



CHITTAGONG UNIVERSITY OF ENGINEERING & TECHNOLOGY

Lab Report
CSE-326

Internet Programming (Sessional)
Department of Computer Science & Engineering

Report Title

Overview of Printing Management System (PrintHobe)

Section: B

Group: B2-07

Submitted By

Name	Student ID
Md. Shahriar Mahmud Turab	2104113
Tahmima Hoque Eid	2104114
Chumui Tripura	2104131

Submitted To

Md. Rashadur Rahman
Assistant Professor,
Department of CSE, CUET

Date of Submission

22 July, 2025

Introduction

In modern academic institutions, managing printing requests efficiently is essential to support the daily academic and administrative workflows. A centralized, desktop-based website dedicated to printing management can streamline these operations by automating request handling, tracking orders, and facilitating seamless communication between the printing operator and the institution's students, teachers, and staffs.

Maintaining a single channel for printing process will reduce errors, provide real-time updates, and simplify payment tracking. The system will help the printing operator manage requests systematically and offer users a reliable, user-friendly platform to submit and track printing jobs. The platform will be designed to allow future expansion with additional features to improve service quality and control.

Objectives

- To showcase a desktop-based website that centralizes printing request management.
- To streamline communication between the printing operator and users for efficient order processing.
- To implement real-time tracking of printing jobs and payment records.
- To show a user-friendly interface for submitting, monitoring, and collecting printing orders.
- To design the system with scalability in mind, allowing integration of additional features over time.

System Overview

Homepage Section

The home page of PrintHobe features a clean and user-friendly interface designed to streamline the document printing process. The header displays the website name, "PrintHobe," along with a greeting, "Hello, CSE - 21," and options to "Sign Up / Login."

The central section introduces the platform with the tagline: "A ONE STOP SOLUTION FOR PRINTING YOUR DOCUMENTS," followed by a call-to-action button, "JOIN AS AN OPERATOR." Below this, a step-by-step guide outlines the printing process:

Sign Up / Login – Users begin by creating an account or logging in.

File Upload & Customization – Users can upload their files and adjust printing preferences.

Payment – The system directs users to complete the payment securely.

Collection – Users receive their printed documents.



Figure-01: Homepage of PrintHobe

SignUp For Operators

The Printer Operator sign-up form provides a streamlined registration process for individuals looking to join as printing service providers. The form features:

First Name, Last Name, Email (for account creation), Password and Confirm Password (for secure access).

The Printer Details form is designed for administrators to configure and manage printer settings efficiently. The form includes the following fields:

Printer Name, Location, Time Per Page - Separate fields for B&W and Color printing speeds (in seconds), Cost Settings – Fields to input pricing for B&W (e.g., 2.00 TK) and Color (e.g., 5.00 TK) per page.

Bkash Number – For payment integration.

A "Create Package" section allows operators to bundle services, with a "Submit" button to save configurations.

✕

Sign Up

Start printing and earn money with every page.

👤

First Name

👤

Last Name

✉

Enter email

🔑

Password

🔒

Confirm Password

Next →

Printer Details

✕

🖨

Printer Name

📍

Location (e.g., Lab 1)

⌚

Time Per Page (B&W) in sec

⌚

Time Per Page (Color) in sec

📄

Tk. B&W per page (e.g., 2.00)

📄

Tk. Color per page (e.g., 5.00)

💳

Bkash Number

Create Package

Submit

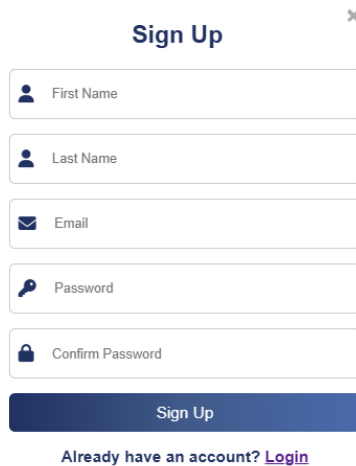
← Back

Figure-02: Operator Sign Up Form

User SignUp

The sign-up form on PrintHobe provides a streamlined and secure process for new users to create an account. The form includes-First Name, Last Name, Email, Password, Confirm Password fields.

A "Sign Up" button submits the details, while the option "Already have an account? Login" directs existing users to the login page. The minimalist design ensures ease of use while maintaining data security and user convenience



The sign-up form is titled "Sign Up" with a close button (X) in the top right corner. It contains five input fields, each with a placeholder icon and text: "First Name" (person icon), "Last Name" (person icon), "Email" (envelope icon), "Password" (key icon), and "Confirm Password" (lock icon). Below these fields is a blue "Sign Up" button. At the bottom, there is a link: "Already have an account? [Login](#)".

Figure-03: User Sign up Form

Login Form

The login form provides a secure and user-friendly interface for registered users to access their accounts. The form includes the following fields:

Email – For entering the registered email address.

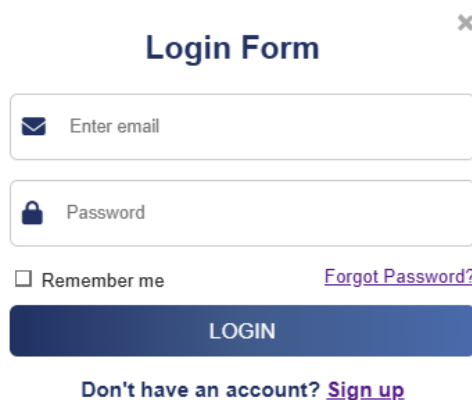
Password – For entering the account password.

Additional features include:

Remember me – An optional checkbox to save login credentials for future sessions.

LOGIN – A button to submit the credentials and authenticate the user.

A prompt, "Don't have an account? Sign up", is provided for new users to navigate to the registration page. The design is clean and intuitive, ensuring a seamless login experience while prioritizing security



The login form is titled "Login Form" with a close button (X) in the top right corner. It contains two input fields: "Enter email" (envelope icon) and "Password" (lock icon). Below these fields is a checkbox labeled "Remember me" and a link: "[Forgot Password?](#)". Below the checkbox and link is a blue "LOGIN" button. At the bottom, there is a link: "Don't have an account? [Sign up](#)".

Figure-04: Login From for all users

User Dashboard

The user interface of PrintHobe is designed for a seamless and intuitive experience, featuring:

- User Greeting: Personalized welcome message ("Hello, Tahmima Hoque") and a Logout option.
- Package Promotion: Highlights a convenience offer—"Tired of filling the Reference No. every time? Buy Now"—promoting a 45-page package for 150 TK.
- Printer Status: Displays the OS Lab Printer status ("Currently Busy") with pricing details:

Navigation Sections:

- Ongoing Printings – Active print jobs.
- Payment History – Transaction records.
- Printings History – Past print logs.
- Package Status: Alerts users if they have "No current package".

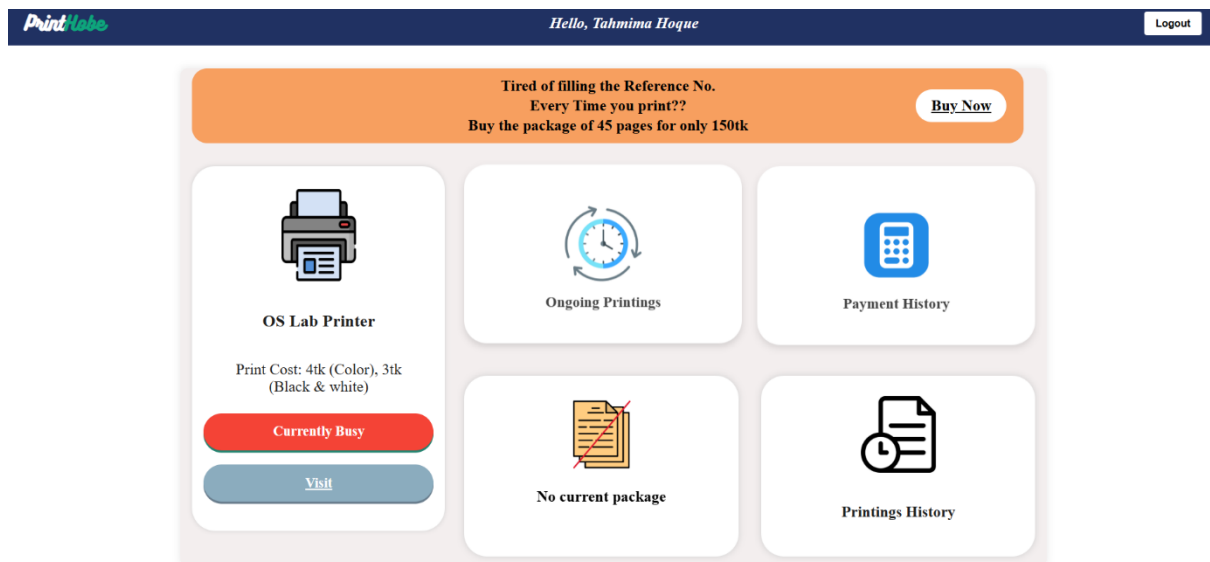


Figure-05: User Dashboard

Printing History

The Print History page provides users with a detailed and organized record of their past printing activities. Key features include:

- Date & Time – Timestamp of each print job (e.g., 11/05/2025 02:00:00).
- File Name – Name of the printed document (e.g., Lab_Report_01).
- Pages – Number of pages printed per job (e.g., 05).
- Type – Printing mode (Color or Black & White).
- Cost – Total cost per job (e.g., 20 TK).

The minimalist design ensures readability, making it easy for users to monitor their printing history and financial transactions at a glance.

History				
Date & Time	File Name	Pages	Type	Cost
11/05/2025 02:00:00	Lab_Report_01	05	Color	20
11/05/2025 11:00:00	Lab_Report_02	04	Color	16
11/05/2025 09:00:00	Lab_Report_03	04	Color	16

Figure-06: Printing History of Users

Document Submission

The document submission form on PrintHobe provides a straightforward and efficient way for users to upload and customize their printing requests. The form features a clear heading, "Drop Pay & Relax," emphasizing ease of use.


Key components include:

- File Upload: Users can drag and drop or select a PDF file for printing.


Printing Preferences: Customizable options such as:

- Pages (e.g., number of pages)
- Copies (number of duplicates)
- Sides (single or double-sided printing)
- Color (color or black-and-white)
- Punching (optional binding holes)
- Cost & Time Estimation: Displays the Amount and Estimated Time for transparency.
- Payment: Users can apply their package benefits or proceed with "Pay Now."

Drop Pay & Relax



Drop Your File here (.pdf)



Choose Your Preferences

Pages:

1

Copies:

1

Sides:

Single Sided

▼

Color:

Color

▼

Punching:

No

▼

Amount:

0

Estimated Time:

Use Your Package

Pay Now

Figure-07: Document submission form with details

Operator Dashboard

The operator interface of PrintHobe is designed to efficiently manage and monitor printing tasks. A Printing Schedule section displaying the current printing deadline (e.g., "04:30 PM") with an "Update now" option for adjustments.

The core of the interface is a structured table listing active printing jobs, organized by:

- Reference No. – Unique identifiers for tracking.
- Document's Name – File names (e.g., DataFlowDiagramFinal.pdf).
- Color – Printing mode (COLOR/B&W).
- No. Of Copies – Quantity requested.
- Status – Current state of each job (e.g., "Completed").

Each entry includes actionable buttons (Print or Reject) for operators to process jobs promptly.

The Printing Packages page provides a structured overview of user printing plans and transactions. Key elements include:

- Reference ID – A unique identifier (e.g., adfer223XXWe) for tracking each package payment.
- Pages – Displays the allocated or used page quota.
- Amount – Shows the total cost associated with the package (e.g., 150 TK).

- Status – Indicates the current state of the package (Accept, Reject, etc.).

The minimalist table layout ensures clarity, allowing users to quickly review their printing packages, usage, and financial details. This page enhances transparency and simplifies package management for both users and administrators.

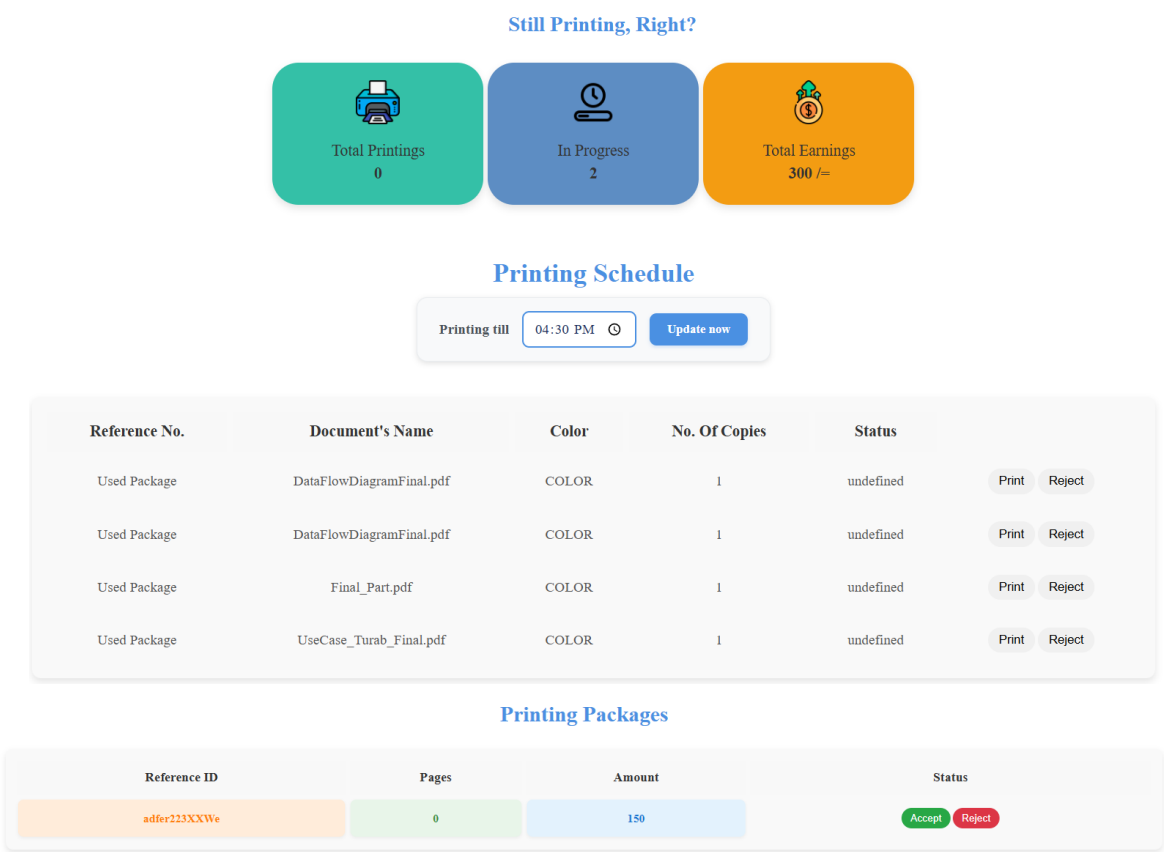


Figure-08: Operator Dashboard with printing documents and package list

ER Diagram (Entity-Relationship Diagram)

An ER Diagram is a visual representation of the database structure. It shows how entities (such as tables) relate to each other within the system. The main components of an ER diagram include:

- **Entities:** Represent objects or things in the system, usually depicted as rectangles.
- **Attributes:** Characteristics or properties of entities, shown as ovals connected to their entities.
- **Relationships:** Connections between entities that define how they interact, illustrated by diamonds or lines.

ER diagrams help in designing databases by organizing data clearly, showing the structure, and highlighting the relationships among data elements. They are essential for planning the database schema and ensuring data integrity.

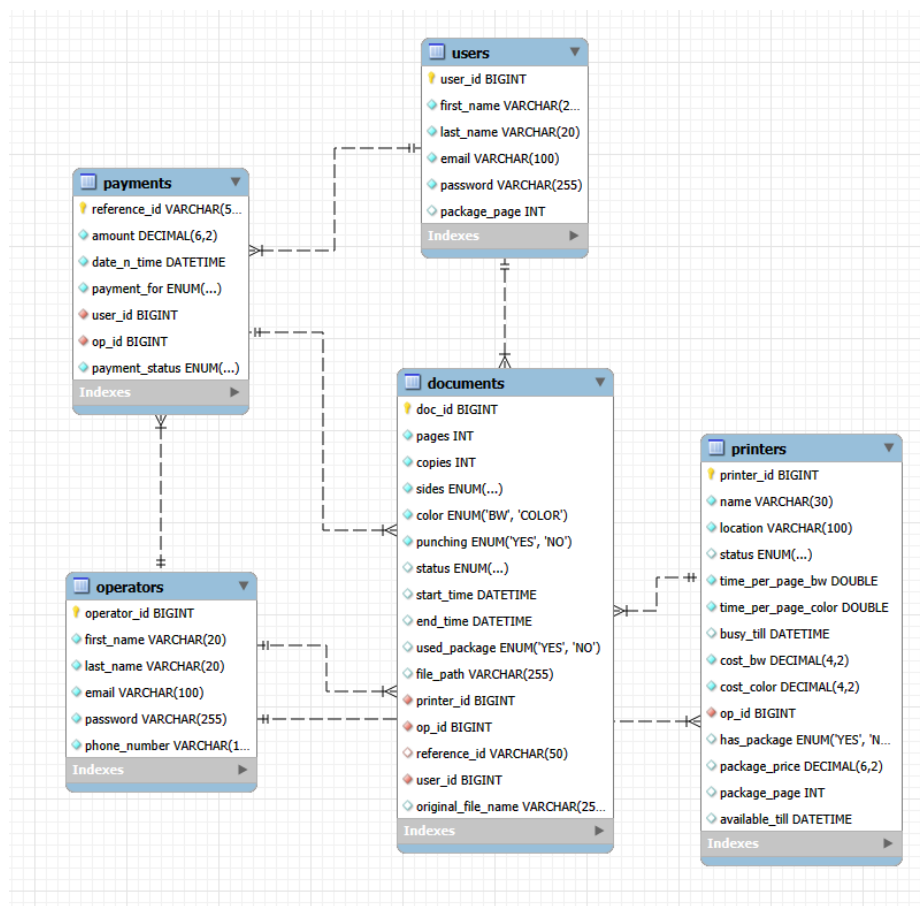


Figure-09: ER diagram of Printing Management System

Database

In the database, we have 5 tables. The tables are-

1. User
2. Payment
3. Document
4. Operator
5. Printer

User Table

User table contains 6 fields. They are-

- User_Id: To identify each user uniquely.
- First_Name: Stores the first name of the user.
- Last_Name: Stores the last name of the user.
- Email: Stores the email of the user.
- Password: Stores the password of the user
- Package_Page: Keeps track of the package page available for the user.

	Field	Type	Null	Key	Default	Extra
▶	user_id	bigint	NO	PRI	<small>HULL</small>	auto_increment
	first_name	varchar(20)	NO		<small>HULL</small>	
	last_name	varchar(20)	NO		<small>HULL</small>	
	email	varchar(100)	NO	UNI	<small>HULL</small>	
	password	varchar(255)	NO		<small>HULL</small>	
	package_page	int	YES		0	

Figure-10: User Table

Payment Table

Payment table has 6 fields. They are-

- Reference_id: To identify each payment and verify payments.
- Amount: To store the amount of money paid by the users.
- Date_N_Time: To store the date and time of the payment.
- Payment_For: Stores why the payment was made. It can be for either printing of purchasing package.
- User_Id: To help identify the user who made the payment.
- Operator_Id: To help identify the operator will receive the payment.
- Payment_Status: Helps to know the status of the payment.

	Field	Type	Null	Key	Default	Extra
▶	reference_id	varchar(50)	NO	PRI	NULL	
	amount	decimal(6,2)	NO		NULL	
	date_n_time	datetime	NO		NULL	
	payment_for	enum('PACKAGE','PRINTING')	NO		NULL	
	user_id	bigint	NO	MUL	NULL	
	op_id	bigint	NO	MUL	NULL	
	payment_status	enum('APPROVED','REJECTED','VERI...)	NO		VERIFYING	

Figure-11: Payment Table

Document Table

The documents table has 16 fields. They are-

- Doc_Id: To identify each document uniquely.
- Pages: Stores the number of pages of the document.
- Copies: Stores how many copies of the document will be printed.
- Sides: Stores whether the document will be printed on single-sided or on both-sided.
- Color: Stores whether the document will be printed with color or in black and white.
- Punching: Whether punching is required after the document is printed.
- Status: Whether the document is printed or not.
- Start_Time: Stores when the document is started for printing.
- End_Time: Stores when the document is finished printing.
- Used_Package: Stores whether the document was given for print using package.
- File_Path: Stores the file path where the document was stored on the pc.
- Printer_Id: Stores where the document will be printed.
- Op_Id: Stores the operator id who will print the document.
- Reference_Id: Stores the reference_id of the payment made for the document.
- User_Id: Stores the user_id who submitted the document.
- Original_File_Name: Stores the file name of the document.

	Field	Type	Null	Key	Default	Extra
►	doc_id	bigint	NO	PRI	NULL	auto_increment
	pages	int	NO		NULL	
	copies	int	NO		NULL	
	sides	enum('SINGLE_SIDED','DOUBLE_SID...	NO		NULL	
	color	enum('BW','COLOR')	NO		NULL	
	punching	enum('YES','NO')	NO		NULL	
	status	enum('COMPLETED','REJECTED','AP...	YES		VERIFYING	
	start_time	datetime	YES		NULL	
	end_time	datetime	YES		NULL	
	used_package	enum('YES','NO')	YES		NO	
	file_path	varchar(255)	YES		NULL	
	printer_id	bigint	NO	MUL	NULL	
	op_id	bigint	NO	MUL	NULL	
	reference_id	varchar(50)	YES	MUL	NULL	
	user_id	bigint	NO	MUL	NULL	
	original_file_n...	varchar(255)	YES		NULL	

Figure-12: Document Table

Operator Table

Operator table has 6 fields. They are-

- Operator_Id: To identify each operator uniquely.
- First_Name: Stores the first name of the operator.
- Last_Name: Stores the last name of the operator.
- Email: Stores the email of the operator.
- Password: Stores the password of the operator.
- Phone_Number: Stores the phone number of the operator that will be used for sending money to the operator.

	Field	Type	Null	Key	Default	Extra
►	operator_id	bigint	NO	PRI	NULL	auto_increment
	first_name	varchar(20)	NO		NULL	
	last_name	varchar(20)	NO		NULL	
	email	varchar(100)	NO	UNI	NULL	
	password	varchar(255)	NO		NULL	
	phone_number	varchar(14)	NO		NULL	

Figure-13: Operator Table

Printers Table

Printer table has 14 fields. They are-

- Printer_Id: To identify each printer uniquely.
- Name: Stores the name of the printer.
- Location: Stores where the printer is located.
- Status: Keeps track of the printer whether the printer is available or not.

- Time_Per_Page_BW: Stores the time required for printing document in black and white.
- Time_Per_Page_Color: Stores the time required for printing document in color.
- Busy_Till: Stores the time until when the printer is busy.
- Cost_BW: Stores the cost of printing the document in black and white.
- Cost_Color: Stores the cost of printing the document in color.
- Op_Id: Stores the operator id of the printer to keep track of owner of the printer.
- Has_Package: Stores whether there are any packages available for the printer.
- Package_Price: Stores the price of the package if there are any.
- Package_Page: Stores the number of pages in the package.
- Available_Till: Keeps track of the time until when the printer is available.

Field	Type	Null	Key	Default	Extra
printer_id	bigint	NO	PRI	<small>NULL</small>	auto_increment
name	varchar(30)	NO		<small>NULL</small>	
location	varchar(100)	NO		<small>NULL</small>	
status	enum('AVAILABLE','NOT_AVAILABLE')	YES		NOT_AVAILABLE	
time_per_page_bw	double	NO		<small>NULL</small>	
time_per_page_color	double	NO		<small>NULL</small>	
busy_till	datetime	YES		<small>NULL</small>	
cost_bw	decimal(4,2)	NO		<small>NULL</small>	
cost_color	decimal(4,2)	NO		<small>NULL</small>	
op_id	bigint	NO	MUL	<small>NULL</small>	
has_package	enum('YES','NO')	YES		NO	
package_price	decimal(6,2)	YES		<small>NULL</small>	
package_page	int	YES		<small>NULL</small>	
available_till	datetime	YES		<small>NULL</small>	

Figure-14: Printer Table

Backend

The backend of an application refers to the server-side part that handles the business logic, database interactions, authentication, and more. It is responsible for processing requests from the front end (user interface) and returning the appropriate responses.

In the context of databases:

- Database Management: The backend communicates with a database (like MySQL, MongoDB, etc.) to store, retrieve, and update data.
- Server-Side Logic: It processes user input, runs algorithms, and performs operations like calculations, file handling, or content management.
- APIs: The backend often exposes APIs (Application Programming Interfaces) to allow the frontend to interact with the server.

For the backend of this project, we have used spring-boot, which is a java-based framework.



The screenshot shows the Spring Initializr web form. On the left, under 'Project', 'Maven' is selected. Under 'Language', 'Java' is selected. Under 'Spring Boot', '3.5.0' is selected. The 'Project Metadata' section includes fields for Group (com.PrintHobe), Artifact (PrintingManagement), Name (PrintingManagement), Description (Demo project for Spring Boot), Package name (com.PrintHobe.PrintingManagement), Packaging (.Jar), and Java version (24). On the right, the 'Dependencies' section lists 'Spring Web' (WEB), 'Spring Data JPA' (SQL), 'MySQL Driver' (SQL), and 'Spring Boot DevTools' (DEVELOPER TOOLS). At the bottom, there are buttons for 'GENERATE' (CTRL + G), 'EXPLORE' (CTRL + SPACE), and a menu button (three dots).

Figure-15: Creation of backend and dependencies

Application Properties Configuration in Spring Boot

The application.properties file in Spring Boot configures various settings for the application:

- **spring.application.name=PrintingManagement**: Sets the application name as "PrintingManagement".
- **spring.datasource.url=jdbc:mysql://localhost:3306/PrintManagement**: Specifies the connection URL for the MySQL database.
- **spring.datasource.username=root** and **spring.datasource.password=root**: Defines the database username and password for authentication.
- **spring.jpa.hibernate.ddl-auto=validate**: Validates the database schema against the entity model without making changes.
- **spring.jpa.show-sql=true**: Logs SQL queries generated by Hibernate for debugging.
- **spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect**: Uses the MySQL 8 dialect for SQL compatibility with Hibernate.
- **server.port=8080**: Specifies the port for the application to run (default is 8080).
- **spring.servlet.multipart.max-file-size=10MB** and **spring.servlet.multipart.max-request-size=10MB**: Limits the maximum file size and request size for file uploads to 10MB.

These properties configure essential aspects of the backend, such as database connection, file upload limits, and application behavior.

```
spring.application.name=PrintingManagement

# My DataBase Connection
spring.datasource.url=jdbc:mysql://localhost:3306/PrintManagement
spring.datasource.username=root
spring.datasource.password=root

#JPA and Hibernate Settings
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

#Server Port
server.port=8080

#Max File Size
spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=10MB
```

Figure-16: Application properties configuration and database connection

Entity in Spring Boot

In Spring Boot, an Entity represents a table in the database. It is a Java class annotated with `@Entity` and used by JPA (Java Persistence API) to map the object to a relational database table. Each field in the Entity class corresponds to a column in the database table.

Key points about Entities in Spring Boot:

- `@Entity`: The main annotation to mark a class as an Entity.
- `@Id`: Specifies the primary key of the entity.
- `@GeneratedValue`: Indicates how the primary key is generated (e.g., auto-increment).
- `@Column`: Customizes the mapping of a field to a database column.
- Relationships: Entities can be related to each other using annotations like `@OneToMany`, `@ManyToOne`, `@ManyToMany`, etc.

Entities are fundamental in Spring Boot applications that use JPA to interact with the database. They help in persisting data, performing CRUD operations, and ensuring that the Java object model matches the database schema.

As, in the database, we have 5 tables, we have 5 entities in the backend. They are-

1. User

2. Payment
3. Document
4. Operator
5. Printer

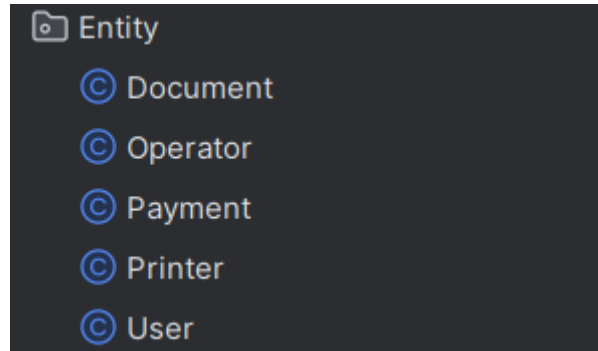


Figure-17: List of entities in the backend

Repositories in Spring Boot

In Spring Boot, a Repository is responsible for handling data access operations between the application and the database. Repositories provide methods to perform CRUD (Create, Read, Update, Delete) operations on Entities without requiring boilerplate SQL code.

Key points about Repositories:

- `@Repository`: This annotation marks an interface or class as a Spring Data repository, allowing it to be automatically implemented by Spring.
- `JpaRepository` or `CrudRepository`: These are common interfaces that extend `Repository` and provide a set of built-in methods to interact with the database (e.g., `save()`, `findById()`, `delete()`, etc.).
- Custom Queries: You can define custom methods using JPQL (Java Persistence Query Language) or native SQL with the `@Query` annotation.

Repositories simplify database interactions by abstracting complex queries and reducing boilerplate code, making development faster and more efficient.

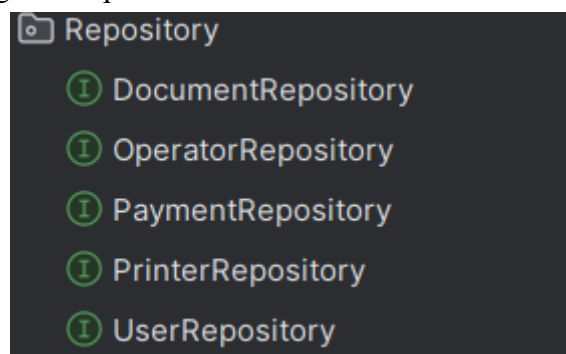


Figure-18: List of repositories in the backend

We have five repositories in the backend, each one related to each entity.

Service in Spring Boot

In Spring Boot, a Service is a class that contains business logic and acts as an intermediary between the Controller and the Repository layers. It is typically annotated with `@Service` and is responsible for processing data, performing complex operations, and managing transactions.

Key points about Services:

- `@Service`: Marks a class as a service, making it a Spring-managed bean that can be injected into other components like controllers or repositories.
- Business Logic: Services are where you implement the core business logic of the application (e.g., data validation, calculations).
- Transaction Management: Services handle transactional operations, ensuring that database changes are committed or rolled back correctly.
- Dependency Injection: Services are injected into controllers to keep the controller layer clean and focused on HTTP request handling, not business logic.

In summary, a service acts as the middle layer in a Spring Boot application, isolating business logic from the web layer and providing an abstraction over data access.

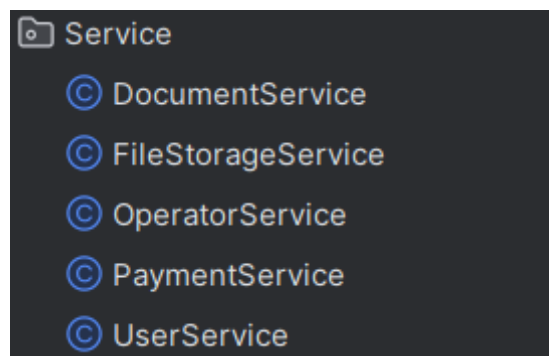


Figure-19: List of services in the backend

In our backend, we have five services corresponding to the entities. These service stores the business logic for the entities.

Controllers in Spring Boot

In Spring Boot, a Controller is a class that handles HTTP requests and maps them to appropriate service methods. It acts as the interface between the frontend (client) and backend (application logic). Controllers are typically annotated with `@RestController` or `@Controller`.

Key points about Controllers:

- `@RestController`: A combination of `@Controller` and `@ResponseBody`, used for RESTful web services where data is returned directly to the client (often in JSON or XML format).

- **@RequestMapping:** Used to map HTTP requests to handler methods of the controller. You can specify request types like GET, POST, PUT, DELETE using annotations like `@GetMapping`, `@PostMapping`, etc.
- **Service Layer Integration:** Controllers call the Service layer to handle business logic, ensuring separation of concerns between request handling and business processing.
- **Input Handling:** Controllers handle input parameters (path variables, request bodies, query parameters) using annotations like `@RequestParam`, `@PathVariable`, and `@RequestBody`.

Controllers are key in processing HTTP requests, directing them to the right service, and returning the appropriate response to the client.

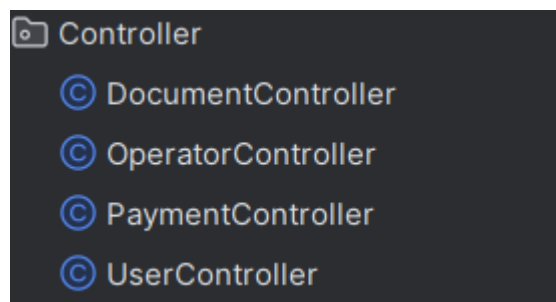


Figure-20: List of controllers in the backend

We have four controllers in the backend. They are responsible for the request handling from the front-end.

DTO (Data Transfer Object) in Spring Boot

In Spring Boot, a DTO (Data Transfer Object) is an object that is used to transfer data between different layers of an application, particularly between the Controller and the Service or Client. DTOs are primarily used to encapsulate data in a way that reduces the number of method calls, especially when handling complex objects or large amounts of data.

Key points about DTOs:

- **Purpose:** DTOs are used to carry data between processes. They are often used to return specific fields or simplified representations of entities.
- **Decoupling:** By using DTOs, you decouple the internal database model (Entity) from the client-facing API, allowing flexibility in how data is presented without exposing internal structures.
- **Validation:** DTOs can include validation annotations (e.g., `@NotNull`, `@Size`) to ensure data integrity before it's passed to the backend.
- **Efficiency:** DTOs can optimize performance by only including the fields that are needed for a specific operation, reducing unnecessary data transfer.

For example, when a user requests only their name and email, you can return a UserDTO with just those fields, instead of returning the entire UserEntity with all columns.

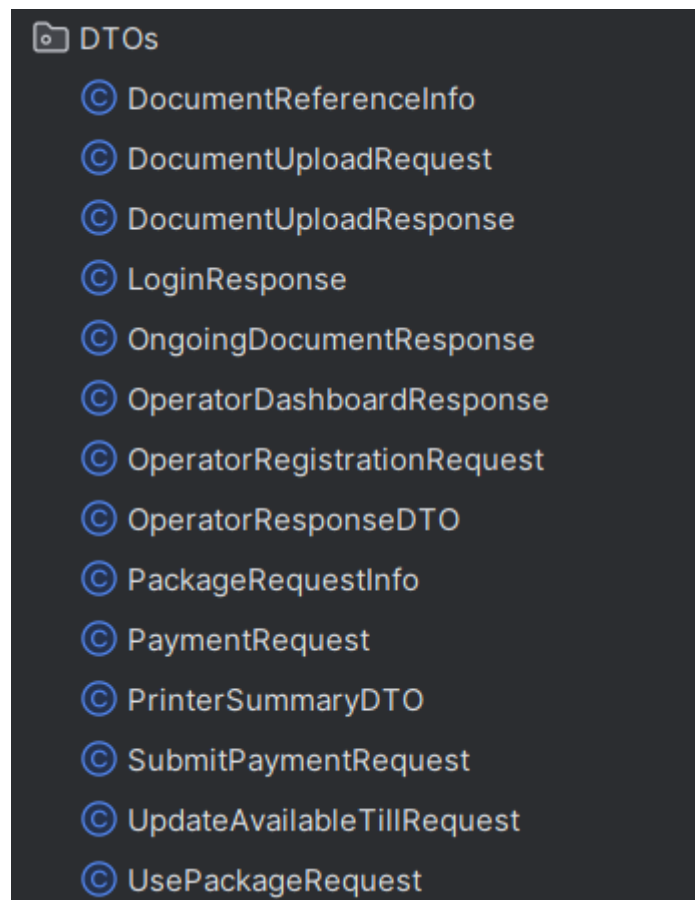


Figure-21: List of DTOs in the backend

We have 14 DTOs in the backend. Each one is responsible for each specific task.

Conclusion

In conclusion, the Printing Management System developed for academic institutions provides a streamlined, centralized platform that efficiently handles printing requests and order processing. By integrating both front-end and back-end components, the system offers a user-friendly interface for students, faculty, and staff to submit, track, and manage printing jobs in real-time.

The front-end web pages are designed to be intuitive and accessible, enabling users to easily place orders, monitor their progress, and track payments. The back-end integrates a database system to ensure seamless communication between the printing operator and users, while maintaining an organized structure for job orders, payment tracking, and reporting.

This centralized approach not only reduces errors and enhances workflow efficiency, but also provides real-time updates, enabling faster and more accurate printing job management.

Additionally, the design and structure of the system ensure scalability, allowing future features to be added easily as the needs of the institution evolve.

Overall, this system achieves the objectives of simplifying and automating printing management while maintaining a focus on ease of use, efficiency, and future growth. It serves as a foundation for enhancing the quality of service in academic and administrative settings, ensuring long-term reliability and functionality.