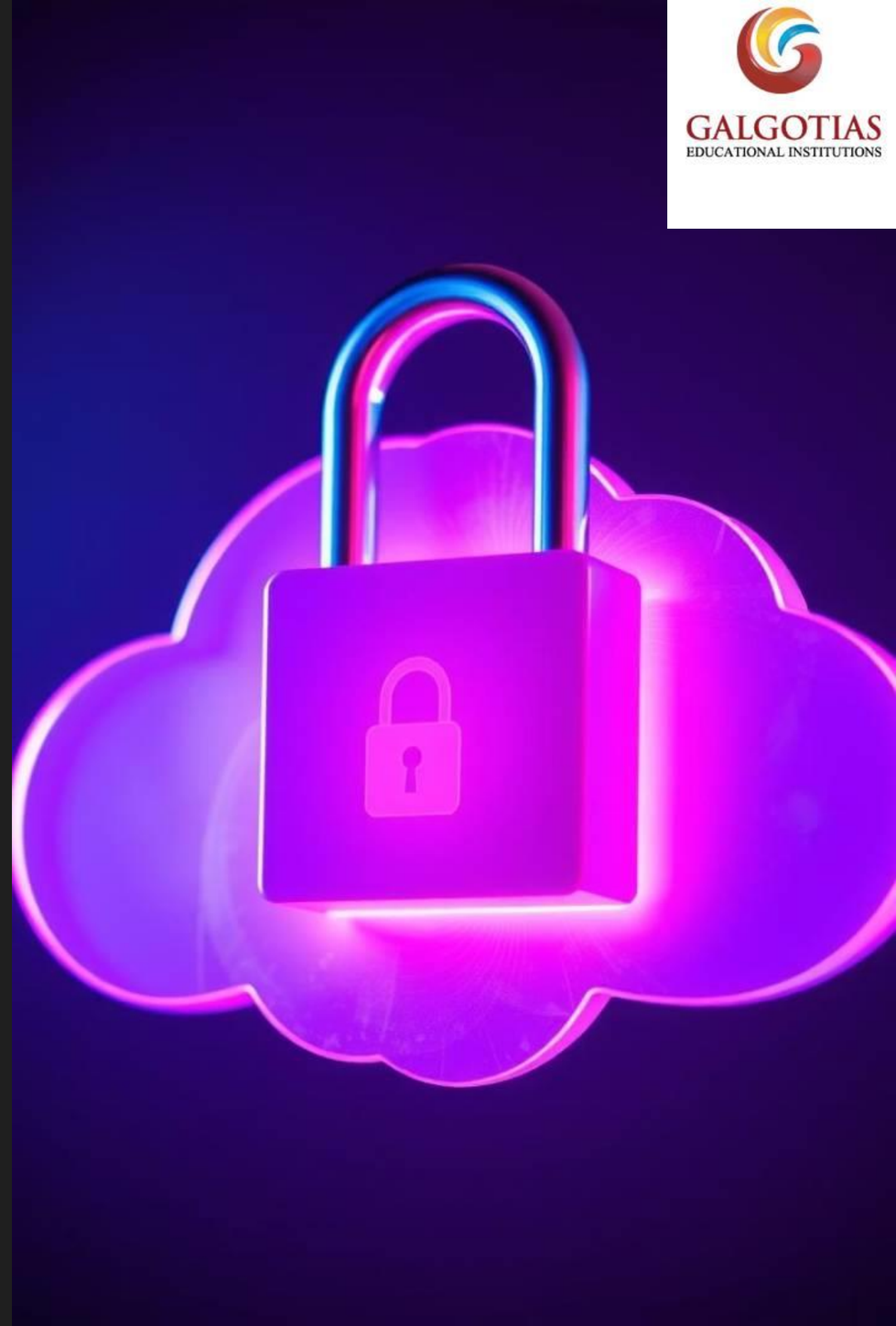
OUR TEAM

Name:-	Addmission No.
Raunak Rajput	24SCSE1011563
Mrinal Kumari Jha	24SCSE1410326
Suprova Rani Das	24SCSE1011533
Sumit Kaushal	24SCSE1011569

GALGOTIAS
EDUCATIONAL INSTITUTIONS



Digital Locker Systems: A GUI-Based Project Journey

This presentation will guide you through the process of building a digital locker system using Java file handling and a GUI. We'll cover everything from project setup to database integration and UI design principles, providing a comprehensive understanding of the development lifecycle for GUI-based projects.

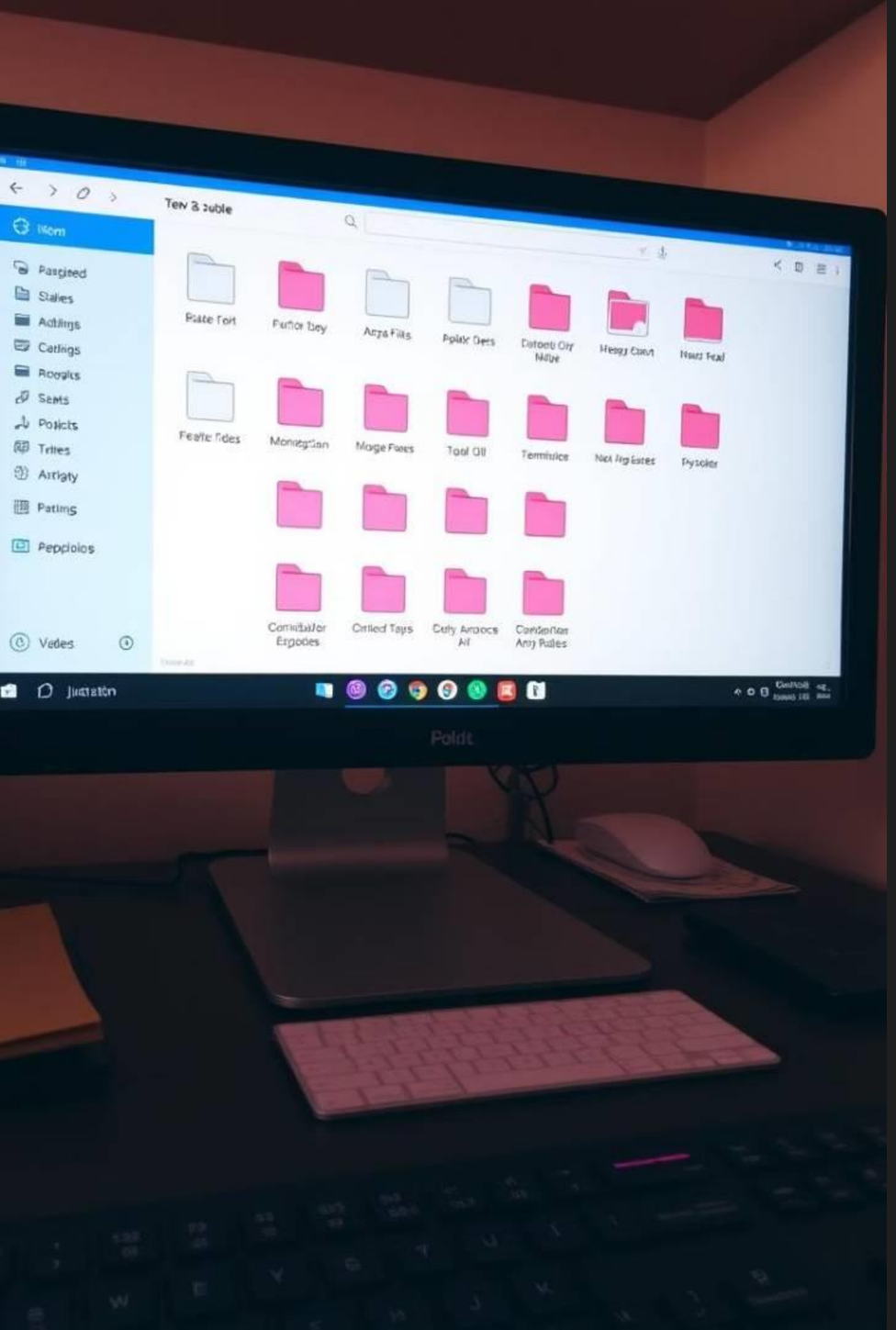
Project Initialization: Setting Up JDK and IDE

JDK Installation

The Java Development Kit (JDK) is the foundational requirement for any Java project. It includes the Java Runtime Environment (JRE), compilers, and other essential tools. Ensure you have the latest stable version installed and configured correctly on your system.

IDE Selection & Setup

An Integrated Development Environment (IDE) like IntelliJ IDEA, Eclipse, or NetBeans streamlines the development process. Choose an IDE that suits your preference and configure it with the installed JDK. This setup provides a robust environment for coding, debugging, and project management.



Defining the Digital Locker Project Structure

Root Project Directory

Establish a clear root directory for your project, typically named after the application (e.g., "DigitalLocker"). This helps organize all sub-components.

Source Code Organization

Create 'src' or 'main/java' folders for your Java source files. Sub-packages should reflect application layers (e.g., 'com.digitallocker.ui', 'com.digitallocker.model', 'com.digitallocker.dao').

Resource & Library Management

Include 'resources' for configuration files (like database properties) and 'lib' or 'dependencies' for external JAR files (e.g., JDBC drivers). Maintain a clean and logical file hierarchy.

Designing the Database Schema for Digital Lockers

User Table (Users)

This table stores user credentials. Fields might include user_id (Primary Key NOT NULL), username(VARCHAR), password_hash (VARCHAR for secure storage of hashed passwords), email (VARCHAR, Unique), and created_at (TIMESTAMP).

File Table (Files)

This table stores metadata about the files in the locker. Fields could be file_id (Primary Key), user_id (Foreign Key to Users), file_name (VARCHAR), file_path (VARCHAR, local path or cloud identifier), file_size (BIGINT), upload_date (TIMESTAMP), and encryption_status (BOOLEAN).

Permissions Table (Optional)

For more advanced features, a Permissions table could manage sharing or access rights. Fields might include permission_id (PK), file_id (FK), shared_with_user_id (FK), and access_level (VARCHAR).

Creating the MySQL Table for Digital Locker Data

After designing your schema, the next step is to physically create these tables in your MySQL database. This involves using SQL DDL (Data Definition Language) commands. For example, to create the `Users` table:

```
CREATE TABLE Users (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    email VARCHAR(100) UNIQUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Similarly, you would create the 'Files' table, ensuring proper foreign key constraints are defined to maintain data integrity between your tables.

Implementing JDBC for Database Connectivity

Driver Loading

Load the appropriate JDBC driver (e.g., Class.forName("com.mysql.cj.jdbc.Driver");) to enable your Java application to communicate with the MySQL database.

Connection Establishment

Establish a connection using DriverManager.getConnection(url, user, password);. The URL specifies the database location and name.

Statement Execution

Create ‘Statement’ or ‘PreparedStatement’ objects to execute SQL queries (INSERT, SELECT, UPDATE, DELETE). ‘PreparedStatement’ is crucial for preventing SQL injection.

Resource Closing

Always close ‘ResultSet’, ‘Statement’, and ‘Connection’ objects in a ‘finally’ block to release database resources and prevent memory leaks.

Crafting Model and DAO Classes for Database Operations

Model Classes

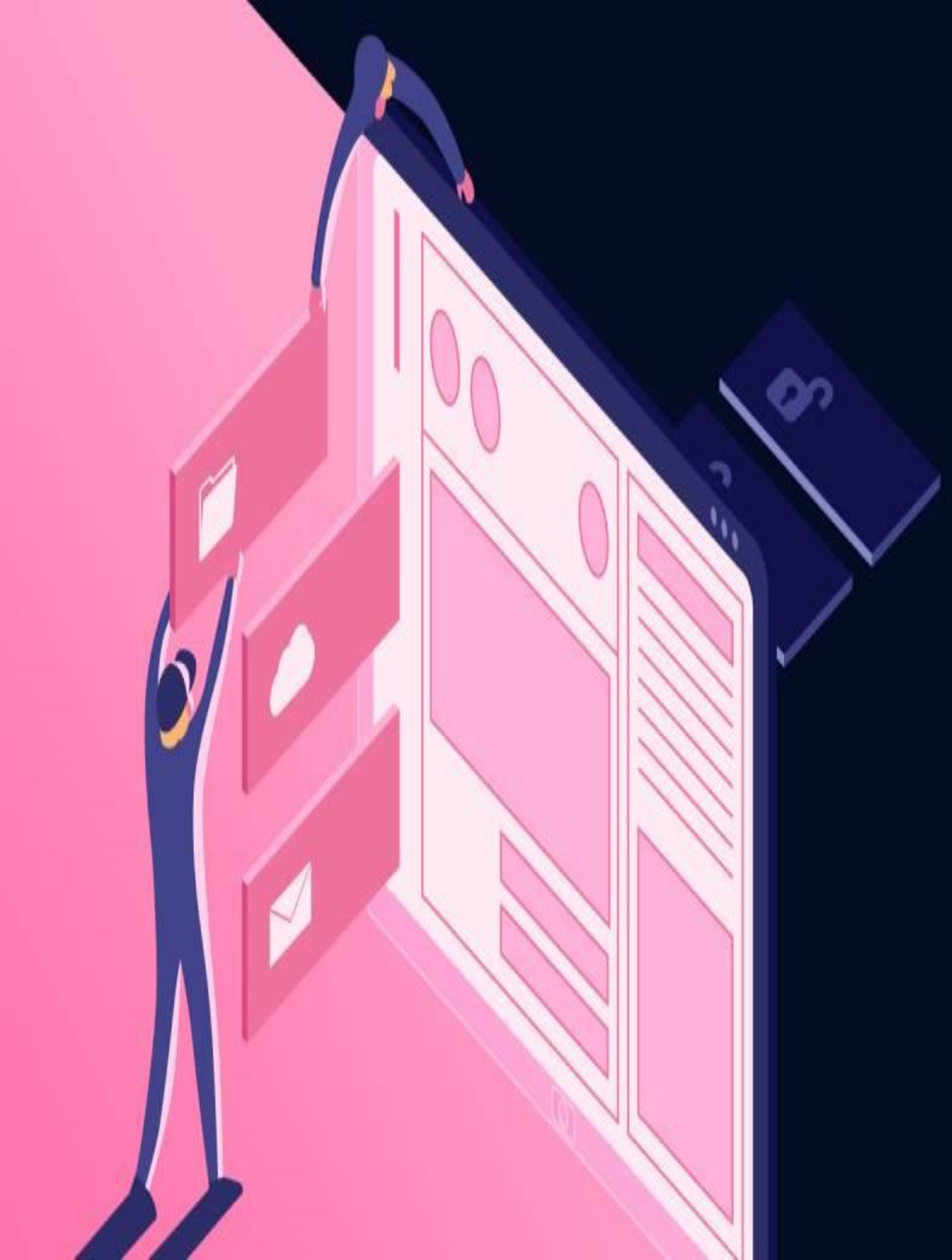
These represent the data entities (e.g., User, File). They are simple POJOs (Plain Old Java Objects) with private fields, getters, and setters that mirror your database table columns. For instance, a ‘User’ model would have ‘userId’, ‘username’, ‘passwordHash’, etc.

DAO Classes

Data Access Object (DAO) classes abstract and encapsulate all access to the data source. Each DAO class typically corresponds to a model class (e.g., UserDAO, FileDAO). They contain methods for CRUD (Create, Read, Update, Delete) operations, interacting directly with JDBC to perform database queries.

Separation of Concerns

This design pattern promotes a clear separation of concerns, making your application more modular, maintainable, and testable. The UI layer interacts only with DAO methods, never directly with JDBC.



Enhancing UI Aesthetics and Visual Appeal

Consistent Theme & Color Palette

Choose a cohesive color scheme (2-3 primary colors, plus neutrals) and stick to it across all UI components. This creates a professional and visually pleasing appearance.

Typography Selection

Select readable fonts and maintain consistent font sizes for headings, body text, and labels. Good typography improves readability and overall user experience.

Iconography & Imagery

Use high-quality, relevant icons and images to break up text and provide visual cues. Ensure they are consistent in style and size. Visuals can significantly enhance user engagement.

Optimizing UI Component Placement, Alignment, and Responsiveness

Component Placement & Alignment

Arrange UI elements logically, following common design patterns (e.g., navigation at the top/side, primary actions prominent). Use consistent spacing and alignment to create visual harmony and a clear flow. Think about the user's eye movement.

Responsiveness & Accessibility

Design your GUI to adapt to different screen sizes and resolutions. Use flexible layout managers (like `GridBagLayout` or `BorderLayout` in Swing) rather than absolute positioning. Consider accessibility by ensuring keyboard navigation, clear focus indicators, and sufficient color contrast for all users.