Chapter 1

INTRODUCTION

The **Student Attendance system** (SAS) is a stand-alone application. It provides a user-friendly, interactive Graphical User Interface (GUI) based on Java's Swing components. All data is stored in an Oracle database. The application uses a thin Oracle JDBC driver to communicate with the database.

1.1 Purpose

SAS is used by the Student to view his/her attendance in the subjects registered, only in an active semester and academic year. The Staff will be able to record the daily attendance in the subject(s) allocated to them, for the class(es) assigned to them. Staff will be able to generate percentage attendance reports in an Excel spreadsheet of the class(es) assigned to them. SAS is used by the Admin to configure. and maintain all data, relevant and critical, for the application to run and export attendance data into Excel spreadsheet.

1.2 Scope

Student Attendance System has a scope of three levels of users. At the first level, the ADMIN is allowed to maintain data about staff, students, academic calendars and user authentication, relevant and critical, for the application to run He may also backup the attendance to an Excel spreadsheet.

At the second level, the STAFF are allowed to take daily attendance for each subject allocated to them, in the respective classes assigned to them. A detailed percentage attendance report can be obtained at any point of time.

At the third level, the percentage attendance in subjects can be viewed by the students as part of the self-service.

This application restricts its scope to being a stand-alone applications. It can further be enhanced by making it Web-based.

1.3 Motivation

The main motivation behind the selection of this project was to design, develop and implement a software application which will be useful for the main stake holders of a typical educational institution, namely the faculty and students. Further to make application user interface interactive and user-friendly and at the same time challenging to design and develop in core Java.

1.4 Software Requirement Specifications

A **Software Requirements Specification** (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide.

1.4.1 Overall Description

The proposed system has been designed to support the users to have seem-less, interactive experience. The application is open to all levels of users. Each user can interact with the application, on authentication, only during an active semester of an academic year. The system supports the generation of a summarized and detailed percentage attendance report, in an Excel spreadsheet. It also enables the ADMIN to create, modify, remove and view all the content that actually resides in the database.

1.4.2 Functionality

There are three user-access levels

- STUDENT
- STAFF
- ADMIN

Therefore, the requirements could be efficiently analyzed depending on the user group and the functionalities they should be allowed to perform**.**

1.4.2.1 Functional Requirements -User

Each user has his separate account, to perform his own functionalities, identified by a user name and password. Every user is made to create an account before using the application.

1.4 2.2 Security Requirements

It is of utmost importance to ensure that there is protection against unauthorized access to the attendance details of students. Users must be provided with a login ID and password which grant access only to their respective account. Similarly, every user can only view his or her attendance thereby maintaining the integrity of the data viewed by them. The ADMIN on the other hand has privileged access in order to view all data and ensure seem less and efficient experience to all of its users.

1.4.2.3 Performance Requirement

The PCs used must be at least be Pentium 4 machines so that they can give optimum performance of the product. In addition to these requirements, the system should also embrace the following requirements:-

- **Reliability:** The system should have little or no downtime.
- **Ease of Use:** The general and administrative views should be easy to use and intuitive.

1.4.2.4 Design Constraints

The designers must design the database is such a way that any change in the information of a client should be updated and saved effectively in the database.

1.4.2.5 Hardware Requirement

- 64-bit processor
- Internet Connectivity 100 Mbps
- 2 GB memory

1.4.2.6 Software Requirement

1. Front-end tool: Eclipse (Oxygen version) or BlueJ Version 4.1.2
2. Back-end tool: Oracle Database 11G (Express edition)
3. Operating System: Windows 10
4. Oracle JDBC drivers for 11g
5. Java Calendar library
6. Java SDK 9

1.4.2.7 Interface Requirement

The interface which is provided in this software allows students to view attendance, staff to take attendance and view the percentage attendance reports, and the administrator to track all the interactions of the current users. The database designed should be very easy to use and user friendly.

- An account is maintained for every user to ensure security and anonymity.
- User can view his present attendance.
- The attendance taking interface is followed by a percentage attendance report.
- User can also make use of the email functionality to receive his password, both on creation of a new account and on reset, at his email.

Communication between the Oracle database and front-end is through a thin JDBC driver. It is connected with front-end with a connection, established by the Driver Manager. We can have two types of authentication that are user and SQL server. It has inbuilt API for connectivity. For implementing email functionality, GMAIL SMTP server with SSL access is used.

Chapter 2

E R DIAGRAM AND RELATIONAL SCHEMA DIAGRAM

Entity Relationship Diagram, also known as ERD, ER Diagram or ER model, is a type of structural diagram for use in database design. An entity-relationship diagram (ERD) is a graphical representation of an information system that shows the relationship between people, objects, places, concepts or events within that system.

2.1 E R Diagram

An ERD is a data modeling technique that helps to define business processes which can be used as the foundation for a relational database. An ERD contains different symbols and connectors that visualize two important information**:** The major entities within the system scope**,** and the inter-relationships among these entities. Fig. 2.1 represents the ER Diagram for student attendance system.

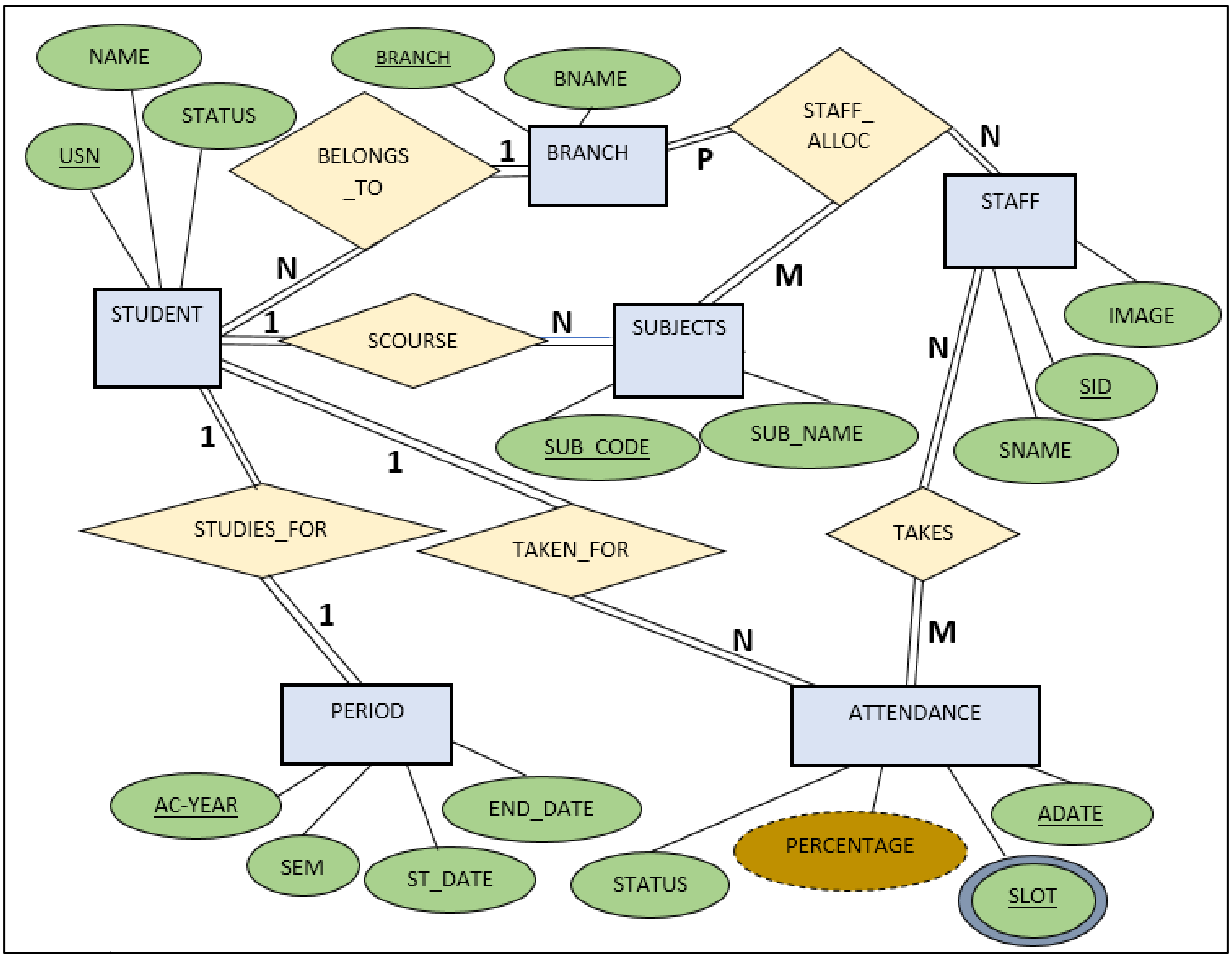


Fig. 2.1 E R Diagram for SAS

Three main components of an ERD are the entities, which are objects or concepts that can have data stored about them, the relationship between those entities, and the cardinality, which defines that relationship in terms of numbers.

In the SAS ER diagram, the Entity types like STUDENT and STAFF are in rectangular boxes. Relationships like BELONGS_TO and TEACHES are in diamond boxes, attached to entity types with straight lines. Attributes are shown in ovals, each attached by a straight line to entity or relationship type. Multivalued attributes (like SLOT) are in double ovals. Key attributes (like USN) are underlined. Component attributes of a composite attribute are attached to oval representing it. Derived attributes (like PERCENTAGE) are in dotted ovals.

1. In BELONGS_TO (Binary) Relationship, STUDENT: BRANCH is of cardinality N:1 as many students can belong to the same branch and 1 branch can have more than 1 student. There is total participation of both entities as :

- A student cannot exist without belonging to a branch.
- A branch cannot exist without students that belong to it.

2. In STUDIES_FOR (Binary) Relationship, STUDENT: PERIOD is of cardinality 1:1 as 1 student can study only 1 semester at a time. There is total participation of both entities as:

- A student cannot exist without studying in a semester.
- A period cannot exist without students studying in a semester.

3. In TAKEN_FOR (Binary) Relationship, STUDENT: ATTENDANCE is of cardinality 1:N as a student can have attendance for more than 1 day for more than 1 subject. There is total participation of both entities as:

- A student cannot exist without having attendance.
- Attendance cannot exist without students studying in a semester

4. In SCOURSE (Binary) Relationship, STUDENT: SUBJECT is of cardinality 1:N as a student can register for more than 1 subject. There is total participation of both entities as:

- A student cannot exist without registering for subjects.
- Subjects cannot exist without students registering for them.

5. In STAFF_ALLOC (Ternary) Relationship,

- For a given SUBJECT and BRANCH, more than 1 STAFF can be allocated and hence the cardinality of STAFF is N.

- For a given STAFF and BRANCH, more than 1 SUBJECT can be allocated and hence the cardinality of SUBJECT is M.
- For a given SUBJECT and STAFF, more than 1 BRANCH can be taught and hence the cardinality of BRANCH is P

There is total participation of all 3 entities as:

- A branch cannot exist without staff and staff cannot exist without teaching atleast 1 branch
- A subject cannot exist without staff to teach it and staff cannot exist without handling atleast 1 subject.
- A subject cannot exist without branch to be taught and branch cannot exist without atleast 1 subject being taught

2.2 Database Schema Diagram

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data. Fig. 2.2 presents the SAS schema diagram.

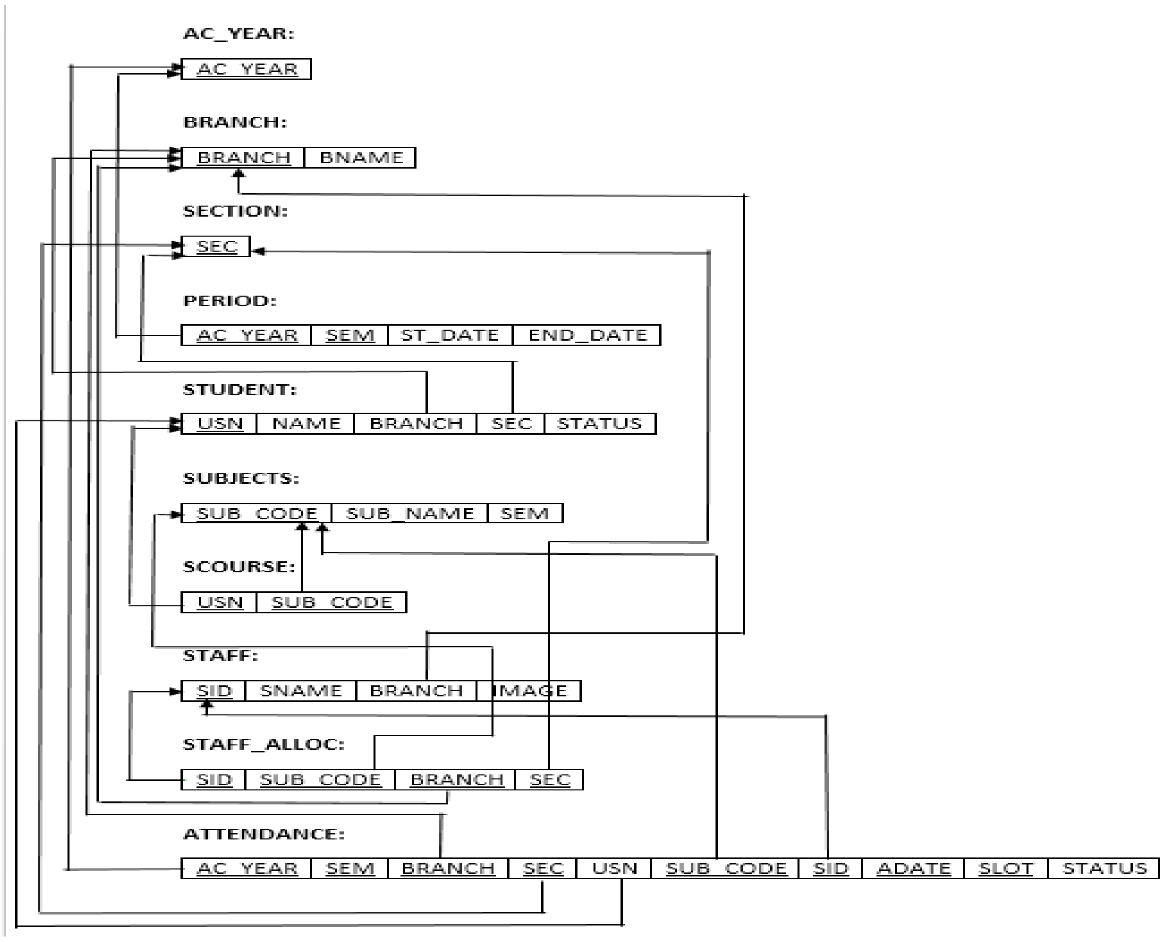

AC_YEAR:
AC_YEAR
BRANCH:
BRANCH | BNAME
SECTION:
SEC
PERIOD:
AC_YEAR | SEM | ST_DATE | END_DATE
STUDENT:
USN | NAME | BRANCH | SEC | STATUS
SUBJECTS:
SUB_CODE | SUB_NAME | SEM
SCOURSE:
USN | SUB_CODE
STAFF:
SID | SNAME | BRANCH | IMAGE
STAFF_ALLOC:
SID | SUB_CODE | BRANCH | SEC
ATTENDANCE:
AC_YEAR | SEM | BRANCH | SEC | USN | SUB_CODE | SID | ADATE | SLOT | STATUS

Fig. 2.2 Schema Diagram for SAS

Chapter 3

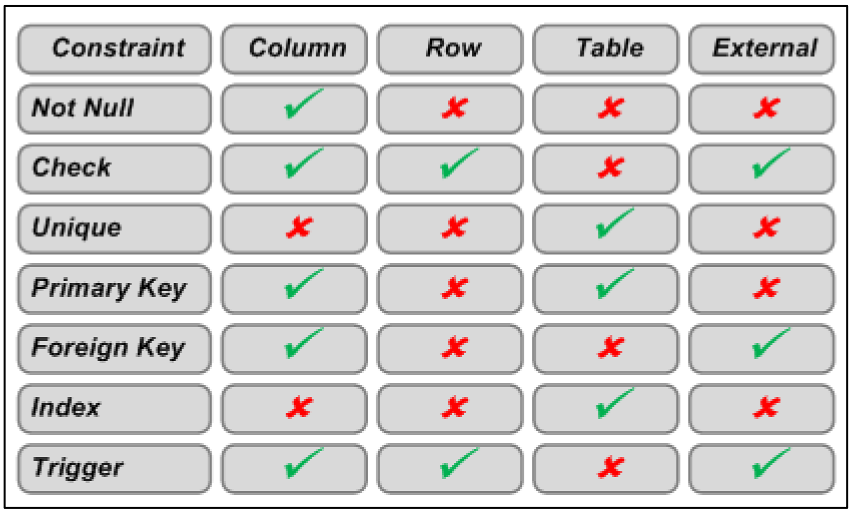
SYSTEM DESIGN

System Design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. System design could be seen as the application of systems theory to product development.

3.1 General Constraints

Constraints enforce limits to the data or type of data that can be inserted/updated/deleted from a table. The whole purpose of constraints is to maintain the **data integrity** during an update/delete/insert into a table.

Constraints could be either on a column level or a table level. The column level constraints are applied only to one column, whereas the table level constraints are applied to the whole table. Following are some of the most commonly used constraints available in SQL. While all databases support the set of possible constraints. They sometimes provide system specific alternatives. The following chart in Fig. 3.1 describes the types of restrictions you can place on columns and tables by using database-level constraints and triggers.



Constraint	Column	Row	Table	External
Not Null	✓	✗	✗	✗
Check	✓	✓	✗	✓
Unique	✗	✗	✓	✗
Primary Key	✓	✗	✓	✗
Foreign Key	✓	✗	✗	✓
Index	✗	✗	✓	✗
Trigger	✓	✓	✗	✓

Fig. 3.1 Various Database-level constraints

- NULL Constraint

Attributes that are under NOT NULL constraints have to be filled compulsorily. Almost all the attributes in the project are under NOT NULL constraint.

- Entity Integrity Constraint

  This constraint makes sure that no primary key can have a NULL value assigned to it. The primary keys involved in the project include:

  ▪ Username

  ▪ USN

  ▪ SID

  ▪ BID

  ▪ SEC

  ▪ AC_YEAR

- Referential Integrity Constraints

  A table in the back end of the project may have references pointing to an attribute in another table. For example: USN in the SCOURSE table refers to USN in the STUDENT table.

- Semantic Constraints

  These constraints are specified and enforced as per requirement.

3.2 System Block Diagram

Class diagrams are easier than they look. They're simply diagrams that describe the structure of a system by modeling its classes, attributes, operations, and relationships between objects as shown in Fig. 3.2.

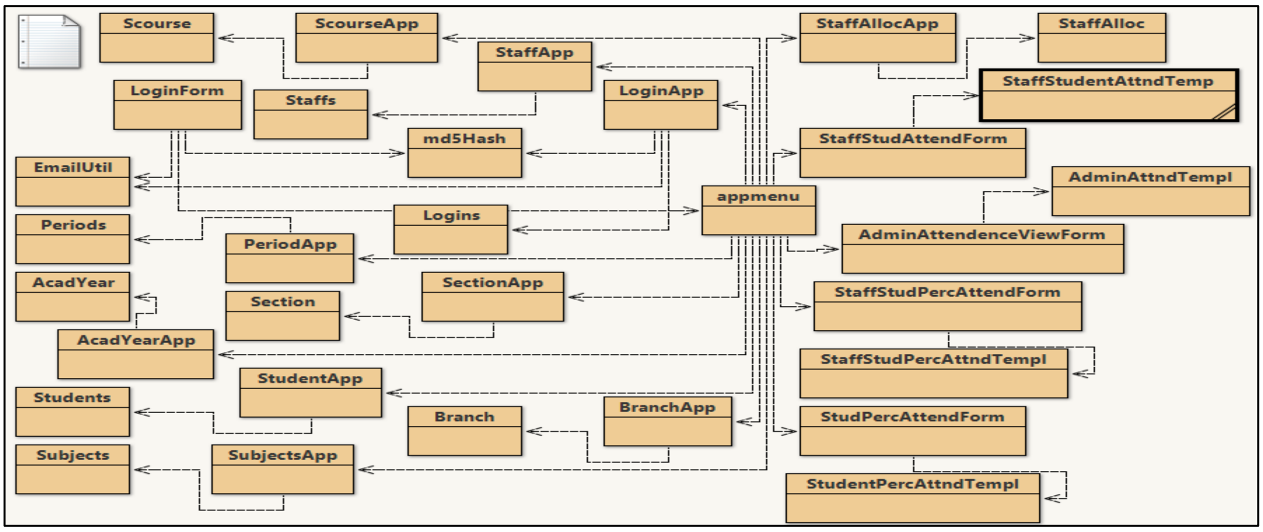


Fig. 3.2 Java Class Hierarchy Diagram for Student Attendance System

3.3 Table Description

A relational database is made up of several components, of which the table is most significant. In relational database terms, a *table* is responsible for storing data in the database. A database consists of one or more tables. Database tables consist of *rows* and *columns*. The SQL script used to create the tables required for the student attendance system is presented in this section. The parent Tables "ACADYR","BRANCH","SECTION", "PERIOD", "STUDENT", "SUBJECTS" and "STAFF" must be created first in that order and data must be configured by ADMIN.

Later the, the child tables like "SCOURSE", "STAFF_ALLOC", and "LOGIN" must be created and data must be configured by ADMIN. Lastly the "ATTENDANCE" must be created by ADMIN so that staff can login and generate and capture student attendance. The "AUDITLOGIN" is created to capture the audit trail of password resets by each user. The ATTENDANCE data is dynamic in nature and gets populated based on the daily attendance updated by the staff.

```
CREATE TABLE  "ACADYR"  ("AC_YEAR" NUMBER(4,0),
CONSTRAINT "PK_ACADYR" PRIMARY KEY ("AC_YEAR") ENABLE);

CREATE TABLE  "BRANCH" ("BRANCH" VARCHAR2(3),
"BNAME" VARCHAR2(40) NOT NULL ENABLE,
CONSTRAINT "PK_BRANCH" PRIMARY KEY ("BRANCH") ENABLE);

CREATE TABLE  "SECTION" ("SEC" CHAR(1),
CONSTRAINT "PK_SEC" PRIMARY KEY ("SEC") ENABLE);

CREATE TABLE  "PERIOD" ("AC_YEAR" NUMBER(4,0),
"SEM" NUMBER(1,0), "ST_DATE" DATE, "END_DATE" DATE,
CONSTRAINT "PK_PERIOD" PRIMARY KEY ("AC_YEAR", "SEM") ENABLE);

CREATE TABLE  "STUDENT"
("USN" VARCHAR2(10), "NAME" VARCHAR2(32), "SEC" CHAR(1),
"BRANCH" VARCHAR2(3), "STATUS" CHAR(2),
CONSTRAINT "PK_USN" PRIMARY KEY ("USN") ENABLE);

ALTER TABLE  "STUDENT" ADD CONSTRAINT "FK_SBRANCH" FOREIGN KEY
("BRANCH") REFERENCES  "BRANCH" ("BRANCH") ENABLE;
ALTER TABLE  "STUDENT" ADD CONSTRAINT "FK_SSECTION" FOREIGN KEY
("SEC") REFERENCES  "SECTION" ("SEC") ENABLE;

CREATE TABLE  "SUBJECTS" ("SUB_CODE" VARCHAR2(7),
"SUB_NAME" VARCHAR2(32),"SEM" NUMBER(1,0),
CONSTRAINT "PK_SUB_CODE" PRIMARY KEY ("SUB_CODE") ENABLE);
```

```
CREATE TABLE  "STAFF" ("SID" NUMBER(3,0), "SNAME" VARCHAR2(32),
"BRANCH" VARCHAR2(3), "IMAGE" BLOB DEFAULT NULL,
CONSTRAINT "PK_SID" PRIMARY KEY ("SID") ENABLE);

ALTER TABLE  "STAFF" ADD CONSTRAINT "FK_STFBRANCH" FOREIGN KEY
("BRANCH")REFERENCES  "BRANCH" ("BRANCH") ENABLE;

CREATE TABLE  "SCOURSE" ("USN" VARCHAR2(10), "SUB_CODE"
VARCHAR2(7),
CONSTRAINT "PK_USNSCODE" PRIMARY KEY ("USN", "SUB_CODE") ENABLE);

ALTER TABLE  "SCOURSE" ADD CONSTRAINT "FK_SCSUBCODE" FOREIGN KEY
("SUB_CODE") REFERENCES  "SUBJECTS" ("SUB_CODE") ENABLE;
ALTER TABLE  "SCOURSE" ADD CONSTRAINT "FK_SCUSN" FOREIGN KEY
("USN") REFERENCES  "STUDENT" ("USN") ENABLE;

CREATE TABLE  "STAFF_ALLOC" ("SID" NUMBER(3,0), "SUB_CODE"
VARCHAR2(7),"SEC" CHAR(1),"BRANCH" VARCHAR2(3) DEFAULT 'ISE' NOT
NULL ENABLE, CONSTRAINT "PK_SIDSCODESECBR"
PRIMARY KEY ("SID", "SUB_CODE", "SEC", "BRANCH") ENABLE);

ALTER TABLE  "STAFF_ALLOC" ADD CONSTRAINT "FK_SIDBRANCH" FOREIGN
KEY ("BRANCH") REFERENCES  "BRANCH" ("BRANCH") ENABLE;
ALTER TABLE  "STAFF_ALLOC" ADD CONSTRAINT "FK_STFSCODE" FOREIGN
KEY ("SUB_CODE") REFERENCES  "SUBJECTS" ("SUB_CODE") ENABLE;
ALTER TABLE  "STAFF_ALLOC" ADD CONSTRAINT "FK_STFSEC" FOREIGN KEY
("SEC") REFERENCES  "SECTION" ("SEC") ENABLE;
ALTER TABLE  "STAFF_ALLOC" ADD CONSTRAINT "FK_STFSID" FOREIGN KEY
("SID") REFERENCES  "STAFF" ("SID") ENABLE;

CREATE TABLE  "ATTENDANCE" ("AC_YEAR" NUMBER(4,0), "SEM"
NUMBER(1,0),"BRANCH" VARCHAR2(3), "SEC" CHAR(1),
"SUB_CODE" VARCHAR2(7), "USN" VARCHAR2(10),"SID" NUMBER(3,0),
"ADATE" DATE, "SLOT" NUMBER(1,0), "STATUS" VARCHAR2(2),
CONSTRAINT "PK_ATTENDANCE" PRIMARY KEY
("AC_YEAR","SEM","BRANCH","SEC","SUB_CODE","SID", "ADATE",
"SLOT") ENABLE);
```

```
ALTER TABLE  "ATTENDANCE" ADD CONSTRAINT "FK_ATTBRANCH" FOREIGN
KEY ("BRANCH") REFERENCES  "BRANCH" ("BRANCH") ENABLE;
ALTER TABLE  "ATTENDANCE" ADD CONSTRAINT "FK_ATTSCODE" FOREIGN
KEY ("SUB_CODE")REFERENCES  "SUBJECTS" ("SUB_CODE") ENABLE;
ALTER TABLE  "ATTENDANCE" ADD CONSTRAINT "FK_ATTSEC" FOREIGN KEY
("SEC") REFERENCES  "SECTION" ("SEC") ENABLE;
ALTER TABLE  "ATTENDANCE" ADD CONSTRAINT "FK_ATTSID" FOREIGN KEY
("SID") REFERENCES  "STAFF" ("SID") ENABLE;

ALTER TABLE  "ATTENDANCE" ADD CONSTRAINT "FK_PERYRSEM" FOREIGN
KEY ("AC_YEAR", "SEM")REFERENCES  "PERIOD" ("AC_YEAR", "SEM")
ENABLE;

CREATE TABLE  "LOGIN" ("USERNAME" VARCHAR2(50), "PASSWORD"
VARCHAR2(15), "ENCRYPTEDPASS" VARCHAR2(50),"SID" NUMBER(3,0),
"USN" VARCHAR2(10), "USRROLE" VARCHAR2(20),"RSTCOUNT" NUMBER(4,0)
DEFAULT 0 NOT NULL ENABLE,
CONSTRAINT "PK_USERNAME" PRIMARY KEY ("USERNAME") ENABLE);

ALTER TABLE  "LOGIN" ADD CONSTRAINT "FK_LOGSID" FOREIGN KEY
("SID") REFERENCES  "STAFF" ("SID") ENABLE;
ALTER TABLE  "LOGIN" ADD CONSTRAINT "FK_LOGUSN" FOREIGN KEY
("USN") REFERENCES  "STUDENT" ("USN") ENABLE;

CREATE TABLE  "AUDITLOGIN" ("LOGINID" VARCHAR2(50) NOT NULL
ENABLE,  "TIMESTAMP" DATE NOT NULL ENABLE);
```

3.4 View Description

A database view is a virtual table or logical table which is defined as a **SQL SELECT** query with or without joins. Because a database view is similar to a database table, which consists of rows and columns, so you can query data against it. Most database management systems, including allow you to update data in the underlying tables through the database view with some prerequisites. A database view is dynamic because it is not related to the physical schema. Some of the advantages of views are (a) It allows user to simplify complex queries (b) It helps limit data access to specific users (c) It provides extra security layer (d) It enables computed columns (e) It enables backward compatibility. In this SAS project the views have been used for the reason (a) to (d).

```
CREATE OR REPLACE VIEW  "V1" ("USN", "SUB_CODE", "AC_YEAR",
"SEM", "SEC", "PRESENT") AS
SELECT USN,SUB_CODE,AC_YEAR,SEM,SEC,COUNT(*) AS PRESENT FROM
ATTENDANCE  WHERE STATUS='P' AND USN IN
(SELECT USN FROM STUDENT WHERE STATUS='AC')
GROUP BY USN,SUB_CODE,AC_YEAR,SEM,SEC;

CREATE OR REPLACE VIEW  "V2" ("USN", "SUB_CODE", "AC_YEAR",
"SEM", "SEC", "TOTAL") AS
SELECT USN,SUB_CODE,AC_YEAR,SEM,SEC,COUNT(*) AS TOTAL FROM
ATTENDANCE WHERE USN IN
(SELECT USN FROM STUDENT WHERE STATUS='AC')
GROUP BY USN,SUB_CODE,AC_YEAR,SEM,SEC;

CREATE OR REPLACE VIEW  "V3" ("USN", "SUB_CODE", "AC_YEAR",
"SEM", "SEC", "PERC") AS
SELECT USN,SUB_CODE,AC_YEAR,SEM,SEC,ROUND((PRESENT/TOTAL)*100) AS
PERC FROM V1 NATURAL JOIN V2 ORDER BY SUB_CODE;
```

3.5 Trigger

A Trigger is used to update a table, as a result of a state change of another table; here, record the timestamp, corresponding to each password reset done by a user.

```
CREATE OR REPLACE TRIGGER AUDIT_LOGINS
AFTER UPDATE ON LOGIN
FOR EACH ROW
BEGIN
INSERT INTO AUDITLOGIN (LOGINID,TIMESTAMP) VALUES
(:OLD.USERNAME,SYSDATE);
END;
```

3.6 Stored Procedure

A Stored Procedure is used to perform a repetitive task; here, to generate a day's attendance. The prepareCall() and executeUpdate() methods are used to make a call to a Stored Procedure

```
CREATE OR REPLACE FUNCTION GEN_ATTENDANCE(
V_AC_YEAR IN ATTENDANCE.AC_YEAR%TYPE,
V_SEM IN ATTENDANCE.SEM%TYPE,
V_BRANCH IN ATTENDANCE.BRANCH%TYPE,
V_SEC IN ATTENDANCE.SEC%TYPE,
V_SUB_CODE IN ATTENDANCE.SUB_CODE%TYPE,
V_SID IN ATTENDANCE.SID%TYPE,
V_ADATE IN ATTENDANCE.ADATE%TYPE,
V_SLOT IN ATTENDANCE.SLOT%TYPE,
V_STATUS IN ATTENDANCE.STATUS%TYPE)
RETURN VARCHAR2
IS PRAGMA AUTONOMOUS_TRANSACTION;
V_NUM NUMBER(6,0):=0;
V_DUP NUMBER(6,0):=0;
V_MEM NUMBER(6,0):=0;
V_SAME NUMBER(6,0):=0;
BEGIN
SELECT COUNT(USN) INTO V_NUM FROM ATTENDANCE
WHERE
AC_YEAR=V_AC_YEAR AND SEM=V_SEM AND BRANCH=V_BRANCH AND
SEC=V_SEC AND SUB_CODE=V_SUB_CODE AND SID=V_SID AND
ADATE=V_ADATE AND SLOT=V_SLOT;

SELECT COUNT(USN) INTO V_DUP FROM ATTENDANCE
WHERE AC_YEAR=V_AC_YEAR AND SEM=V_SEM  AND SID=V_SID AND
SLOT=V_SLOT AND ADATE=V_ADATE AND (SID,SLOT,ADATE) IN
(SELECT SID, SLOT, ADATE FROM ATTENDANCE
WHERE AC_YEAR=V_AC_YEAR AND SEM=V_SEM);

SELECT COUNT(*) INTO
V_MEM FROM SUBJECTS A, STAFF_ALLOC B,PERIOD P
WHERE A.SUB_CODE=B.SUB_CODE AND A.SEM=V_SEM
AND A.SUB_CODE=V_SUB_CODE AND B.BRANCH=V_BRANCH AND B.SEC=V_SEC
AND B.SID=V_SID AND P.SEM=A.SEM AND P.AC_YEAR=V_AC_YEAR
AND V_ADATE<=SYSDATE AND SYSDATE BETWEEN P.ST_DATE AND P.END_DATE
AND V_ADATE BETWEEN P.ST_DATE AND P.END_DATE;
```

```
SELECT COUNT(*) INTO V_SAME FROM ATTENDANCE
WHERE AC_YEAR=V_AC_YEAR AND SEM=V_SEM AND BRANCH=V_BRANCH
AND SEC=V_SEC AND SLOT=V_SLOT AND ADATE=V_ADATE;

IF (V_NUM > 0) THEN  RETURN 'DUPL';
ELSIF ((V_DUP > 0) OR (V_SAME > 0)) THEN   RETURN 'CLASH';
ELSIF (V_MEM=0) THEN  RETURN 'MISM';
ELSE
INSERT INTO
ATTENDANCE(AC_YEAR,SEM,BRANCH,SEC,SUB_CODE,USN,SID,ADATE,SLOT,STA
TUS,STATUS1)
SELECT A.AC_YEAR, A.SEM, UPPER(B.BRANCH),UPPER(B.SEC),
UPPER(B.SUB_CODE), UPPER(C.USN),B.SID,V_ADATE ADATE,V_SLOT SLOT,
'P' STATUS,1 STATUS1
FROM PERIOD A, STAFF_ALLOC B, SCOURSE C, STUDENT D
WHERE
A.AC_YEAR = V_AC_YEAR AND A.SEM = V_SEM AND B.BRANCH = V_BRANCH
AND B.SEC = D.SEC AND B.SEC = V_SEC AND B.SUB_CODE = C.SUB_CODE
AND B.SUB_CODE = V_SUB_CODE AND B.SID = V_SID
AND C.USN = D.USN AND UPPER(D.STATUS)='AC';
COMMIT;
RETURN 'PASS';
END IF;
END;
```

Chapter 4

IMPLEMENTATION

Implementation is the process of: defining how the system should be built, ensuring that it is operational and meets quality standards. It is a systematic and structured approach for effectively integrating a software based service or component into the requirements of end users.

4.1 Selection of the platform

The platform selected for the project is Windows as it is compatible with Java in the front end, Oracle in the back end. Eclipse and BlueJ has been used as the IDE.

4.2 Selection of the programming language

Java and SQL are used in programming between back end and front end.

JAVA: Java technology is both a programming language and a platform. The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Object oriented
- Distributed
- Multithreaded
- Dynamic
- Architecture neutral
- Portable
- High performance
- Robust
- Secure

Like any programming language, the Java language has its own structure, syntax rules, and programming paradigm. The Java language's programming paradigm is based on the concept of OOP, which the language's features support

The Java *platform* is the hardware or software environment (Microsoft Windows, Linux, Solaris OS, and Mac OS) in which a program runs. Most platforms can be described as a combination of the operating system and underlying hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms. The Java platform has two components:

- The Java Virtual Machine (JVM)
- The Java Application Programming Interface (API)

SQL (pronounced "ess-que-el") stands for Structured Query Language. SQL is used to communicate with a database. It is a standardized programming language used for managing relational databases and performing various operations on the data in them. The uses of SQL include modifying database table and index structures; adding, updating and deleting rows of data; and retrieving subsets of information from within a database for transaction processing and analytics applications. Queries and other SQL operations take the form of commands written as statements-- commonly used SQL statements include select, add, insert, update, delete, create, alter and truncate.

4.2.1 Front End: Java and SQL

The programming language used for the development work is Java. The reason for selection of this language includes among many others the following few.

- In the development of complex, interactive applications, Java can be used to embed rich GUI using swing components.
- Java enables designers to reuse, the portable code.
- Java enables designer to use rich form elements. There will be different types of text inputs and different Swing components implemented for different purposes. Some of the form elements used in this SAS project are JTextField, JComboBox, JButtons, JPasswordField, JLabel, JTable, JFileChooser, JPanel, JFrame, JScrollPane, JOptionPane. The Layouts used are panel and frames are GridLayout and BorderLayout.

4.2.2 Back End: PL/SQL and ORACLE

Oracle is a relational data base management system (RDBMS) that runs as a server providing multi-user access to a number of databases.

- Oracle 11G Express Edition is an open source tool.
- Oracle is a popular choice of database for use in web and stand-alone applications.
- Oracle 11G Express Edition provides web-based tools to administer databases and manage data contained within.
- Oracle 11G Express Edition provides an ease in creating tables by a graphical as well as query based interface.
- Oracle PL/SQL Function and Database Triggers are executed based on the functionality.
- SQL statements are executed against one or more database tables based on the front end GUI functionality.

Oracle implements the following feature, which some other RDBMS systems may not:

- Commit grouping, gathering multiple transactions from multiple connections together to increase the number of commits per second.

4.2.3 Transaction Concepts

A **transaction** is a sequence of operations performed as a single logical unit of work. A logical unit of work must exhibit four properties, called the atomicity, consistency, isolation, and durability (ACID) properties, to qualify as a transaction.

- ResultSets are used as Cursors for record-at-a-time retrieval, of results of execution of SQL queries.
- The next() function is used to access 1 tuple of a result , at a time.
- It is usually used in a "while" loop to access all the tuples retrieved.
- The executeQuery() function is used to execute a "SELECT" statement.
- It returns a pointer to the set of tuples retrieved as the result.
- The executeUpdate() function is used to execute "INSERT" , "DELETE" and "UPDATE".
- It returns the number of rows affected by the execution of any 1 of the above statements.

4.3 Programming Coding Guidelines-Java, PL SQL

Following Code guidelines are important to programmers for a number of reasons:

- 80% of the lifetime cost of a piece of software goes to maintenance.
- Hardly any software is maintained for its whole life by the original author.
- Code conventions improve the readability of the software, allowing engineers to understand new code more quickly and thoroughly.
- If you ship your source code as a product, you need to make sure it is as well packaged and clean as any other product you create.

4.3.1 Java with PL SQL Coding Guidelines

4.3.1.1 Naming Conventions

- Each identifier name follows the conventions for naming identifiers in Java.
- The names given to all attributes and tables give the meaning of the entity they represent.
- The names containing more than one word use an underscore to separate the two names. For eg, STAFF_ALLOC table etc.

4.3.1.2 Methods Usage for Database access

- Use direct queries in front end.

- Usage of classes containing in-built methods to connect to the backend databases and execute queries.
- This improves flexibility as it can be ported to other platforms with no change of code.

4.3.1.3 Button click coding

- As soon as the button is clicked a set of code are executed before forwarding the user to the next module.

4.3.2 Discussion Of Code Segments

4.3.2.1 Including Braces

Braces should **always** be included when writing code using `if`, `for`, `while` etc. blocks. There are **no exceptions** to this rule, even if the braces could be omitted. Leaving out braces makes code harder to maintain in the future and can also cause bugs that are very difficult to track down.

4.3.2.2 Where to put the braces

Braces should always be placed on a line on their own; again there are no exceptions to this rule. Braces should also align properly (use tabs to achieve this) so a closing brace is always in the same column as the corresponding opening brace.

4.3.2.3 Spaces between Tokens

There should always be one space on either side of a token in expressions, statements etc. The only exceptions are commas (which should have one space after, but none before), semi-colons (which should not have spaces on either side if they are at the end of a line, and one space after otherwise). Methods should follow the rules laid out already, i.e. no spaces between the method name and the opening bracket and no space between the brackets and the arguments, but one space between each argument.

4.3.2.4 Control Structures

These include if, if –else, for, while etc. control statements should have one space between the control keyword and opening parenthesis, to distinguish them from method calls. You are strongly encouraged to always use curly braces even in situations where they are technically optional. Having them increases readability and decreases the likelihood of logic errors being introduced when new lines are added.

4.3.2.5 Method Calls

Methods should be called with no spaces between the method name, the opening parenthesis, and the first parameter; spaces between commas and each parameter, and no space between the last parameter, the closing parenthesis, and the semicolon.

4.3.2.6 Comments

Complete inline documentation comment blocks (docblocks) must be provided. Non-documentation comments are strongly encouraged. C style comments (/* */) and standard C++ comments (//) are both fine. The use of Perl / shell style comments (#) is strongly discouraged.
A detailed discussion on GUI design and few important code functionality implemented in JAVA and PL/SQL are discussed in chapter 6 and 7.

4.4 Testing

Software Testing is the process used to help identify the correctness, completeness, security and quality of the developed computer software. Testing is the process of technical investigation and includes the process of executing a program or application with the intent of finding errors.

4.4.1 Test Strategies

Test strategy tells the test plan of the project. It also tells how to test and what to test. The testing done in this project are Unit testing and Integration testing.

- Features to be tested: Form navigation and generation of reports.
- Items to be tested: Functioning of forms and buttons.
- Purpose of testing: To check the effective working of SAS.
- Pass / Fail Criteria: Changes made on the back end like recreation of tables should affect the front end as well. If so, the test is successful.
- Assumptions and Constraints: Tables should be created and values have to be entered at the back end before testing and entity integrity and referential integrity constraints should be taken care.

4.4.2 Unit Testing

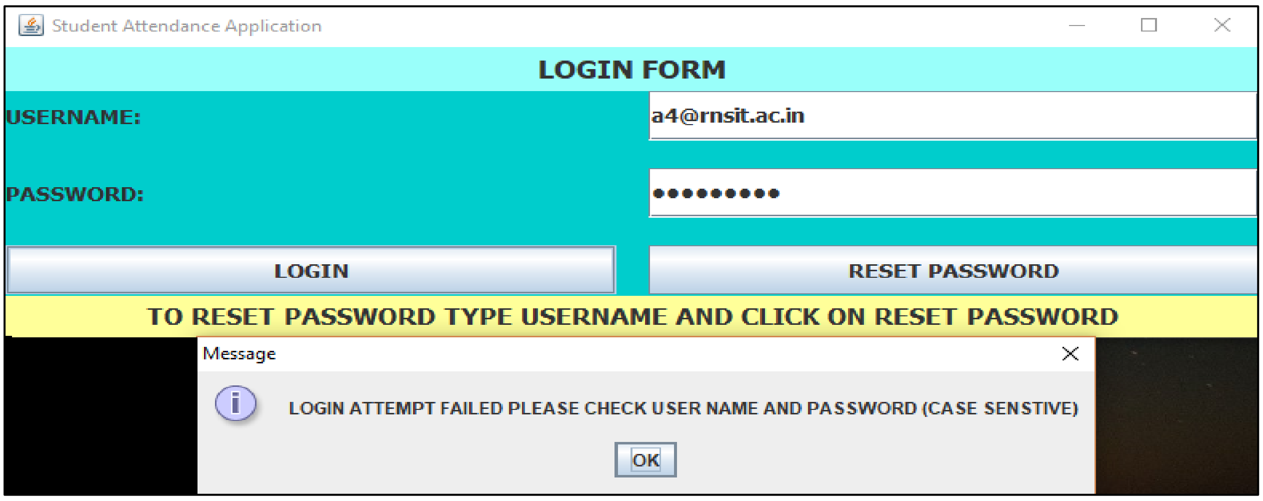
Unit testing is a software verification and validation method in which a programmer tests if individual units of source code are fit for use. Some of the tests performed in the project are insert, delete, retrieve and modify.

4.4.2.1 Unit Test Case1

This Test Case checks if the User has entered a valid Username and Password (Case Sensitive). On clicking the LOGIN button, the Username and password entered are validated against the stored value in database. The password entered is encrypted and compared against the encrypted password stored in the Login table against the username. If either Username or Password or both does not match, the "Login Attempt Failed" message is displayed in the Dialog box as shown in Fig. 4.1.

Table 4.1 Unit test case for Login Information Check operation

Sl No. of test case :	1
Name of test :	Check test
Item / Feature being tested :	Check button
Sample Input :	USERNAME='a4@rnsit.ac.in', PASSWORD='dfdsvjghj' Upon mouse click on button.
Expected output :	Valid if login information is correct if not "Login Attempt Failed Please check Username and Password(Case Sensitive)" appears.
Actual output :	Message: Login Attempt Failed Please check Username and Password(Case Sensitive)
Remarks :	Test succeeded


Student Attendance Application

LOGIN FORM

USERNAME: a4@rnsit.ac.in

PASSWORD: •••••••••

LOGIN RESET PASSWORD

TO RESET PASSWORD TYPE USERNAME AND CLICK ON RESET PASSWORD

Message

LOGIN ATTEMPT FAILED PLEASE CHECK USER NAME AND PASSWORD (CASE SENSTIVE)

OK

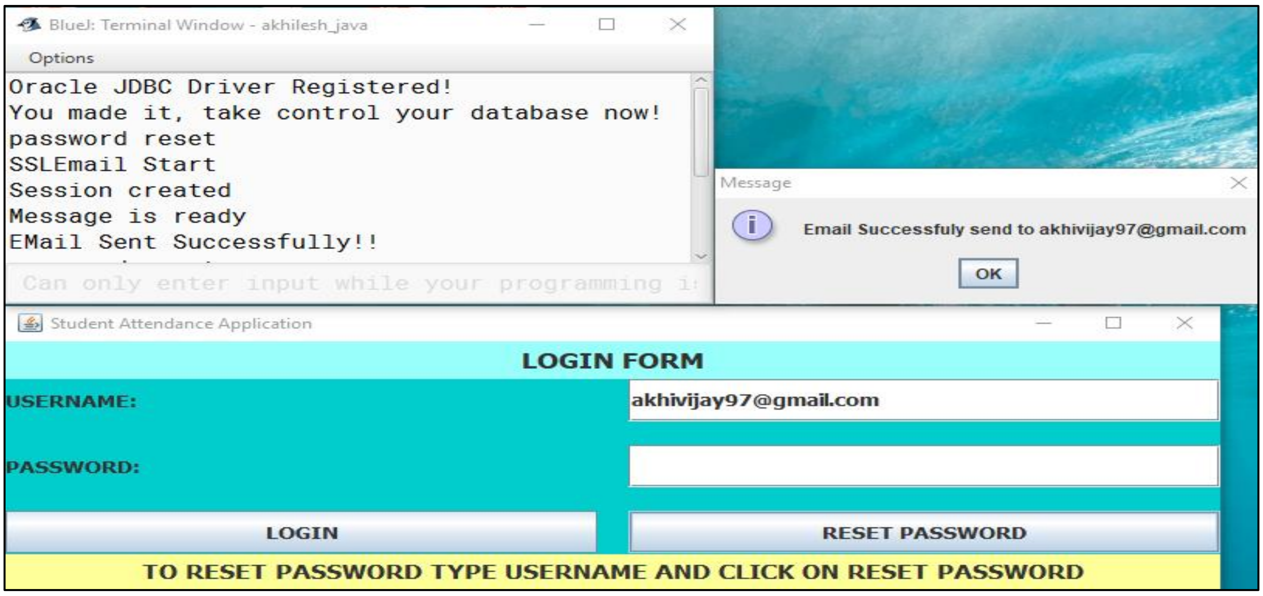
Fig. 4.1 Unit test case for Login Information Check operation

4.4.2.2 Unit Test Case2

This Test Case checks to see if the User has entered a valid Username, to send the reset password via email. On clicking the RESET PASSWORD button, a random 8 character password string is generated by the system against the user and is sent to the user along with message via email and a success message is displayed as a Dialog box as in Fig. 4.2. Further, the user's new password and its encrypted form are updated in the Login table.

Table 4.2 Unit test case for Code Verification For Reset Password

Sl No. of test case :	2
Name of test :	Verify Code
Item / Feature being tested :	Textbox and Reset Password Button
Sample Input :	Valid Username
Expected output :	Random password sent to email
Actual output :	Random password sent to email
Remarks :	Test succeeded


BlueJ: Terminal Window - akhilesh_java
Options

```
Oracle JDBC Driver Registered!
You made it, take control your database now!
password reset
SSLEmail Start
Session created
Message is ready
EMail Sent Successfully!!
```

Can only enter input while your programming i
Message

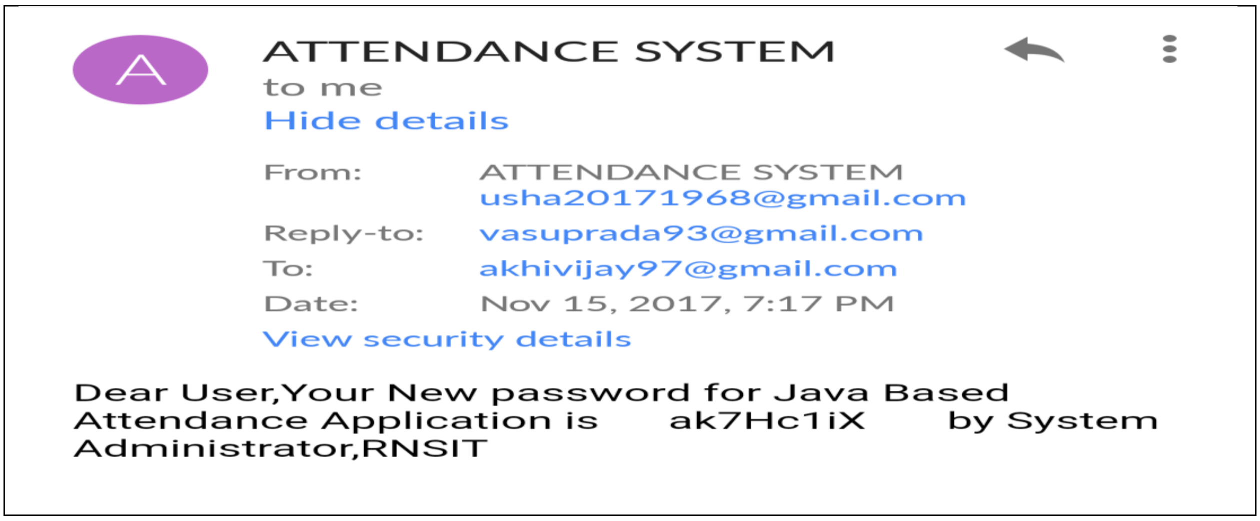
Email Successfuly send to akhivijay97@gmail.com

OK
Student Attendance Application
LOGIN FORM

USERNAME: akhivijay97@gmail.com

PASSWORD:

LOGIN
RESET PASSWORD
TO RESET PASSWORD TYPE USERNAME AND CLICK ON RESET PASSWORD



ATTENDANCE SYSTEM
to me

From: ATTENDANCE SYSTEM usha20171968@gmail.com
Reply-to: vasuprada93@gmail.com
To: akhivijay97@gmail.com
Date: Nov 15, 2017, 7:17 PM

Dear User,Your New password for Java Based Attendance Application is ak7Hc1iX by System Administrator,RNSIT

Fig. 4.2 Unit test case for Code Verification For Reset Password

4.4.3 Integration Testing

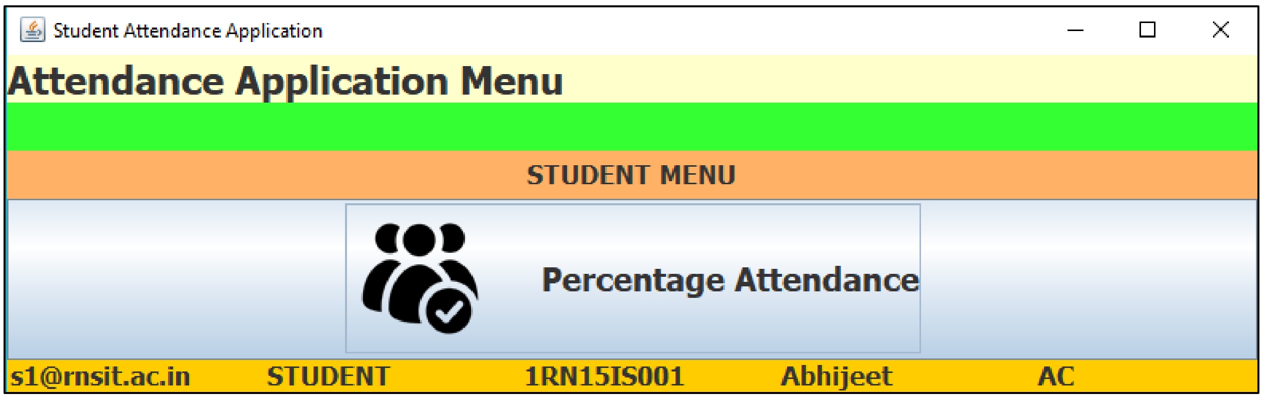
Integration testing is the phase in which individual functional modules are combined and tested as a group. It is done after unit testing and before validation testing. It is performed to expose defects in the interfaces and in the interactions between integrated components or systems.

4.4.3.1 Integration Test Case1

The Test Case checks the Navigation to the Student Menu when a Student enters a valid Username and Password. If both fields are valid, then a separate window for Student Menu appears on the screen with Percentage Attendance button along with student details as shown in Fig. 4.3.

Table 4.3 Integrated test case for Navigation to Student Menu

Sl No. of test case :	1
Name of test :	Navigation to Menu based on User role
Item / Feature being tested :	Login Button
Sample Input :	Valid Student Username and Password
Expected output :	Student Menu appears
Actual output :	Student Menu appears
Remarks :	Test succeeded



s1@rnsit.ac.in STUDENT 1RN15IS001 Abhijeet AC

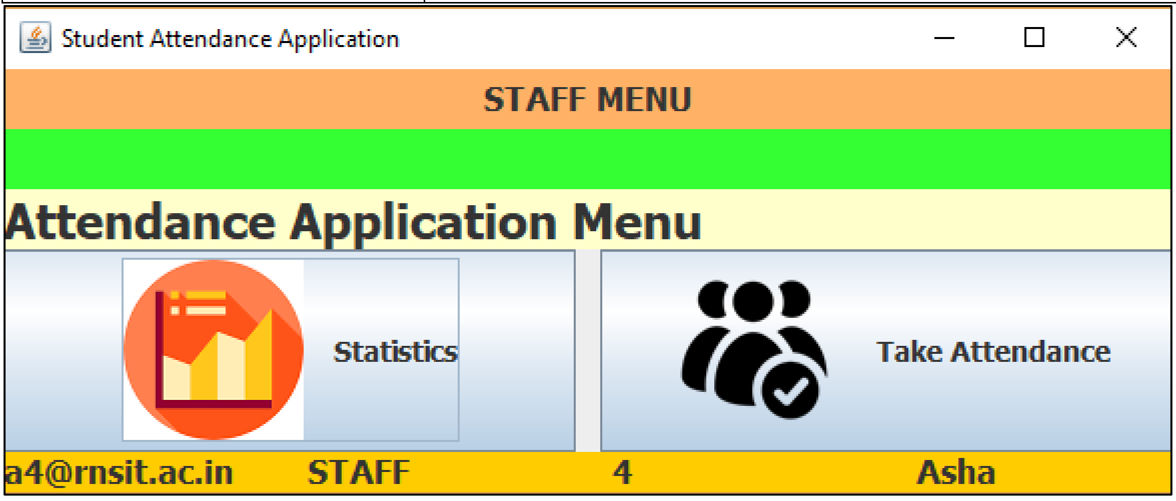
Fig. 4.3 Integrated test case for Navigation to Student Menu

4.4.3.2 Integration Test Case2

In this Test Case Navigation to the Staff Menu is checked, when a Staff enters a valid Username and Password. On successful login, the Staff Menu is presented on the screen as separate window along with two menu options, namely statistics and student attendance along with staff details as in Fig. 4.4.

Table 4.4 Integrated test case for Navigation to Staff Menu

Sl No. of test case :	2
Name of test :	Navigation to Staff Menu
Item / Feature being tested :	Login Button
Sample Input :	Valid Staff Username and Password
Expected output :	Staff Menu appears
Actual output :	Staff Menu appears
Remarks :	Test succeeded


Student Attendance Application

STAFF MENU

Attendance Application Menu

a4@rnsit.ac.in STAFF 4 Asha

Fig. 4.4 Integrated test case for Navigation to Staff Menu

4.4.3.3 Integration Test Case3

The Test Case checks the Navigation to the Admin Menu when an Admin enters a valid Username and Password. If both fields are valid, then the Admin Menu with eleven administrative functionality is presented in separate window along with user details as in Fig. 4.5.

Table 4.5 Integrated test case for Navigation to Admin Menu

Sl No. of test case :	3
Name of test :	Navigation to Admin Menu
Item / Feature being tested :	Login Button
Sample Input :	Valid Admin Username and Password
Expected output :	Admin Menu appears
Actual output :	Admin Menu appears
Remarks :	Test succeeded

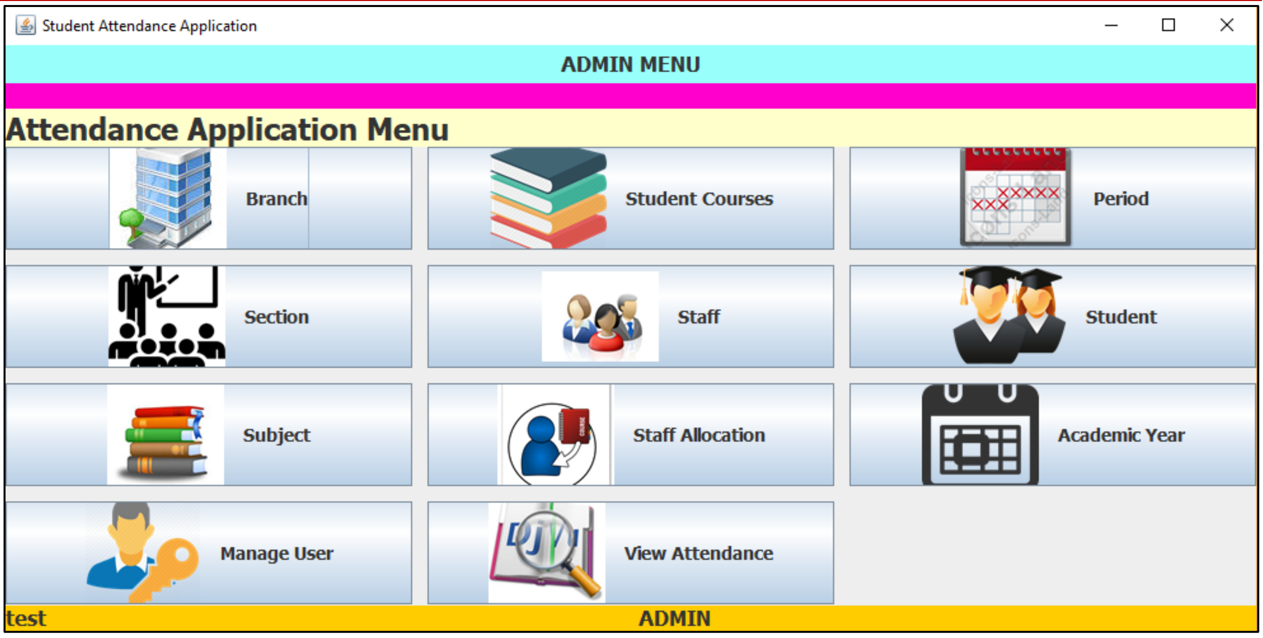


Fig. 4.5 Unit test case for Navigation to Admin Menu

4.4.4 System Testing

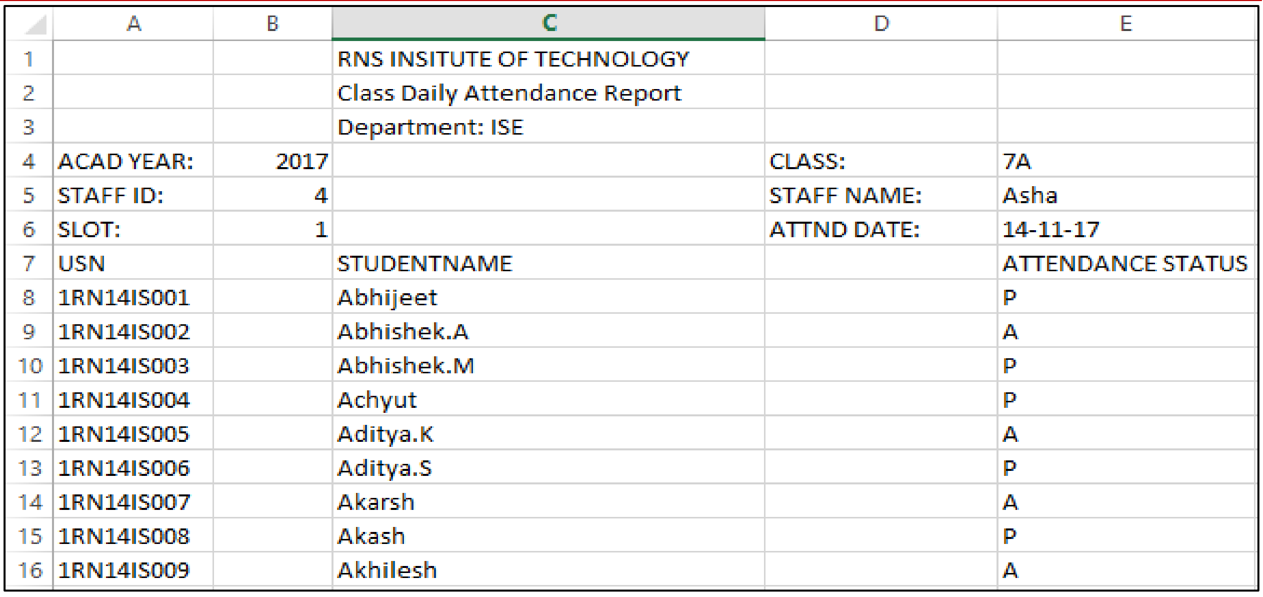
System Testing is a level of the software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements.

4.4.4.1 System Test Case1

The Test Case tests the overall system functionality by generating reports. If the data entered through the Staff Attendance Form is valid, then the Daily Class Attendance report is generated and presented in an Excel spreadsheet.

Table 4.6 System test case for overall system check

Sl No. of test case :	1
Name of test :	System overall test
Item / Feature being tested :	Generate Report button
Sample Input :	Data entered through Staff Attendance Form
Expected output :	Daily Class Attendance displayed in generated reports
Actual output :	Daily Class Attendance displayed in generated reports
Remarks :	Test succeeded



	A	B	C	D	E
1			RNS INSITUTE OF TECHNOLOGY		
2			Class Daily Attendance Report		
3			Department: ISE		
4	ACAD YEAR:	2017		CLASS:	7A
5	STAFF ID:	4		STAFF NAME:	Asha
6	SLOT:	1		ATTND DATE:	14-11-17
7	USN		STUDENTNAME		ATTENDANCE STATUS
8	1RN14IS001		Abhijeet		P
9	1RN14IS002		Abhishek.A		A
10	1RN14IS003		Abhishek.M		P
11	1RN14IS004		Achyut		P
12	1RN14IS005		Aditya.K		A
13	1RN14IS006		Aditya.S		P
14	1RN14IS007		Akarsh		A
15	1RN14IS008		Akash		P
16	1RN14IS009		Akhilesh		A

Fig. 4.6 System test case for overall system check

Furthermore, staff class percentage attendance report and admin raw attendance back up report in excel format can be generated using SAS.

4.5 Discussion Of Results

4.5.1 Snapshots

The Login box authenticates users by their usernames (case-insensitive) and password (case-sensitive). It also allows a user to reset his password, which will be sent to his email. The Username is compared with that stored in the Login table. The password entered is first encrypted and this encrypted form is compared with that stored in the Login table.

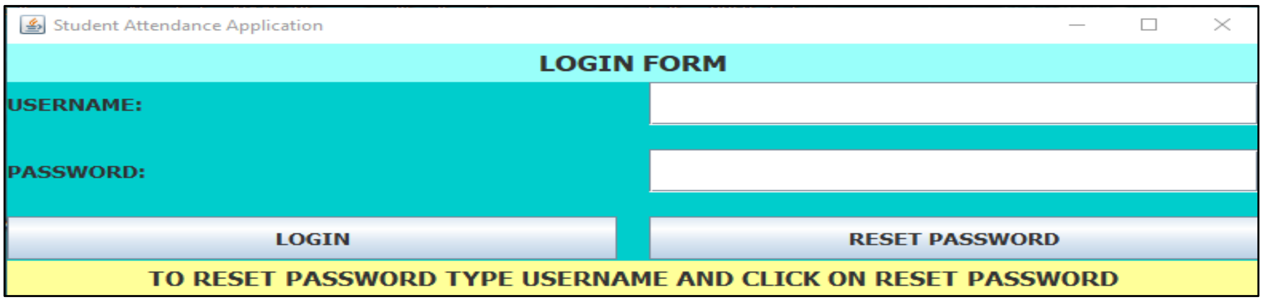
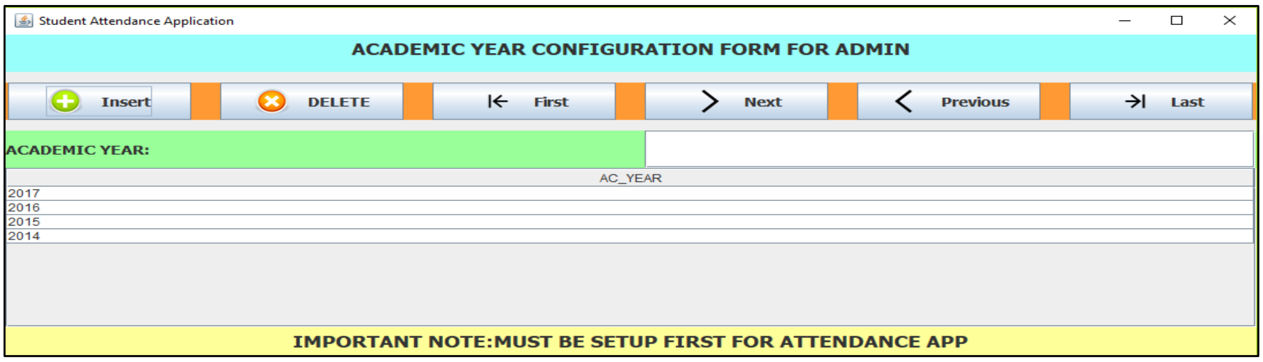

Student Attendance Application
LOGIN FORM
USERNAME:
PASSWORD:
LOGIN
RESET PASSWORD
TO RESET PASSWORD TYPE USERNAME AND CLICK ON RESET PASSWORD

Fig. 4.7 Snapshot of User Login Page

The Admin has the privilege to view, insert and delete the entries in the Academic year table. While viewing the entries, the First, Last, Previous and Next buttons can be used for effective navigation.

This table is one of the Parent tables. The entry to be inserted or deleted is identified by the ACADEMIC YEAR entered.



ACADEMIC YEAR CONFIGURATION FORM FOR ADMIN

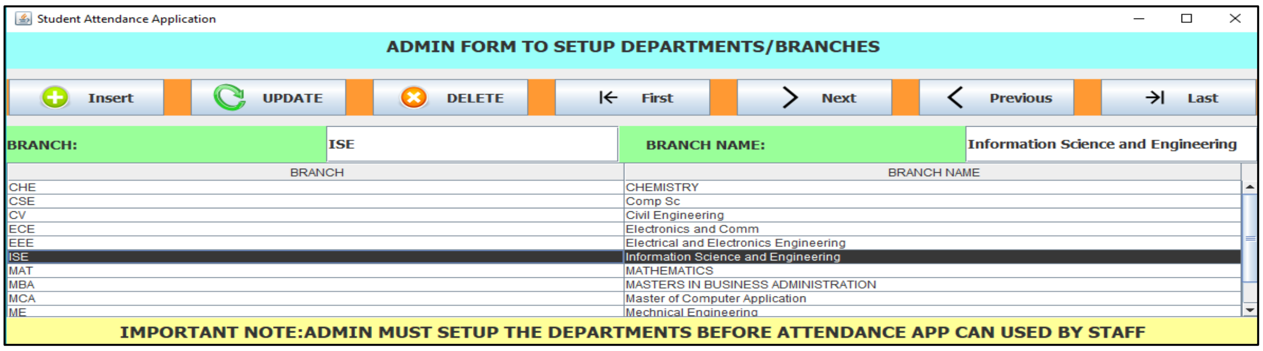
ACADEMIC YEAR:	

AC_YEAR
2017
2016
2015
2014

IMPORTANT NOTE:MUST BE SETUP FIRST FOR ATTENDANCE APP

Fig. 4.8 Snapshot of Admin's Academic-Year table

The Admin has the privilege to view, insert, delete and update the entries in the Branch table. While viewing the entries, the First, Last, Previous and Next buttons can be used for effective navigation. This table is one of the Parent tables. All fields must be filled to insert a new entry or update an existing one. On update, the BRANCH NAME of the corresponding BRANCH will be updated. The entry to be deleted is identified by the BRANCH entered.



ADMIN FORM TO SETUP DEPARTMENTS/BRANCHES

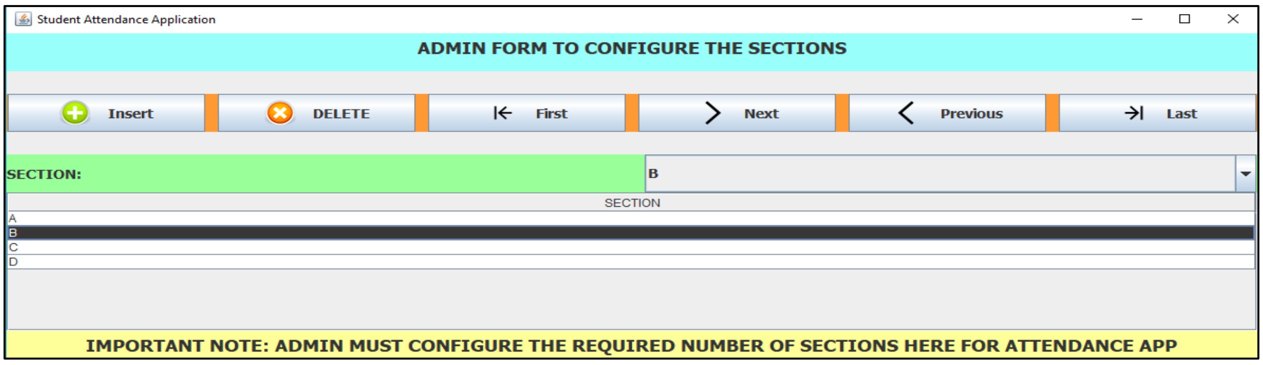
BRANCH:	ISE	BRANCH NAME:	Information Science and Engineering

BRANCH	BRANCH NAME
CHE	CHEMISTRY
CSE	Comp Sc
CV	Civil Engineering
ECE	Electronics and Comm
EEE	Electrical and Electronics Engineering
ISE	Information Science and Engineering
MAT	MATHEMATICS
MBA	MASTERS IN BUSINESS ADMINISTRATION
MCA	Master of Computer Application
ME	Mechnical Engineering

IMPORTANT NOTE:ADMIN MUST SETUP THE DEPARTMENTS BEFORE ATTENDANCE APP CAN USED BY STAFF

Fig. 4.9 Snapshot of Admin's Branch table

The Admin has the privilege to view, insert and delete the entries in the Section table. While viewing the entries, the First, Last, Previous and Next buttons can be used for effective navigation. This table is one of the Parent tables. SECTION field must be filled to insert a new entry. The entry to be deleted is identified by the SECTION entered.



ADMIN FORM TO CONFIGURE THE SECTIONS

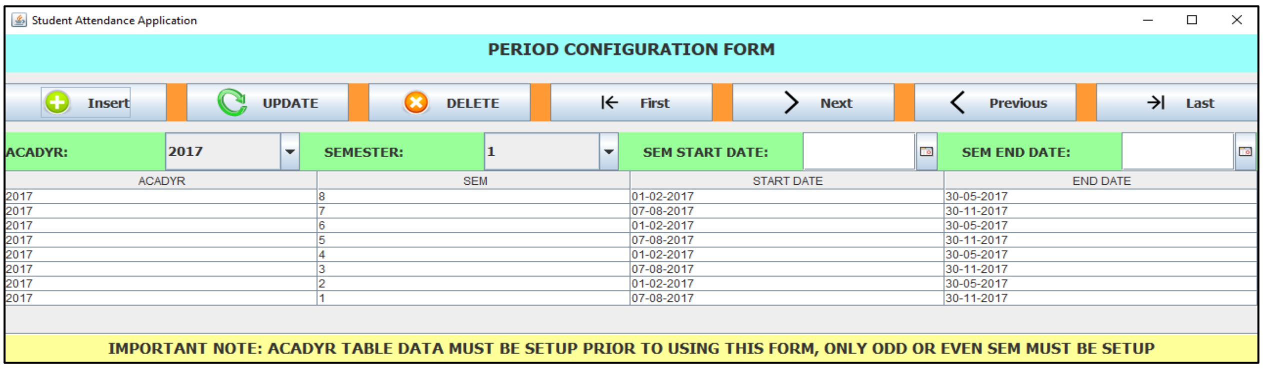
SECTION:	B

SECTION
A
B
C
D

IMPORTANT NOTE: ADMIN MUST CONFIGURE THE REQUIRED NUMBER OF SECTIONS HERE FOR ATTENDANCE APP

Fig. 4.10 Snapshot of Admin's Section table

The Admin has the privilege to view, insert, delete and update the entries in the Period table. While viewing the entries, the First, Last, Previous and Next buttons can be used for effective navigation. All fields must be filled to insert a new entry or update an existing one. On update, SEM START DATE and SEM END DATE of the corresponding ACADYR and SEMESTER will be updated. The entry to be deleted is identified by the ACADYR and SEMESTER entered. For start and end date calendaring support is provided for entry and validation.



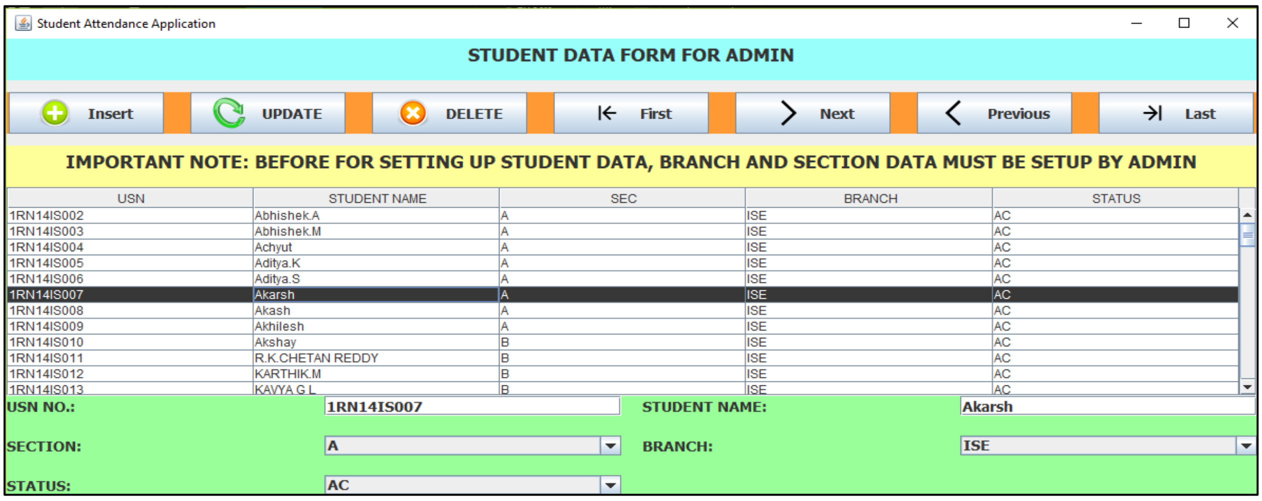
ACADYR:	2017	SEMESTER:	1	SEM START DATE:		SEM END DATE:	

ACADYR	SEM	START DATE	END DATE
2017	8	01-02-2017	30-05-2017
2017	7	07-08-2017	30-11-2017
2017	6	01-02-2017	30-05-2017
2017	5	07-08-2017	30-11-2017
2017	4	01-02-2017	30-05-2017
2017	3	07-08-2017	30-11-2017
2017	2	01-02-2017	30-05-2017
2017	1	07-08-2017	30-11-2017

IMPORTANT NOTE: ACADYR TABLE DATA MUST BE SETUP PRIOR TO USING THIS FORM, ONLY ODD OR EVEN SEM MUST BE SETUP

Fig. 4.11 Snapshot of Admin's Period table

The Admin has the privilege to view, insert, delete and update the entries in the Student table. While viewing the entries, the First, Last, Previous and Next buttons can be used for effective navigation. All fields must be filled to insert a new entry or update an existing one. On update, STUDENT NAME, BRANCH, SECTION and STATUS (as AC or NA) of the corresponding USN NO will be updated. The entry to be deleted is identified by the USN NO entered. The SECTION & BRANCH are to be selected from the dropdown list populated from the parent tables.



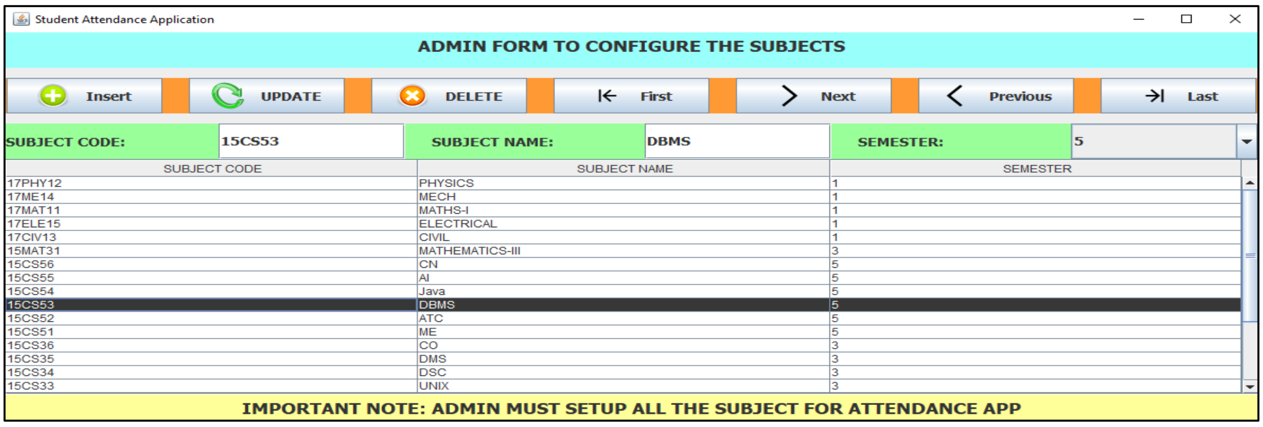
IMPORTANT NOTE: BEFORE FOR SETTING UP STUDENT DATA, BRANCH AND SECTION DATA MUST BE SETUP BY ADMIN

USN	STUDENT NAME	SEC	BRANCH	STATUS
1RN14IS002	Abhishek.A	A	ISE	AC
1RN14IS003	Abhishek.M	A	ISE	AC
1RN14IS004	Achyut	A	ISE	AC
1RN14IS005	Aditya.K	A	ISE	AC
1RN14IS006	Aditya.S	A	ISE	AC
1RN14IS007	Akarsh	A	ISE	AC
1RN14IS008	Akash	A	ISE	AC
1RN14IS009	Akhilesh	A	ISE	AC
1RN14IS010	Akshay	B	ISE	AC
1RN14IS011	R.K.CHETAN REDDY	B	ISE	AC
1RN14IS012	KARTHIK.M	B	ISE	AC
1RN14IS013	KAVYA G L	B	ISE	AC

USN NO.:	1RN14IS007	STUDENT NAME:	Akarsh
SECTION:	A	BRANCH:	ISE
STATUS:	AC		

Fig. 4.12 Snapshot of Admin's Student table

The Admin has the privilege to view, insert, delete and update the entries in the Subjects table. While viewing the entries, the First, Last, Previous and Next buttons can be used for effective navigation. This table is one of the Parent tables. All fields must be filled to insert a new entry or update an existing one. On update, the SUBJECT NAME and SEMESTER of the corresponding SUBJECT CODE will be updated. The entry to be deleted is identified by the SUBJECT CODE entered. The SEMESTER must be selected only from the dropdown list populated from PERIOD table.


Student Attendance Application
ADMIN FORM TO CONFIGURE THE SUBJECTS
Insert
UPDATE
DELETE
First
Next
Previous
Last

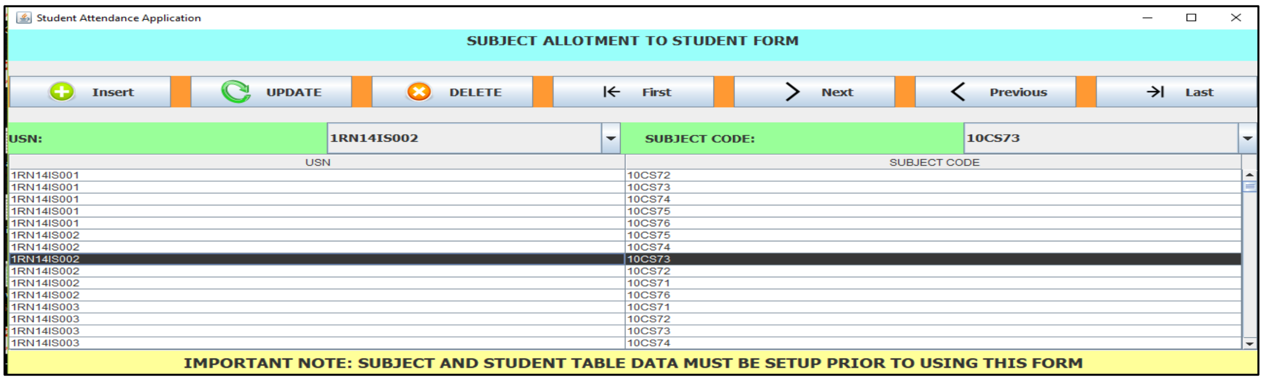
SUBJECT CODE: 15CS53 SUBJECT NAME: DBMS SEMESTER: 5

SUBJECT CODE	SUBJECT NAME	SEMESTER
17PHY12	PHYSICS	1
17ME14	MECH	1
17MAT11	MATHS-I	1
17ELE15	ELECTRICAL	1
17CIV13	CIVIL	1
15MAT31	MATHEMATICS-III	3
15CS56	CN	5
15CS55	AI	5
15CS54	Java	5
15CS53	DBMS	5
15CS52	ATC	5
15CS51	ME	5
15CS36	CO	3
15CS35	DMS	3
15CS34	DSC	3
15CS33	UNIX	3

IMPORTANT NOTE: ADMIN MUST SETUP ALL THE SUBJECT FOR ATTENDANCE APP

Fig. 4.13 Snapshot of Admin's Subjects table

The Admin has the privilege to view, insert, delete and update the entries in the Student-Course Allocation table. While viewing the entries, the First, Last, Previous and Next buttons can be used for effective navigation. Both fields must be filled to insert a new entry, or, to update or delete an existing one. Both USN and SUBJECT CODE have been populated from the respective parent tables.


Student Attendance Application
SUBJECT ALLOTMENT TO STUDENT FORM
Insert
UPDATE
DELETE
First
Next
Previous
Last

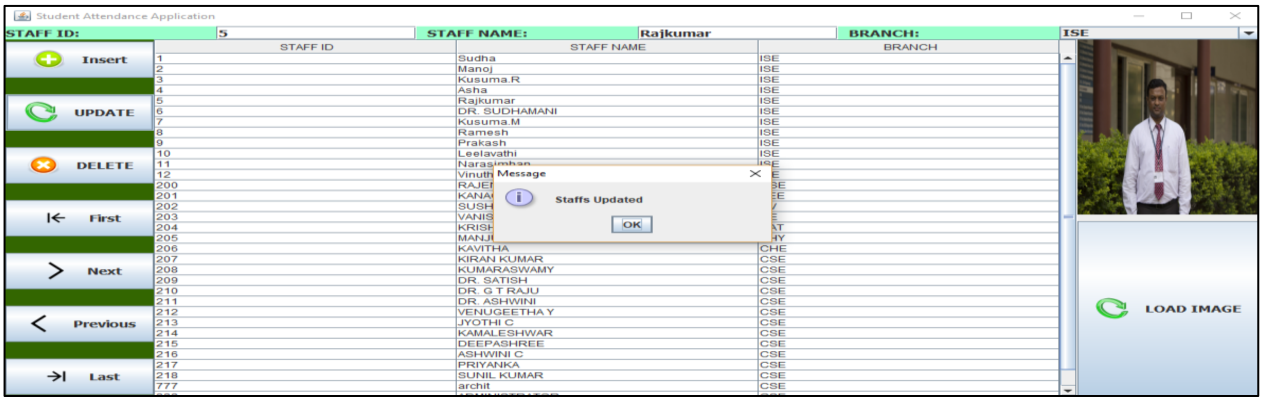
USN: 1RN14IS002 SUBJECT CODE: 10CS73

USN	SUBJECT CODE
1RN14IS001	10CS72
1RN14IS001	10CS73
1RN14IS001	10CS74
1RN14IS001	10CS75
1RN14IS001	10CS76
1RN14IS002	10CS75
1RN14IS002	10CS74
1RN14IS002	10CS73
1RN14IS002	10CS72
1RN14IS002	10CS71
1RN14IS002	10CS76
1RN14IS003	10CS71
1RN14IS003	10CS72
1RN14IS003	10CS73
1RN14IS003	10CS74

IMPORTANT NOTE: SUBJECT AND STUDENT TABLE DATA MUST BE SETUP PRIOR TO USING THIS FORM

Fig. 4.14 Snapshot of Admin's Student-Course Allocation table

The Admin has the privilege to view, insert, delete and update the entries in the Staff table. While viewing the entries, the First, Last, Previous and Next buttons can be used for effective navigation. This table is one of the Parent tables. All fields must be filled to insert a new entry or update an existing one. On update, the STAFF NAME, BRANCH and Staff Photo (IMAGE) of the corresponding STAFF ID will be updated. The entry to be deleted is identified by the STAFF ID entered.


Student Attendance Application

STAFF ID: 5 STAFF NAME: Rajkumar BRANCH: ISE

STAFF ID	STAFF NAME	BRANCH
1	Sudha	ISE
2	Manoj	ISE
3	Kusuma.R	ISE
4	Asha	ISE
5	Rajkumar	ISE
6	DR. SUDHAMANI	ISE
7	Kusuma.M	ISE
8	Ramesh	ISE
9	Prakash	ISE
10	Leelavathi	ISE
11	Narasimhan	ISE
12	Vinuth	E
200	RAJE	SE
201	KANA	EE
202	SUSH	/
203	VANIS	E
204	KRISH	AT
205	MANJ	HY
206	KAVITHA	CHE
207	KIRAN KUMAR	CSE
208	KUMARASWAMY	CSE
209	DR. SATISH	CSE
210	DR. G T RAJU	CSE
211	DR. ASHWINI	CSE
212	VENUGEETHA Y	CSE
213	JYOTHI C	CSE
214	KAMALESHWAR	CSE
215	DEEPASHREE	CSE
216	ASHWINI C	CSE
217	PRIYANKA	CSE
218	SUNIL KUMAR	CSE
777	archit	CSE

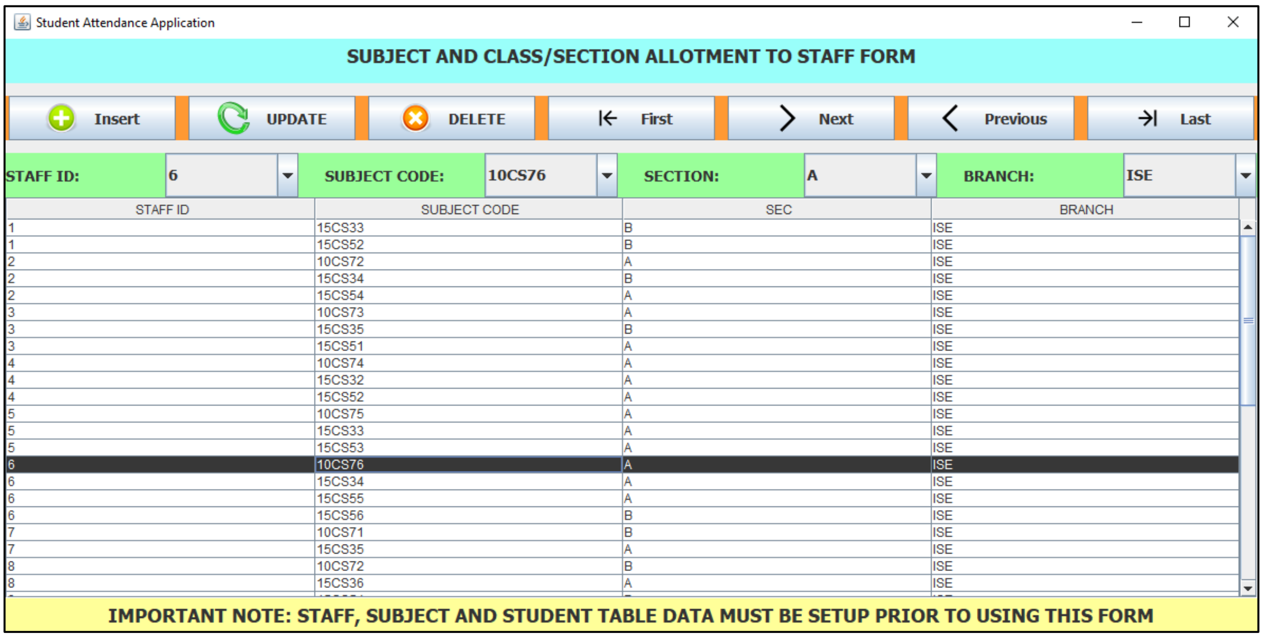
Message

Staffs Updated

OK

Fig. 4.15 Snapshot of Admin's Staff table

The Admin has the privilege to view, insert, delete and update the entries in the Staff Allocation table. While viewing the entries, the First, Last, Previous and Next buttons can be used for effective navigation. All fields must be filled to insert a new entry, or, to update or delete an existing one.


Student Attendance Application

SUBJECT AND CLASS/SECTION ALLOTMENT TO STAFF FORM

STAFF ID: 6 SUBJECT CODE: 10CS76 SECTION: A BRANCH: ISE

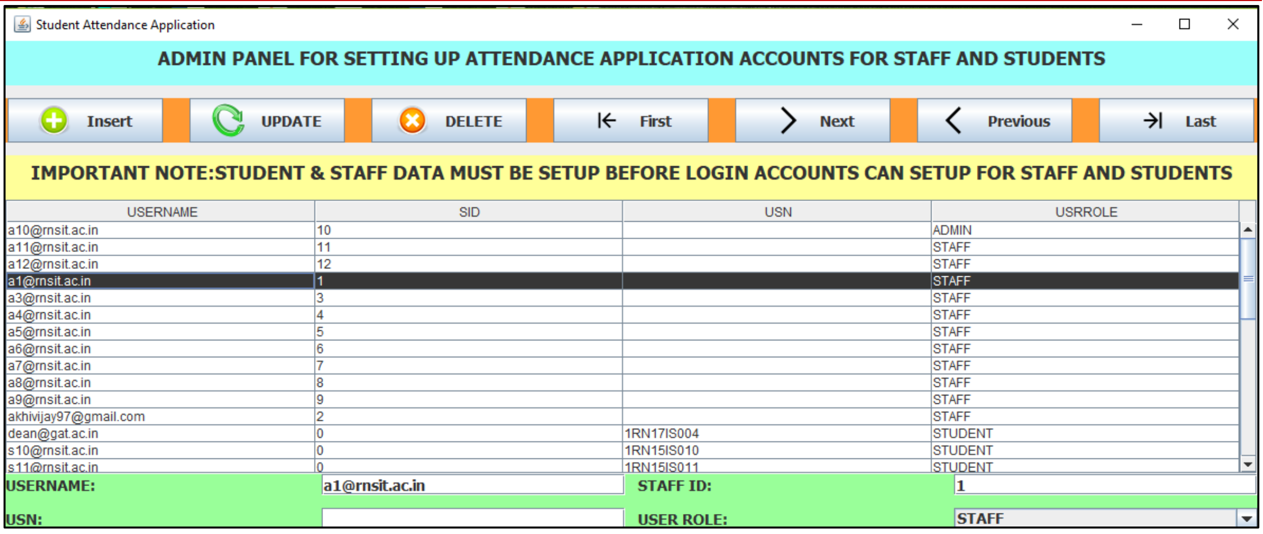
STAFF ID	SUBJECT CODE	SEC	BRANCH
1	15CS33	B	ISE
1	15CS52	B	ISE
2	10CS72	A	ISE
2	15CS34	B	ISE
2	15CS54	A	ISE
3	10CS73	A	ISE
3	15CS35	B	ISE
3	15CS51	A	ISE
4	10CS74	A	ISE
4	15CS32	A	ISE
4	15CS52	A	ISE
5	10CS75	A	ISE
5	15CS33	A	ISE
5	15CS53	A	ISE
6	10CS76	A	ISE
6	15CS34	A	ISE
6	15CS55	A	ISE
6	15CS56	B	ISE
7	10CS71	B	ISE
7	15CS35	A	ISE
8	10CS72	B	ISE
8	15CS36	A	ISE

IMPORTANT NOTE: STAFF, SUBJECT AND STUDENT TABLE DATA MUST BE SETUP PRIOR TO USING THIS FORM

Fig. 4.16 Snapshot of Admin's Staff Allocation table

The Admin has the privilege to view, insert, delete and update the entries in the Login table but has no access to the passwords and encrypted passwords stored for security reasons. While viewing the entries, the First, Last, Previous and Next buttons can be used for effective navigation. The USERNAME, USERROLE, and either the STAFF ID or USN, fields must be filled to insert a new entry or update an existing one. On update, the USERROLE, and either STAFF ID or USN, of the corresponding USERNAME will be updated. The entry to be deleted is identified by the USERNAME entered.

Note: Admin Account must be created first at the backend to use this application interface.


Student Attendance Application

ADMIN PANEL FOR SETTING UP ATTENDANCE APPLICATION ACCOUNTS FOR STAFF AND STUDENTS

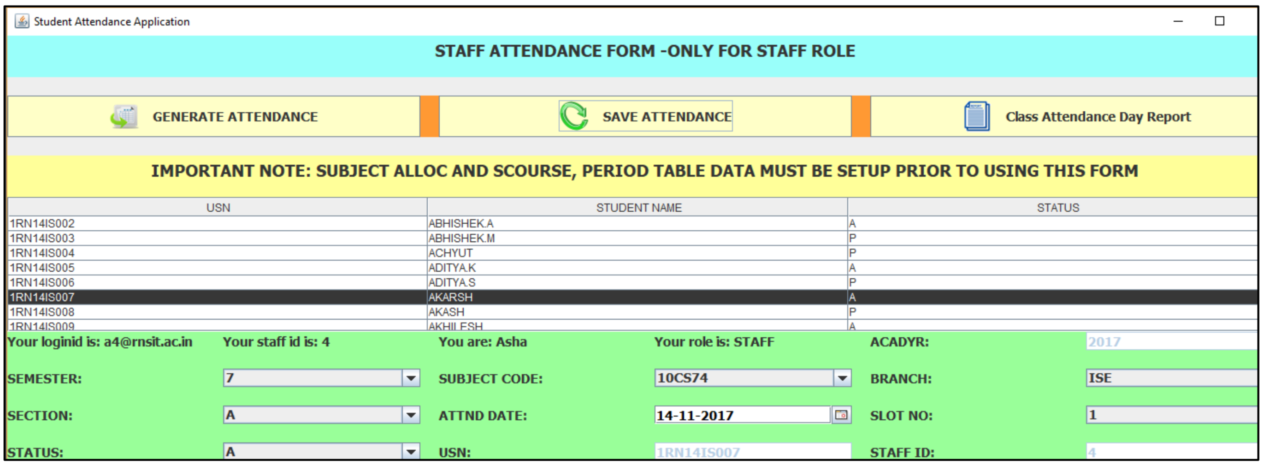
IMPORTANT NOTE:STUDENT & STAFF DATA MUST BE SETUP BEFORE LOGIN ACCOUNTS CAN SETUP FOR STAFF AND STUDENTS

USERNAME	SID	USN	USRROLE
a10@rnsit.ac.in	10		ADMIN
a11@rnsit.ac.in	11		STAFF
a12@rnsit.ac.in	12		STAFF
a1@rnsit.ac.in	1		STAFF
a3@rnsit.ac.in	3		STAFF
a4@rnsit.ac.in	4		STAFF
a5@rnsit.ac.in	5		STAFF
a6@rnsit.ac.in	6		STAFF
a7@rnsit.ac.in	7		STAFF
a8@rnsit.ac.in	8		STAFF
a9@rnsit.ac.in	9		STAFF
akhivijay97@gmail.com	2		STAFF
dean@gat.ac.in	0	1RN17IS004	STUDENT
s10@rnsit.ac.in	0	1RN15IS010	STUDENT
s11@rnsit.ac.in	0	1RN15IS011	STUDENT

USERNAME:	a1@rnsit.ac.in	STAFF ID:	1
USN:		USER ROLE:	STAFF

Fig. 4.17 Snapshot of Admin's Login table

The Staff has the privilege to generate a day's attendance for an assigned class and allocated subject, for an active academic year and semester. If the Staff Allocation details entered, match an entry in the Staff Allocation table, then a success message is displayed as a Dialog box.


Student Attendance Application

STAFF ATTENDANCE FORM -ONLY FOR STAFF ROLE

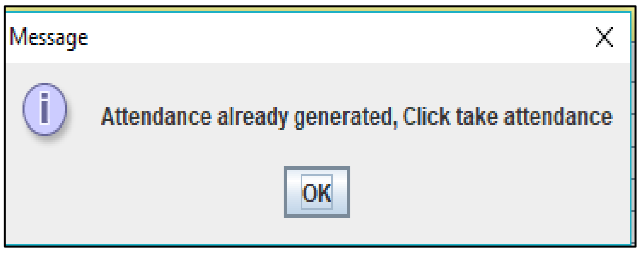
IMPORTANT NOTE: SUBJECT ALLOC AND SCOURSE, PERIOD TABLE DATA MUST BE SETUP PRIOR TO USING THIS FORM

USN	STUDENT NAME	STATUS
1RN14IS002	ABHISHEK.A	A
1RN14IS003	ABHISHEK.M	P
1RN14IS004	ACHYUT	P
1RN14IS005	ADITYA.K	A
1RN14IS006	ADITYA.S	P
1RN14IS007	AKARSH	A
1RN14IS008	AKASH	P
1RN14IS009	AKHILESH	A

Your loginid is: a4@rnsit.ac.in	Your staff id is: 4	You are: Asha	Your role is: STAFF	ACADYR:	2017
SEMESTER:	7	SUBJECT CODE:	10CS74	BRANCH:	ISE
SECTION:	A	ATTND DATE:	14-11-2017	SLOT NO:	1
STATUS:	A	USN:	1RN14IS007	STAFF ID:	4

Fig. 4.18 Snapshot of Staff Generating Daily Attendance

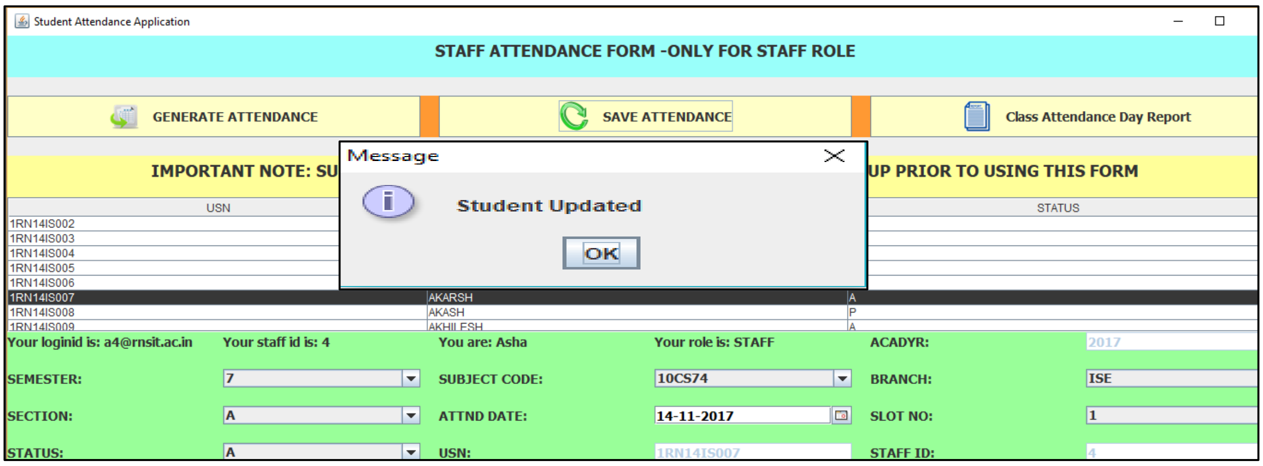
If the Staff Allocation details entered, match an entry in the Staff Allocation table, but the attendance entries already exist in the Attendance table, then an "Attendance already generated" message is displayed as a Dialog box.


Message

Attendance already generated, Click take attendance

Fig. 4.19 Snapshot If the attendance has already been generated

The Staff has the privilege to update a day's attendance for an assigned class and allocated subject, for an active academic year and semester. This involves changing the STATUS of attendance of students from P to A or vice-versa. Each change is saved on clicking the SAVE ATTENDANCE button.


Student Attendance Application
STAFF ATTENDANCE FORM -ONLY FOR STAFF ROLE
GENERATE ATTENDANCE
SAVE ATTENDANCE
Class Attendance Day Report
OK

IMPORTANT NOTE: SU … UP PRIOR TO USING THIS FORM

Message

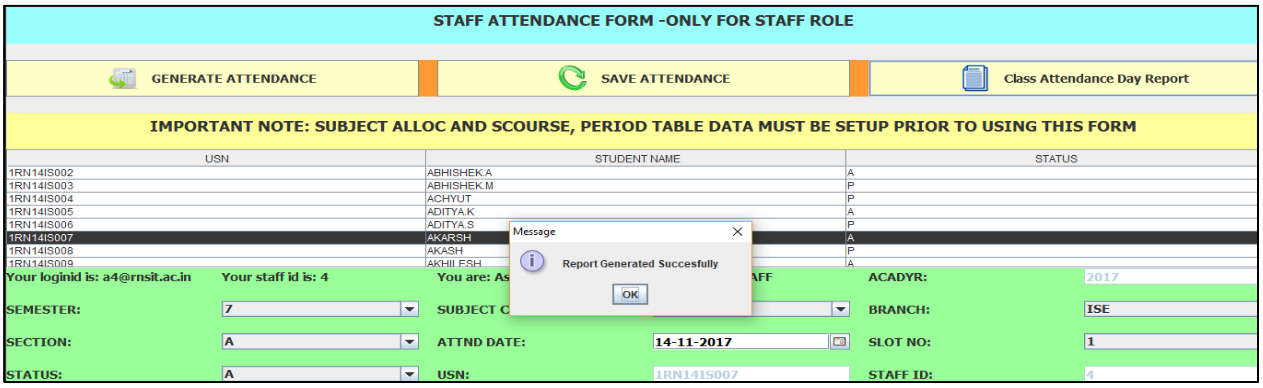
Student Updated

USN		STATUS
1RN14IS002		
1RN14IS003		
1RN14IS004		
1RN14IS005		
1RN14IS006		
1RN14IS007	AKARSH	A
1RN14IS008	AKASH	P
1RN14IS009	AKHILESH	A

Your loginid is: a4@rnsit.ac.in	Your staff id is: 4	You are: Asha	Your role is: STAFF	ACADYR:	2017
SEMESTER:	7	SUBJECT CODE:	10CS74	BRANCH:	ISE
SECTION:	A	ATTND DATE:	14-11-2017	SLOT NO:	1
STATUS:	A	USN:	1RN14IS007	STAFF ID:	4

Fig. 4.20 Snapshot of Staff Updating Daily Attendance

The Staff has the privilege to generate a Daily Class Attendance report for an assigned class and allocated subject, for an active academic year and semester.


STAFF ATTENDANCE FORM -ONLY FOR STAFF ROLE
GENERATE ATTENDANCE
SAVE ATTENDANCE
Class Attendance Day Report
OK

IMPORTANT NOTE: SUBJECT ALLOC AND SCOURSE, PERIOD TABLE DATA MUST BE SETUP PRIOR TO USING THIS FORM

USN	STUDENT NAME	STATUS
1RN14IS002	ABHISHEK.A	A
1RN14IS003	ABHISHEK.M	P
1RN14IS004	ACHYUT	P
1RN14IS005	ADITYA.K	A
1RN14IS006	ADITYA.S	P
1RN14IS007	AKARSH	A
1RN14IS008	AKASH	P
1RN14IS009	AKHILESH	A

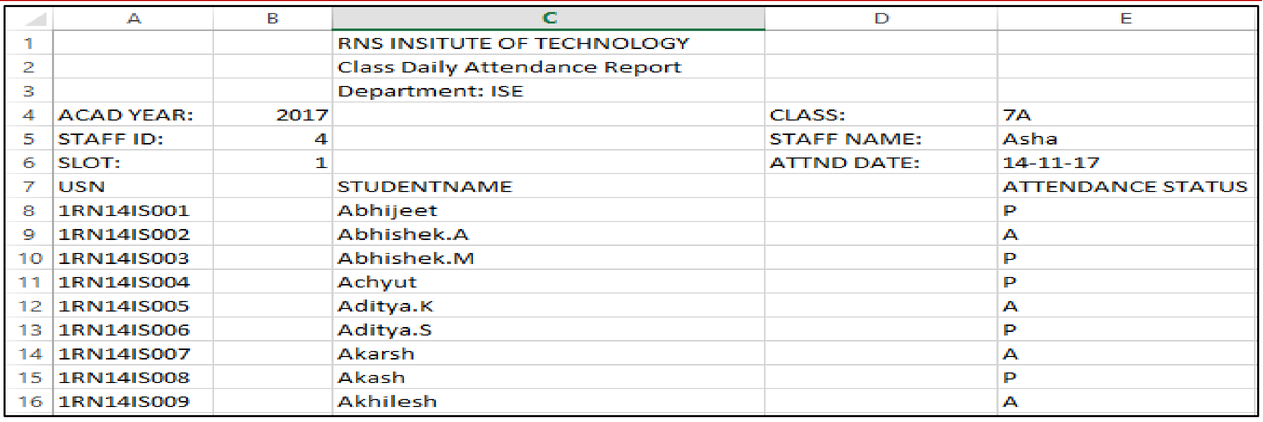
Message

Report Generated Succesfully

Your loginid is: a4@rnsit.ac.in	Your staff id is: 4	You are: As	AFF	ACADYR:	2017
SEMESTER:	7	SUBJECT C		BRANCH:	ISE
SECTION:	A	ATTND DATE:	14-11-2017	SLOT NO:	1
STATUS:	A	USN:	1RN14IS007	STAFF ID:	4

Fig. 4.21 Snapshot of Staff Generating Daily Attendance Report

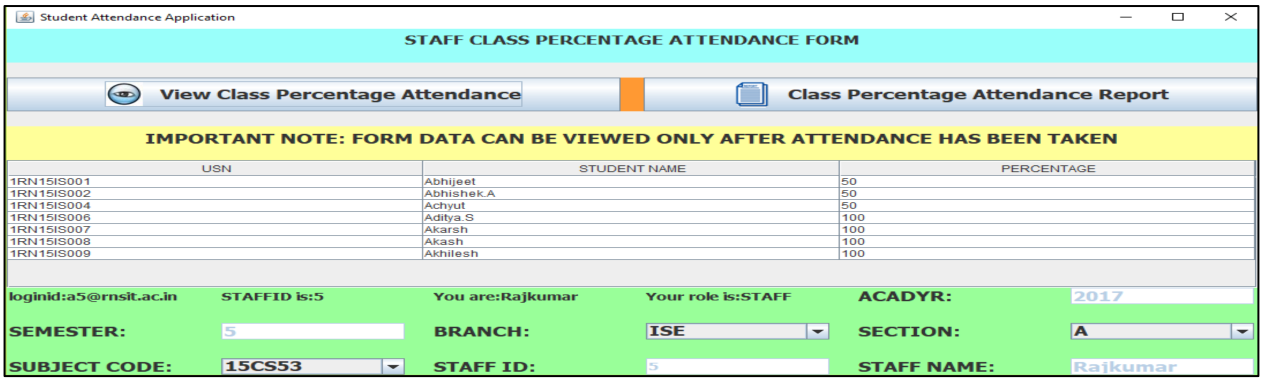
If the data entered through the Staff Attendance Form matches entries in the Attendance table, then the report is generated and presented in an Excel spreadsheet.



	A	B	C	D	E
1			RNS INSITUTE OF TECHNOLOGY		
2			Class Daily Attendance Report		
3			Department: ISE		
4	ACAD YEAR:	2017		CLASS:	7A
5	STAFF ID:	4		STAFF NAME:	Asha
6	SLOT:	1		ATTND DATE:	14-11-17
7	USN		STUDENTNAME		ATTENDANCE STATUS
8	1RN14IS001		Abhijeet		P
9	1RN14IS002		Abhishek.A		A
10	1RN14IS003		Abhishek.M		P
11	1RN14IS004		Achyut		P
12	1RN14IS005		Aditya.K		A
13	1RN14IS006		Aditya.S		P
14	1RN14IS007		Akarsh		A
15	1RN14IS008		Akash		P
16	1RN14IS009		Akhilesh		A

Fig. 4.22 Snapshot of Daily Attendance Report

The Staff has the privilege to view the percentage attendance of students in an assigned class and allocated subject, in an active academic year and semester. If the data entered through the Staff Class Percentage Attendance Form matches an entry in the Staff Allocation table for the logged in Staff, then the output is displayed as a table.



Student Attendance Application
STAFF CLASS PERCENTAGE ATTENDANCE FORM
View Class Percentage Attendance
Class Percentage Attendance Report

IMPORTANT NOTE: FORM DATA CAN BE VIEWED ONLY AFTER ATTENDANCE HAS BEEN TAKEN

USN	STUDENT NAME	PERCENTAGE
1RN15IS001	Abhijeet	50
1RN15IS002	Abhishek.A	50
1RN15IS004	Achyut	50
1RN15IS006	Aditya.S	100
1RN15IS007	Akarsh	100
1RN15IS008	Akash	100
1RN15IS009	Akhilesh	100

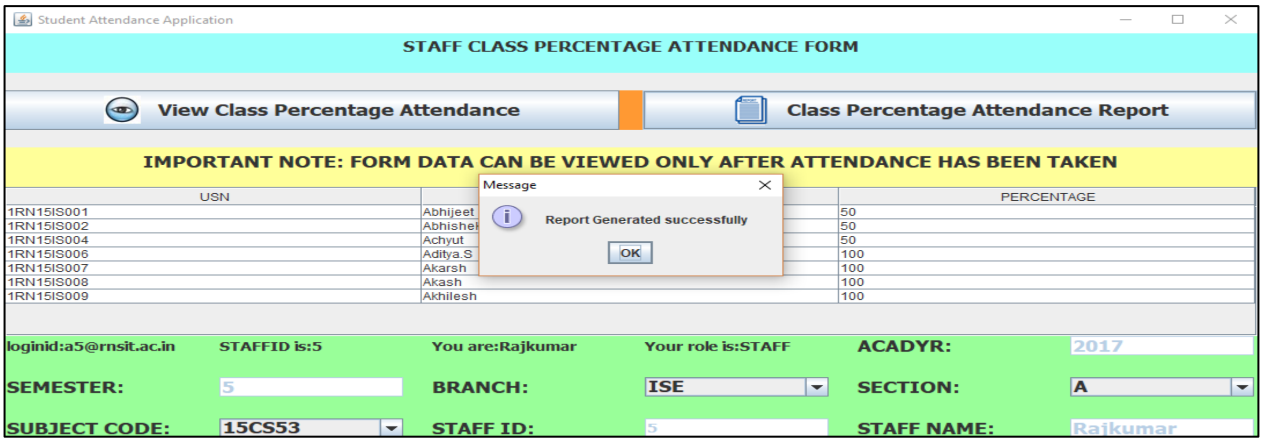
loginid:a5@rnsit.ac.in STAFFID is:5 You are:Rajkumar Your role is:STAFF ACADYR: 2017

SEMESTER: 5 BRANCH: ISE SECTION: A

SUBJECT CODE: 15CS53 STAFF ID: 5 STAFF NAME: Rajkumar

Fig. 4.23 Snapshot of Staff Viewing Class Percentage

The Staff has the privilege to generate a Class Percentage Attendance report for an assigned class and allocated subject, for an active academic year and semester.



Student Attendance Application
STAFF CLASS PERCENTAGE ATTENDANCE FORM
View Class Percentage Attendance
Class Percentage Attendance Report
Message
OK

IMPORTANT NOTE: FORM DATA CAN BE VIEWED ONLY AFTER ATTENDANCE HAS BEEN TAKEN

Report Generated successfully

USN		PERCENTAGE
1RN15IS001	Abhijeet	50
1RN15IS002	Abhishe	50
1RN15IS004	Achyut	50
1RN15IS006	Aditya.S	100
1RN15IS007	Akarsh	100
1RN15IS008	Akash	100
1RN15IS009	Akhilesh	100

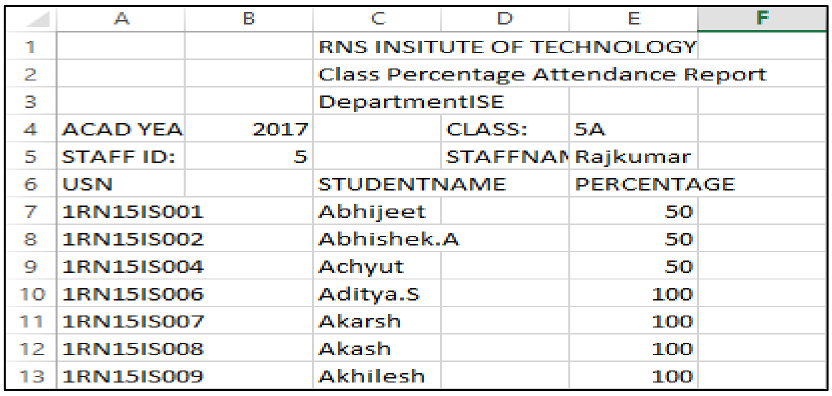
loginid:a5@rnsit.ac.in STAFFID is:5 You are:Rajkumar Your role is:STAFF ACADYR: 2017

SEMESTER: 5 BRANCH: ISE SECTION: A

SUBJECT CODE: 15CS53 STAFF ID: 5 STAFF NAME: Rajkumar

Fig. 4.24 Snapshot of Staff Generating Class Percentage Attendance Report

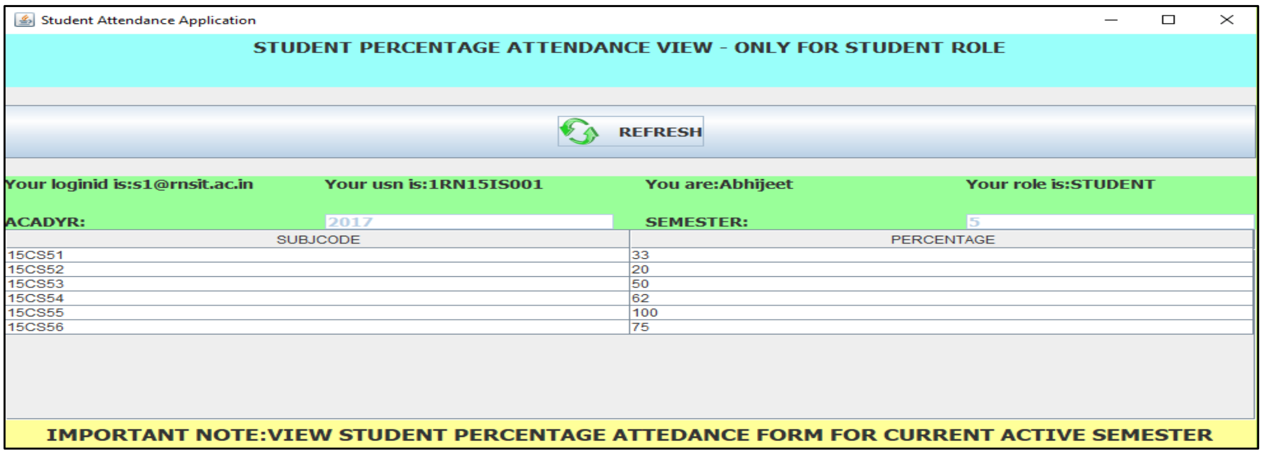
If the data entered through the Staff Attendance Form matches an entry in the Staff Allocation table for the logged in Staff, then the report is generated and presented in an Excel spreadsheet.



	A	B	C	D	E	F
1			RNS INSITUTE OF TECHNOLOGY			
2			Class Percentage Attendance Report			
3			DepartmentISE			
4	ACAD YEA	2017		CLASS:	5A	
5	STAFF ID:	5		STAFFNAM	Rajkumar	
6	USN		STUDENTNAME		PERCENTAGE	
7	1RN15IS001		Abhijeet		50	
8	1RN15IS002		Abhishek.A		50	
9	1RN15IS004		Achyut		50	
10	1RN15IS006		Aditya.S		100	
11	1RN15IS007		Akarsh		100	
12	1RN15IS008		Akash		100	
13	1RN15IS009		Akhilesh		100	

Fig. 4.25 Snapshot of Class Percentage Attendance Report

The Student has the privilege to view his/her attendance for all registered subjects, in the active academic year and semester.



Student Attendance Application

STUDENT PERCENTAGE ATTENDANCE VIEW - ONLY FOR STUDENT ROLE

REFRESH

Your loginid is:s1@rnsit.ac.in Your usn is:1RN15IS001 You are:Abhijeet Your role is:STUDENT

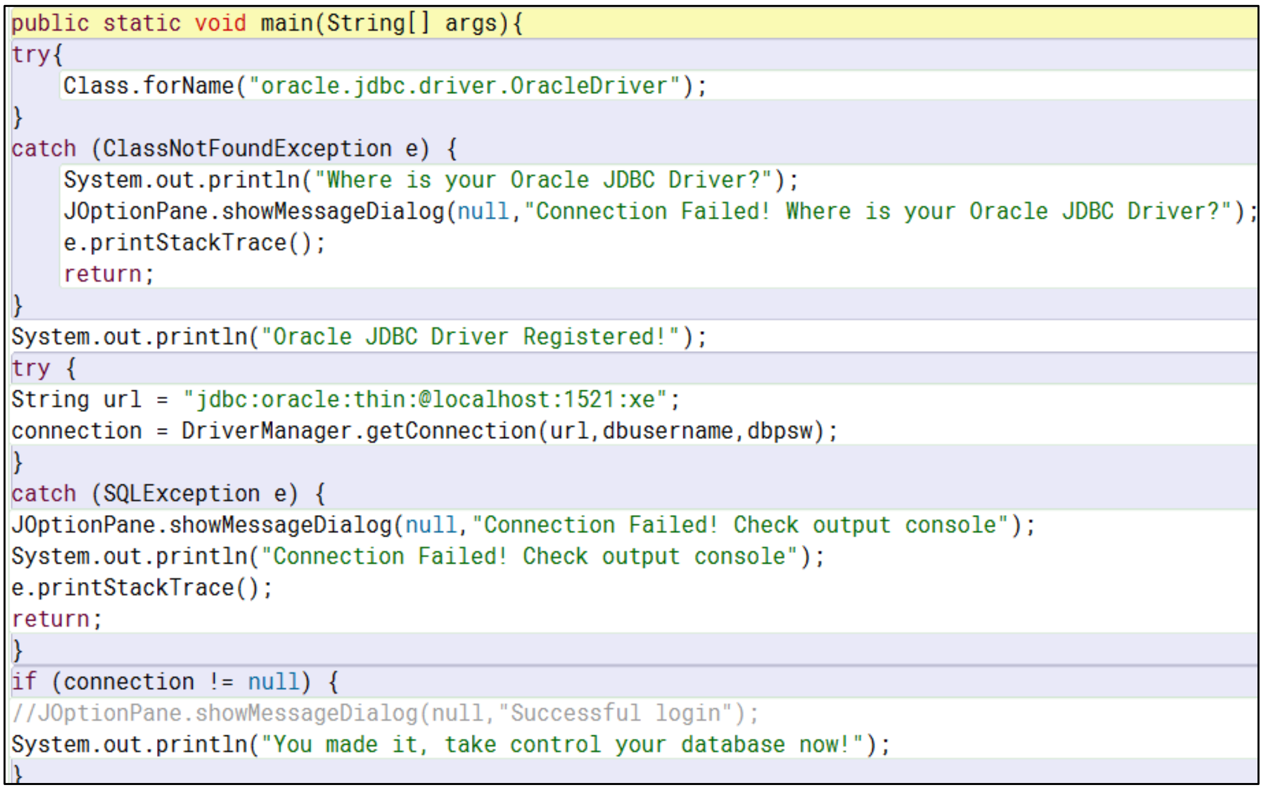
ACADYR: 2017 SEMESTER: 5

SUBJCODE	PERCENTAGE
15CS51	33
15CS52	20
15CS53	50
15CS54	62
15CS55	100
15CS56	75

IMPORTANT NOTE:VIEW STUDENT PERCENTAGE ATTEDANCE FORM FOR CURRENT ACTIVE SEMESTER

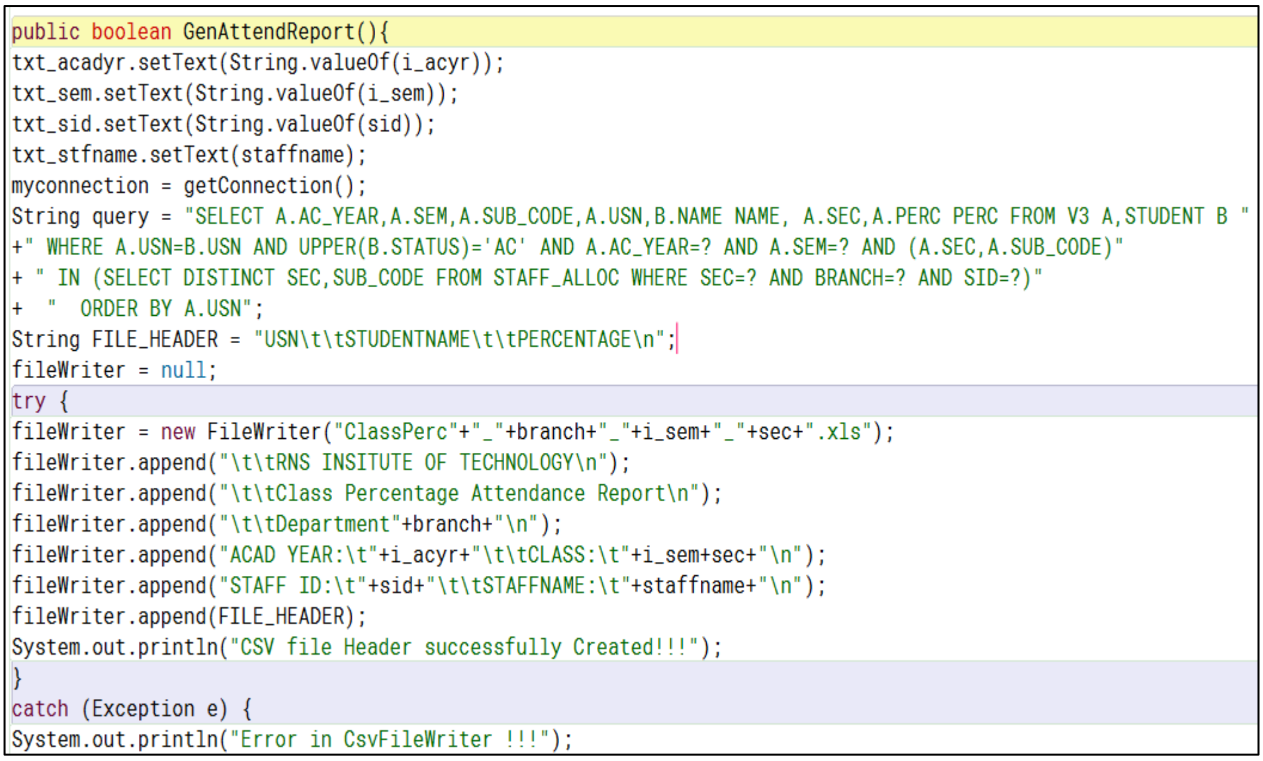
Fig. 4.26 Snapshot of Student Viewing Percentage Attendance

4.5.2 Java and PL/SQL Code Snippets



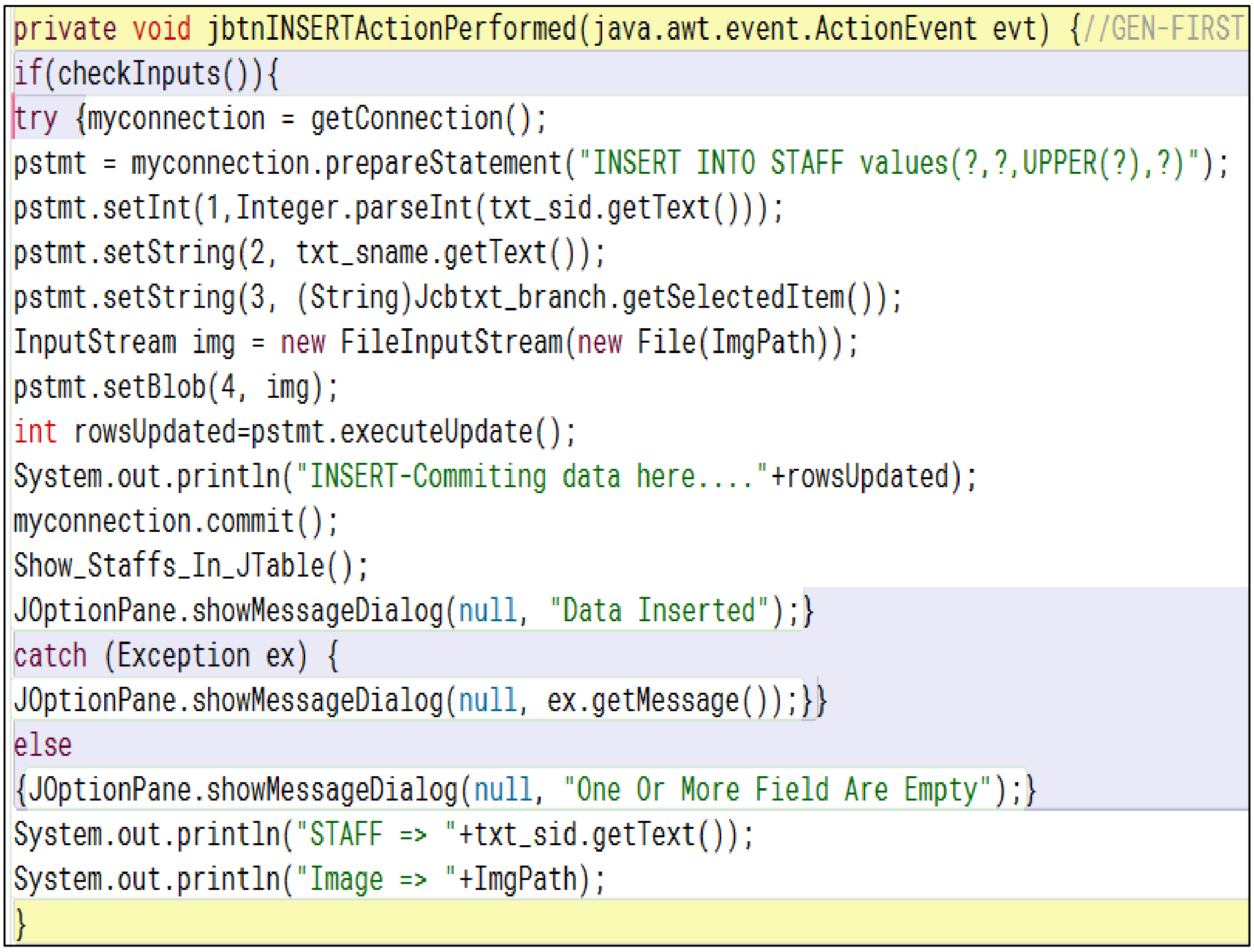
```
public static void main(String[] args){
try{
    Class.forName("oracle.jdbc.driver.OracleDriver");
}
catch (ClassNotFoundException e) {
    System.out.println("Where is your Oracle JDBC Driver?");
    JOptionPane.showMessageDialog(null,"Connection Failed! Where is your Oracle JDBC Driver?");
    e.printStackTrace();
    return;
}
System.out.println("Oracle JDBC Driver Registered!");
try {
String url = "jdbc:oracle:thin:@localhost:1521:xe";
connection = DriverManager.getConnection(url,dbusername,dbpsw);
}
catch (SQLException e) {
JOptionPane.showMessageDialog(null,"Connection Failed! Check output console");
System.out.println("Connection Failed! Check output console");
e.printStackTrace();
return;
}
if (connection != null) {
//JOptionPane.showMessageDialog(null,"Successful login");
System.out.println("You made it, take control your database now!");
}
```

Fig. 4.27 Java Code for Loading Oracle JDBC Driver and Creating DB Connection



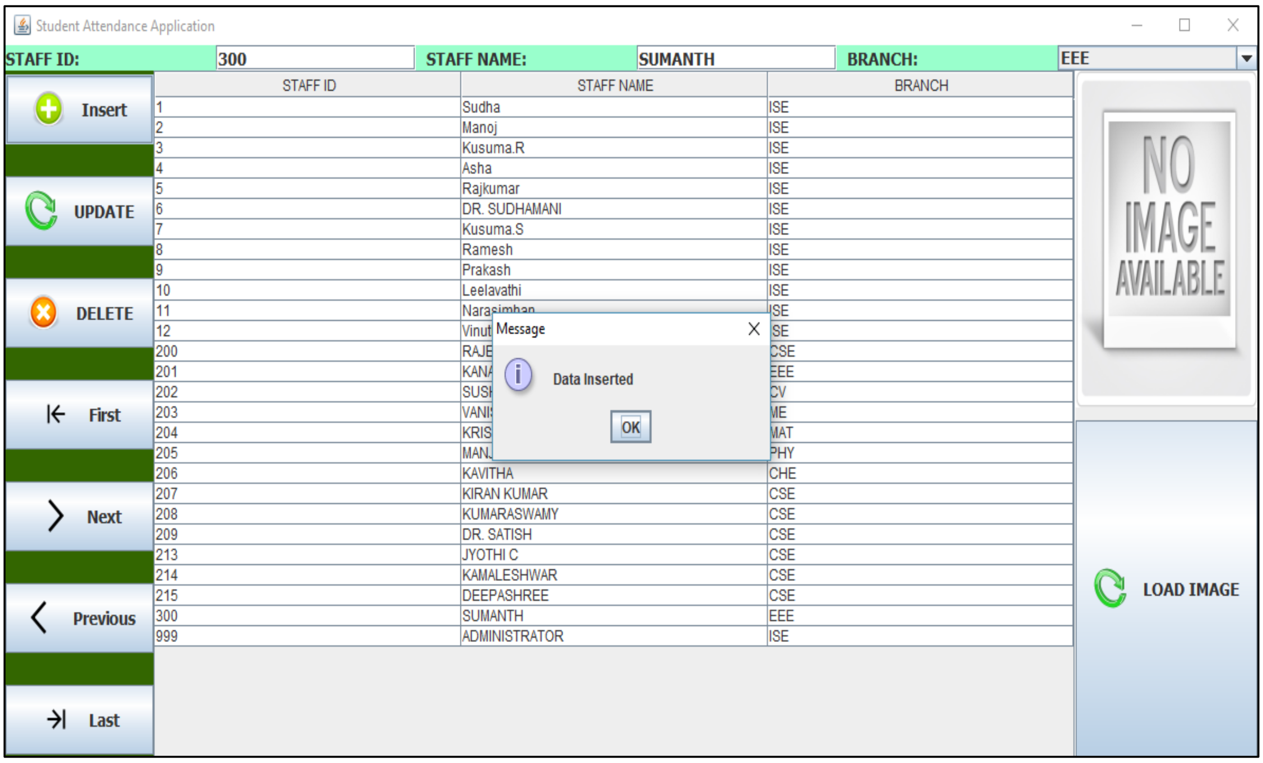
```
public boolean GenAttendReport(){
txt_acadyr.setText(String.valueOf(i_acyr));
txt_sem.setText(String.valueOf(i_sem));
txt_sid.setText(String.valueOf(sid));
txt_stfname.setText(staffname);
myconnection = getConnection();
String query = "SELECT A.AC_YEAR,A.SEM,A.SUB_CODE,A.USN,B.NAME NAME, A.SEC,A.PERC PERC FROM V3 A,STUDENT B "
+" WHERE A.USN=B.USN AND UPPER(B.STATUS)='AC' AND A.AC_YEAR=? AND A.SEM=? AND (A.SEC,A.SUB_CODE)"
+ " IN (SELECT DISTINCT SEC,SUB_CODE FROM STAFF_ALLOC WHERE SEC=? AND BRANCH=? AND SID=?)"
+  "  ORDER BY A.USN";
String FILE_HEADER = "USN\t\tSTUDENTNAME\t\tPERCENTAGE\n";
fileWriter = null;
try {
fileWriter = new FileWriter("ClassPerc"+"_"+branch+"_"+i_sem+"_"+sec+".xls");
fileWriter.append("\t\tRNS INSITUTE OF TECHNOLOGY\n");
fileWriter.append("\t\tClass Percentage Attendance Report\n");
fileWriter.append("\t\tDepartment"+branch+"\n");
fileWriter.append("ACAD YEAR:\t"+i_acyr+"\t\tCLASS:\t"+i_sem+sec+"\n");
fileWriter.append("STAFF ID:\t"+sid+"\t\tSTAFFNAME:\t"+staffname+"\n");
fileWriter.append(FILE_HEADER);
System.out.println("CSV file Header successfully Created!!!");
}
catch (Exception e) {
System.out.println("Error in CsvFileWriter !!!");
```

Fig. 4.28 Java Code to Display(Select) Generated Attendance



```
private void jbtnINSERTActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST
if(checkInputs()){
try {myconnection = getConnection();
pstmt = myconnection.prepareStatement("INSERT INTO STAFF values(?,?,UPPER(?),?)");
pstmt.setInt(1,Integer.parseInt(txt_sid.getText()));
pstmt.setString(2, txt_sname.getText());
pstmt.setString(3, (String)Jcbtxt_branch.getSelectedItem());
InputStream img = new FileInputStream(new File(ImgPath));
pstmt.setBlob(4, img);
int rowsUpdated=pstmt.executeUpdate();
System.out.println("INSERT-Commiting data here...."+rowsUpdated);
myconnection.commit();
Show_Staffs_In_JTable();
JOptionPane.showMessageDialog(null, "Data Inserted");}
catch (Exception ex) {
JOptionPane.showMessageDialog(null, ex.getMessage());}}
else
{JOptionPane.showMessageDialog(null, "One Or More Field Are Empty");}
System.out.println("STAFF => "+txt_sid.getText());
System.out.println("Image => "+ImgPath);
}
```

Fig. 4.29 Java Code to Insert a Staff



Student Attendance Application

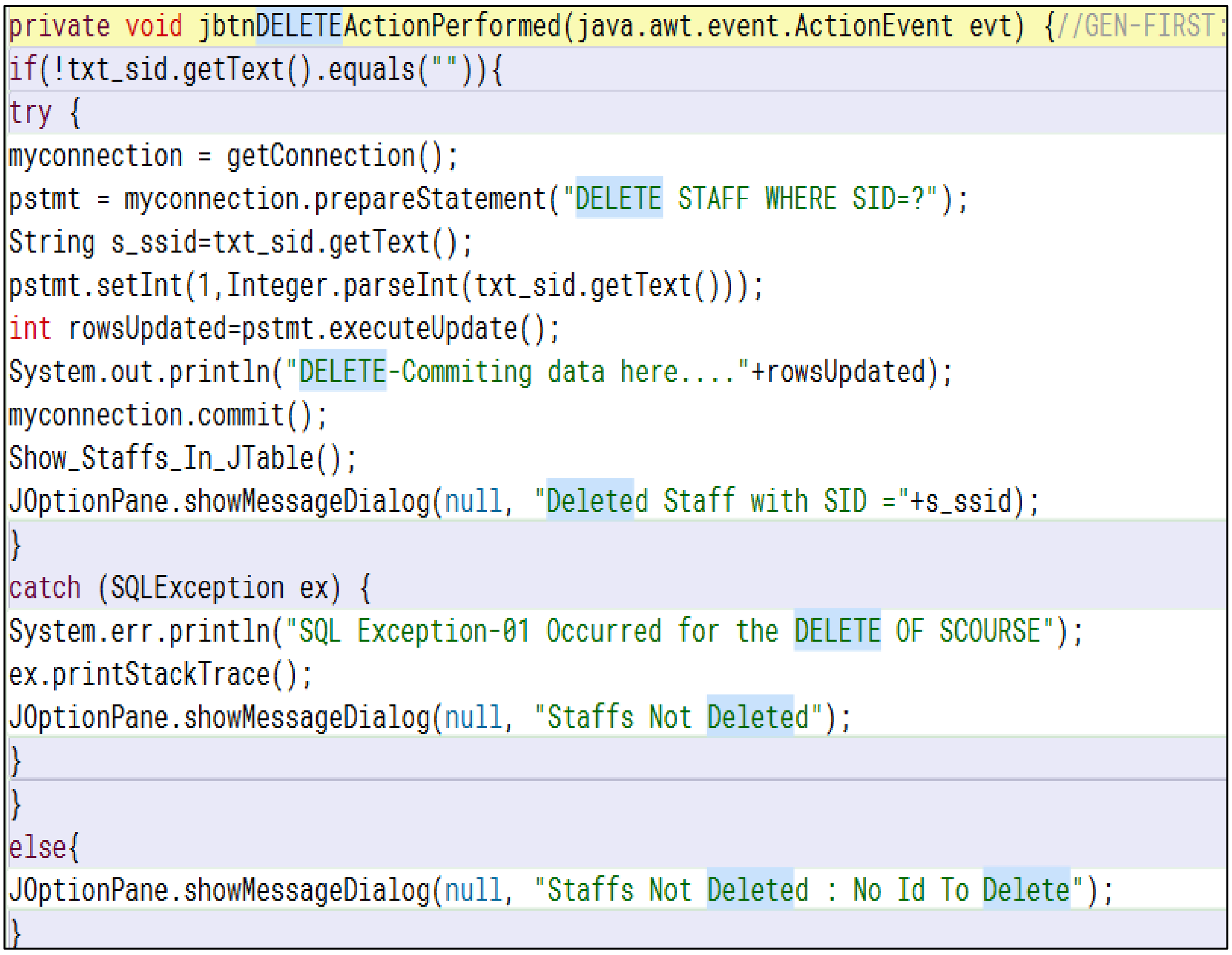
STAFF ID: 300 STAFF NAME: SUMANTH BRANCH: EEE

STAFF ID	STAFF NAME	BRANCH
1	Sudha	ISE
2	Manoj	ISE
3	Kusuma.R	ISE
4	Asha	ISE
5	Rajkumar	ISE
6	DR. SUDHAMANI	ISE
7	Kusuma.S	ISE
8	Ramesh	ISE
9	Prakash	ISE
10	Leelavathi	ISE
11	Narasimhan	ISE
12	Vinut	SE
200	RAJE	CSE
201	KANA	EEE
202	SUSH	CV
203	VANIS	ME
204	KRIS	MAT
205	MANJ	PHY
206	KAVITHA	CHE
207	KIRAN KUMAR	CSE
208	KUMARASWAMY	CSE
209	DR. SATISH	CSE
213	JYOTHI C	CSE
214	KAMALESHWAR	CSE
215	DEEPASHREE	CSE
300	SUMANTH	EEE
999	ADMINISTRATOR	ISE

Message

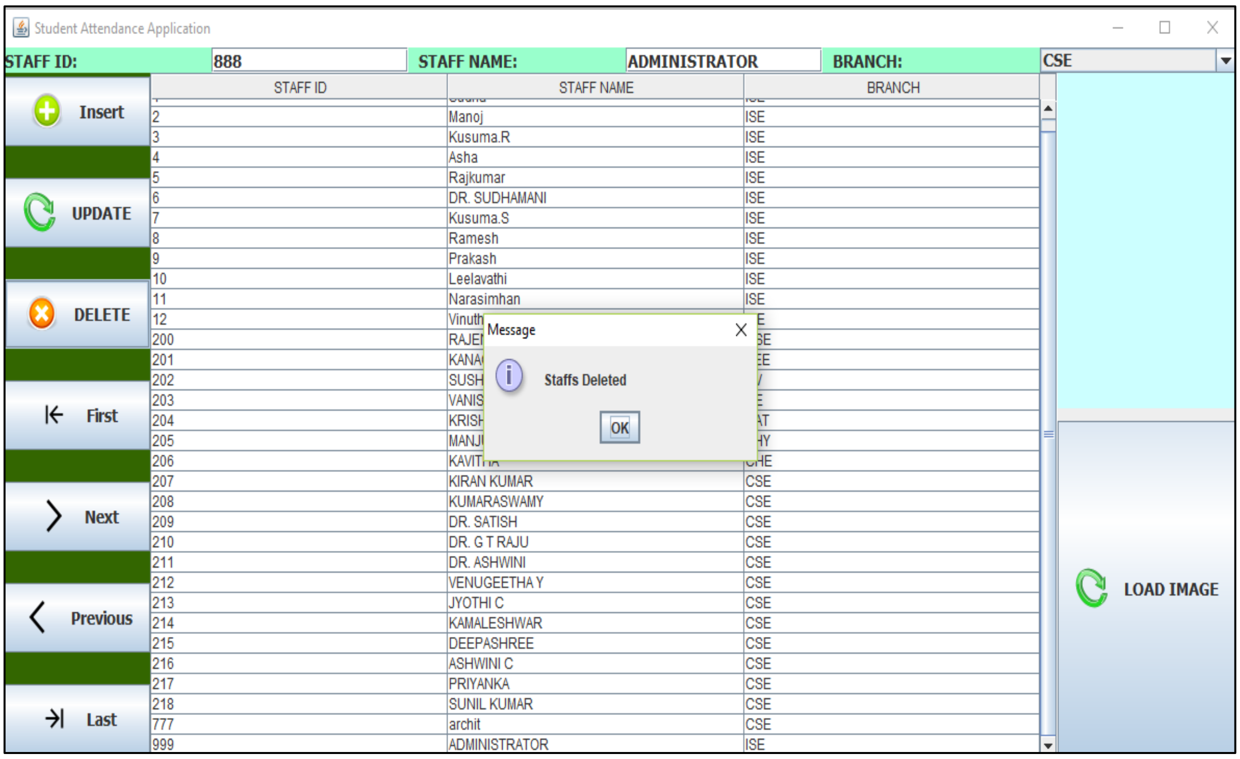
Data Inserted

Fig. 4.30 GUI and Popup message on insertion of staff



```
private void jbtnDELETEActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
if(!txt_sid.getText().equals("")){
try {
myconnection = getConnection();
pstmt = myconnection.prepareStatement("DELETE STAFF WHERE SID=?");
String s_ssid=txt_sid.getText();
pstmt.setInt(1,Integer.parseInt(txt_sid.getText()));
int rowsUpdated=pstmt.executeUpdate();
System.out.println("DELETE-Commiting data here...."+rowsUpdated);
myconnection.commit();
Show_Staffs_In_JTable();
JOptionPane.showMessageDialog(null, "Deleted Staff with SID ="+s_ssid);
}
catch (SQLException ex) {
System.err.println("SQL Exception-01 Occurred for the DELETE OF SCOURSE");
ex.printStackTrace();
JOptionPane.showMessageDialog(null, "Staffs Not Deleted");
}
}
else{
JOptionPane.showMessageDialog(null, "Staffs Not Deleted : No Id To Delete");
}
```

Fig. 4.31 Java Code to Delete a Staff



STAFF ID: 888 STAFF NAME: ADMINISTRATOR BRANCH: CSE

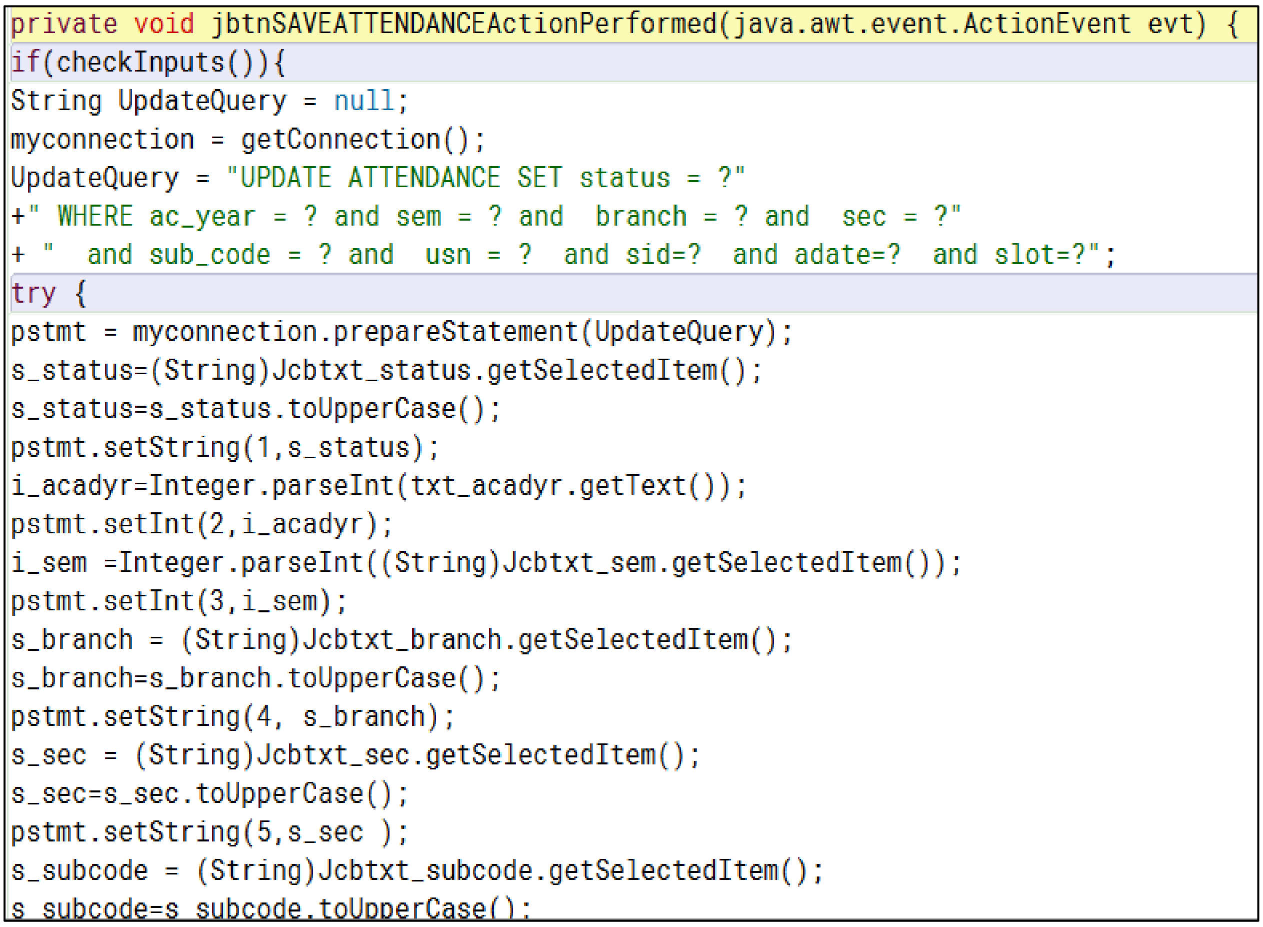
STAFF ID	STAFF NAME	BRANCH
2	Manoj	ISE
3	Kusuma.R	ISE
4	Asha	ISE
5	Rajkumar	ISE
6	DR. SUDHAMANI	ISE
7	Kusuma.S	ISE
8	Ramesh	ISE
9	Prakash	ISE
10	Leelavathi	ISE
11	Narasimhan	ISE
12	Vinuth	E
200	RAJE	SE
201	KANA	EE
202	SUSH	/
203	VANIS	E
204	KRISH	AT
205	MANJ	HY
206	KAVITHA	CHE
207	KIRAN KUMAR	CSE
208	KUMARASWAMY	CSE
209	DR. SATISH	CSE
210	DR. G T RAJU	CSE
211	DR. ASHWINI	CSE
212	VENUGEETHA Y	CSE
213	JYOTHI C	CSE
214	KAMALESHWAR	CSE
215	DEEPASHREE	CSE
216	ASHWINI C	CSE
217	PRIYANKA	CSE
218	SUNIL KUMAR	CSE
777	archit	CSE
999	ADMINISTRATOR	ISE

Message

Staffs Deleted

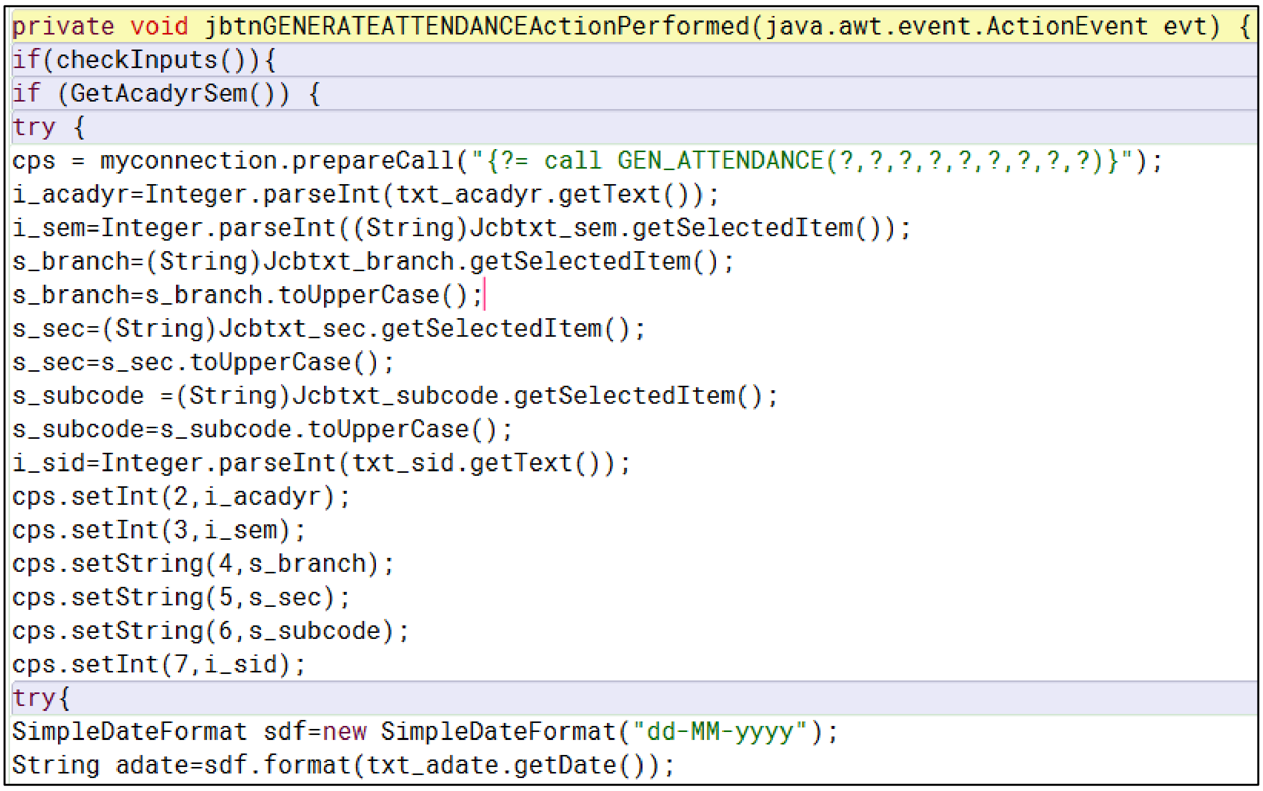
OK

Fig. 4.32 GUI and Popup message on deletion of staff



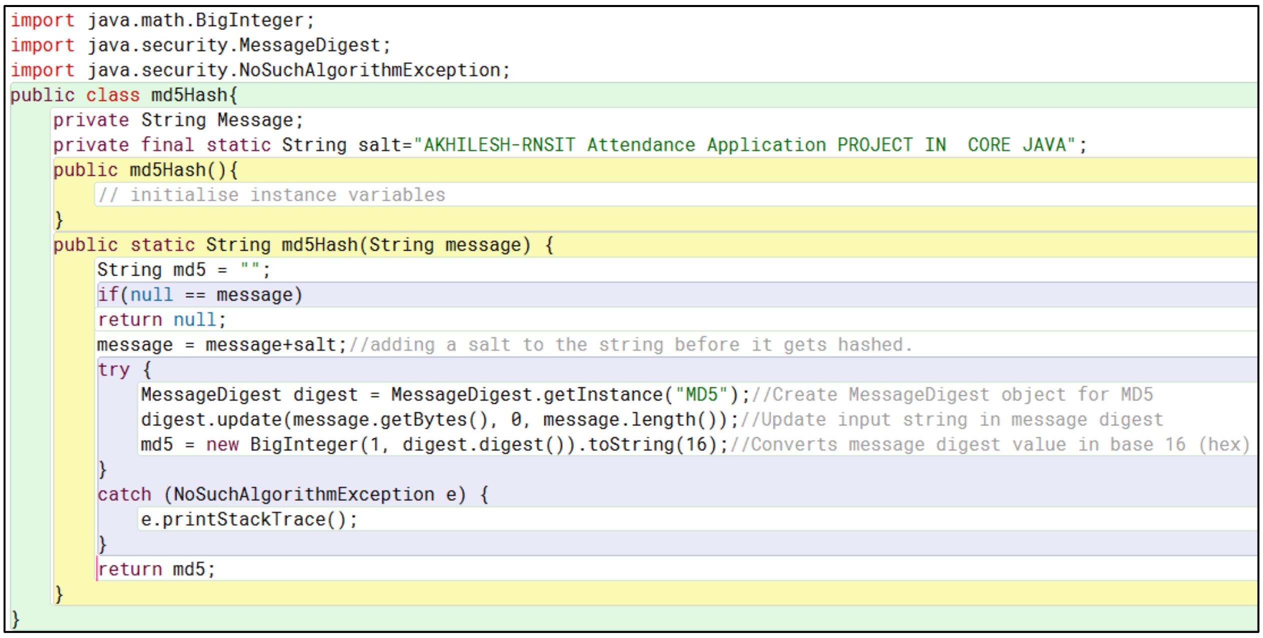
```
private void jbtnSAVEATTENDANCEActionPerformed(java.awt.event.ActionEvent evt) {
if(checkInputs()){
String UpdateQuery = null;
myconnection = getConnection();
UpdateQuery = "UPDATE ATTENDANCE SET status = ?"
+" WHERE ac_year = ? and sem = ? and  branch = ? and  sec = ?"
+ "  and sub_code = ? and  usn = ?  and sid=?  and adate=?  and slot=?";
try {
pstmt = myconnection.prepareStatement(UpdateQuery);
s_status=(String)Jcbtxt_status.getSelectedItem();
s_status=s_status.toUpperCase();
pstmt.setString(1,s_status);
i_acadyr=Integer.parseInt(txt_acadyr.getText());
pstmt.setInt(2,i_acadyr);
i_sem =Integer.parseInt((String)Jcbtxt_sem.getSelectedItem());
pstmt.setInt(3,i_sem);
s_branch = (String)Jcbtxt_branch.getSelectedItem();
s_branch=s_branch.toUpperCase();
pstmt.setString(4, s_branch);
s_sec = (String)Jcbtxt_sec.getSelectedItem();
s_sec=s_sec.toUpperCase();
pstmt.setString(5,s_sec );
s_subcode = (String)Jcbtxt_subcode.getSelectedItem();
s_subcode=s_subcode.toUpperCase();
```

Fig. 4.33 Java Code to Update Generated Attendance



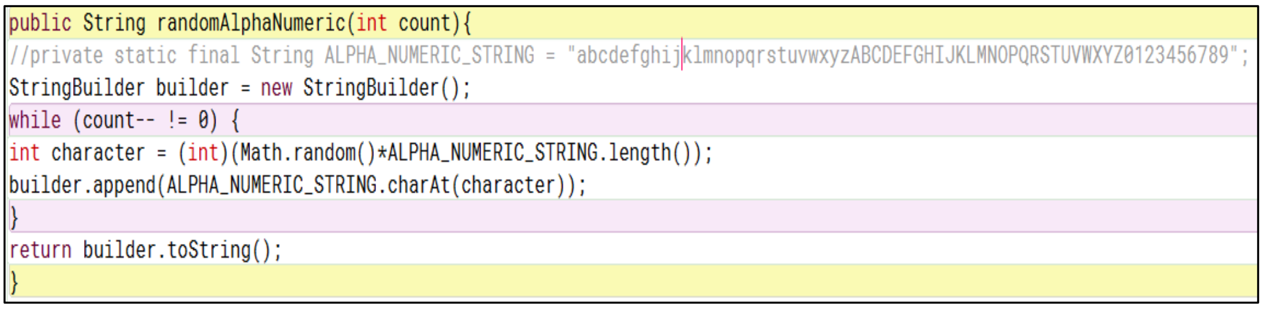
```
private void jbtnGENERATEATTENDANCEActionPerformed(java.awt.event.ActionEvent evt) {
if(checkInputs()){
if (GetAcadyrSem()) {
try {
cps = myconnection.prepareCall("{?= call GEN_ATTENDANCE(?,?,?,?,?,?,?,?,?)}");
i_acadyr=Integer.parseInt(txt_acadyr.getText());
i_sem=Integer.parseInt((String)Jcbtxt_sem.getSelectedItem());
s_branch=(String)Jcbtxt_branch.getSelectedItem();
s_branch=s_branch.toUpperCase();
s_sec=(String)Jcbtxt_sec.getSelectedItem();
s_sec=s_sec.toUpperCase();
s_subcode =(String)Jcbtxt_subcode.getSelectedItem();
s_subcode=s_subcode.toUpperCase();
i_sid=Integer.parseInt(txt_sid.getText());
cps.setInt(2,i_acadyr);
cps.setInt(3,i_sem);
cps.setString(4,s_branch);
cps.setString(5,s_sec);
cps.setString(6,s_subcode);
cps.setInt(7,i_sid);
try{
SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
String adate=sdf.format(txt_adate.getDate());
```

Fig. 4.34 Java Code to Call Stored Procedure



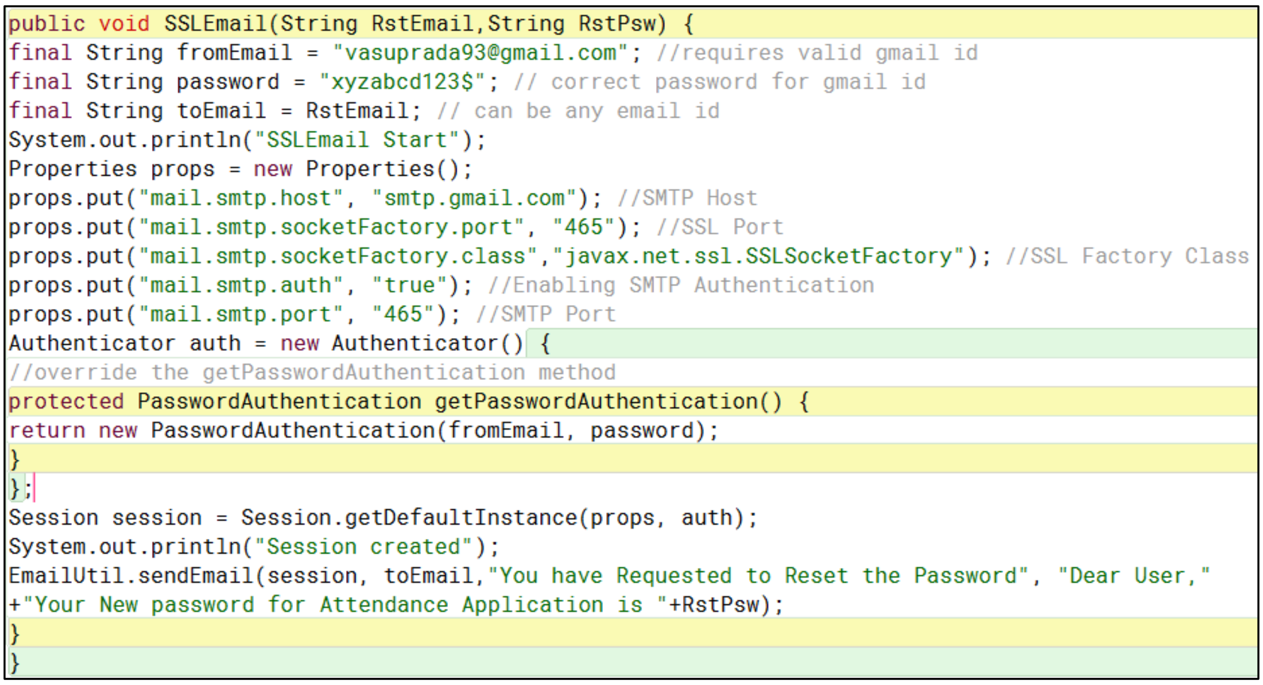
```
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
public class md5Hash{
    private String Message;
    private final static String salt="AKHILESH-RNSIT Attendance Application PROJECT IN  CORE JAVA";
    public md5Hash(){
        // initialise instance variables
    }
    public static String md5Hash(String message) {
        String md5 = "";
        if(null == message)
        return null;
        message = message+salt;//adding a salt to the string before it gets hashed.
        try {
            MessageDigest digest = MessageDigest.getInstance("MD5");//Create MessageDigest object for MD5
            digest.update(message.getBytes(), 0, message.length());//Update input string in message digest
            md5 = new BigInteger(1, digest.digest()).toString(16);//Converts message digest value in base 16 (hex)
        }
        catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        return md5;
    }
}
```

Fig. 4.35 Java Code for One Way Password Encryption-(MD5+SALT string)



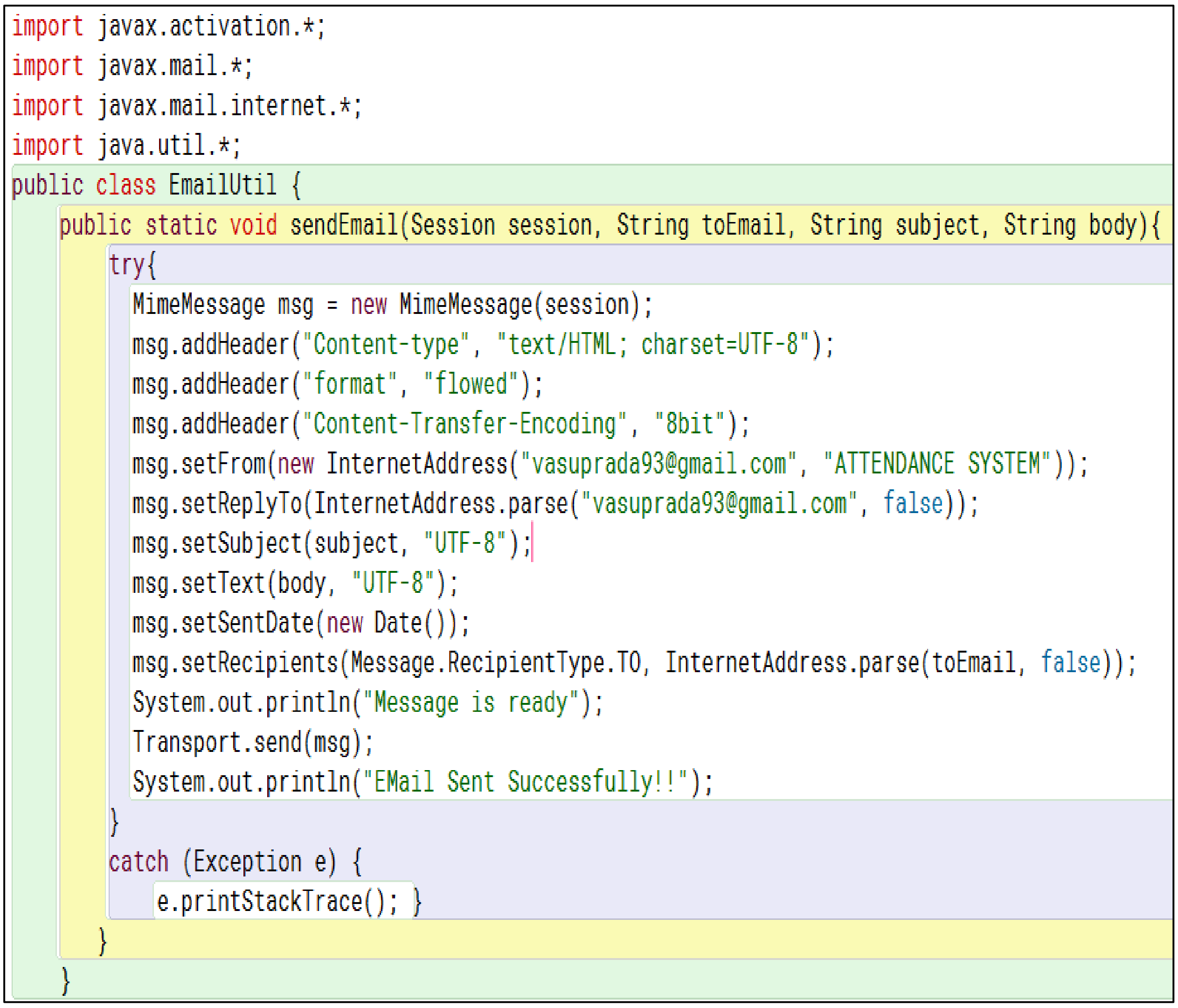
```
public String randomAlphaNumeric(int count){
//private static final String ALPHA_NUMERIC_STRING = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
StringBuilder builder = new StringBuilder();
while (count-- != 0) {
int character = (int)(Math.random()*ALPHA_NUMERIC_STRING.length());
builder.append(ALPHA_NUMERIC_STRING.charAt(character));
}
return builder.toString();
}
```

Fig. 4.36 Java Code for Random Password Generation



```
public void SSLEmail(String RstEmail,String RstPsw) {
final String fromEmail = "vasuprada93@gmail.com"; //requires valid gmail id
final String password = "xyzabcd123$"; // correct password for gmail id
final String toEmail = RstEmail; // can be any email id
System.out.println("SSLEmail Start");
Properties props = new Properties();
props.put("mail.smtp.host", "smtp.gmail.com"); //SMTP Host
props.put("mail.smtp.socketFactory.port", "465"); //SSL Port
props.put("mail.smtp.socketFactory.class","javax.net.ssl.SSLSocketFactory"); //SSL Factory Class
props.put("mail.smtp.auth", "true"); //Enabling SMTP Authentication
props.put("mail.smtp.port", "465"); //SMTP Port
Authenticator auth = new Authenticator() {
//override the getPasswordAuthentication method
protected PasswordAuthentication getPasswordAuthentication() {
return new PasswordAuthentication(fromEmail, password);
}
};
Session session = Session.getDefaultInstance(props, auth);
System.out.println("Session created");
EmailUtil.sendEmail(session, toEmail,"You have Requested to Reset the Password", "Dear User,"
+"Your New password for Attendance Application is "+RstPsw);
}
}
```

Fig. 4.37 Java Code for SMTP Email Creation and Sending of Reset Password



```java
import javax.activation.*;
import javax.mail.*;
import javax.mail.internet.*;
import java.util.*;
public class EmailUtil {
    public static void sendEmail(Session session, String toEmail, String subject, String body){
        try{
          MimeMessage msg = new MimeMessage(session);
          msg.addHeader("Content-type", "text/HTML; charset=UTF-8");
          msg.addHeader("format", "flowed");
          msg.addHeader("Content-Transfer-Encoding", "8bit");
          msg.setFrom(new InternetAddress("vasuprada93@gmail.com", "ATTENDANCE SYSTEM"));
          msg.setReplyTo(InternetAddress.parse("vasuprada93@gmail.com", false));
          msg.setSubject(subject, "UTF-8");
          msg.setText(body, "UTF-8");
          msg.setSentDate(new Date());
          msg.setRecipients(Message.RecipientType.TO, InternetAddress.parse(toEmail, false));
          System.out.println("Message is ready");
          Transport.send(msg);
          System.out.println("EMail Sent Successfully!!");
        }
        catch (Exception e) {
            e.printStackTrace(); }
       }
   }
```

Fig. 4.38 Java Code for Sending of Reset Password via Email

4.5.3 Advantages of the project

This project is built to provide the following advantages.

- Generation of Attendance report at the click of a button.
- Provides user the ability to change his password using the “Reset Password” operation.
- Isolation and security of data among user levels.

4.5.4 Limitations of the project

The project has its own limitations

- This module does not offer Web-based interface.
- Active semester data can only be viewed.

Chapter 5

CONCLUSION AND FUTURE ENHANCEMENTS

The Student Attendance System focuses on providing the STAFF the ability to take attendance and the STUDENT to view attendance. The ADMIN configures the database with critical data prior to the use of the Student Attendance System application. The system is tested and re-tested with varying constraints to ensure its effectiveness and provide error free functionality to the end user.

It assures saving of time in performing maintenance operations in case of attendance. Hence, the Student Attendance System serves as an interesting tool that caters to all user's requirements when they are taking/viewing attendance and provides accurate class percentage attendance and daily class attendance reports.

- It can be tightly integrated with class timetable module
- It can integrated with college administration system.
- This module can be extended to offer a Web-based interface.
- Functionality can be extended to include examinations module and HR module.

REFERENCES

1. Raghu Ramakrishnan and Johannes Gehrke , Database Management Systems , McGRAW HILL , 3rd Edition.
2. Ramez Elmasri and Shamkant B. Navathe , Fundamentals of Database Systems , Pearson , 7th Edition
3. Herbert Schildt , Java:The Complete Reference , McGRAW HILL , 7th Edition.
4. For Database and Java SDK: https://www.oracle.com/index.html
5. For Java IDE tool: https://www.bluej.org/
6. For Java IDE tool: https://www.eclipse.org/