Software Requirements Specification for Student Smart Printing Service

Version 5.5 approved

Prepared by:

1. **Nguyen Quang Phu - 2252621**
2. **Nguyen Ngoc Khoi - 2252378**
3. **Nguyen Nhat Khoi - 2252379**
4. **Nguyen Minh Khoi - 2252376**
5. **Nguyen Quang Vinh - 2213973**

Department of Software Engineering

Faculty of Computer Science and Engineering

Ho Chi Minh City University of Technology – VNU-HCM

12/12/2024

Table of Contents

I. Requirement elicitation **9**

1. Overview (Task 1.1) **9**

1.1. Domain Context ***9***

1.2. Stakeholders and Needs ***9***

1.3. Benefits of the System ***11***

2. Functional and Non-Functional Requirements (Task 1.2) **12**

2.1. Functional Requirements ***12***

2.1.1. For Students: *12*

2.1.2. For SPSO: *13*

2.1.3. For University Administration: *14*

2.1.4. For HCMUT_SSO Authentication Service: *15*

2.1.5. For BKPay Payment System Provider : *15*

2.2. Non-Functional Requirements ***16***

2.2.1. Security *16*

2.2.2. Reliability *16*

2.2.3. Usability *16*

2.2.4. Compatibility *16*

2.2.5. Scalability *17*

3. Use-case Diagrams and Scenarios (Task 1.3) **18**

3.1. Use-case table ***18***

3.2. Use-case Diagram for the Whole System ***19***

3.3. Module: Authentication ***20***

3.3.1. The Use Case Diagram in Authentication Module *20*

3.3.2. The Use Case Scenario: Authentication/Login *20*

3.4. Module: Printing ***23***

3.4.1. The Use Case Diagram in Printing Module *23*

3.4.2. The Use Case Scenario: Printing Documents *24*

3.4.3. The Use Case Scenario: Buy printing pages *26*

3.4.4. The Use Case Scenario: Document upload *28*

3.4.5. The Use Case Scenario: Customize printing options *30*

3.4.6. The Use Case Scenario: Track print status *32*

3.5. Module: Printer Management ***34***

3.5.1. The Use Case Diagram in Printer Management Module..... 34
3.5.2. The Use Case Scenario: Manage printers..... 35
3.5.3. The Use Case Scenario: Add Printer..... 36
3.5.4. The Use Case Scenario: Enable printer.....37
3.5.5. The Use Case Scenario: Disable printer.....38
II. System Modeling..... 40
1. Activity Diagrams (Task 2.1).....40
1.1. Module: Authentication..... 40
1.2. Module: Printing..... 41
1.3. Submodule: Buying pages.....43
1.4. Module: Printer Management..... 44
2. Sequence Diagram (Task 2.2).....45
2.1. Module: Authentication..... 45
2.2. Module: Printing..... 46
2.2.1. Submodule: Buying Printing Pages..... 47
2.2.2. Submodule: Filter Printers.....48
2.2.3. Submodule: Upload Documents.....49
2.3. Module: Printer Management..... 50
2.3.1. Submodule: Enable and Disable Printers.....51
2.3.2. Submodule: Add Printers..... 52
3. Class Diagram (Task 2.3).....54
3.1. Authentication Module.....55
3.2. Printing Module..... 57
3.3. Printer Management Module.....61
4. User Interface (Task 2.4).....63
4.1. Log in..... 63
4.2. Printing..... 67
4.3. Printer Management..... 71
III. Architecture Design (Task 3)..... 73
1. MVC Architecture.....73
1.1. Advantages and Disadvantages of MVC..... 74
1.2. Reasons for Choosing MVC over Layered Architecture..... 75
2. Architectural Diagram and Deployment Diagram.....77

2.1. Architectural Diagram for the overall design of HCMUT-SSPS system..... 77

2.2. Deployment Diagram for the overall design of HCMUT-SSPS system.....84

2.2.1. Deployment Diagram..... 84

2.2.2. Description for the Deployment diagram.....85

3. Component Diagram.....90

3.1. Module: Authentication..... 90

3.2. Module: Printing..... 91

3.3. Module: Printer Management..... 92

3.4. Interactions between specific components in the Models, Controller, View..... 93

IV. Implementation - Sprint 1 (Task 4).....96

1. Setting up an online repository for version control.....96

2. Adding documents, materials and folders for Requirement, System modeling and Architectural design.....98

3. Conducted a usability test with the user interface you developed in MVP 1..... 106

3.1. Test the Usability of MVP 1.....106

3.1.1. Homepage Interface..... 107

3.1.2. Login Interface..... 108

3.1.3. Student's Dashboard Interface..... 109

3.1.4. SPSO's Dashboard Interface..... 110

3.1.5. Choosing Printer Interface..... 111

3.1.6. Uploading Files Interface.....112

3.1.7. Printing Configuration Interface.....113

3.1.8. Buying Printing Pages.....114

3.1.9. Printer Management Interface..... 115

3.2. Solution for those remaining problems of MVP 1..... 116

3.2.1. Summary of the criterias and other things that we must improve based on user feedbacks.....116

3.2.2. Our suggested solutions.....117

V. Implementation - Sprint 2 (Task 5)..... 124

1. Using Frameworks..... 124

1.1. Front-end..... 124

1.2. Back-end..... 126

2. Demo Application.....129

2.1. Our Application ***129***
2.2. Login Page ***131***
2.3. Homepage View of Student vs SPSO ***133***
2.4. Profile Page of a User (Student and SPSO) ***135***
2.5. Printing Actions from Student ***136***
2.5.1. List of Printers *136*
2.5.2. Buying Pages *137*
2.5.3. Upload Files *139*
2.5.4. Configure Files for Printing *142*
2.5.5. View Files that is scheduled for Printing *145*
2.6. Printing Management from SPSO ***148***
2.6.1. View Current List of Printers *148*
2.6.2. Enable the Printer *149*
2.6.3. Add a new Printer *150*
2.6.4. Delete a Printer *152*
2.6.5. Printing Configuration: Configure the File Type *153*
2.6.6. Printing Configuration: Giving Pages *156*
2.6.7. Printing Configuration: Export Report *159*
2.7. Extension 1: SPSO can View User List and Specific SPSO can edit the Role ***163***
2.8. Extension 2: Rush hour Solution ***164***
2.9. Extension 3: Provide the UI for Loading and 404 - Error Page ***166***
2.10. Summary of Things that we have done and still not done ***166***
3. Presentation **167**
VI. Lessons and Conclusion **168**
VII. References **170**

Revision History

Name	Date	Reason For Changes	Version
Draft Plan.	19/09/2024	Draft Version.	1.0
Update Domain Context.	20/09/2024	Update Domain Context.	1.1
Update 1..2 & 1.3.	21/09/2024	Improve 1.2 & 1.3 by reconsidering stakeholders.	1.2
Update 1.4 & 1.5.	22/09/2024	Improve 1.4 & 1.5 by adding more requirements.	1.3
Update Use case.	25/09/2024	Update use case diagrams	1.4
Further Elaboration on use case.	27/09/2024	Improvement of the use case diagrams.	1.5
Beautify use case diagrams.	30/09/2024	Improve on the visual of diagrams	1.6
Update Activity Diagrams (2.1).	04/10/2024	Refinement of activity diagrams for clarity and alignment with the latest functional requirements.	2.0
Sequence Diagram Improvement (2.2).	08/10/2024	Enhance sequence diagrams for better representation of Printer Management interactions	2.1
Class Diagram Update (2.3).	13/10/2024	Update class diagrams with new attributes and methods	2.2
UI Diagram Adjustment.	16/10/2024	Minor UI updates to reflect changes in workflow	2.3
Complete System Modeling Report.	20/10/2024	Final modification of System Modeling part in this report	2.4
Update MVC	23/10/2024	Added advantages, disadvantages,	3.0

Architecture Section.		and reasons for MVC choice over layered architecture	
Architectural and Deployment Diagrams Refinement	26/10/2024	Refined architectural and deployment diagrams for alignment with system components	3.1
Component Diagram Update	27/10/2024	Enhanced component diagrams with detailed descriptions for authentication, printing, and printer management modules	3.2
Final Review and Quality Check for Report of Task 3	30/10/2024	Final proofreading, consistency check, and format adjustments for the Architecture Design section	3.3
Repository Construction and Note for the Report.	05/11/2024	Initial setup of an online repository for version control and collaborative work.	4.0
Adding Documents & Materials	10/11/2024	Added folders and files for Requirement, System Modeling, and Architecture Design. Organized the structure for easy access.	4.1
Usability Testing of MVP 1	19/11/2024	Conducted usability tests on the interfaces developed in MVP 1. Collected user feedback for further improvements.	4.2
Solutions for MVP 1 Feedback	24/11/2024	Addressed issues and suggested solutions based on user feedback from usability testing. Prepared for the next sprint.	4.3
Framework and Stack Selection.	25/11/2024	Considering several criteria for choosing the technologies used for this project.	5.0

Draft setup for the Backend and Frontend.	27/11/2024	Setup database connection, path routing, basic model schema and simple frontend site.	5.1
Application Done v0	01/11/2024	In this version, we have finished everything from backend and frontend. Also, we have successfully used OAuth 2.0 for authentication via Google Account.	5.2
Application Done v1	03/12/2024	This is the completed version from the version 5.2. We, as the team members, individually participate in the testing and validating process of the application to debug and fix minor issues on the view site, also, confirming that all the views are consistent to some extent.	5.3
Recheck and Fix of Task 1 and Task 2.	05/12/2024	Fix some Use case diagrams (alternative flows, pre and post conditions…), divide the printing module of the sequence diagram into smaller submodules and give each of them a separate description.	5.4
Fix Task 3 after Teacher Review	12/12/2024	Fix the MVC Architecture diagram, fix the component diagrams by modifying the old ones with correct interactions and adding three new component diagrams for each of the Model, View, and Controller respectively.	5.5

I. Requirement elicitation

1. Overview (Task 1.1)

1.1. Domain Context

The **HCMUT Smart Printing Service for Students (HCMUT-SPSS)** is designed to streamline document printing for students at **Ho Chi Minh City University of Technology (HCMUT)**. By offering on-campus printing, students can save time, avoid travel, and conveniently access nearby printing facilities.

Currently, the **limited** number of print points (only 3-4 across both campuses) cannot meet the **high demand**, especially during midterms and finals, leading to long waiting times. Manual printer operations further limit efficiency, while restricted working hours prevent students from printing at their convenience.

In traditional off-campus printing, students face even more difficulties. The number of students far exceeds the number of print shops, resulting in long queues and extended waiting times during high-demand periods. Additionally, customizing print formats can be challenging due to miscommunication between students and shop owners. Security is also a concern, as there is no guarantee that sensitive information in the documents won't be disclosed when shared with print shops. Moreover, sending files through multiple platforms (e.g., email, Zalo, Messenger) increases the risk of errors and makes file management more complicated.

The **HCMUT-SPSS resolves these issues** by providing an online platform where students can upload documents, customize print settings, and submit jobs remotely. They only need to pick up their materials when ready, reducing waiting times and manual intervention. The platform's accessibility allows students to manage their printing anytime, improving control and reducing errors. This smart solution addresses key pain points while creating a more efficient and reliable printing experience.

1.2. Stakeholders and Needs

Students

Description: Students are the **primary** users of the system. They rely on it for printing academic and personal documents across campus using available printers. Each student has an account with a limited number of printing pages provided by the university, which they can manage through the system.

Needs:

- Upload documents and specify printing preferences (paper size, double-sided, etc.).
- Monitor and manage their page balance, and purchase more pages if needed.

- View their personal printing history and usage summary.
- Securely log in using the university's SSO system.

Student Printing Service Officer (SPSO)

Description: The SPSO manages the **technical and operational** aspects of the printing service. They are responsible for configuring system settings, managing printers, and overseeing system usage by students.

Needs:

- Manage **printers** (add, enable, disable) and **system configurations** (e.g., default page limits, accepted file types).
- Access and filter the **printing logs** of students by date or printer.
- View automated reports on system usage (monthly and yearly).
- Provide troubleshooting and technical support for system issues.

University Administration

Description: The administration oversees the system's alignment with university policies and ensures it operates smoothly to meet student needs. They are also concerned with financial and operational aspects of the service.

Needs:

- Ensure the **printing service** is effective and meets **university policy standards**.
- Monitor system efficiency and cost management (e.g., page allocation, student payments).
- Ensure compliance with privacy and data protection regulations.

BKPay Payment System Provider

Description: BKPay is the **university's online payment platform** integrated with HCMUT_SSPS. It allows students to purchase additional printing pages and ensures secure transactions.

Needs:

- Enable secure and seamless payment processing within the printing system.
- Maintain accurate transaction logs for both students and the system.
- Ensure payment security and prevent fraud.

Printer Manufacturers

Description: Printer manufacturers **supply and maintain** the printers located around the university's campuses. They ensure the hardware functions smoothly and integrates with the HCMUT_SSPS system.

Needs:

- Ensure printers are compatible with the system's requirements.
- Provide regular maintenance and troubleshooting services for the printers.

HCMUT_SSO Authentication Service

Description: HCMUT_SSO is the university's **Single Sign-On authentication service** that provides secure access to the HCMUT_SSPS system for both students and staff.

Needs:

- Provide secure, reliable login for all users accessing the printing system.
- Ensure smooth integration between the authentication system and HCMUT_SSPS.

1.3. Benefits of the System

Students:

- **Convenient** document printing with flexible options.
- **Easy** management of printing page balance and history.
- **Secure** access through HCMUT_SSO.

Student Printing Service Officer (SPSO):

- **Full control** over printer management and system settings.
- Access to detailed **logs** and automated **reports**.
- Streamlined oversight of system usage and student activity.
- Ensures system stability and quick troubleshooting.

University Administration:

- Ensures compliance with printing **policies** and privacy **regulations**.
- Improves cost management and operational efficiency.
- Data-driven insights from automated reports.

BKPay Payment System Provider:

- Increased **transaction volume** through student purchases.
- Accurate and **secure** payment tracking.
- Opens doors for **potential** future partnerships with the university for other services.

Printer Manufacturers:

- Promotes their printers as reliable and compatible with the university's system.

HCMUT_SSO Authentication Service:

- Provides **secure** and **seamless** login for all users.
- Simplified integration with the printing system.

2. Functional and Non-Functional Requirements (Task 1.2)

2.1. Functional Requirements

2.1.1. For Students:

User Registration and Authentication

- Students must be able to register and log in using their university credentials (student ID and password), ensuring only authorized users can access the service by enforcing **HCMUT Single Sign-On (SSO)** authentication.

Document Upload

- Students must be able to upload documents in various formats (PDF, DOCX, PPT, etc.). Specifying **limits on formats** could improve system performance.
- Batch upload functionality should be available for multiple documents to be uploaded simultaneously.

Printing Customization

- Students should have options to customize **print settings**, including:
    + Number of copies
    + Paper size (A4, A3, etc.)
    + Single-sided or double-sided printing
    + Color or black-and-white
    + Orientation (portrait or landscape)
    + Binding options (e.g., stapling)
- A **real-time print preview** should be provided to verify the job before submission. It should also offer **cost estimations** based on customization.

Payment Integration

- Students must be able to **make payments through integrated options** like university accounts, e-wallets, or credit/debit cards.
- The system should display an itemized cost breakdown based on the selected print settings.

Print Job Status Tracking

- Students should be able to **track the progress** of their print jobs, including **statuses** like “submitted”, “in progress”, “completed” and “ready for pickup”.
- The system must send notifications (via email or app) when their print job is ready for pickup.

Document Security and Privacy

- Students must have the assurance that their uploaded documents are **securely stored and accessible** only by authorized personnel.
- Encryption should be used to ensure secure file uploads and storage.

Error Handling and Reprinting

- Students must have the **option to request reprints** in case of *system* errors (e.g., formatting mistakes, incomplete prints).
- The system should assist in preventing *user* errors (e.g., wrong settings) through warning messages.
- The system should facilitate free reprinting if errors are due to the system or print service, without charging students again.

Document History and Management

- Students must have access to their **document history**, allowing them to view previous jobs, reprint documents, or download receipts.
- The system should offer the option to automatically **delete files post-printing after a certain number of days** to maintain security and privacy.

2.1.2. For SPSO:

Printer Management

- The system must allow the SPSO to add, enable, disable, and manage printers across campus.

Printer Configuration

- The system must be able to modify the default number of printing pages assigned to students and configure the specific dates for **distributing these page credits** each semester.
- The system also has the ability to manage and update the **types of file formats** allowed for printing, ensuring compliance with system constraints.

System Maintenance and Troubleshooting

- The system must provide tools for SPSO staff to diagnose and fix printer issues or malfunctions quickly. Remote diagnostics could be useful for large, distributed campuses like HCMUT.
- The system should enable **maintenance tasks** like printer calibration and updating software without interrupting active print jobs.

Report Generation and Usage Insights

- The system must generate **reports** detailing print job volume, total revenue, and print service usage, allowing SPSO staff to monitor **service performance**. These reports can be exported in various formats (CSV, PDF) or integrated with other university systems for analytics.
- **Usage statistics** should provide insights for optimizing printing services, identifying trends, and ensuring resource efficiency.

Student Printing History Access

- The system must provide the SPSO with the ability to view the **printing history** of all students or a specific student for a selected time period.

Efficient Resource Management

- The system must provide **real-time data** on printer resources, such as paper levels and ink/toner status.
- **Alerts** should be generated when resources (paper, ink) are running low to ensure timely replenishment.

2.1.3. For University Administration:

Access to Aggregate Printing Reports

- The system must provide university administrators with access to comprehensive, aggregate **reports on printing activities**. This includes tracking overall usage trends, print volume per department, system performance, and identifying patterns in student printing behaviors (e.g., preferred printing times, document types).

Revenue Monitoring

- Administrators must be able to **monitor the revenue** generated from paid printing services, with clear financial reports. The system should offer insights into total revenue, revenue breakdown by user groups (students, faculty), and cost analysis to help inform **pricing strategies and resource allocation**.

University-Wide Policy Management

- The system must offer tools for administrators to set and enforce university-wide printing policies, such as:
    + **Printing quotas** for students (e.g., free prints per semester or paid prints beyond a certain limit).
    + **Printing fees** for different types of printing (e.g., black-and-white vs. color, single-sided vs. double-sided).
    + **Content restrictions**, ensuring that only permissible content can be printed, in line with university policies.

Global System Configuration

- The system must allow administrators to configure global system settings, including:
    + **Default printer page allocations** for students, staff, and faculty.
    + **Printer maintenance schedules**, ensuring that printers are regularly serviced and have minimal downtime.
    + **User access levels**, determining which users have access to certain printers or settings (e.g., high-quality or specialty printers).

Peak Time Reporting and Optimization

- The system must provide detailed reporting on **peak printing times** (e.g., during exams or project submission periods) to help **optimize printer distribution and resource allocation across campus**. This could include:
    + Data on which printers are used most frequently.
    + Recommendations for redistributing resources or adding printers in high-demand areas.
    + Insights for adjusting maintenance schedules during non-peak times to avoid disruptions.
    + Dynamic resource allocation (moving jobs to available printers) to further improve overall performance.

2.1.4. For HCMUT_SSO Authentication Service:

- The **system** must be integrated with the HCMUT_SSO authentication system.
- All **user login operations** must go through the HCMUT_SSO authentication system.
- The system should offer **multi-factor authentication** for added security.

2.1.5. For BKPay Payment System Provider :

- The system must integrate with **at least one electronic payment method**.

- The system must send a payment receipt to students via email.
- Transaction logging/history is available.

2.2. Non-Functional Requirements

2.2.1. Security

- **Account recovery** options such as password resets should also be addressed.
- **Sensitive** data, such as payment transactions and personal student information, must be encrypted both in transit and at rest to ensure confidentiality and data integrity.

2.2.2. Reliability

- The system should **guarantee high uptime** (e.g., 99.9% availability) with minimal downtime for maintenance or updates.
- **Maintenance** (downtime) should be scheduled during **non-peak hours** to avoid disruptions, ensuring the system is available during critical periods such as exam seasons, thesis submissions, or other peak student usage times. It is handled through notice periods or compensation.

2.2.3. Usability

- The system must feature a **user-friendly interface** on both web and mobile platforms, allowing for easy navigation and functionality.
- The system should cater to users of varying technical proficiency, providing clear instructions, intuitive design, and support for common tasks such as document uploading, print customization, and payment processing.
- **Predictive analytics** for HCMUT SPSO is included to anticipate when supplies will run out.
- Refund policies and transaction failure handling should be implemented for smoother user experience.

2.2.4. Compatibility

- The system should be **compatible with a wide range of devices and operating systems**, allowing students to access the service from laptops, desktops, smartphones, and tablets.
- It must **support modern web browsers** (e.g., Chrome, Firefox, Safari) and **mobile operating systems** (iOS and Android), ensuring that students can print from different platforms with consistent performance.

2.2.5. Scalability

- The system must be **capable of scaling to accommodate increased user load**, **particularly during peak periods** like exam times or major academic events (e.g., thesis submissions, project deadlines), without performance degradation or delays. Automatically prioritize print jobs based on urgency.
- It should dynamically adjust to **handle a high number of concurrent users and large file uploads** to maintain system efficiency and user satisfaction.

3. Use-case Diagrams and Scenarios (Task 1.3)

3.1. Use-case table

ID	Use case Name	Description
UC001	Login	The user logins to the system
UC002	Print documents	The user uses the printing service
UC003	Buy printing pages	The user buys more printing pages
UC004	Document upload	The user uploads a document
UC005	Customize printing options	The user configures the printing options
UC006	Track printing status	The user tracks the printing status of the documents
UC007	Manage printer	SPSO manages the printer
UC008	Add printer	SPSO adds a printer
UC009	Enable printer	SPSO enables a printer
UC010	Disable printer	SPSO disables a printer

3.2. Use-case Diagram for the Whole System

Figure 1: Use case diagram for the whole system

Here, we show our Use Case diagram for our whole system (HCMUT - SSPS).

3.3. Module: Authentication

3.3.1. The Use Case Diagram in Authentication Module

Figure 2: Use case diagram for the Authentication module

Here is the Use Case diagram for the Authentication module. The detailed description about this Use Case diagram will be shown in part 3.3.2.

3.3.2. The Use Case Scenario: Authentication/Login

Use Case ID	**UC001**		
Use case name:	**Login**		
Created by:	Nguyen Quang Vinh	**Lasted updated by:**	Nguyen Quang Vinh
Dated created:	25/09/2024	**Dated last updated:**	01/12/2024
Actors:	Guest (Unauthenticated User), Student (Authenticated User), HCMUT_SSO (Single Sign-On System)		
Description:	This use case allows an unauthenticated guest to log into the system using their **HCMUT_SSO credentials**. Users can log in with their accounts to access the system's features.		
Trigger:	The user (guess or student) initiates the login process by clicking the **"Login" button** on the main interface of the website or application.		
Preconditions:	- The guest has **not** logged into the system.		

	- The user possesses **valid** credentials for HCMUT_SSO. - The login account has been assigned student **permissions**. - The user's device is connected to the **internet**. - The system is online and available for login attempts.
Postconditions:	- The user is **successfully** authenticated. - The system has **recorded** the successful application login. - The system has created a **session** for the user. - The user gains access to their personalized features based on their role (guest or student).
Normal Flows:	1. The user accesses the application (web). 2. The user selects the login method using an account. 3. The system prompts the user to enter their username and password. 4. The user enters the username & password, then clicks the login button. 5. The system successfully verifies the login information and grants the user access to the application. 6. The system logs the successful application login activity. 7. The system updates the interface according to the information of the Guest account.
Alternative Flows:	**A1: At step 4** 4.1. The user selects the option to **save the login information** before clicking the login button, and the system **automatically saves** the username and password for future logins. 4.2. Return to step 5.
Exceptions:	**E1: At step 6** ***6.1. Invalid Credentials*** If the user enters an invalid username or password: - The system displays a generic error message (e.g., "Invalid username or password"). ⇒ This avoids revealing whether the username or password was incorrect for security reasons. ***6.2. Options for the User***

	The user has the following options: - Retry: They can go back and re-enter their username and password. The system returns to Step 3 in the normal flow. - Forgot Password: The system should provide a link for users to reset their password if they've forgotten it. - Cancel: The user can cancel the login process and return to the main page.
Note and issues:	**Note**: Implement strong session management, enforce **2FA**, allow users to monitor active sessions, and notify users of new logins. **Issues**: Multiple active sessions could increase the chance of unauthorized access if one device is compromised.

3.4. Module: Printing

3.4.1. The Use Case Diagram in Printing Module

Figure 3: Use case diagram for the Printing module

Here is the Use Case diagram for the Printing module. The detailed description about this Use Case diagram will be shown from part 3.4.2 to 3.4.6.

3.4.2. The Use Case Scenario: Printing Documents

Use Case ID	**UC002**		
Use case name:	**Print documents**		
Created by:	Nguyen Quang Vinh	**Lasted updated by:**	Nguyen Quang Phu
Dated created:	25/09/2024	**Dated last updated:**	04/12/2024
Actors:	Student		
Description:	This use case allows users to upload documents and print them after approval.		
Trigger:	The user clicks the **"Print Now" button** on the navigation bar.		
Preconditions:	- The user must have an account on the system. - The user is logged into the system. - The document must be successfully uploaded and approved. - The user must have sufficient print credits. - The user selects the time, location, and printing configurations before submitting the print request.		
Postconditions:	The user will be placed in the print queue for the selected printer.		
Normal Flows:	1. The user clicks **"Print Now"** on the navigation bar. 2. The system displays a **list of available printers** for the user to select. The user selects a printer from the list. 3. The user uploads one or multiple files for printing. 4. The system **validates** each uploaded file and only displays successfully validated files in the user's file list. 5. The user selects a file from the validated file list to print. 6. The system redirects the user to a **configuration page** where they adjust print settings (e.g., single-sided, double-sided, time, date). 7. The user confirms and submits the print job. 8. The system processes the print job and adds it to the queue of the selected printer.		

	9. The user monitors the print job's progress in the **Print History** page. 10. If the print job is **still in progress**, the user can cancel it, though any completed pages will still be deducted from their print balance.
Alternative Flows:	**A1: At Step 1:** 1.1: If the user is not logged into the system, the system redirects them to the login page. 1.2: The user enters their credentials and logs in. 1.3: Continue with Step 2 in the Normal Flow. **A2: At Step 4:** 4.1: If no printers are available (e.g., under maintenance), the system notifies the user of the unavailability. 4.2: The user is notified to select another printer or return later. 4.3: The system then navigates the user back to step 2.
Exceptions:	**E1: At Step 4:** - If all printers do not meet the **filtering criteria** due to maintenance or other reasons: ⇒ The system notifies the user. **E2: At Step 7:** - The system **detects** that the uploaded file does **not meet the verification criteria** (file format, content issues, etc.): ⇒ The system notifies the user of the issue and requests a re-upload. **E3: At Step 8:** - The system **detects** that the user's remaining print credits are insufficient for the print job: ⇒ The system notifies the user of the issue and redirects them to the "Buy Print Credits" use case to purchase additional credits.
Note and issues:	No notes or issues identified.

3.4.3. The Use Case Scenario: Buy printing pages

Use Case ID	**UC003**		
Use case name:	**Buy printing pages**		
Created by:	Nguyen Quang Vinh	**Lasted updated by:**	Nguyen Nhat Khoi
Dated created:	26/09/2024	**Dated last updated:**	04/12/2024
Actors:	Student		
Description:	Allows the user to **purchase additional print pages** for their account through the integrated BK Pay payment system. The purchased pages will be available for printing.		
Trigger:	The user clicks the "Buy More Pages" button in the Printing Module interface.		
Preconditions:	- The user must have an account on the system. - The user is logged into the system. - The user's account is linked with BK Pay. - The user has selected a printer or a print job requiring additional pages.		
Postconditions:	- The user's **print page balance** is increased, and a **payment receipt** is sent to both the system and the BK Pay Payment System. - The user can now proceed with printing.		
Normal Flows:	1. The user accesses the Printing Module and clicks the **"Buy More Pages" button**. 2. The system **displays** the Buy More Pages interface, allowing the user to input the desired number of pages for purchase. 3. The user **selects** the number of pages to purchase and confirms the payment via BK Pay. 4. The system processes the payment and **sends a receipt** to the user's account and to the BK Pay Payment System. 5. Once the payment is confirmed, the system updates the user's print page balance and redirects them back to the printing options.		

Alternative Flows:	**A1: At step 2 (BK Pay Account Not Linked):** - If the user does not have a linked BK Pay account: + The system prompts the user to link their account to BK Pay. + The user follows the instructions to link the payment method. - Then the user is redirected back to step 3 to complete the transaction.
Exceptions:	**E1: Payment Failure:** - If the payment **fails** due to insufficient funds, expired account details, or a system error: ⇒ The system notifies the user of the payment failure and provides the option to retry the payment. ⇒ The system maintains the print job in the queue but does not proceed until sufficient funds are available in the BK Pay account. **E2: System Unavailability:** - If BK Pay is **temporarily unavailable**: ⇒ The system notifies the user that the payment cannot be processed at the moment. ⇒ The user is asked to try again later when BK Pay is back online.
Note and issues:	- The system should ensure that **sensitive** payment information is **securely** processed using BK Pay. - The user should receive a notification if their print job is delayed due to lack of pages or failed payment. - A clear **record of page purchases and payment history** should be available in the user's account for tracking purposes.

3.4.4. The Use Case Scenario: Document upload

Use Case ID	**UC004**		
Use case name:	**Document upload**		
Created by:	Nguyen Quang Vinh	**Lasted updated by:**	Nguyen Quang Vinh
Dated created:	26/09/2024	**Dated last updated:**	04/12/2024
Actors:	Student		
Description:	Allows the user to upload documents for printing through the system. The uploaded document will be verified before proceeding to print.		
Trigger:	The user clicks the "Upload Document" button in the Printing Module interface.		
Preconditions:	- The user must have an account on the system. - The user is logged into the system. - The user has access to a document ready for upload. - The document meets the acceptable file format criteria.		
Postconditions:	- The document is **uploaded** and **verified** for printing. - The user can proceed to customize printing options once the document is verified.		
Normal Flows:	1. The user clicks **"Upload Document"** in the Printing Module. 2. The system opens a file selection dialog for the user to choose one or multiple documents. 3. The user **selects** the files and confirms the upload. 4. The system **validates** each document for: - Supported file format. - File size limit. - Basic content compliance. For each **valid** document: + The system confirms successful upload and adds the document to the user's list of printable files.		

	For any **invalid** document: + The system notifies the user of the specific issue (e.g., unsupported format, size too large). + The invalid document is not uploaded. 5. Once all files are processed, the system provides an overview: - List of successfully uploaded files. - List of files that failed with error details. 6. The user can proceed to customize printing options for the successfully uploaded documents.
Alternative Flows:	**A1: Unsupported File Format:** - At step 4, If the uploaded document is in an unsupported format: + The system notifies the user and suggests supported formats (e.g., PDF, DOCX). + The user is prompted to upload a compatible document and return to step 4. **A2: Document Too Large:** - At step 4, If the document exceeds the allowed file size: + The system notifies the user of the size limit and prompts them to upload a smaller document and navigates the user back to the start of step 4
Exceptions:	**E1: Upload Failure:** - If the upload process fails due to network or system issues: ⇒ The system displays an error message and suggests retrying the upload or checking the network connection.
Note and issues:	- The system should allow the user to see a preview of the uploaded document to ensure accuracy. - The document verification process should be quick to avoid user delays. - Any failed uploads should be logged, and the user should receive feedback on why the upload was unsuccessful.

3.4.5. The Use Case Scenario: Customize printing options

Use Case ID	**UC005**		
Use case name:	**Customize printing options**		
Created by:	Nguyen Quang Vinh	**Lasted updated by:**	Nguyen Quang Phu
Dated created:	26/09/2024	**Dated last updated:**	30/09/2024
Actors:	Student		
Description:	Allows the user to **adjust printing preferences** such as paper size, number of copies, and binding options before submitting the document for printing.		
Trigger:	The user clicks the **"Customize Printing Options" button** after uploading a document in the Printing Module.		
Preconditions:	- The user must have an account on the system. - The user is logged into the system. - The document has been successfully uploaded and verified. - The user has available print pages for the selected options.		
Postconditions:	The user's printing preferences are saved, and the document is ready for printing.		
Normal Flows:	1. The user selects "Customize Printing Options" after uploading a document. 2. The system **displays** a set of available printing options, including: - Adjust the number of copies. - Select paper size (A4, A3, etc.). - Choose the printing type (single or double-sided). - Set page orientation (portrait or landscape). - Adjust binding options (if available). 3. The user **selects** the desired options for their print job. 4. The system calculates the total number of print pages and verifies that the user has enough pages available.		

	5. The user confirms the settings and proceeds to print or save the configuration for later use.
Alternative Flows:	**A1: Insufficient Print Pages:** - If the user does not have enough print pages to accommodate the selected options: + The system notifies the user of the shortage and provides the option to purchase more pages via BK Pay. - After finishing purchasing more pages, the user will be redirected to the printing options set (Step2).
Exceptions:	**E1: System Error During Customization:** - If a system error occurs while the user is selecting or confirming options: ⇒ The system notifies the user of the issue and suggests retrying the customization or saving the settings for later.
Note and issues:	- The system should provide a **real-time cost estimate** based on the selected printing options to help users make informed choices. - Ensure that the user is **notified** if any selected options (e.g., paper size or binding) are unavailable at the selected printer. - The system should allow users to save their preferred printing configurations for future use.

3.4.6. The Use Case Scenario: Track print status

Use Case ID	**UC006**		
Use case name:	**Track printing status**		
Created by:	Nguyen Quang Vinh	**Lasted updated by:**	Nguyen Quang Phu
Dated created:	26/09/2024	**Dated last updated:**	30/09/2024
Actors:	Student		
Description:	Allows the user to track the status of their submitted print job in real-time through the system. This helps the user stay informed of their print job's progress.		
Trigger:	The user clicks the **"Track Print Status"** option after submitting a print job in the Printing Module interface.		
Preconditions:	- The user must have an account on the system. - The user is logged into the system. - The user has submitted a print job for printing. - The system is connected to the printers to receive real-time updates.		
Postconditions:	- The user can view the **current status** of their print job (e.g., in queue, printing, completed). - The user receives a **notification** when the print job is completed.		
Normal Flows:	1. The user **submits** a document for printing and **selects** the "Track Print Status" option. 2. The system **retrieves** the current status of the print job from the printer. 3. The system displays the **print job's status**, which can include: - <u>In Queue</u>: The print job is waiting in line. - <u>Printing</u>: The document is currently being printed. - <u>Completed</u>: The print job has finished. - <u>Error</u>: There is an issue with the print job (e.g., paper jam, insufficient ink). 4. The user can **choose to receive notifications** on the progress of the print job.		

	5. Once the print job is completed, the system sends a notification to the user.
Alternative Flows:	**A1: Print Job Delay:** - At step 3, if the print job is delayed due to high volume or printer maintenance: + The system notifies the user of the expected wait time or delay. + The user can choose to cancel the job or continue waiting. - After the user decides, the system resumes the process in step 4.
Exceptions:	**E1: Printer Error:** - If a printer **error** occurs (e.g., paper jam, ink error): ⇒ The system notifies the user of the issue and suggests possible solutions (e.g., contact SPSO for support). ⇒ The print job status is updated accordingly (e.g., paused or canceled). **E2: Network/Connection Failure:** - If the system **fails** to retrieve the print status due to a network issue: ⇒ The system notifies the user and suggests trying again later.
Note and issues:	- The user should be able to **track multiple print jobs simultaneously** if they have submitted more than one. - Notifications should be timely and accurate to avoid confusion. - Ensure that the user can view detailed error messages if the print job fails, and provide actionable steps for resolution.

3.5. Module: Printer Management

3.5.1. The Use Case Diagram in Printer Management Module

Figure 4: Use case diagram for the Printing Configuration module

Here is the Use Case diagram for the Printer Management Module. The detailed description about this Use Case diagram will be shown from part 3.5.2 to 3.5.5.

3.5.2. The Use Case Scenario: Manage printers

Use Case ID	**UC007**		
Use case name:	**Manage printers**		
Created by:	Nguyen Nhat Khoi	**Lasted updated by:**	Nguyen Quang Phu
Dated created:	25/09/2024	**Dated last updated:**	30/09/2024
Actors:	SPSO		
Description:	Allows SPSO to manage printers around the campus		
Trigger:	The user clicks on the **"Manage printers"** on the navigation bar		
Preconditions:	- SPSO must be authenticated or authorized within the system - SPSO already login to the website - SPSO's device connected to the internet		
Postconditions:	The printer management interface pops up for SPSO.		
Normal Flows:	1. SPSO chooses **"Manage printers"** on the navigation bar. 2. The printer management interface pops up with the list of all the printers around the campus and the options to add, enable and disable a printer.		
Alternative Flows:	None		
Exceptions:	None		
Note and issues:	No notes or issues identified.		

3.5.3. The Use Case Scenario: Add Printer

Use Case ID	**UC008**		
Use case name:	**Add printer**		
Created by:	Nguyen Nhat Khoi	**Lasted updated by:**	Nguyen Quang Phu
Dated created:	25/09/2024	**Dated last updated:**	30/09/2024
Actors:	SPSO		
Description:	Allows SPSO to add an additional printer to the system		
Trigger:	The user clicks on the **"Add printer"** printer management interface		
Preconditions:	- SPSO must be authenticated or authorized within the system - SPSO already login to the website - SPSO's device connected to the internet - The printer details provided by SPSO must pass validation checks.		
Postconditions:	- The data of the new printer is added to the **system database** and the list of printers - The updated printer list is displayed in the printer management interface in real time. - A confirmation message is shown to SPSO indicating successful addition. - Any unsuccessful operation ensures no incomplete or invalid data is saved.		
Normal Flows:	1. SPSO chooses the option **"Add printer"** on the printer management interface. 2. The system **displays** the information of the new printer and SPSO **fills** in the new printer's details. 3. The system asks for **confirmation**. 4. SPSO confirms the operation. 5. The system **updates** the changes to the database.		
Alternative Flows:	None		

Exceptions:	**E1: At step 5** - SPSO does not confirm the operation - The system does not save the changes that has been made by the SPSO
Note and issues:	- Ensure the system validates printer details before saving to prevent invalid or incomplete data from being added. - Implement robust error handling to manage database connectivity issues. - Provide clear and actionable error messages to SPSO in case of exceptions. - Ensure that the printer list reflects the latest updates in real time after successful additions.

3.5.4. The Use Case Scenario: Enable printer

Use Case ID	**UC009**		
Use case name:	**Enable printer**		
Created by:	Nguyen Nhat Khoi	**Lasted updated by:**	Nguyen Quang Phu
Dated created:	25/09/2024	**Dated last updated:**	30/09/2024
Actors:	SPSO		
Description:	Allows SPSO to enable a printer.		
Trigger:	The user clicks on the **"Enable printer"** printer management interface.		
Preconditions:	- SPSO must be **authenticated** or **authorized** within the system. - SPSO **already logins** to the website. - SPSO's device connected to the **internet.** - The printer must currently be in a disabled state.		
Postconditions:	- The data of the printer is updated to the system database and the list of printers		

	- The updated list of printers is displayed in the printer management interface in real time. - SPSO receives a confirmation message indicating successful enabling of the printer.
Normal Flows:	1. SPSO chooses the option **"Enable printer"** on the printer management interface. 2. The system asked for **confirmation.** 3. SPSO confirms the operation. 4. The system **updates** the changes to the database.
Alternative Flows:	None
Exceptions:	**E1: At step 3** - SPSO does not confirm the operation - The system does not save the changes that has been made by the SPSO
Note and issues:	No notes or issues identified.

3.5.5. The Use Case Scenario: Disable printer

Use Case ID	**UC010**		
Use case name:	**Disable printer**		
Created by:	Nguyen Quang Vinh	Last updated by:	Nguyen Quang Phu
Dated created:	26/09/2024	Date last updated:	30/09/2024
Actors:	SPSO		
Description	Allows SPSO to disable a printer, making it temporarily unavailable for users via the printer management interface.		
Trigger:	The SPSO clicks the **"Disable" button** on a printer in the printer management interface.		
Preconditions:	- SPSO has a **valid** account with appropriate permissions.		

	- SPSO has successfully logged into the system. - The SPSO's device is connected to the **internet**. - The printer is **registered** and **online** and **enabled**.
Postconditions:	The printer is marked as disabled in the system, and it is no longer available for print jobs until re-enabled.
Normal Flows:	1. The SPSO selects **"Disable"** for a specific printer. 2. The system prompts the SPSO for **confirmation**. 3. The SPSO confirms the action. 4. The system updates the printer's status in the database, marking it as disabled. 5. The printer is **removed from the list of available devices** in the user interface.
Alternative Flows:	**A1: SPSO Cancels Action:** - The SPSO chooses not to confirm the action. - The system cancels the disable request, and the printer remains active and available.
Exceptions:	**E1: Network or Database Failure:** - If the system fails to update the printer status due to a network or database error: ⇒ The system displays an error message to the SPSO and logs the failure. ⇒ The printer's status remains unchanged.
Note and issues:	No notes or issues identified.

II. System Modeling

1. Activity Diagrams (Task 2.1)

1.1. Module: Authentication

Figure 5: Activity diagram for authentication

Description:

First, the **Student** (initially as Guest) clicks on the **"*Login*" button**, then the system will display the login interface. Here, the Student can enter their **username** and **password**, then the system will verify these to see whether they are **valid or not.** If so, the system will display the **User Interface t**o Student, otherwise it'll display the wrong username/password message and return to the login interface. Next if the user wants to logout via the **"*Logout*" button**, the system will display the Guest interface.

1.2. Module: Printing

Figure 6: Activity Diagram for Printing module

Description:

First, the **Student** chooses the **"*Print*" option** on the navigation menu, then the system will display the list of printers. After that, the Student will filter the printer list by filtering the facility, building and room.

The system will check if there are any printers (not under maintenance) that satisfy these criteria. **If** there are no available printers satisfying the **criteria**, the system will return to displaying the full printer list, **otherwise** the system will display the list of printers that satisfy the criteria.

Here, the Student will choose a printer, then the system will display a **printing form** and the **Student** will **upload the document file** needed to print into the system. The system will then verify if the document is valid or not. If not, the system will display an error message onto the screen. Otherwise, the Student will **customize the printing options** by adjusting the number of copies to print, adjusting the paper size, adjusting the printing type (single or double sided), adjusting the orientation of the document (portrait or landscape), and adjusting the binding options, and confirm the changes.

Then the system will check if the **number of pages** Student needed to print is enough or not. If not, it'll display the error message to Student, and Student must either reduce the number of pages or buy extra pages to print. Otherwise, the system will add the Student's printing request to the queue list. Once the printing is done, the system will send a notification on print status to Student, or Student can track the print status whenever they want.

1.3. Submodule: Buying pages

Figure 7: Activity Diagram for Buying pages

Description:

At the **"*print*" interface,** the Student chooses **"*Buy printing pages*"** and then the system will display the buy printing pages interface. Next, the Student will enter the **number of pages needed** to purchase and choose the payment method.

Afterwards, the **payment system** will process the payment, if **unsuccessful**, the system will display an error message onto the screen, otherwise, Student's page balance will be updated.

1.4. Module: Printer Management

Figure 8: Activity Diagram for Printer Management

Description:

First, SPSO chooses **"*Manage Printer*"**, the system will display the **"Manage Printer"** interface and SPSO will be able to view the list of printers.

Here, SPSO can choose **"*Enable Printer*"** to license a printer, **"*Disable Printer*"** to disable a printer, or add a printer via **"*Add Printer*"**. If SPSO chooses **"*Add Printer*"**, the system will display the new printer form that SPSO must fill in and save. After the above tasks, SPSO must confirm the changes, if yes then the database will be updated and return to displaying the list of printers, otherwise it'll discard the changes and return to displaying the printer list.

Besides, SPSO can choose **"*View printer*"** to see the information. Then, the system will display the information of the printer.

2. Sequence Diagram (Task 2.2)

2.1. Module: Authentication

Figure 9: Sequence Diagram for Authentication

Description:

The authentication module begins with the **Student** entering their **username** and **password** through the **submitLoginRequest(username, password)** function in the **LoginScreen (ls)**. This request is forwarded to the **userController (uc)** via the **Login(username, password)**.

The **userController (uc)** then interacts with the **userModel (um)** by invoking the **authUser(req, res)** function to **validate** the provided credentials. The **authUser** function checks the credentials and returns the authentication result to the **userController**. Upon receiving the result:

If the credentials are valid:

- The **userController** sends a message to the **UserScreen (us)** via the **changeView()** function to **update** the display for the user. The **UserScreen (us)** then invokes the **displayView()** function to show the appropriate user interface.

If the credentials are invalid:

- The **LoginScreen (ls)** prompts the **Student** to re-enter the credentials through the **requestInputAgain()** function. This repeats the login attempt in a loop until valid

credentials are provided or the process is terminated. The current session terminates, as indicated by the **destroy** symbol, if the user exits the loop without success.

2.2. Module: Printing

Figure 10: Sequence Diagram for Printing

Description:

First, Students will open the **pv** via ***ViewPrinter()*** function. Then the **pc** calls the ***FindAllPrinterInfo()*** function to retrieve all printers present in the **pv**. **pc** then requests **pm** to call the function ***GetAllPrinterInfo()*** to retrieve the list of all printers from the database, after that the list of printers will be returned and displayed at the **pv**.

2.2.1. Submodule: Buying Printing Pages

Figure 11: Sequence Diagram for Buying Printing Pages

Description:

Student chooses **"*Buy printing pages*"**, opens **bv (buyView)** via ***ViewBuying(user)*** function.

- If the inputted user is a Staff then return an error message to **pv** and display onto the screen.
- If the inputted user is a Student, then **bv** requests **bc (buyController)** to call the function ***CreatePaymentForm()*** to create a form with payment information, then return to the screen. Then Student provides payment information including the number of pages and the payment method and call ***FillPayment(value, method)*** function from **bv**, after that **bv** requests **bc** to call ***CreatePayment(value, method)***, process the transaction via the payment system and returns the transaction result to the screen.

2.2.2. Submodule: Filter Printers

Figure 12: Sequence Diagram for Filter Printers

Description:

Student filters the printer information, **pv** will call ***PrinterFilter(printer_info)*** function based on the filtered information, then **bc** calls ***FindPrinterInfo(prtiner_info)*** function to find the appropriate printers to display onto the screen.

Next **pc** requests **pm** to call ***GetPrinterInfo(printer_info)*** function to retrieve the list of printers from the database according to the provided information.

- If there are no printers satisfying the filtered information or if there are but the printer(s) are under maintenance then return an error message to **pv** and display onto the screen.
- Otherwise, return the result to **pv** and display the list of appropriate printers onto the screen.

2.2.3. Submodule: Upload Documents

Figure 13: Sequence Diagram for Upload Documents

Description:

Student then chooses a printer to print and upload the document. After that **pv** will call ***uploadDocs(printer_id, file)*** function and requests **vc (viewController)** to call ***VerifyFile(file)*** to verify the contents of the uploaded document.

If the uploaded file contains unsafe content:

- **vc** will send an error message to **pv** and display the error message onto the screen.

Otherwise:

- **vc** will return the **successful verification** result and display it onto the screen.
- After that, Student will modify and provide the printing information (time, number of pages), **pv** will call ***Modify(user, printer,id, file, time, page)***, then **pc** will create an order via the function ***CreateOrder(user, printer_id, file, time, page)*** and requests **pm** to call ***CheckOrder(user, printer_id, file, time, page)*** to export into the database in order to verify Student's information.

  If the Student's number of pages is not enough:

  + **pm** will send an error message to **pv** and display onto the screen.

  If the Student has enough number of pages:

+ **pm** returns the result to **pc**, then **pc** will proceed to request **pm** to call the function ***updateList(user, printer_id, file, time, page)*** to update the information into the database.
+ Finally **pm** returns the result to **pv** and displays onto the screen.

2.3. Module: Printer Management

Figure 14: Sequence Diagram for Printer Management

Description:

- First, **pmv (printerManagementView)** is opened by SPSO via the function ***managePrinters()***.
- Then, **pc (printerController)** calls ***getAllPrinter()*** function to retrieve all printers to display onto **pmv.**
- After that, **pc** requests **pm (printerModel)** to call ***getListPrinters()*** function to retrieve the list of printers from the database.
- Finally, the printer list is returned via ***return()*** and displayed at **pmv**

2.3.1. Submodule: Enable and Disable Printers

Figure 15: Sequence Diagram for Enable and Disable Printers

Description:

Here, SPSO has 2 choices:

A1: For enable a Printer

- First, the SPSO chooses **"Enable printer"**, then **pmv** will call ***submit("enable printer")***.
- Then, **pc** calls ***enablePrinter(ID)*** according to the ID of the chosen printer to activate that printer.
- After that, **pm** calls *enablePrinterByID(req, res)* from **pc**'s request to set the state value of printer to 1.

- Finally, update the database, *return()* and display at **pmv.**

A2: For disable a Printer

- SPSO chooses **"Disable printer"**, then **pmv** calls ***submit("disable printer").***
- **pc** calls ***disablePrinter(ID)*** according to the ID of the chosen printer to deactivate that printer.
- **pm** call ***disablePrinterByID(req, res)*** from **pc**'s request to set the state value of printer to 0.
- Update the database, ***return()*** and display at **pmv.**

2.3.2. Submodule: Add Printers

Figure 16: Sequence Diagram for Add Printers

Description:

The sequence diagram describes the process of **adding a new printer** to the system, starting from the SPSO's action and involving interactions between the **printerManagementView (pmv), printerController (pc)**, and **printerModel (pm).**

Initiating the Add Printer Process:

- The SPSO selects the **"Add printer"** button in the **pmv**.
- The **pmv** invokes the **submit("add printer")** function to start the process.

Filling Printer Details:

- The **SPSO** enters the details of the new printer (e.g., brand, model, description).
- The **pmv** sends the entered information to the **pc** using the **submit(brand, model, description, location)** function.

Controller Processing:

- The **pc** calls the **addNewPrinter(brand, model, description, location)** function to process the addition of the new printer.
- The **pc** forwards the request to the **pm** by invoking the **addPrinter(req, res)** function to save the new printer data in the database.

Database Update:

- The **pm** processes the request and updates the database with the new printer.
- Once the operation is complete, the **pm** sends a response back to the **pc**, which then forwards the result to the **pmv**.

Confirming or Canceling Changes:

- At this point, the SPSO has **two options:**

  Confirm Changes:

  + The SPSO selects **"Confirm Changes"** in the **pmv**.
  + The **pc** calls the **confirmOperation()** function to finalize the changes.
  + Once confirmed, the database is updated, and a response (**return()**) is sent back to the **pmv**.
  + The **pmv** displays the updated state reflecting the new printer.

  Cancel Changes:

  + The SPSO selects **"Cancel Changes"** in the **pmv**.
  + The **pc** calls the **confirmOperation()** function to discard the changes.
  + The changes are not saved in the database, and the process ends.

3. Class Diagram (Task 2.3)

In the report, our group decided to design the class diagram using the **MVC architecture**. **MVC (Model - View - Controller)** is a software design pattern used for developing user interfaces and organizing application logic. It divides an application into three interconnected components:

1. **Model**: This represents the **application's data and business logic**. It manages the data, responds to requests for information, and updates when necessary.
2. **View**: The View is responsible for **displaying the data** from the Model to the user. It defines how the user interface looks and interacts but doesn't contain logic for manipulating the data itself.
3. **Controller**: The Controller acts as an **intermediary between the Model and the View**. It handles user input, processes it, and sends instructions to the Model to update, while also determining which View to display.

By separating these concerns, MVC helps create **more maintainable, scalable, and testable** applications, making it a popular choice in web and software development frameworks. In the class diagram, each class type is represented by a different color: controllers are red, models are blue, views are yellow, and external components are purple, following the MVC pattern.

3.1. Authentication Module

Figure 17: Authentication Module Class Diagram

Description:

UserController

- Responsible for handling requests from the view and interacting with the model.
- **Key methods:**
  + ***register(userData)***: Registers a new user.
  + ***updateProfile(userID, userData)***: Updates the profile of a user.

UserView

- This class is part of the view, responsible for interacting with the user interface.
- **Key methods:**
  + ***login(username, password)***: Handles user login.
  + ***logout()***: Handles user logout.

UserModel

- Represents the data and business logic related to users.
- **Attributes:**
  + ***ID***, ***username***, ***password***, ***email***, ***role***: Fields to store user information.
- **Key methods:**
  + ***new()***: Creates a new user.
  + ***get()***: Retrieves user information.
  + ***set()***: Updates user information.

Authentication System

- Handles authentication logic separate from the user model.
- **Key method:**
  + ***authenticate(username, password)***: Authenticates the user's credentials.

GuestModel, **StudentModel**, **SPSOModel**

- These are specialized models inheriting from ***UserModel***.
- Both ***StudentModel*** and ***SPSOModel*** contain:
  + ***login()***: Logs in a specific type of user.
  + ***logout()***: Logs out the user.

HCMUT_SSO

- Likely part of an external single sign-on (SSO) system.
- **Key method:**
  + ***validateCredentials(username, password)***: Validates the user's credentials using the SSO system.

Key Relationships:

- **UserController** interacts with **UserModel**, using it to manage user data (e.g., registration and profile updates).
- **UserView** communicates with **UserController** to display user data and handle actions like login and logout.
- **UserController** also uses an **Authentication System** for user authentication.
- The **Authentication System** integrates with **HCMUT_SSO** to validate credentials.
- **UserModel** serves as the parent class for **GuestModel**, **StudentModel**, and **SPSOModel**, which inherit login and logout functionality.

3.2. Printing Module

Figure 18: Printing Module Class Diagram

Description:

PrintView

- This class is responsible for the user interface related to printing.
- **Key methods:**
  + ***choosePrinter()***: Allows the user to select a printer.
  + ***Upload(printerID, file)***: Uploads the file to be printed.
  + ***Modify(paper_size, printing_type, orientation, no_copy)***: Modifies printing settings.
  + ***trackPrintStatus()***: Tracks the status of the print job.

PrintController

- Manages user requests related to printing, interacting with the model to create or update print jobs.
- **Key methods:**
  + ***GetPrinterInfo()***: Retrieves information about the printer.
  + ***CreatePrintJob(userID, printerID, file, time, pageCount)***: Creates a new print job.
  + ***CancelPrintJob(printJobID)***: Cancels an existing print job.

PrintModel

- Represents the business logic and data for print jobs.
- **Attributes:**
  + ***ID***, ***user*** **(from** ***UserModel*****)**, ***file*** **(from** ***DocumentModel*****)**, ***printer*** **(from** ***PrinterModel*****)**.
- **Key methods:**
  + ***SavePrintJob(printJobID)***: Saves a new print job.
  + ***UpdatePrintJobStatus(printJobID)***: Updates the status of a print job.
  + ***DeletePrintJob(printJobID)***: Deletes a print job.

PrinterOptions

- Represents various settings related to a print job..
- **Attributes:**
  + ***numberOfCopies (integer):*** Specifies how many copies to print
  + ***pageSize (string):*** Indicates the size of the paper (e.g., A4, Letter***)***
  + ***printingType (string):*** Likely represents the type of printing (e.g., one-sided)
  + ***orientation (string):*** Specifies page orientation (e.g., portrait, landscape)
  + ***bindingOption (string):*** Indicates any binding preferences
- **Key Methods**:
  + ***new()***: Initializes a new printer instance.
  + ***get()*****:** Retrieves the details of a specific printer.
  + ***set()*****:** Updates the printer's details.

PrinterModel

- Represents the data related to the printer itself.
- **Attributes:**
  + ***ID***, ***location***, ***status***: Stores information about the printer and its availability.
- **Key Methods**:
  + ***new()***: Initializes a new printer instance.
  + ***get()*****:** Retrieves the details of a specific printer.
  + ***set()*****:** Updates the printer's details.

DocumentModel

- This class models the document to be printed.
- **Attributes:**

    + ***ID***, ***type***, ***name***, ***pageNumber***, ***verifyStatus***: Describes the document's type, name, number of pages, and whether it has been verified.

- **Key Methods**:
    + ***new()***: Initializes a new printer instance.
    + ***get()*:** Retrieves the details of a specific printer.
    + ***set()*:** Updates the printer's details.

VerifyController

- This controller handles verification logic for documents before printing.
- **Key method:**
    + ***Verify(document)***: Verifies the file before allowing it to be printed.

BuyView

- Responsible for presenting the purchase information and payment forms to the user.
- **Key methods:**
    + ***viewBuyingAmount(user)***: Displays the total amount for a user to pay.
    + ***DisplayPaymentMethod()***: Shows the available payment methods.
    + ***DisplayPaymentForm(value, method, amount)***: Shows a payment form based on the selected value, method, and amount.
    + ***viewPageAmount(user):*** Display the remaining amount of printing pages the user had left in the account.

BuyController

- Manages user requests related to purchasing and payments.
- **Key methods:**
    + ***CreatePaymentForm()***: Creates a form for the payment process.
    + ***CreatePayment(value, method, amount)***: Processes the payment.
    + ***GeneratePaymentReceipt(PaymentID)***: Generates a receipt after successful payment.

StudentModel

- Contains the data related to the student user..
- **Attributes**:
    + ***pageAmount***: Stores the page amount left in the student's account..
- **Key Methods**:
    + ***new()***: Initializes a new printer instance.
    + ***get()*:** Retrieves the details of a specific printer.
    + ***set()*:** Updates the printer's details.

UserModel

- Contains the data related to the user.
- **Attributes**:
  + ***username***, ***password***, ***role***: Stores user credentials, roles for tracking purposes.
- **Key Methods**:
  + ***new()***: Initializes a new printer instance.
  + ***get()*:** Retrieves the details of a specific printer.
  + ***set()*:** Updates the printer's details.

Key Relationships:

- **PrintView** interacts with **PrintController** to modify printing settings, upload files, and track print job status, also retrieving information from the **StudentModel**.
- **PrintController** communicates with **PrintModel** to create, update, or delete print jobs, while also retrieving information from the **PrinterModel**.
- **PrintModel** links to **StudentModel** (representing the user who submitted the print job), **PrintingOptions**(representing the configurations of a print job) and **DocumentModel** (representing the document to be printed).
- **VerifyController** is responsible for verifying documents before they are printed, interacting with the **Document** class.
- **BuyView** and **BuyController** manage payment interactions, where the view handles displaying payment-related information and the controller processes payments and receipts while updating the page amount from the **StudentModel**.
- **UserModel** serves as the parent class for **StudentModel**.

3.3. Printer Management Module

Figure 19: Printer Management Class Diagram

Description:

PrinterManagementView

This class is responsible for the user interface related to managing printers.

- **Key Methods**:
  - + ***DisplayListOfPrinters()***: Displays the list of all printers available in the system.
  - + ***showPrinterDetails(printerID)***: Shows detailed information for a specific printer identified by its ID.

PrinterController

Handles user requests related to managing printers and interacts with the model to perform printer operations.

- **Key Methods**:
  - + ***getAllPrintersInfo()***: Retrieves the information of all the printers from the system.

+ ***addPrinter(ID): bool*:** Adds a new printer to the system by its ID, returning a boolean indicating success or failure.
+ ***enablePrinter(ID): bool***: Enables or activates a printer using its ID, returning a success boolean.
+ ***disablePrinter(ID): bool***: Disables or deactivates a printer using its ID, returning a success boolean.
+ ***monitorPrinterStatus(ID)***: Monitors the status of a printer based on its ID (e.g., active, idle, offline).

PrinterModel

Represents the data and attributes related to a printer.

- **Attributes**:
  + ***ID***: A unique identifier for each printer.
  + ***location***: The physical location of the printer.
  + ***status***: A boolean indicating whether the printer is enabled or disabled.
  + ***pendingNumber****:* The number of requests on each printer.
- **Key Methods**:
  + ***new()***: Initializes a new printer instance.
  + ***get()*:** Retrieves the details of a specific printer.
  + ***set()*:** Updates the printer's details.

Key Relationships:

- **PrinterManagementView** interacts with **PrinterController** to display each printer details.
- **PrinterController** communicates with **PrinterModel** to add, enable, remove, and monitor printers, as well as assigning print jobs to printers.

4. User Interface (Task 2.4)

In this project, our group decided to choose a web application to implement. We use figma to implement our app interface. For further view here are the prototype and figma links:

- **Figma :** View here
- **Prototype:** View here

4.1. Log in

Figure 20: Dashboard

The dashboard of the HCMUT Smart Printing Service highlights its mission to provide an intelligent printing solution. It features a welcoming message emphasizing efficient printing for lecture notes, assignments, articles, and more.

Key elements include:

- ***Header Navigation***: Contains links to "Trang chủ" (Home), "Về chúng tôi" (About Us), "Dịch vụ" (Services), "Liên hệ" (Contact), and "Đăng nhập" (Login), ensuring easy navigation.
- ***Central Content:*** Displays a promotional message alongside an illustration of a person operating a modern printer, reinforcing the smart printing theme.
- ***Call-to-Action***: A prominent "In ngay" (Print Now) button encourages immediate interaction.

- ***Footer:*** Provides contact details, social media links (Facebook, Instagram, YouTube, LinkedIn, TikTok), and resources for students and customers, maintaining accessibility and connectivity.

Figure 21: Log in screen

The login screen for the HCMUT Smart Printing Service features a clean and minimalistic design:

- ***Header Navigation***: Includes navigation links like "Trang chủ" (Home), "Về chúng tôi" (About Us), "Dịch vụ" (Services), "Liên hệ" (Contact), and "Đăng nhập" (Login), maintaining consistency with the platform's overall design.
- ***Login Options:*** Centralized card displaying two login options:
  + "Tài khoản HCMUT (HCMUT Account)" for students and staff.
  + "Quản lý hệ thống (SPSO)" for system administrators.
- ***Logo***: Prominently displays the university's logo to reinforce branding and trust.
- ***Footer***: Provides contact details, campus addresses, and social media links (Facebook, Instagram, YouTube, LinkedIn, TikTok), ensuring easy access to support and additional resources.
- ***Background***: Features a subtle gray-fill background to keep the focus on the login options.

Figure 22: Student's dashboard

The student's dashboard for the HCMUT Smart Printing Service presents a well-organized and user-centric layout:

- ***Header Navigation***: Provides quick access to "Trang chủ" (Home), "Lịch sử in" (Printing History), "In ngay" (Print Now), and "Liên hệ" (Contact).
- ***Service Overview***: Central section titled "SSPS CUNG CẤP DỊCH VỤ" (SSPS Provides Services) showcasing three key features:
    + Fast Printing: Highlights quick and convenient printing anytime, anywhere.
    + Print Tracking: Promotes easy management and tracking of printing history.
    + Online Payment: Emphasizes seamless payment across platforms.
- ***Footer***: Displays university information, campus addresses, and links to student resources.
    + Includes social media icons for engagement (Facebook, Instagram, YouTube, LinkedIn, TikTok).

Figure 23: SPSO's dashboard

The SPSO's dashboard for the HCMUT Smart Printing Service provides an administrative interface for managing and monitoring system performance:

- ***Header Navigation***: Features links to "Trang chủ" (Home), "Quản lí in ấn" (Print Management), "Quản lí máy in" (Printer Management), and "Liên hệ" (Contact), along with profile and notification icons.
- ***Summary Statistics:*** Key metrics displayed in cards:
    + Total Users: 40,689 (with a percentage change indicator).
    + Total Requests: 10,293 (comparison to last week).
    + Total Revenue: $89,000 (with percentage trend).
    + Total Queue: 2,040 (current queue size and daily change).
- ***Revenue Graph***: A line chart visualizes revenue trends over time, enabling quick analysis of financial performance.
- ***Recent Requests***: A table lists recent printing requests, showing: Student ID (MSSV), Name, Timestamp, File Size, Printer, and Status.

4.2. Printing

Figure 24: Choose printer

The "Choose Printer" UI for the HCMUT Smart Printing Service is designed for efficient printer selection:

- ***Header Navigation:*** Provides links to "Trang chủ" (Home), "Lịch sử in" (Printing History), "In ngay" (Print Now), and "Liên hệ" (Contact), along with profile and notification icons for user personalization.
- ***Sidebar Menu***: Lists options for quick navigation:
  + "Chọn máy in" (Choose Printer),
  + "Tải tài liệu" (Upload Document),
  + "Thiết lập cấu hình in" (Configure Printing Settings),
  + "Mua giấy in" (Buy Paper).
- ***Search Bar***: Allows users to search for specific printers by name or location.
- ***Printer List:*** Displays available printers with details: Printer Code (Mã máy), Location (Vị trí), Status (Trạng thái) – e.g., "Sẵn sàng" (Ready). Each printer has a "Chọn" (Select) button for quick selection.

- ***Pagination***: Supports navigating through a large number of printers, displaying "1 - 4 trong số 524" (1-4 out of 524).
- ***Footer***: Contains institutional details, support information, and social media links (Facebook, Instagram, YouTube, LinkedIn, TikTok).

Figure 25: Upload files

Figure 26: Printing configuration

The "Upload Files" and "Printing Configuration" UI for the HCMUT Smart Printing Service provides a user-friendly interface for managing document uploads:

- ***Header Navigation:*** Consistent navigation links: "Trang chủ" (Home), "Lịch sử in" (Print History), "In ngay" (Print Now), and "Liên hệ" (Contact), with profile and notification icons.
- ***Sidebar Menu:*** Left-aligned options for quick navigation:
  + "Chọn máy in" (Choose Printer),
  + "Tải tài liệu" (Upload Document),
  + "Thiết lập cấu hình in" (Configure Print Settings),
  + "Mua giấy in" (Buy Paper).
- ***Main Upload Section***:
  + Title: "Tải tài liệu" (Upload Document).
  + Drag-and-Drop Feature:
  + Users can drag and drop files or click "Duyệt" (Browse) to select files.

+ Progress Indicator: Displays real-time upload progress for each document.

- ***File List:*** Shows uploaded files with a delete button for managing the list.
- ***Action Buttons:***
  + "Xem trước và thiết lập cấu hình in" (Preview and Configure Printing) for further adjustments.
  + "In ngay" (Print Now) for immediate action.
- ***Footer***: Provides institutional details, contact links, and social media access (Facebook, Instagram, YouTube, LinkedIn, TikTok).

Figure 27: Buy printing paper

The "Buy Printing Paper" UI for the HCMUT Smart Printing Service is designed for a seamless paper-purchasing experience:

- ***Header Navigation***: Provides consistent navigation links: "Trang chủ" (Home), "Lịch sử in" (Printing History), "In ngay" (Print Now), and "Liên hệ" (Contact), with profile and notification icons.
- ***Sidebar Menu:*** Allows quick access to:
  + "Chọn máy in" (Choose Printer),
  + "Tải tài liệu" (Upload Document),

+ "Thiết lập cấu hình in" (Configure Print Settings),
+ "Mua giấy in" (Buy Paper).

- ***Available Stock***: Displays remaining stock (e.g., "30 tờ A4" - 30 sheets of A4 paper).
- ***Price***: Lists price per sheet (e.g., "300đ/tờ").
- ***Quantity Selector***: Allows users to adjust quantity using "+" and "-" buttons.
- ***Payment Options:*** "Thanh toán qua BKPay" (Pay via BKPay) with a discount or "Thanh toán bằng thẻ" (Pay via card).
- ***Order Summary***: Displays total price breakdown with discount details and final amount.
- ***Action Button***: A prominent "Mua ngay" (Buy Now) button for completing the purchase.
- ***Footer***: Includes institutional details, contact links, and social media buttons for user convenience.

4.3. Printer Management

Figure 28: Printer management

The "Manage Printers" UI for the HCMUT Smart Printing Service provides administrators with an organized interface to monitor and control printer settings:

- ***Header Navigation***: Includes links to "Trang chủ" (Home), "Quản lí in ấn" (Print Management), "Quản lí máy in" (Printer Management), and "Liên hệ" (Contact), along with profile and notification icons.
- ***Sidebar Menu***: Features quick access options: "Chọn máy in" (Choose Printer), "Thêm máy in" (Add Printer).
- ***Printer List***: Displays a list of printers with key details: Printer Code (Mã máy), Location (Vị trí), Queue Size (Hàng chờ). Also it includes toggle buttons to enable or disable printers. "Info" icons provide additional details about each printer.
- ***Search and Pagination***: A search bar allows filtering of printers by name or location Pagination supports navigation through a large list of printers, showing "1 - 4 trong số 524" (1-4 out of 524).
- ***Footer***: Provides institutional information, support links, and social media access (Facebook, Instagram, YouTube, LinkedIn, TikTok).

III. Architecture Design (Task 3)

1. MVC Architecture

The team has chosen the MVC (Model - View - Controller) architecture, one of the most popular architectures, instead of using the Layered architecture to design the HCMUT_SPSS system. The reasons for this will be described clearly in the next two subsections.

Figure 29: MVC Architectural Model (Source: GeeksforGeeks.org)

1.1. Advantages and Disadvantages of MVC

Advantages of MVC:

- **Clear Separation of Responsibilities**: MVC divides the application into three distinct components - Model, View, and Controller - each with well-defined roles. This separation simplifies code organization, making it easier to manage and maintain.
- **Enhanced Scalability and Maintainability**: By isolating each component, the MVC architecture allows for easy modifications to individual parts without impacting the entire system. This modularity promotes scalability and long-term maintenance as the application evolves.
- **Parallel Development Capability**: The MVC pattern allows multiple developers to work on different layers simultaneously - one can work on the View while another focuses on the Controller, and a third works on the Model.
- **Flexible Integration of New Features**: MVC supports easy swapping or modification of components. Developers can add or replace features with minimal impact on the other parts of the application, facilitating smooth integration of new functionalities.
- **Reusable Components**: MVC encourages the development of reusable components. Views and Models, for instance, can often be repurposed in other applications, improving development efficiency and consistency across projects.

Disadvantages of MVC:

- **Increased Complexity**: The MVC architecture can introduce additional complexity, as developers need to manage the relationships and data flow between Models, Views, and Controllers.
- **Higher Learning Curve for New Developers**: For developers unfamiliar with MVC, the structure may be challenging to grasp at first, especially in understanding how each component interacts and the conventions that must be followed.
- **Potential for Performance Overhead**: In applications where each component frequently communicates with the others, there can be performance overhead due to the additional layers of abstraction. This may require optimization to maintain efficient performance.

1.2. Reasons for Choosing MVC over Layered Architecture

Figure 30: Difference between 3-tier and MVC architecture

After analyzing the MVC and Layered Architecture models, the team opted for MVC for the following reasons:

- ***Easy Task Allocation***: The MVC architecture divides the application into three main components - Model, View, and Controller. This separation simplifies task distribution within the team, enabling each member to focus on a distinct component without overlapping responsibilities. Unlike the Layered Architecture, where each layer depends sequentially on the lower layers, MVC allows for clear task delineation, reducing interdependence among team members.
- ***Concurrent Development***: MVC supports concurrent development as the Model, View, and Controller can be developed independently, promoting collaboration among multiple developers. This is especially advantageous for medium-scale projects like the SPSS system, where parallel workstreams can reduce development time. In contrast, the Layered Architecture's dependency chain between layers makes simultaneous work challenging, as testing upper layers often requires completed lower layers.
- ***Support from Frameworks and Libraries:*** MVC is widely supported by a variety of frameworks and libraries, such as ASP.NET, AngularJS, and Ruby on Rails, making it easier to adopt and implement for modern web applications. This broad support accelerates development and provides readily available resources, community support, and documentation. While the Layered Architecture has its advantages in enterprise-level applications, MVC's popularity makes it an efficient choice for smaller, adaptable projects like SPSS.

- ***Adaptability for Project Scale***: The SPSS project is a medium-sized system, where MVC's structure is particularly suitable. MVC offers flexibility by allowing components to be developed and tested individually, which simplifies management and reduces complexity. In a Layered Architecture, the tight dependency among layers could complicate updates or modifications, which is less desirable for a system that requires agility.

Figure 31: Web MVC Architecture

- ***Learning and Practical Application***: Given that MVC is a commonly used architecture pattern in web development, implementing it in the SPSS project is an excellent opportunity for the team to deepen their skills. This familiarity with MVC prepares team members for future projects, as many organizations utilize MVC for its balance of separation and integration of concerns.
- ***Scalability and Flexibility***: MVC allows for straightforward scalability and adaptation to new requirements. Each component (Model, View, Controller) can be modified independently as new features or changes are introduced. This is particularly beneficial for projects expected to evolve over time, as the SPSS system might. The Layered Architecture can also be scalable, but often at the cost of complex refactoring when requirements change.

By selecting MVC over Layered Architecture, the team aligns the SPSS project's goals with an architecture that supports **efficient development, task separation, adaptability, and real-time user interaction** - qualities that are essential for a responsive and scalable application.

2. Architectural Diagram and Deployment Diagram

2.1. Architectural Diagram for the overall design of HCMUT-SSPS system

Figure 32: Architectural diagram

This is our Architecture Diagram of the whole system. We will show more details about the interactions between each of the small parts inside View, Model and Controller in the Component Diagram parts.

Presentation Strategy

The presentation strategy for the HCMUT-SSPS system emphasizes providing a **user-friendly and intuitive interface** through distinct View components. Core screens such as User View, Authentication View, Log in View, and Printer Management View allow students to easily navigate and access functions like account management, printing services, and page purchasing. By clearly delineating each interface, the system enhances user experience and reduces the time required to locate information. Using frontend frameworks

like **React** or Angular would further streamline the development of these Views due to their flexibility and efficiency.

Data Storage Approach

In designing the **data storage strategy** for the HCMUT-SSPS system, the team is undertaking a comparative evaluation of **SQL** and **NoSQL** databases to identify the most suitable solution. The SQL option under consideration is **MySQL**, a well-established relational database management system (RDBMS), whereas **MongoDB** represents the NoSQL category as a document-oriented database. Below, we outline the **primary advantages and limitations** of each approach, followed by a **conclusion** based on their alignment with the system's data requirements.

SQL (MySQL)

- **Advantages**:
  + **Data Integrity and ACID Compliance**: MySQL ensures adherence to ACID properties (*Atomicity, Consistency, Isolation, and Durability*). Thereby providing a high degree of data reliability, essential for applications requiring precise transactional integrity.
  + **Structured Data Management**: As a relational database, MySQL is optimized for handling structured data within a well-defined schema. It is particularly effective for applications that require complex join operations and precise data relationships.
  + **Mature Ecosystem and Tooling**: MySQL, as a widely adopted SQL database, benefits from a robust ecosystem of tools, extensive documentation, and a supportive community. This maturity facilitates integrations, maintenance, and reliable performance monitoring.
- **Limitations**:
  + **Scalability Constraints**: MySQL typically relies on **vertical scaling**, which may not be optimal for applications that need to scale horizontally. This limitation can pose challenges when handling rapidly expanding data volumes or data that require distributed storage across multiple nodes.
  + **Schema Rigidity**: MySQL's **strict schema** enforcement can hinder flexibility, as altering table structures can be complex and time-consuming. This constraint may become a disadvantage in systems where data requirements are dynamic and subject to frequent changes.

NoSQL (MongoDB)

- **Advantages**:
  + **Horizontal Scalability and High Availability**: MongoDB is designed with **horizontal scaling** capabilities, allowing it to distribute data across multiple servers (sharding). This design is well-suited for handling large and continuously growing datasets in distributed environments.
  + **Schema Flexibility and Agility**: MongoDB's schema-less structure permits flexible document-based storage, making it ideal for storing heterogeneous data that may not conform to a rigid schema. This flexibility is advantageous in rapidly evolving systems where data models may change frequently.
  + **Optimized for Large Datasets and High-Performance Read/Write Operations**: MongoDB's document-oriented storage architecture enables it to efficiently handle high-velocity data and large datasets. Its architecture facilitates fast read and write operations, enhancing performance in applications that process substantial volumes of unstructured or semi-structured data.
  + **Alignment with Agile Development Practices**: MongoDB's adaptability to store complex, hierarchical data structures directly within documents aligns well with agile development methodologies, supporting rapid prototyping and iterative development cycles.
- **Limitations**:
  + **Potential for Reduced Consistency**: MongoDB, like many NoSQL databases, operates on the principles of eventual consistency, prioritizing availability and partition tolerance. As a result, it may not provide the same level of transactional integrity as MySQL, which could be a drawback for applications requiring strict data consistency.
  + **Limited Support for Complex Join Operations**: Unlike relational databases, MongoDB is less suited for complex querying involving multiple collections (equivalent to tables). This limitation could impact analytical workloads that require intricate data joins or aggregations.

Conclusion

Based on the comparative analysis, **MongoDB appears more suitable** for the HCMUT-SSPS system's anticipated data needs. The system is expected to handle diverse, potentially unstructured data at a significant scale, making MongoDB's schema flexibility, horizontal scalability, and efficient handling of large datasets advantageous. Consequently, the team is inclined to proceed with MongoDB in the subsequent stages of implementation,

recognizing that its adaptability will support both the system's current requirements and its projected growth.

API Management

API management in the HCMUT-SSPS system will follow **RESTful standards**, ensuring seamless and structured communication between **Controller** and **Model** components, as depicted in the diagram. RESTful principles enable the APIs to provide well-defined endpoints that facilitate data retrieval and updates across different system components. Each endpoint is organized around specific resources, aligning with HTTP methods such as GET, POST, PUT, and DELETE, which handle operations like fetching user profiles, updating printer settings, or logging print activities.

To secure these APIs, the system will implement robust authentication protocols, such as **JWT (JSON Web Token)** or **OAuth2**, to ensure that only **authorized users** can access sensitive data and perform specific actions. This security layer prevents unauthorized access and enhances data integrity across the **Controller** and **Model** layers.

Furthermore, the RESTful standards promote **stateless interactions**, meaning each API request from the client to the server is independent, carrying all necessary information without relying on previous interactions. This approach allows for easier scalability and faster response times, as no session state is stored on the server. The APIs will be further optimized to minimize latency, employing techniques such as caching for frequently requested data and reducing payload sizes, which will enhance overall performance and user experience in real-time interactions.

List of Models in the Diagram:

Figure 33: List of Models

- **User Model**: Manages information and data related to users.
- **Authentication Model**: Handles functions related to user authentication.
- **Log in Model**: Manages data associated with user login activities.
- **Printer Management Model**: Manages information and the status of printers.
- **Buying Pages Model**: Processes information related to purchasing additional printing pages.
- **Print Model**: Manages data related to printing requests.

These models support various functionalities of the HCMUT-SSPS system.

List of Views in the Diagram:

Figure 34: List of Views

- **User View**: Displays information and interface elements related to user data.
- **Authentication View**: Interface for user authentication.
- **Log in View**: Interface for user login to the system.
- **Printer Management View**: Interface for managing printer details and status.
- **Buying Pages View**: Interface for purchasing additional printing pages.
- **Print View**: Interface for executing and managing print requests.

Each of these Views is designed to provide user access to different functionalities of the HCMUT-SSPS system.

List of Controllers in the Diagram:

Figure 35: List of Controllers

- **User Controller**: Controls functionalities related to user management.
- **Authentication Controller**:
  + Manages activities related to user authentication, including secure login and logout operations.
  + Interfaces with **HCMUT_SSO** for Single Sign-On (SSO) authentication to validate user credentials seamlessly and securely.
- **Log in Controller**: Handles requests related to user login.
- **Printer Management Controller**: Controls printer management processes.
- **Buying Pages Controller**:
  + Manages requests related to the purchase of additional printing pages.
  + Communicates with **BKPay** to process payments securely. The controller ensures the transaction is completed successfully and updates the user's page balance upon confirmation from **BKPay**.
- **Print Controller**: Controls and manages print request operations.

The Controllers coordinate with the Views and Models to ensure smooth and efficient system operations within the HCMUT-SSPS.

2.2. Deployment Diagram for the overall design of HCMUT-SSPS system

2.2.1. Deployment Diagram

Figure 36: Deployment diagram

Here is the **Deployment diagram** of the whole system. Some main parts of this diagram are: the Client Computer, the Frontend Environment, the Backend Environment, the API gateway, and the Cloud database server.

2.2.2. Description for the Deployment diagram

The deployment diagram for the Student Smart Printing Service (**SSPS**) system illustrates the allocation of components across distinct hardware and software environments. This architecture is structured into three primary nodes: **Client's Computer**, **SSPS Server**, and **Database Server**. Each node hosts specific components and execution environments, contributing unique roles to fulfill the overall system functionality.

Client's Computer

Role: The Client's Computer serves as the user interface, facilitating interactions between the user and the SSPS system.

- **Components**:
  + **Web Browser**: Acts as the main access point, allowing users to connect to the SSPS system and initiate interactions such as logging in, managing print jobs, and purchasing additional pages.
  + **HTML5 Artifact**: Represents the front-end interface rendered within the web browser, providing users with a responsive and accessible application experience.
- **Communication Protocol**: The Client's Computer connects to the SSPS Server via the **TCP/IP** protocol. Through this setup, user requests (e.g., login or print commands) are transmitted to the server, which processes them and returns responses (e.g., printing status updates) for display on the client-side interface.

Figure 37: Client's Computer in SSPS system.

SSPS Server

Role: The SSPS Server is responsible for handling application logic, managing user interactions, and coordinating the system's printing functionalities. It is divided into two primary execution environments to support both back-end processing and front-end interface logic.

NodeJS Environment (Back-End)

- **Purpose**: This environment handles the core back-end functionalities, such as user authentication, print job management, and page purchasing processes.
- **Framework**: The back-end processing leverages the **Express.js** framework, which is instrumental in developing RESTful APIs and managing routing efficiently.
- **Components**:
  + **Authentication Component**: Implements user authentication processes, ensuring secure access to the system, and is represented by `Authentication.js`.
  + **Buying Pages Component**: Manages the purchasing of additional printing pages for users, implemented within `BuyingPages.js`.
  + **Printing Component**: Oversees the processing and management of print jobs, represented by `Printing.js`.
  + **Printer Management Component**: Handles configuration and monitoring of printer status, contained within `PrinterManagement.js`.
- **Implementation**: Each component in this environment is encapsulated in a JavaScript artifact (file) dedicated to handling specific back-end operations. This modular approach enhances code maintainability, allowing each component to perform discrete tasks without interdependency.

ReactJS Environment (Front-End)

- **Purpose**: The front-end environment manages client-side logic, including data routing and front-end data modeling, to support a dynamic and interactive user interface.
- **Framework**: The front-end environment utilizes Vite as the build tool alongside React and Material UI to design a cohesive, accessible, and responsive user interface.
- **Components**:
  + **Controllers Component**: Encapsulates front-end control logic, providing business logic processing for each module, contained in `Controllers.js`.
  + **Routes Component**: Manages application routing, defining pathways between different views, implemented within `Routes.js`.

    + **Models Component**: Defines front-end data models to interact with and represent the server-side data, implemented as `Models.js`.
    + **App Component**: Acts as the main entry point for the ReactJS application, configuring Express and connecting routes, represented by `App.js`.
    + **Server Component**: Facilitates communication between the front-end and back-end environments, implemented in `Server.js`.

- **Implementation**: Each front-end component is implemented as a modular JavaScript file, providing a structured and organized approach to managing the client-side application logic.
- **Communication**: The SSPS Server establishes communication with the Database Server through **REST API** over the TCP/IP protocol. This RESTful architecture supports modular interactions between the server and the database, ensuring data consistency and efficiency in data handling operations.

Figure 38: SSPS Server (includes Frontend, Backend and API gateway) in SSPS system.

Database Server

Role: The Database Server is responsible for storing and managing all system data, including user profiles, printing logs, printer configurations, and transactional data.

- **Environment**: The SSPS system utilizes **MongoDB Atlas Cloud Database** as its primary data storage solution, chosen for its scalability, high availability, and remote accessibility.
- **Database Features**:
  + **Remote Connection**: The **MongoDB Atlas** cloud environment facilitates remote data management, removing the necessity for on-premises storage and enabling distributed access for enhanced scalability.
  + **Data Management**: MongoDB Atlas provides comprehensive data management tools, allowing administrators to view, query, and delete data directly from the online interface, supporting flexible data manipulation.
  + **Schemas**: The database organizes data into various collections, including `Log Collection`, `Customer Collection`, `Printer Collection`, `Transactions`, `Orders`, and `Products`. Each collection is structured to support specific data types required by the SSPS system and to maintain relational consistency among data elements.

Figure 39: Database Server in SSPS system.

3. Component Diagram

3.1. Module: Authentication

Figure 40: Component Diagram for Authentication Module

Description:

The component diagram for the Authentication module illustrates a well-structured system using the **MVC** architecture.

- The **View Layer** is represented by the **Login component**, which provides the interface for users to log in and log out of their accounts. This component is responsible for collecting user input and passing it to the controller.
- The **Controller Layer** is represented by the **UserController component**, which serves as the intermediary between the view and the model. It handles the business logic for authenticating user accounts and processes login and logout requests.
- The **Model Layer** is represented by the **UserModel component**, which manages user account data. It is responsible for storing user information and performing database queries related to authentication.

Communication flows from the **Login component** to the **UserController** for processing, and then to the **UserModel** for data validation and storage. This structure ensures separation of concerns, scalability, and maintainability of the system.

3.2. Module: Printing

Figure 41: Component Diagram for Printing Module

Description:

In the component diagram of the printing module, the view layer includes two components called **printView -** which provides the interface for the user to perform the operations such as choosing a printer, switching to buying printing pages - and **BuyScreen -** which provides the interface for the user to purchase printing pages. The controller layer includes two components, **PrintController** and **BuyController**. The **PrintController** component provides the methods for searching information and choosing the printer. The **BuyController** component provides the methods for creating the buying printing pages form. The model layer includes a component called **PrintModel** for storing and performing operations on printing such as get printer information, verify and store the document file for printing.

3.3. Module: Printer Management

Figure 42: Component Diagram for Printer Management Module

Description:

In the component diagram of the Printer Management module, the view layer includes one component called **pmView** providing the interface for SPSO to manage the printers (add, enable, disable printers). The controller layer includes one component called **PrinterController** providing the methods for managing the printers such as add, enable, and disable printers. The model layer includes one component called **PrinterModel** for storing and performing actions related to managing printers such as add, enable and disable printers. Inside the **PrinterModel** contains 2 objects called **PrinterDAO** (to perform operations on the database) and **PrinterDTO** (to store the printer's data).

The interfaces include:

- **managePrinter**: provides the methods for managing the printers (add, enable, disable).
- **printerData**: provides the API to retrieve the printer's data for management (add, enable, disable).

3.4. Interactions between specific components in the Models, Controller, View

Figure 43a: Component Diagram for Models

Description:

This diagram represents the Model Layer of the Student Smart Printing Service (SSPS) system. The **User Model** acts as the central hub, connecting with other models to facilitate core functionalities such as managing printers (**PrinterManagement Model**) (if the user is the SPSO), handling document printing (**Print Model**), and purchasing printing pages (**Buying Pages Model**). Secure login functionality is managed by the **Login Model**, which works with the **Authentication Model** to verify user credentials (usernames and passwords).

The diagram highlights the backend's modular structure, ensuring separation of concerns and efficient data flow. The **User Model** coordinates actions like printing page purchases, document printing, and printer management (SPSO), while authentication ensures system security. This design supports clean interactions between models, enabling smooth operations and scalability of the SSPS system.

Figure 43b: Component Diagram for Views

Description:

This diagram represents the View Layer of the HCMUT-SSPS system. It consists of six main components: **User View** (central hub), **PrinterManagement View**, **Print View**, **Buying Pages View**, **Authentication View**, and **Login View**. The **User View** connects to other components, enabling users to manage printers, print documents, and purchase pages, while ensuring secure access through the **Login View** and **Authentication View**. Each component has a distinct role, promoting modularity and separation of concerns.

The flow starts with users logging in via the **Login View**, authenticated through the **Authentication View**. Once logged in, users can access functionalities such as managing printers (**PrinterManagement View**) (only for SPSO), printing documents (**Print View**), or buying printing pages (**Buying Pages View**). This modular design ensures a clear and organized user interface, with data flowing seamlessly between the components, controllers, and models in the broader MVC architecture.

Figure 43c: Component Diagram for Controllers

Description:

This diagram represents the Controller Layer of the Student Smart Printing Service (SSPS) system, showing how different controllers manage interactions between the views and models. The **User Controller** is central, coordinating tasks such as managing printers (**PrinterManagement Controller**) (SPSO only), handling printing operations (**Print Controller**), and enabling printing page purchases (**Buying Pages Controller**). Secure login functionality is managed by the **Login Controller**, which interacts with the **Authentication Controller** for credential verification.

Each controller is responsible for processing user requests and directing actions to the appropriate models, ensuring smooth communication between the system's components. The **User Controller** links core functionalities, including printing, buying pages, and managing printers, while the **Login Controller** ensures system security through user authentication. This modular and organized structure enables clear separation of concerns and efficient control flow within the SSPS system.

IV. Implementation - Sprint 1 (Task 4)

1. Setting up an online repository for version control

Before using the following github repository, our group decided to use Google Drive to keep track of all the references, images, documents, group information, etc, in order to finish from task 1 to 3.

Figure 44a: Our Drive Folder for collaborating with

Figure 44b: Details about the subfolders in our Drive folder

Now, we have create and push our old work to the new github repository and it can be accessed via this link: https://github.com/Frankie2030/HCMUT-SSPS-241

Figure 45: Our Github Repository

Figure 46: A part of the README file on our Github Repository

2. Adding documents, materials and folders for Requirement, System modeling and Architectural design

Our repository has 2 main folders: code and documentation.

Figure 47: 2 folders and a README file

First, let's talk about the **documentation folder.**

This folder contains all the documents and materials related to this project. We have further splitted this folder into several subfolders such as:

- **Signature**: here we store the signature of team members in order for later used as a digital signature.
- **Team Information:** here we contain our team member information such as: fullname, student id, student email.
- **Task**: here is the main folder of the documentation folder, in which we further separate into smaller folders based on the task splitted from the assignment.
    + ***Requirement Elicitation:*** This folder contains the information that we maintain for task 1 of the project.
    + ***System Modeling:*** This folder contains the information that we maintain for task 2 of the project. Also all the images, drawings, diagrams link (for example, from draw.io…) are here too.
    + ***Architecture Design:*** This folder contains the information that we maintain for task 3 of the project.

Figure 48: Detailed subfolders and Files in Document Folder

Figure 49: Our Team Member Information

Figure 50: The divided folders based on the Task given

Figure 51: Preparation for the Task 2

Figure 52: First page of our Report (at the submission of task 2)

Figure 53: Preparation for the Task 3

Second, about the ***code folder.*** This contains the code for the **frontend** and **backend** parts.

The structure of the code is organized as followed:

- **frontend**: this folder contains the source code of our frontend of the project, which will be built via React. The **src** folder here is divided into:
    + **assets**: this folder contains images, icons and some other files that will be used in this application of the project.
    + **components**: this folder contains all the components that can be reused throughout the project
    + **pages**: this folder contains all the React pages of the project.
    + **slices**: this folder contains all the slices of Redux, which we will use to manage the state of the project application.
    + **main.jsx:** this file is the starting point of the application, including the display of the App and also the Redux and routing configuration.

Figure 54: View of the Code Folder

Figure 55: View of our frontend folder

Figure 56: View of our src subfolder of the frontend folder

Figure 57: View of our components subfolder of the frontend folder

Figure 58: View of our pages subfolder of the frontend folder

Figure 59: View of our slices subfolder of the frontend folder

Figure 60: View of part of the main.jsx file of the frontend folder

- **backend**: this folder contains the source code for the back-end of the application, including the server and the configuration of the server. This part has the purpose of managing the connection between the printers and the database behind.

Figure 61: View of folders and files contained in the backend folder.

3. Conducted a usability test with the user interface you developed in MVP 1

3.1. Test the Usability of MVP 1

To conduct an evaluation, the team decided to create a platform and gather **feedback** from users regarding their experience with the team's website to determine whether they liked it. The results showed that most users felt comfortable with the user interface of **MVP1**, suggesting that only minor improvements would be needed to achieve a more polished look in **MVP2**. However, Figma proved to be somewhat challenging in conveying the team's intended design changes to users. The questions in the **Google Forms** survey were primarily used to collect preliminary or basic opinions on the user interface from the users' perspectives. After the survey, the team made several adjustments to enhance the system's user-friendliness and improve the overall user experience.

Survey link: https://forms.gle/CBV6Xk8bt27TXTs5A.

After some time waiting for the testers and participants to check our UI of the application and grade them, we got numerous feedback which we found very insightful. From here we will call those testers and participants generally as users.

Figure 62: Statistical data about the class that the participants took part in.

As can be seen, the class that has the most people taking part in this survey is class CC01 (Group 8), which is also our current class. And in total, there are 20 users that help us to do this check, and all of them are currently studying Software Engineering this semester in class CC also.

3.1.1. Homepage Interface

Figure 63: The Question & User Ratings data about the Homepage Interface.

Most users rated it a 3 (60%), indicating a neutral opinion. Other ratings include 4 (15%), and 5 (25%), with no users rating it a 1 or 2. The **"Homepage" interface** received mostly

average ratings, suggesting room for improvement to enhance user satisfaction and make the interface more engaging or user-friendly.

3.1.2. Login Interface

Figure 64: The Question & User Ratings data about the Login Interface

The chart shows user ratings for the **"Login" interface** of a website. Most users rated it a 4 (50%), while 45% rated it a 5, just 1 tester rated at 3 and no users rated it below a 3. The **"Login" interface** received high ratings, indicating strong user satisfaction and effectiveness in design. This suggests that the login process is intuitive and meets user expectations well.

3.1.3. Student's Dashboard Interface

Figure 65: The Question & User Ratings data about the Student Dashboard (Image 1)

Figure 66: The Question & User Ratings data about the Student Dashboard (Image 2)

A majority of users rated it a 4 (50%), while 45% rated it a 5. Only 1 user rated it below a 4. The interface received high ratings, indicating that users are very satisfied with its functionality and usability, with the vast majority finding it effective and easy to navigate.

3.1.4. SPSO's Dashboard Interface

Figure 67: The Question & User Ratings data about the SPSO Dashboard (Image 1)

Figure 68: The Question & User Ratings data about the SPSO Dashboard (Image 2)

The question received 20 responses. Most users (65%) rated the interface a 5, while 25% rated it a 4. Only 2 users rated 3, indicating a positive user experience.

3.1.5. Choosing Printer Interface

Figure 69: The Question & User Ratings data about the choosing Printer Interface (Image 1)

Figure 70: The Question & User Ratings data about the choosing Printer Interface (Image 2)

The chart displays user ratings for the **"Choosing Printer" interface** on a website from 20 users. The majority (55%) rated it a 3, indicating a neutral opinion, with smaller groups rating it a 2 (5%), 4 (20%), or 5 (20%). No users rated it a 1.

3.1.6. Uploading Files Interface

Figure 71: The Question & User Ratings data about the Uploading File Interface (Image 1)

Figure 72: The Question & User Ratings data about the Uploading File Interface (Image 2)

The majority of users rated it a 3 (55%), with smaller groups rating it a 4 (25%), and 5 (20%). No users rated it a 1 or 2. The ratings suggest that users have a generally neutral to positive view of the **"Uploading File" interface**, with a significant portion giving it an average rating. Improvements could be made to enhance user satisfaction and raise the ratings.

3.1.7. Printing Configuration Interface

Figure 73: The Question & User Ratings about Printing Configuration interface (Image 1)

Figure 74: The Question & User Ratings about Printing Configuration interface (Image 2)

A majority of users rated it a 4 (55%), while 35% rated it a 5. Only 2 users rated at 3. No users rated it below a 3. The **"Printing Configuration" interface** received high satisfaction ratings, with nearly all users giving it a 4 or 5. This indicates that users find the interface highly functional and user-friendly, with little room for improvement needed.

3.1.8. Buying Printing Pages

Figure 75: The Question & User Ratings data about the Buy Papers interface (Image 1).

Figure 76: The Question & User Ratings data about the Buy Papers interface (Image 2).

The majority of users rated it a 4 (55%), while 35% of users rated it a 5 and 10% rated it a 3. No users rated it below a 3. This suggests that the interface is effective and well-designed, meeting user needs and expectations.

3.1.9. Printer Management Interface

Figure 77: The Question & User Ratings data about Printer Management interface (Image 1)

Figure 78: The Question & User Ratings data about Printer Management interface (Image 2)

Most users rated it a 4 (40%), followed by a 5 (50%), with a smaller group rating it a 3 (10%). No users rated it below 3. The **"Printer Management" interface** has high user satisfaction, as indicated by the majority of positive ratings. This suggests that the interface is effective and meets user expectations, with a potential for minor improvements to further enhance user experience.

3.2. Solution for those remaining problems of MVP 1

3.2.1. Summary of the criterias and other things that we must improve based on user feedbacks

The below chart displays **overall user experience ratings** for a website, based on 102 responses. Most users rated it a 8 (45%), followed by a 9 (30%) and a 10 (15%). A smaller group rated it an 7 (10%), with no ratings below 7. The high ratings reflect strong overall user satisfaction, with the majority of users expressing a very positive experience. This indicates that the website meets user expectations well, with only minor areas for potential enhancement.

Figure 79: The Overall rating for the User Experience

The team also conducted a survey on website proficiency among some users.

Figure 80: The User confidence when experiencing the website

After getting users to help us fill in the form, we get:

- 20% (Red) chose "Maybe (50% confident)" indicating some uncertainty.
- 75% (Blue) selected "Yes (100% confident)" showing full confidence in using the website.
- 5% (Yellow) chose "No (0% confident)" expressing a lack of confidence.

Conclusion: Overall, the website is intuitive and usable, with nearly a third of users expressing full confidence in navigating it without guidance. However, with 25% of users feeling less and no confidence, some minor improvements must be done to enhance the usability and clarity. These adjustments would likely boost user confidence, making the website even more user-friendly for all users.

3.2.2. Our suggested solutions

After summarizing all the feedback, we have made some improvements to our user interface:

Improvements on the Homepage Interface

The overall rating of our **Homepage** is not high (60% of users rated a 3/5). After considering all possible problems and discussing for finding the best solution, our team decided to remake the homepage by changing the header, footer and also adding a gray-fill background to make it look more attractive. We consider synchronizing these changes to all the other pages because all the pages use the same kind of header, footer and background.

Here is the Design after Improvements.

Figure 81: Improvement on the UI of the Homepage (Image 1)

Figure 82: Improvement on the UI of the Homepage (Image 2)

The new homepage has been improved with better UI and design. Here's a detailed breakdown of the homepage, split into sections:

Header Section

- *University Logo and Name:* The header prominently features the logo of the university (BK) on the left, indicating branding consistency.
- *Navigation Bar:* Includes key links like "Về chúng tôi" (About Us) and "Đăng nhập" (Login), ensuring ease of navigation.
- *Background Design*: A blue-patterned background gives a professional and clean feel, representing the institution's technical focus.

Hero Section

Main Image Slider:

- Displays a large image of the university campus as the focal point, accompanied by text overlay: "Quản lý và theo dõi lịch sử in ấn dễ dàng" (Manage and track printing history easily).
- This section highlights the primary service in an eye-catching manner.

Slider Functionality: Users can navigate through other potential visuals or messages, though only one image is shown here.

Service Introduction Section

Section Title: "SSPS CUNG CẤP DỊCH VỤ" (SSPS Provides Services).

Three Key Services:

- *Printing Services:* Illustrated with an image of printers and printing supplies.
- *Management and Tracking*: Depicted with a calendar/diary-like image.
- *Online Payment*: Shows an image of a smartphone with a coffee cup.

Commitment Section

Section Title: "SSPS CAM KẾT MANG ĐẾN DỊCH VỤ" (SSPS Commitment to Services).

Four Key Commitments:

- *User-Friendly (Thân thiện):* Icon of a smiley face, symbolizing an easy-to-use interface.
- *Compatibility (Tương thích)*: Icon of a checkmark, indicating broad support.
- *Performance (Hiệu suất)*: Icon of a graph or speedometer.
- *Security (Bảo mật):* Shield icon representing data security.

Footer Section

- *University Branding:* Reiterates the university's logo, name, and address details.
- *Contact Information:* Provides essential support links and contact details for users needing help.
- *Social Media Links*: Icons for platforms like Facebook, YouTube, and TikTok ensure users can engage with the institution on multiple channels.

<u>Improvements on the Uploading Interface and Choosing Printer Interface</u>

- This page's overall rating is also as **low** as the Homepage. We have rechecked the visualization of this page and guess that maybe the clients / customers need to see more information about the Printer in order for them to gain more trust about the Printer they would choose (e.g. description and brand of the printer).
- Also the rating of both interfaces is low, maybe because the box for choosing among the 4 buttons on the left is not very clean and seems bad. **Our solution** is to change their sizes and colors, also reordering and make them have more consistent manner.

Figure 83: Improvement on the UI of the Choosing Printer Interface

The improved UI of the **"Choosing Printer Interface"** is designed for a streamlined and user-friendly experience. It features:

Header Navigation: A header with navigation links ("Trang chủ," "Lịch sử in," and "In ngay") for quick access to key functionalities. A consistent design with the institution's logo.

Sidebar Menu:

- A left-aligned menu displaying options like "Danh sách máy in" (Printer List), "Danh sách tài liệu đã duyệt" (Approved Documents), "Xem trước & chỉnh cấu hình in" (Preview & Configure Print), and "Mua giấy in" (Buy Paper).
- The menu provides clear, categorized actions for users.

Printer Selection Panel: Displays a list of printers with details such as printer code ("Mã máy"), location ("Vị trí"), and queue status ("Hàng chờ"). Each printer is accompanied by an icon and a "Chọn" (Select) button for easy interaction.

Pagination: At the bottom of the printer list, pagination options ("1 - 5 trong số 1000") allow users to navigate through large lists efficiently.

Figure 84: Improvement on the UI of the Uploading File Interface

The improved UI of the "Uploading File Interface" is designed to enhance the file-uploading experience with a clean and user-friendly layout:

Header Navigation: Consistent with other pages, the header includes navigation links ("Trang chủ," "Lịch sử in," and "In ngay") and the institution's logo for branding.

Sidebar Menu: A left-aligned menu provides quick access to key features, including:

- "Danh sách máy in" (Printer List),
- "Danh sách tài liệu đã duyệt" (Approved Documents),
- "Xem trước & chỉnh cấu hình in" (Preview & Configure Print),
- "Mua giấy in" (Buy Paper).

Main Upload Section:

Title: "Tải tài liệu" (Upload Document).

Drag-and-Drop Feature:

- A central area allows users to drag and drop files or click the "Duyệt" (Browse) button to select files manually.
- Supported file formats (PDF, PNG, JPG, etc.) are clearly indicated.

File List: Displays the names of uploaded files with an option to remove files using the red trash can icon.

Progress Bar: Provides real-time feedback on upload progress for each file.

Action Buttons:

- "Xem trước và thiết lập cấu hình in" (Preview and Set Print Configuration) button in purple for additional options.
- "In ngay" (Print Now) button in green for immediate action.

V. Implementation - Sprint 2 (Task 5)

1. Using Frameworks

1.1. Front-end

For the front-end implementation of our project, we utilized several **modern web technologies** to achieve a clean, responsive, and efficient user interface.

ReactJS Framework: We chose **ReactJS**, a powerful JavaScript library, to design our front-end due to its flexibility and component-based architecture. ReactJS enables developers to create complex user interfaces by breaking down the application into **smaller, reusable components**, which streamlines the development process. By leveraging React's capabilities, our team was able to:

- **Create modular components**: This made it easier to manage, debug, and scale the project by isolating functionality into distinct parts.
- **Implement a SPA (Single Page Application)**: ReactJS allows our web application to function as a **single-page application** where the initial load fetches the **entire codebase**. This approach results in slightly longer initial load times but ensures a smoother and faster user experience during subsequent interactions, mimicking the feel of a native mobile app.
- **Optimize rendering with the Virtual DOM**: React's Virtual DOM ensures efficient rendering by only updating components that have changed, enhancing the overall performance.

Vite Build Tool: To streamline our development workflow, we integrated **Vite** as our build tool. Vite is optimized for speed and offers **hot module replacement (HMR)**, allowing us to see changes in real-time as we develop. The benefits of using Vite include:

- **Faster build times**: Vite pre-bundles dependencies using **esbuild**, resulting in much faster hot reloads.
- **Modern JavaScript support**: It leverages native ES modules and modern browser features, optimizing performance.

TailwindCSS for Styling: For styling, we opted for **TailwindCSS**, a **utility-first CSS framework** that allows for **rapid and custom styling** directly in our markup without writing traditional CSS files. Unlike other CSS frameworks like Bootstrap, Tailwind focuses on utility classes, giving us full control over the design without being constrained by predefined components. The advantages of using TailwindCSS include:

- **Highly customizable**: By using utility classes, we can design elements exactly as we need without being restricted by predefined styles.
- **Responsive design**: Tailwind makes it easy to implement responsive layouts with minimal effort, ensuring the site looks great on all devices.
- **Clean and maintainable code**: Since TailwindCSS allows for inline styling, our codebase remains clean and easy to maintain.

Material Tailwind & Flowbite: We extended the styling capabilities of TailwindCSS by incorporating **Material Tailwind** and **Flowbite** for pre-built components such as buttons, cards, and forms. These libraries provide a cohesive design system that integrates well with Tailwind, allowing us to rapidly prototype and develop UI components without compromising on design quality.

- **Material Tailwind**: This library brings Material Design principles into our TailwindCSS workflow, giving our project a modern, sleek appearance.
- **Flowbite**: Provides ready-to-use components that are customizable with Tailwind classes, further speeding up our development process.

React Router for Navigation: For handling routing in our single-page application, we used **React Router**. This library allows us to implement dynamic routing, enabling users to navigate through different pages of our application without triggering a full-page reload. This approach ensures a seamless user experience.

State Management with Redux Toolkit: To manage global state across our application, we integrated **Redux Toolkit**. This is essential for handling complex state logic, especially when dealing with user authentication, notifications, and data that needs to be accessed by multiple components. Additionally, **Redux Toolkit** provides simplified and standardized methods for creating Redux slices, actions, and reducers, reducing boilerplate code and improving maintainability.

Additional Libraries and Tools using in this Project:

- **React ApexCharts**: We utilized this library for visualizing data and displaying charts, which is crucial for representing statistics and reports within our application.
- **React-PDF & PDF-Lib**: These libraries are used to render and manipulate PDF documents directly in the browser, allowing users to view and download reports.
- **Moment.js**: To handle date and time formatting, we employed Moment.js, ensuring consistency throughout the application.

- **React-Redux & React Hooks**: These packages are essential for leveraging the power of hooks in managing state and lifecycle methods efficiently.

Code Quality and Linting: To maintain code quality and ensure consistency, we integrated **ESLint** along with several plugins:

- **ESLint Plugin for React & Hooks**: Ensures adherence to React best practices.
- **Prettier**: Automatically formats code according to a defined style guide, helping the team maintain clean and readable code.
- **Prettier Plugin for TailwindCSS**: Ensures that Tailwind utility classes are organized in a consistent manner.

By combining the power of **ReactJS, TailwindCSS, and a suite of supporting libraries and tools**, we were able to develop a front-end that is not only efficient and scalable but also visually appealing and user-friendly. The use of **modern** frameworks like Vite, React Router, and Redux Toolkit, along with the adoption of best practices for code quality, has allowed our team to deliver a responsive, high-performance web application that enhances user experience.

1.2. Back-end

For the backend of the **Student Smart Printing Service (SSPS)** project, the team utilized a combination of modern web technologies and libraries to create an efficient, scalable, and secure system. Below is an overview of the technologies used and the rationale behind their selection.

Node.js & Asynchronous Event-Driven Model

The backend server is built on **Node.js** due to its **non-blocking, asynchronous, and event-driven architecture**. This design is highly efficient for handling multiple simultaneous requests, making it ideal for applications where performance and scalability are critical. The asynchronous nature of Node.js allows it to handle I/O-bound operations (like database queries) without blocking the main thread, optimizing resource usage and improving the overall performance of the application.

Express.js Framework

To simplify **server setup and routing**, the team used **Express.js**, a lightweight web framework for Node.js. Express provides a straightforward way to define **routes**, manage **middleware**, and build **RESTful APIs**, allowing the team to develop robust back-end services quickly and efficiently. Additionally, **Express** is widely used due to its flexibility and minimalistic approach, making it an excellent choice for building scalable applications.

MongoDB with Mongoose

For the database, **MongoDB** was chosen due to its flexible schema design, which does not require predefined structures. This **NoSQL** database allows us to store data in collections without enforcing a rigid schema, making it easier to adapt to changes in data requirements over time. **Mongoose** is used as an **Object Data Modeling (ODM)** library, providing a convenient way to interact with MongoDB by defining schemas, performing validation, and executing **CRUD** operations. It simplifies working with MongoDB by abstracting the complexity of database queries.

User Authentication with Passport.js and Google OAuth

The system uses **Passport.js** for authentication, particularly with **Google OAuth 2.0**. This allows users to log in using their **Google accounts**, simplifying the authentication process and enhancing security. The use of Google OAuth ensures that user credentials are handled securely, without storing passwords directly in the system.

Security Enhancements

To protect the application from common security vulnerabilities, the team integrated several middleware packages:

- **Helmet**: Adds various HTTP headers to secure the app against potential threats like cross-site scripting (XSS) and clickjacking.
- **Cookie-session**: Manages user sessions securely using cookies, reducing the risk of session hijacking.
- **CORS (Cross-Origin Resource Sharing)**: Ensures that the front-end (running on a different port) can communicate with the back-end without triggering cross-origin issues.

File Uploads with Multer

The project uses **Multer** to handle file uploads, such as documents that students need to print. Multer efficiently manages multipart/form-data requests, which is necessary for handling file uploads in a Node.js environment.

Scheduled Tasks with Node-cron

To automate scheduled tasks, the team utilized **Node-cron**. This is particularly useful for automating maintenance tasks like clearing old print jobs, sending notifications, or generating periodic reports.

Logging with Morgan

Morgan is used for HTTP request logging, making it easier to monitor server activity and troubleshoot issues during development and production.

Environment Configuration with Dotenv

The **Dotenv** package is used to manage environment variables, ensuring that sensitive information like database credentials and API keys are kept secure and not hard-coded into the source code.

Concurrently & Nodemon for Development

- **Concurrently**: This package allows the front-end and back-end servers to run simultaneously during development with a single command, improving developer productivity.
- **Nodemon**: Automatically restarts the server when file changes are detected, making the development process faster and more efficient.

By leveraging the power of Node.js, Express.js, and MongoDB, combined with modern tools like Passport.js and Multer, the project achieves high performance, scalability, and security. This technology stack allows the Student Smart Printing Service to handle user authentication, file uploads, scheduled tasks, and real-time communication between the front-end and back-end efficiently.

2. Demo Application

2.1. Our Application

Figure 85: Application View of our Website (Image 1)

Figure 86: Application View of our Website (Image 2)

The above image has been viewed by browsing over Google Chrome with the Zoom ratio of 67% for viewing the full page.

This interface of the Student Smart Printing Services (HCMUT_SSPS) application is divided into four main sections, as described below:

Header Section

Elements:

- ***Logo:*** Positioned in the top-left corner, the logo represents Ho Chi Minh City University of Technology (HCMUT), ensuring clear branding for the application.
- ***Page Title***: The text "Student Smart Printing Services" is prominently displayed next to the logo, emphasizing the application's purpose.
- ***Navigation Bar:*** On the right side of the header, a Login button allows users to securely access their accounts.
- ***Background Design:*** A modern blue-themed background complements the university branding and enhances visual appeal.

Carousel/Introduction Section

This section displays a large carousel showcasing various images related to the university and its facilities, reflecting the academic environment.

Navigation Controls: Users can interact with left and right navigation arrows to browse through multiple slides within the carousel.

Services Section

Located below the carousel, this section introduces the application's key services, displayed as clickable cards.

Card Layout: Each card represents a specific service with an image and title. Examples of services might include "Document Printing," "Scheduling," and "Payment Options."

Interactive Elements: Clicking on a card navigates users to detailed pages explaining the respective service.

Footer Section

A footer includes additional links, copyright information, and contact details, offering easy navigation for users requiring support.

2.2. Login Page

For this Authentication in the Login part, we use **OAuth 2.0**, the **industry-standard protocol for authorization**.

Figure 87: Login Views of our Website

Logo Section: At the top-center of the page is the BK TP.HCM logo, representing Ho Chi Minh City University of Technology.

User Group Selection:

This section includes three distinct buttons for selecting the user group:

- Cán bộ/Sinh viên Trường ĐH Bách Khoa TPHCM: Designed for staff and students of HCMUT.
- Nhân viên dịch vụ in ấn sinh viên - SPSO: Tailored for printing service staff.
- Quản trị hệ thống/Người dùng khác: Allows access for administrators and other users.

Language Selector and Cookies Notice:

Users can choose their preferred language from the "Select Language" dropdown.

A Cookies Notice button ensures compliance with privacy policies, allowing users to learn about or configure their cookie preferences.

Figure 88: Views from the OAuth 2.0 of Google Authentication

Account Selection Prompt and Available Accounts display

A dark-themed Google OAuth 2.0 interface prompts the user to choose a Google account to continue with SSPS.

Two accounts are listed (as I have loginned them for testing before so they are listed here):

- Phú Nguyễn Quang with an institutional email.
- Phu Nguyen with a personal Gmail account.

Figure 89: Views from the OAuth 2.0 of Google Authentication (Continue)

Authorization Message:

This page confirms the Google account selected for login.

A message reads: "Đăng nhập vào SSPS" (Log in to SSPS), clarifying the intent of the action.

Action Buttons:

- *Tiếp tục (Continue):* Confirms the login process and authorizes the application.
- *Hủy (Cancel):* Allows users to back out of the process without proceeding.

2.3. Homepage View of Student vs SPSO

Figure 90: First Image is the Homepage View from SPSO and Second Image is the Homepage View from the Student

Now we go into the detailed description of the 2 Homepage Views of SPSO and Student.

First Image: Homepage View for SPSO (Student Printing Services Staff)

Header Section:

Logo and Title:

- The *BK TP.HCM* logo is prominently displayed on the left, ensuring clear institutional branding.

- The title, "Student Smart Printing Services," communicates the application's purpose.

Navigation Menu:

- The header includes navigational links: Homepage, Printer, Printing, Printing History, and User List, tailored for SPSO staff. This means that the SPSO staff can view the detailed information about the printers, printing configurations and also the list of users of the application.
- The User Icon on the far right indicates the logged-in user and provides access to profile management options.

Carousel Section:

Image Display: A rotating carousel displays images of the university campus and infrastructure.

Navigation Controls: Users can browse through multiple slides using the left and right navigation arrows.

Services Section:

- Below the carousel, the "Our Services" section features clickable cards representing various services offered by SSPS.
- Each card contains an image and title summarizing the service, such as printing options, payment integration, and support.

Second Image: Homepage View for Students

Header Section:

Logo and Title:

- Similar to the SPSO view, the *BK TP.HCM* logo is positioned on the left, maintaining consistency across user roles.

Navigation Menu:

- The header includes simplified navigational links specific to students: Homepage, Printing History, and Print Now.
- A profile picture icon on the far right allows students to access their profile details.

Carousel Section:

Image Display:

- Similar to the SPSO view, the rotating carousel presents images with the same message: "We are pleased to bring smart printing solutions to the School and the Students."

Navigation Controls:

- Students can also browse the carousel using the left and right arrows.

Services Section:

- The "Our Services" section showcases various options available to students, such as document uploading, real-time status tracking, and payment methods.

2.4. Profile Page of a User (Student and SPSO)

Figure 91: First Image is the Homepage View from Student and Second Image is the Homepage View from the SPSO

Here is the detailed description of the Profile Page interface for a user in the Student Smart Printing Services (SSPS) application:

User Name: Displays the user's full name (e.g., "Phu Nguyen") to identify the account holder.

Role: Indicates the user role in the system, such as:

- "Customer" for students.
- "Staff" or "Administrator" for SPSO staff or other roles.

Email: Provides the registered email address (e.g., quangphunguyen1804@gmail.com), which serves as the primary contact for notifications and account recovery.

Page Balance: Displays the user's current balance of printable pages (e.g., 352 pages), helping the user manage their printing quota.

2.5. Printing Actions from Student

2.5.1. List of Printers

Figure 92: Student View of List of Allowed Printers

The List of Printers page is organized into three main sections to provide students with an easy and intuitive way to select from the available printers.

Header Section

Logo and Title:

At the top-left corner, the BK TP.HCM logo and title "Student Smart Printing Services" reinforce institutional branding and the application's purpose.

Navigation Menu:

The header includes links for:

- *Homepage*: Returns to the main landing page.
- *Printing History:* Allows students to review previous printing jobs.
- *Print Now:* Highlights the current page where students can select a printer and begin a new task.

User Profile Icon:

The student's profile picture is displayed on the far-right, ensuring personalization and quick access to profile management.

Sidebar Navigation

Positioned on the left side of the page, the sidebar includes the following options:

- ***Printer List:*** Displays the current page where students can browse through a list of printers available to them.
- ***File List***: Redirects students to a page where they can manage their uploaded files.
- ***Buying Pages***: Provides access to purchase additional printing pages if required.

Printer Selection Section

Printer Cards:

Each available printer is displayed as a printer card with the following details:

- *Printer Number*: Unique identifier for the printer (e.g., M15W-1231).
- *Location*: Indicates the printer's physical location (e.g., CS1 - B4 - 101), ensuring students can find the device easily.
- *In Queue:* Displays the number of pending print jobs for that printer (e.g., 0), helping students select a printer with minimal wait times.
- *Description:* Provides additional information about the printer.
- Note that these Printers that the Student (User) can see here have been enabled by the SPSO.

Choose Button:

A blue "Choose" button next to each printer allows the student to select their preferred printer to proceed with the printing task.

2.5.2. Buying Pages

Figure 93: Buying Pages View

The Buying Pages View interface is organized into four main sections, providing students with a clear and efficient way to purchase additional printing pages.

Header Section

Logo and Title:

The *BK TP.HCM* logo and title "Student Smart Printing Services" are displayed at the top-left, ensuring consistency across the application.

Navigation Menu:

- *Homepage*: Directs users to the main landing page.
- *Printing History:* Allows users to review their past printing transactions.
- *Print Now:* Navigates users to start a new printing task.

User Profile Icon: The user's profile picture is displayed on the far-right for personalization and quick access to account details.

Sidebar Navigation

- *Printer List*: Redirects users to the list of available printers.
- *File List:* Allows users to manage their uploaded files.
- *Buying Pages*: Highlights the current page, making it clear that users are in the process of purchasing additional pages.

Main Content Section

Remaining Pages

- Displays the current balance of printing pages (e.g., 502 A4 pages) at the top of this section.
- Includes a thumbnail image of a stack of A4 papers and a note about the price per page (e.g., 200đ/A4 pages).

Quantity Selection

- Increment/Decrement Buttons: Allows users to specify the quantity of pages they want to purchase using "+" and "-" buttons for precision.
- Input Field: Displays the selected quantity, providing a visual confirmation of the order size.

Payment Methods

- Lists the available payment options with corresponding radio buttons for selection: BKPay, MoMo, ZaloPay, and VNPay. Each method is clearly labeled for easy identification.
- **Note**: Here we have not developed the real payment method function. Specifically, we just provide the user to buy any number of pages they need. We keep the real payment method like any other application for future advancement.

Promotion Package Section

Highlights promotional offers for bulk purchases:

- *100 A4 Pages*: Priced at 19,000đ with a 5% discount.
- *200 A4 Pages:* Priced at 36,000đ with a 10% discount.
- *500 A4 Pages:* Priced at 80,000đ with a 20% discount.
- *Choose Button:* Enables users to select a promotional package easily.

Order Section

- Displays a table summarizing the order details, including: Quantity, Rate, Amount.
- The Total field at the bottom dynamically updates based on the selected quantity or promotional package.
- *Buy Now Button*: Finalizes the purchase, taking the user to the payment confirmation screen.

2.5.3. Upload Files

Figure 94: Uploading Allowed Files (Image 1)

Figure 95: Uploading Allowed Files (Image 2)

The File Upload Dialog is a simple modal interface that allows users to upload their files for printing. It consists of the following sections:

File Selection Input

Choose Files Button:

- The button labeled "Choose Files" opens the file picker, allowing users to browse their local storage for files they wish to upload.
- The placeholder text, "No file chosen," dynamically updates to reflect the selected file names.

Input Field:

- Displays the names of selected files to confirm the user's choice before proceeding.
- Note that the User can upload one or multiple files here.

File Types:

- The system supports PDF and DOCX files by default, as defined by the SPSOs.

Action Buttons

- *OK Button:* Clicking the OK button confirms the file selection and uploads the files to the system.
- *Cancel Button*: we do not implement this but we allow user to click outside the File Upload Dialog to get out of that, which means users can exit the upload dialog without uploading any files.

Figure 96: View of File lists after being uploaded

The File List View displays all uploaded files with details and options for further actions. This page consists of the following sections:

Header Section

Navigation Menu:

- Links include Homepage, Printing History, and Print Now, providing seamless navigation.
- The User Profile Icon is displayed on the far right, ensuring personalization.

Sidebar Navigation

- *Printer List:* Redirects users to the available printers.
- *File List:* Highlights the current page for managing uploaded files.
- *Buying Pages:* Redirects to the page for purchasing additional printing pages.

Uploaded Files List

Each uploaded file is displayed in a file card format with the following details:

- *File Name*: Displays the file name (e.g., Task 1 (1.3) [Bản chính].docx.pdf).
- *Page Number:* Indicates the total number of pages in the file (e.g., 33 pages).
- *Printer Name:* Displays the assigned printer for this file (e.g., M15W-1231).
- *Queue Information*: Shows the current number of files in the printer's queue (e.g., 0).
- *Upload Status*: A green status bar confirms that the file has been successfully uploaded and verified (e.g., uploaded at 21/11/2024).

Action Buttons

- ***Print Button:*** Located next to each file, the Print button allows users to initiate the printing process for the corresponding file.
- ***Pagination Controls***: At the bottom of the list, pagination buttons (e.g., Page 1 of 1) enable navigation through multiple pages of uploaded files.

2.5.4. Configure Files for Printing

Figure 97: *Configuring Printed File Process (Image 1)*

Figure 98: *Configuring Printed File Process (Image 2)*

Figure 99: *Configuring Printed File Process (Image 3)*

Figure 100: *Configuring Printed File Process (Image 4)*

Figure 101: *Configuring Printed File Process (Image 5)*

Figure 102: *Configuring Printed File Process (Image 6)*

Here, we demonstrate the flow for configure and scheduled for a file to be printed.

Firstly, we will describe in detailed about the Printing Interface of the Configuration Files:

The Configure Files for Printing interface provides a detailed preview of the document and a comprehensive set of configuration options for customizing the print settings. This page is divided into two main sections.

Document Preview Section

Preview Pane:

- Displays a full-size preview of the document selected for printing.
- Users can view the content to ensure that the document is correct and formatted properly before proceeding.

Page Navigation:

Previous and Next Buttons:

- Located below the preview pane, these buttons allow users to navigate through the pages of the document (e.g., Page 1 of 33).
- Ensures users can review each page individually.

Purpose: This section ensures accuracy by allowing users to visually confirm the document content before finalizing print settings.

File Configuration Section

This section contains a set of customizable options that allow users to configure the printing parameters to meet their specific requirements.

Configuration Options:

Page Number: Dropdown menu allows users to select:

- All Pages: Prints the entire document.
- Custom Range: Prints a specific range of pages (e.g., 1-5).

Page Layout: Dropdown menu with options:

- Portrait: Standard vertical page orientation.
- Landscape: Horizontal page orientation for wide documents.

Page Size: Allows users to select the paper size:

- Options include A4, A5, Letter, and more.

Number of Pages Per Side: Field to set how many document pages will be printed on one side of the paper (e.g., 1, 2, or 4 pages).

Number of Sides: Dropdown with options:

- 1 Side: Single-sided printing.
- 2 Sides: Double-sided printing for paper efficiency.

Alignment: Specifies text alignment on the printed page (e.g., Default, Left, Center, Right).

Number of Copies: Field to set the desired number of copies to print (e.g., 1, 2, 3, etc.).

Confirm Button: The Confirm button saves the selected settings and proceeds to the final printing step. Ensures that all configurations are applied before initiating the print job.

Secondly, we now describe what those images we include in this 2.5.4 section means:

- Students select the file that they currently upload and then they can preview the file. They can click on Previous and Next to move back to the previous and move forward to the next page.
- Then they have multiple configured options to select. Here we keep everything by default except for "Num Side" to be 1 and "Number of Copies" to be 3. As this file has 33 pages, when we configure the file like above, it will cost us 99 pages of A4 to print the document. Then we choose Confirm.
- As I have said, the screen now shows our Page Balance and Page Needed for printing this document. Then we just need to select the time for the file to be printed and click Ok. If you do not have enough pageBalance for printed, you may receive an error. Else you can continue viewing your document in the queue.

2.5.5. View Files that is scheduled for Printing

Figure 103: View File Scheduled for Printing (Image 1)

Figure 104: View File Scheduled for Printing (Image 2)

Figure 105: View File Scheduled for Printing (Image 3)

Now we will describe in detail about these interfaces we should consider while viewing the printed files.

First and Third Image: Printing History Interface

The Printing History interface provides users with a comprehensive overview of their scheduled and completed printing jobs. This page is divided into the following sections:

Header Section

Logo and Title: The BK TP.HCM logo and title "Student Smart Printing Services" are displayed prominently at the top for branding and context.

Navigation Menu:

Links include:

- Homepage: Returns to the main landing page.
- Printing History: Highlights the current page for tracking print jobs.
- Print Now: Directs users to initiate a new print job.

User Profile Icon: Displays the user's profile picture on the right, ensuring a personalized experience.

Filter Options (Sidebar Menu)

Provides quick filters for viewing different categories of print jobs:

- *All*: Displays all print jobs.
- *Arranged*: Shows jobs that are queued and ready for printing.
- *Canceled*: Lists jobs that were canceled by the user or system.
- *Printed*: Displays completed print jobs.

Printing History Cards

Each job is represented as a card with the following details:

- *File Name*: Displays the document name (e.g., Task 1 (1.3) [Bản chính].docx.pdf).
- *Page Number*: Shows the total number of pages in the file (e.g., 33 pages).
- *Printer Name*: Indicates the assigned printer (e.g., CS1 - B4 - 101).
- *Queued*: Displays a gray bar with a timestamp. This is the one that is still not the time for it to be printed (the time here has been selected in the configuring steps).
- *Canceled*: Highlights canceled jobs with a red bar and timestamp. This is the one that has been printed completely.
- *Completed*: The green bar and timestamp is for the file that is queued already, if Student chooses to cancel the printing process, the file will be canceled. BUT the student still loses the number of pages that are needed for printing that document (we define this as a PENALTY for students to think carefully before selecting Confirm to print the document).
- *Copies*: Shows the number of copies requested.
- *Details Button*: Opens a detailed view of the print job.

Second Image: Printing Job Details Dialog

The Details Dialog provides additional information about a specific printing job.

Details Displayed:

- *Pages to Be Printed:* Specifies the page range (e.g., all).
- *Page Layout:* Indicates the layout (e.g., portrait).
- *Paper Size*: Displays the size of the paper (e.g., A4).
- *Pages Per Side:* Shows how many pages are printed on one side (e.g., 1).
- *Double Sided*: Indicates whether the job is double-sided (e.g., one).
- *Margin Settings*: Details the margins (e.g., Top: 1, Bottom: 1, Left: 1, Right: 1).
- *Number of Copies*: Indicates the number of copies requested (e.g., 3).

- *Take Date*: Displays the scheduled print date and time (e.g., 22/11/2024, 1:33:15 AM).

Action Buttons:

- *Cancel Button:* Allows the user to cancel the job.
- *OK Button*: Closes the dialog.

2.6. Printing Management from SPSO

2.6.1. View Current List of Printers

Figure 106: View of all the printers the school currently has (Image 1)

Figure 107: View of all the printers the school currently has (Image 2)

Here we will describe more specifically about the printing management processes.

SPSO can view the list of all printers that it currently has in the school. Also, they can view the detailed information of each printer and can edit them too.

From the 3 above images, we want to demonstrate that a SPSO can modify the field of a printer via the Printer Information Dialog. Here is the description about the sections that is included in the Printer Information Dialog:

Editable Fields:

- Brand/Model Selector: Dropdown menu for selecting the printer's brand or model (e.g., HP).
- Location: Input field for updating the printer's location (e.g., CS1 - B4 - 101).
- Identifier: Field for editing the unique printer number (e.g., M15W-1231).
- Description: Allows editing of the printer's description for clarity.

Action Buttons:

- Save Button: Applies the changes made to the printer's information.
- Cancel Button: Closes the dialog without saving changes.

2.6.2. Enable the Printer

Figure 108: Enable and Disable the Printer (Image 1)

Figure 109: Enable and Disable the Printer (Image 2)

Figure 110: Enable and Disable the Printer (Image 3)

Figure 111: Enable and Disable the Printer (Image 4)

From these 4 images, we want to demonstrate that when an SPSO disables a printer, then it will disappear in the list of available printers for choosing on the view of the Student.

- The *first 2 images* show that when the Printer with the Number of M15W-1231 is enabled, the Student can see that in its List of Printers for choosing to print.
- The *last 2 images* show when the Printer has been disabled, the Student cannot see it anymore.

2.6.3. Add a new Printer

Figure 112: Add a new Printer (Image 1)

Figure 113: Add a new Printer (Image 2)

Firstly, these 2 images show that we have successfully added a new printer with the number of A1111A-11111.

Secondly, in this 2.6.3, we will show the interfaces that appear in the process of adding a new printer.

Sidebar Navigation

- *Printer List:* A button that redirects SPSO to the main list of currently available printers.
- *Add More Printers:* Highlights the current page, indicating that the user is in the process of adding a new printer.

Printer Configuration Form

The central area of the interface is dedicated to configuring the new printer with multiple fields to ensure all relevant details are captured.

Location Section

- Field Name: Location (Campus - Building - Room)
- Input Field: Allows the SPSO to specify the exact location of the printer (e.g., CS1 - B1 - 101). This ensures users know where to physically find the printer.

Printer ID

- Field Name: Printer ID
- Input Field: Enables the SPSO to assign a unique identifier to the printer (e.g., A1111A-11111), which is essential for distinguishing between multiple printers.

Printer Brand

- Field Name: Brand
- Dropdown Menu: Allows SPSO to select the brand of the printer (e.g., Canon). This helps categorize printers for maintenance or compatibility purposes.

Printer Description

- Field Name: Description
- Text Area: Provides space for SPSO to add a detailed description of the printer (e.g., This Printer is added for Testing the Add and Delete Printer Function of our Application). This description can include unique features or intended use cases for the printer.

Action Buttons

- *Confirm Button:* A blue Confirm button finalizes the addition of the new printer to the system. Clicking this button saves all entered information and makes the printer available in the system.
- *Cancel Button (Optional)*: Though not visible in this design, a Cancel button is typically available to discard the input and return to the main printer list.

2.6.4. Delete a Printer

Figure 114: Delete a Printer (Image 1)

Figure 115: Delete a Printer (Image 2)

This part contains the exact same interfaces as described in those last parts except for the Confirmation Deletion Dialog, which we want to ask the SPSO to be sure of.

2.6.5. Printing Configuration: Configure the File Type

Figure 116: Configure the Type for Uploading and Printing (Image 1)

Figure 117: Configure the Type for Uploading and Printing (Image 2)

Figure 118: Configure the Type for Uploading and Printing (Image 3)

Figure 119: Configure the Type for Uploading and Printing (Image 4)

Figure 120: Configure the Type for Uploading and Printing (Image 5)

Figure 121: Configure the Type for Uploading and Printing (Image 6)

This part (2.6.5), we demonstrate the way to configure the allowed file type for uploading and printing. But first let's dive into the way we design our included interfaces in the process.

First Image: File Type Configuration Interface

The File Type Configuration interface allows the SPSO (Student Printing Service Operator) to manage and define the file types supported by the printing service. The page consists of three main sections:

Header Section

Logo and Title: Displays the BK TP.HCM logo and the title "Student Smart Printing Services" at the top-left corner for branding consistency.

Navigation Menu: Includes the following links:

- *Homepage:* Redirects to the main dashboard.
- *Printer:* Links to printer management.
- *Printing:* Redirects to current printing operations.
- *Printing History:* Allows access to past print job records.

User List: Provides access to the list of registered users.

User Profile Icon: Displays the profile picture of the logged-in SPSO for personalization.

Sidebar Navigation

- *Configure File Type*: Highlights the current page, where the SPSO can manage file types.
- *Giving Pages*: Links to a page where SPSO can allocate printing pages to users.
- *Export Report*: Provides a link to download printing activity reports.

File Type Options

- Displays a set of large, visually distinct cards for the supported file types:
- Add More File Types: A large "+" card allows SPSO to add new file types.

Second Image: Add or Configure File Types

The Add File Type Dialog allows SPSO to specify additional file types supported by the system.

Dropdown Menu

- *Label*: Enter the Type You Want
- *Options*: Provides a list of file types to select from, including: jpg, msword, doc, docx, pdf, png, pptx, svg, text, xls

Now, we will describe what the 5 given images above want to demonstrate: By let the SPSO adds a new type for uploading file (here it is pptx), the Student can now upload his pptx file and can completely print it and view the successful printed pptx file in the PRINTING HISTORY part.

2.6.6. Printing Configuration: Giving Pages

Figure 122: Configure the Type for Uploading and Printing (Image 1)

Figure 123: Configure the Type for Uploading and Printing (Image 2)

```
[0] It's time to distribute pages!
[0] UpdateMany result: {
[0]   acknowledged: true,
[0]   modifiedCount: 3,
[0]   upsertedId: null,
[0]   upsertedCount: 0,
[0]   matchedCount: 3
[0] }
[0] Added 100 pages to all users' pageBalance.
```

Figure 124: Configure the Type for Uploading and Printing (Image 3)

Figure 125: Configure the Type for Uploading and Printing (Image 4)

Before explaining the meaning of those 3 images, we will describe our main design for the interfaces first.

First and Fourth Images: User List

This interface consists of individual user cards, each displaying detailed information and action buttons for the corresponding user.

User Information:

- *Name*: Displays the full name of the user (e.g., Phu Nguyen).
- *Email*: Provides the email address of the user for communication and identification (e.g., quangphunguyen1804@gmail.com).
- *Page Balance*: Indicates the remaining number of printable pages in the user's account (e.g., 610 pages).
- *Page Used:* Shows the total pages the user has printed so far (e.g., 0 pages).
- *Role*: Displays the user's role in the system:

+ *SPSO*: For operators managing the system.
+ *Customer*: For standard users accessing the printing services.

Special Indicators:

The card for the logged-in user includes a "(You)" tag to distinguish it from other accounts. It also states, "You do not have permission to edit this user," preventing the user from making changes to their own role.

Action Buttons:

Edit Role Button: Allows SPSO to change the role of other users (e.g., from Customer to SPSO) for better access control. Detailed about the use of this function we will say in part 2.7.

Second Image: Manage Printing - Giving Pages

Sidebar Navigation

- *Configure File Type:* Redirects to the page for managing file formats supported by the printing system.
- *Giving Pages:* Highlights the current page, indicating that the operator is configuring free page allocation for users.
- *Export Report:* Links to a report generation feature for printing activity summaries.

Page Allocation Configuration Section

This section contains fields and options for configuring the allocation of free printing pages.

Date Selection

- Label: Choose date for giving pages.
- Input Field: Allows the SPSO to select the specific date for the free pages to be added to user accounts (e.g., 11/22/2024).

Time Selection

- Label: Choose time for giving pages.
- Input Field: Specifies the exact time when the free pages will be added (e.g., 02:14 AM).

Number of Pages

- Label: Number of free given pages.
- Input Field: Enables the SPSO to set the number of free pages to allocate to users (e.g., 100 pages).

Action Button

Confirm Button: A blue Confirm button allows the SPSO to save and finalize the allocation settings. Clicking this button schedules the page allocation process.

Now we will talk about the meaning of those 4 given images. We want to demonstrate that SPSO can select a specific date for giving free pages to all the users. From the images we can see that:

- In the first image, at 2:11 AM (22/11/2024), the 3 users in turn have 502, 100, and 501 pages.
- In the second image, the SPSO user has chosen the date and time for giving free pages is 2:14 AM (22/11/2024) and he enters 100 which means that all the users will freely receive this amount of free pages at the given time.
- The third image shows that it is the time for distributing the free pages (2:14 AM - 22/11/2024) and all the users will receive 100 free pages.
- In the fourth image, the 3 users sequentially get 602, 200, and 601 pages, which means that SPSO has successfully given the free pages to all the Users and Users successfully get that.

2.6.7. Printing Configuration: Export Report

Figure 126: Export the Report and Statistic Data of the Application (Image 1)

Figure 127: *Export the Report and Statistic Data of the Application (Image 2)*

Figure 128: *Export the Report and Statistic Data of the Application (Image 3)*

Figure 129: *Export the Report and Statistic Data of the Application (Image 4)*

Figure 130: Export the Report and Statistic Data of the Application (Image 5)

Because we do not have enough accounts and a long period of time (months) for getting all the data, so for only this part of the exporting report, we use static data for visualization. For all the other parts of this project, we store the information in the database.

Image 1: Revenue Growth Chart

This section provides a Revenue Growth Chart, visualizing the monthly progression of revenue generated from sold printing pages. It consists of 2 main elements:

Header Section

- The *logo and title*, "Student Smart Printing Services", ensure branding consistency.
- *Navigation links*, including Homepage, Printer, Printing, Printing History, and User List, provide seamless access to other functionalities.
- The *Export Report option* is highlighted in the sidebar, indicating the current page's purpose.

Revenue Visualization

- The Revenue Growth Chart displays a line graph illustrating the increase in revenue across months.
- Data points (e.g., April: 50, December: 1200) are marked for clarity.
- *Export Button*

⇒ A prominent Export button allows the SPSO to download this report in PDF or other formats for administrative purposes.

Image 2: New Users Statistics

This section provides a scatter plot visualizing the number of new users who used the service each month.

Header Section

Similar to Image 1, it contains the navigation and sidebar highlighting the Export Report feature.

New User Visualization

- The scatter plot displays the count of first-time users for each month.
- Points such as April (40 users) and December (140 users) are plotted along the timeline for an easy-to-read trend.

Image 3: Printer and User Printing Statistics

This section provides detailed statistics on printer usage and user printing activity.

Printer Usage

- A bar chart displays the number of pages printed by each printer.
- Example data: Printer 1: 500 pages, Printer 10: 1000 pages.
- This allows the SPSO to identify which printers are utilized most frequently.

User Printing Distribution

- A pie chart breaks down the percentage of pages printed by individual users.
- Example data:

  User 1: 18.2%

  User 10: 9.1%

⇒ This helps track individual contributions to total printing.

Image 4: Exporting the Report

This image demonstrates the Export Report feature and configuration options for exporting data.

Data Overview

- Displays all charts, including revenue growth, new users, and printer/user usage.

Export Configuration: The Print dialog on the right provides options for:

- Destination (e.g., Microsoft Print to PDF).
- Page selection (e.g., All pages).
- Layout (e.g., Portrait or Landscape).

- Color settings (e.g., Color or Grayscale).

Export Button: Allows the SPSO to generate and save the report.

2.7. Extension 1: SPSO can View User List and Specific SPSO can edit the Role

Figure 131: List of Users using the Application

Here is the things we have done and want to demonstrate as an **extension for this project:**

- SPSO has the ability to **view all the Users** who are currently enrolled. Students cannot view this.
- **Only a specific SPSO user has the ability to change the Role** of other Users: Here PHÚ NGUYỄN QUANG is that one. He has changed the Role of Phu Nguyen from

Customer to SPSO. But then Phu Nguyen cannot change the role of the other user although he is an SPSO user.

2.8. Extension 2: Rush hour Solution

```
async function start() {
  console.log("Cron job started...");

  const timeZone = "Asia/Ho_Chi_Minh"; // Desired time zone

  cron.schedule("* * * * *", async function () {
    console.log("Running scheduled tasks...");

    const now = new Date(); // Current time in UTC
    const localNow = convertToTimeZone(now, timeZone); // Convert to local time using time zone

    // console.log("Current Time (UTC):", now.toISOString());
    console.log("Current Time (Local):", localNow.toISOString());

    // Task 1: Add load-balancing printers during peak hours
    let peakHoursStart = new Date();
    peakHoursStart.setHours(3, 21, 0); // Set peak hours start time (3:19 AM local time)
    peakHoursStart = convertToTimeZone(peakHoursStart, timeZone);

    let peakHoursEnd = new Date(peakHoursStart.getTime() + 3 * 60 * 1000);      You, 9 seconds a
    peakHoursEnd = convertToTimeZone(peakHoursEnd, timeZone);
```

```
[0] Peak hours active: Adding load-balancing printers...
```

Figure 132: Solution to Rush Hours (Image 1)

Figure 133: Solution to Rush Hours (Image 2)

It is practical that there will be **some main time of the days** that Students and Teachers may use the Printers more than the other time, we call them **peak hours or rush hours.**

So our team solution is that we will **prepare another 5 printers** that are **only used for the case when rush hours happen.** Then after a specific time, these printers would then be brought back and keep them "**sleep**" (In the code we just demonstrate for the time range of 3 minutes, which means for demonstration, after 3 minutes of the rush hours, these printers will be removed). Also note that during the rush hours, these printers cannot be touched, which means that they cannot be disabled or deleted and by default they would be enabled for use in the rush hours, which is realistic.

2.9. Extension 3: Provide the UI for Loading and 404 - Error Page

Figure 134: UI for the Loading and 404 - Error Page

2.10. Summary of Things that we have done and still not done

Summary of Accomplishments

In this project, we developed a comprehensive demo application for managing student printing activities within a university. The application provides distinct functionalities tailored for students and the **Student Printing Service Operator (SPSO).** The Login Page ensures secure access and role-based redirection for users. For students, the Homepage View offers intuitive access to **printing-related actions**, while SPSOs can efficiently **manage the printer network.**

Students can perform various **Printing Actions**, such as viewing available printers, purchasing pages, uploading files, configuring printing options, and managing files scheduled for printing. From the SPSO perspective, the application includes robust **Printing Management features**, such as viewing the current list of printers, enabling or disabling them, adding or deleting printers, and configuring file types for printing. **Additional features** allow SPSOs to **allocate pages** to students and **export reports** for administrative purposes.

The project also includes **several extensions**: SPSOs can view a user list and edit roles, implement a rush-hour solution to streamline printing during peak times, and enjoy a polished user experience through a **loading screen** and a **custom 404 error page**. These functionalities collectively make the application a strong demonstration of the envisioned system.

Limitations and Future Work

Despite its strengths, the current application remains a demo with certain limitations. The system does **not integrate with real payment methods** or APIs, relying instead on placeholders for purchasing pages. SPSO-generated system reports currently **display static data** for visualization rather than dynamic data from the database. The application is designed for demonstration purposes, so it operates with a **small dataset** and does not handle a large volume of data.

The database schema is also relatively simple, with **basic models and limited relationships**. Future development should focus on creating a more sophisticated schema to support complex data interactions. Expanding the dataset, integrating real payment systems, and connecting dynamic data for reporting will enhance the application's utility and scalability. These improvements will transform the demo into a robust, fully functional system suitable for real-world deployment.

3. Presentation

Presentation Slide Link: https://shorturl.at/5Fd2P (Link to our Canva Presentation)

Presentation and Demo Video Link: https://shorturl.at/vd8Q9 (Video on Google Drive)

Presentation and Demo Video Link: https://shorturl.at/5Qg6v (Video on Youtube)

VI. Lessons and Conclusion

The development of the **Student Smart Printing Service (SSPS)** project has been a long, difficult but fun journey, providing our team with **substantial knowledge and hands-on experience** in building a comprehensive software solution. Throughout the project, we gained key insights into every stage of the software engineering lifecycle, from requirement elicitation and system modeling to implementation and testing, all while honing our technical and collaborative skills.

In the **requirement elicitation phase**, we learned the importance of deeply understanding the domain context and actively engaging with **stakeholders** to align the system's goals with user needs. Defining clear **functional and non-functional requirements** not only clarified the application's objectives but also helped us foresee potential interactions and edge cases through detailed **use-case scenarios.** This planning phase emphasized the value of meticulous preparation before diving into development.

The **system modeling stage** provided an invaluable opportunity to visualize system behavior and interactions. By creating **activity, sequence, and class diagrams**, we mapped out workflows for modules like authentication, printing, and printer management. These models served as a critical foundation, guiding development while ensuring team alignment on the system's structure and functionality. Even though the initial database schema was kept simple, the modeling process highlighted areas for future scalability and complexity.

During the **implementation phase**, the choice of **modern technologies** played a pivotal role in the project's success. ReactJS enabled the creation of a responsive and modular user interface, while tools like Redux Toolkit and React Router simplified state management and navigation. TailwindCSS, along with Material Tailwind and Flowbite, accelerated UI styling and customization. On the backend, Node.js and Express.js provided an efficient framework, and MongoDB offered the flexibility needed for data storage. The integration of Passport.js for secure authentication and Multer for file uploads further strengthened the system's robustness.

The **iterative development approach** was particularly insightful, as it underscored the importance of **continuous improvement. Usability testing** conducted after the first sprint revealed actionable feedback on interfaces, such as clearer workflows and improved layouts. Addressing these issues in subsequent iterations not only enhanced the user experience but also reinforced the value of user-centered design in refining a system. This iterative process also allowed us to manage timelines effectively and deliver a functional demo within the given constraints.

However, the project also highlighted areas for **improvement and future development.** Features such as **real payment method integration** and **dynamic reporting** were not fully implemented, with placeholders currently serving those functionalities. Additionally, our

database schema needs further refinement to handle complex relationships and larger datasets, a challenge we aim to address in future iterations. The system's modular architecture, while instrumental in maintaining code clarity and scalability, also required careful balancing between simplicity and long-term flexibility.

Beyond technical lessons, the project was an excellent exercise in **teamwork and project management**. Overcoming challenges like **task allocation and team coordination** taught us the importance of clear communication, efficient division of labor, and conflict resolution. **Regular team meetings and collaborative strategies** not only improved our productivity but also strengthened our cohesion as a team.

In conclusion, the SSPS project has been a **comprehensive learning experience**, enriching both our technical expertise and soft skills. From leveraging tools like Figma for wireframing and GitHub for version control to developing user interfaces and conducting usability testing, we gained a holistic understanding of software development best practices. We are particularly grateful for the guidance and feedback from our instructors, which significantly influenced the quality of our final product. This project has equipped us with the skills, knowledge, and confidence to tackle larger and more complex software engineering challenges in the future.

VII. References

[1] Australian Nursing & Midwifery Federation. (n.d.). *What is Activity Diagram?* Visual Paradigm. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/

[2] Australian Nursing & Midwifery Federation. (n.d.). *What is Sequence Diagram?* Visual Paradigm. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/

[3] Australian Nursing & Midwifery Federation. (n.d.). *What is Class Diagram?* Visual Paradigm. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/

[5] GeeksforGeeks. (2024, Oct 11.). *MVC Design Pattern*. GeeksforGeeks. https://www.geeksforgeeks.org/mvc-design-pattern/

[6] Lucidchart. (n.d.). Architecture diagrams. https://www.lucidchart.com/pages/architecture-diagram

[7] Oracle Corporation. (n.d.). *MySQL*. https://www.mysql.com/

[8] MongoDB Inc. (n.d.). *MongoDB*. https://www.mongodb.com/

[9] MongoDB Inc. (n.d.). *ACID transactions*. https://www.mongodb.com/docs/manual/core/transactions/

[10] Representational State Transfer. (n.d.). *RESTful standards*. https://restfulapi.net/

[11] Auth0. (n.d.). *JSON Web Token (JWT)*. JWT.io. https://jwt.io/

[12] Auth0. (n.d.). *Authentication and Authorization Flows*. Auth0. https://auth0.com/docs/get-started/authentication-and-authorization-flow/

[13] Meta Platforms, Inc. (n.d.). *React*. React Documentation. https://react.dev/

[14] Mozilla Foundation. (n.d.). *Single Page Application (SPA)*. https://developer.mozilla.org/en-US/docs/Glossary/SPA

[15] Mozilla Foundation. (n.d.). *DOM (Document Object Model)*. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

[16] React. (n.d.). *Virtual DOM and Internals*. https://legacy.reactjs.org/docs/faq-internals.html

[17] Vite. (n.d.). *Vite*. https://vite.dev/

[18] (n.d.). *esbuild: An extremely fast bundler for the web*. https://esbuild.github.io/

[19] Tailwind Labs. (n.d.). *Tailwind CSS*. https://tailwindcss.com/

[20] Material Tailwind. (n.d.). *The Tailwind CSS Components Library For Coding 10x Faster*. https://www.material-tailwind.com/

[21] Flowbite. (n.d.). *Build websites even faster with components on top of Tailwind CSS.* https://flowbite.com/

[22] (n.d.). *React Router Documentation.* https://reactrouter.com/

[23] Redux Toolkit. (n.d.). *The official, opinionated, batteries-included toolset for efficient Redux development.* https://redux-toolkit.js.org/

[24] ApexCharts. (n.d.). *React Charts*. https://apexcharts.com/docs/react-charts/

[25] React-PDF. (n.d.). *React renderer for creating PDF files on the browser and server*. https://react-pdf.org/

[26] PDF-Lib. (n.d.). *PDF-Lib Documentation*. https://pdf-lib.js.org/

[27] Moment.js. (n.d.). *Parse, validate, manipulate, and display dates and times in JavaScript*. https://momentjs.com/

[28] Prettier. (n.d.). *Prettier Documentation*. https://prettier.io/

[29] Node.js Foundation. (n.d.). *Run JavaScript Everywhere*. Node.js. https://nodejs.org/

[30] Express.js. (n.d.). *Fast, unopinionated, minimalist web framework for Node.js*. https://expressjs.com/

[31] MongoDB Inc. (n.d.). *Understanding SQL vs NoSQL Databases*. https://www.mongodb.com/resources/basics/databases/nosql-explained/nosql-vs-sql

[32] MongoDB Atlas. (n.d.). *MongoDB Atlas*. https://www.mongodb.com/atlas

[33] Mongoose. (n.d.). *Elegant MongoDB object modeling for Node.js.* https://mongoosejs.com/

[34] MongoDB Inc. (n.d.). *MongoDB CRUD Operations*. https://www.mongodb.com/docs/manual/crud/

[35] Passport.js. (n.d.). *Simple, unobtrusive authentication for Node.js*. http://www.passportjs.org/

[36] Helmet.js. (n.d.). *Helmet*. https://helmetjs.github.io/

[37] Express CORS. (n.d.). *cors*. https://expressjs.com/en/resources/middleware/cors.html

[38] Multer. (n.d.). *Node.js middleware for handling multipart/form-data.* https://github.com/expressjs/multer

[39] Node-cron. (n.d.). Node-cron. Node-cron Documentation. https://github.com/node-cron/node-cron

[40] Morgan. (n.d.). *HTTP request logger middleware for node.js*. https://github.com/expressjs/morgan

[41] Dotenv. (n.d.). *Loads environment variables from .env for node.js projects.*. https://github.com/motdotla/dotenv

[42] Nodemon. (n.d.). *nodemon reload, automatically.* https://nodemon.io/