and organizational policies.

These product functions collectively enable users, ROs, GROs, and administrators to effectively manage and resolve public grievances, ensuring transparency, accountability, and timely redressal of complaints.

**3.3 General Constraints**

The system follows the general constraints as below,

* All the fields are required.
* Password must be of 6 to 40 characters.
* Mail address be of some specific format.
* Normal user can not view the complaint list.
* Every user type has different functionalities, and that should be shown in their navigation bar.

**3.4 Special Requirements**

**3.4.1 Hardware Tools used**

* Processor : i3 or above
* Hard disk : 40GB
* Memory : 4GB RAM

**3.4.2 Software Tools used**

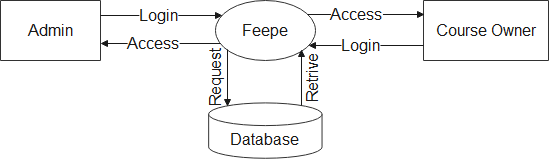
* Operating System : Windows 7 or above
* Web Server : Tomcat
* Front End : ReactJs
* Back End : Springboot
* Web browsers : Google chrome/Mozilla Firefox
* Database : MYSQL
* Tools : VS code, Eclipse

**CHAPTER-4**

**SYSTEM DESIGN**

**4.1 Context-Flow Diagram - CFD**

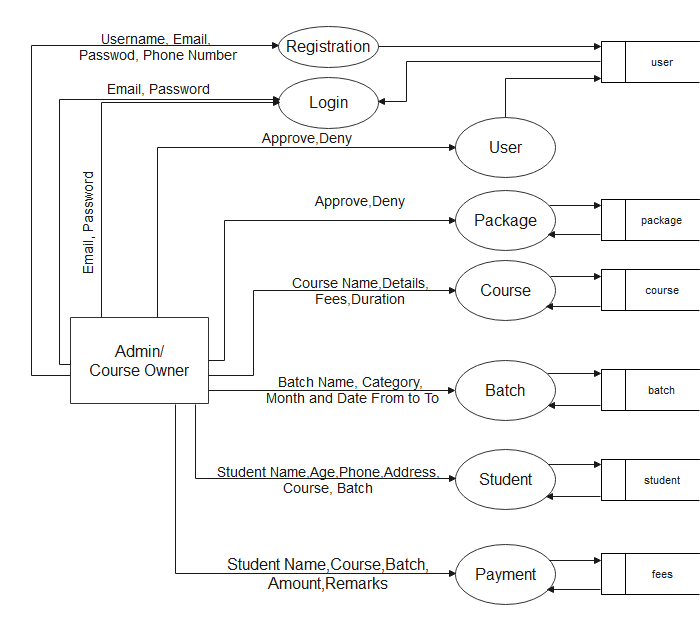
Context flow diagram is the Top-level data flow diagram (0 Level). It only contains the one process node that generalizes the functions of the entire system in relationship to external entities. With help of below the CFD diagram shows how the process of our project contains.

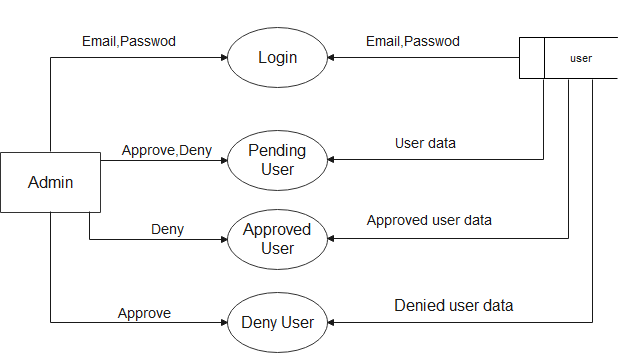


**4.2 Data-Flow Diagram- DFD (Level 1 and Level 2)**

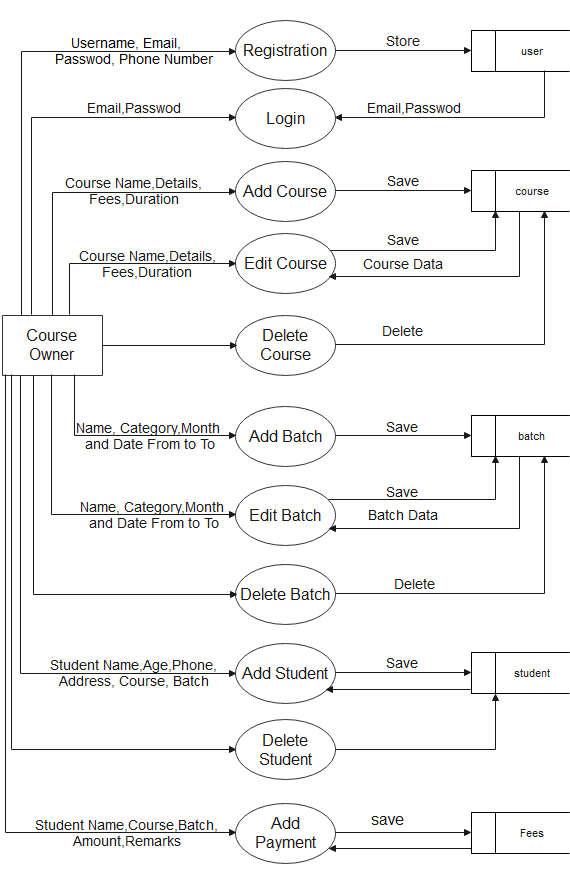
Data flow diagram (DFD) maps out the flow of the information for any process or system. With the help of this we can get to know what is the flow of our project and how it will be work. It shows the Input, output and each working flow of the project. And data flow diagram.

**4.2.1 DFD LEVEL 1**



**4.2.2 DFD LEVEL 2 ADMIN**

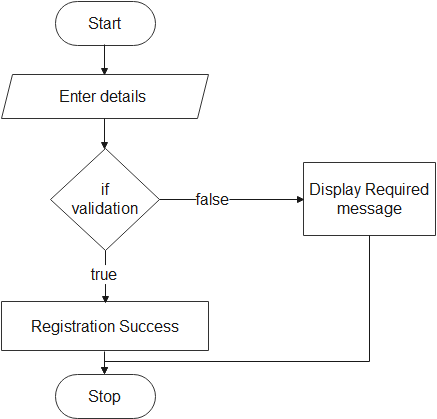
**4.2.3 DFD LEVEL 2 COURSE OWNER**



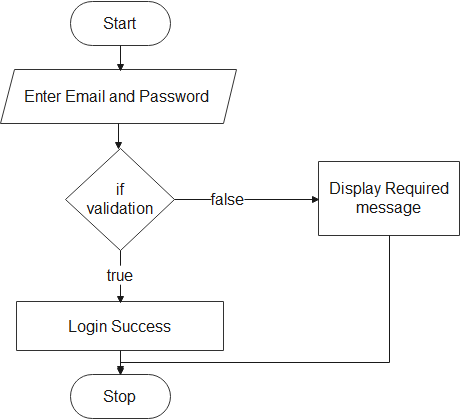
**CHAPTER-5**

**DETAILED DESIGN**

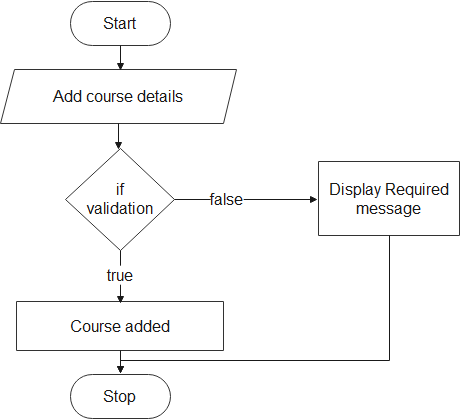
* 1. **Modular Decomposition of the system**
     1. **User Registration**
        1. **Inputs:** username, password, confirm password, email, phone number.
        2. **Procedural Details:**



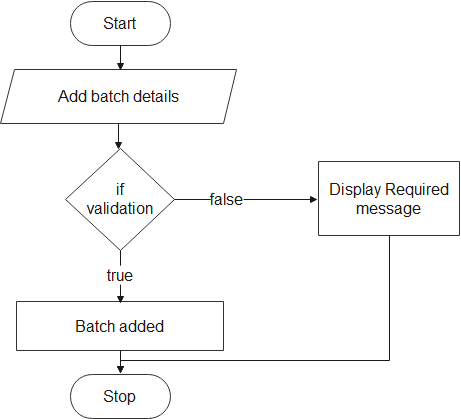
* + - 1. **Outputs:** Registration done and added to the database.
    1. **User Login**
       1. **Inputs:** User Email id and password
       2. **Procedural Details:**



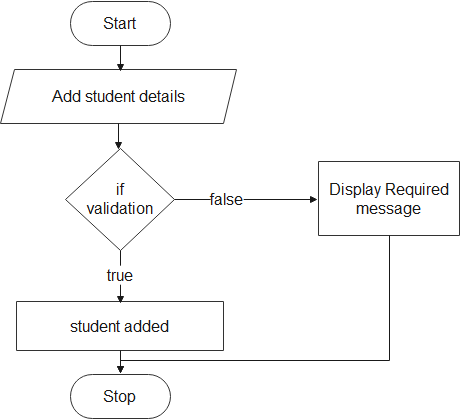
* + - 1. **Outputs:** Check validityand Login successful. Redirect to the homepage.
    1. **Add Course**
       1. **Inputs:** Course name, Course detail, fees and duration.
       2. **Procedural Details:**



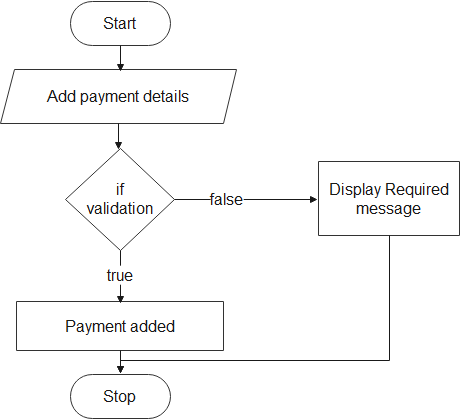
* + - 1. **Outputs:** Course added and stored in database.
    1. **Add Batch**
       1. **Inputs:** Batch name, course, date, time and batch details.
       2. **Procedural Details:**



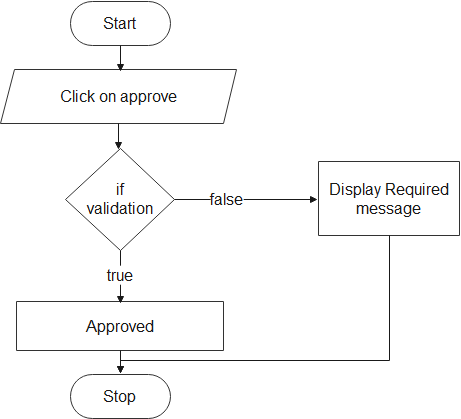
* + - 1. **Outputs:** Batch added and stored in database.
    1. **Add Student**
       1. **Inputs:** student name, age, phone, address batch and course.
       2. **Procedural Details:**



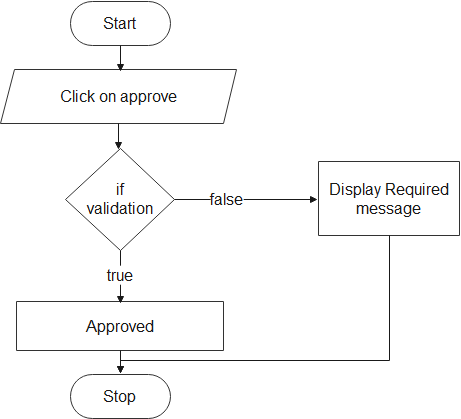
* + - 1. **Outputs:** Student added and stored in database.
    1. **Payment**
       1. **Inputs:** student name, batch, course and amount to pay and remark.
       2. **Procedural Details:**



* + - 1. **Outputs:** Payment detail update done and stored in database.
    1. **User approval**
       1. **Inputs:** Click on approve button.
       2. **Procedural Details:**



* + - 1. **Outputs:** User approved.
    1. **User deny**
       1. **Inputs:** Click on deny button.
       2. **Procedural Details:**



* + - 1. **Outputs:** User denied.

**CHAPTER-6**

**CODING**

* 1. **Database Connection**

|  |
| --- |
| spring.jpa.hibernate.ddl-auto=update |
|  |

|  |
| --- |
| spring.datasource.url=jdbc:mysql://localhost:3306/pub?useSSL=false |
|  |

|  |
| --- |
| spring.datasource.username=root |
|  |

|  |
| --- |
| spring.datasource.password=Mayura@1 |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver |
|  |

|  |
| --- |
| #spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.MySQL5InnoDBDialect |
|  |

spring.jpa.database-platform=org.hibernate.dialect.MySQL5Dialect

* 1. **Data Store/Retrieval/Update/Delete**
     1. **Data Store**

|  |
| --- |
| 1. package com.example.saaku.controller; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import java.util.List; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import org.springframework.beans.factory.annotation.Autowired; |
|  |

|  |
| --- |
| import org.springframework.web.bind.annotation.CrossOrigin; |
|  |

|  |
| --- |
| import org.springframework.web.bind.annotation.GetMapping; |
|  |

|  |
| --- |
| import org.springframework.web.bind.annotation.PostMapping; |
|  |

|  |
| --- |
| import org.springframework.web.bind.annotation.RequestBody; |
|  |

|  |
| --- |
| import org.springframework.web.bind.annotation.RestController; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import com.example.saaku.model.Complaints; |
|  |

|  |
| --- |
| //import com.example.saaku.model.User; |
|  |

|  |
| --- |
| import com.example.saaku.repository.ComplaintRepository; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @RestController |
|  |

|  |
| --- |
| @CrossOrigin(origins = "\*", maxAge = 3600) |
|  |

|  |
| --- |
| public class ComplaintController { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Autowired |
|  |

|  |
| --- |
| private ComplaintRepository complaintRepository; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @GetMapping("/complaints") |
|  |

|  |
| --- |
| List<Complaints> getAllComplaints(){ |
|  |

|  |
| --- |
| return complaintRepository.findAll(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @PostMapping("/addCom") |
|  |

|  |
| --- |
| Complaints newComplaint(@RequestBody Complaints newComplaint) { |
|  |

|  |
| --- |
| return complaintRepository.save(newComplaint); |
|  |

|  |
| --- |
| } |
|  |

}

* + 1. **Retrieval**

<?php

session\_start();

include("functions.php");

include("connection.php");

$user\_data=check\_login($con);

$uid=$user\_data['uid'];

$querry2="select \* from student where bid in(select bid from batch where cid in (select cid from course where uid ='$uid'))";

$result2=$con->query($querry2);

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=t, initial-scale=1.0">

<title>Document</title>

<style>

.btns{

background: #667db6; /\* fallback for old browsers \*/

background: -webkit-linear-gradient(to right, #667db6, #0082c8, #0082c8, #667db6); /\* Chrome 10-25, Safari 5.1-6 \*/

background: linear-gradient(to right, #667db6, #0082c8, #0082c8, #667db6); /\* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ \*/}

</style></head>

<?php

require('nav.php');

?>

* + 1. **Update**

<body>

<div class="row mt-5 mb-5 bg-warning" style="margin-left:20px ; margin-right:20px" >

<div class="col-6 "> <h3 class="float-end" style="font-family: Brush Script MT;font-size:3rem;" >Student</h3></div>

<div class="col-6 "><a class="nav-link btn btns w-25 mt-2 p-2 float-end mr-5 " style="color:white;"

href="student.php">Add Student <i class="fa fa-plus" aria-hidden="true"></i></a></div>

</div>

<div class="container mt-5">

<table class="table table-striped table-hover rounded border-light">

<tr style="text-align:center ;">

<th>ID</th>

<th>NAME</th>

<th>AGE</th>

<th>PHONE</th>

<th>ADDRESS</th>

<th>COURSE NAME</th>

<th>BATCH NAME</th>

<th>DELETE </th></tr>

<?php

session\_start();

error\_reporting(0);

include("connection.php");

$bid=$\_REQUEST['bid'];

if($\_SERVER['REQUEST\_METHOD'] == "POST")

{

$bname=$\_POST['bname'];

$day\_frm=$\_POST['day\_from'];

$day\_to=$\_POST['day\_to'];

$time\_frm=$\_POST['time\_from'];

$time\_to=$\_POST['time\_to'];

$details=$\_POST['details'];

$query2="select \* from batch where bname='$bname' and cid='$cno' and day\_from='$day\_frm' and day\_to='$day\_to' and time\_from='$time\_frm' and time\_to='$time\_to'";

if(mysqli\_num\_rows($con->query($query2))>0)

echo "ALready Exists";

else{

$query="update batch set bname='$bname',details='$details',day\_from='$day\_frm',day\_to='$day\_to',time\_from='$time\_frm',time\_to='$time\_to' where bid='$bid'";

if($con->query($query))

echo "UPDATED";

else

echo "NOT INSERTED";

} }

$query="select \* from batch where bid='$bid'";

if($result=$con->query($query)){

$row=mysqli\_fetch\_assoc($result);

$cno=$row['cid'];

?>

* + 1. **Delete**

<?php

foreach($result2 as $index){

?><tr> <td><?php echo $index['sid'];?> </td><td>

<?php echo $index['sname'];?> </td>

<td><?php echo $index['age'];?></td><td>

<?php echo $index['phone'];?></td>

<td><?php echo $index['address'];?></td> <td>

<?php

$bid=$index['bid'];

$query3="Select \* from batch where bid='$bid'";

$res3=$con->query($query3);

$row3=mysqli\_fetch\_assoc($res3);

$cid=$row3['cid'];

$query4="select \* from course where cid='$cid'";

$res4=$con->query($query4);

$row4=mysqli\_fetch\_assoc($res4);

echo $row4['cname'];?>

</td> <td><?php echo $row3['bname'];?></td><td><a class="btn btn-danger nav-link " href="student\_delete.php?sid=<?php echo $index["sid"]; ?>">Delete</a></td></tr<?php } ?>

</table></div>

<?php

require('footer.php'); ?></body></html>

**6.3 Data Validation**

<?php

session\_start():

include("connection.php");

$error\_msg="";

if($\_SERVER['REQUEST\_METHOD'] == "POST"){

$uname=$\_POST["name"];

$pass=$\_POST["pass"];

$rpass=$\_POST["rpass"];

$mail=$\_POST["email"];

$phone=$\_POST["phone"];

$query1="select \* from user where uname='$uname' or email='$mail'";

$res=$con->query($query1);

if(mysqli\_num\_rows($res) == 0){

if(!empty($uname) && !empty($pass) && !is\_numeric($uname)){

if(strlen($pass)>5){

if( preg\_match('([a-zA-Z].\*[0-9]|[0-9].\*[a-zA-Z])', $pass) ) {

if($pass==$rpass){

$query2="insert into user (uname,password,email,phone,account\_type) values ('$uname','$pass','$mail','$phone','guide')";

if($con->query($query2))

header("Location: login.php");

else

echo"NOT INSERTED";

}

}else $error\_msg="Password does not match";

}else $error\_msg="Password must be more than 5 characters!";

}else $error\_msg="Password must be alpha-numeric!";

}else $error\_msg="Invalid user name or password";

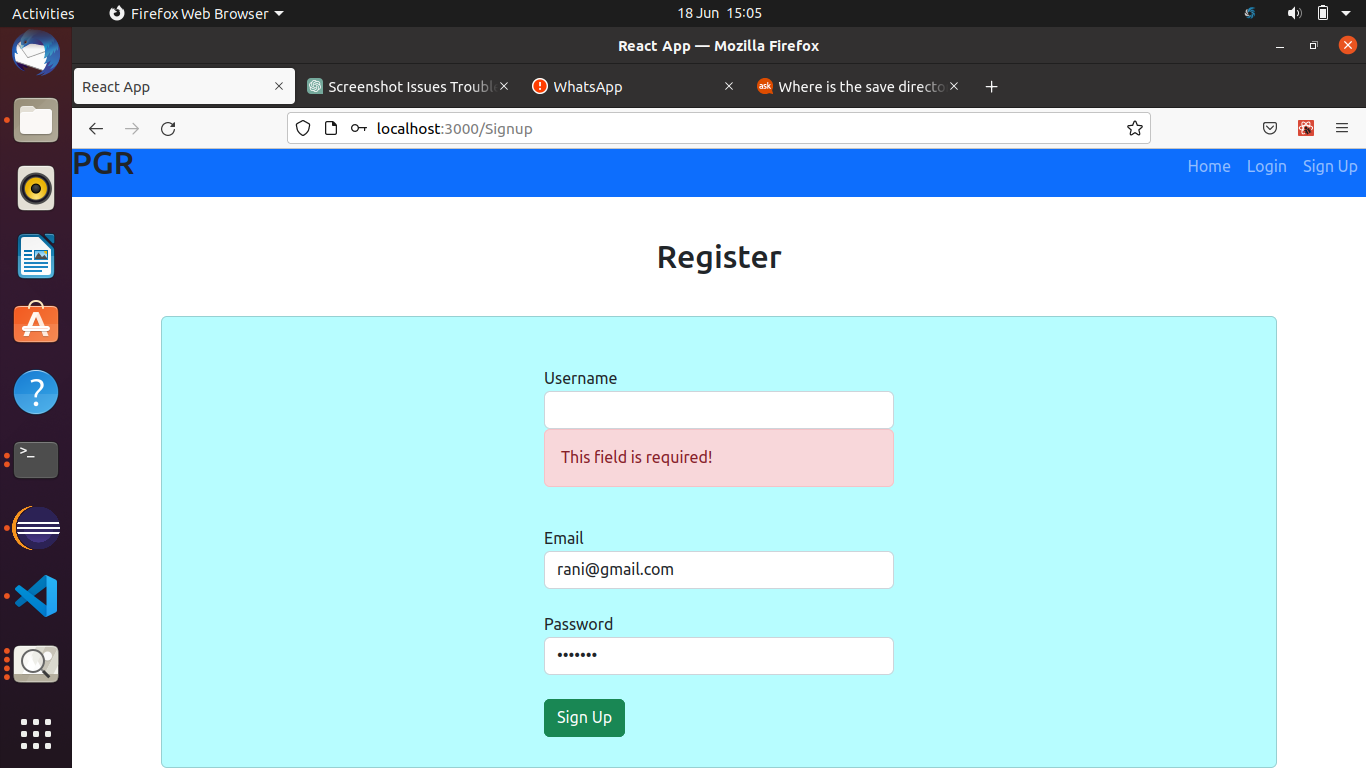
} ?>

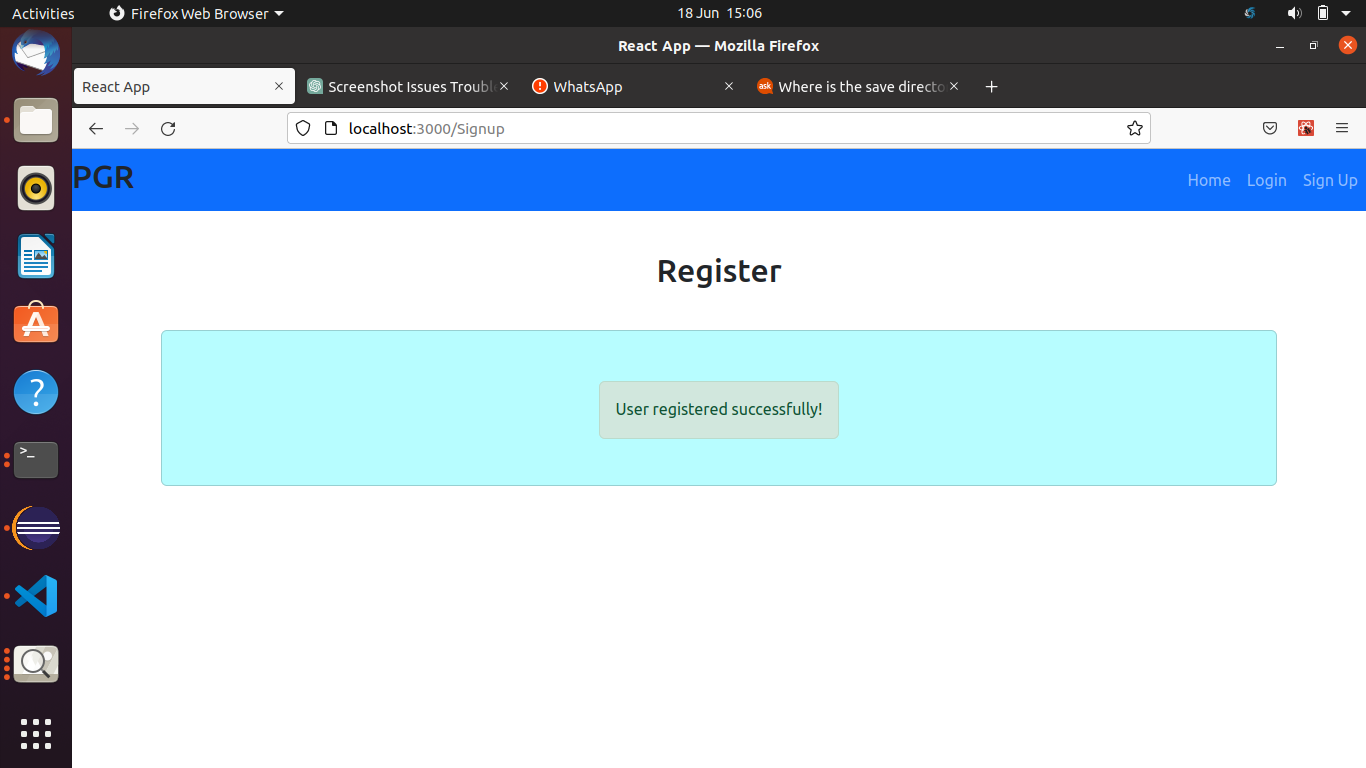
**CHAPTER-7**

**USER INTERFACE**

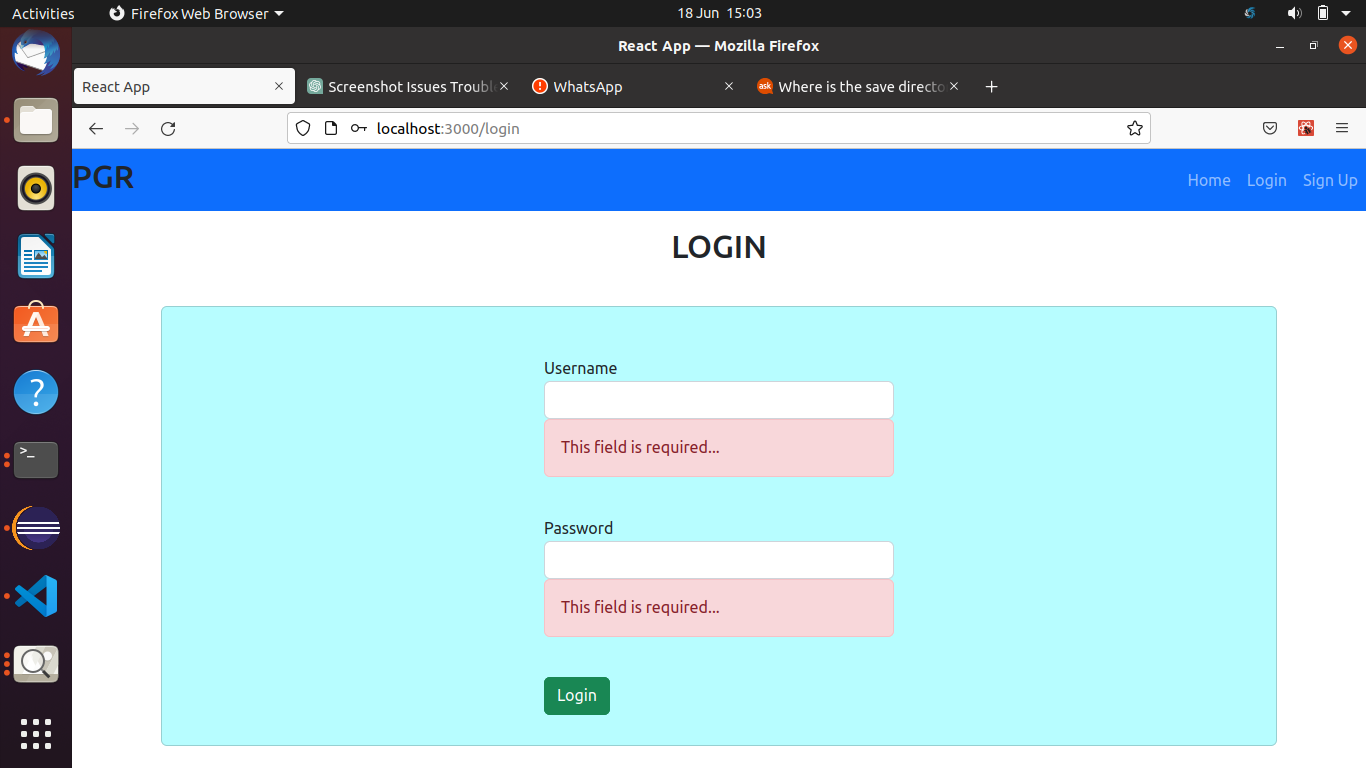
**7.1 Registration and Login Page**

* Using this registration page course owner can possible to create and account into the system. All the fields are mandatory, so that if any field left blank will lead to error generation.



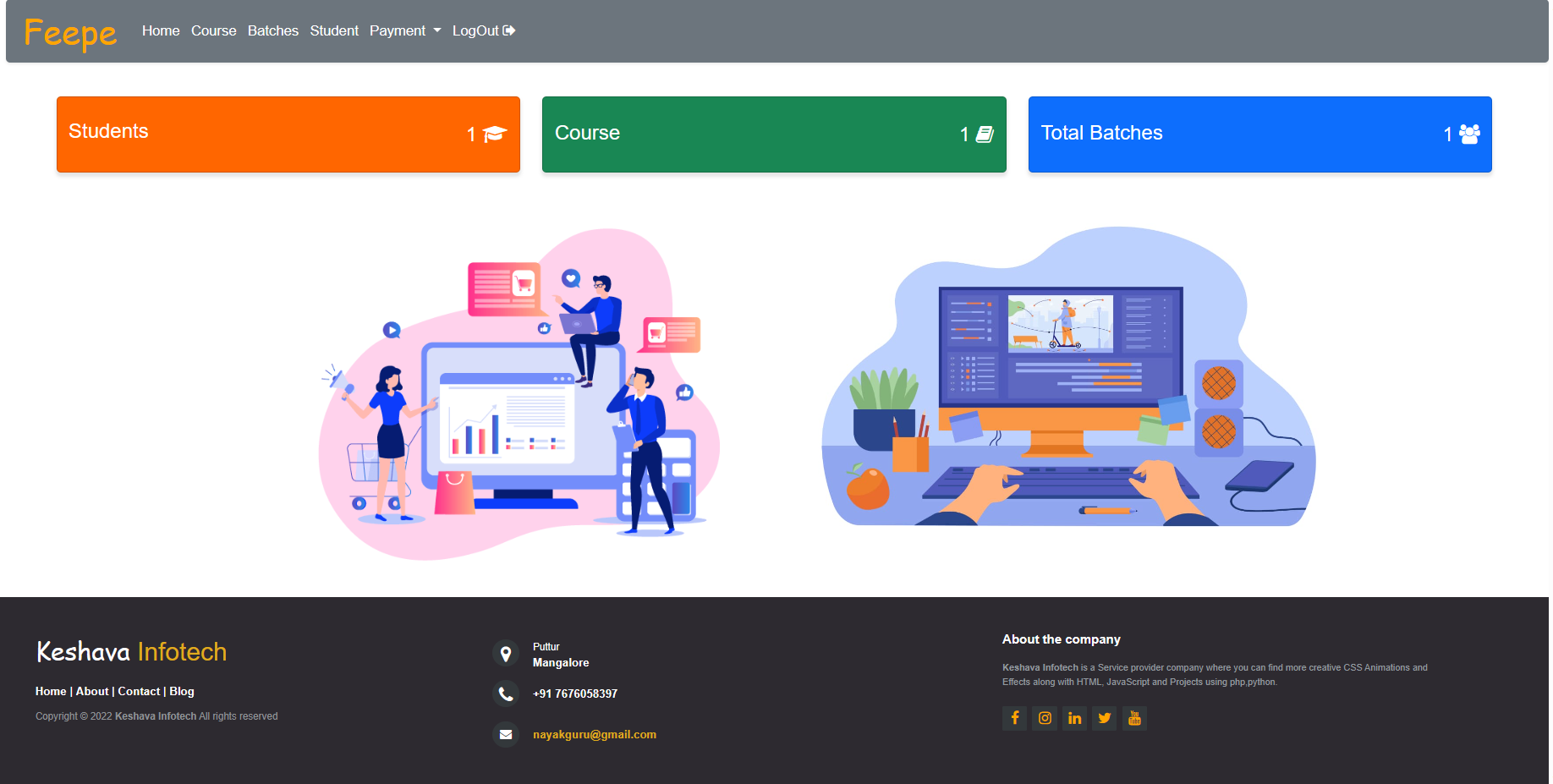


* Login page is used to authenticate the user. In this project, there are multiple roles are introduced, so that the authentication process will have very accurate and strong.

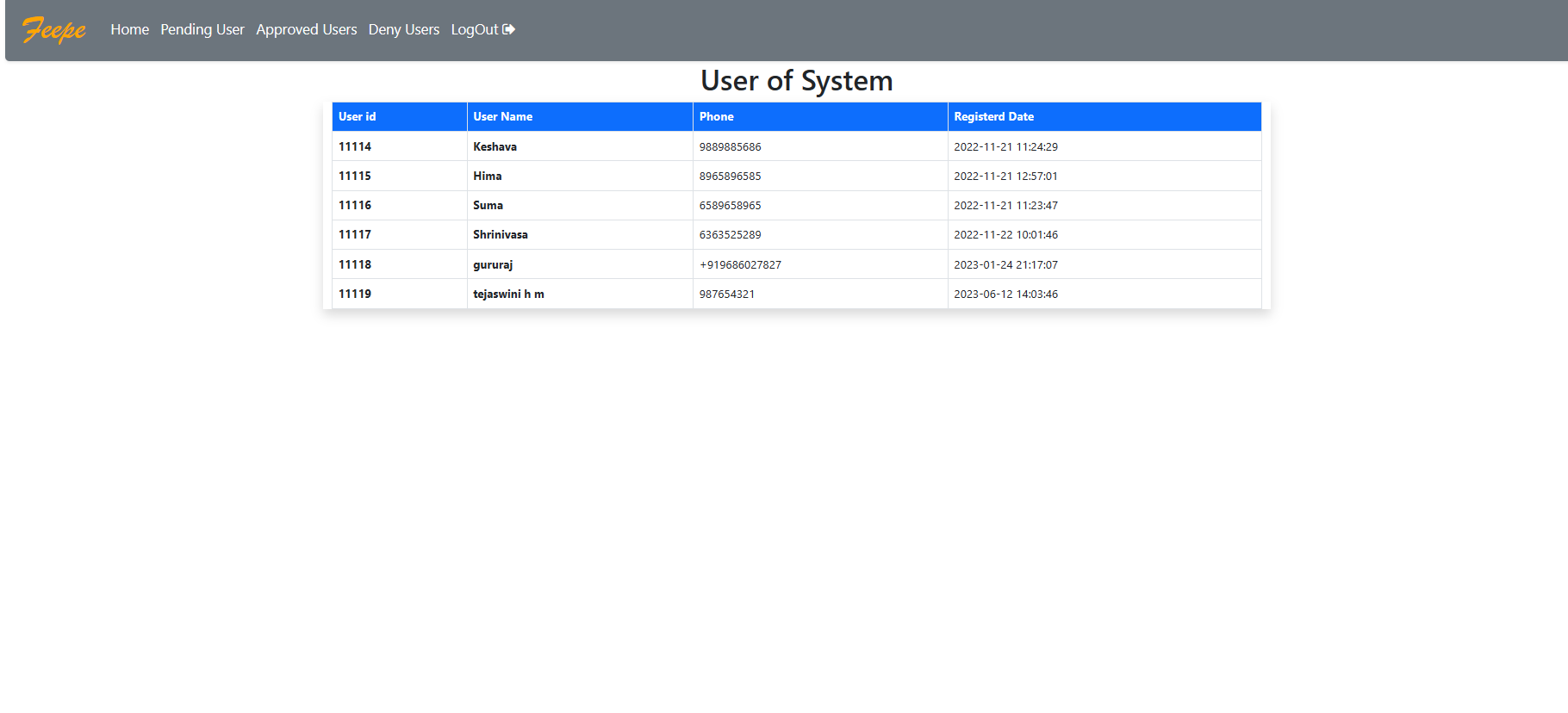
****

**7.2 Home page**

* Home page for the course owner.

****

* Home page for the admin.

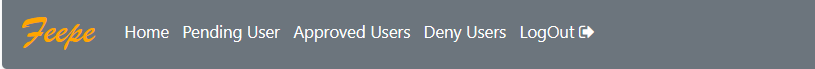
****

**7.3 Menus**

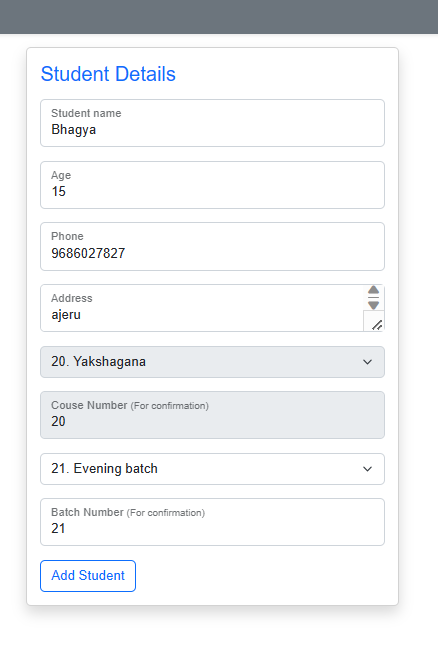
* Menus for the course owner.



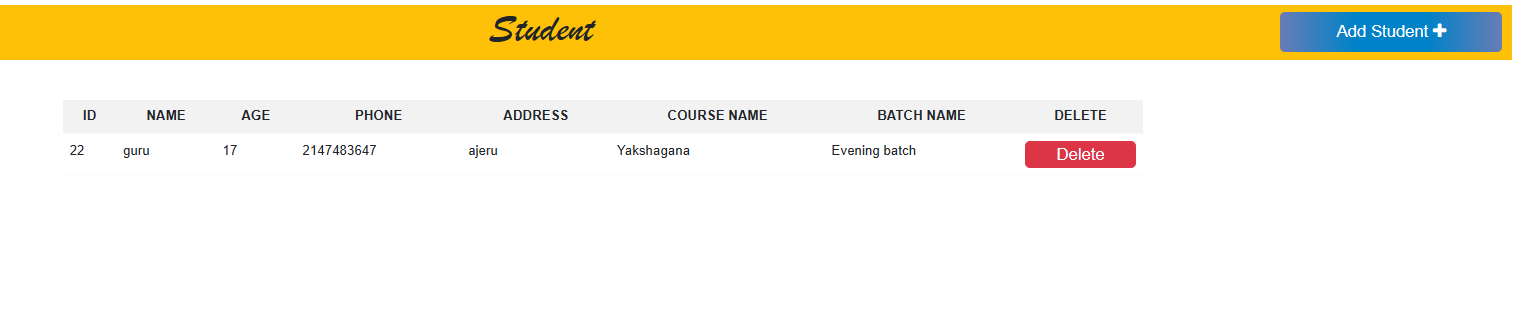
* Menus for admin.



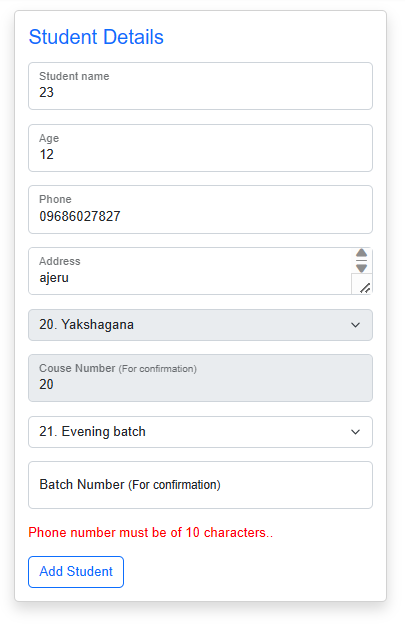
* 1. **Data Store/Retrieval/Delete**
     1. **Datastore**
* Student data storage using this form.

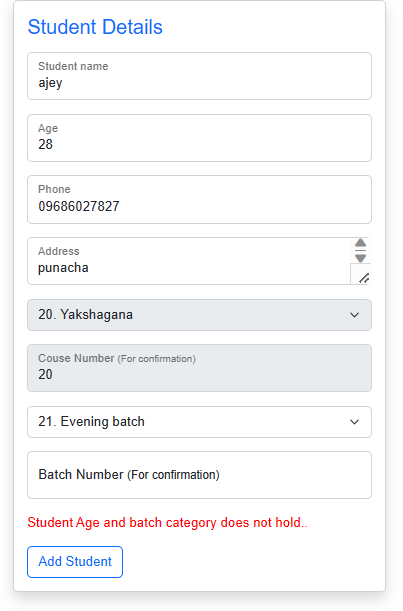
****

* + 1. **Retrieval and Delete**
* Student data retrieval and delete.

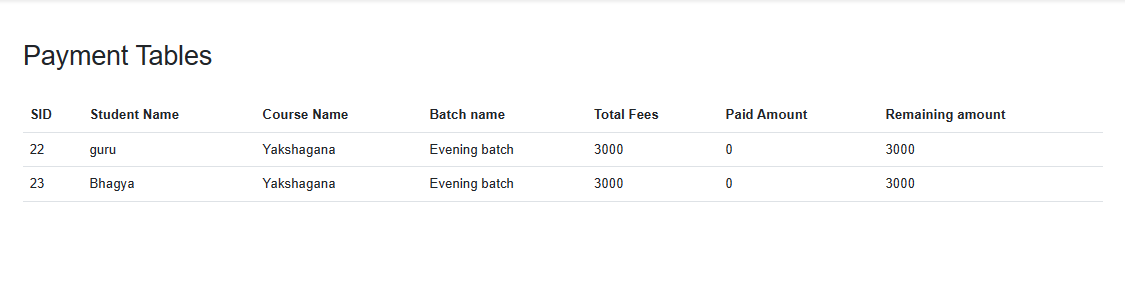


* 1. **Data Validation**
* Used for data validation while registering the student details.

****

****

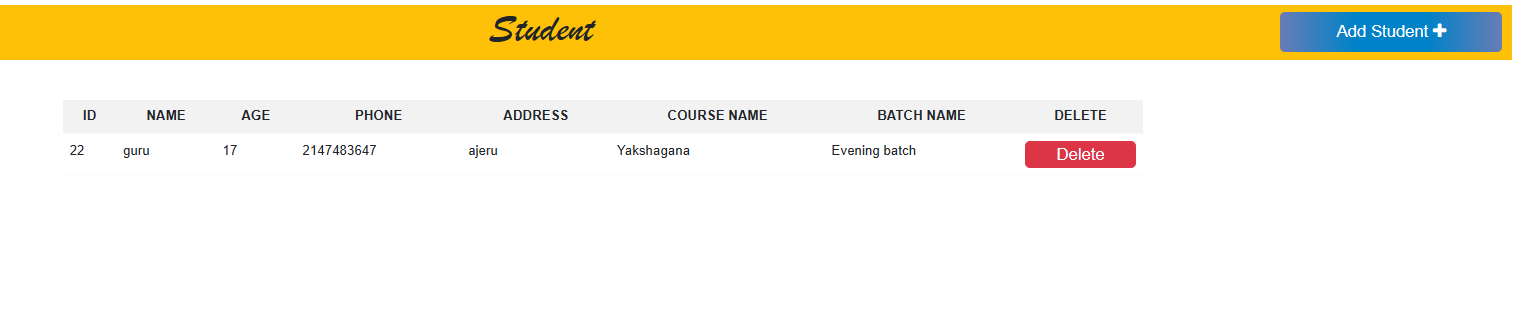
* 1. **Data Report**
* Data report of fees payment.



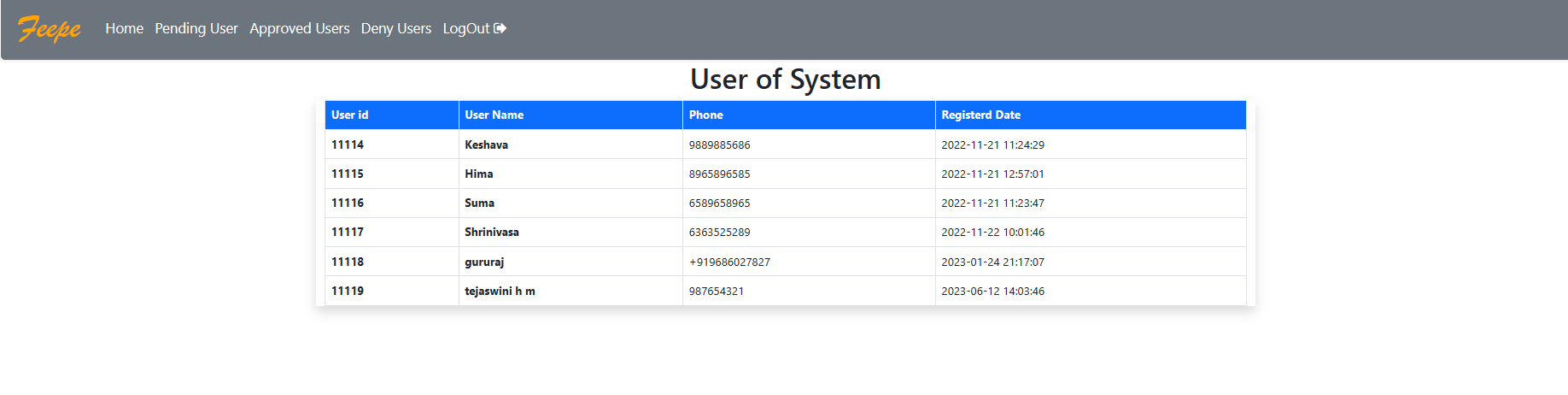
* Data report of total number of students, course and batch.



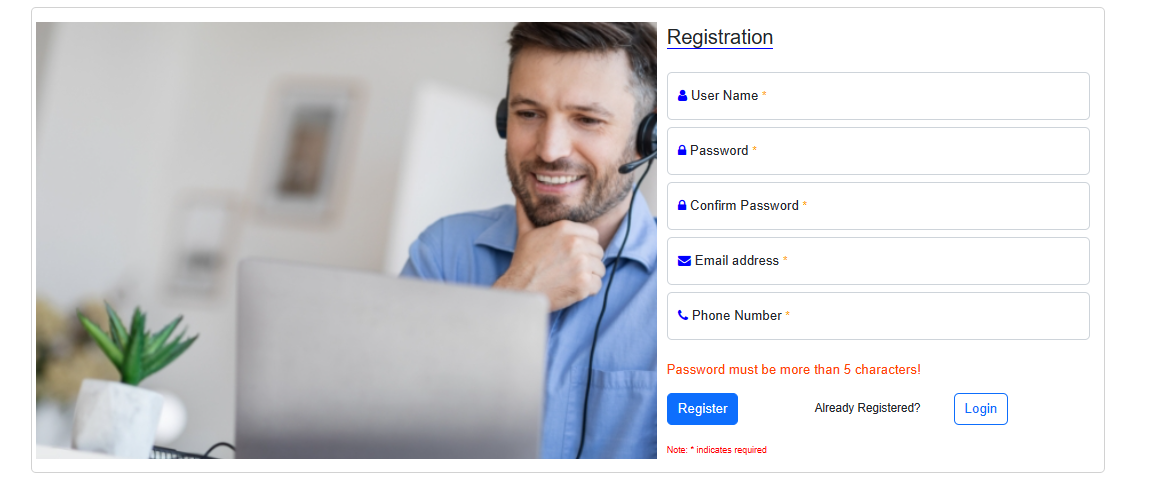
* Data report of student individual student.



* Data report for admin of the total number of the system.



* 1. **Data Alert**
* Alert message when user registration.

****

**CHAPTER-8**

**TESTING**

* 1. **Test Report**
     1. **Unit Testing**
     2. **Integration Testing**
     3. **System Testing**

