D:\HP New laptop Data\PANKTI\Sem 2\Java\IET (1).jpg

**OOP PROJECT ON**

**STUDENT PROFILE MANAGEMENT SYSTEM**

**DEVELOPED BY:**

* **Bansi Gajera (201501028)**
* **Maharshi Doshi (201501049)**
* **Dhwani Mehta (201501054)**
* **Pankti Vadalia (201501068)**
* **Priya Mehta (201501085)**

**GROUP: 28**

**UNDER THE GUIDANCE OF:**

**Mr. Sujoy Pal,**

**Institute of Engineering and Technology (IET), Ahmedabad University**

1. **INTRODUCTION**

The main motive of this application is storing the scholastic and co-scholastic performance of the student during his/her college tenure. The application facilitates the student to enter details about his/her academic performance (Semester results in the form of GPA), Projects and Workshops attended, Internships worked in, and co-curricular performance in terms of competitions involving activities from various spheres. With the help of this application, a student is saved the bother of remembering and keeping track of his/her progress and activities. At the end of the four years of Engineering under graduation, he/she can have a precise and logical overview. Moreover the application will also help in making of his Curriculum Vitae, as the application contains all the data the interviewer would need.

1. **DETAILED FEATURES**
2. Each user is identified by the means of his University Roll number
3. The user can register into the application by filling a very simple registration form, adding certain basic details like Date of birth, Admission year, etc. This information is kept secure, and is not visible to other users using the application
4. After registering into the application, the user is ready to go
5. The application allows the user to enter
6. Semester results (GPA)
7. Details about various Workshops attended, conducted by renowned Companies and personalities
8. Being a part of interesting Projects, or Companies as interns, and mentioning respective contribution
9. Attending different competitions involving various fields of creativity, mentioning host academy and certification
10. Not making it too tedious, the user only has to enter his academic results at the end of each semester, making a mention about projects, workshops and internships
11. Keeping in mind the fact that some students may be very active into extra-curricular activities, the application gives them the luxury to enter as many records as they want, at any time they wish.
12. A student may want an overview of his activities till date so that he can work upon his strengths and weaknesses. For meeting the demand, the student can display records entered by him/her whenever he wishes
13. Considering human errors, the application provides the option of updating the information that he/she has entered into the application. When the user invokes his demand of update, the application provides him the chance of any updating any information in that domain
14. Initially when the user is not on-hands with the application, he may want to check it out by adding dummy information. This information may need to be discarded in a while. Considering this demand, the application provides the option of deleting the whole record in that domain
15. **ADVANCED FEATURES AND LIMITATIONS**

* The application comes with a very convenient option. The user does not need to search the display window for updating or deleting his/her record. What needs to be done is that the user just has to click the record, and he gets an exclusive view of the record, with the choice of updating or deleting the record. This feature is available in both, the Academic, as well as the Non-Academic sections.

We would like to mention the limitations as an extension and further scope of this project, as these can be done if given due time and opportunity:

* The application can be made more elaborate by storing subjects corresponding to each semester, and give the user a more precise comparison by making comparisons among subjects as well
* For the Non-Academic section, there should be a constraint such that only those competitions/events will be entertained whose hard/soft copy certification is uploaded by the user (We have already coded the logic for browsing and viewing files)
* The application can be extended into a web based application that would generate the complete CV of the student at the end of eight semesters

1. **ASSUMPTIONS**

The assumptions that we have made are quite obvious so that the application can reach and ponder to the needs of a student who does not possess the knowledge of the coding in the back end, and can use the application with basic common sense. After login by the means of roll number, we expect the user to enter valid, precise, authentic, and to-the-point information as asked in the different frames. Playing with information will not really do good to the user himself. For the given application, we have kept in mind a student studying in a course with as many as eight semesters/four years of tenure (which can be extended for users from other fields by making small changes). For the new user frame, we have made an obvious assumption that any user trying to register would be able to fill out all his basic details like Gender, Date of birth, etc., and so we urge the user to fill out the necessary fields, assuring complete security of his/her data. For the frame asking for academic information, we have assumed that for a given semester, the user may not have attended any projects, workshops or internships; so we have given the liberty to leave the fields empty, unlike the others such as Grade and Semester. For the non-academic frame, we have assumed that the user would wish to enter information only about important competitions and events, ready with a certificate or appreciation letter; so we mandate the user to fill out all the information that the fields ask for in the frame.

1. **CLASS DIAGRAM**
2. **DATA STORAGE STRUCTURE**

For storing the data, we had the option of either using files as the storage component, or making the use of DBMS. The coding of a JAVA file was very well known to us, but we needed a very precise and structured storage of data, not compromising on any bit of information. Therefore we decided to use MySQL. MySQL is an open source relational database management software (RDBMS). In MySQL, the data is stored in the form of related tables. A MySQL table can be accessed directly using C, C++, JAVA, Eiffel, Perl, PHP and Python. For the given application, we have used JAVA. MySQL databases are queried using a subset of the standard Structured Query Language (SQL) commands. The use of MySQL demanded the knowledge of how data is stored and implemented in MySQL tables, and queries for the same. For the use of MySQL into the given application, it was first connected to the IDE with the help of the connection coding. Tables were created in the MySQL Workbench by inserting them in a Schema. A Schema is a collection of database objects (here, tables). After the creation of table in the Workbench, the remaining queries could be fired in the IDE. Under the head of MySQL, extensive use of queries like SELECT, UPDATE, DELETE, INSERT has been made. While wanting to edit a given record, the application makes sure that the record already exists in the Database (as for modification, the user just has to click his/her mouse pointer the records from the list displayed on the screen). Therefore we made the choice of using the UPDATE query against the MODIFY query, as for MODIFY, MySQL creates a new record if the records asked for doesn’t exist. The understanding of various datatypes available in MySQL like INT, DOUBLE, VARCHAR(), DATE proved helpful. Certain data was mandatory to be entered by each user across the Database (example, roll number); for that we made the use of NOT NULL Key in MySQL which, as name suggests, does not allow NULL values. Also, PRIMARY KEY has been inserted in each table for its unique identification of each tupple (row). In the given application, the roll number has been set as the PRIMARY KEY. For fetching out certain data, UNIQUE KEY has been used. The UNIQUE KEY is given data by merging two attributes (columns) of the table. In the given application, there are 3 tables-

1. Main Database: that stores basic details (Name, Roll number, Date of birth, Gender, Contact number, email id) of all the students across the system
2. Academic: that stores the academic excellence of a given student (Semester GPA, Projects, Internships, and Workshops). Each student will be viewing only his/her own record. Other records are kept hidden
3. Non-Academic: that records the extra-curricular activities of the student (Certification, Activity name, Activity details, Host Academy). Yet again, each student will be viewing only his/her own record. Other records are kept hidden

For the Main Database, each student will have a different roll number given to him by the University, and that will work as his sole identification. For the Academic Database, a given student (with a given roll number) has to be given the opportunity to enter as many as 8 records (one for each semester). In that case, providing roll number as the PRIMARY KEY would generate unwanted errors. Therefore our team came up with the solution to declare a MySQL UNIQUE KEY by making a set of {Roll Number, Semester}. This unique was now able to identify each row uniquely. For the Non-Academic Database, no restriction has been posed on the system or the user. A user has the luxury to enter as many records as he/she wishes under the same roll number and semester.

1. **DIVISION OF TASK AMONG TEAM MEMBERS**

Division of task was a task requiring precise identification of strengths and capabilities of various group members. We identified different layers of the project as sharing each member’s point of view and ideas about the given application, what each member expected from the application as a user of the same, how each member wanted to proceed with the project, and finally going about the actual working and coding.

After having long discussions, arguments and logical clashes, the team members had finally agreed upon the final idea. The flow of frames and formulation of the ideas were designed by Pankti Vadalia and Priya Mehta. After taking consent of the team members, the team members were ready to go. Being good at designing, Maharshi Doshi took the task of making some of the GUI Frames, implementing various JAVA Swing features. Dhwani Mehta and Bansi Gajera took on the important task of Exception Handling, a very important feature of JAVA Programming, and our application. Priya Mehta took on the task of inserting action listeners for the buttons, with Pankti Vadalia in assistance. Pankti Vadalia and Priya Mehta coded the logic for connecting JAVA IDE with MySQL. The report has been made by Pankti Vadalia.

1. **NAMING CONVENTION AND PACKAGE-FILE DETAILS**

All data members, methods, classes, interfaces and packages are made to follow camel-case.

All the packages and files are named according to their respective tasks, so that it is easier for even other programmers to figure out the working of each file.

Below given are the packages that constitute our application, complete with the files contained in them:

1. StudentProfileManagement: Being the main subject of our application, this package is not only the root of all other packages of the application, but also made to contain the class with Main() function, and the basic Connection files.
2. AcademicManagement: Contains all the files relating to Academics:
   1. Academic: Basic class of the package; contains the properties of Academics, and methods that can felicitate movement of data fields across other packages, and within itself.
   2. AcademicGUI: Contains the frame window which is created to accept values to be stored in the database
   3. AcademicFocusListener: Contains the mouse listener. When the user clicks a record in the window, he gets an exclusive view of the record
   4. DisplayFrame: This is the file that gets invoked upon getting an enlarged view of the record along with the options of updating and deleting the same
   5. AcademicActionListener: Contains the action listener. When the user presses Submit/Display/Update/Delete buttons, the respective tasks are executed.
   6. AcademicList: Contains Array List to store all details. These details are then sent to the MySQL database using getter and setter functions from basic class Academic. Various queries are fired regarding Insertion, Updation, Deletion upon invocation.
   7. PanelAcademicRecord: This file has been created to fetch data from database, store the data in labels, and display them on the panel in the GUI Window
3. NonAcademicManagement: Contains all the files relating to Academics:
   1. NonAcademic: Basic class of the package; contains the properties of Academics, and methods that can felicitate movement of data fields across other packages, and within itself.
   2. NonAcademicGUI: Contains the frame window which is created to accept values to be stored in the database
   3. NonAcademicFocusListener: Contains the mouse listener. When the user clicks a record in the window, he gets an exclusive view of the record
   4. DisplayFrame1: This is the file that gets invoked upon getting an enlarged view of the record along with the options of updating and deleting the same
   5. NonAcademicActionListener: Contains the action listener. When the user presses Submit/Display/Update/Delete buttons, the respective tasks are executed.
   6. NonAcademicList: Contains Array List to store all details. These details are then sent to the MySQL database using getter and setter functions from basic class Academic. Various queries are fired regarding Insertion, Updation, Deletion upon invocation.
   7. PanelNonAcademicRecord: This file has been created to fetch data from database, store the data in labels, and display them on the panel in the GUI Window
4. StudentCategory:
   1. Category: Gives the user to choose between Academic and Non-Academic
5. StudentDetailsManagement:
   1. FormFrame: New user form. Asks for all basic details of a new user
   2. FormFrameActionListener: Contains action listener for the form upon clicking the Submit button
   3. PanelStudentRecord: This file has been created to fetch data from database, store the data in labels, and display them on the panel in the GUI Window
   4. Student: Basic class of the package; contains the properties of Academics, and methods that can felicitate movement of data fields across other packages, and within itself.
   5. StudentList: Contains Array List to store all details. These details are then sent to the MySQL database using getter and setter functions from basic class Academic. Various queries are fired regarding Insertion, Updation, Deletion upon invocation.
   6. DisplayStudentRecords:
6. StudentLogin:
   1. LoginFrame: First GUI displayed on screen. Asks for login based on roll number and password, and provides option to register as new user
7. StudentMenu:
   1. FrameCompare: Compares the grades of a given student using the data he has previously entered in the database
8. **OBJECT ORIENTED APPROACH IN DESIGN AND CODING**

JAVA is a fully object oriented language, and therefore supports all features of OOP. In JAVA, only objects can be used to access the data and methods of any class. We have made use of inbuilt object oriented features of JAVA, and implemented as many as possible through our coding logic.

The list of inbuilt Object Oriented features that we have used in our code is as follows:

* Inheritance from classes:

1. AcademicGUI 🡪 JFrame
2. NonAcademicGUI 🡪 JFrame
3. DisplayFrame 🡪 JFrame
4. PanelAcademicRecord 🡪 JPanel
5. PanelNonAcademicRecord 🡪 JPanel
6. PanelStudentRecord 🡪 JPanel

* Inheritance from Interfaces:

1. AcademicListener 🡪 ActionListener
2. NonAcademicListener 🡪 ActionListener
3. DisplayFrameActionListener 🡪 ActionListener
4. DisplayFrame1ActionListener 🡪 ActionListener
5. AcademicFocusListener 🡪 MouseListener
6. NonAcademicFocusListener 🡪 MouseListener

* Aggregation

1. FormFrame3, CategoriesFrame11CheckboxGroup
2. FormFrame3, CategoriesFrame11  Checkbox
3. FormFrame3, CategoriesFrame11, AcademicGUI, NonAcademicGUI, DisplayFrame, DisplayFrame1  JButton
4. FormFrame3, CategoriesFrame11, AcademicGUI, NonAcademicGUI, DisplayFrame, DisplayFrame1  JLabel
5. FormFrame3, CategoriesFrame11, AcademicGUI, NonAcademicGUI  JPanel
6. CategoriesFrame11, AcademicGUI, NonAcademicGUI  JFrame
7. FormFrame3, CategoriesFrame11, AcademicGUI, NonAcademicGUI  JTextField
8. CategoriesFrame11, AcademicGUI, NonAcademicGUI  Color
9. CategoriesFrame11, AcademicGUI, NonAcademicGUI  Dimension
10. CategoriesFrame11, AcademicGUI, NonAcademicGUI  Font

* Dynamic binding:

1. Connection conn =StudentConnection.getConnection();
2. Statement stmt = conn.createStatement();
3. ResultSet rs = stmt.executeQuery(sql);

* Dynamic Polymorphism/ Method Overriding (highlighting widely used methods):

1. void mouseClicked (mouseEvent e)
2. void actionPerformed (actionEvent e)
3. getSource()
4. getText(), setText()
5. addActionListener()
6. getSelectedItem()
7. getState()
8. getConnection()
9. add(-Parameters-)
10. createStatement()
11. getInt()
12. getString()
13. getActionCommand()
14. addRecord()
15. fetchRecords(-Parameter-)

* Wrapper classes and Wrapper functions

1. Integer 🡪 (Abstract) Number
2. Methods: Integer.parseInt(), Integer.toString()

Three main classes were identified to work upon for the given application. Those were

1. Student
2. Academics
3. Non-Academics

Each of these classes possesses distinct properties and functions. The Student class represents a student, and therefore has properties like Name, Roll Number, Gender, Date of Birth, Batch year, Contact number, email id, Branch. Implementing Object oriented design in the code, we made these properties the data members of our class Student.

The Academics class represents Academics, and has properties like Semester Grade Point, Projects, Workshops, and Internships. Again implementing Object oriented design in the code, we made these properties the data members of our class Academics.

The Non-Academics class represents Non-Academics, and has properties like Activity name, Activity Details, Certification and Host Academy. Yet again, under the implementation of Object Oriented design in the code, we made these the data members of class Non-Academics.

* **Abstraction**: Abstraction is displaying only essential features by hiding unnecessary details. We have implemented abstraction by specifying the visibility of certain functions/data fields as private, to hide them from the rest of the classes outside itself.
* **Encapsulation**: There is an evident inclusion of encapsulation in each and every file, where data fields and methods are wrapped up into a single entity: a class. And with the access specifiers, data and methods are kept hidden in the object.
* **Static Polymorphism (Method Overloading):** Method overloading is implemented by methods having the same names but different signatures.

1. Academic (int roll, …) and Academic(String roll, …)
2. Student (int roll, …) and Student (String roll, …)

* **Static binding:** Static binding is a feature of JAVA where the type of object is determined and allocated memory during compile time.

1. Academic aca= new Academic()
2. AcademicList list = new AcademicList()
3. NonAcademicList list1 = new NonAcademicList()

* **Modularization**: Modularization is decomposing the code into simple, identifiable modules, is achieved by breaking down the code into different packages, and further, into files.

1. StudentProfileManagement
   1. StudentConnection
   2. StudentProfileManagement
2. StudentLogin
   1. LoginFrame
3. StudentMenu
   1. FrameCompare
4. StudentDetailsManagament
   1. FormFrame
   2. FormFrameActionListener
   3. PanelStudentRecord
   4. Student
   5. StudentList
   6. DisplayStudentRecords
5. StudentCategory
   1. Category
6. NonAcademicManagement
   1. NonAcademic
   2. NonAcademicFocusListener
   3. NonAcademicList
   4. NonAcademicGUI
   5. NonAcademicListener
   6. DisplayFrame1
   7. NonPanelAcademicRecord
7. AcademicManagement
   1. Academic
   2. AcademicFocusListener
   3. AcademicList
   4. AcademicGUI
   5. AcademicListener
   6. DisplayFrame
   7. PanelAcademicRecord

* **Inheritance (IS-A Relationship)**: Inheritance is noticeably implemented in the files.

1. AcademicList 🡪 Academic
2. NonAcademicList 🡪 NonAcademic
3. StudentList 🡪 Student

* **Aggregation (HAS-A Relationship)**: Aggregation is implemented when an object of one class is present in another class. We have made extensive use of aggregation in our application

1. object “agui” of AcademicGUI in AcademicFocusListener
2. object “frame” of AcademicGUI in AcademicListener
3. object “fc” of FrameCompare in FrameCompareActionListener
4. object “ngui” of NonAcademicGUI in NonAcademicFocusListener
5. object “mygui” of NonAcademicGUI in NonAcademicListener
6. object “disp” of DisplayFrame in AcademicList).
7. **CONCLUSION**

The given application has been designed to summarize the user’s progress in his Engineering under-graduation tenure. The team has worked sincerely towards making the application helpful and easy to use. The project was a very informative experience for all of us. Not only did our concepts get clearer, and we came to know about a hundred more features of JAVA, but also we polished our task of working as a team. Encouraging group members with their strengths, so that they can yield upto their highest potential has been one of our most learning experiences. We thank the authorities for giving us this opportunity.

1. **REFERENCES**

* Javaforums.org
* Stackoverflow
* Oracle.com
* Ntu.edu.sg
* Java.net
* Forums.netbeans.org
* Tutorialspoint.com
* Mnemstudio.org
* Daniweb.com