Java Project Report on Fee Management System

School of computer science and Engineering

Course code: CSE310

Topic: Fee Management System using the Java GUI

BY

CHINTALA GURUPREETH REDDY

12110193

INTRODUCTION

Fee collection and management are crucial tasks for all educational institutions. In the earlier days, school office staff used to collect the fees manually from the students and parents, providing printed receipts to each of the students after fee collection. It was tedious and hectic work for the administrators to collect and manage fees. But in today's world, educational institutions started implementing cloud and mobile-based school management software that made the process, smoother and much easier.

A fee management software is a task management system that automates fee collection and receipt generation. It also automates entries into the school accounts that help in reducing errors and eliminating duplicate data entries. The system supports both private and public schools of all sizes. The school management system software that includes fee management and accounts management modules can assist the school authorities in automating and performing various finance-related tasks. Such tasks include fee collection, customizing fee structure, setting discounts, tracking fraudulent transactions, adding fees, improving the cash management process and much more. Using this software, you can easily keep a real-time track of fee payments and other financial records. If you prefer to use a cloud system, you can transfer student details and fees details to the cloud by a single click. It is not a difficult task for a user to integrate the fee management module with other modules of a school management software. Integrating fee management with other modules provides a wider engagement for students and parents with the institution and improved productivity for the school.

Advantages

Data can be accessed remotely

You can easily access the data from anywhere at any time using your device. Various departments can transfer and import student data from devices located at multiple locations. Automated reporting increases the revenue and productivity of an institution. It also helps in eliminating errors that occur during the manual calculation.

Avoiding processing fees

Some banks provide corporate banking solutions for schools and other institutions that help to avoid processing fees and other similar charges for online transactions. In such a way, schools can save money by eliminating card payment processing charges. If an institution is a non-profit organization, then that institution gets even better options to save such charges.

A simple and safe fee payment system for parents

Online payment methods for collecting fees helps parents to easily make payment through a single click and track the payments whenever they like. Parents can deposit fees through secure payment gateways, using their mobile app as well as from a PC or any other device. In this way, they can keep real-time tracking of their ward's fee payment details.

Easy Accounting

Utilizing fee management software for accounting purposes has helped educational institutions save time and manage finance effectively. It assisted various institutions to reduce paperwork and issue receipts through email, SMS, etc. The fee management system can generate student-wise or class-wise fee collection reports and account-related reports that assist the finance management faculties to manage the accounts smoothly.

Besides these advantages, a fee management system also offers numerous benefits such as generating a fee structure, cancelling transactions, defining fee slabs for various standards, defining fee discounts, and much more.

Description About Project

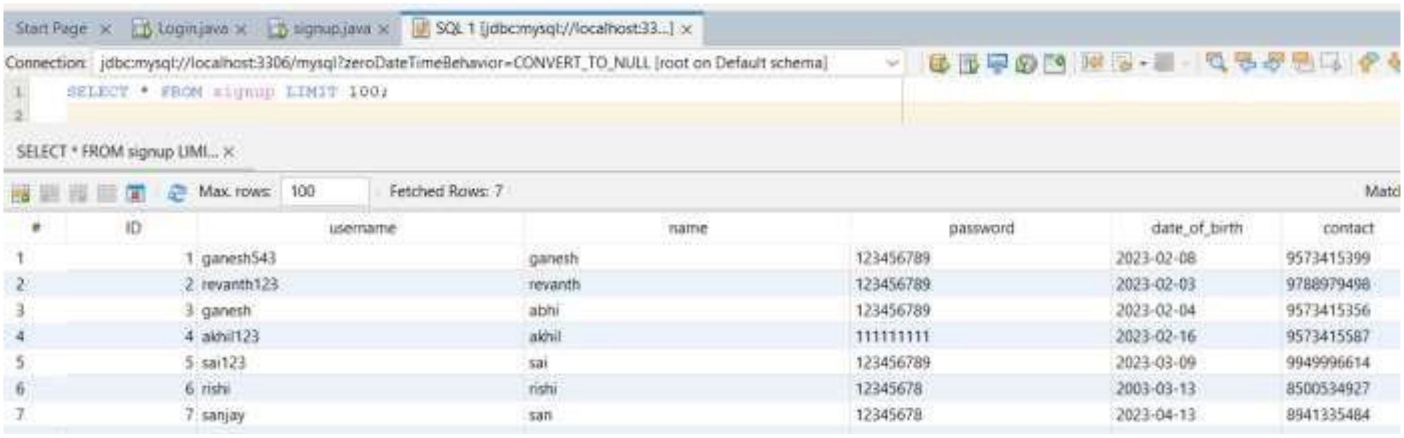
The project is prepared using Java **GUI** in compiler NetBeans IDE-16 along with database **Mysql**.

This project consists of 6 main Java modules:

1. Signup.java
2. Login.java
3. home.java
4. addFee.java
5. editCourse.java
6. searchRecord.java

Features of each module

1.Signup.java: It is typically a page that allows users to create an account or register for a service. The page may ask for basic information such as name, email address, and password. The data entered by the user is stored in MySql table called **signup.**


Start Page × Login.java × signup.java × SQL 1 [jdbc:mysql://localhost:33...] ×

Connection: jdbc:mysql://localhost:3306/mysql?zeroDateTimeBehavior=CONVERT_TO_NULL [root on Default schema]

```
SELECT * FROM signup LIMIT 100;
```

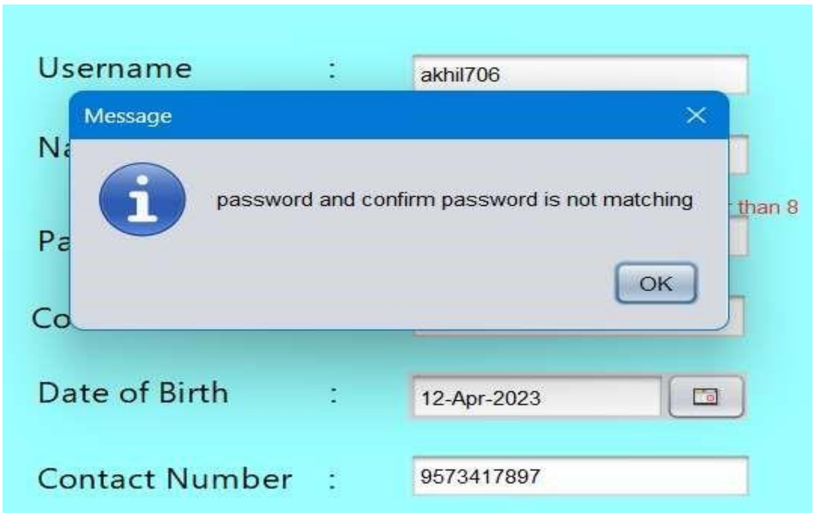
SELECT * FROM signup LIMI... ×

Max. rows: 100 Fetched Rows: 7

#	ID	username	name	password	date_of_birth	contact
1	1	ganesh543	ganesh	123456789	2023-02-08	9573415399
2	2	revanth123	revanth	123456789	2023-02-03	9788979498
3	3	ganesh	abhi	123456789	2023-02-04	9573415356
4	4	akhil123	akhil	111111111	2023-02-16	9573415587
5	5	sai123	sai	123456789	2023-03-09	9949996614
6	6	rishi	rishi	12345678	2003-03-13	8500534927
7	7	sanjay	san	12345678	2023-04-13	8941335484

Characteristics:

- In this ID are generated automatically with the help of an function **getId()** function.
- To validate and check the entered data, **boolean validation()** function is declared which show an popup if any entry is of invalid input by the user using function in swing library called **JOptionPane.showMessageDialog().**
- Password should be of 8 characters minimum, both Password and Confirm Password should be matched. No text field should be empty
- To insert data into signup table entered by the user **void insertDetails()** is declared.
- Contact Number should be 10 digit numbers only.


Username : akhil706

Message

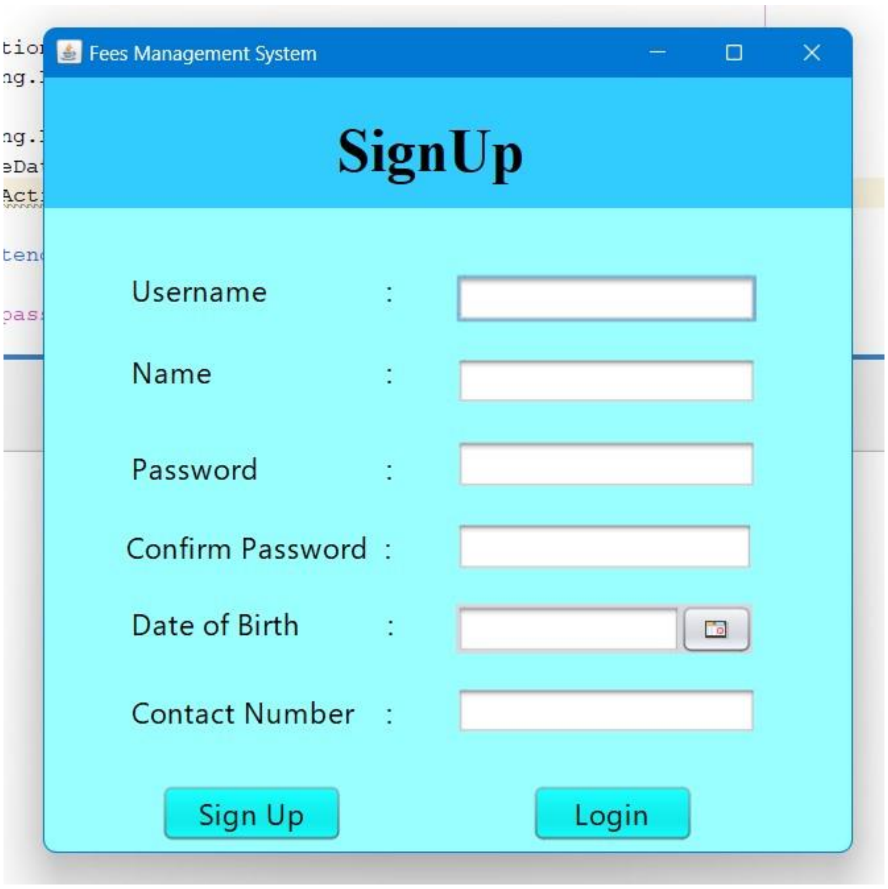
password and confirm password is not matching

OK

Date of Birth : 12-Apr-2023

Contact Number : 9573417897

This is the User-Interface of an Signup page



❼JButton : Signup, Login at page bottom.

❼JLabels : used in describing each field

❼ JTextField: for user input.

❼ jPanel: To differentiate and design the upper, and lower fields.

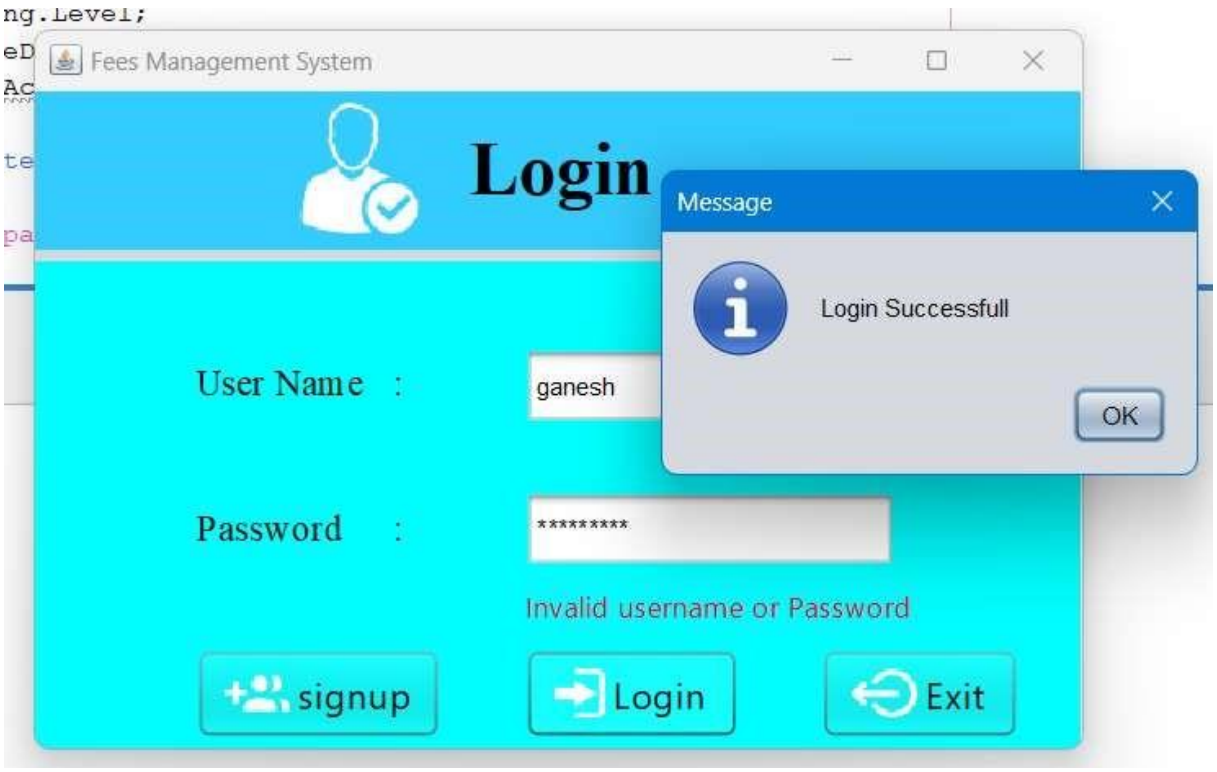
2.Login.java: The login page typically consists of a form with fields for the user to enter their login information. These fields may include a username, email address, or another identifier, as well as a password field for the user to enter their password.

Characteristics:

- ✞ In this module when user enters input for the text fields and press on Login button it validates event in java called **void btnLoginActionPerformed() .**
- ✞ The function used to validate whether the entered data is present in signup table database is verified by **void userVerification(String username,String password).**
- ✞ On pressing the signup buttons page will navigate to signup page

- On pressing the Exit button using **dispose() function** gui will exit from running.

This is the User-Interface of an Login page



Login Successfull

JComponents used in this module are:

❼JButton : Signup, Login, Exit at page bottom.

❼JLabels : used in describing each field

❼ JTextField: for user input.

❼ jPanel: To differentiate and design the upper, and lower fields.

3.Home.java: This is used to navigate to different modules and acts as a home/dashboard of the whole project in which all modules are present.

Characteristics:

- Mouse clicked, entered, and exited events are used to change the color of the text field to act like a hover function.
- Each field as JPlane on the back of the JLabel of the text field.

JComponents used in this module are:

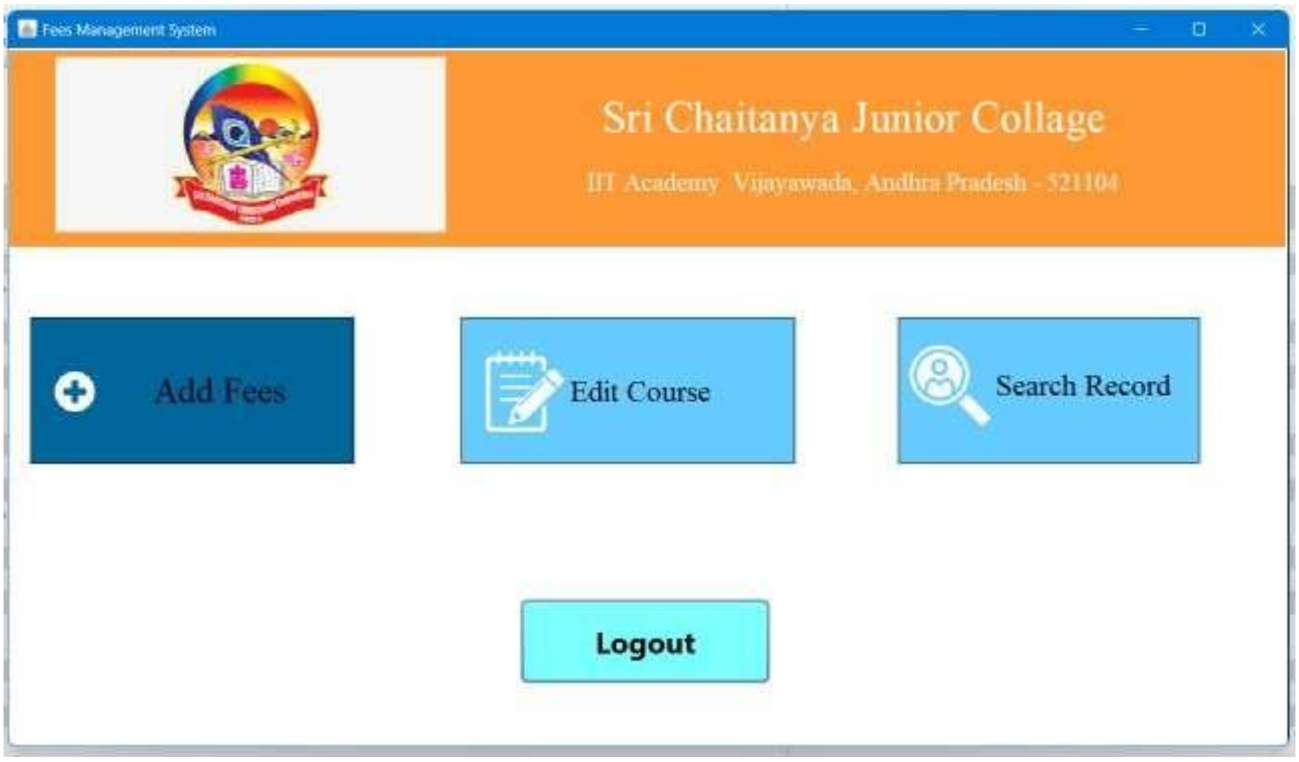
❼JButton : Logout at page bottom.

❼JLabels : Add Fees, Edit Course, Search Record and address, Collage name at top.

❼ JTextField: for user input.

❼ jPanel: To differentiate and design the upper, and lower fields.

This is the User-Interface of an Home page

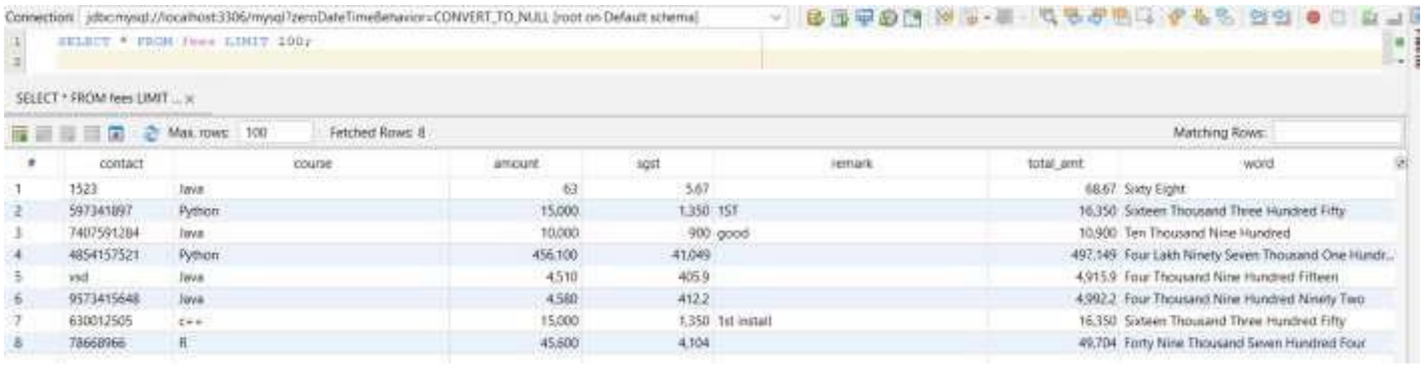

Fees Management System
Sri Chaitanya Junior Collage
IIT Academy Vijayawada, Andhra Pradesh - 521104
Add Fees
Edit Course
Search Record
Logout

4.addFee.java: This is the main page of the project as it is used to update the details of the student fees. In which there are four modes of payment **DD, Cheque, Card and Cash**. When the user selects the cash Bank Name, DD/Cheque number will be disabled using the function **setVisible(false)** to prevent visibility on JFrame.

Characteristics:

- ✞ The data entered is stored in a table called **fees** where Receipt_no act as a primary key for the table.
- ✞ **Receipt No.** is generated automatically using user-defined function **getRecieptNo()** which it gets the max value from the **fees** table in a database.
- ✞ Date of Transaction, registration_no, contact number, amount field cannot be empty otherwise it will throw an pop message.

- Courses will be auto updated from Edit courses module.
- When amount is entered SGST 9% is automatically updated using **event** txt_amountActionPerformed.
- To convert amount entered by the user into words an class function is declared **NumberToWordsConverter.**
- To insert data from text-field into database user-defined **innerData()** function is used.



Connection: jdbc:mysql://localhost:3306/mysql?zeroDateTimeBehavior=CONVERT_TO_NULL [root on Default schema]

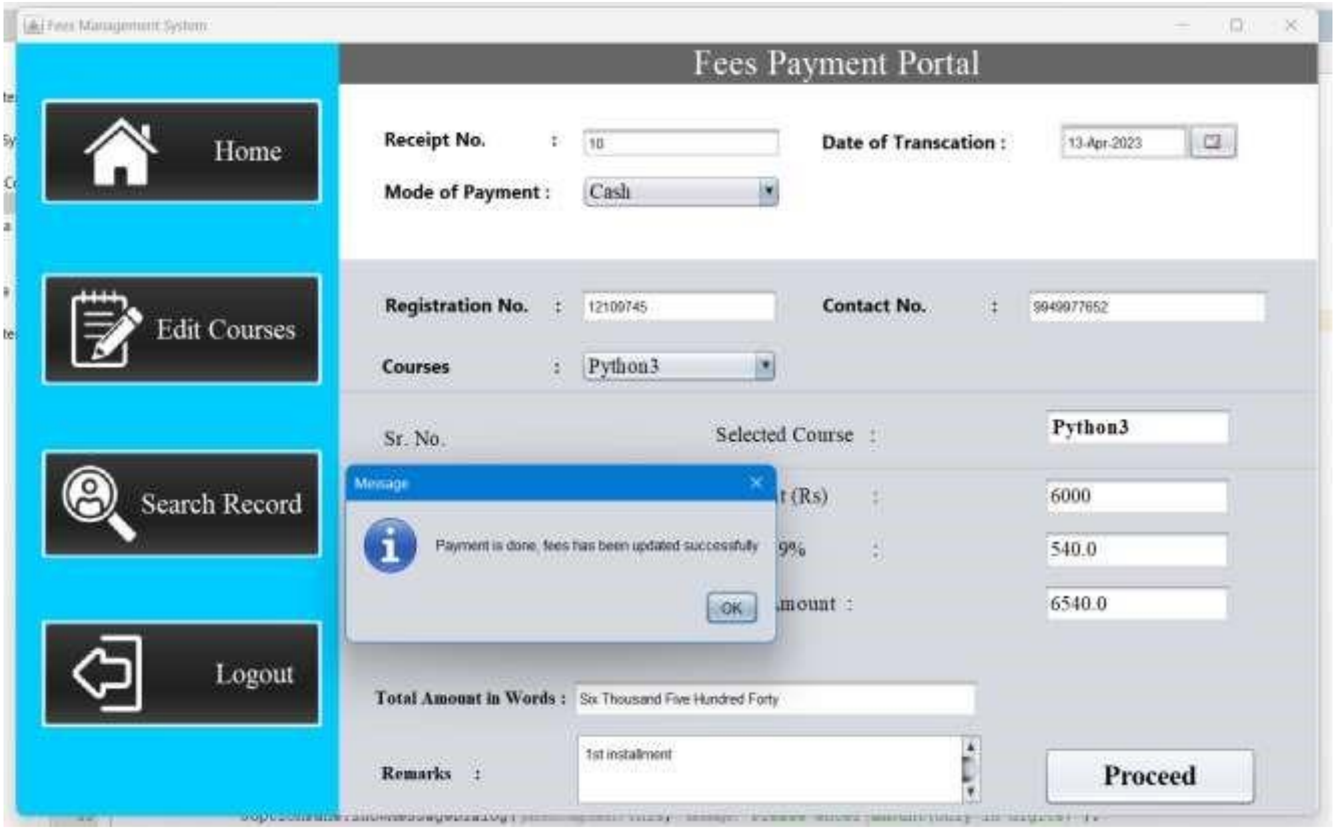
```
SELECT * FROM fees LIMIT 100;
```

SELECT * FROM fees LIMIT ...

Max. rows: 100 Fetched Rows: 8 Matching Rows:

#	contact	course	amount	sgst	remark	total_amt	word
1	1523	Java	63	5.67		68.67	Sixty Eight
2	597341897	Python	15,000	1,350	1ST	16,350	Sixteen Thousand Three Hundred Fifty
3	7407591284	Java	10,000	900	good	10,900	Ten Thousand Nine Hundred
4	4854157521	Python	456,100	41,049		497,149	Four Lakh Ninety Seven Thousand One Hundr...
5	vsd	Java	4,510	405.9		4,915.9	Four Thousand Nine Hundred Fifteen
6	9573415648	Java	4,580	412.2		4,992.2	Four Thousand Nine Hundred Ninety Two
7	630012505	c++	15,000	1,350	1st install	16,350	Sixteen Thousand Three Hundred Fifty
8	78668966	R	45,600	4,104		49,704	Forty Nine Thousand Seven Hundred Four

This is the User-Interface of an AddFees page



Fees Payment Portal

Receipt No. :	10	Date of Transcation :	13-Apr-2023
Mode of Payment :	Cash		
Registration No. :	12109745	Contact No. :	9949977652
Courses :	Python3		
Sr. No.		Selected Course :	Python3
		t (Rs) :	6000
		9% :	540.0
		mount :	6540.0
Total Amount in Words :	Six Thousand Five Hundred Forty		
Remarks :	1st installment		

Message: Payment is done, fees has been updated successfully

- Validation of whole data entered is verified wheater in text-fileds are of correct-type, any empty fields, etc are checked using **boolean validation1()** function. When **Proceed** button is pressed using events validation1() fun is called if return type is true, then it displays

“Payment is done, fees has been updated successfully”, if return type is false then it shows **“Data updation failed”.**

JComponents used in this module are:

❼JButton : Home, edit courses, Search Record and Logout on left side, Proceed on bottom.

❼ JTextField: for user input.

❼jScrollPane: used to writing remarks

❼jSeparator: used to separate an upper, lower Gui for design purpose.

❼ jPanel: To differentiate and design the upper, and lower fields.

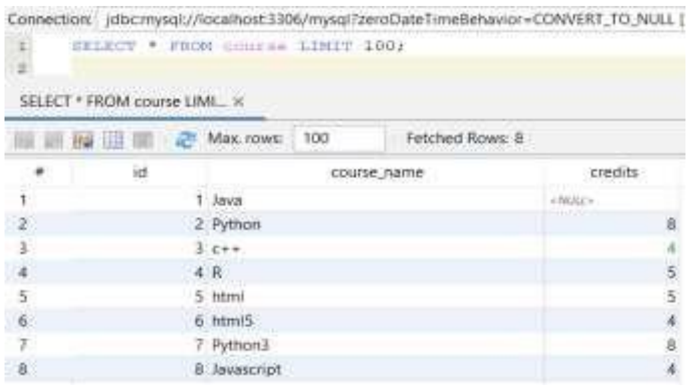
❼JComboBox<String>: To choose mode of payment where data-types are of string type.

6. **editCourse.java:** It is used for adding new courses, update the existing courses or to terminate the courses.

Characteristics:

- All the data of the courses are stored in table called “**course**” in database, where course_id act as an primary key for the table, when new course is added using same course id which already in the database it throws an pop message which is achieved by error handling method in database.

Database of course table



Connection: jdbc:mysql://localhost:3306/mysql?zeroDateTimeBehavior=CONVERT_TO_NULL

```
SELECT * FROM course LIMIT 100;
```

SELECT * FROM course LIMI... ×

Max. rows: 100   Fetched Rows: 8

#	id	course_name	credits
1	1	Java	<NULL>
2	2	Python	8
3	3	c++	4
4	4	R	5
5	5	html	5
6	6	html5	4
7	7	Python3	8
8	8	Javascript	4

- **void setRecords()** is defined for showing the selected data on text fields when clicked on particular table row.
- **void addCourse(int id,String cname,int sub)** used for add courses into database and called insert event button.

✞ **void update(int id,String cname,int sub)** used to update the data in table and called in action event on pressing update button.

✞ **void delete(int id)** which takes selected id as an input and delete entire row ad ID is an primary key of the table.

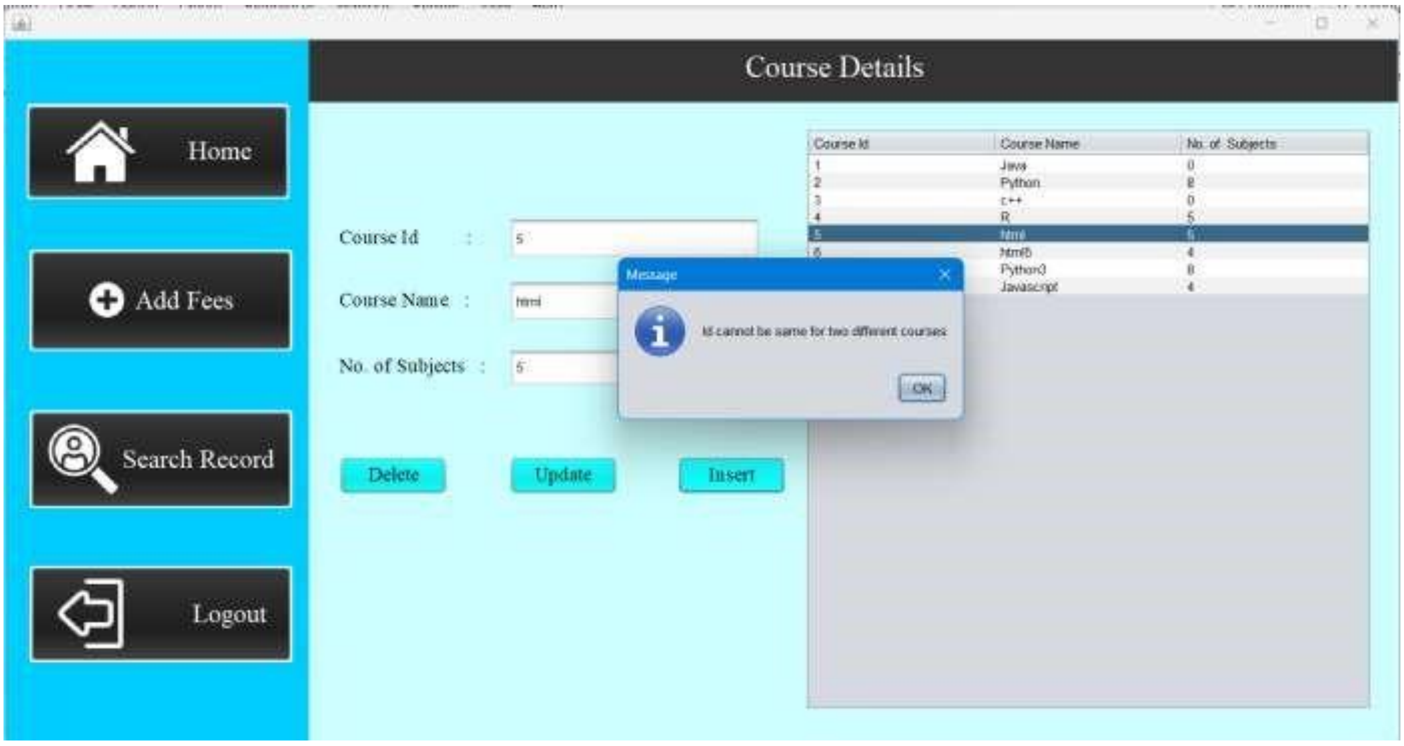
✞ **JComponents used in this module are:**

❼JButton : Delete, Update, Insert

❼ jPanel: There are two planes inside JFrame, one is Parent_plane for menu bar on left side, child Plane for remaining part.

❼ JTable: for showing the course_id, course name, no of subjects in table format.

This is the User-Interface of an editCourse page



Course Id : 5
Course Name : html
No. of Subjects : 5

Course Id	Course Name	No. of Subjects
1	Java	0
2	Python	8
3	c++	0
4	R	5
5	html	5
6	html5	4
[illegible]	Python3	8
[illegible]	Javascript	4

Message

Id cannot be same for two different courses

7.SearchRecord.java: This page consists of the student’s fee records from database of table “fees”.

Characteristics:

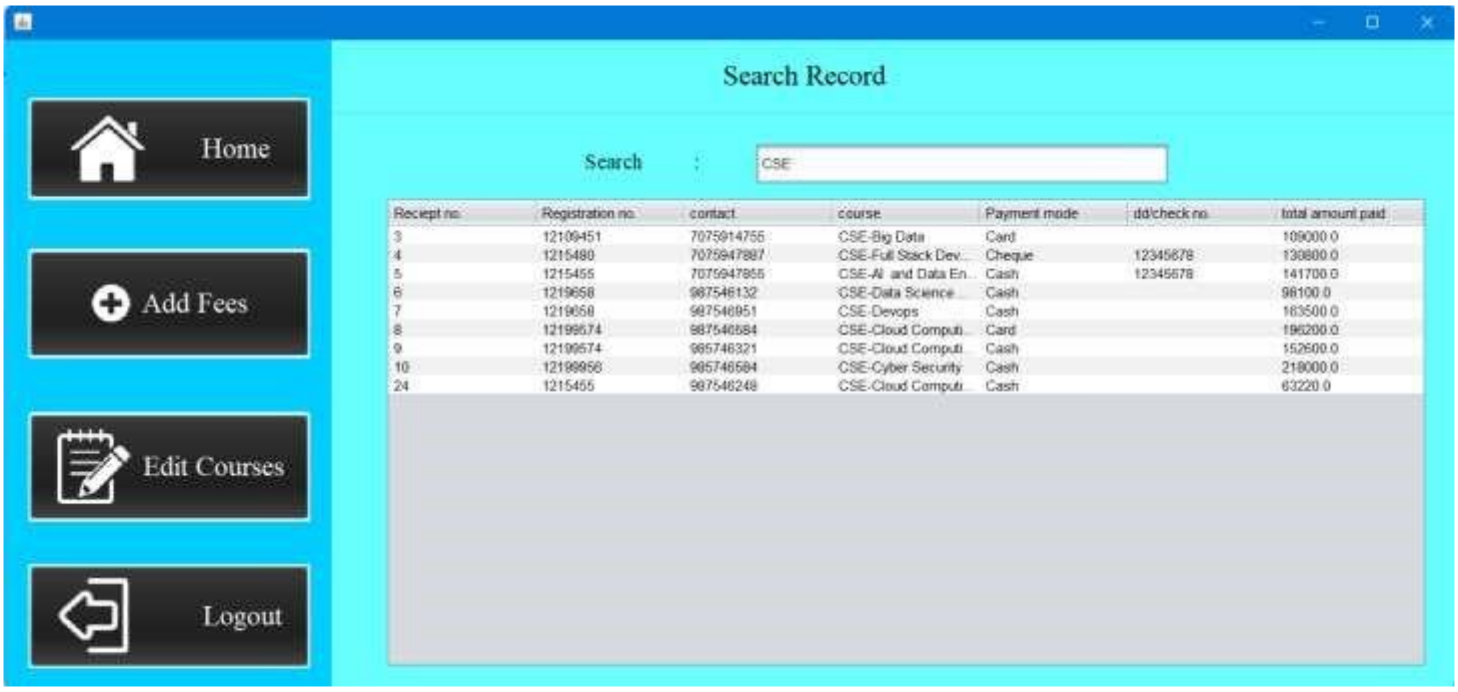
✞ **void setRecords()** is called in the class constructor itself so that the table will be filled with data automatically. This function uses **getString()** for string type, and **getFloat**()

for float type and assigns them to the temporary variables of the same data types. This temporary variable is assigned through the Object [] class in order to the table using **addRow().**

✞ **void search(String str)** where “str” indicates the text entered by the user to search. Below is the code this search function.

```
  public void search(String str){
      model = (DefaultTableModel)tbl_stud.getModel();
      TableRowSorter<DefaultTableModel> trs = new TableRowSorter<>(model);
tbl_stud.setRowSorter(trs);
      trs.setRowFilter(RowFilter.regexFilter(str));
  }
```

This is the User-Interface of a Search Record page



CSE

Reciept no	Registration no	contact	course	Payment mode	dd/check no	total amount paid
3	12109451	7075914755	CSE-Big Data	Card		109000.0
4	1215480	7075947887	CSE-Full Stack Dev...	Cheque	12345678	130800.0
5	1215455	7075947855	CSE-AI and Data En...	Cash	12345678	141700.0
6	1219658	987546132	CSE-Data Science ...	Cash		98100.0
7	1219658	987546951	CSE-Devops	Cash		183500.0
8	12199574	987546584	CSE-Cloud Computi...	Card		196200.0
9	12199574	985746321	CSE-Cloud Computi...	Cash		152600.0
10	12199958	985746584	CSE-Cyber Security	Cash		218000.0
24	1215455	987546248	CSE-Cloud Computi...	Cash		63220.0

Some common functions used in this Project are:

- **equals():** used for comparing two strings to check whether they are equal or not.
- **setVisible(Boolean):** If boolean is **true** then JComponents add to the frame, vice versa.
- **getText():** To get the text from database of string type.
- **setString():** To assign some value to JLabels or other components this is used.

Some Functions of the Database used in a project are:

```
Class.forName("com.mysql.cj.jdbc.Driver");
```

It is used to load the MySQL JDBC driver dynamically at runtime. The "Class.forName()" method is used to dynamically load a class by name, and in this case, the class that is being loaded is "com.mysql.cj.jdbc.Driver". This class is the driver for the MySQL Connector/J JDBC driver, which is used to connect to MySQL databases. When the "Class.forName()" method is called, the JVM looks for the class with the specified name and loads it into memory if it is not already loaded. Once the class is loaded, its static initializer is executed, which registers the driver with the DriverManager. After loading the driver, you can use it to connect to a MySQL database using JDBC.

```
DriverManager.getConnection("jdbc:mysql://localhost:3306/mysql?zeroDateTimeBehavior=CONVERT_T
O_NULL","root","*******");
```

This code is used to establish a connection to a MySQL database using the JDBC driver. Let me explain the different parts of the code:

DriverManager.getConnection(): This is a method from the java.sql.DriverManager class, which is used to establish a connection to a database.

"jdbc:mysql://localhost:3306/mysql?zeroDateTimeBehavior=CONVERT_TO_NULL" This is the URL of the MySQL database that we want to connect to.

❼'jdbc' stands for Java Database Connectivity, which is a Java API for connecting to databases.

❼ 'zeroDateTimeBehavior=CONVERT_TO_NULL' is an optional parameter that specifies how the driver should handle zero dates (i.e., dates with all zeros). In this case, it is set to
❼'CONVERT_TO_NULL', which means that the driver will convert zero dates to null values.

Java Concepts used are:

1. Inheritance
2. Exception Handling
3. Functional Interface and Lambda Expressions
4. Events in Java