A PROJECT REPORT

ON

EMPLOYEE PERFORMANACE SYSTEM

Submitted in partial fulfillment of the requirements for the award of

ANUDIP FULL STACK DEVELOPMENT TRAINING

UNDER THE GUIDANCEOF

G.PREETHA, Trainer

SUBMITTED BY:

M.Bhagyalaxmi(AF0339449)

INDEX

1. SYNOPSIS

The Employee Performance Management System is designed to centralize and integrate various activities related to employee performance management.

The Employee Performance Management System maintains the following core activities and core processes of Times.

1. User Authentication:
2. Manager Menu
3. Employee Menu
4. Adding and Managing Employee and Evaluation Data
5. Display Employee Performance

The provided Java program is a command-line application for managing employee performance evaluation records. It allows both managers and employees to log in and perform various actions related to employee details and performance evaluations. Below is a synopsis of the key features and functionalities of the program:

1. User Authentication:

- The program includes a basic login system for both managers and employees. Users are required to enter their usernames and passwords.
- Manager and employee login credentials are stored in separate tables: MANAGER_LOGIN and EMPLOYEE_LOGIN.

2. Manager Menu:

❖ After a manager successfully logs in, they have access to a menu with the following options:

- Add Employee Details: Allows the manager to add new employee records to the database, including details like name, email, gender, hire date, designation, salary, and mobile number.
- Add Evaluation Details: Enables the manager to add performance evaluation details for employees, including evaluation date, rating, and feedback.
- Update Evaluation Details: Allows the manager to update existing evaluation details, including the evaluation date, rating, and feedback, for a specific employee.
- Delete Employee: Allows the manager to delete employee records from the database using the employee's ID.
- Delete Evaluation Details: Allows the manager to delete evaluation records from the database using the evaluation ID.
- Logout: Logs the manager out of the system and returns to the main menu.

3. Employee Menu:

❖ After an employee successfully logs in, they have access to a menu with the following options:

- Display Employee Performance: Displays a list of employee performance details, including their evaluation date, rating, and feedback.
- Logout: Logs the employee out of the system and returns to the main menu.

4. Adding and Managing Employee and Evaluation Data:

- Managers can add new employee and evaluation records to the database.
- Managers can update evaluation details, but only for existing records.
- Managers can delete employee and evaluation records based on their respective IDs.

5. Display Employee Performance:

- Employees can view their own performance details, including evaluation date, rating, and feedback.

Users can choose to exit the program, which closes the database connection and terminates the application.

Access to employee performance details may be role-based, ensuring that only authorized personnel can view this sensitive information.
Data security and access control mechanisms are typically in place to protect employee privacy and maintain confidentiality.

2. INTRODUTION

The introduction of employee performance refers to the initial phase of evaluating and managing an employee's job-related achievements, behavior, and contributions within an organization. It typically encompasses the establishment of performance expectations, setting objectives, and outlining the framework for ongoing performance assessment and feedback.

Employee performance refers to the act of recognizing and appreciating an employee's efforts, achievements, and contributions within the workplace.

Employee performance is defined as how well a person executes their job duties and responsibilities. Many companies assess their employees' performance on an annual or quarterly basis to define certain areas that need improvement and to encourage further success in areas that are meeting or exceeding expectations.

3. MODULAR DESCRIPTION

The Java program for managing employee performance evaluation can be modularized into the following main modules:

Database Connection Module:

- Responsible for establishing a connection to the MySQL database.
- Initializes the Connection and Statement objects for database interaction.

User Authentication Module:

- Handles user authentication for both managers and employees.
- Validates usernames and passwords by querying the database tables (MANAGER_LOGIN and EMPLOYEE_LOGIN).

Manager Menu Module:

- Provides functionality for managers after successful login.
- Displays a menu with options for adding employee details, adding evaluation details, updating evaluation details, deleting employee records, deleting evaluation records, and logging out.
- Invokes corresponding methods based on manager's choices.

Employee Menu Module:

- Provides functionality for employees after successful login.
- Displays a menu with an option to view their performance details.
- Invokes the displayEmployeePerformance() method when the employee chooses to view performance details.
- Allows employees to log out.

Employee Details Management Module:

- Includes methods for adding employee details (addEmployee()), which collects employee information and inserts it into the Employee table in the database.
- Supports deleting employee records (deleteEmployee()) based on the provided employee ID.

Evaluation Details Management Module:

- Contains methods for adding evaluation details (addEvaluation()), which collects evaluation information and inserts it into the Evaluation table in the database.
- Provides the ability to update evaluation details (updateEvaluation()) based on the provided evaluation ID.
- Allows for the deletion of evaluation records (deleteEvaluation()) based on the provided evaluation ID.

Employee Performance Display Module:

- Retrieves and displays a list of employee performance details by joining data from the Employee and Evaluation tables.
- Part of the displayEmployeePerformance() method.

Exit Module:

- Closes the database connection gracefully (exit() method) when the program exits.
- Exits the program after closing the connection.

These modular components divide the program's functionality into clear, organized sections, making it easier to understand, maintain, and extend. Each module focuses on a specific aspect of employee performance evaluation, enhancing the program's readability and maintainability.

4. SYSTEM REQUIREMENTS

4.1 Software Requirements

OperatingSystem	:	WINDOWS 10
Programming Language	:	Java
Database	:	MySQL
IDE	:	IDEs like Eclipse, IntelliJ IDEA

4.2 Hardware Requirements

Processor	:	Intel Core i3-2348M
CPU Speed	:	2.30 GHz
Memory	:	2 GB (RAM)

4.3 System Analysis

Introduction to System Analysis

System analysis is a process of gathering interpreting facts, diagnosing problems, and using facts to improve the system. The objective of the system analysis is to understand the important facts of current system by studying it in detail. To accomplish this objective the following have to be done.

- Learn the details of the system as well as procedures currently in practice.

- Develop insight into future demands of the organization on its growth; hike in Competition, evolving new financial structures, introduction of new technology, and changes in the customer needs.
- Documentation details of the current system for discussion and review by others.

- Evaluate effectiveness and efficiency of the current system and procedure taking into account the impact of anticipating future demands.

- Recommend any revisions and enhancements to the current system, indicating how they are justified. If appropriate, an entire new system may be purposed.

- Document the new system features at a level of details that allows others to understand its components and manage the new system developed.

4.4 System Design

The design of a system produces the details that state how a system meet the requirements identified during system analysis. System specialists often refer to this stage as logical design, in contrast to the process of developing program software, which is referred to as physical design.

Data Flow Diagrams have been used in the design of the system. Data Flow Diagram is a graphical tool used to describe and analyze the movement of data. The transformation of data from input to output, through processes may be described logically using these Data Flow Diagrams.

The DFD shown to the user must represent only the major functions being performed by the system. This is called Top Level DFD. If this process is complex enough, it can be broken further into different levels. This process can be continued till the process is simple. This is called the leveling of DFD‟s.

5. SYSTEM TESTING AND IMPLEMENTATION

5.1 System Testing

Theoretically, a new designed system should have all the pieces in working order, but in reality, each piece works independently. Now is the time to put all pieces into one system and test it to determine whether it meets the user‟s requirements. The purpose of the system is to consider all the likely variations to which it will be subjected and then push the system to its limits. It is tedious but necessary step in system development. One needs to be familiar with the following basic terms.

- **Unit Testing:** Unit Testing is testing changes made in an existing or a new program.

- **Sequential or Series Testing:** Sequential or Series Testing is checking the logic of one or more programs in the candidate system, where the output of one program will affect the processing done by another program.

- **System Testing:** System Testing is executing a program to check logic changes made in it and with the intention of finding errors making the program fail.

- **Acceptance Testing:** Acceptance Testing is running the system with live data by the actual user of the system.

Testing is vital to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved. In adequate testing or no-testing leads to errors that may not appear until months later. Another reason for system testing is its utility as a user-oriented vehicle before implementation. The best program and the user have communication barriers due to different backgrounds. The system tester (designer, programmer, or user) who has developed some computer mastery can bridge this barrier.

(i) Unit Testing:

This focuses on the smallest unit of software design. The module using the details design description as a guide; important control paths are tested to uncover errors within the boundary of the module.

Unit test consideration:

The module interface id tested to ensure that information properly flows into and out of the program unit under test. The local data structures are examined to ensure that the data stored temporarily maintains it integrity during all steps in an algorithm execution. Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing. The test of data flow across a module interface is required before any other test. If data do not enter and exit properly, all other tests are moot.

Unit Procedure:

- Unit test is normally considered adjunct to the coding style.

- After source level code has been developed, reviewed and verified for correct syntax, unit test case design begins. Each test case should be coupled with a set of expected results.
- Normally, a driver is a “main program” that accepts test case data, passes such data to the module to be tested and prints the relevant results. Stubs serve to replace modules that are subroutines called by the module to be tested. A Stub or „dummy stub program“ uses the subroutine module“s interface to do minimal data manipulation and returns.

- Unit testing is simplified when a module with high cohesion is designed. When a module addresses only one function, the number of test cases is reduced and errors can be more easily predicted and uncovered.

(ii) **Integration Testing:**

Integration is a systematic technique for constructing the program structure, while at the same time conducting tests to uncover errors associated with interfacing. The objective is to make unit-tested modules and build a program structure that has been dictated by design. Incremental integration is the program that is the program that is constructed and tested in small segments where errors are easier to isolate and correct.

Top down Integration:

Top-down integration is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main control module. Module subroutine to the main control module is incorporated into the structure either in a depth-first

manner is engaged for this system. Breadth-first incorporates all modules directly subroutine at each level, moving across the structure horizontally.

The integration process is performed in a series of five steps:

a. The main control module is used as a test driver and stubs are substituted for all modules directly subroutine to the main control module.
b. Depending on the integration approach selected (depth or breadth first) subroutine stub are replaced one at a time with actual modules.
c. Tests are conducted as each module is integrated.
d. On the completion of each set of test, another stub is replaced with the real module.
e. Registration testing is conducted to ensure that new errors have not been introduced.

(iii) Validation testing:

At the end of integration testing, the system is completely assembled as a package with interfacing errors corrected after which a final series of software tests namely validation testing begins. Validation succeeds when the software functions in a manner that can be reasonably expected by the user.

Criteria:

Software validation is achieved through black box tests that demonstrates conformity with requirements.

(iV) System Testing:

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different

purpose, all work should verify that all system elements have been properly integrated and perform allocated functions. Being the most important test, the performance test is covered briefly below:

a. Performance Testing:

For real-time systems, software that provides required function but does not conform to performance requirement is unacceptable. Performance testing is designed to test the run-time performance of software within the context of an integrated system. Performance testing occurs throughout all steps in the testing process. Even at the unit level, the performance of an individual module may be accessed as tests are conducted. However, it is not until all system elements are fully integrated that the true performance of a system can be ascertained.

b. Debugging:

Debugging is not testing not occurs as a consequence of testing, that is when a test case uncovers an error, debugging is the process that results in the removal of the error.

Normally three categories for debugging approaches are proposed:

a) Brute force
b) Back-tracking
c) Cause-elimination

a) Brute force:

This is the most common and least efficient method for isolating the cost of a software error. Brute-force debugging method is usually applied when all else fails. Using a "let the computer finding the error" philosophy, memory-dumps are taken, runtime traces are invoked and the program is loaded with WRITE (in this case message box) statements. In

the information that is produced, a clue is found leading to the cause of the error.

b) Back-tracking:

This is fairly common debugging approach that can be used successfully in small programs. Beginning at the site where a symptom has been uncovered, the source code is traced backwards (manually) until the site of the cost is found.

c) Cause-Elimination:

This approach is manifested by induction/deduction and introduces the concept of „binary partition". A „cause hypothesis" is devised and the error related data are used to prove or disprove the hypothesis. Alternatively, a list of all possible causes is developed, and tests are conducted to eliminate each. If initial tests indicate that a particular cause hypothesis shows promise, that data are refined in an attempt to isolate the path.

Each of the debugging approaches can be supplemented with debugging tools. A wide variety of debugging compilers, dynamic debugging aids (tracers), automatic test case generators, memory dumps and cross-reference maps can be applied. However, tools are not a substitute for careful evaluation, based on a complete software design document and clear source code.

5.2 SYSTEM IMPLEMENTATION

After proper testing and validation, the question arises whether the system can be implemented or not. Implementation includes all those activities that take place to convert from the old system to the new. The new system may be totally new, replacing an existing module or automated system, or it may be major modification to an existing system. In either case proper implementation is essential to provide a reliable to provide a reliable system to meet organization requirements.

All planing has now, be completed and the transformation to a fully operational system can commence. The first job will be writing, debugging documenting of all computer programs and their integration into a total system. The master and transaction files are decided, and this general processing of the system is established. Programming is complete when the programs conformed to the detailed specification.

When the system is ready for implementation, emphasis switches to communicate with the finance department staff. Open discussion with the staff is important form the beginning of the project. Staff can be expected to the concerned about the effect of the automation on their jobs and the fear of redundancy or loss of status must be allayed immediately. During the implementation phase it is important that all staff concerned be apprised of the objectives of overall operation of the system. They will need shinning on how computerization will change their duties and need to understand how their role relates to the system as a whole. An organization-training program is advisable; this can include demonstrations, newsletters, seminars etc.

The department should allocate a member of staff, who understands the system and the equipment, and should be made responsible for the smooth operation of the system. An administrator should coordinate the users to the system.

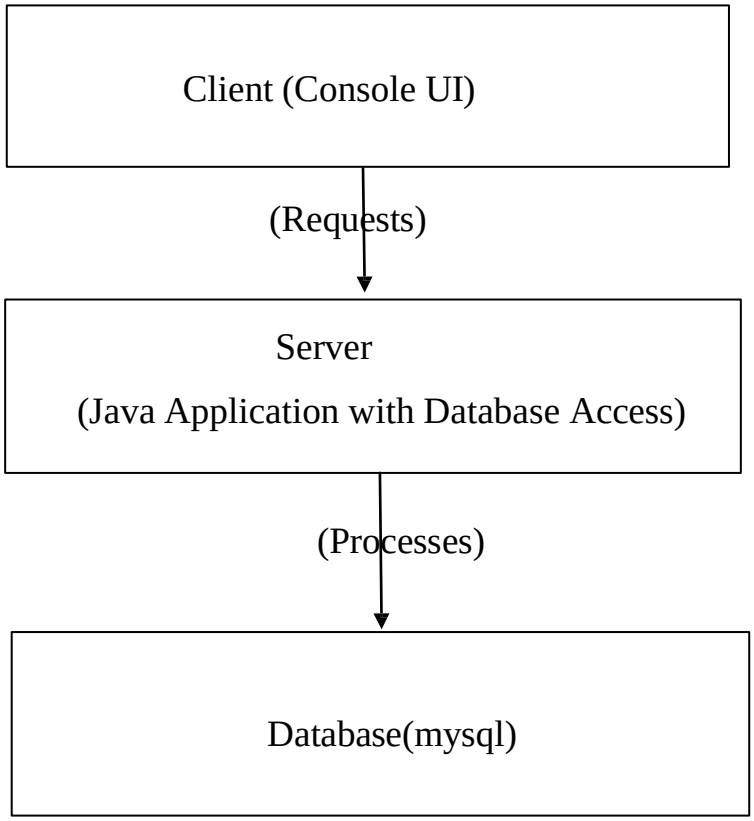
Users should be informed about new aspects of the system that will, affect them. The features of the system explained with the adequate documentation. New services such as security, on-line application from the back-ups must be advertise on the staff when the time is ripe.

Existing documents such as employee loan details should be entered into the new system. Since these files are very large, conversion of these may continue long after the system based on current files has been implemented. Hence we need to assign responsibility for each activity.

The system may come into full operation via number of possible routes. Complete change over at one point time is conceptually the most tidy. But this approach requires careful planning and coordination, particularly during the changeover. A phased approach, possible implementing the system of the section relating to one operation or procedure first and processing to more novel or complex subsystems in the fullness of time. These likely to be less traumatic. A phased approach gives the staff time to adjust to the new system. But depends on being able to split the system, without reliance on it. Thus approach is sensible when the consequences of failure are disastrous, but will require extra staff time. The fourth angle, is pilot operation permits any problems to be tackled on a smaller scale operation. Pilot operation generally means the implementation of the complete system, but at one location or branch only.

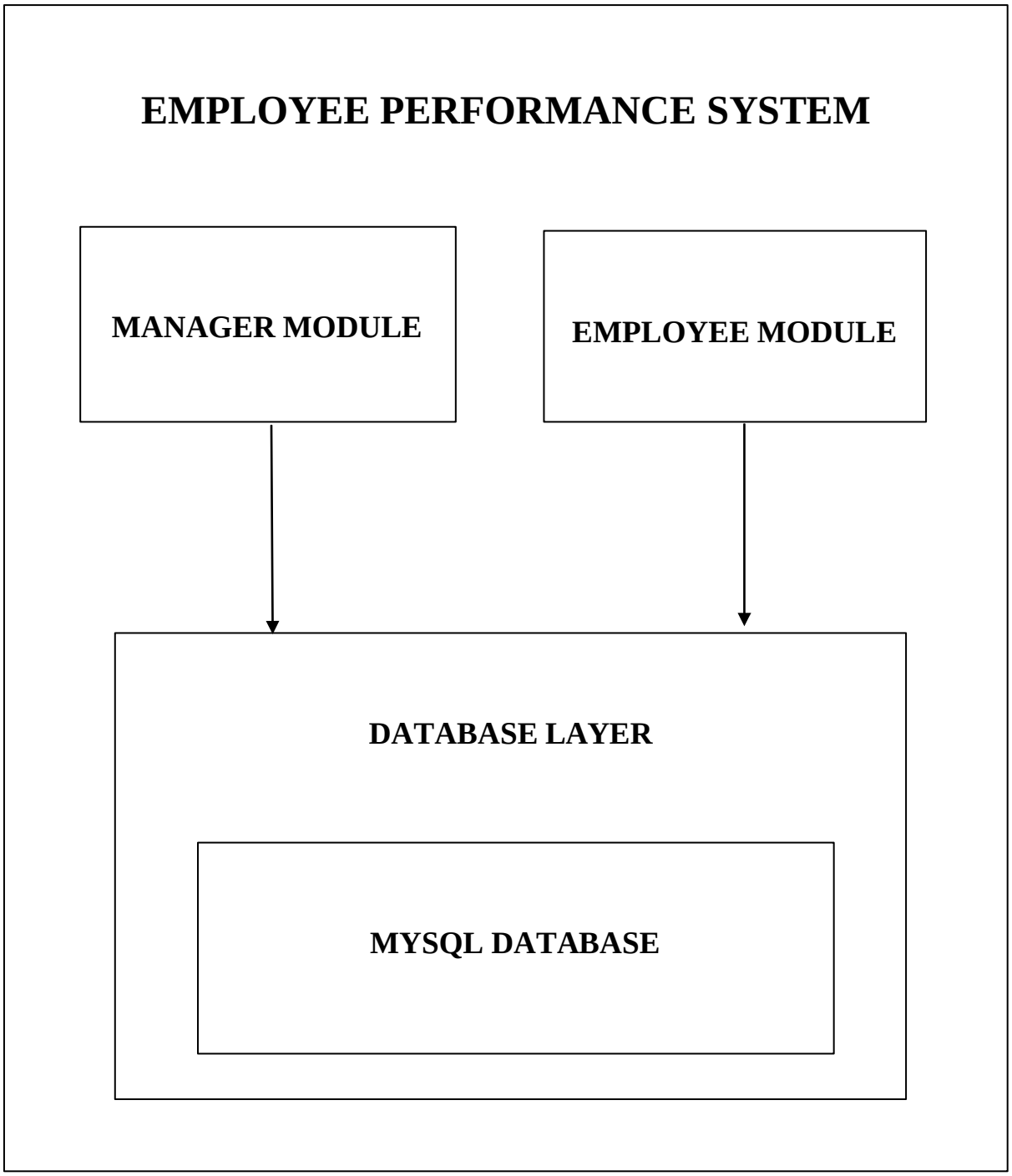
6. SOFTWARE ARCHITECTURE

The following diagram illustrates the links between various components involved in this system:



In the following sections the salient features of each of their components are discussed:

- Java
- JDBC
- MY SQL



This architectural representation provides an overview of the key components and their interactions within the Employee Performance System. You can use this as a starting point to create a more detailed and visually appealing software architecture diagram using a diagramming tool.

6.1 JAVA

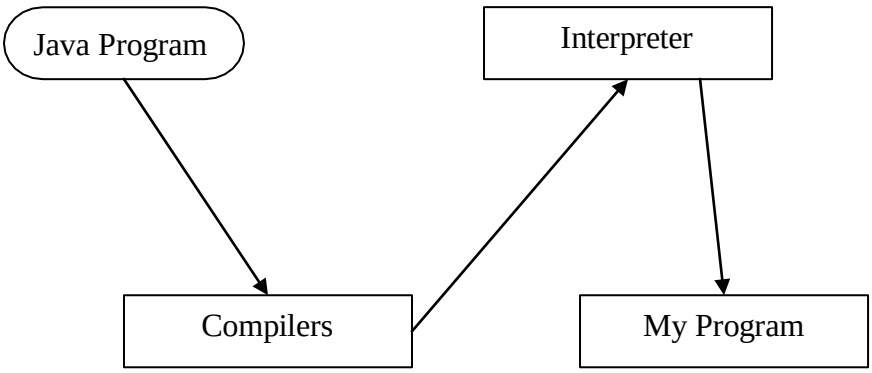
WHAT IS JAVA?

Java ha two things: a programming language and a platform.Java is a high-level programming language that is all of the following

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

Java is also unusual in that each Java program is both compiled and interpreted. With a compile, you translate a Java program into an intermediate language called Java bytecodes the platform-independent code instruction is passed and run on the computer.

Compilation happens just once; interpretation occurs each time the program is executed. The figure illustrates how this works.



You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it‟s a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java byte codes help make “write once, run anywhere” possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh.

JAVA PLATFORM

A platform is the hardware of software environment in which a program runs. The Java platform differs from most other platforms in that it‟s a software only platform that runs on the top of other, hardware-based platform. Most other platforms are described as a combination of hardware and operating system.

The Java platform has two components:

1. The Java Virtual Machine (Java VM)
2. The Java Application Programming Interface (Java API)

You have already been introduced to the Java VM. It‟s the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets.

The Java API is grouped into libraries (package) of related components. The next sections, what can Java do? Highlights each area of functionally provided by the package in the Java API.

The following figure depicts a Java program, such as an application or applet, that‟s running on the Java platform. A special kind of application known as a server serves and supports clients on a network. Examples of the servers include Web Servers, proxy servers, and mail servers, print servers, and boot servers. Another specialized program is a Servlet. Servlets are similar to applets in that they are runtime extensions of the application. Instead of working in browsers, though, Servlets run with in Java Web Servers, configuring of tailoring the server.

How does the Java API support all of these kinds of programs? With packages of software components, that provides a wide range of functionality. The API is the API included in every full implementation of the platform

The core API gives you the following features:

1. The Essentials: Objects, Strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
2. Applets: The set of conventions used by Java applets.
3. Networking URL‟s TCP and UDP sockets and IP addresses.
4. Internationalization: Help for writing programs that can be localized for users.

Worldwide programs can automatically adept to specific locates and be displayed in the appropriate language.

JAVA PROGRAM

- Java API
- Java Virtual Machine
- Java Program
- Hard Ware

API and Virtual Machine insulates the Java program from hardware dependencies. As a platform-independent environment, Java can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and Just-in-time-byte-code compilers can bring Java‟s performance close to the native code without threatening portability.

WHAT CAN JAVA DO?

However, Java is not just for writing cut, entertaining applets for the World Wide Web (WWW). Java is a general purpose, high-level programming language and a powerful software platform. Using the fineries Java API, you can write many types of programs.

The most common types of program are probably applets and application, where a Java application is a standalone program that runs directly on the Java platform.

Security:

Both low-level and high-level, including electronic signatures, public/private key management, accesses control, and certificate.

6.2 JAVA DATABASE CONNECTIVITY (JDBC)

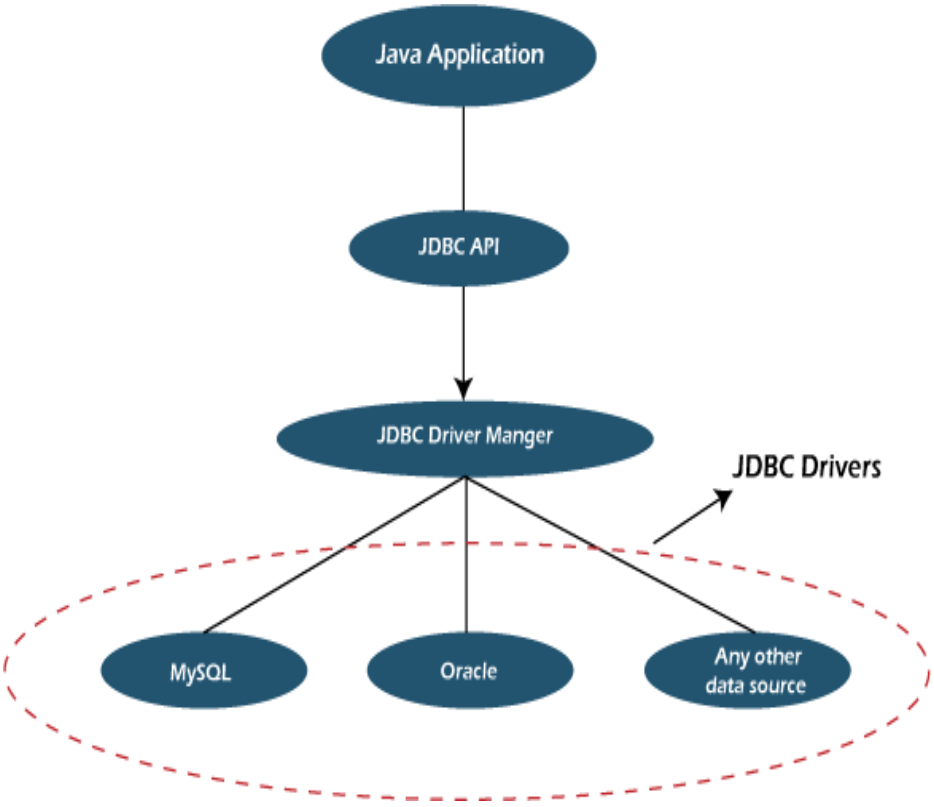
JDBC AND ODBC IN JAVA:

Most popular and widely accepted database connectivity called Open Database Connectivity (ODBC) is used to access the relational databases. It offers the ability to connect to almost all the databases on almost all platforms. Java applications can also use this ODBC to communicate with a database. Then we need JDBC why? There are several reasons:

- ODBC API was completely written in C language and it makes an extensive use of pointers. Calls from Java to native C code have a number of drawbacks in the security, implementation, robustness and automatic portability of applications.
- ODBC is hard to learn. It mixes simple and advanced features together, and it has complex options even for simple queries.
- ODBC drivers must be installed on client‟s machine.

Architecture of JDBC:

JDBC Architecture contains three layers:



1. Application Layer: Java program wants to get a connection to a database. It needs the information from the database to display on the screen or to modify the existing data or to insert the data into the table.
2. Driver Manager: The layer is the backbone of the JDBC architecture. When it receives a connection-request form.
3. The JDBC Application Layer: It tries to find the appropriate driver by iterating through all the available drivers, which are currently registered with Device Manager. After finding out the right driver, it connects the application to appropriate database.

4. JDBC Driver layers: This layer accepts the SQL calls from the application and converts them into native calls to the database and vice-versa. A JDBC Driver is responsible for ensuring that an application has consistent and uniform m access to any database.

When a request received by the application, the JDBC driver passes the request to the ODBC driver, the ODBC driver communicates with the database, sends the request, and gets the results. The results will be passed to the JDBC driver and in turn to the application. So, the JDBC driver has no knowledge about the actual database, it knows how to pass the application request o the ODBC and get the results from the ODBC.

The JDBC and ODBC interact with each other, how? The reason is both the JDBC API and ODBC are built on an interface called "Call Level Interface" (CLI). Because of this reason, the JDBC driver translates the request to an ODBC call. The ODBC then converts the request again and presents it to the database. The results of the request are then fed back through the same channel in reverse.

JDBC DRIVERS:

JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:

1. JDBC-ODBC bridge driver
2. Native-API driver (partially java driver)
3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver)

1. JDBC-ODBC bridge driver:

The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. This is now discouraged because of thin driver.

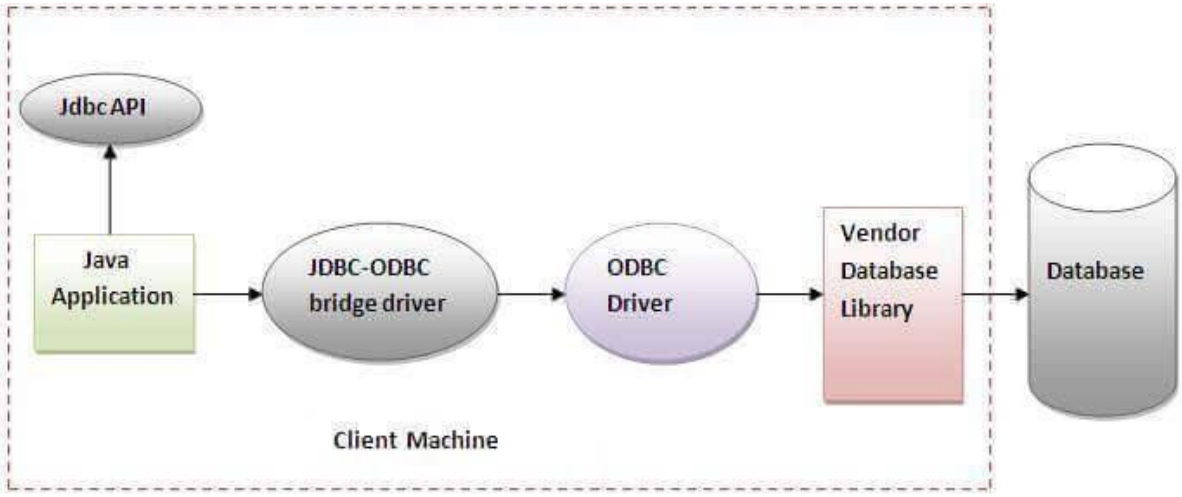


Figure- JDBC-ODBC Bridge Driver

Oracle does not support the JDBC-ODBC Bridge from Java 8. Oracle recommends that you use JDBC drivers provided by the vendor of your database instead of the JDBC-ODBC Bridge.

Advantages:

- Easy to use.
- Can be easily connected to any database.

Disadvantages:

- Performance degraded because JDBC method call is converted into the ODBC function calls.
- The ODBC driver needs to be installed on the client machine.

2.Native-API driver(partially java driver):

The Native API driver uses the client-side libraries of the database. The driver converts JDBC method calls into native calls of the database API. It is not writtenentirely in java.

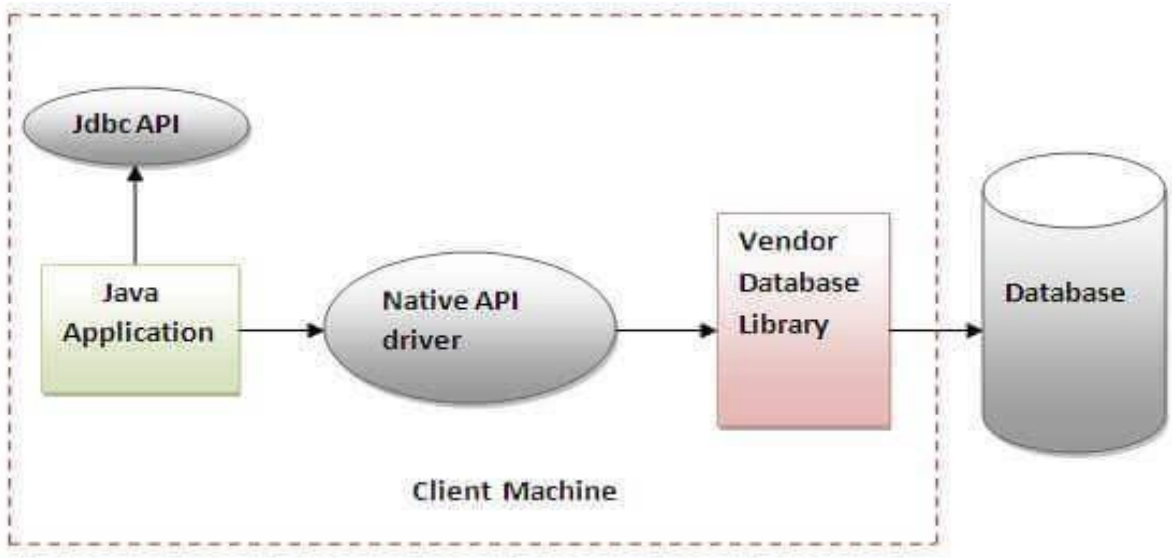


Figure- Native API Driver

Advantage:

- performance upgraded than JDBC-ODBC bridge driver.

Disadvantage:

- The Native driver needs to be installed on the each client machine.
- The Vendor client library needs to be installed on client machine.

3.Network Protocol driver:

The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. It is fully written in java.

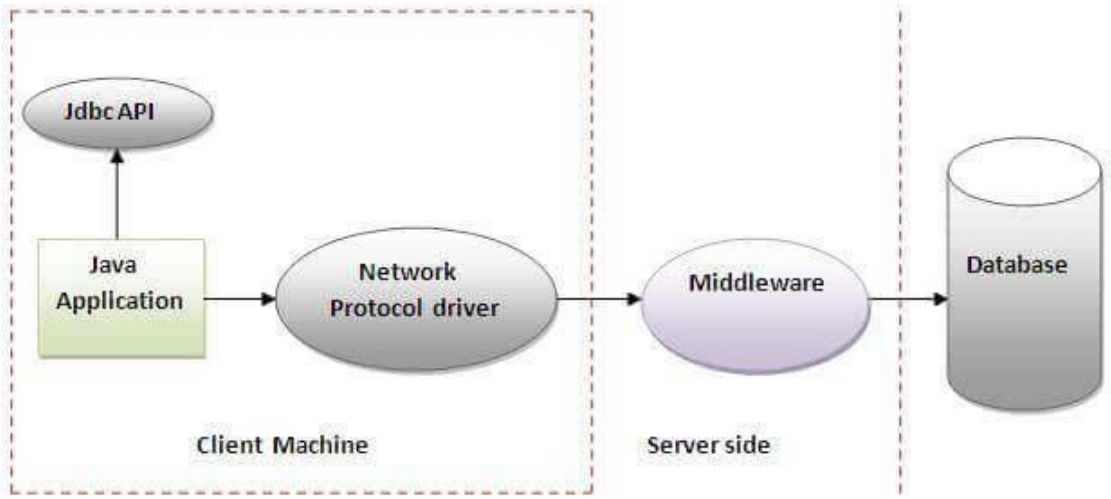


Figure- Network Protocol Driver

Advantage:

- No client side library is required because of application server that can perform many tasks like auditing, load balancing, logging etc.

Disadvantages:

- Network support is required on client machine.
- Requires database-specific coding to be done in the middle tier.
- Maintenance of Network Protocol driver becomes costly because it requires database-specific coding to be done in the middle tier.

4. Thin driver:

The thin driver converts JDBC calls directly into the vendor-specific database protocol. That is why it is known as thin driver. It is fully written in Java language.

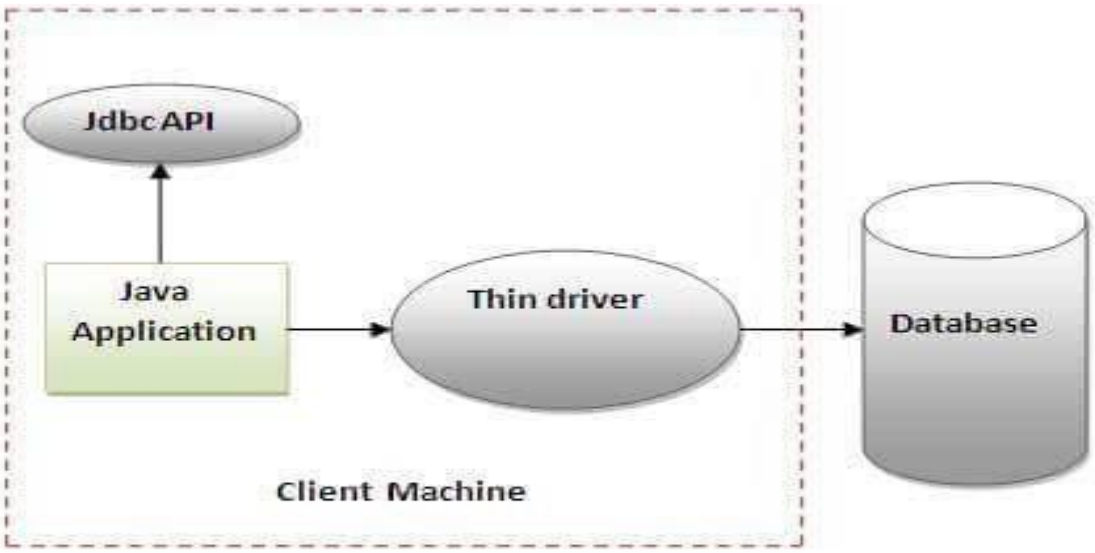


Figure- Thin Driver

Advantage:

- Better performance than all other drivers.
- No software is required at client side or server side.

Disadvantage:

- Drivers depend on the Database

When a request received by the application, the JDBC driver passes the request to the ODBC driver, the ODBC driver communicates with the database, sends the request, and gets the results. The results will be passed to the JDBC driver and in turn to the application. So, the JDBC driver has no knowledge about the actual database, it knows how to pass the application request o the ODBC and get the results from the ODBC.

The JDBC and ODBC interact with each other, how? The reason is both the JDBC API and ODBC are built on an interface called “Call Level Interface” (CLI). Because of this reason, the JDBC driver translates the request to an ODBC call. The ODBC then converts the request again and presents it to the database. The results of the request are then fed back through the same channel in reverse.

Why we using JDBC?

Java applications that interact with databases:

Database Connectivity: JDBC provides a standardized and consistent way to establish connections to relational databases from Java applications. This connectivity is crucial for applications that need to store, retrieve, or manipulate data stored in databases.

Database Independence**:** JDBC abstracts the underlying database-specific details, allowing developers to write database code in a database-agnostic manner. This means you can write Java code that can work with different relational database management systems (DBMS) without making significant changes to the codebase.

Data Retrieval and Manipulation**:** JDBC allows you to execute SQL queries and updates against the database. This enables your Java application to retrieve data from the database, insert, update, or delete records, and perform other database operations.

Parameterized Queries: JDBC supports prepared statements (via Prepared Statement), which are precompiled SQL queries with placeholders for parameters. This feature enhances security and performance by preventing SQL injection attacks and optimizing query execution.

Transaction Management**:** JDBC provides APIs for managing database transactions. You can use JDBC to start, commit, or roll back transactions, ensuring data integrity and consistency.

Error Handling: JDBC includes a mechanism for handling database-related exceptions (e.g., SQL Exception) that may occur during database operations. This allows you to gracefully handle errors and provide appropriate feedback to users.

Scalability: JDBC can be used in both small-scale and large-scale applications. It scales well to handle a wide range of database-related tasks, from simple queries to complex database interactions.

Integration with Java Ecosystem**:** JDBC integrates seamlessly with other Java technologies and frameworks. For example, it can be used in Java EE (Enterprise Edition) applications, Spring Framework applications, and more.

Legacy Systems**:** In many cases, organizations have legacy systems that rely on relational databases. JDBC provides a means to integrate and interact with these existing database systems while still leveraging Java's capabilities.

Flexibility: Developers have fine-grained control over the database interactions through JDBC. This flexibility allows you to optimize database access for your specific application needs.

6.3 Structured Query Language (SQL)

SQL (Pronounced Sequel) is the programming language that defines and manipulates the database. SQL databases are relational databases; this means simply the data is store in a set of simple relations. A database can have one or more table. You can define and manipulate data in a table with SQL commands. You use the data definition language (DDL) commands to creating and altering databases and tables.

You can update, delete or retrieve data in a table with data manipulation commands (DML). DML commands include commands to alter and fetch data.

The most common SQL commands include commands is the SELECT command, which allows you to retrieve data from the database.

In addition to SQL commands, the oracle server has a procedural language called PL/SQL. PL/SQL enables the programmer to program SQL statement. It allows you to control the flow of a SQL program, to use variables, and to write error-handling procedures.

What are MySQL's technical requirements?

These are MySQL's technical requirements:

• Manageability and Usability

• Flexible architecture

• High accessibility and replication

• Drivers

• MySQL Enterprise Manager

• JSON Compatibility

• High Availability and Replication

• Controlling storage and security

• Transactions and OLTP

• High performance that is easy to use and manage

• MySQL Enterprise Protection

• Graphical Instruments

• Geo-Spatial Assistance

FEATURES OF SQL:

SQL (Structured Query Language) is a domain-specific language used for managing and manipulating relational databases. It is the standard language for interacting with relational database management systems (RDBMS) like MySQL, PostgreSQL, Oracle, SQL Server, and others. SQL has several key features that make it a powerful tool for working with databases:

- Data Querying:SQL allows users to retrieve data from a database by writing queries. These queries can range from simple SELECT statements to complex joins and sub queries, making it easy to filter, sort, and extract specific data sets.
- Data Manipulation: SQL provides commands for adding, updating, and deleting data within a database. These commands are essential for maintaining the integrity and accuracy of data.
- Data Definition: SQL includes commands for defining the structure of a database, such as creating tables, defining columns and their data types, and specifying constraints (e.g., primary keys, foreign keys).
- Data Control: SQL offers data control features that enable administrators to manage access to data. This includes creating user accounts, granting or revoking privileges, and setting up roles and permissions.
- Transaction Control:SQL supports transactions, which are sequences of one or more SQL statements that are executed as a single unit of work.

This ensures data consistency and integrity, allowing users to commit or roll back changes.

- Data Integrity: SQL enforces data integrity rules through constraints, such as primary keys, unique constraints, foreign keys, and check constraints. These rules ensure that data remains accurate and consistent.
- Indexing: SQL allows you to create indexes on columns to improve query performance. Indexes speed up data retrieval by providing a quick way to look up records based on indexed columns.
- Joins: SQL supports various types of joins (e.g., INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN) to combine data from multiple tables based on specified relationships, enabling complex data analysis.
- Aggregation: SQL provides aggregate functions (e.g., SUM, AVG, COUNT, MAX, MIN) that allow users to perform calculations on sets of data, making it possible to generate summary information from large datasets.
- Sub queries: SQL allows you to nest queries within other queries, known as sub queries. This feature is valuable for retrieving data that depends on the results of another query.
- Views: SQL supports the creation of views, which are virtual tables based on the results of a query. Views simplify complex queries, enhance data security, and provide a way to present data in a more user-friendly format.
- Stored Procedures and Functions: SQL allows you to create reusable code blocks called stored procedures and functions. These are useful for encapsulating complex logic within the database, promoting code reusability and maintainability.
- Triggers: Triggers in SQL are special types of stored procedures that are automatically executed in response to predefined events, such as data

modifications (INSERT, UPDATE, DELETE). Triggers are useful for enforcing business rules and auditing changes.

- Compatibility: SQL is a standardized language, and most RDBMS systems adhere to the SQL standard to some degree. This ensures a level of portability and familiarity across different database platforms.
- Concurrency Control: SQL provides mechanisms for handling concurrent access to data, such as locking and isolation levels, to maintain data consistency in multi-user environments.

These features collectively make SQL a versatile and powerful language for managing and querying relational databases, making it an essential tool for data management and analysis in a wide range of applications and industries.

WHY WE USING SQL COMMANDS?

SQL commands are used for various purposes in database management and data manipulation. They serve several essential functions and provide numerous benefits, which is why they are widely used:

- Data Retrieval: SQL SELECT statements allow users to retrieve data from a database. This is one of the fundamental uses of SQL, enabling users to extract specific information from large datasets.
- Data Modification: SQL provides commands for adding, updating, and deleting data in a database. These commands are crucial for maintaining the accuracy and integrity of data.
- Data Definition: SQL commands like CREATE TABLE, ALTER TABLE, and DROP TABLE are used to define and manage the structure of a database, including tables, columns, and constraints.

- Data Control: SQL commands like GRANT and REVOKE are used to manage access to data. Database administrators can grant or revoke privileges to control who can view or modify data.
- Data Aggregation: SQL includes aggregate functions (e.g., SUM, AVG, COUNT) that allow users to perform calculations on sets of data, which is essential for generating summary reports and statistics.
- Data Filtering: SQL WHERE clauses and conditions help users filter and narrow down results, allowing for precise data retrieval.
- Data Sorting: SQL commands can sort query results in ascending or descending order, making it easier to analyze data and present it in a meaningful way.
- Data Joins: SQL supports various types of joins (e.g., INNER JOIN, LEFT JOIN) to combine data from multiple tables based on specified relationships, facilitating complex data analysis.
- Data Validation: SQL constraints (e.g., primary keys, foreign keys, check constraints) ensure data integrity by enforcing rules and preventing invalid data entries.
- Data Indexing: SQL allows the creation of indexes on columns to improve query performance, enabling faster data retrieval.
- Data Transformation: SQL commands can be used to transform data, such as converting data types, performing calculations, and formatting output.
- Data Security: SQL commands can be used to define and enforce security policies, restrict unauthorized access, and protect sensitive data.
- Automation: SQL commands can be scripted and automated, making it easier to perform repetitive database tasks and batch processing.
- Reporting: SQL queries can be used to extract data for generating reports, charts, and dashboards, aiding decision-making processes.

- Scalability: SQL databases are highly scalable, allowing them to handle large volumes of data and grow as an organization's data needs increase.
- Consistency and Reliability: SQL databases provide transaction management and data consistency, ensuring that data remains accurate and reliable even in multi-user environments.
- Compatibility: SQL is a standardized language, and most relational database management systems (RDBMS) adhere to the SQL standard, providing a consistent interface across different database platforms.

Overall, SQL commands are a fundamental tool for managing, querying, and manipulating data within relational databases. They enable efficient data storage, retrieval, and analysis, making them essential for organizations and individuals working with structured data.

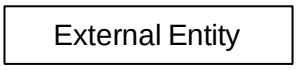
7. Data Flow Diagrams

Definition:

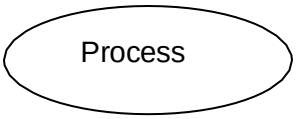
A data flow diagram is a graphical technique that depicts information flow and the transforms that applied as data move from input to output. The Data flow diagram used to represent a system or software at any level of abstraction. In fact DFDs may be portioned into levels.

A level of DFD, also called a context model, represents the entire software elements as a single bubble with input and output by arrow. A level of DFD is portioned into several bubbles with inter connecting arrows. Each of the process represented at level one is sub function of the over all depicted in the context model**.**

The DFD Notations:



Hardware person and other program.

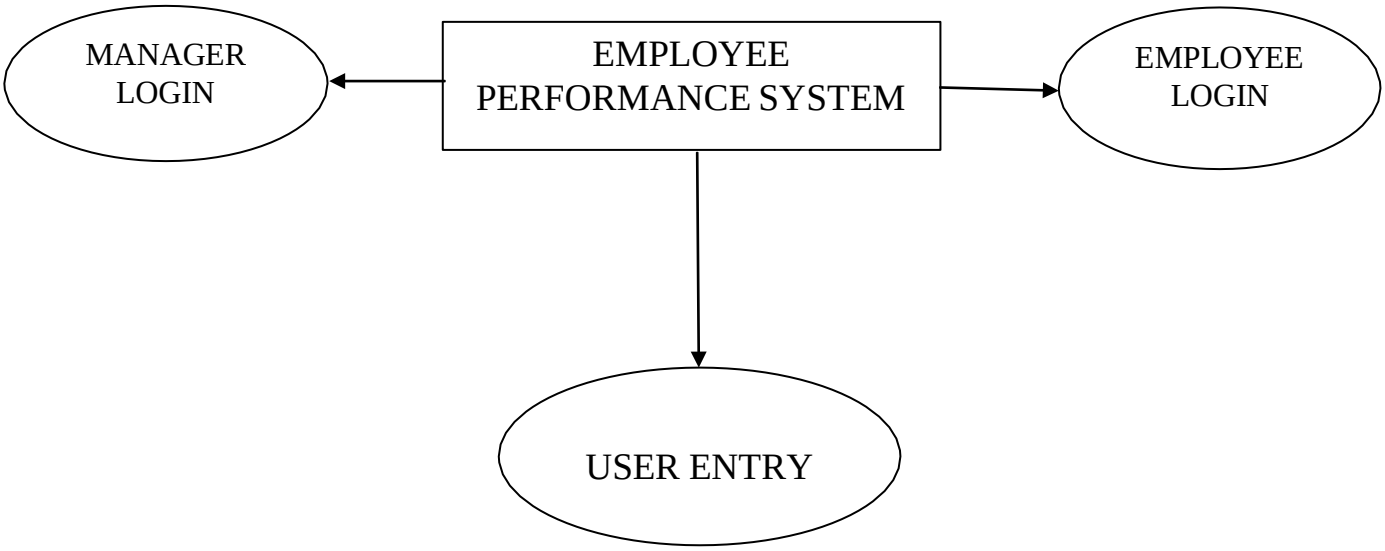


Information of the system to be modeled.

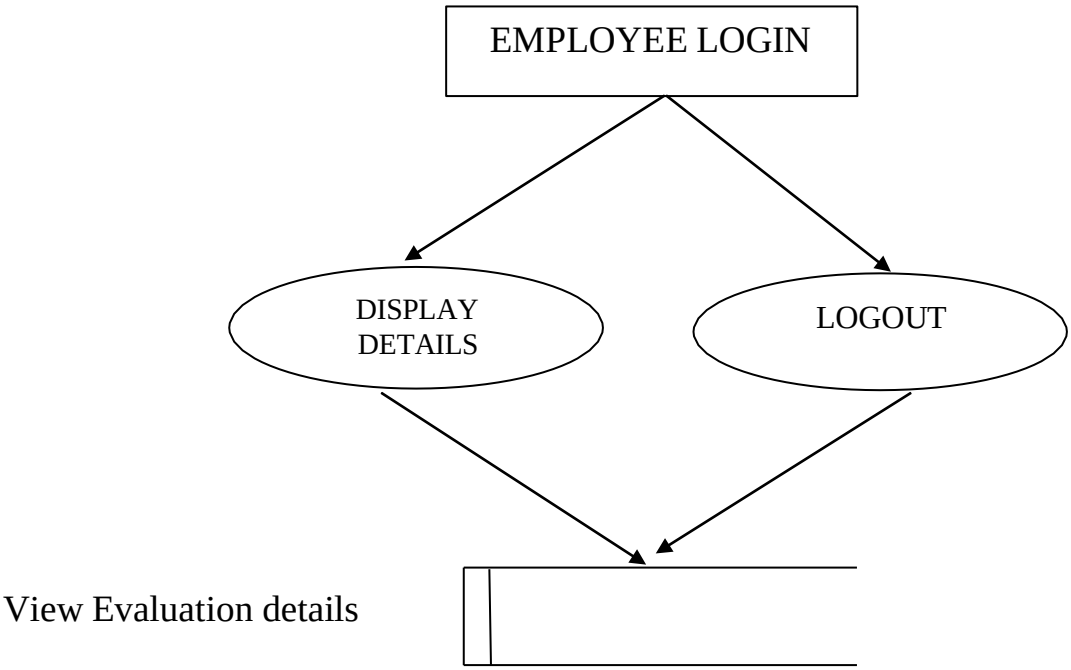
Data Item(s): Arrowhead indicates the direction of flow

Stored Information that is used by the s/w

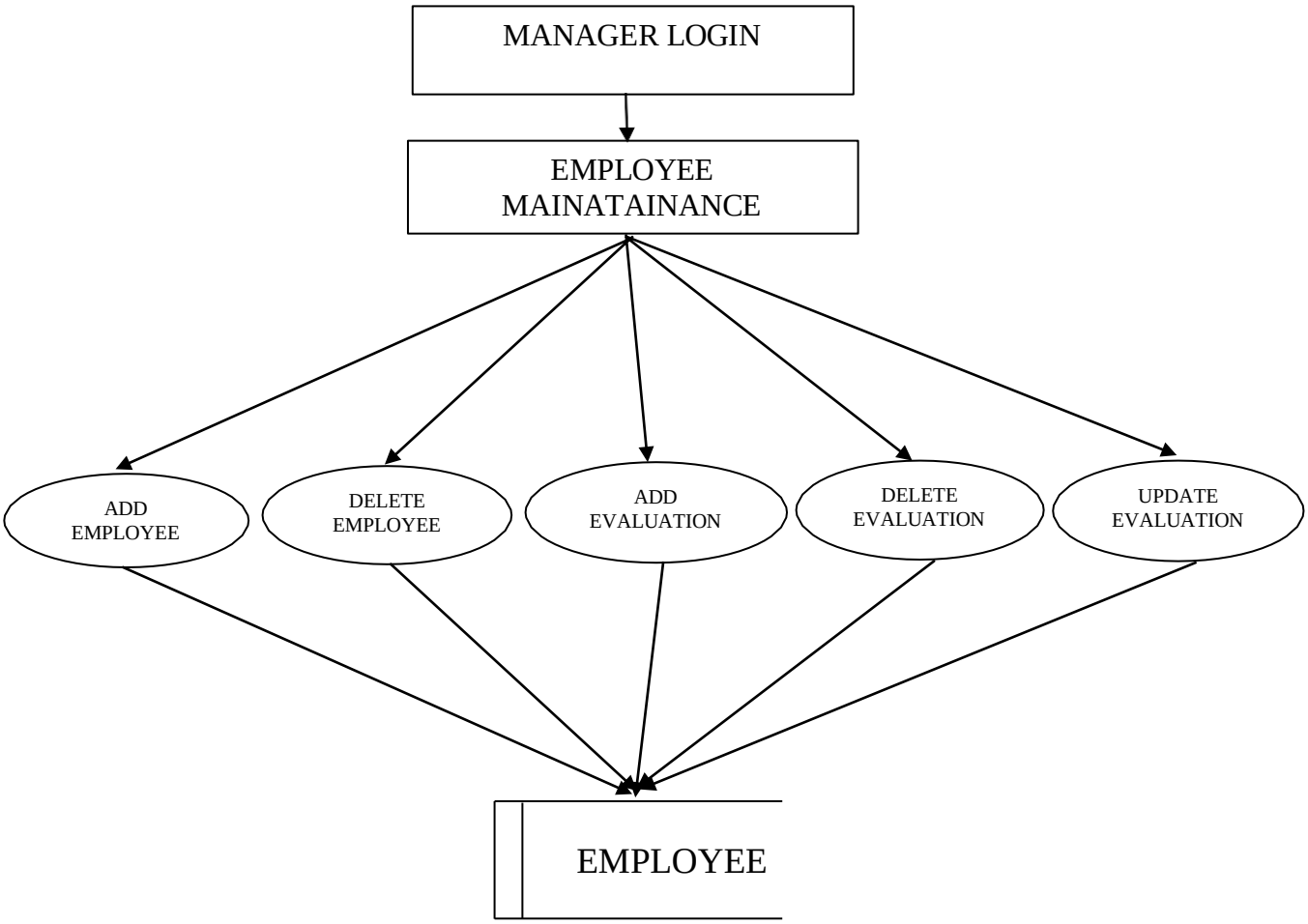
DATA FLOW DIAGRAM FOR OVERALL SYSTEM:



EMPLOYEE MENU:



MANAGER MENU:



8. DATABASE DESIGN

Designing an employee performance project using Core Java involves several components, such as creating a database schema, implementing Java classes setting up for database interaction, and managing employee performance data. Below is a step-by-step guide to help you get started:

Database Design:

Design the database schema to store employee information and performance data. Here's a basic example of tables you might use:

Manager Login:

CREATE TABLE MANAGER_LOGIN(USERNAME VARCHAR(255), PASSWORD VARCHAR(255));

insert into manager_login values('Anvitha', 'Anvi123');

Columns name	datatype	rule
username	varchar(255)	Not null
password	varchar(255)	Not null

Employee Login:

CREATE TABLE EMPLOYEE_LOGIN(Id INT PRIMARY KEY AUTO_INCREMENT, USERNAME VARCHAR(255), PASSWORD VARCHAR(255));

Columns name	datatype	rule
username	varchar(255)	Not null
password	varchar(255)	Not null

Employee:

```
CREATE TABLE Employee (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Email VARCHAR(255) NOT NULL,
    Gender VARCHAR(255) CHECK (Gender = 'Male' OR Gender = 'Female'),
    Hiredate DATETIME NOT NULL,
    Designation VARCHAR(255) NOT NULL,
    Salary FLOAT NOT NULL,
    Mobileno bigint NOT NULL);
```

Columns name	datatype	rule
Id	Auto_increment	Primary key
Name	varchar(255)	Not null
Email	varchar(255)	Not null

Gender	varchar(255)	Male/Female
Hiredate	datetime	Not null
Designation	varchar(255)	Not null
Salary	float	Not null
Mobileno	bigint	Not null

Evaluation Table:

```
CREATE TABLE Evaluation (
    Id INT  PRIMARY KEY,
    EvaluationDate DATE,
    Rating INT,
    Feedback varchar(255),
    FOREIGN KEY (Id) REFERENCES Employee(Id));
```

Columns name	datatype	rule
Id	int	Foreignkey
EvaluationDate	datetime	Not null
Rating	int	Not null
Feedback	varchar(255)	Not null

9. SOURCE CODE

```
packageEmployee_Perfomance;

importjava.sql.*;
importjava.util.Scanner;

publicclass Employee {
static Connection conn;
static Statement stmt;
static Scanner scanner = newScanner(System.in);

publicstaticvoidmain(String[] args) {
boolean loggedIn  =  false;

while(!loggedIn) {
try {
Class.forName("com.mysql.cj.jdbc.Driver");
conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/Empperf",
"root", "root");
stmt = conn.createStatement();
loggedIn = true; // Set loggedIn to true to exit the loop when the connection is
established
} catch (ClassNotFoundException | SQLExceptione) {
e.printStackTrace();
}
}
```

```
while (true) {

System.out.println("Employee Performance Evaluation");
System.out.println("1. Manager Login");
System.out.println("2. Employee Login");
System.out.println("3. Exit");
System.out.print("Enter your choice: ");
intchoice = scanner.nextInt();
scanner.nextLine(); // Consume the newline character

switch (choice) {
case 1:
managerLogin();
break;
case 2:
employeeLogin();
break;
case 3:
exit();
break;
default:
System.out.println("Invalid choice. Please try again.");
}
}
}

staticvoidmanagerLogin() {
booleanloggedIn = false;
```

```
while(!loggedIn) {
try {
System.out.println("Manager Login");
System.out.print("Enter username: ");
String username = scanner.nextLine();

System.out.print("Enter password: ");
String password = scanner.nextLine();

String selectQuery = "SELECT * FROM MANAGER_LOGIN WHERE
username = ? AND password = ?";
PreparedStatementpreparedStatement = conn.prepareStatement(selectQuery);
preparedStatement.setString(1, username);
preparedStatement.setString(2, password);

ResultSetresultSet = preparedStatement.executeQuery();

if (resultSet.next()) {
System.out.println("Login successful! Welcome, " + username);
loggedIn = true; // Set the flag to exit the loop
} else {
System.out.println("Login failed. Invalid username or password, Please try
again.");
}
} catch (SQLExceptione) {
e.printStackTrace();
}
}
```

```
while (loggedIn) {
System.out.println("\nManager Menu");
System.out.println("1. Add Employee Details");
System.out.println("2. Add Evaluation Details");
System.out.println("3. Update Evaluation Details");
System.out.println("4. Delete Employee");
System.out.println("5. Delete Evaluation Details");
System.out.println("6. Logout");
System.out.print("Enter your choice: ");
intmanagerChoice = scanner.nextInt();
scanner.nextLine(); // Consume the newline character

switch (managerChoice) {
case 1:
addEmployee();
break;
case 2:
addEvaluation();
break;
case 3:
updateEvaluation();
break;
case 4:
deleteEmployee();
break;
case 5:
deleteEvaluation();
break;
case 6:
```

```
loggedIn = false; // Logout and return to the main menu
break;
default:
System.out.println("Invalid choice. Please try again.");
}
}
}

staticvoidemployeeLogin() {
booleanloggedIn = false;

while(!loggedIn) {
try {
System.out.println("Employee Login");
System.out.print("Enter username: ");
String username = scanner.nextLine();

System.out.print("Enter password: ");
String password = scanner.nextLine();

String selectQuery = "SELECT * FROM EMPLOYEE_LOGIN WHERE
username = ? AND password = ?";
PreparedStatementpreparedStatement = conn.prepareStatement(selectQuery);
preparedStatement.setString(1, username);
preparedStatement.setString(2, password);

ResultSetresultSet = preparedStatement.executeQuery();

if (resultSet.next()) {
```

```
System.out.println("Login successful! Welcome, " + username);
loggedIn = true; // Set the flag to exit the loop
} else {
System.out.println("Login failed. Invalid username or password, Please try
again.");
}
} catch (SQLExceptione) {
e.printStackTrace();
}
}

while (loggedIn) {
System.out.println("\nEmployee Menu");
System.out.println("1. Display Employee Performance");
System.out.println("2. Logout");
System.out.print("Enter your choice: ");
intemployeeChoice = scanner.nextInt();
scanner.nextLine(); // Consume the newline character

switch (employeeChoice) {
case 1:
displayEmployeePerformance();
break;
case 2:
loggedIn = false; // Logout and return to the main menu
break;
default:
System.out.println("Invalid choice. Please try again.");
}
```

```
}
}

// Add other methods for managing employee details, evaluations, etc.

staticvoidaddEmployee() {
try {
System.out.println("Add Employee");

//  System.out.print("Enter Employee Id: ");
//  int Id = scanner.nextInt();

System.out.print("Enter Employee Name: ");
String name = scanner.nextLine();

System.out.print("Enter Email: ");
String email = scanner.nextLine();

System.out.print("Enter Gender: ");
String gender = scanner.nextLine();

System.out.print("Enter Hiredate: ");
String hiredate = scanner.nextLine();

System.out.print("Enter Designation: ");
String designation = scanner.nextLine();

System.out.print("Enter Salary: ");
Float salary = scanner.nextFloat();
```

```
System.out.print("Enter Mobile Number: ");
longmobileno = scanner.nextLong();



// You can add more fields

String insertQuery = "INSERT INTO Employee (Name, Email, Gender,
Hiredate, Designation, Salary, Mobileno) VALUES (?, ?, ?, ?, ?, ?, ?)";
PreparedStatementpreparedStatement = conn.prepareStatement(insertQuery);
//  preparedStatement.setInt(1, Id);
preparedStatement.setString(1, name);
preparedStatement.setString(2, email);
preparedStatement.setString(3, gender);
preparedStatement.setString(4, hiredate);
preparedStatement.setString(5, designation);
preparedStatement.setFloat(6, salary);
preparedStatement.setLong(7, mobileno);

introwsInserted = preparedStatement.executeUpdate();

if (rowsInserted> 0) {
System.out.println("Employee added successfully!");
} else {
System.out.println("Failed to add the Employee.");
}
} catch (SQLExceptione) {
e.printStackTrace();
```

```
}
}
staticvoidaddEvaluation() {
try {
System.out.println("Add Evaluation Details");

System.out.print("Enter Employee ID: ");
intId = scanner.nextInt();
scanner.nextLine();

System.out.print("Enter Evaluation Date: ");
String date = scanner.nextLine();

System.out.print("Enter Rating: ");
intrating = scanner.nextInt();
scanner.nextLine();

System.out.print("Enter Feedback: ");
String feedback = scanner.nextLine();


// You can add more fields

String insertQuery = "INSERT INTO Evaluation (Id, EvaluationDate, Rating,
Feedback) VALUES (?, ?, ?, ?)";
PreparedStatementpreparedStatement = conn.prepareStatement(insertQuery);
preparedStatement.setInt(1, Id);
preparedStatement.setString(2, date);
preparedStatement.setInt(3, rating);
```

```
preparedStatement.setString(4,  feedback);

introwsInserted = preparedStatement.executeUpdate();

if (rowsInserted> 0) {
System.out.println("Evaluation Detials added successfully!");
} else {
System.out.println("Failed to add the Evaluation Details.");
}
} catch (SQLExceptione) {
e.printStackTrace();
}
}

staticvoidupdateEvaluation() {
try {
System.out.println("Update an Evaluation Details");
System.out.print("Enter Evaluation Id to update: ");
String Id = scanner.nextLine();

// Check if the Evaluation with the provided Id exists
String selectQuery = "SELECT * FROM Evaluation WHERE Id=?";
PreparedStatementselectStatement = conn.prepareStatement(selectQuery);
selectStatement.setString(1, Id);

ResultSetresultSet = selectStatement.executeQuery();

if (resultSet.next()) {
// Employee exists, allow updates
```

```
System.out.print("Enter new Evaluation Date (leave empty to keep existing): ");
String Date = scanner.nextLine();

System.out.print("Enter new Rating (leave empty to keep existing): ");
String rating = scanner.nextLine();

System.out.print("Enter new Feedback (leave empty to keep existing): ");
String feedback = scanner.nextLine();

// Build the update query dynamically based on user input
StringBuilder updateQuery = newStringBuilder("UPDATE Evaluation SET ");
booleanneedsComma = false;

if(!Date.isEmpty()) {
updateQuery.append("EvaluationDate= ?");
needsComma = true;
}

if(!rating.isEmpty()) {
if (needsComma) {
updateQuery.append(", ");
}
updateQuery.append("Rating = ?");
needsComma = true;
}

if(!feedback.isEmpty()) {
if (needsComma) {
updateQuery.append(", ");
```

```
}
updateQuery.append("Feedback = ?");
}

updateQuery.append(" WHERE Id = ?");
PreparedStatementupdateStatement =
conn.prepareStatement(updateQuery.toString());

intparameterIndex = 1;

if(!Date.isEmpty()) {
updateStatement.setString(parameterIndex++, Date);
}

if(!rating.isEmpty()) {
updateStatement.setInt(parameterIndex++,  Integer.parseInt(rating));
}

if(!feedback.isEmpty()) {
updateStatement.setString(parameterIndex++, feedback);
}

updateStatement.setString(parameterIndex,  Id);

introwsUpdated = updateStatement.executeUpdate();

if (rowsUpdated> 0) {
System.out.println("Evaluation updated successfully!");
} else {
```

```
System.out.println("No changes made to the Evaluation.");
}
} else {
System.out.println("Evaluation with Id " + Id + " does not exist.");
}
} catch (SQLExceptione) {
// Handle SQLException properly, e.g., log the error
e.printStackTrace();
}
}

staticvoiddeleteEmployee() {
try {
System.out.println("Delete a Employee Details");
System.out.print("Enter Employee Id to delete: ");
String Id = scanner.nextLine();

// Check if the Employee with the provided Id exists
String selectQuery = "SELECT * FROM Employee WHERE Id = ?";
PreparedStatementselectStatement = conn.prepareStatement(selectQuery);
selectStatement.setString(1, Id);

ResultSetresultSet = selectStatement.executeQuery();

if (resultSet.next()) {
// Book exists, proceed with deletion
String deleteQuery = "DELETE FROM Employee WHERE Id = ?";
PreparedStatementdeleteStatement = conn.prepareStatement(deleteQuery);
deleteStatement.setString(1, Id);
```

```
introwsDeleted = deleteStatement.executeUpdate();

if (rowsDeleted> 0) {
System.out.println("Employee deleted successfully!");
} else {
System.out.println("Failed to delete the Employee.");
}
} else {
System.out.println("Employee with Id " + Id + " does not exist.");
}
} catch (SQLExceptione) {
e.printStackTrace();
}
}


staticvoiddeleteEvaluation() {
try {
System.out.println("Delete a Evaluation Details");
System.out.print("Enter Evaluation Id to Delete: ");
String Id = scanner.nextLine();

// Check if the Employee with the provided Id exists
String selectQuery = "SELECT * FROM Evaluation WHERE Id = ?";
PreparedStatementselectStatement = conn.prepareStatement(selectQuery);
selectStatement.setString(1, Id);

ResultSetresultSet = selectStatement.executeQuery();
```

```
if (resultSet.next()) {
// Employee exists, proceed with deletion
String deleteQuery = "DELETE FROM Evaluation WHERE Id = ?";
PreparedStatementdeleteStatement = conn.prepareStatement(deleteQuery);
deleteStatement.setString(1, Id);

introwsDeleted = deleteStatement.executeUpdate();

if (rowsDeleted> 0) {
System.out.println("Evaluation deleted successfully!");
} else {
System.out.println("Failed to delete the Evaluation.");
}
} else {
System.out.println("Evaluation with Id " + Id + " does not exist.");
}
} catch (SQLExceptione) {
e.printStackTrace();
}
}


staticvoiddisplayEmployeePerformance() {
try {
System.out.println("List of Employee Performance Details:");

// SQL query to retrieve employee performance details by joining two tables
```

```
String selectQuery = "SELECT e.Id, e.Name, e.Email, ev.EvaluationDate,
ev.Rating, ev.Feedback "
+ "FROM Employee e "
+ "INNER JOIN Evaluation ev ON e.Id = ev.Id";
PreparedStatementpreparedStatement = conn.prepareStatement(selectQuery);

ResultSetresultSet = preparedStatement.executeQuery();

// Check if there are any employee performance details in the database
booleanhasPerformanceDetails = false;
while (resultSet.next()) {
hasPerformanceDetails = true;
intId = resultSet.getInt("Id");
String Name = resultSet.getString("Name");
String Email = resultSet.getString("Email");
String date = resultSet.getString("EvaluationDate");
doublerating = resultSet.getDouble("Rating");
String feedback = resultSet.getString("Feedback");

// Display employee performance information

System.out.println("Employee ID: " + Id);
System.out.println("Employee Name: " + Name);
System.out.println("Employee Email: " + Email);
System.out.println("Employee Evaluation Date: " + date);
System.out.println("Performance Rating: " + rating);
System.out.println("Employee Feedback: " + feedback);
System.out.println(" ------------------- ");
}
```

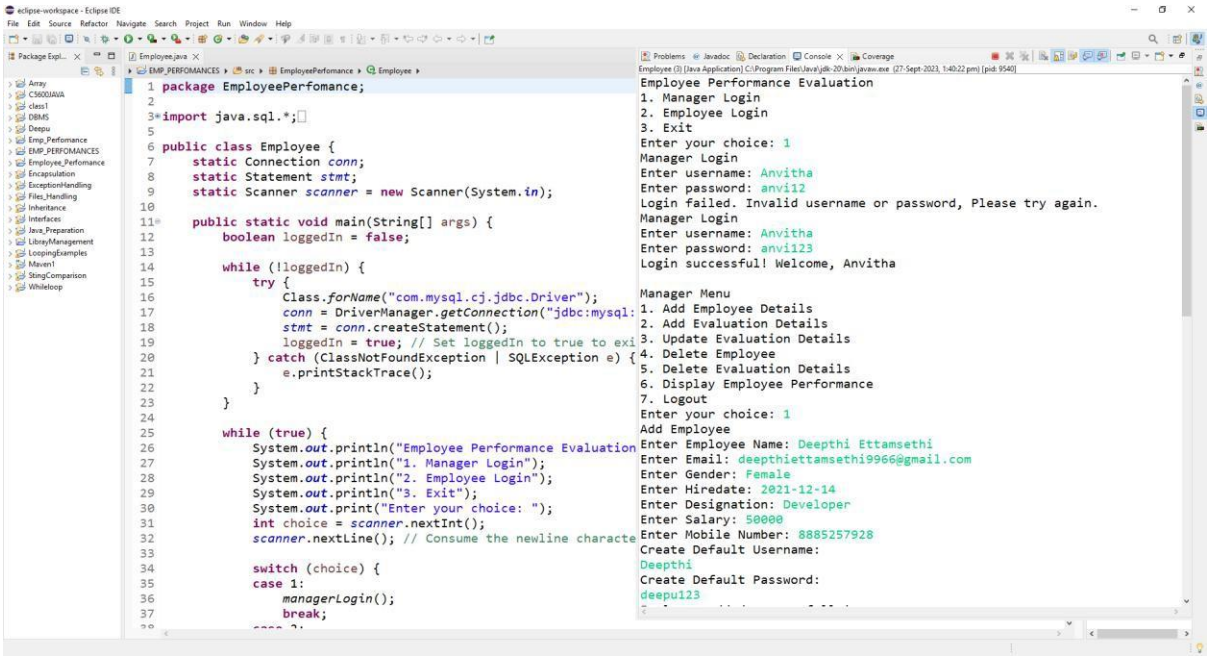
```
if(!hasPerformanceDetails) {
System.out.println("No employee performance details found in the database.");
}
} catch (SQLExceptione) {
e.printStackTrace();
}
}

staticvoidexit() {
try {
conn.close();
} catch (SQLExceptione) {
e.printStackTrace();
}
System.exit(0);
}
}
```

10. SCREEN LAYOUTS

Output:

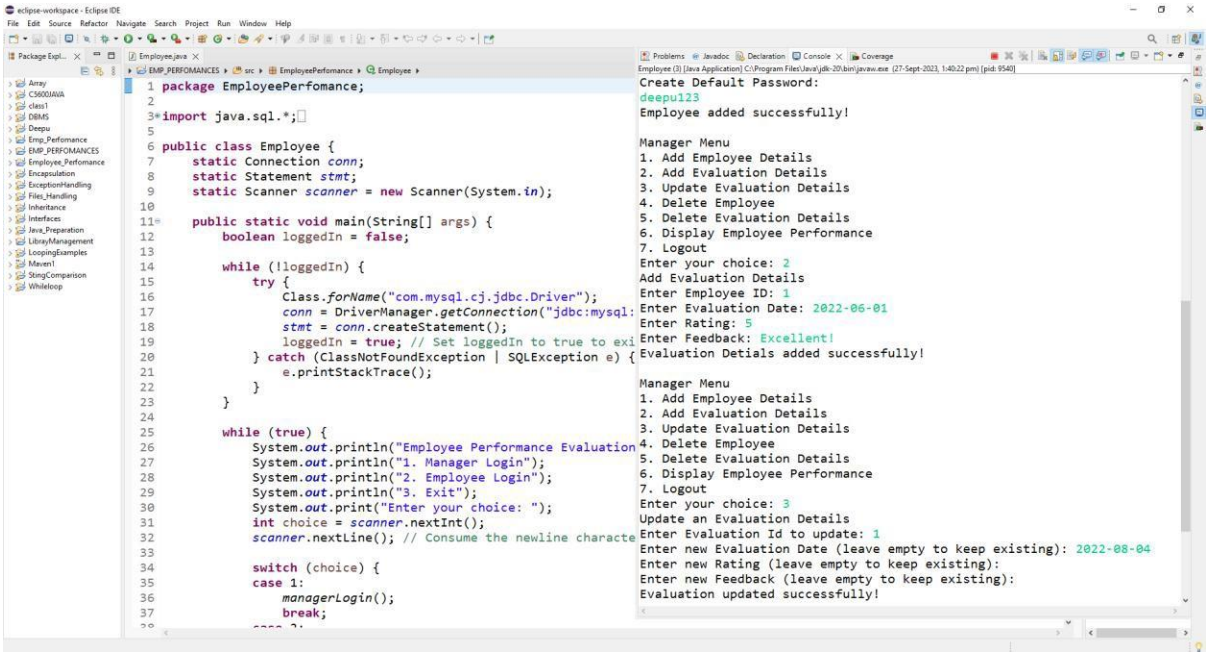
MySQL 8.0 Command Line Client

```
| product               |
| springdb              |
| stu_project           |
| test123               |
+-----------------------+
13 rows in set (0.01 sec)

mysql> Create database empperf;
Query OK, 1 row affected (0.05 sec)

mysql> use empperf;
Database changed
mysql> CREATE TABLE MANAGER_LOGIN(USERNAME VARCHAR(255), PASSWORD VARCHAR(255));
Query OK, 0 rows affected (0.13 sec)

mysql> insert into manager_login values('Anvitha', 'Anvi123');
Query OK, 1 row affected (0.03 sec)

mysql> CREATE TABLE EMPLOYEE_LOGIN(Id INT PRIMARY KEY AUTO_INCREMENT, USERNAME VARCHAR(255), PASSWORD VARCHAR(255));
Query OK, 0 rows affected (0.10 sec)

mysql> CREATE TABLE Employee (
    ->      Id INT AUTO_INCREMENT PRIMARY KEY,
    ->      Name VARCHAR(255) NOT NULL,
    ->      Email VARCHAR(255) NOT NULL,
    ->      Gender VARCHAR(255) CHECK (Gender = 'Male' OR Gender = 'Female'),
    ->      Hiredate DATETIME NOT NULL,
    ->      Designation VARCHAR(255) NOT NULL,
    ->      Salary FLOAT NOT NULL,
    ->      Mobileno bigint NOT NULL);
Query OK, 0 rows affected (0.09 sec)

mysql> CREATE TABLE Evaluation (
    ->     Id INT  PRIMARY KEY,
    ->      EvaluationDate DATE,
    ->      Rating INT,
    ->      Feedback varchar(255),
    ->      FOREIGN KEY (Id) REFERENCES Employee(Id));
Query OK, 0 rows affected (0.28 sec)

mysql> show tables;
+------------------+
| Tables_in_empperf |
+------------------+
| employee         |
| employee_login   |
| evaluation       |
| manager_login    |
+------------------+
4 rows in set (0.00 sec)

mysql> select * from manager_login;
+----------+----------+
| USERNAME | PASSWORD |
+----------+----------+
| Anvitha  | Anvi123  |
+----------+----------+
1 row in set (0.00 sec)

mysql> select * from employee_login;
Empty set (0.00 sec)

mysql>
```



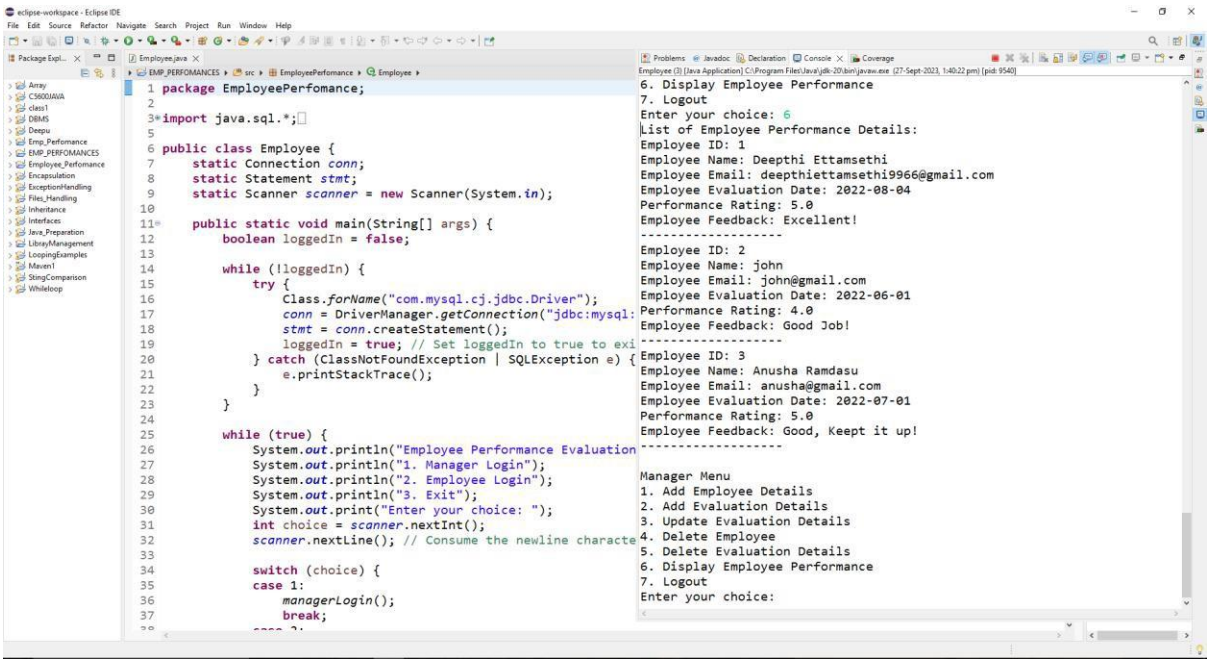
```
package EmployeePerfomance;

import java.sql.*;

public class Employee {
    static Connection conn;
    static Statement stmt;
    static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        boolean loggedIn = false;

        while (!loggedIn) {
            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
                conn = DriverManager.getConnection("jdbc:mysql:
                stmt = conn.createStatement();
                loggedIn = true; // Set loggedIn to true to exi
            } catch (ClassNotFoundException | SQLException e) {
                e.printStackTrace();
            }
        }

        while (true) {
            System.out.println("Employee Performance Evaluation
            System.out.println("1. Manager Login");
            System.out.println("2. Employee Login");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume the newline characte

            switch (choice) {
            case 1:
                managerLogin();
                break;
```

```
Employee Performance Evaluation
1. Manager Login
2. Employee Login
3. Exit
Enter your choice: 1
Manager Login
Enter username: Anvitha
Enter password: anvi12
Login failed. Invalid username or password, Please try again.
Manager Login
Enter username: Anvitha
Enter password: anvi123
Login successful! Welcome, Anvitha

Manager Menu
1. Add Employee Details
2. Add Evaluation Details
3. Update Evaluation Details
4. Delete Employee
5. Delete Evaluation Details
6. Display Employee Performance
7. Logout
Enter your choice: 1
Add Employee
Enter Employee Name: Deepthi Ettamsethi
Enter Email: deepthiettamsethi9966@gmail.com
Enter Gender: Female
Enter Hiredate: 2021-12-14
Enter Designation: Developer
Enter Salary: 50000
Enter Mobile Number: 8885257928
Create Default Username:
Deepthi
Create Default Password:
deepu123
```



```
package EmployeePerfomance;

import java.sql.*;

public class Employee {
    static Connection conn;
    static Statement stmt;
    static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        boolean loggedIn = false;

        while (!loggedIn) {
            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
                conn = DriverManager.getConnection("jdbc:mysql:
                stmt = conn.createStatement();
                loggedIn = true; // Set loggedIn to true to exi
            } catch (ClassNotFoundException | SQLException e) {
                e.printStackTrace();
            }
        }

        while (true) {
            System.out.println("Employee Performance Evaluation
            System.out.println("1. Manager Login");
            System.out.println("2. Employee Login");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume the newline characte

            switch (choice) {
            case 1:
                managerLogin();
                break;
```

```
Create Default Password:
deepu123
Employee added successfully!

Manager Menu
1. Add Employee Details
2. Add Evaluation Details
3. Update Evaluation Details
4. Delete Employee
5. Delete Evaluation Details
6. Display Employee Performance
7. Logout
Enter your choice: 2
Add Evaluation Details
Enter Employee ID: 1
Enter Evaluation Date: 2022-06-01
Enter Rating: 5
Enter Feedback: Excellent!
Evaluation Detials added successfully!

Manager Menu
1. Add Employee Details
2. Add Evaluation Details
3. Update Evaluation Details
4. Delete Employee
5. Delete Evaluation Details
6. Display Employee Performance
7. Logout
Enter your choice: 3
Update an Evaluation Details
Enter Evaluation Id to update: 1
Enter new Evaluation Date (leave empty to keep existing): 2022-08-04
Enter new Rating (leave empty to keep existing):
Enter new Feedback (leave empty to keep existing):
Evaluation updated successfully!
```

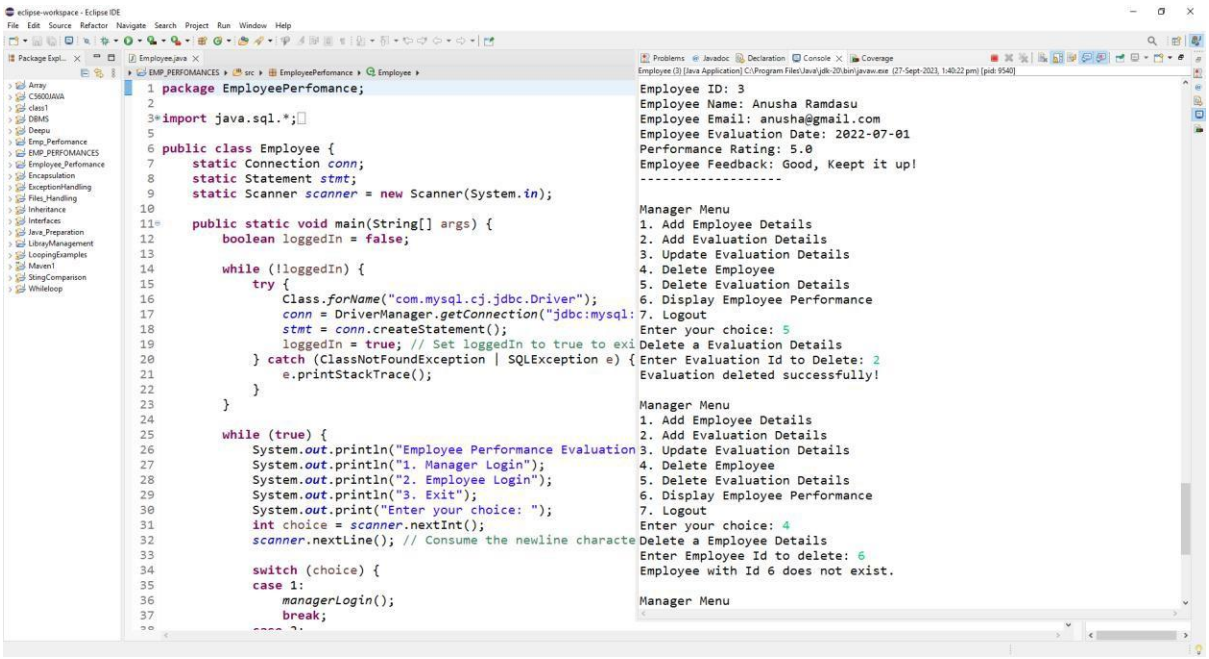


```
package EmployeePerfomance;

import java.sql.*;

public class Employee {
    static Connection conn;
    static Statement stmt;
    static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        boolean loggedIn = false;

        while (!loggedIn) {
            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
                conn = DriverManager.getConnection("jdbc:mysql:
                stmt = conn.createStatement();
                loggedIn = true; // Set loggedIn to true to exi
            } catch (ClassNotFoundException | SQLException e) {
                e.printStackTrace();
            }
        }

        while (true) {
            System.out.println("Employee Performance Evaluation
            System.out.println("1. Manager Login");
            System.out.println("2. Employee Login");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume the newline characte

            switch (choice) {
            case 1:
                managerLogin();
                break;
```

```
6. Display Employee Performance
7. Logout
Enter your choice: 6
List of Employee Performance Details:
Employee ID: 1
Employee Name: Deepthi Ettamsethi
Employee Email: deepthiettamsethi9966@gmail.com
Employee Evaluation Date: 2022-08-04
Performance Rating: 5.0
Employee Feedback: Excellent!
--------------------
Employee ID: 2
Employee Name: john
Employee Email: john@gmail.com
Employee Evaluation Date: 2022-06-01
Performance Rating: 4.0
Employee Feedback: Good Job!
--------------------
Employee ID: 3
Employee Name: Anusha Ramdasu
Employee Email: anusha@gmail.com
Employee Evaluation Date: 2022-07-01
Performance Rating: 5.0
Employee Feedback: Good, Keept it up!
--------------------

Manager Menu
1. Add Employee Details
2. Add Evaluation Details
3. Update Evaluation Details
4. Delete Employee
5. Delete Evaluation Details
6. Display Employee Performance
7. Logout
Enter your choice:
```

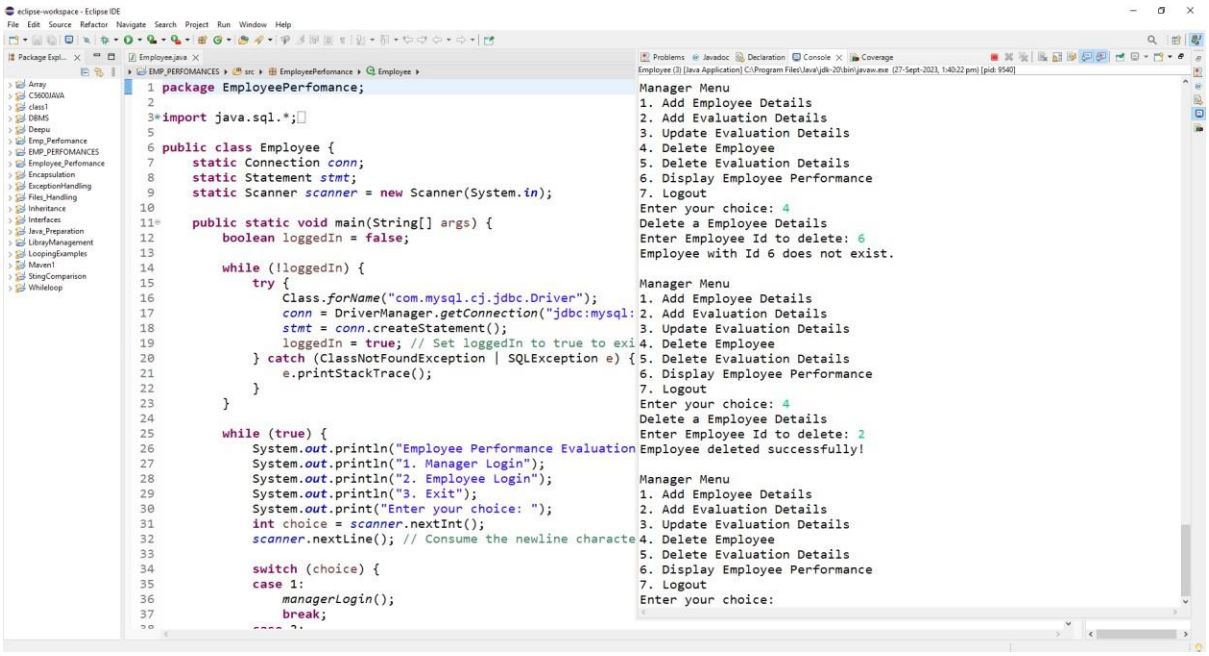
MySQL 8.0 Command Line Client

```
mysql> show tables;
+-------------------+
| Tables_in_empperf |
+-------------------+
| employee          |
| employee_login    |
| evaluation        |
| manager_login     |
+-------------------+
4 rows in set (0.00 sec)

mysql> select * from manager_login;
+----------+----------+
| USERNAME | PASSWORD |
+----------+----------+
| Anvitha  | Anvi123  |
+----------+----------+
1 row in set (0.00 sec)

mysql> select * from employee_login;
Empty set (0.00 sec)

mysql> select * from employee_login;
+----+----------+----------+
| Id | USERNAME | PASSWORD |
+----+----------+----------+
|  1 | Deepthi  | deepu123 |
|  2 | john     | john123  |
|  3 | Anusha   | anu123   |
+----+----------+----------+
3 rows in set (0.00 sec)

mysql> select * from employee;
+----+--------------------+---------------------------------+--------+---------------------+-------------+--------+------------+
| Id | Name               | Email                           | Gender | Hiredate            | Designation | Salary | Mobileno   |
+----+--------------------+---------------------------------+--------+---------------------+-------------+--------+------------+
|  1 | Deepthi Ettamsethi | deepthiettamsethi9966@gmail.com | Female | 2021-12-14 00:00:00 | Developer   |  50000 | 8885257928 |
|  2 | john               | john@gmail.com                  | Male   | 2022-06-01 00:00:00 | Tester      |  35000 | 9874563329 |
|  3 | Anusha Ramdasu     | anusha@gmail.com                | Female | 2021-11-15 00:00:00 | Analyst     |  50000 | 9632587749 |
+----+--------------------+---------------------------------+--------+---------------------+-------------+--------+------------+
3 rows in set (0.00 sec)

mysql> select * from evaluation;
+----+----------------+--------+--------------------+
| Id | EvaluationDate | Rating | Feedback           |
+----+----------------+--------+--------------------+
|  1 | 2022-08-04     |      5 | Excellent!         |
|  2 | 2022-06-01     |      4 | Good Job!          |
|  3 | 2022-07-01     |      5 | Good, Keept it up! |
+----+----------------+--------+--------------------+
3 rows in set (0.00 sec)

mysql>
```



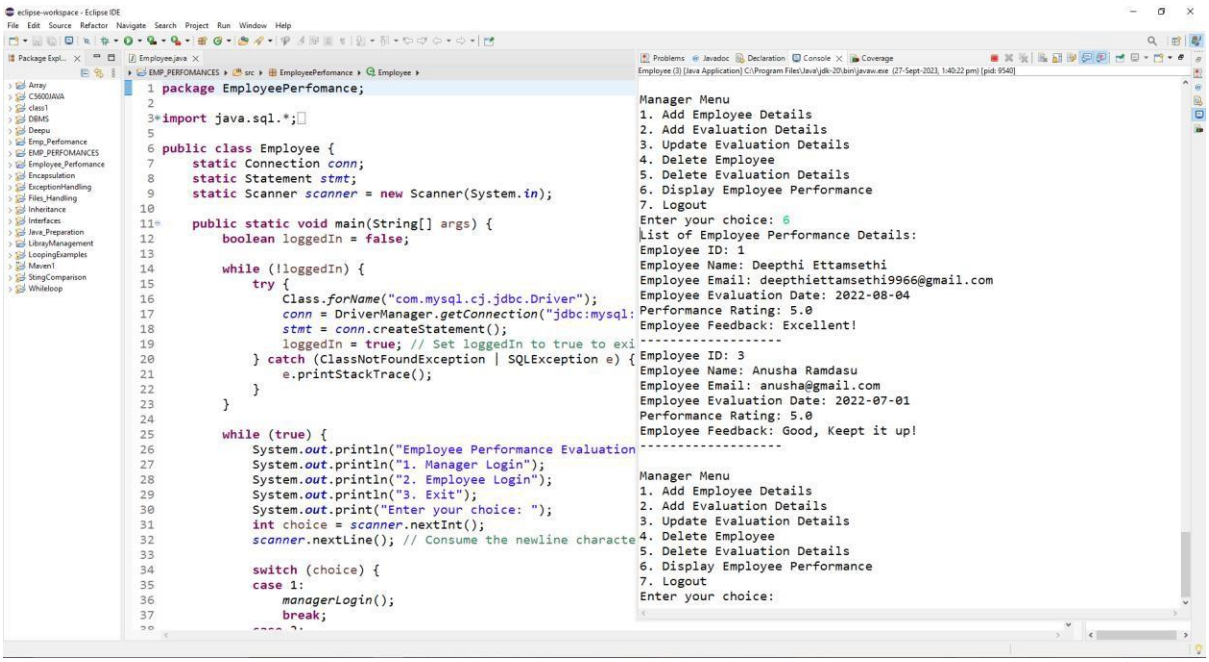
```java
package EmployeePerfomance;

import java.sql.*;

public class Employee {
    static Connection conn;
    static Statement stmt;
    static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        boolean loggedIn = false;

        while (!loggedIn) {
            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
                conn = DriverManager.getConnection("jdbc:mysql:
                stmt = conn.createStatement();
                loggedIn = true; // Set loggedIn to true to exi
            } catch (ClassNotFoundException | SQLException e) {
                e.printStackTrace();
            }
        }

        while (true) {
            System.out.println("Employee Performance Evaluation
            System.out.println("1. Manager Login");
            System.out.println("2. Employee Login");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume the newline characte

            switch (choice) {
            case 1:
                managerLogin();
                break;
```

```
Employee ID: 3
Employee Name: Anusha Ramdasu
Employee Email: anusha@gmail.com
Employee Evaluation Date: 2022-07-01
Performance Rating: 5.0
Employee Feedback: Good, Keept it up!
--------------------

Manager Menu
1. Add Employee Details
2. Add Evaluation Details
3. Update Evaluation Details
4. Delete Employee
5. Delete Evaluation Details
6. Display Employee Performance
7. Logout
Enter your choice: 5
Delete a Evaluation Details
Enter Evaluation Id to Delete: 2
Evaluation deleted successfully!

Manager Menu
1. Add Employee Details
2. Add Evaluation Details
3. Update Evaluation Details
4. Delete Employee
5. Delete Evaluation Details
6. Display Employee Performance
7. Logout
Enter your choice: 4
Delete a Employee Details
Enter Employee Id to delete: 6
Employee with Id 6 does not exist.

Manager Menu
```



```java
package EmployeePerfomance;

import java.sql.*;

public class Employee {
    static Connection conn;
    static Statement stmt;
    static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        boolean loggedIn = false;

        while (!loggedIn) {
            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
                conn = DriverManager.getConnection("jdbc:mysql:
                stmt = conn.createStatement();
                loggedIn = true; // Set loggedIn to true to exi
            } catch (ClassNotFoundException | SQLException e) {
                e.printStackTrace();
            }
        }

        while (true) {
            System.out.println("Employee Performance Evaluation
            System.out.println("1. Manager Login");
            System.out.println("2. Employee Login");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume the newline characte

            switch (choice) {
            case 1:
                managerLogin();
                break;
```

```
Manager Menu
1. Add Employee Details
2. Add Evaluation Details
3. Update Evaluation Details
4. Delete Employee
5. Delete Evaluation Details
6. Display Employee Performance
7. Logout
Enter your choice: 4
Delete a Employee Details
Enter Employee Id to delete: 6
Employee with Id 6 does not exist.

Manager Menu
1. Add Employee Details
2. Add Evaluation Details
3. Update Evaluation Details
4. Delete Employee
5. Delete Evaluation Details
6. Display Employee Performance
7. Logout
Enter your choice: 4
Delete a Employee Details
Enter Employee Id to delete: 2
Employee deleted successfully!

Manager Menu
1. Add Employee Details
2. Add Evaluation Details
3. Update Evaluation Details
4. Delete Employee
5. Delete Evaluation Details
6. Display Employee Performance
7. Logout
Enter your choice:
```

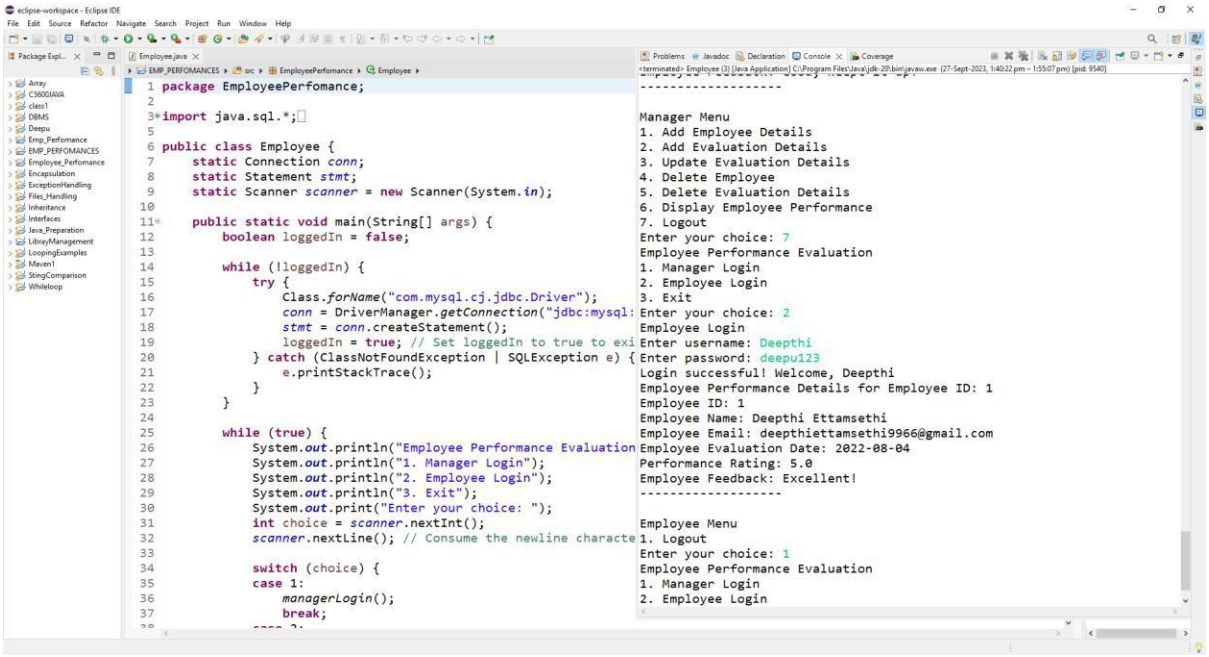


```
package EmployeePerfomance;

import java.sql.*;

public class Employee {
    static Connection conn;
    static Statement stmt;
    static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        boolean loggedIn = false;

        while (!loggedIn) {
            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
                conn = DriverManager.getConnection("jdbc:mysql:
                stmt = conn.createStatement();
                loggedIn = true; // Set loggedIn to true to exi
            } catch (ClassNotFoundException | SQLException e) {
                e.printStackTrace();
            }
        }

        while (true) {
            System.out.println("Employee Performance Evaluation
            System.out.println("1. Manager Login");
            System.out.println("2. Employee Login");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume the newline characte

            switch (choice) {
            case 1:
                managerLogin();
                break;
```

```
Manager Menu
1. Add Employee Details
2. Add Evaluation Details
3. Update Evaluation Details
4. Delete Employee
5. Delete Evaluation Details
6. Display Employee Performance
7. Logout
Enter your choice: 6
List of Employee Performance Details:
Employee ID: 1
Employee Name: Deepthi Ettamsethi
Employee Email: deepthiettamsethi9966@gmail.com
Employee Evaluation Date: 2022-08-04
Performance Rating: 5.0
Employee Feedback: Excellent!
-------------------
Employee ID: 3
Employee Name: Anusha Ramdasu
Employee Email: anusha@gmail.com
Employee Evaluation Date: 2022-07-01
Performance Rating: 5.0
Employee Feedback: Good, Keept it up!
-------------------

Manager Menu
1. Add Employee Details
2. Add Evaluation Details
3. Update Evaluation Details
4. Delete Employee
5. Delete Evaluation Details
6. Display Employee Performance
7. Logout
Enter your choice:
```

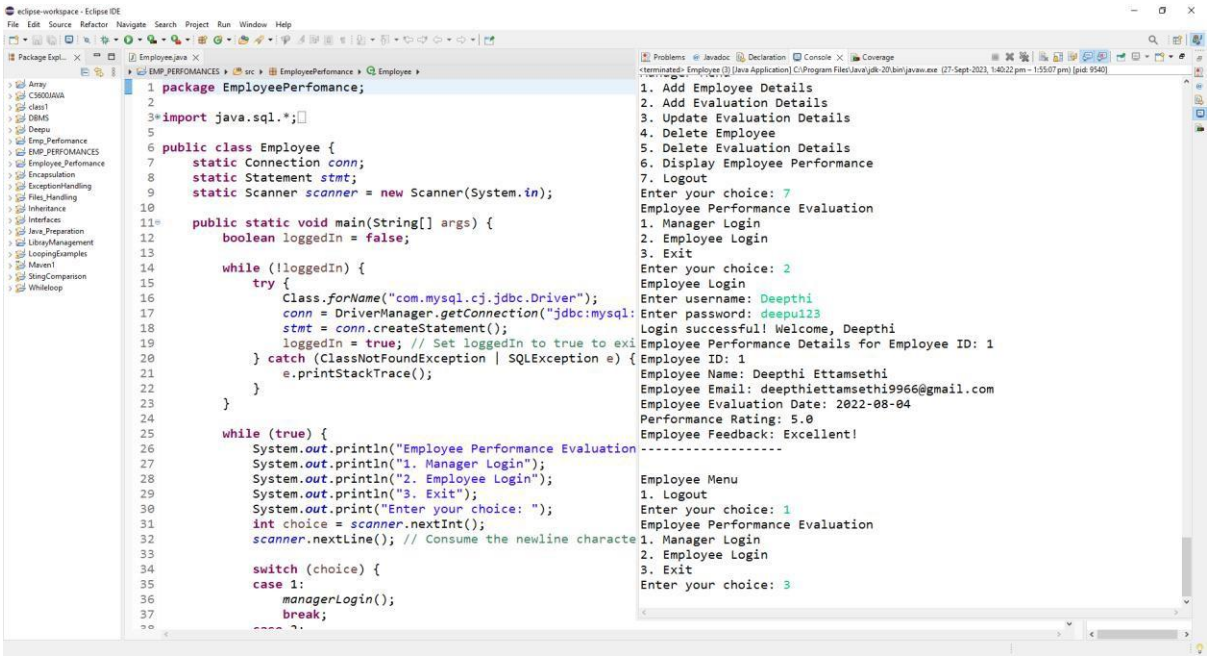
MySQL 8.0 Command Line Client

```
+----------+----------+
| Anvitha  | Anvi123  |
+----------+----------+
1 row in set (0.00 sec)

mysql> select * from employee_login;
Empty set (0.00 sec)

mysql> select * from employee_login;
+----+----------+----------+
| Id | USERNAME | PASSWORD |
+----+----------+----------+
|  1 | Deepthi  | deepu123 |
|  2 | john     | john123  |
|  3 | Anusha   | anu123   |
+----+----------+----------+
3 rows in set (0.00 sec)

mysql> select * from employee;
+----+--------------------+--------------------------------+--------+---------------------+-------------+--------+------------+
| Id | Name               | Email                          | Gender | Hiredate            | Designation | Salary | Mobileno   |
+----+--------------------+--------------------------------+--------+---------------------+-------------+--------+------------+
|  1 | Deepthi Ettamsethi | deepthiettamsethi9966@gmail.com | Female | 2021-12-14 00:00:00 | Developer   |  50000 | 8885257928 |
|  2 | john               | john@gmail.com                 | Male   | 2022-06-01 00:00:00 | Tester      |  35000 | 9874563329 |
|  3 | Anusha Ramdasu     | anusha@gmail.com               | Female | 2021-11-15 00:00:00 | Analyst     |  50000 | 9632587749 |
+----+--------------------+--------------------------------+--------+---------------------+-------------+--------+------------+
3 rows in set (0.00 sec)

mysql> select * from evaluation;
+----+----------------+--------+--------------------+
| Id | EvaluationDate | Rating | Feedback           |
+----+----------------+--------+--------------------+
|  1 | 2022-08-04     |      5 | Excellent!         |
|  2 | 2022-06-01     |      4 | Good Job!          |
|  3 | 2022-07-01     |      5 | Good, Keept it up! |
+----+----------------+--------+--------------------+
3 rows in set (0.00 sec)

mysql> select * from employee;
+----+--------------------+--------------------------------+--------+---------------------+-------------+--------+------------+
| Id | Name               | Email                          | Gender | Hiredate            | Designation | Salary | Mobileno   |
+----+--------------------+--------------------------------+--------+---------------------+-------------+--------+------------+
|  1 | Deepthi Ettamsethi | deepthiettamsethi9966@gmail.com | Female | 2021-12-14 00:00:00 | Developer   |  50000 | 8885257928 |
|  3 | Anusha Ramdasu     | anusha@gmail.com               | Female | 2021-11-15 00:00:00 | Analyst     |  50000 | 9632587749 |
+----+--------------------+--------------------------------+--------+---------------------+-------------+--------+------------+
2 rows in set (0.00 sec)

mysql> select * from evaluation;
+----+----------------+--------+--------------------+
| Id | EvaluationDate | Rating | Feedback           |
+----+----------------+--------+--------------------+
|  1 | 2022-08-04     |      5 | Excellent!         |
|  3 | 2022-07-01     |      5 | Good, Keept it up! |
+----+----------------+--------+--------------------+
2 rows in set (0.00 sec)

mysql>
```



```
package EmployeePerfomance;

import java.sql.*;

public class Employee {
    static Connection conn;
    static Statement stmt;
    static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        boolean loggedIn = false;

        while (!loggedIn) {
            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
                conn = DriverManager.getConnection("jdbc:mysql:
                stmt = conn.createStatement();
                loggedIn = true; // Set loggedIn to true to exi
            } catch (ClassNotFoundException | SQLException e) {
                e.printStackTrace();
            }
        }

        while (true) {
            System.out.println("Employee Performance Evaluation
            System.out.println("1. Manager Login");
            System.out.println("2. Employee Login");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume the newline characte

            switch (choice) {
            case 1:
                managerLogin();
                break;
```

```
--------------------

Manager Menu
1. Add Employee Details
2. Add Evaluation Details
3. Update Evaluation Details
4. Delete Employee
5. Delete Evaluation Details
6. Display Employee Performance
7. Logout
Enter your choice: 7
Employee Performance Evaluation
1. Manager Login
2. Employee Login
3. Exit
Enter your choice: 2
Employee Login
Enter username: Deepthi
Enter password: deepu123
Login successful! Welcome, Deepthi
Employee Performance Details for Employee ID: 1
Employee ID: 1
Employee Name: Deepthi Ettamsethi
Employee Email: deepthiettamsethi9966@gmail.com
Employee Evaluation Date: 2022-08-04
Performance Rating: 5.0
Employee Feedback: Excellent!
--------------------

Employee Menu
1. Logout
Enter your choice: 1
Employee Performance Evaluation
1. Manager Login
2. Employee Login
```



```
package EmployeePerfomance;

import java.sql.*;

public class Employee {
    static Connection conn;
    static Statement stmt;
    static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        boolean loggedIn = false;

        while (!loggedIn) {
            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
                conn = DriverManager.getConnection("jdbc:mysql:
                stmt = conn.createStatement();
                loggedIn = true; // Set loggedIn to true to exi
            } catch (ClassNotFoundException | SQLException e) {
                e.printStackTrace();
            }
        }

        while (true) {
            System.out.println("Employee Performance Evaluation
            System.out.println("1. Manager Login");
            System.out.println("2. Employee Login");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume the newline characte

            switch (choice) {
            case 1:
                managerLogin();
                break;
```

```
1. Add Employee Details
2. Add Evaluation Details
3. Update Evaluation Details
4. Delete Employee
5. Delete Evaluation Details
6. Display Employee Performance
7. Logout
Enter your choice: 7
Employee Performance Evaluation
1. Manager Login
2. Employee Login
3. Exit
Enter your choice: 2
Employee Login
Enter username: Deepthi
Enter password: deepu123
Login successful! Welcome, Deepthi
Employee Performance Details for Employee ID: 1
Employee ID: 1
Employee Name: Deepthi Ettamsethi
Employee Email: deepthiettamsethi9966@gmail.com
Employee Evaluation Date: 2022-08-04
Performance Rating: 5.0
Employee Feedback: Excellent!
--------------------

Employee Menu
1. Logout
Enter your choice: 1
Employee Performance Evaluation
1. Manager Login
2. Employee Login
3. Exit
Enter your choice: 3
```

11. CONCLUSION

The provided code appears to be the implementation of a basic Employee Performance Evaluation system using Java and a MySQL database.

The project allows for two types of users to log in: managers and employees. Managers can perform actions such as adding employee details, adding evaluation details, updating evaluation details, deleting employees, and deleting evaluation details. Employees, on the other hand, can log in to view their performance details.

The project utilizes a MySQL database to store information about employees, evaluations, manager logins, and employee logins. It establishes a connection to the database and performs SQL operations as needed.

The project provides a simple text-based user interface where users can choose options from a menu.

Basic error handling is implemented with exception handling to handle issues such as database connectivity problems.

Improvements:

While this project provides a foundation for managing employee performance, several areas could be enhanced and extended for a more robust and user-friendly system:

- Security Enhancements: Implement robust security measures, including password hashing, encryption, and protection against SQL injection.
- Data Validation: Implement thorough data validation to ensure that inputs are correctly formatted and within acceptable ranges.
- Logging: Add comprehensive logging capabilities to record system activities, errors, and user actions for auditing and troubleshooting.
- User Experience: Improve the user interface with better feedback, error messages, and a more intuitive menu system.
- User Roles and Permissions: Define and implement roles and permissions more comprehensively to control access to specific features.
- Reporting and Visualization: Add reporting and data visualization features to provide insights into employee performance.
- Scalability: Optimize database queries for better performance when handling a larger number of employees and evaluations.
- Testing: Implement rigorous testing, including unit tests and integration tests, to ensure the reliability and correctness of the system.
- Documentation: Document the code and system architecture thoroughly to aid developers and maintainers.

While the provided code serves as a basic Employee Performance Evaluation system, building a complete and production-ready system would involve further development, testing, and security enhancements. The project has the potential to be extended into a valuable tool for managing and analyzing employee performance data.

12. BIBLIOGRAPHY

1. JAVA2 The Complete Reference - Herbert Schildt

2. JAVA handbook - TATA InfoTech

3. Software Engineering - Richard Fairley

4. System analysis and design - Elias. M.Award

5. DataBase System - Abraham Silberschartz, Henry F.Korth and S.Sudarshan