Software Requirements Specification

for

A smart printing service for students at HCMUT

Version 2.0 approved

Prepared by:

1. **Nguyễn Trương Thái Bảo – 2210251**
2. **Nguyễn Truyền Tuấn – 2213795**
3. **Nguyễn Tuấn Huy – 2211253**
4. **Nguyễn Văn Nhật Huy – 2211254**
5. **Phạm Bá Nhật Khang – 2211460**

Department of Software Engineering
Faculty of Computer Science and Engineering
Ho Chi Minh City University of Technology – VNU-HCM

17/9/2024

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
All members	22/09/2024	Completed Requirement elicitation (Task 1.1, 1.2)	1.0
All members	29/09/2024	Completed Use-case Diagrams (Task 1.3)	1.0
All members	19/10/2024	Completed System Modeling (Task 2)	1.0
All members	08/11/2024	Completed Architecture Design (Task 3)	1.0
All members	01/12/2024	Completed Implementation – Sprint 1 (Task 4)	1.0
All members	21/12/2024	Revised previous tasks and Completed Implementation – Sprint 2 (Task 5)	2.0

1. Task 1: Requirement elicitation (1.1, 1.2)

1.1 Domain Context

The Ho Chi Minh City University of Technology (HCMUT) is developing the HCMUT Student Smart Printing Service (HCMUT-SSPS) to provide a convenient and efficient solution for students to print their documents across various campuses. The service integrates multiple printers, a web-based application, a mobile app, and a management system, streamlining the printing process, reducing costs, and enhancing the overall student experience. Aimed at improving academic services and convenience, HCMUT-SSPS will enable students to print documents quickly and efficiently for their academic and research needs, offering a fast, time-saving printing service within the campus.

1.2 Stakeholders and Needs

- **HCMUT Students (Users):** As the primary users, students need a convenient, fast, and reliable way to print their academic documents across different campuses. They require easy access to printers through a web or mobile app, the ability to upload documents, choose printer settings, and track their printing history. Students also need a system that manages their printing quotas and allows for seamless online payments when additional pages are needed.
- **Student Printing Service Officers (SPSO):** These administrators manage the printing system, including the configuration of printers and student accounts. They need access to logs of student print jobs, the ability to adjust settings like allowed file types, manage printer statuses, and oversee the allocation of printing quotas. They also require automatic generation of reports on system usage for administrative purposes.
- **HCMUT Managers:** The university leadership requires the system to improve overall student services, enhance campus efficiency, and optimize resource usage. They need the system to be cost-effective, reduce administrative burden, and provide insights into printing system usage to make informed decisions regarding resource allocation and improvements.

1.3 Benefits of the System

- **For HCMUT Students:** The system provides a fast, convenient, and efficient way to print academic documents, saving time by allowing them to select printers and

configure print settings directly through a web or mobile app. It also helps them manage their printing quotas easily and make online payments when necessary. Furthermore, students can track their printing history and usage, ensuring transparency and control over their print jobs.

- **For Student Printing Service Officers (SPSOs):** The system simplifies the management of printers and printing services. SPSOs benefit from having centralized control over printer configurations, monitoring student print activity, and accessing detailed reports on system usage. This automation reduces manual work and enables better service oversight and resource management.
- **For HCMUT Managers:** The system enhances overall campus service efficiency, providing a streamlined printing solution that reduces costs and administrative overhead. It also helps the university improve the student experience by offering a modern and user-friendly service, while the data collected from the system can be used to optimize printer distribution and resource allocation, ultimately enhancing campus operations

1.4 Functional Requirements

- **Students:**
    - Students are able to upload document files onto the system.
    - Students can choose which printer they want to print documents.
    - Students are allowed to customize printing properties such as paper size, number of printing pages, one-/doubled-sized, number of copies, etc.
    - Students can view their printing log for time period and information about the number of printed pages for each page size.
    - Students can buy more printing pages through the Buy Printing Pages feature of the system.
    - Students are able to pay for buying more printing pages through an online payment system like the BKPay system of the university.
    - A student will be not allowed to print if the printing action exceeds his/her account balance.
- **Student Printing Service Officer (SPSO):**
    - SPSO can configure file types allowed to be printed.
    - SPSO can modify the default number of pages.
    - SPSO can set the dates to update the default number of pages to all students.

- SPSO is able to view the printing log of all students or a student for specified time (date to date) and for some printers or all of them.
- SPSO can add a printer to the system.
- SPSO can disable or enable a particular printer.
- SPSO is able to view the activity reports of the system.

- **General requirements:**
  - The system should log all the printing actions (student ID, printer ID, file name, printing start and end time, number of pages for each page size) of all students.
  - The system should automatically generate activity reports at the end of each month and each year.
  - The reports should be stored in the system.
  - All users must be authenticated by HCMUT_SSO authentication before using the system.

1.5 Non- Functional Requirements

- **Speed:**
  - The system must ensure a response time of no more than 3 seconds for file uploads, submitting print requests, and printer selection.
  - Printing logs and history must be displayed to users within 3 seconds of their request.
  - Monthly and yearly reports must be automatically generated at the end of each period and should be accessible for viewing within 5 seconds upon request by the SPSO.
  - The system must be capable of handling at least 1,000 concurrent users without performance degradation.
- **Size:**
  - Each document uploaded must not exceed 100 MB in size, with total system storage scalable to 50 terabytes of documents and logs.
  - The system must be able to store printing logs for each student for at least 4 years (or until they graduate) without performance degradation.
- **Ease of Use:**
  - Users and SPSO can access the web-based app at any time.

- The web-based app must adapt its user interface based on the screen size and orientation of the device.
- The web-based app must function seamlessly across popular web browsers (Chrome, Safari, Firefox, and Edge) without requiring additional plugins.

- **Reliability:**
  - The printing service should be available from 7:00 to 20:00 on every working day (Monday to Friday) and from 7:30 to 17:00 on Saturday.
  - The system must operate properly (no errors, no crashes, no failures) during working days from Monday to Saturday. If there is an error from the printers, it will automatically try to restart up to three times.
- **Robustness:**
  - The system must maintain data integrity and accuracy in logging student activities and printer usage.
  - The system must back up data regularly, with a backup schedule of at least once daily to prevent data loss.
- **Portability:**
  - Printers must be placed in each faculty office and in the library.
  - Each faculty office must have at least one printer, and the library must have at least five.
  - The web-based app must be designed to be accessible from a variety of devices, including desktops, laptops, tablets, =and smartphones.

2. Use-case Diagrams (1.3)

2.1 Use-case Diagram for the Whole System

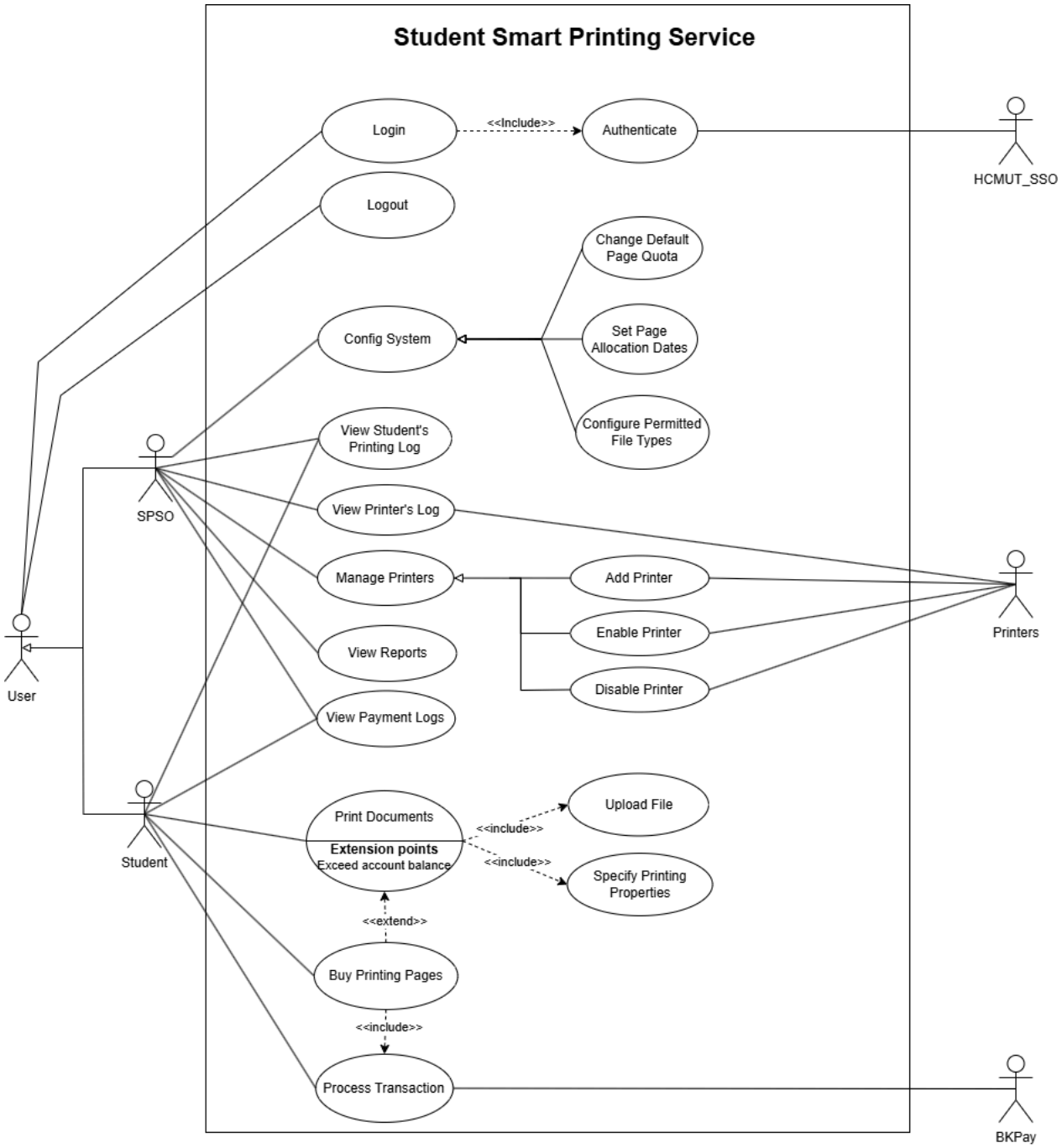

Figure 1. Use-case Diagram for the Whole System

2.2 Use-case Diagram for Printing Module

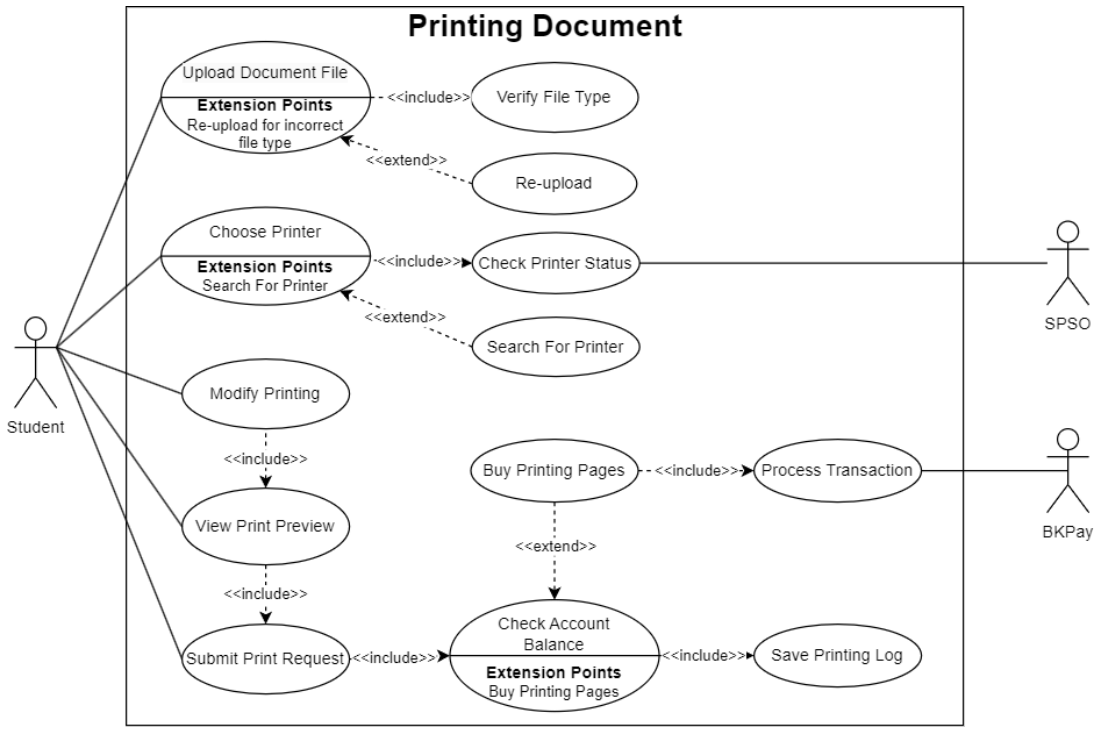

Figure 2. Use-case Diagram for Printing Module

2.3 The Details of Usecases in Printing Document Module

1. Usecase Upload Document File

Use Case ID	UC-PD01		
Use-case name	**Upload Document File**		
Created by	Tuấn Huy	**Last Updated By**	Nhật Huy
Date Created	25/9/2024	**Date Last Updated**	29/9/2024
Actor	Students		
Trigger	Student want to upload document file for printing		
Description	Student select file(s) on his device and upload them to the system for printing		

Preconditions	- The student has logged into the system - The student's device can connect to the internet and the system
Postconditions	A completely new file is uploaded to the system
Normal Flow	1) The student selects the "Upload file" button 2) The system opens the interface to select the file to upload 3) The student selects the file to upload 4) The student clicks the confirm button
Alternative Flows	1) In step 2.1, the student can access the web interface provided by the printing system and upload files directly from their computer or cloud storage (e.g., Dropbox, Google Drive, etc.) 2) In step 2.2, the student can drag and drop the file into the designated area on the web interface for a quick upload
Exceptions	Exception 1: At step 5, the file upload fails because the file size is too big

2. Usecase Re-upload

Use Case ID	UC-PD02		
Use-case name	**Re-upload**		
Created By	Tuấn Huy	**Last Updated By**	Nhật Huy
Date Created	25/9/2024	**Date Last Update**	29/9/2024
Actor	Students		
Trigger	The student wants to re-upload the desired file. Or the system requires re-uploading the file when he uploads the incorrect file type.		
Description	Select a new file and upload it to the system for printing		
Preconditions	- The student has logged into the system - The student's device can connect to the internet and the system		
Postconditions	A completely new file is uploaded to the system.		
Normal Flow	1) The student selects the "Re-upload" button		

	2) The system opens the interface to select the file to upload 3) The student selects the file to upload 4) The student clicks the confirm button
Alternative Flows	After step 2, the system displays 3 buttons: "Select," "Cancel," and "Upload new file": 1) In step 2.1, the student clicks "Cancel" if they cannot find the file to print. 2) In step 2.2, the student clicks "Upload new file" to upload a new file.
Exceptions	None

3. Usecase Verify File Type

Use Case ID	UC-PD03		
Use-case name	**Verify File Type**		
Created by	Tuấn Huy	**Last Updated By**	Nhật Huy
Date Created	25/9/2024	**Date Last Updated**	29/9/2024
Actor	Students		
Trigger	The file upload process by the student is completed and the system will verify file type.		
Description	Once the student uploads a document to the system, the system automatically initiates the "Verify File Type" use case to ensure that the uploaded file format is allowed by the system, based on the configurations set by the SPSO.		
Preconditions	- The student has selected a file to upload. - The file must be in a supported format (e.g., PDF, DOCX, JPEG).		
Postconditions	A completely new file is uploaded to the system		
Normal Flow	1) The student selects the file to upload. 2) The system checks the file type. 3) If the file type is supported, the system proceeds with the upload.		

	4) If the file type is invalid, the system prompts the user to select a valid file format.
Alternative Flows	The system offers a list of supported file types before uploading, so the student is aware of acceptable formats.
Exceptions	The system rejects the file because it is of an unsupported format (e.g., an executable file or an unknown format).

4. Usecase Search For Printer

Use Case ID	UC-PD04		
Use-case name	**Search For Printer**		
Created by	Tuấn Huy	**Last Updated By**	Nhật Huy
Date Created	25/9/2024	**Date Last Updated**	29/9/2024
Actor	Students		
Trigger	The student needs to find a specific printer on campus		
Description	The student can search for a specific printer based on information such as location, printer ID, printer brand, or model		
Preconditions	- The student has successfully logged into the system - The student has uploaded the document to be printed to the system		
Postconditions	The student has found a suitable printer		
Normal flow	1) The student selects the "Search printer" option if they need to find a suitable printer on campus 2) The system displays a search interface with multiple search criteria, such as location, printer ID, and wait time 3) The student enters the search criteria to find a printer 4) The system displays search results based on the student's entered criteria and provides a list of suitable printers 5) The student reviews the results in the list and selects the most appropriate printer		
Alternative flow	None		

Exceptions	Exception 1: At step 4, a system error occurs during the search process. The system displays an error message and asks the student to try again Exception 2: At step 4, the student cannot find a suitable printer in the list. The system displays a message stating that no printer matches the search criteria and suggests adjusting the search criteria

5. Usecase Choose Printer

Use Case ID	UC-PD05		
Use-case name	**Choose Printer**		
Created by	Tuấn Huy	**Last Updated By**	Nhật Huy
Date Created	25/9/2024	**Date Last Updated**	29/9/2024
Actor	Students		
Trigger	The student has uploaded the document to the system and needs to choose a printer		
Description	The student can select a printer from a list of available printers to print their document		
Preconditions	- The student has successfully logged into the system - The student has uploaded the document to be printed to the system		
Postconditions	- The student has successfully chosen a printer - The system confirms that the printer has been selected		
Normal flow	1) The system displays an interface with a list of available printers for the student 2) The student reviews the printer information from the list 3) The student selects a printer from the list 4) The student confirms the selected printer 5) The system records the student's selection		
Alternative flow	None		

Exceptions	Exception 1: At step 2, a system error occurs while displaying the list of available printers. The system displays an error message and asks the student to try again Exception 2: At step 3, the selected printer is already in use or encounters an error. The system notifies the student and offers options to either choose another printer or cancel the print job

6. Usecase Check Printer Status

Use Case ID	UC-PD06		
Use-case name	**Check Printer Status**		
Created by	Tuấn Huy	**Last Updated By**	Nhật Huy
Date Created	25/9/2024	**Date Last Updated**	29/9/2024
Actor	Students, SPSO		
Trigger	The student or SPSO wants to know the current status of a specific printer		
Description	The student or SPSO can check the status of a printer		
Preconditions	The student or SPSO has successfully logged into the system		
Postconditions	The student or SPSO has obtained the current status information of the selected printer		
Normal flow	1) The student selects the "Check printer status" option to view the current status of a specific printer 2) The system displays an interface with a list of printers that the student or SPSO can check 3) The student or SPSO selects a printer from the list to check 4) The system then displays the current status of the printer 5) Based on the status information, the student can decide which printer to use, or SPSO can proceed with maintenance or repairs 6) If the student or SPSO determines the printer to be used, they can select "Choose printer" to interact with the chosen printer		
Alternative flow	None		

Exceptions	Exception 1: If a system error occurs while retrieving the status information, the system displays an error message and asks the student or SPSO to try again

7. Usecase Modify Printing

Use Case ID	UC-PD07		
Use-case name	**Modify Printing**		
Created By	Tuấn Huy	**Last Updated By**	Nhật Huy
Date Created	25/9/2024	**Date Last Updated**	29/9/2024
Actor	Students		
Trigger	Student want to modify some printing properties		
Description	The student can change the printing properties (such as paper size, pages to print, number of copies, or one-/double-sided printing).		
Preconditions	1) The system is operational 2) The database is connected to SSPS 3) Internet connection is available 4) The student is successfully logged in and authenticated 5) The student has uploaded the file to be printed		
Postconditions	None		
Normal flow	1) The student selects the file for which they want to modify the print settings 2) The student selects the "Modify printing" button 3) The system displays a modification dialog box 4) The student updates and selects the desired print settings 5) The student clicks the "Finish" button to complete the modification process		
Alternative flow	1) In the step 5.1: After step 5, if the student wants to modify the settings again, the student selects the "Modify printing" button to continue adjusting the settings		

	2) In the step 5.2: The student selects the "Refresh" button, and the system displays a new modification dialog box, allowing the student to make further adjustments
Exceptions	None

8. Usecase View Print Preview

Use Case ID	UC-PD08		
Use-case name	**View Print Preview**		
Created By	Tuấn Huy	**Last Updated By**	Nhật Huy
Date Created	25/9/2024	**Date Last Updated**	29/9/2024
Actor	Students		
Trigger	The student wants to see how the document will look based on the selected printing settings		
Description	The student previews the print layout before submitting a print request and the system then generates and displays a visual representation of the document based on the selected printing options		
Preconditions	1) The system is operational 2) The database is connected to SSPS 3) Internet connection is available 4) The student is successfully logged in and authenticated 5) The student has uploaded the file to be printed		
Postconditions	None		
Normal flow	1) The student selects the file to preview 2) The student clicks the "Preview" button 3) The system presents the print preview 4) The student clicks the "Print" button to print the file		
Alternative flow	1) In step 1, the student can select a file that has already been uploaded or click "Preview" after completing the **Modify Print Settings**		

	2) After step 3, the system displays two buttons for selection: "Edit" and "Print": 3.1. The student selects the "Edit" button to modify the print settings 3.2. The student selects the "Preview" button to review the print layout again 3.3. The student selects the "Print" button to print
Exceptions	None

9. Usecase Buy Printing Pages

Use Case ID	UC-PD09		
Use-case name	**Buy Printing Pages**		
Created By	Tuấn Huy	**Last Updated By**	Nhật Huy
Date Created	25/9/2024	**Date Last Updated**	29/9/2024
Actor	Students, BK Pay		
Trigger	The student attempts to submit a print request but discovers an insufficient balance of printing pages		
Description	The student can purchase additional printing pages using BK Pay		
Preconditions	1) The system is operational 2) The database is connected to SSPS 3) Internet connection is available 4) The student is successfully logged in and authenticated 5) The student has uploaded the file and selected to print it		
Postconditions	None		
Normal flow	1) The system notifies the student of the number of additional pages needed for printing 2) The student selects the "Buy more pages" button 3) The system presents a dialog box for purchasing pages and informs the student of the required payment amount		

	4) The student enters the necessary information as required by the system to complete the transaction 5) The student selects the "Confirm" button to finalize the transaction
Alternative flow	After step 1, the system displays two buttons: "Buy more pages" and "Cancel": 2.1. The student selects the "Cancel" button to cancel the print job
Exceptions	None

10. Usecase Submit Print Request

Use Case ID	UC-PD10		
Use-case name	**Submit Print Request**		
Created by	Tuấn Huy	**Last Updated By**	Nhật Huy
Date Created	25/9/2024	**Date Last Update**	29/9/2024
Actor	Students		
Trigger	The student wants to send the print job request to the system		
Description	The student submits a request to print a document that has been successfully uploaded to the system.		
Preconditions	- The student has successfully uploaded a document to the system. - The student is logged into the system.		
Postconditions	A print request is submitted. Ready to check account balance.		
Normal Flow	1) The student selects the "Submit" button. 2) The system confirms the print request has been submitted successfully.		
Alternative Flows	None		
Exceptions	None		

11. Usecase Check Account Balance

Use Case ID	UC-PD11		
Use-case name	**Check Account Balance**		
Created By	Tuấn Huy	**Last Updated By**	Nhật Huy
Date Created	25/9/2024	**Date Last Update**	29/9/2024
Actor	Student		
Trigger	The student wants to verify their remaining printing pages or buying additional pages		
Description	Students check account balance and BK Pay ensures sufficient funds for printing or other services.		
Preconditions	- The student is logged into the system. - The student's account contains information regarding available funds.		
Postconditions	The system displays the student's account balance and notify students whether they have enough money to make the transaction or not.		
Normal Flow	1) The student selects the "Check Balance" button. 2) The system retrieves the account balance from the database. 3) The system displays the account balance to the student.		
Alternative Flows	None		
Exceptions	None		

12. Usecase Process Transaction

Use Case ID	UC-PD12		
Use-case name	**Process Transaction**		
Created By	Tuấn Huy	**Last Updated By**	Nhật Huy
Date Created	25/9/2024	**Date Last Updated**	29/9/2024

Actor	Students, BKPay
Trigger	The student requests to buy printing pages, prompting the system to process the payment.
Description	This use case describes the process of purchasing printing pages by interacting with the payment system (BKPay).
Preconditions	1) The student must have selected the option to buy printing pages. 2) The system must be connected to the BKPay payment gateway.
Postconditions	The printing pages are added to their account.
Normal flow	1) The student initiates the process to buy printing pages. 2) System checks the student's account balance via BKPay. 3) BKPay processes the payment transaction. 4) The system confirms the transaction and updates the account balance.
Alternative flow	In step 2, if there are insufficient funds, the student is prompted to add more funds.
Exceptions	Exception 1: If the payment fails, the system notifies the student, and the transaction is not processed.

13. Usecase Save Printing Log

Use Case ID	UC-PD13		
Use-case name	Save printing log		
Created By	Tuấn Huy	**Last Updated By**	Nhật Huy
Date Created	25/9/2024	**Date Last Updated**	29/9/2024
Actor	SPSO		
Trigger	The system successfully processes a print request		
Description	This use case describes the logging of print job information for future reference or auditing.		

Preconditions	The print job must have been successfully submitted
Postconditions	The print job information is saved in the system logs
Normal flow	1) After the student submits a print request, the system completes the printing process. 2) The system records the details of the print job (e.g., file name, pages printed, timestamp, printer used). 3) The print log is saved in the system for future auditing or user reference. 4) The system sends a confirmation to the student indicating that the print job was successful and the log has been saved.
Alternative flow	In step 1.1, if the print job fails, the system logs the error details.
Exceptions	None

3. Task 2: System Modeling

3.1 Activity Diagrams

3.1.1 Printer selecting diagram

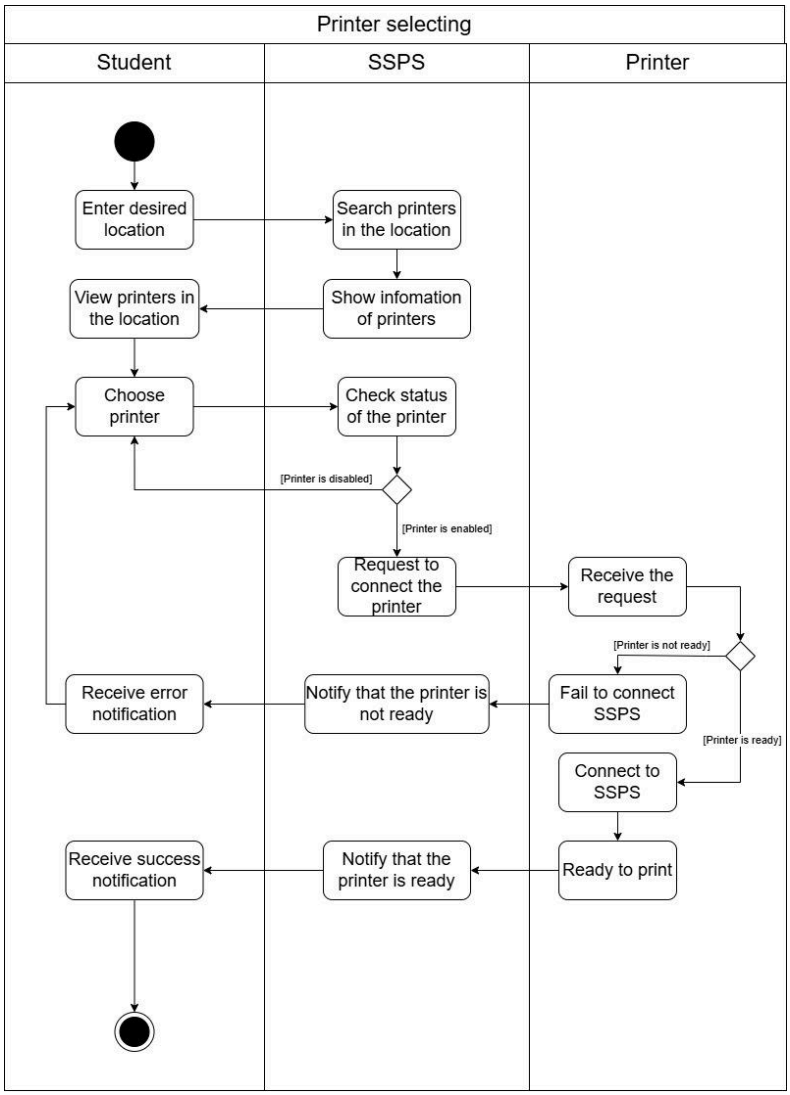

Figure 3. Printer selecting - Activity Diagram

The printer choosing diagram totally contains fifthteen steps for a student to select his/her desired printer. In the first stage, the student will have to enter an expected location and the SSPS will do a search in this area to find nearby printers. All information of the printers will be shown up to the student and then he/she has to choose a desired printer. After selecting a printer, SSPS continues checking the status of it, in case that printer has been disabled by SPSO before, the system will make the student choose another printer in

the location. If the status is “enabled”, the SPSO will send a request to the printer to make sure it is available to print. The printer receives the request and if there are some problems with the printer such as broken one, out of printing ink, failure transmission, etc, the SSPS cannot connect to it, then it will inform the student that the printer is not ready and make him/her choose another one. The student will receive the failure notification and change the printer. If the printer is working, it will connect to SSPS and be ready to print, then the SSPS recognizes that successful signal and notify the students that the printer choosing process is completed. The student will receive the notification and then the process ends.

3.1.2 File uploading diagram

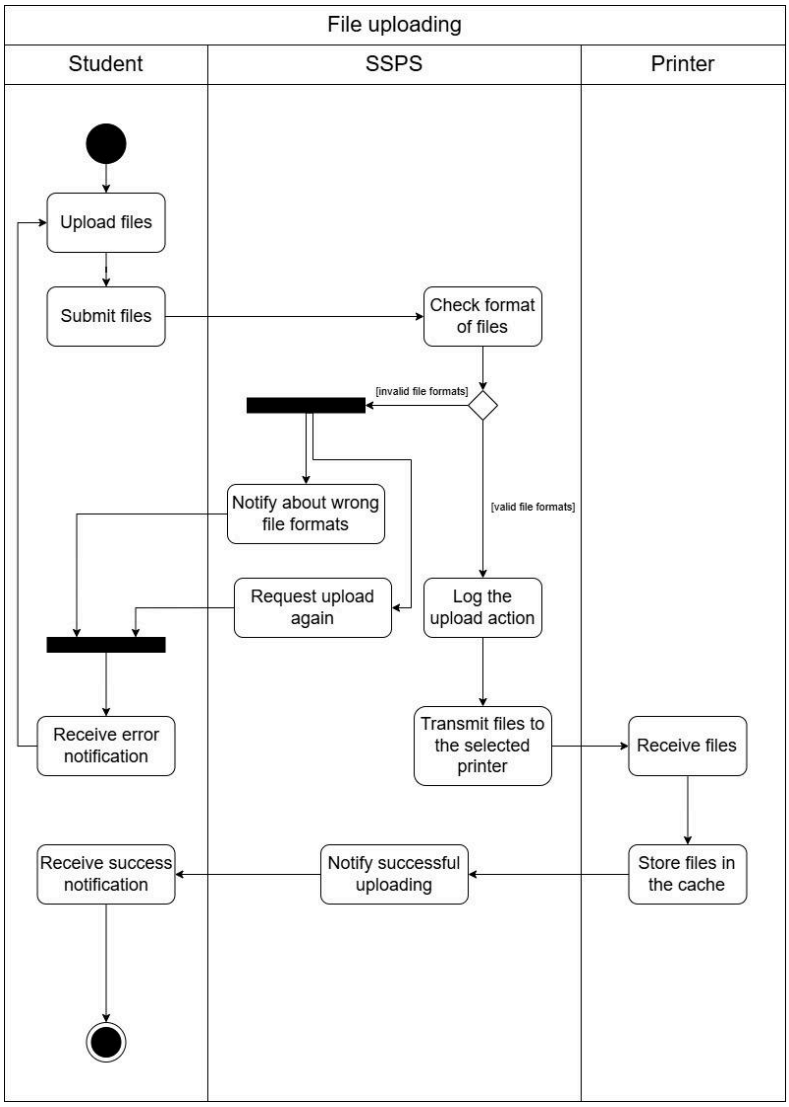

Figure 4. File uploading - Activity Diagram

The activity diagram of file uploading use case has twelve steps in the uploading procedure. Firstly, the student will start with uploading their desired files, documents to print and then they have to click the submit button to confirm the process. The SSPS system will receive the files and start to check the format of them to make sure they are valid types which are determined by the SPSO. In case of invalid formats, the SSPS will simultaneously notify the student about the errors and request he/she to upload again files with acceptable formats. The student will receive the failure notification and come back to the first stage to upload files again. If the formats are appropriate, the SSPS recognizes the uploading action of the student and logs it into the stored record, then the files will be transmitted to the selected printer which is decided by the student in the printer choosing phase. The printer receives the files, stores them in its cache and sends back to the SSPS a signal that the printer is ready to print. The SSPS recognizes the signal and informs the student that the uploading process is successfully finished, then the file uploading procedure ends.

3.1.3 Printing submission

The printing submission diagram totally consists of fifteen stages. Firstly, the user will configure the properties for the printer such as paper sizes, number of pages, one/doubled-sized. The SSPS will recognize the settings and the student has to click on the submission button to confirm these configurations. After that, the SSPS will check the account balance of the student, in case the number of printing pages exceeds the balance, it will request the user to purchase more pages. Therefore, the student has to enter his/her desired number of pages and the system will show the total amount of money needed for purchasing. To confirm the purchase, the student will send a request to the BKay and then it will handle the transaction. If the number of pages does not exceed the balance, the SSPS will send the configured properties which were determined by the user previously to the printer. The printer will receive them and start printing files with the desired settings. After the file printing stage, SSPS will log the printing action into the report and inform the student that the process is successfully completed. The student will receive this notification and the file printing process ends.

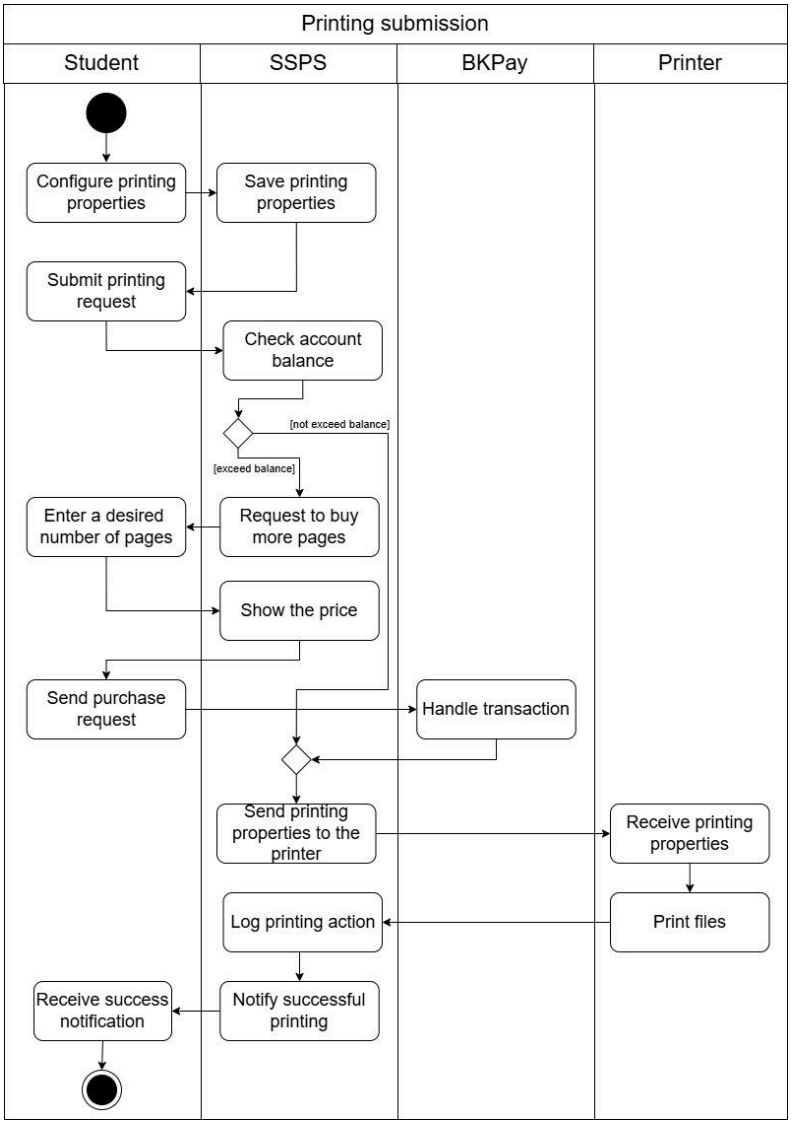

Figure 5. Printing submission - Activity Diagram

3.2 Sequence Diagrams

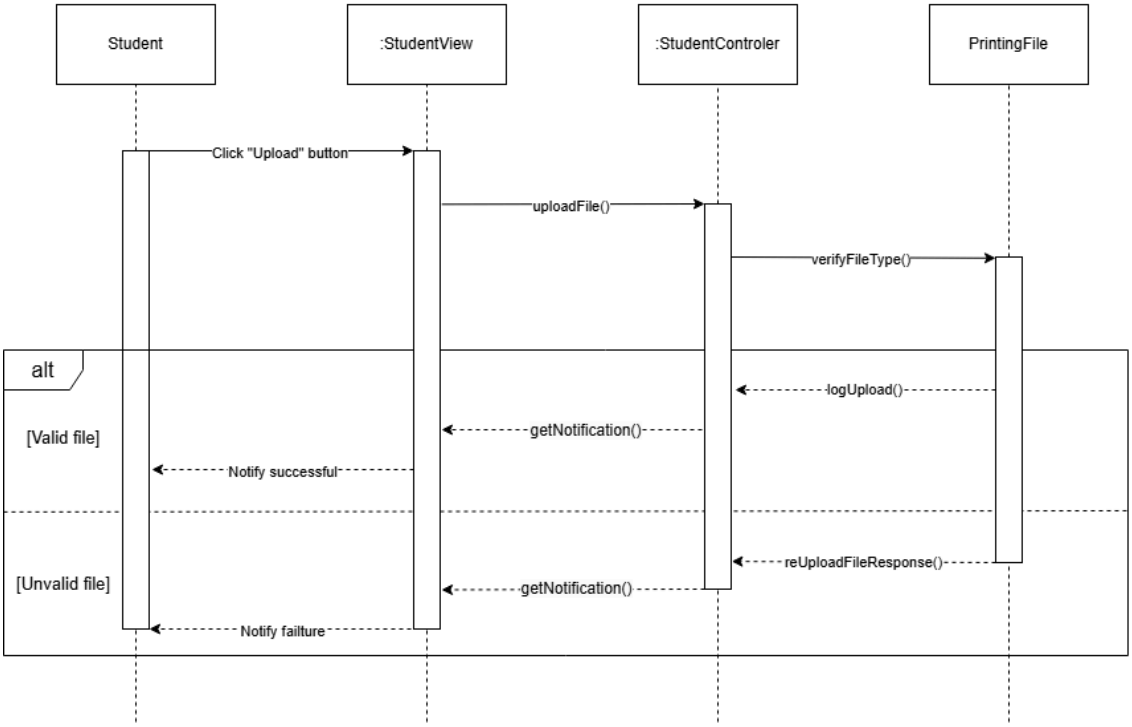

Figure 6. Upload file - Sequence Diagram

Describe upload file diagram (Figure 6):

1. The student initiates the process by clicking the "Upload" button in the StudentView.
2. The StudentView sends a request with the selected file (*uploadFile()*) to the StudentController to begin the file upload process.
3. The StudentController communicates with the PrintingFile to verify the type of the file being uploaded (*verifyFileType()*).
4. There will be 2 cases happening here:
    - If the file is valid, the system logs the upload (*logUpload()*), and the StudentController sends a notification back to the student indicating a successful upload (*getNotification()*).
    - If the file is invalid, the system will request a re-upload (*reUploadFileResponse()*) and notify the student of the failure (*getNotification()*).
5. The student receives a notification from the system about the result (either success or failure).

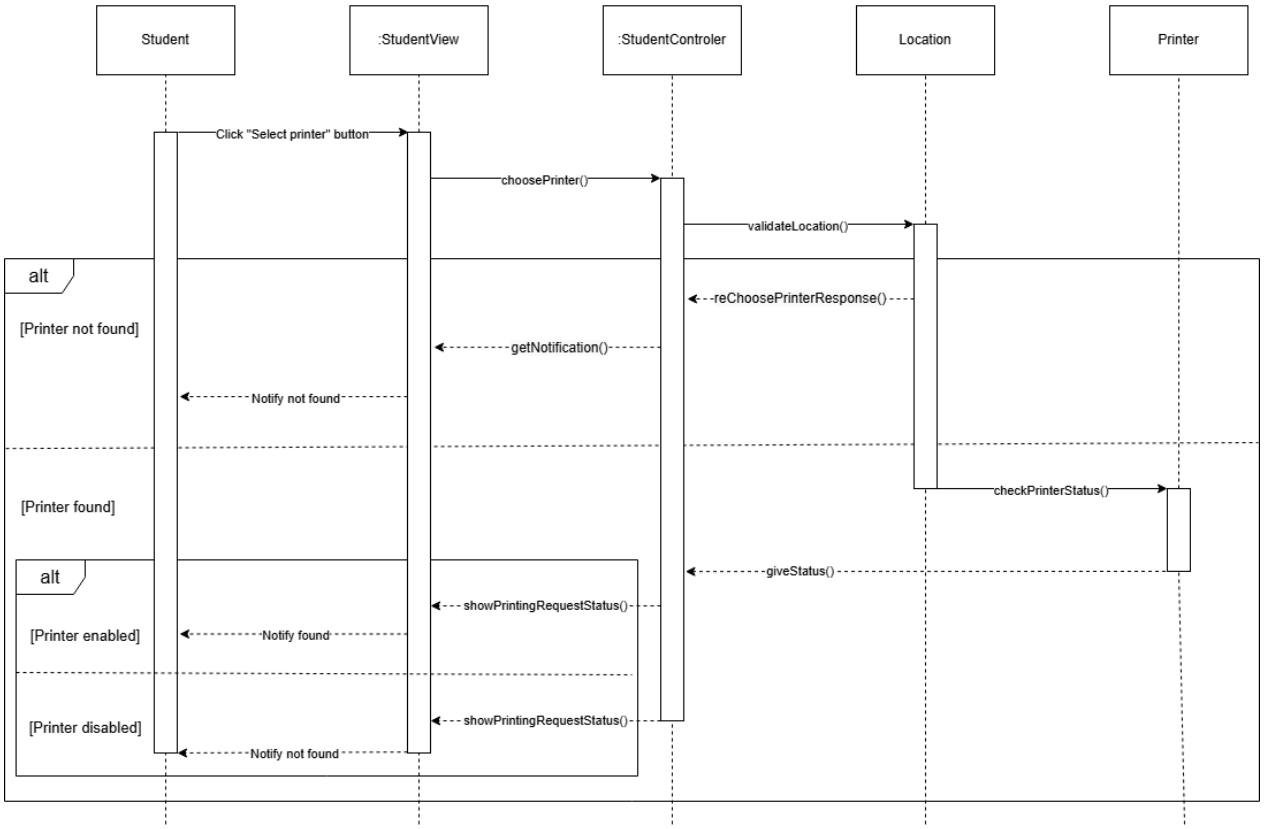

Figure 7. Choose printer - Sequence Diagram

Describe choose printer diagram (Figure 7):

1. The student clicks the "Select printer" button in the StudentView, triggering the process.
2. The StudentView sends a request (*choosePrinter()*) to the StudentController to select the desired printer.
3. The StudentController sends a request to Location to validate the location of the selected printer (*validateLocation()*).
4. There will be 2 cases happening in validate the location:
    - If the printer is not found, the system sends a "Not found" notification back to the student.
    - If the printer is found, the system checks the status of the printer (*checkPrinterStatus()*).
5. Check Printer Status: The StudentController sends a request to the Printer to check its status (checkPrinterStatus()).
6. There will be 2 cases happening in check printer status:
    - If the printer is enabled, the system notifies the student that the printer is available.

- If the printer is disabled, the system sends a notification to the student that the printer is unavailable.

7. The student receives a notification from the system about the printer status (either enable or disable).

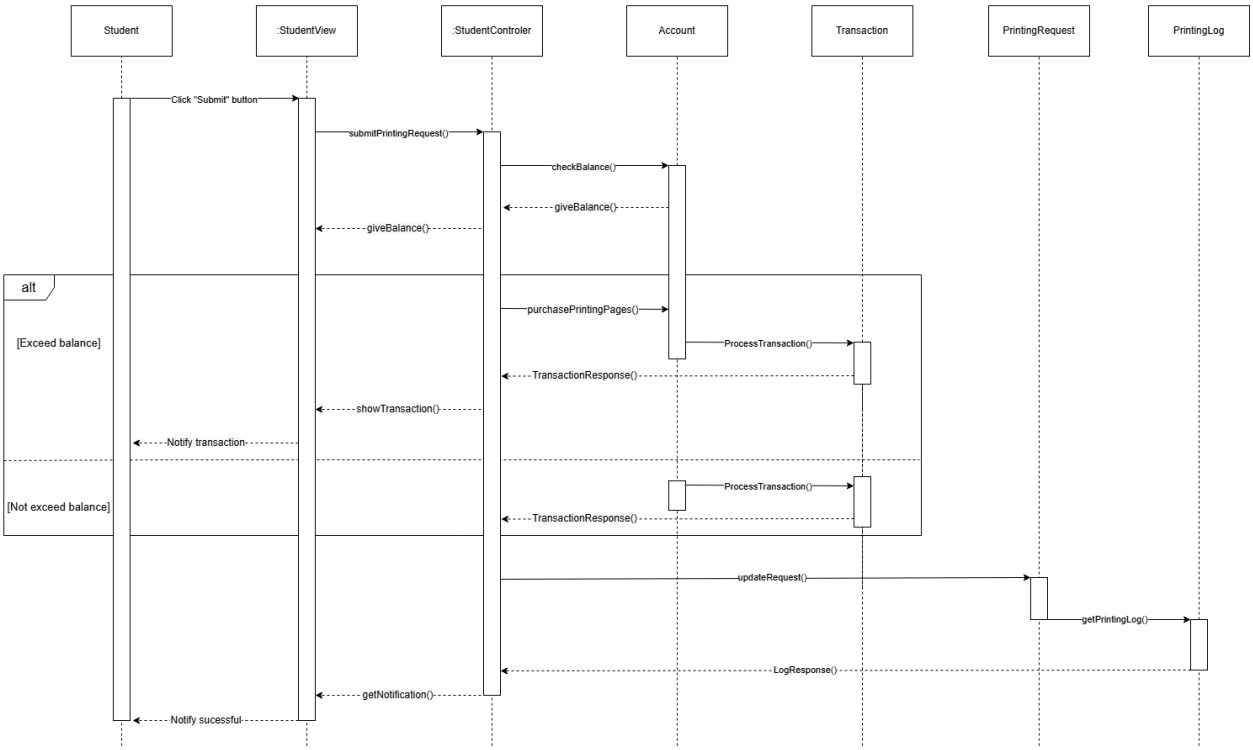

Figure 8. Submit printing file - Sequence Diagram

Describe submit printing file (Figure 8):

1. The student initiates the process by clicking the "Submit" button in the StudentView.
2. The StudentView sends a request (*submitPrintingRequest()*) to the StudentController.
3. The StudentController communicates with the Account to check the balance (*checkBalance()*).
4. There will be 2 cases happening here:
   - If not exceeding balance, the Account communicates with the Transaction and conducts transaction (*processTransaction()*).
   - If exceeding balance, the StudentController communicates with the Account to purchase pages(purchasePrintingPages). After that, the Account communicates with the Transaction and conducts transaction (*processTransaction()*).

5. The StudentController communicates with the PrintingRequest to update requests (*updateRequest()*).
6. The PrintingRequest communicates with the PrintingLog to log the requests (*getPrintinglog()*).
7. The PrintingRequest returns the response to StudentController.
8. The StudentController returns notification to StudentViewer.
9. The student receives a notification from the system about the result from StudentViewer.

3.3 Class Diagram for Whole module

Here are some key operations of the printing module *(See in Figure 9)*:

1. Submitting a printing request:
    - A Student uses the *StudentView* to access the *displayPrintingRequestForm()*.
    - The *StudentController* manages the process, allowing the student to *uploadFile(), choosePrinter()*, and *submitPrintingRequest()*.
    - The *PrintingRequest* is created with details like number of copies, file type, and printer selection.
    - Then the *PrintingRequest* is sent to the *Printer* to print.
    - *PrintingLog* records all *PrintingRequest* activities, accessible via *getPrintingLog()*.
    - Students receive updates and notifications on request status through *getNotification()* in *StudentView*.
2. Printer management:
    - *SPSOController* manages printer operations and can *refillPrinter()* when necessary.
    - *Printer* objects maintain their status, availability, and remaining pages.
    - *SPSO* staff can *checkPrinterStatus()* to ensure functionality.
3. Buying printing pages:
    - Each student owns an *Account* to access the system, each *Account* contains information about its *balance*, *currentPage* available and a list of *Transactions.*
    - A *Transaction* is created when the student wants to buy more printing pages. It contains an *ID*, the *amount* of money, *date*,...

Workflow:

- A student logs into the system, selects the files to print, and submits a request.
- The system checks the student's account current printing page availeble.
- There are 2 cases that can happen:

- If the account does have enough pages, the system will process the request.
- If the account does not have enough pages, the student must create a transaction to buy more printing pages before continuing printing.

- The request is sent to the chosen printer, and the system logs the details.
- Staff manage printer operations, ensuring they are ready and refilled as needed.
- The system notifies the student about the request status and any required actions (e.g., re-uploading files).

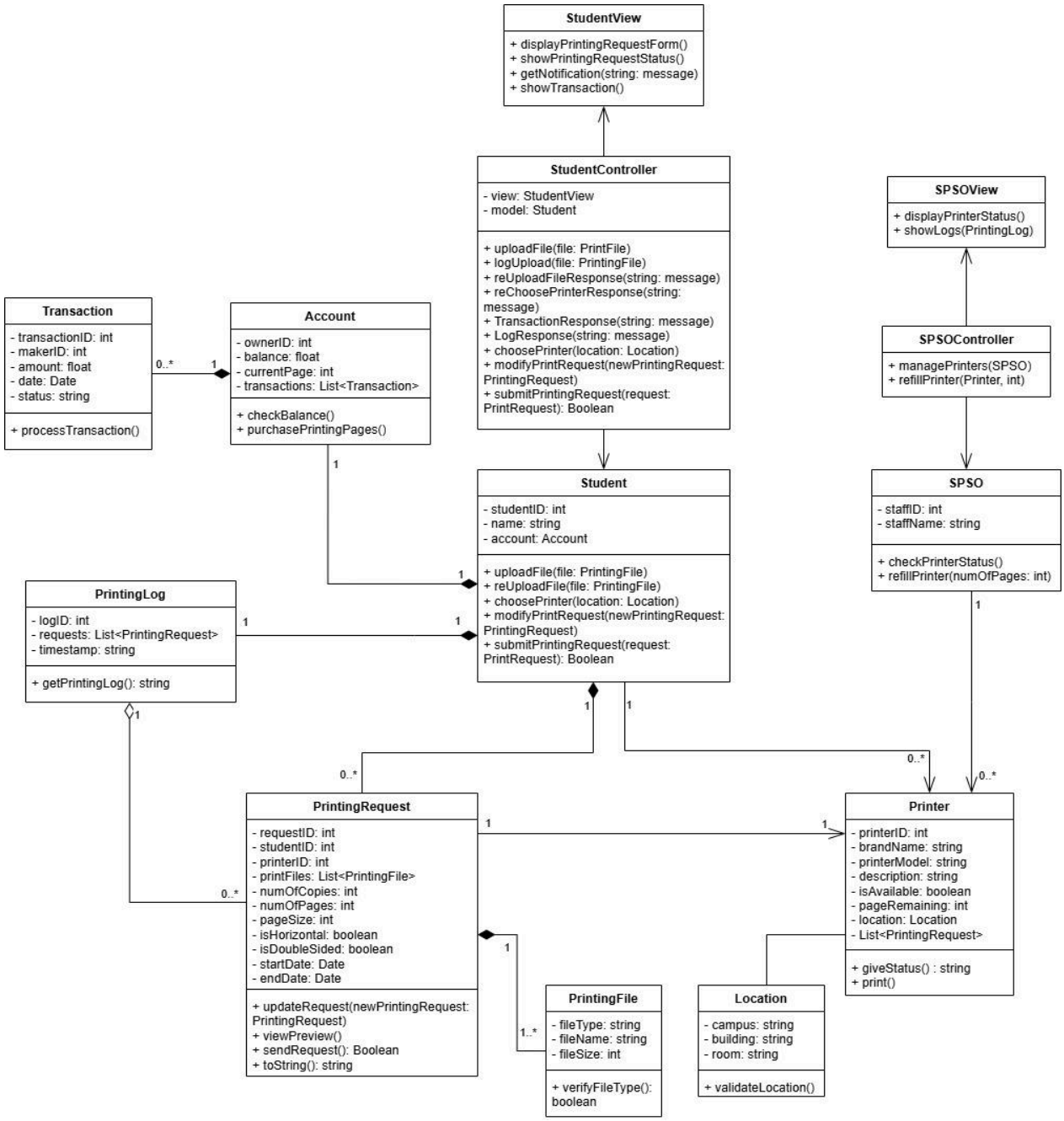

Figure 9. Class Diagram for whole Printing module

3.4 User Interfaces - MVP1

Interface designs are implemented on Figma *(see all screens at the following link)*.

The header of the entire webpage within the system includes navigation bars leading to different pages:

- "Trang chủ": Redirects to the Landing page.
- "Chúng tôi": Redirects to a page introducing the team and the product.
- "Dịch vụ in": Redirects to the Printer Searching page *(Figure 12)*.
- "Giao dịch": Redirects to the Account Balance page *(Figure 16)*. If the user is not logged in, it returns to the Login page.
- "In ngay": Redirects to the Request Printing page *(Figure 13)*.

The sidebar of the entire webpage (for the user role) contains navigation bars leading to various pages:

- "Thông tin": Redirects to the Account Information page *(Figure 10)*. If the user is not logged in, it returns to the Login page.
- "Báo cáo": Displays a popup form *(Figure 11)* for logged-in users to report issues to the SPSO. The user fills in the required fields: printer ID, issue occurrence time, issue description, and optionally uploads issue-related images.
- "Tìm kiếm máy": Redirects to the Printer Searching page *(Figure 12)*.
- "Yêu cầu in": Redirects to the Request Printing page *(Figure 13)*.
- "Lịch sử in": Redirects to the Printing History page *(Figure 15)* for logged-in users; otherwise, it returns to the Login page.
- "Số dư tài khoản": Redirects to the Account Balance page *(Figure 16)* for logged-in users; otherwise, it returns to the Login page.
- "Lịch sử mua": Redirects to the Payment History page *(Figure 17)* for logged-in users; otherwise, it returns to the Login page.

3.4.1 Account Information Page

For logged-in users. The page is divided into two sections:

- The left section displays user (student) information.
- The right section shows bar charts of printing frequency and page purchase statistics (by year). Each chart includes buttons to navigate to the Printing History page *(Figure 14)* (by clicking on "Xem lịch sử in" button) and

Payment History page *(Figure 16)* (by clicking on "Xem lịch sử giao dịch" button).

The design of this page can be seen in Figure 10.

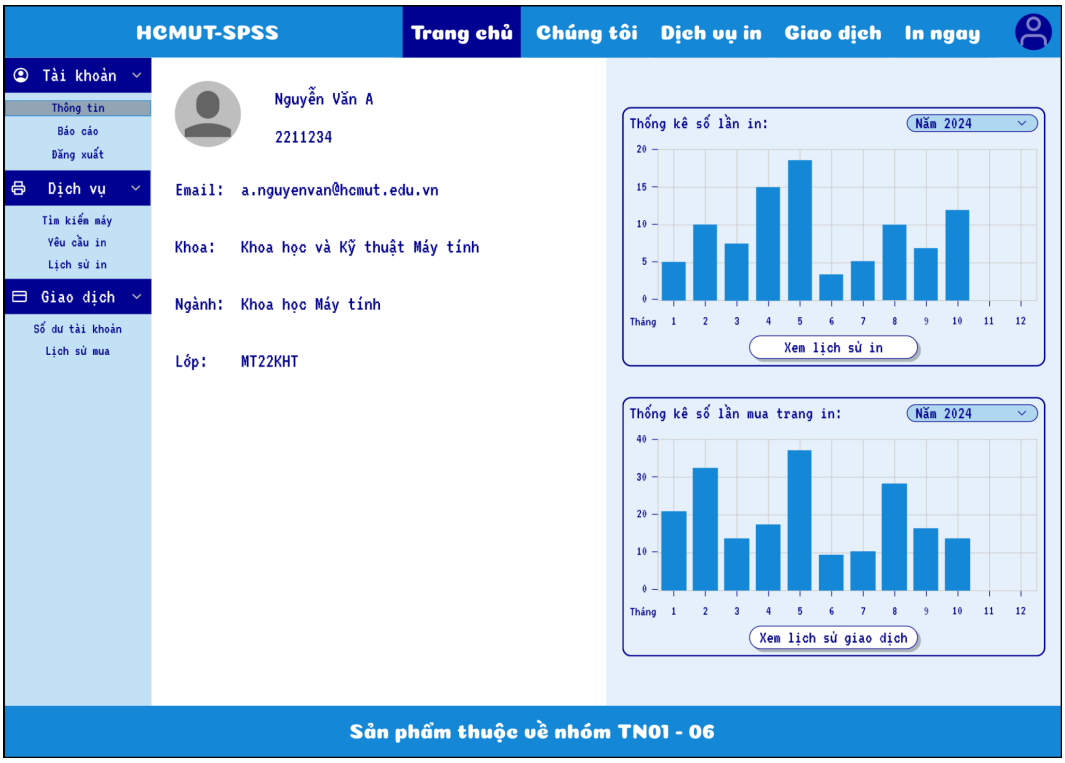

Figure 10. Account information webpage (MVP 1)

3.4.2 Report Issues Form

A popup form (see in *Figure 11*) for logged-in users to report issues to the SPSO. The user fills in the required fields: printer ID, issue occurrence time, issue description, and optionally uploads issue-related images. When you press "Gửi" submit button, this report will be sent to the SPSO management page.

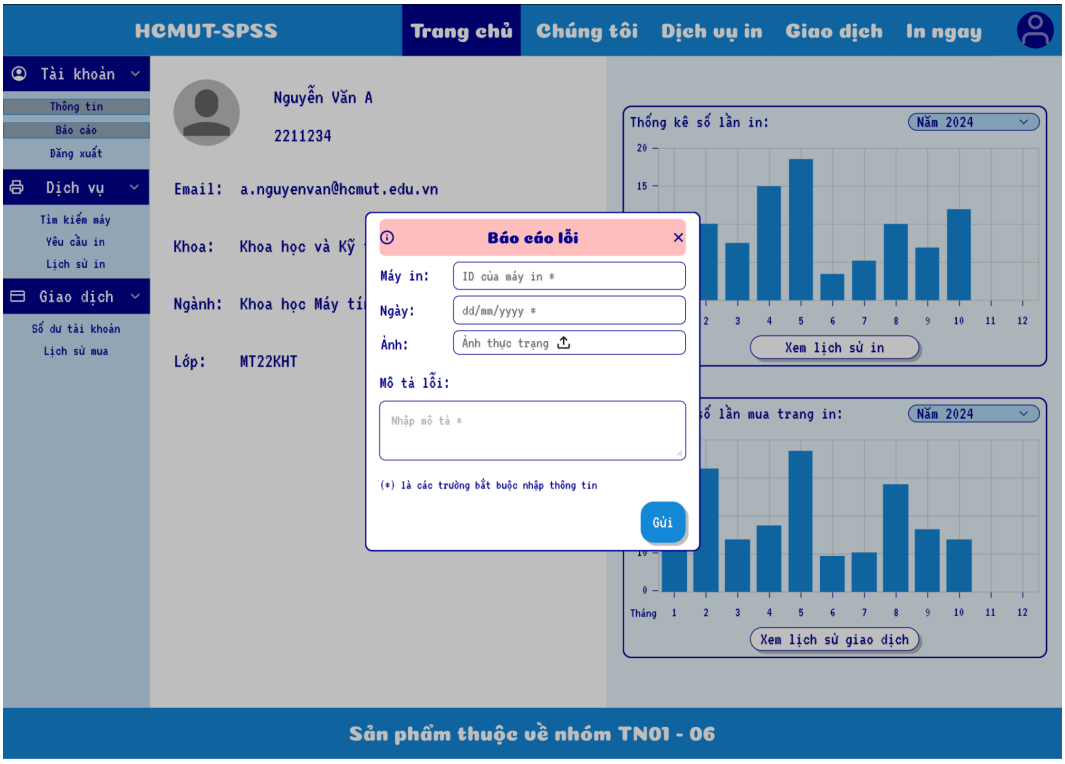

Figure 11. Report issues form (MVP 1)

3.4.2 Searching Printers Page

The Searching Printers Page allows users to find suitable printers for submitting printing requests. The page includes the following features:

- Search Bar: Users can search for printers by name.
- Sorting Options (“Sắp xếp”): Printers can be sorted by ID, name, or model.
- Filters: Users can filter printers by building (“Tòa”) and status (“Trạng thái”) active/inactive.

The results are displayed in a table below, containing the following information for each printer:

- ID: Printer identification number.
- Tên (Name): Printer name.
- Thương hiệu (Brand): Printer brand.
- Mẫu (Model): Printer model type.
- Tòa (Building): Location building.
- Phòng (Room): Room number.

- Trạng thái (Status): Printer status (“hoạt động” - active or “ngừng” - inactive).

And, if there are many printers, pagination is implemented to manage the display of the printer list. To proceed, users can click on a printer in the table and press the "Chọn" button to navigate to the Request Printing Page *(Figure 13)*.

Figure 12. Searching printers webpage (MVP 1)

3.4.3 Request Printing Page

This page is divided into two sections:

- Left Section: “Tạo yêu cầu in” - Create Print Request
- Mandatory Fields:
  - Printer ID (Máy in): Users must enter the printer’s ID either by typing it directly or using the value selected from the Searching Printers Page. A search button allows users to return to the Searching Printers Page *(Figure 12)* to select a suitable printer. All previously entered form data will remain intact upon returning to this page.

- File: Users must upload the file to be printed (only one file is allowed).

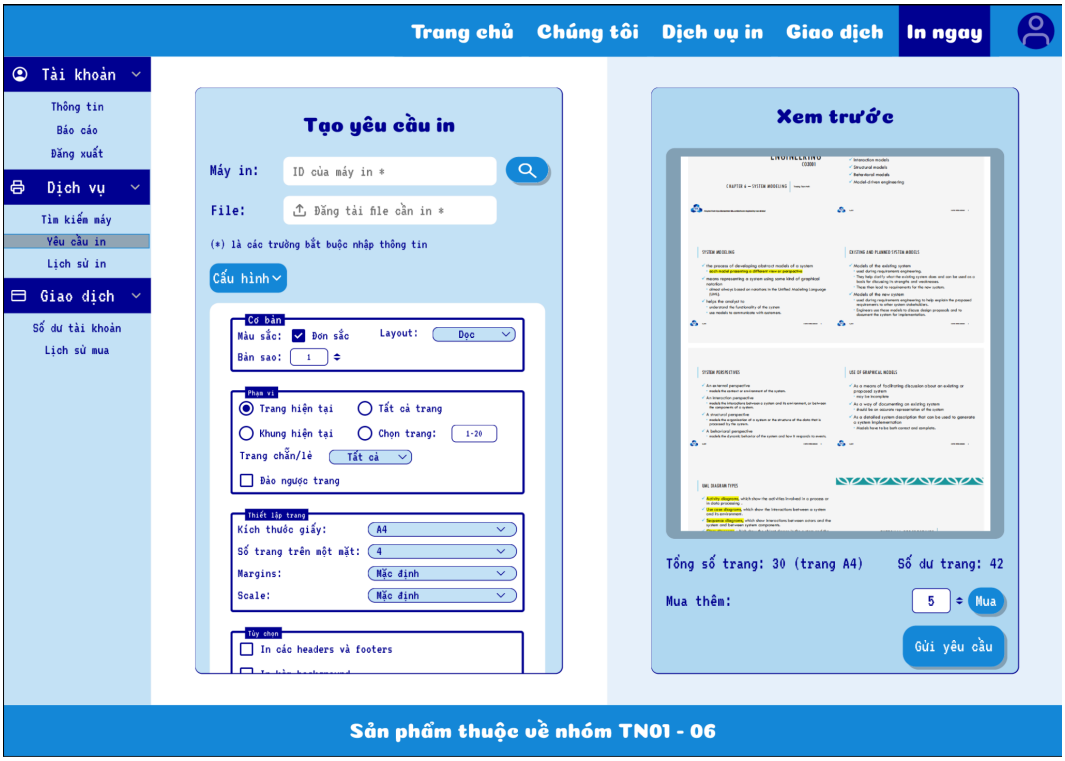

Figure 13. Request Printing webpage (MVP 1)

- Configuration dropdown *(Figure 14)*:
  - After successfully uploading the file, users can click the "Cấu hình" button to expand a dropdown menu with detailed printing configuration options. If users do not modify the settings, the system will use the default configuration saved from previous sessions.
  - Configuration changes are dynamically reflected in the right section "Xem trước".
  - Main options in the configuration dropdown include:
    - "Đặt làm mặc định" - Set as Default Configuration: Save the current settings for future print requests.
    - "Reset": Clear the current settings and revert to the previous default configuration.

- "Xác nhận" - Confirm: Confirm the selected configuration settings. Then the "Xem trước" section will be updated corresponding.

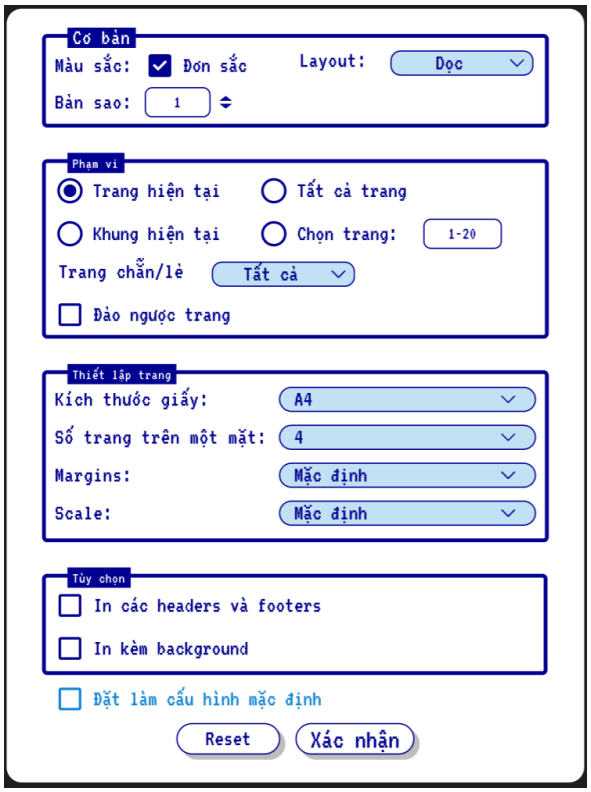

Figure 14. Configure Printing Request Form (MVP 1)

- Right Section: "Xem trước" - Preview
  - Displays a preview of the uploaded file.
  - Shows additional details such as:
    - Total number of pages required for printing.
    - Remaining pages in the user's account balance.
    - "Mua thêm trang" - Buy more printing pages: A button allowing users to purchase additional pages if needed. Clicking this button redirects the user to the BKPay system for processing the purchase.
    - Finally, the "Gửi yêu cầu" button allows users to send the print request to the SPSO management system.

3.4.4 Printing History Page

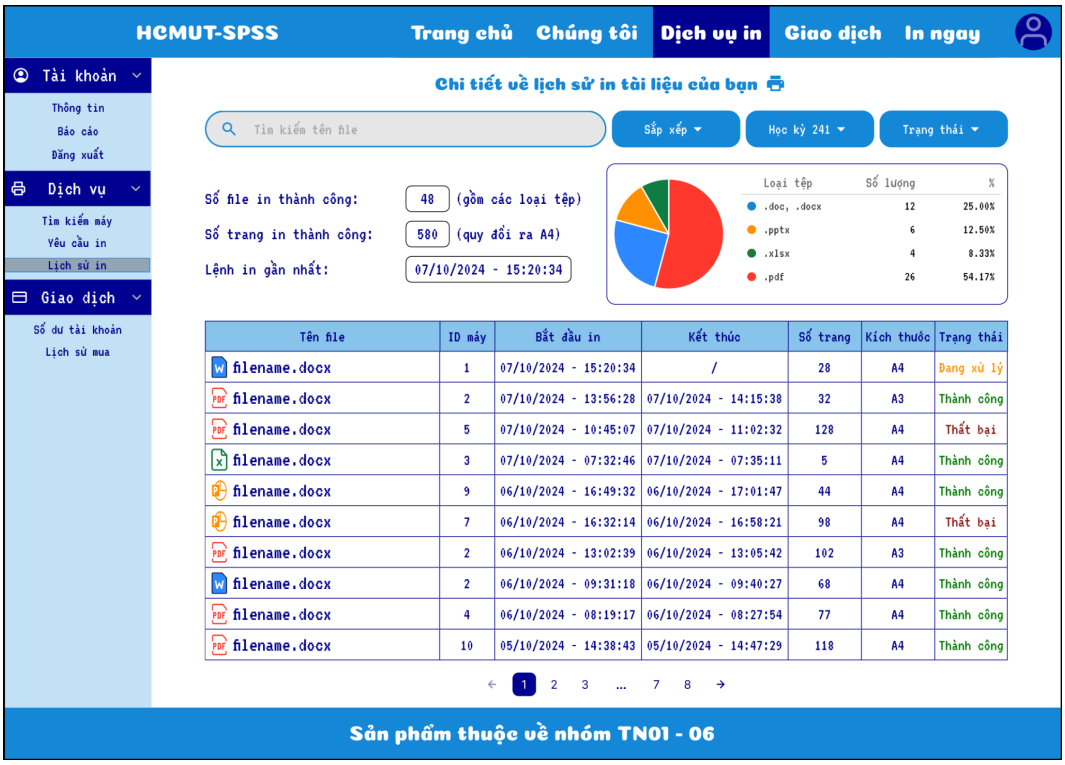

Figure 15. Printing History of a user webpage (MVP 1)

This is the Printing History page for a user in the HCMUT Smart Printing Service. The key features are:

- Search function to find files by filename.
- Sorting options (Sắp xếp) for filename, printer ID, start/end time, number of pages, file size, and print status.
- Filtering by semester and status of printing request.
- Printing history displayed in a table format with details like filename, printer ID, start/end time, number of pages, file size, and print status. And pagination controls to navigate through multiple pages of printing history.
- Printing statistics including total successful prints, total pages printed, and most recent print time. Additionally, there is a pie chart showing the breakdown of file types printed (.doc, .pptx, .xlsx, .pdf)

The page allows the user to easily review their complete printing history, filter and sort the data, and see high-level usage statistics. This provides transparency and helps the user manage their printing activity and page balance.

3.4.5 Account Balance Page

Figure 16. Account Balance webpage (MVP 1)

The Account Balance Page displays detailed information about page balance changes (printing pages) and transactions for purchasing additional pages. It is divided into two sections:

- Left Section: Summary Statistics for the Semester
  - Displays the following details:
    - Current page balance (Số trang in hiện có);
    - Default page allocation for the current semester (Số trang mặc định);
    - Additional pages purchased (Số trang đã mua);
    - Current price per A4 page (Giá hiện tại);
  - Option to purchase additional pages: Users can input the desired number of A4 pages and click the “Mua” button. The system will process the request and redirect the user to the BKPay system for payment.
  - At the bottom, a bar chart shows statistics of page purchase transactions by month for a specific academic year.

- Right Section: Transaction and Print History
    - Shows a summary of recent printing activities and provides links to view the full printing history and payment history.
    - Includes the following buttons:
        - "Xem lịch sử giao dịch" - View Transaction History: Redirects to the Payment History Page *(Figure 17)*.
        - "Xem lịch sử in" - View Printing History: Redirects to the Printing History Page *(Figure 15)*.

3.4.6 Payment History Page

Figure 17. Payment History webpage (MVP 1)

This page displays detailed records of all printing page changes and transactions (decreases when a print job is successfully processed, and increases when additional pages are purchased). It provides the following features for users:

- Displays a log of all printing-related transactions, including successful prints and page purchases.
- Search and Filter Options:
    - Search by transaction ID.

    - Sort by transaction ID, time of change, number of pages changed, and final page balance.
    - Filter by semester and transaction type (purchase or printing).
- Includes an overview of:
    - Total number of pages purchased (Tổng số trang đã mua);
    - Total number of pages printed (Tổng số trang đã dùng);
    - Details of the most recent page purchase (Lần mua gần nhất).
- The filtered results are displayed in a table format with 10 rows per page. And provides pagination controls to navigate through multiple pages of transaction history

The page gives users full transparency into their printing activity and account activity. They can review each individual transaction, see the impact on their page balance, and identify any issues or discrepancies. The comprehensive transaction log and search/filter capabilities make this a valuable tool for students to manage their printing usage and costs within the HCMUT system.

4. Task 3: Architecture Design

4.1 Layered Architecture Diagram

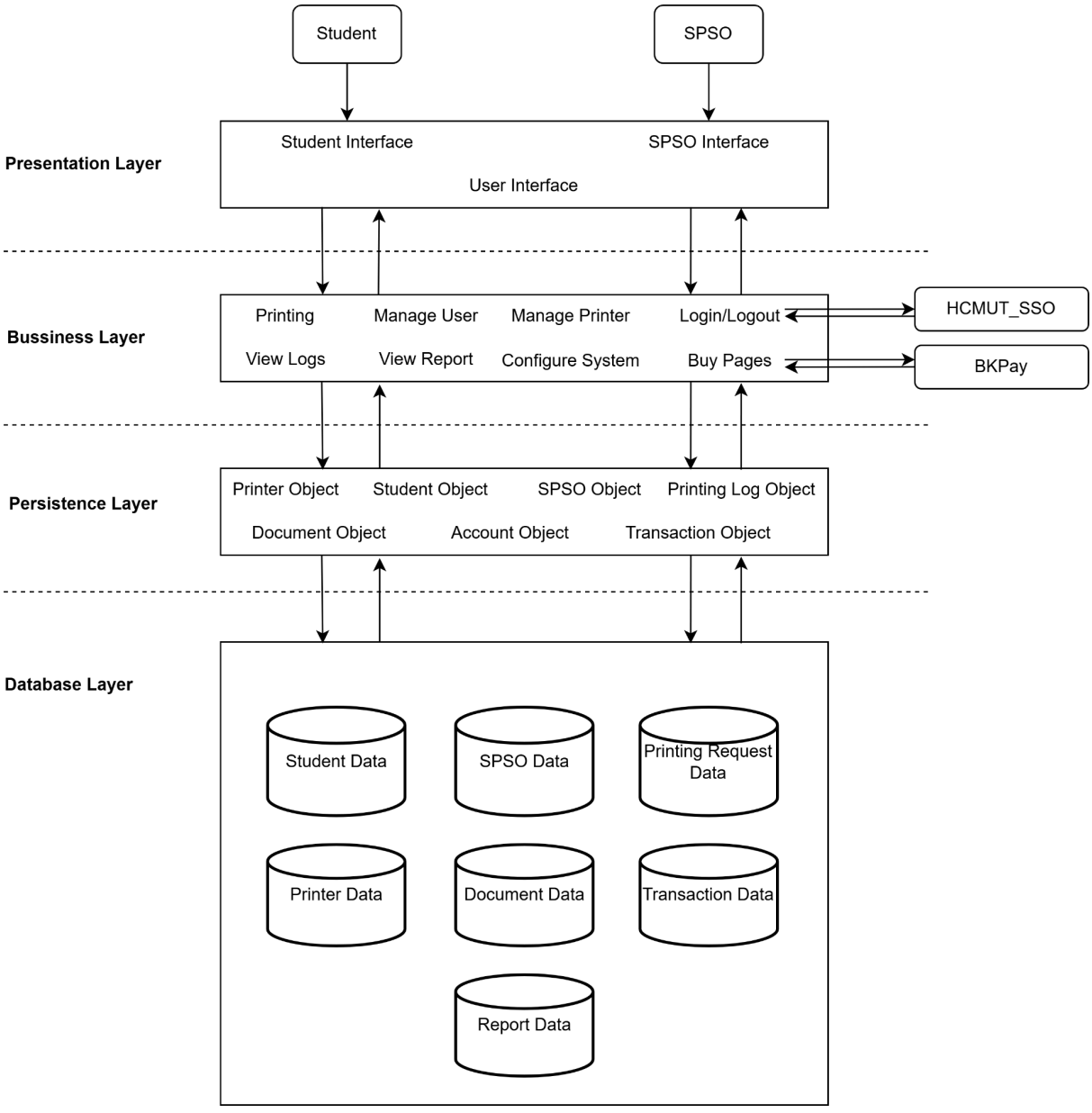

Figure 15. Box-line diagram for architecture of application

4.1.1 Presentation strategy

Here is the first layer in the architecture. We will apply a strategy focused on simplicity, ease of use, and user experience. To achieve this, we will leverage several modern and specific technologies:

- **Front-end library and framework:** We are using front-end development technology such as React. React allows us to create a flexible and efficient interface.
- **Responsive Design:** Ensuring compatibility with all devices used by students and university staff. We will integrate the system with various devices and screen sizes, incorporating flexible forms and interface components to ensure a great user experience across computers, mobile phones, and tablets.
- **User-Friendly Features:** We consider intuitive elements like buttons, forms, and easy-to-use menus, aiming to make the system easily accessible to first-time users.

With the use of React for the user interface, along with flexible design and a focus on user experience, we will create an impressive interface for the HCMUT_SPSS system while ensuring seamless integration with the system's layered architecture.

This layer is dedicated to user interface and interaction, handling user commands and displaying information to enhance user experience. It consists of two main components: the Student Interface and the SPSO Interface. Both share common elements like a Home page, which serves as the entry point when users access or log into SPSS, and a Login page, which adapts based on whether the user is a student or an administrator.

- **Students Interface:**

  The students interface contains some main pages:

  - Home page;
  - Log in;
  - Printer location;
  - Executing transaction page;
  - Printing Page.

- **SPSO Interface**

  The admin interface contains some main pages:

  - Home page;
  - Log in;
  - Dashboard page;
  - Printing page.

The Student Interface provides features tailored to student needs. It includes a Printer Location page that displays available printers and their statuses, helping students

quickly locate functional, nearby printers. The Printing page allows students to upload documents, select a specific printer, and set printing options, ensuring an efficient printing process. If students need additional paper, they can seamlessly navigate to the Buying page to top up their paper balance.

The SPSO Interface, designed for SPSO, includes advanced tools for managing the BKPrint system. The Dashboard page serves as the command center, giving SPSO an overview of user activities, such as print history, printer availability, and real-time printer status. It also allows them to make immediate adjustments to printer settings. SPSO can export periodic reports from this interface, making it invaluable for monitoring system performance and usage trends.

In summary, the User Interface and Interaction layer bridges users with the system's functionalities, optimizing the efficiency and effectiveness of the printing service for both students and administrators.

4.1.2 Data storage approach

In the layered architecture, the database will be the bottom layer, responsible for storing all data and processing it. The application's data will be stored here, and operations like create, search, insert, update, and delete will be frequently performed to interact with the data through a database management system. For the Smart Printing Service project, the team will use a layered architecture, with the database layer stored in a relational database and managed by the MySQL database management system. This means that the application's data will be stored in tables and relationships between them. For the Smart Printing Service system, we need to have the following types of entities:

1. **User**
   - Attributes:
     - User_ID: Unique identifier for each user.
     - Fullname: The full name of the user.
     - Username: The username for user login.
     - Password: The password for user authentication.
   - Relationships:
     - A User can be either a Student or an SPSO.
2. **Student**
   - Attributes:
     - Balance: The current balance of the student.
   - Relationships:

- A Student can Buy and Order printing services.
- A Student can Send multiple Reports.

3. **SPSO (Student Printing Service Officer)**
   - This entity inherits from User and represents the operator role in the system.
4. **Transaction**
   - Attributes:
     - Transaction_ID: Unique identifier for each transaction.
     - Amount: The amount of money involved in the transaction.
     - Balance_after: The balance after the transaction.
     - Timestamp: The date and time of the transaction.
     - Type: The type of transaction.
   - Relationships:
     - A Student can perform multiple Transactions through Buy.
5. **Report**
   - Attributes:
     - Report_ID: Unique identifier for each report.
     - Header: The title or header of the report.
     - Description: Details about the report.
     - Rate: Rating or evaluation related to the report.
   - Relationships:
     - A Student can send multiple Reports.
6. **Printer**
   - Attributes:
     - Printer_ID: Unique identifier for each printer.
     - Brand_name: Brand of the printer.
     - Printer_model: Model of the printer.
     - Description: Additional details about the printer.
     - IsAvailable: Status indicating if the printer is available.
     - Page_remaining: Number of pages left for printing.

    - Location Composite attribute:
        - Campus, Building, Room: Details of the printer's physical location.
- Relationships:
    - A Printer can be used for multiple Print operations.

7. **Printing_request**
    - **Attributes:**
        - Request_ID: Unique identifier for each printing request.
        - Num_of_copies: Number of copies requested.
        - Page_size: Size of the pages requested.
        - IsHorizontal: Orientation of the print (horizontal or vertical).
        - IsDoubledSide: Indicates if printing is double-sided.
        - startDate: Start date of the printing request.
        - endDate: End date of the printing request.
    - **Relationships:**
        - A Printing_request can have multiple Print operations associated with a Printer.
        - A Student can create multiple Printing_request orders.
8. **File**
    - **Attributes:**
        - File_ID: Unique identifier for each file.
        - Size: Size of the file.
        - Type: Type or format of the file.
        - Name: Name of the file.
    - **Relationships:**
        - A File is associated with a Printing_request.

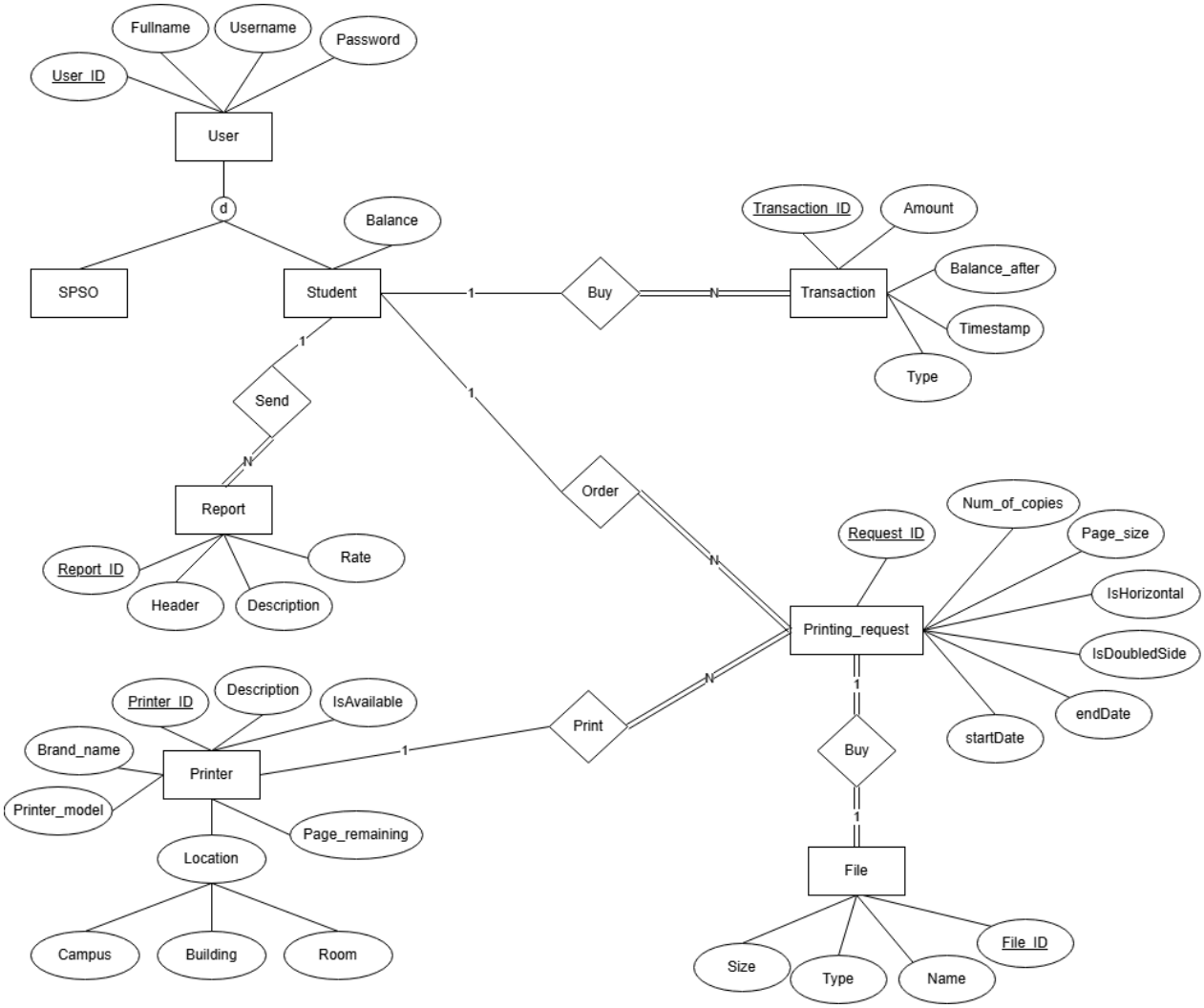

Figure 16. (E)ERD design for database system

4.1.3 API management

APIs (Application Programming Interfaces) are methods and protocols for connecting with libraries and other applications. APIs provide access to a set of commonly used functions, allowing data exchange between applications. The APIs for the HCMUT_SSPS automated printing system include:

- **Security and Authentication API:** Ensures the security of communications and printing operations by providing authentication through API tokens. These tokens are issued by the **HCMUT-SSO authentication service**, where the API sends a request containing the username and password to validate the user's credentials. Upon successful authentication, the **HCMUT-SSO service** issues secure, time-limited tokens for access. Additionally, access rights are managed via API

tokens to control permissions and ensure authorized usage of the system, specifically for Students and SPSO.

- **Get information API**: Retrieve specific data from database. This type of API is typically read-only and focuses on providing requested information without modifying data, used for Student and SPSO.
- **Data Formatting and Input Processing API:** Allows applications to send input data (such as print files, images, or text) to the automated printing system and ensures that this data meets specific formatting and quality requirements.
- **Print Job Management API:** Provides APIs for creating, managing, and tracking print jobs. This can include scheduling prints, monitoring job status, checking available page counts, and canceling print jobs, used for Student.
- **Payment API:** Facilitates payment functionalities (for document printing and purchasing print pages) by integrating with **BKPay**. The Payment API sends the payment details to BKPay, prompting the user to visit the BKPay platform to complete the transaction. Upon successful payment, the Payment API receives a response from BKPay confirming the transaction and updates the user's account balance by adding the purchased pages.
- **Custom Print Template API:** Allows for the creation and management of custom print templates, enabling applications to generate dynamic print templates based on specific needs, used for Student, used for Student.
- **Printer Control API:** Allows applications to interact directly with the printer, like create Printer, used for SPSO.
- **Reporting and Statistics API:** Provides data on completed print jobs, print times, page counts, and other relevant information for monitoring and reporting on printing activities, used for Student.

Some main functions for these above API (see others in Section 4.2):

getPrinterName	Integer (ID)	Pass the printer ID to get the information of printer's brand from the database
getPrinterBrand	Integer (ID)	Pass the printer ID to get the information of printer's name from the database
getStudentFullName	Integer (ID)	Pass the student ID to get

		the information of the student's name from the database.
getAllPrinters	Integer(ID)	Pass the LocationID to get all the information of the printers in this location.
getAccount	Integer(ID)	Pass the StudentID to get all the information of the account of the student.
checkNamePassword	String(name), String(password)	Pass the username and password to authenticate create token for this user
createPrinter	String(name), String(brand), String(description)	Create printer
getPrintingLog	Integer(ID)	Pass the StudentID to get all the information of the printing log of the student.
createPrintingRequest	MultipartFile(file), Integer(copies)	Create PrintingRequest

4.2 Component Diagram

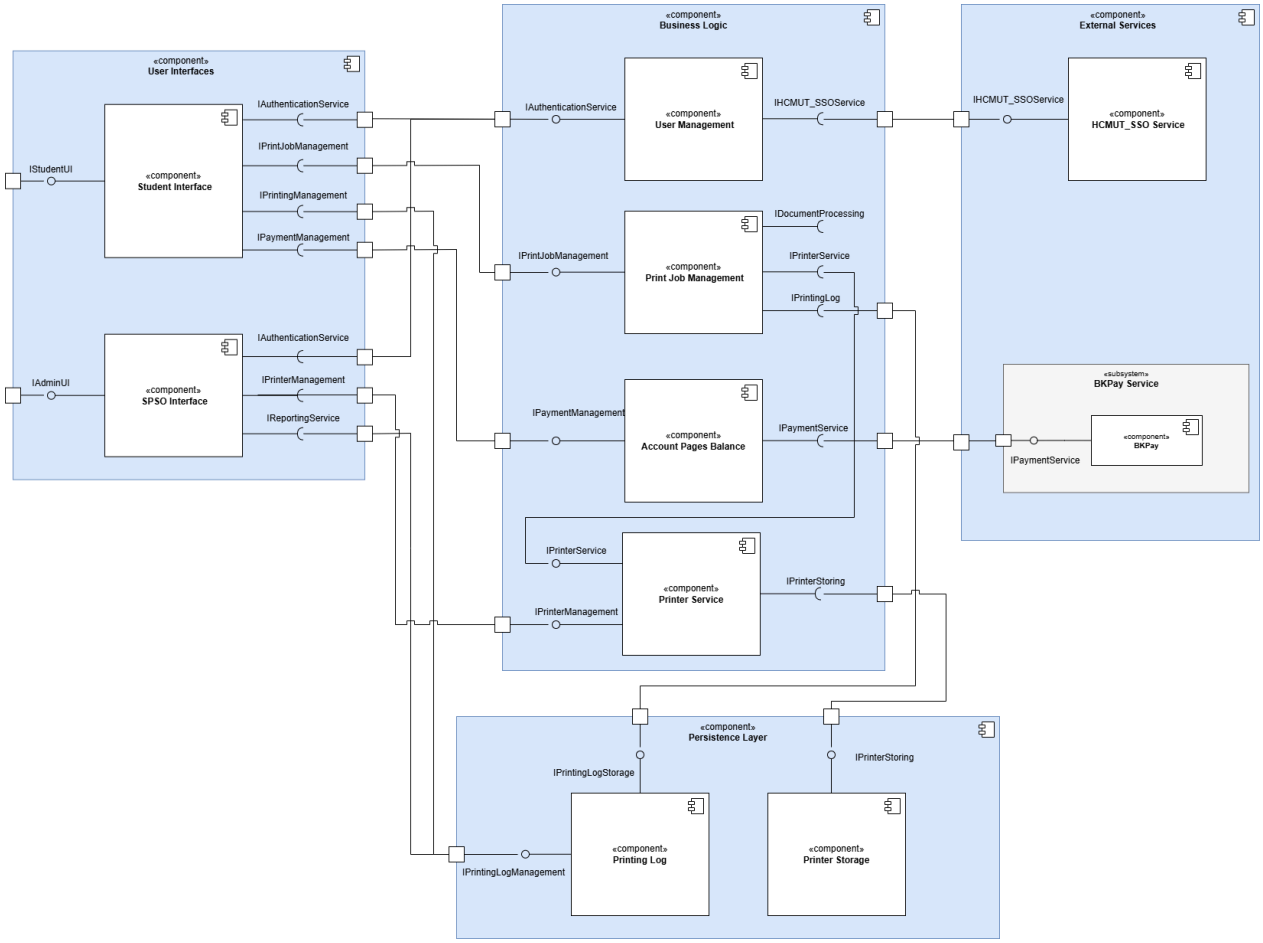

Figure 17. Component Diagram design for Printing module

4.2.1. User Interface Components

This component is designed for students to interact with the Smart Printing System (SPSS). It focuses on enabling features like viewing available printers, uploading documents, managing print jobs, and handling payments.

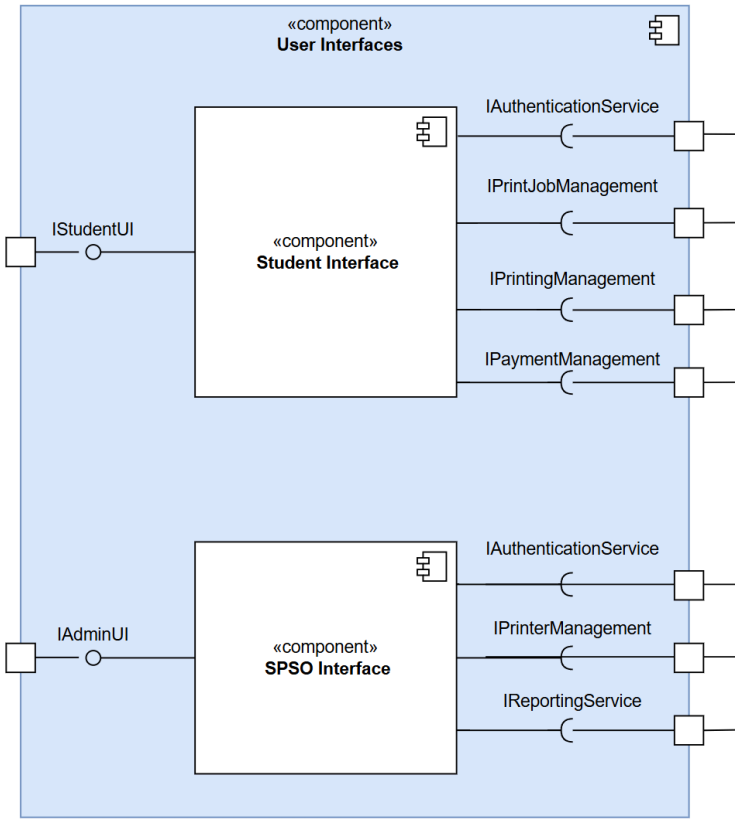

Figure 18. User Interface Components

4.2.1.1 Student Interface Component

The component diagram illustrates the user interface layer of the system. It consists of two primary components: the Student Interface and the SPSO Interface. The Student Interface serves as the entry point for students, providing access to services such as authentication, print job management, and payment management. Similarly, the SPSO Interface caters to administrative users, offering functionalities for authentication, printer management, and reporting services.

Provided Interface:

- **IStudentUI:** Allows students to interact with the system's key features for managing printing tasks. Includes methods:
  - displayPrinterLocations() - Displays a list of available printers and their locations, enabling students to find nearby printers.
  - uploadDocument() - Allows students to upload files to be printed.
  - selectPrinter() - Enables students to choose a specific printer for their print jobs.

- configurePrintOptions() - Provides options for configuring print settings, such as paper size, orientation, and the number of copies.

Required Interfaces:

- **IAuthenticationService:** Handles user authentication and secure access to the system. Methods:
  - authenticateUser(username, password) - Verifies student credentials for secure login.
  - generateAPIToken() - Generates an API token to maintain session security for subsequent requests.

- **IPrintJobManagement:** Manages the creation, tracking, and cancellation of print jobs. Methods:
  - createPrintingRequest(file, copies) – Initiates a print job request by sending the uploaded document and the desired number of copies.
  - trackPrintJobStatus() – Monitors the status of submitted print jobs (e.g., pending, completed).
  - cancelPrintJob() – Cancels a print job if required.

- **IAccountManagement:** Manages student account details, including balance updates and profile details. Methods:
  - getAccountDetails(studentID) – Retrieves information about the student's account, such as personal information, printing history.
  - updateAccountBalance(studentID) – Updates the student's balance when payments are made or refunds are issued.

- **IPaymentManagement:** Handles payments for printing services and account balance checks.
  - processPayment(amount) – Processes payments for printing services.
  - checkBalance(studentID) – Checks the student's account balance to ensure sufficient funds for printing tasks.

4.2.1.2 SPSO Interface Component

This component is intended for SPSO (administrative users) who manage printers, generate reports, and oversee system operations.

Provided Interface:

- **IAdminUI:** Allows administrative users to monitor and manage the printing system effectively.

- displayDashboard() – Displays an overview of system statistics, printer statuses, and user activities.
- managePrinters() – Enables admins to add, remove, and update printer settings.
- generateReports() – Allows the generation of reports on printer usage, revenue, and system performance.

Required Interfaces:

- **IAuthenticationService:** Ensures secure login and user access for administrators.
  - authenticateUser(username, password) – Verifies admin credentials for secure login.
  - generateAPIToken() – Provides API tokens for session security and authorization.
- **IPrinterManagement:** Manages printer configurations and settings.
  - createPrinter() – Adds new printers to the system.
  - getPrinterDetails(printerID) – Retrieves detailed information about specific printers.
  - updatePrinterSettings(printerID) – Updates printer configurations, such as default paper size or printer status.
- **IReportingService:** Provides reporting features to track usage and generate analytics. Methods:
  - generatePrintReport(dateRange, studentID, printerID) – Generates a report for printing activities within a specified date range, for all or some printers, or for students,...
  - exportPrintReport(format) – Exports reports for documentation or analysis.

4.2.2. Business Logic Components

The component diagram represents business logic which manages user authentication, print jobs, account balances, and printer operations. The User Management component handles user authentication and integrates with a Single Sign-On (SSO) service to streamline access. The Print Job Management component oversees the submission, processing, and logging of print jobs, ensuring smooth document handling and tracking. The Account Page Balance component manages users' balances, adjusting them based on print activity and payment interactions. Lastly, the Printer Service component facilitates communication with physical printers, managing job storage and ensuring efficient operation.

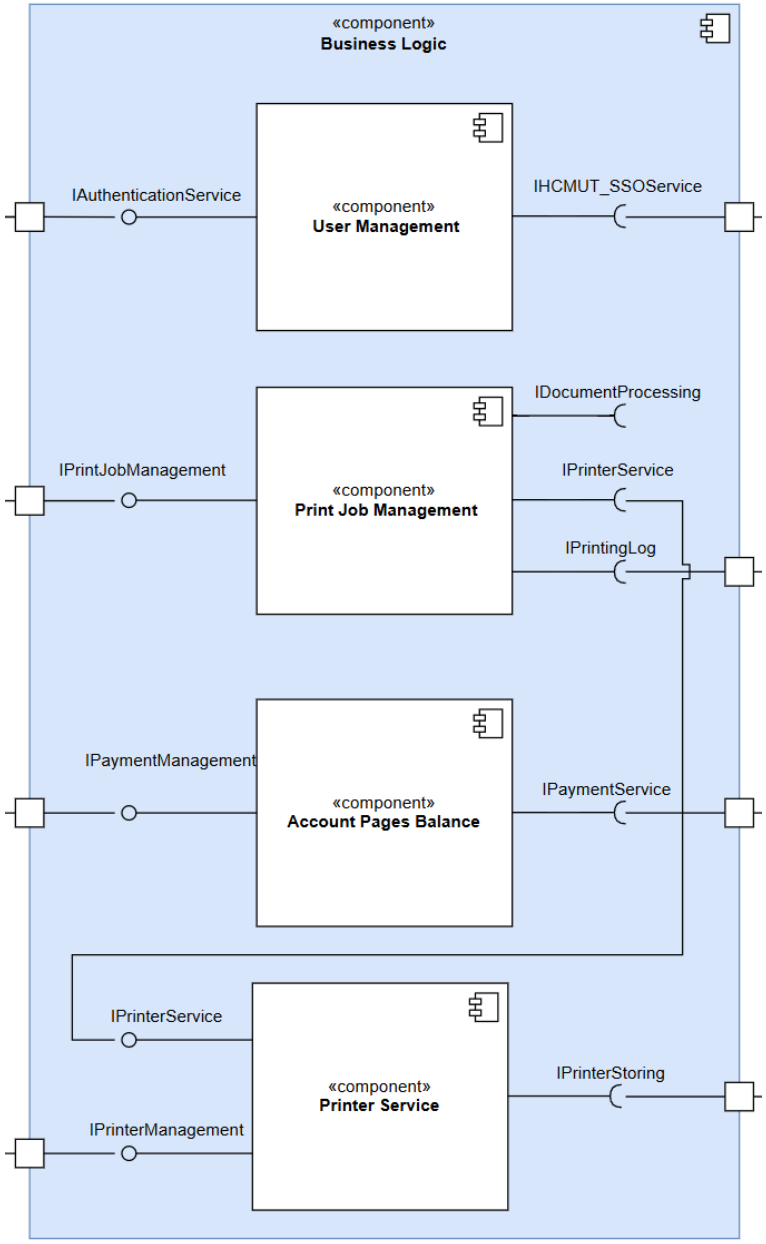

Figure 19. Business Logic Components for Printing module

4.2.2.1 Print Job Management Component

Manages the creation, processing, and tracking of print jobs. It interacts with document processing, printer services, account validation, and logging components to ensure efficient handling of print requests.

Provided Interface:

- **IPrintJobManagement:** Enables functionalities related to print job submission and tracking. Includes methods:
  - createPrintingRequest(file, copies) – Submits a new print job request, including file details and the number of copies required.
  - schedulePrintJob(jobDetails) – Schedules the print job based on printer availability and specified settings.
  - trackPrintJobStatus(jobID) – Retrieves the current status of a print job (e.g., pending, processing, completed).
  - cancelPrintJob(jobID) – Cancels a print job if it is no longer needed or there is an issue.

Required Interfaces:

- **IDocumentProcessing:** Processes and formats documents before submitting them to the printer.
  - processDocument(file) – Prepares the uploaded document for printing, handling compatibility and layout adjustments.
  - formatDocument(file) – Applies required formatting, such as resizing or adding margins, to match printer settings.

- **IPrinterService:** Interacts with physical printers to submit and execute jobs.
  - createPrinter(name, brand, description) – Registers a new printer in the system.
  - getPrinterName(ID) – Retrieves the name of a specific printer.
  - getPrinterBrand(ID) – Fetches the brand of a specified printer.

- **IPrintingLog:** Stores and tracks print job activities for auditing and reporting purposes. Includes methods:
  - storeLog(jobID, studentID) – Logs details of print job creation and processing.
  - updateLog(jobID, studentID) – Updates existing logs, such as when a job status changes or is canceled.

4.2.2.2 Printer Service

Facilitates communication with physical printers for managing print jobs and configurations.

Provided Interface:

- **IPrinterService:** Provides printer-related functionalities to register and retrieve printer details. Includes methods:
  - createPrinter(name, brand, description) – Adds a new printer to the system.
  - getPrinterName(ID) – Fetches the name of a specified printer.
  - getPrinterBrand(ID) – Retrieves the brand of a printer by ID.
  - getAllPrinters(locationID) – Lists all printers available in a specific location.

- **IPrinterManagement:** Focuses on managing printer configurations and maintenance. Include methods:
  - createPrinter() – Registers a new printer in the database.
  - getPrinterDetails(printerID) – Fetches detailed information about a specific printer.
  - updatePrinterSettings(printerID) – Updates settings like default paper size, margins, or duplex printing options.

Required Interfaces:

- **IPrinterStoring:** Manages the storage of print job files, ensuring data consistency and availability. Includes method:
  - storeFile(file) – Saves the print job file in storage.
  - updateFile(file) – Updates stored files, such as adding metadata or changing formats.
  - deleteFile(file) – Deletes stored files when no longer needed.

4.2.2.3 Account Page Balance

Handles student balances, including payments and balance inquiries, ensuring funds are available before processing print jobs.

Provided Interface:

- **IPaymentManagement:** Manages payment processes and balance checks.
  - processPayment(amount) – Processes payment transactions based on printing costs.
  - checkBalance(studentID) – Verifies if a student's page balance is sufficient for a print job.

Required Interface:

- **IPaymentService:** Connects to external payment processing systems for transactions.
  Methods:
  - processPrintPagePayment(amount, studentID) – Deducts the required amount from the student's account for the print job.
  - checkAccountBalance(studentID) – Checks the current balance of the user's account associated with the given studentID. Used to verify that the student has sufficient funds before processing payments.

4.2.2.4 User Management Component

Handles user authentication and integrates with the HCMUT SSO service for Single Sign-On authentication.

Provided Interface:

- **IAuthenticationService:** Secures user access through authentication and token generation. Includes methods:
  - authenticateUser(username, password) – Validates the username and password entered by the user.
  - generateAPIToken() – Generates an API token that is used for secure session management.
  - validateAccessRights(token) – Ensures that the provided token has sufficient permissions for the requested action.

Required Interfaces:

- IHCMUT_SSOService: Communicates with the external HCMUT SSO service for authentication.
  - checkNamePassword(name, password) – Sends the username and password from the SPSS system to the HCMUT_SSO service for validation. Upon successful validation, the service generates a secure API token for the user, which will be used for subsequent actions within the system.
  - validateToken(token) – Verifies the validity of an API token issued by the HCMUT_SSO service, ensuring that only authorized users can access system resources and functionalities.

Note that the checkNamePassword method is specifically designed to authenticate user credentials and issue a secure token. This token is then used by the SPSS system to authorize future requests and operations.

4.2.3. External Services Components

This section illustrates the integration of external services into the system, focusing on authentication and payment functionalities.

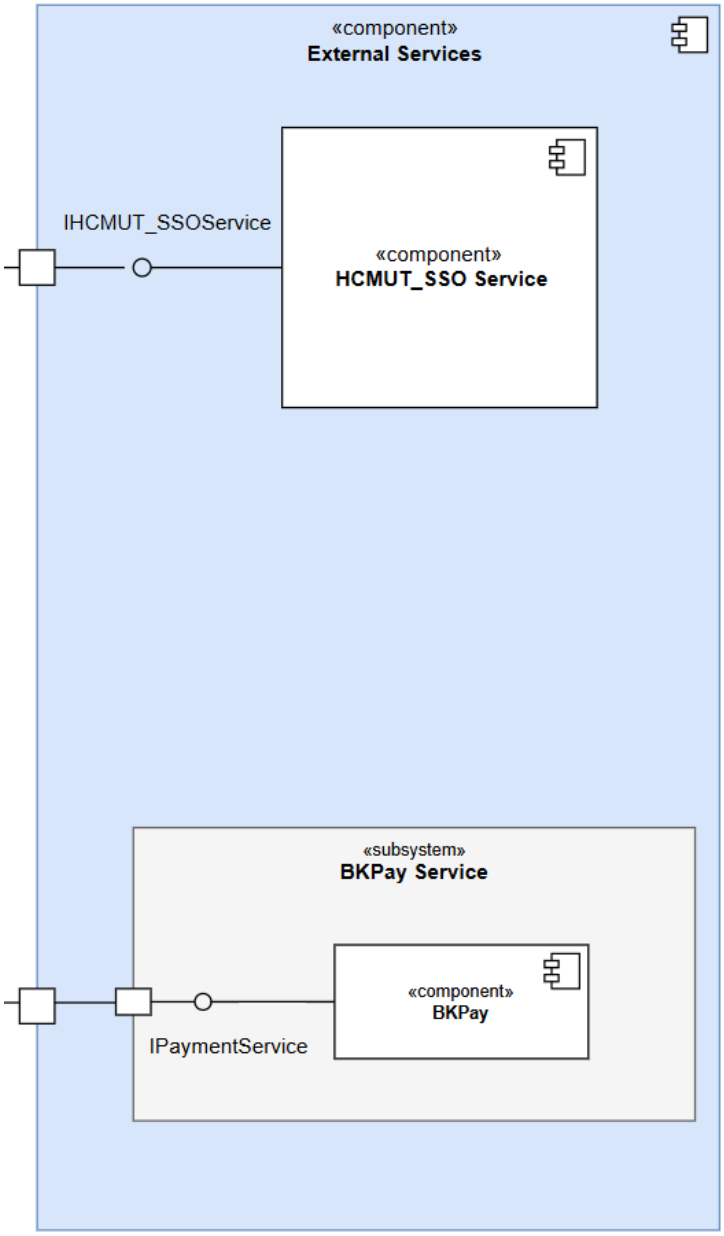

Figure 20. External Services Components

- **HCMUT SSO Service:** Handles user authentication via a centralized Single Sign-On (SSO) mechanism, enabling secure and seamless access to the system.
- **BKPay Service:** Manages payment processing, ensuring secure transactions and smooth financial interactions related to printing activities.

4.2.3.1 HCMUT_SSO Service

Provided Interface:

- **IHCMUT_SSOService**
  - checkNamePassword(name, password) – Validates the provided username and password through the HCMUT_SSO service. Upon successful authentication, a secure token is generated for later use.
  - validateToken(token) – Verifies the validity of a token issued by the HCMUT_SSO service to ensure that only authenticated and authorized users can access system functionalities.

4.2.3.2 BKPay Service

Provided Interface:

- **IPaymentService**
  - processPrintPagePayment(amount, studentID) – Processes payments based on the number of printed pages, deducting the specified amount from the student's account.
  - getPaymentHistory(studentID) – Retrieves transaction history related to the purchase of print pages for the specified student's account. Then return transaction IDs, timestamps, and amounts related to purchasing print pages.

4.2.4. Persistence Layer Components

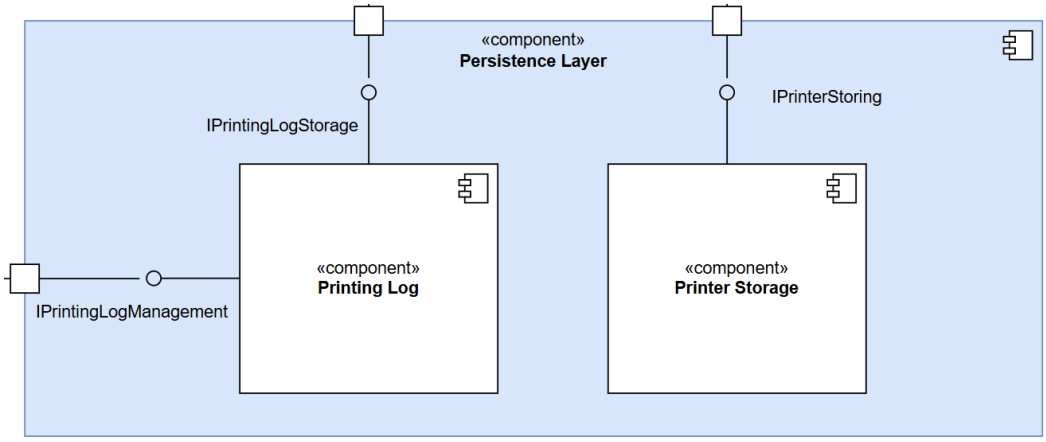

Figure 21. Persistence Layer Components for Printing module

The component diagram represents the Persistence Layer of the system, focusing on data storage and management related to printing activities. The Printing Log component handles the storage and management of logs through the IPrintingLogStorage and IPrintingLogManagement interfaces, ensuring that all printing activities are accurately recorded and retrievable. The Printer Storage component, accessed via the IPrinterStoring interface, manages the persistent storage of printer-related data, such as queued print jobs or configuration settings.

4.2.4.1 Printing Log Component

Provided Interface:

- **IPrintingLogManagement:** Manages the creation and export of print logs for reporting purposes. Includes methods:
  - generatePrintReport(dateRange, studentID, printerID) - Get all printing history within a specified date range for a given student and printer or (all students or all printers).
  - exportPrintReport() - Exports the generated report.
- **IPrintingLogStorage**
  - storeLog(jobID, studentID) - Saves details of a specific print job associated with a jobID and studentID.
  - updateLog(jobID, studentID) - Updates the details of an existing print log entry for a specific job and user.

4.2.4.2 Printer Storage Component

Provided Interface:

- **IPrintingLogManagement:** Manages persistent storage for printer data, including configuration files and queued jobs. Includes methods:
    - storeFile(file) - Stores a file in the system, such as queued print jobs or printer settings. Additionally, saves configuration or print-related files securely in storage.
    - updateFile(file) - Updates existing stored files with new information or modifications.
    - deleteFile(file) - Deletes a specified file from storage when it is no longer needed.

5. Task 4: Implementation - Sprint 1

5.1 Setting up an online repository

Github repo: [CO3001_Software-Engineering-N06]

5.2 Adding materials

Documents, materials, and folders for requirement, system modeling and architecture design have been added to the “docs” folder.

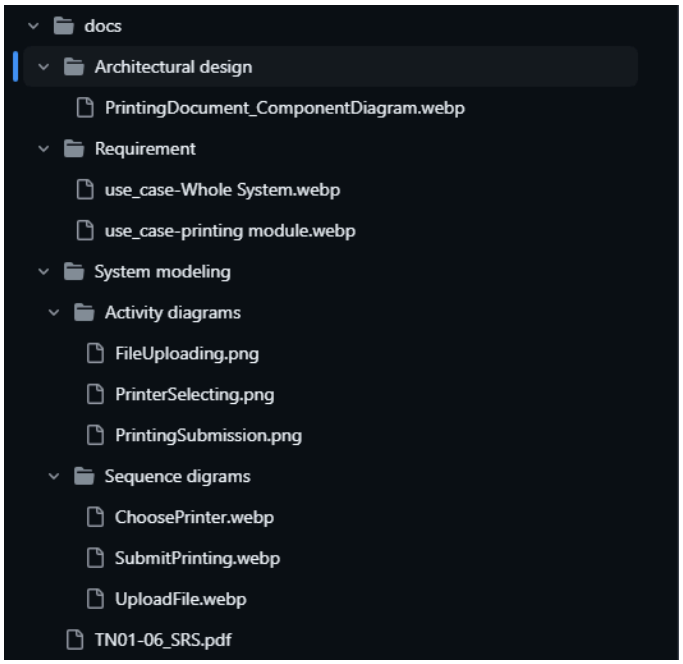

Figure 22. Repository Structure

5.3 Usability test

5.3.1 Overview

We conducted a usability test with the user interface developed in Figma above. The test was conducted remotely with the participation of other team members in the class. My team recorded solutions, task completion rates, comments, overall evaluations, questions, and feedback from all members.

5.3.2 Participants / Testers

Fullname	Note
Võ Hoàng Huy	Member of team 8
Bùi Văn Quốc Bảo	Member of team 1
Nguyễn Quang Huy	Member of team 5
Lê Văn Anh Khoa	Member of team 3

5.3.3 Test Strategy

Our test strategy uses a remote, unmoderated approach focused on qualitative analysis. This flexible method allows participants to interact with the interface at their convenience, without real-time facilitation, capturing a natural user experience. Tasks are designed to assess user satisfaction, perceived usability, and overall experience. Recorded interactions will be analyzed for insights into user behavior, preferences, and challenges, providing valuable data to improve interface usability and intuitiveness.

The test participants will attempt to perform the following tasks and record the completion time:

1. Successfully log in to the SPSS application
2. Search for and select the desired printer to use
3. Upload a file, configure print settings, and send a print request
4. Review the document print history
5. View account balance and page purchase history

6. Submit a printer error report
7. Successfully log out of the SPSS application

After completing the final task, participants provide an overall evaluation of the system using a 5-point scale, ranging from very poor to very good, based on the following factors:

- Ease of use
- Attractive interface
- Layout design
- Searchability and navigation

Additionally, participants are asked to answer the following questions:

- Which part of the application did they like the most?
- Which part of the application did they like the least?
- Suggestions for improving the application

5.3.4 Result

- Completion rate:

Name	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7
H. Huy	x	x	x	x	x	x	x
Bảo	x	x	x	x	x	x	x
Q. Huy	x	x	x	x	x	x	x
Khoa	x	x	x	x	x	x	x
Complete rate	100%	100%	100%	100%	100%	100%	100%

- Completion time (unit: second):

Task / Name	H. Huy	Bảo	Q. Huy	Khoa	Average
1	10	7	8	5	6.67
2	21	7	21	17	16.5

3	23	21	15	19	19.5
4	5	3	3	4	3.75
5	3	6	4	4	4.25
6	15	10	13	11	12.25
7	5	8	7	7	6.75

- Overall evaluation:

Factor	Very poor	Poor	Normal	Good	Very good
Ease of use			Bảo Khoa	Q. Huy H. Huy	
Attractive interface				Bảo Q. Huy	H. Huy Khoa
Layout design				All participants	
Searchability and navigation				All participants	

- Personal opinion:
    - The most liked feature:
        - The design of the account information page
        - The design of the account balance page
    - The least liked feature:
        - None

5.3.5 Suggestions for improving the system

Change	Explanation	Priority level
Add a "Remember Me" button	Save the account details for easier login in the future	Medium
Change the font style	The current font may not align with the formal and professional tone typically expected in a school setting	High

5.3.6 Conclusion

The results show that testers highly appreciated the creativity and user-friendly interface. Despite it being their first experience, the task completion times for testers were quite fast, indicating that the app is easy to use for newcomers. On the other hand, there are still a few design aspects that need to be changed to improve the user experience. Key feedback includes adding a “Remember Me” button to make the login process quicker and more convenient. Additionally, while the overall aesthetics were warmly received, there were suggestions to adjust the font to better suit the educational environment. These feedbacks are invaluable as they provide actionable insights into how we can develop the prototype into a sleeker and more user-friendly product. By addressing these, we can not only meet but exceed user expectations in future iterations.

6. Task 5: Implementation - Sprint 2

6.1 Technologies

- **NodeJS**: Node.js is a powerful, open-source runtime environment that allows developers to build scalable and high-performance applications using JavaScript. Built on Chrome's V8 JavaScript engine, Node.js enables server-side execution of JavaScript, making it possible to use a single programming language for both client-side and server-side development.
- **React**: React is a popular open-source JavaScript library developed by Meta (formerly Facebook) for building user interfaces, particularly for single-page applications. It allows developers to create reusable UI components that manage their own state, making it easier to build complex and interactive user interfaces.
- **ChakraUI**: Chakra UI is a modern, open-source component library for React that focuses on simplicity, accessibility, and flexibility. It provides a set of pre-designed, composable, and highly customizable components that help developers build responsive and visually appealing user interfaces quickly.
- **TailwindCSS**: Tailwind CSS is a popular utility-first CSS framework that simplifies the process of designing modern, responsive user interfaces. Instead of relying on pre-built components, it provides a set of low-level utility classes for us to write custom CSS for most tasks, speeding up development while maintaining flexibility.

- **MySQL on Azure:** It is a fully managed relational database service that provides high availability, security, and scalability in the cloud. Offered through Azure Database for MySQL, it allows us to easily deploy, manage, and scale MySQL databases without worrying about infrastructure maintenance.
- **Spring Boot**: Spring Boot is an open-source Java-based framework used to build stand-alone, production-ready applications with minimal setup and configuration. It simplifies the development process by providing a set of defaults for application settings and configurations, allowing developers to focus on writing business logic rather than dealing with boilerplate code.
- **Postman**: Postman is a popular API development and testing tool that simplifies the process of creating, testing, and managing APIs. It provides an intuitive user interface for sending HTTP requests to a server and receiving responses, which helps developers quickly build and troubleshoot APIs.

6.2 MVP2 Screens

This section demonstrates how our website works by going through each page of it.

6.2.1 Landing Page

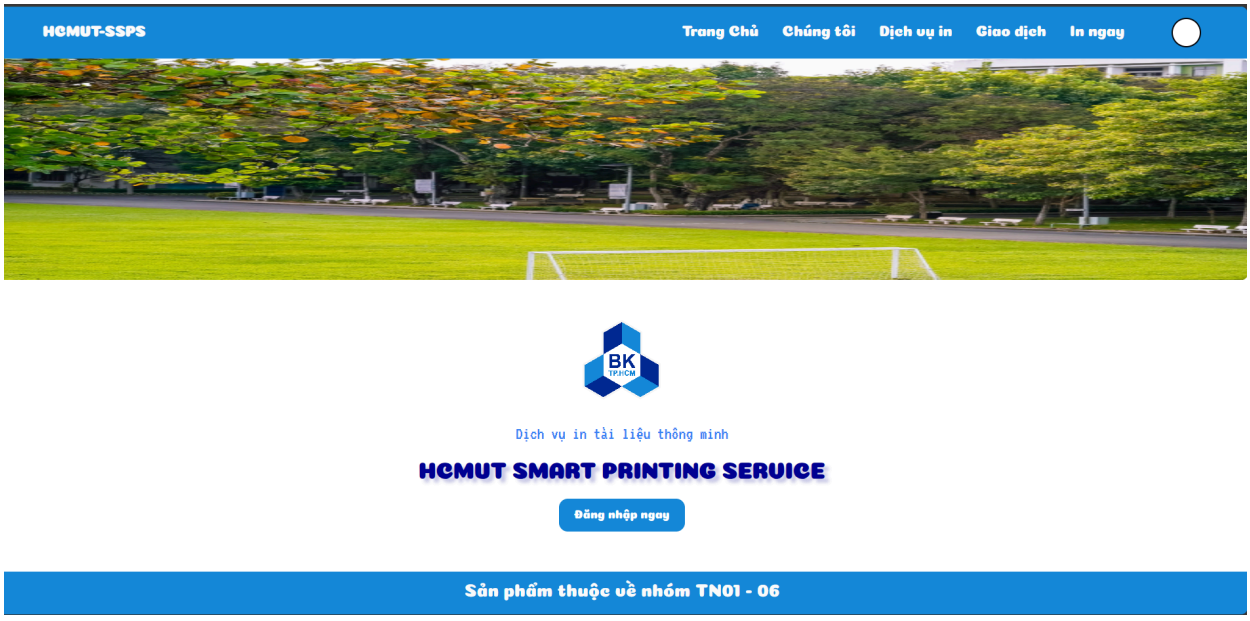

Figure 23. The site's landing page

When first entering the website, users will encounter this page. The page mainly displays the "Đăng nhập ngay" button allowing users to login to the website. But first, in

the header there are also many options such as “Trang chủ” - the current page, “Chúng tôi”- briefly introduce the project and the team information and more options (but since the user is not logged in so those buttons will direct the users to the login page as below).

6.2.2 Login Page

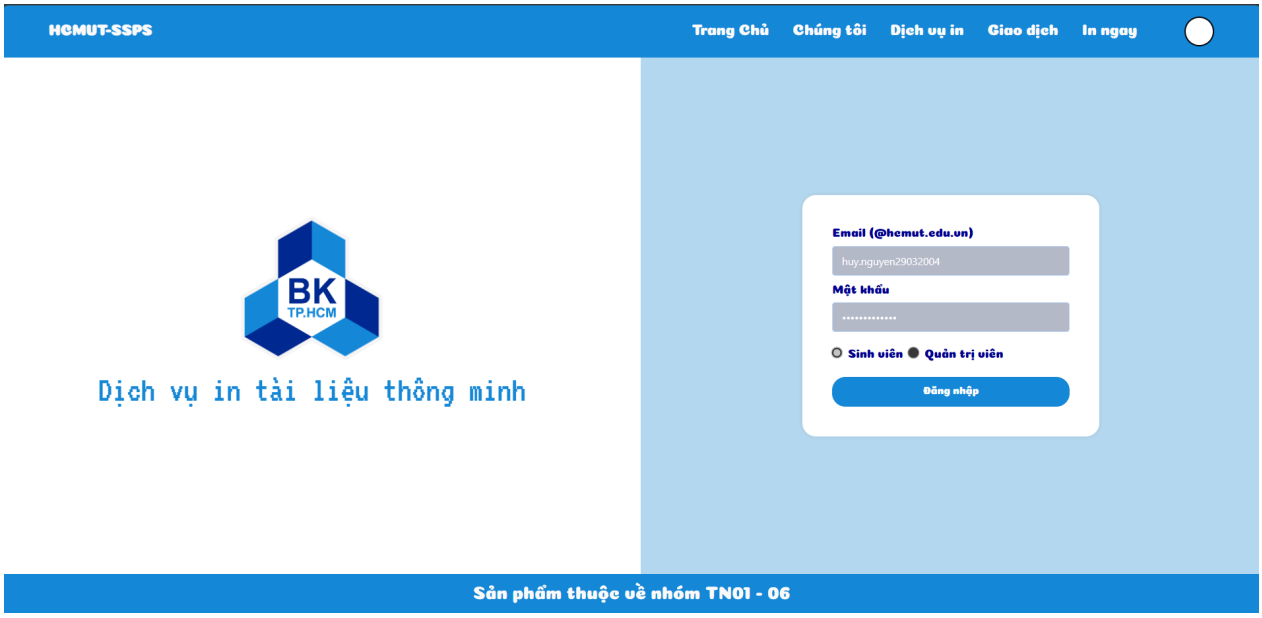

Figure 24. The site’s login page

This page allows the user to enter their actual HCMUT account and password (the one given when entering the university) to login to the site.

The page also has the option to login as an administrator, but since we only implemented modules for the students, those buttons are disabled by now waiting for further implementation of administration modules.

6.2.3 User’s Homepage

After successfully login to the site, the user’s homepage will be displayed, showing the student information including their email address, major, department, and the current printing page available.

On the right hand side of the homepage, there are 2 charts showing statistics on the number of times printed and the number of pages printed during the year.

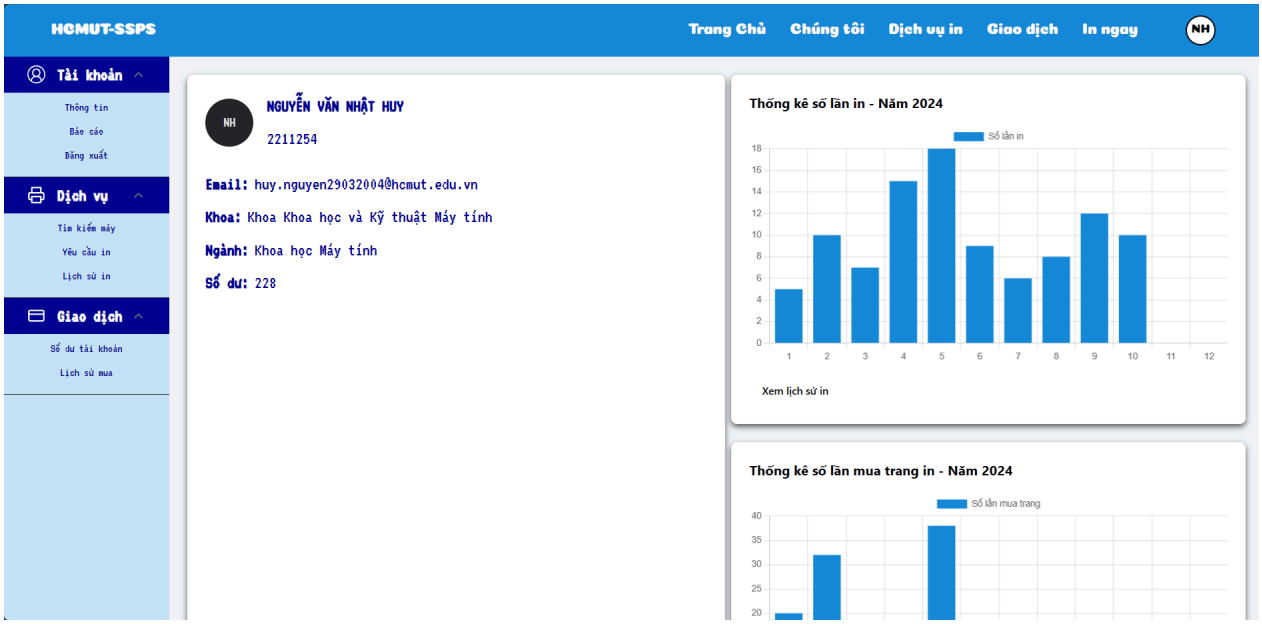

Figure 25. The user's homepage

6.2.4 Finding printer page

Figure 26. The finding printer page

By clicking on the "Tìm kiếm máy " button on the sidebar, users are able to choose a printer to print their documents. The picture above shows all the printers available, but users can also search for printers by entering its name or choosing the building where the printers are located at or by its status (available or stopped).

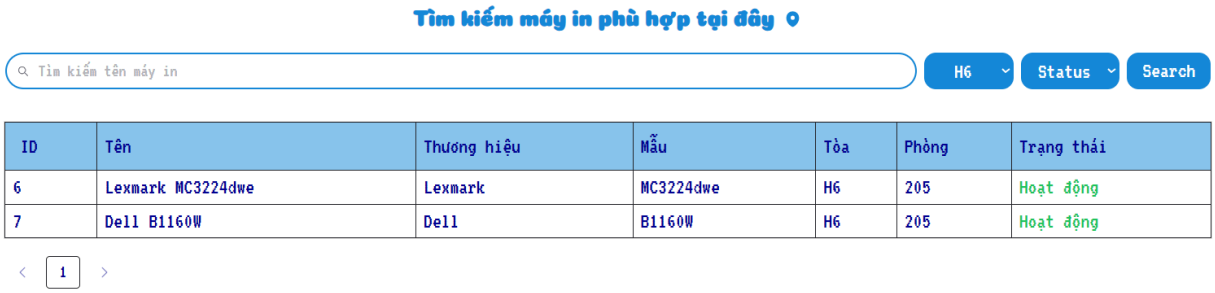

Figure 27. Printer list after choosing ones that at “H6” building

Furthermore, each row in the table is clickable, so when a user chooses a printer that suits their need, the website will automatically go to the print page.

6.2.5 Print page

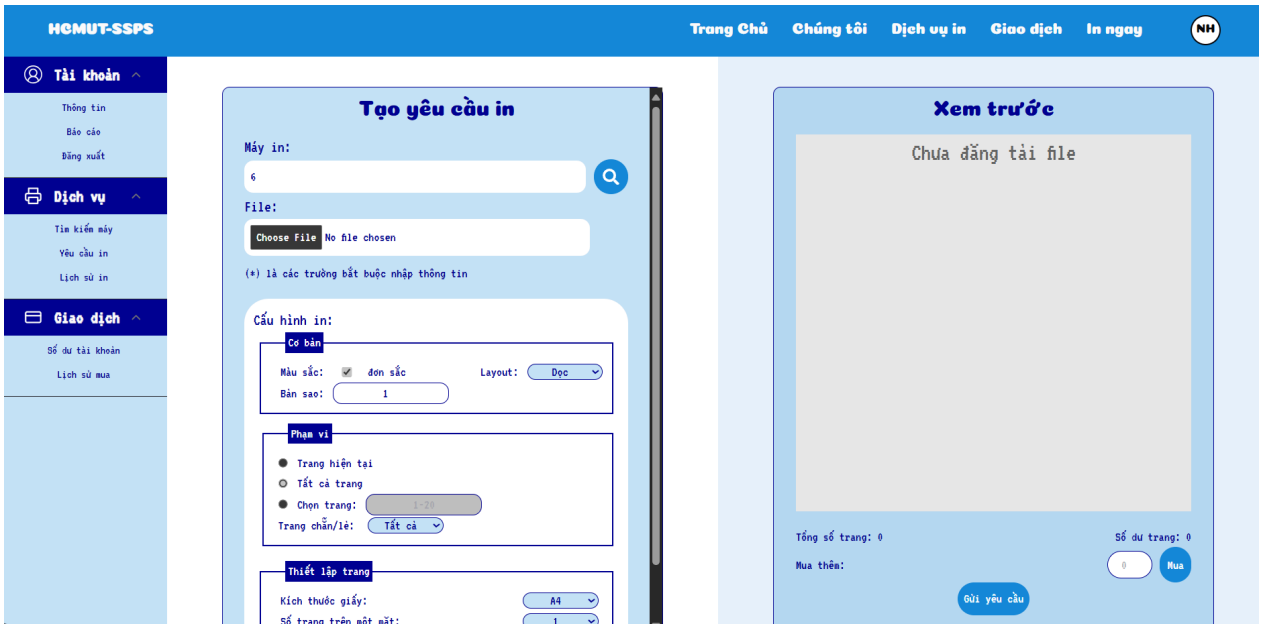

Figure 28. The print page having printer id “6” as chosen printer from the previous page

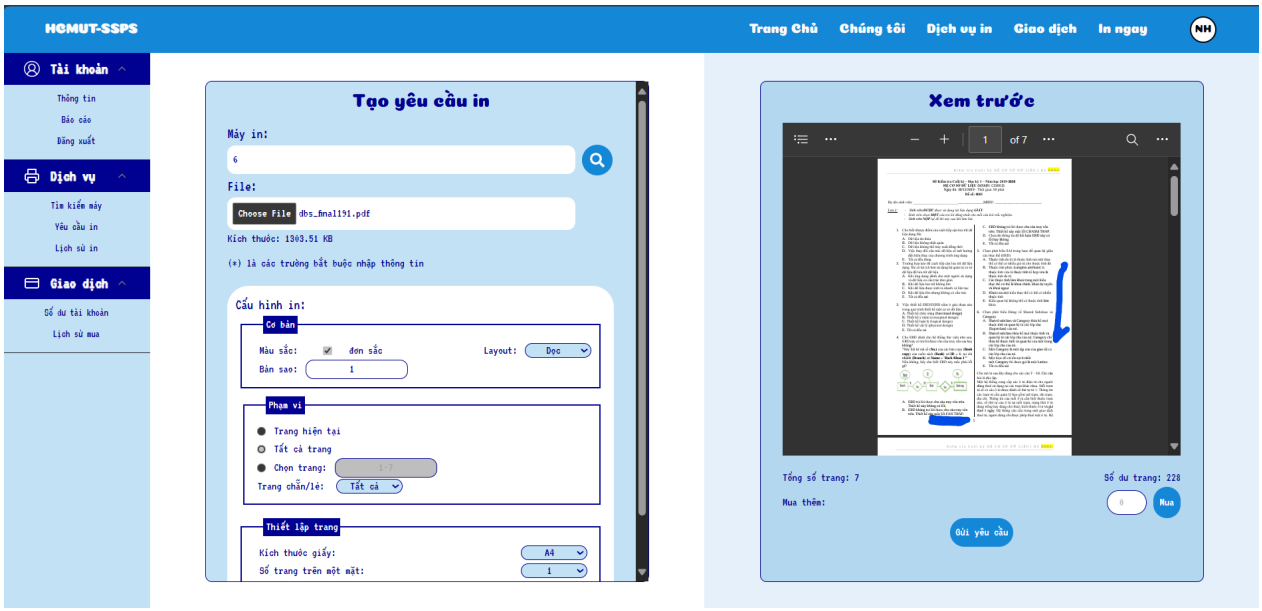

Figure 29. The print page after the user submitting a new file

As shown above, the preview of the file is displayed on the right side of the page. On the other side, there are options to configure the document before printing such as paper type (A4, A3, …), page layout (portrait, landscape), … but due to limitations of the library not supporting changes in the preview, it is not possible for us to display the true preview of the document after configuration (only showing the default preview).

At the bottom right corner of the preview, it is shown that the document is 7 pages long (as we choose to print all the document - choosing “Trang hiện tại” will result in 1 instead of 7) and the current page the user has is 228. Now after everything is set and ready, we click “Gửi yêu cầu” to order the printer to print the document.

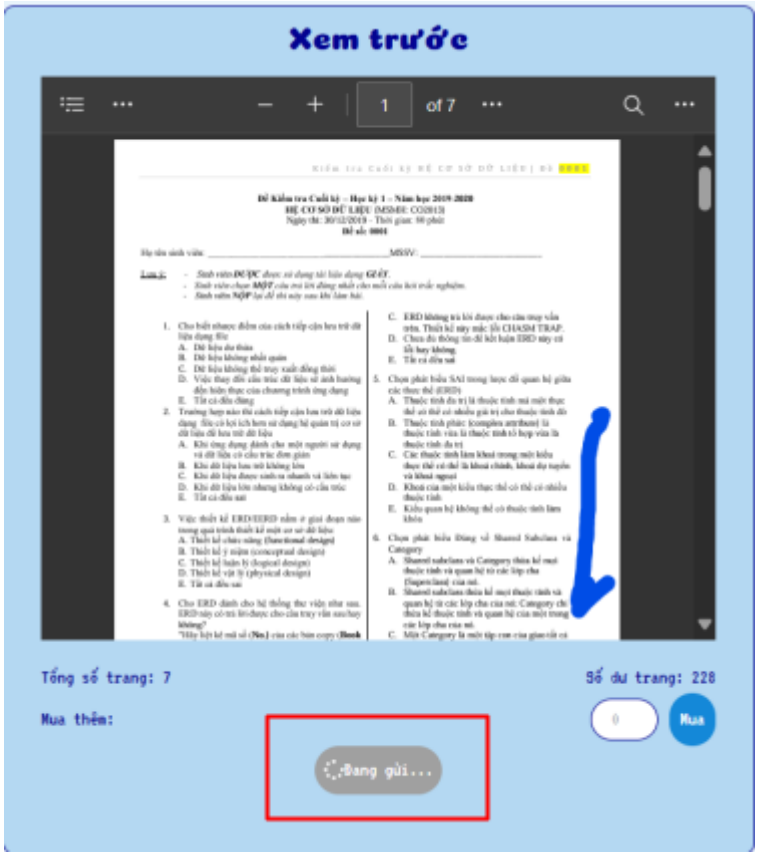

Figure 30. The request is being processed.

After clicking the "Gửi yêu cầu" button, the website will make the user wait for about 10 second (demonstrating real printing action) for the request to be completed. During this time, the printer status will be changed to "Đang hoạt động".

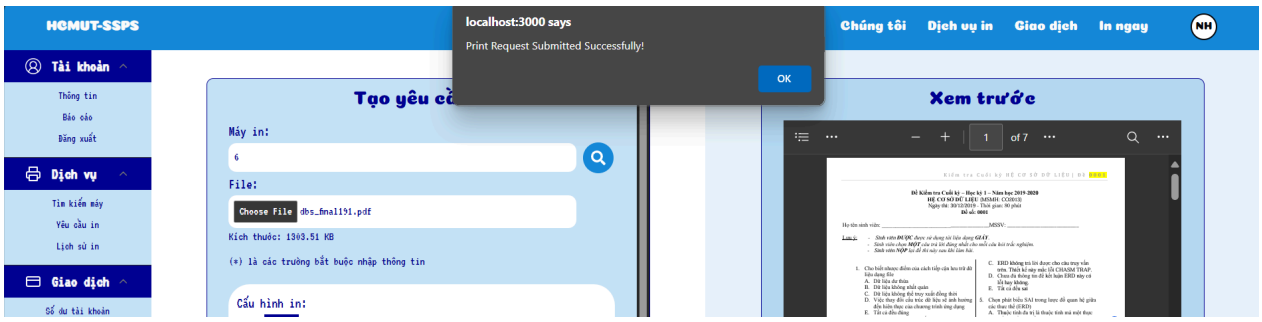

Figure 31. A message informing the user of the successful request.

6.2.6 Printing history page

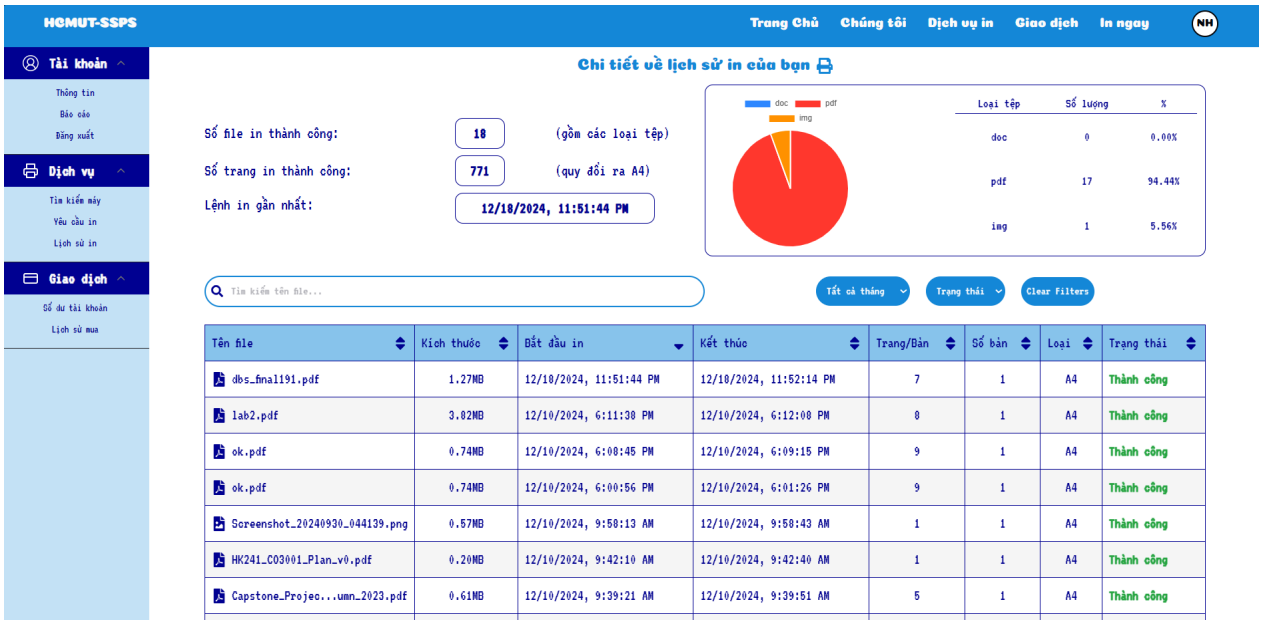

Figure 32. The printing history page

This page shows all printing history of the user, in the first record of the table can be seen the file we just uploaded. All the history records can be searched by the file name, by month or by the printing status (success or failed).

6.2.7 Account balance page

This page shows information about the student's current page available, all the pages he/she has bought,...or the student can buy more pages here (insert the valid number of pages to buy - 3 pages, and click “Mua” will increase the page available).

On the right side can be seen the printing history and the transaction history, the top most record of 2 tables shows the most recent activity of the user.

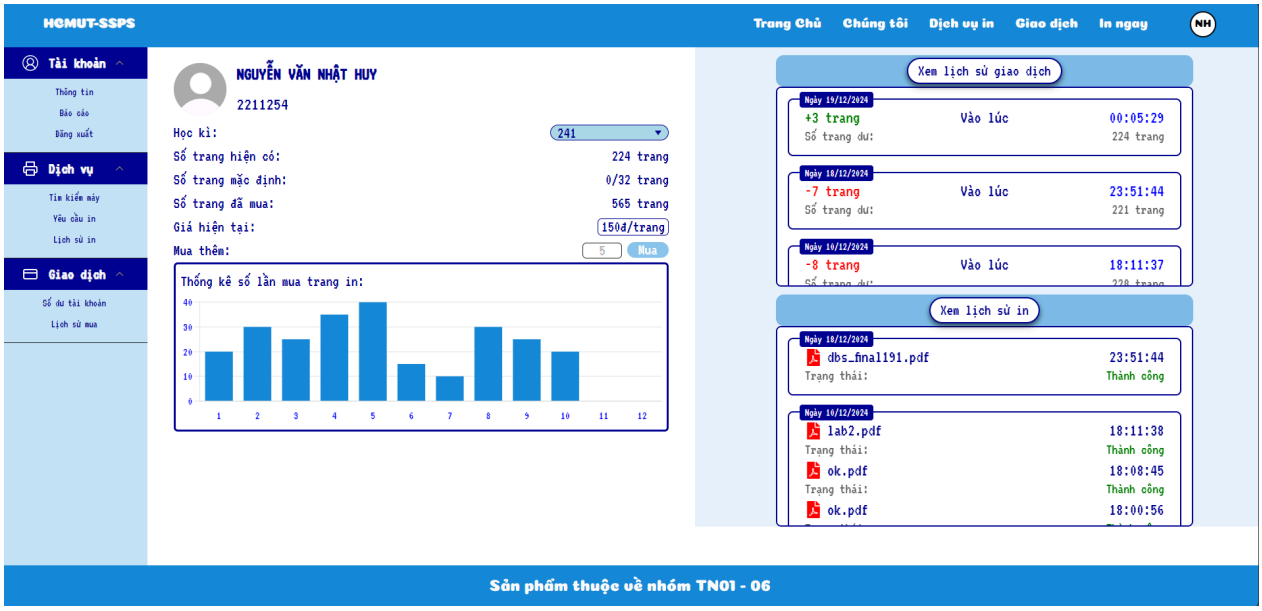

Figure 33. The account balance page

6.2.8 Transaction history page

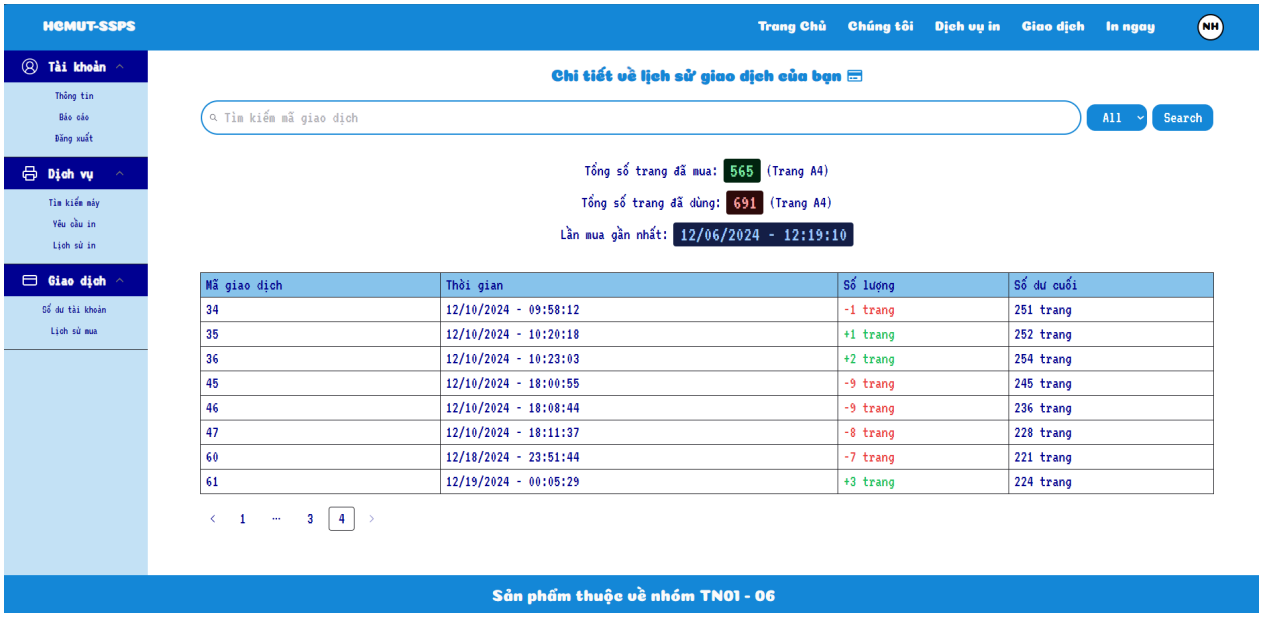

Figure 34. The transaction history page

This page displays all the transactions the user made that changes the current page number. We just print a 7 page document and buy another 3 pages in the account balance page so there are 2 new transactions which are -7 pages and +3 pages.

6.2.9 Logout

When clicking the "Đăng xuất" button in the sidebar, a message is displayed to confirm the user's choice, clicking "OK" will direct the user to the landing page and logged out of the server.

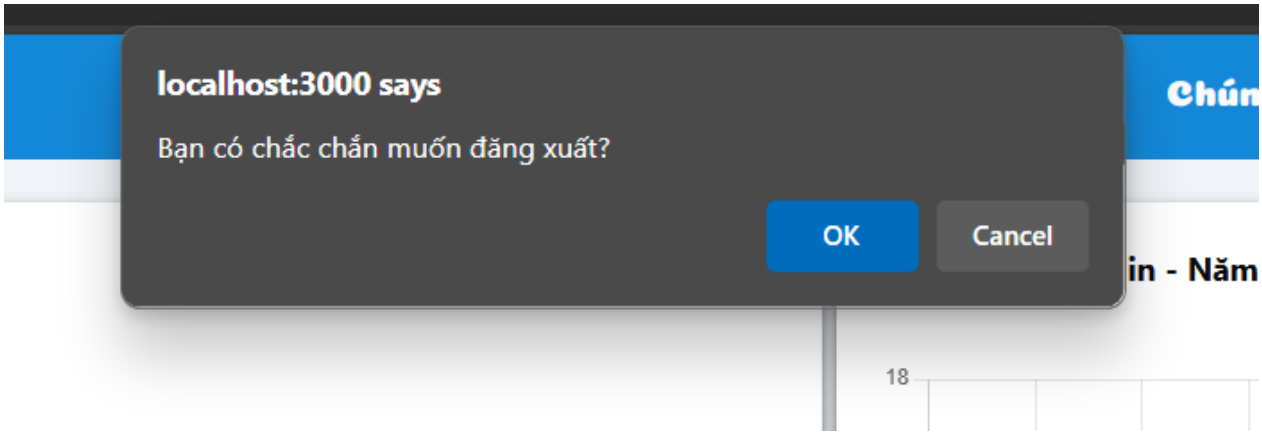

Figure 35. A message displayed to confirm user's choice

6.3 Resources and Instructions

6.3.1 Resources

Source Code: CO3001_Software-Engineering-N06

Presentation: TN01-06_Presentation.pdf

6.3.2 Instructions

- **Step 1: Install Required Tools**

  Before you start, make sure you have the following tools installed on your machine:

  - **Java Development Kit (JDK)**: Install JDK 11 or newer.
    + Download from: AdoptOpenJDK or Oracle JDK.
    + After installation, check by running java -version and javac -version in your terminal.

  - **Maven (Optional, for Maven projects)**: If your project uses Maven as a build tool, you'll need Maven.
    + Download from: Maven.
    + Check installation with *mvn -v* in the terminal.

+ However, it is possible to run a **Spring Boot** backend without manually installing **Maven** on your machine. Most modern IDEs (such as IntelliJ IDEA, Eclipse, or Visual Studio Code) can manage Maven dependencies automatically, so you don't need to install Maven manually.

- **IntelliJ IDEA (Maven Integration)**
  + Open your project in IntelliJ IDEA.
  + If your project uses Maven, IntelliJ IDEA will automatically detect the pom.xml file and download the necessary dependencies.
  + You can run the Spring Boot application directly from the IDE by:

    Right-clicking on the main application class (the one with @SpringBootApplication annotation).

    Selecting Run.
  + This way, IntelliJ takes care of downloading the dependencies (like Maven would do) and runs the application without you having to manually install Maven.

- **Eclipse (Maven Integration)**
  + Open your project in Eclipse.
  + Eclipse will automatically detect and resolve Maven dependencies from the pom.xml file when you open the project.
  + You can run the Spring Boot application by:

    Right-clicking on the main class (the one with @SpringBootApplication annotation).

    Selecting Run As > Java Application.

- **Visual Studio Code (VS Code)**
  + Open the Spring Boot project in VS Code.
  + Install extensions like **Spring Boot Tools** and **Java Extension Pack**.
  + VS Code will automatically download the dependencies listed in pom.xml when you open the project.
  + You can run the application from the integrated terminal in VS Code using the **Spring Boot** plugin or by using **Maven commands** in the terminal.

- **Step 2: Clone the Repository from GitHub**
  - Go to the GitHub repository of the project.
  - Copy the repository URL CO3001_Software-Engineering-N06
  - Open a terminal and navigate to the folder where you want to store the project.
  - Clone the repository with the following command:

git clone https://github.com/username/project-name.git

- **Step 3: Run the Backend (Spring Boot)**
  - Navigate to the backend directory in your project folder:
    cd path/to/your/project/backend
  - Run it
  - The Spring Boot application will usually run on port 8080 by default. You can check by accessing the API at http://localhost:8080.

- **Step 4: Run the Frontend (ReactJS)**
  - Open a new terminal and navigate to the frontend directory:
    cd path/to/your/project/frontend
  - Install the necessary dependencies for ReactJS using npm:
    npm install
  - To start the React development server, use the following command:
    npm start
  - The React app will usually run on port 3000 by default. You can access the frontend by going to http://localhost:3000.

7. Conclusion

7.1 Lessons Learned

Throughout the development of the HCMUT Student Smart Printing Service (HCMUT_SSPS), individual team members gained valuable insights and skills. Clear communication within the team and with stakeholders proved essential for ensuring alignment and smooth collaboration. Iterative prototyping and testing allowed the team to gather feedback and continuously refine designs to meet user expectations. Balancing technical complexity with usability was a key focus, enabling the delivery of a user-friendly experience without compromising functionality. Team members also developed proficiency in using modern design tools such as Draw.io and Figma, as well as coding frameworks like ReactJS and Spring Boot. Additionally, the project provided an opportunity to understand the end-to-end development process, from requirements gathering and modeling to implementation, testing, and deployment.

The team collectively learned the importance of collaboration and the effective use of GitHub for version control to track progress and maintain code integrity. Conducting early usability testing minimized design flaws and prevented costly changes later in the process. The ability to adapt quickly to challenges and incorporate feedback helped the team continuously improve the system and meet evolving requirements.

7.2 Achievements

Key accomplishments in this project include the successful description of the domain context, identification of stakeholders, and analysis of their needs, which laid the foundation for the project's scope and objectives. The team completed system design tasks such as use case diagrams, activity diagrams, sequence diagrams, and class diagrams, providing a clear visualization of processes and relationships within the system. High-fidelity MVPs were built with intuitive user interfaces to demonstrate core features and workflows effectively.

Architectural design decisions focused on implementing a layered architecture that ensured modularity, scalability, and ease of maintenance. A version-controlled repository was established and maintained using GitHub, facilitating collaboration and tracking development progress. Usability tests were conducted to gather feedback before development, ensuring the final product aligned with user expectations. The project was successfully implemented using ReactJS for the frontend, Java Spring Boot for the backend, and Azure Database for MySQL as the database solution.

7.3 Future Works

While the project has achieved its initial goals, several enhancements are planned to improve and expand the system. One key focus is developing a dedicated admin portal for printer management, generating monthly usage reports, and handling error logs. Enhancing the user interface and user experience (UI/UX) is also prioritized to make the system more intuitive and accessible across all devices.

Cross-device compatibility improvements will ensure seamless functionality across desktops, tablets, and mobile devices. Payment integration with BKPay is planned to process payments, manage transactions, and generate invoices efficiently. Performance optimization will involve implementing load balancing to evenly distribute traffic and improve scalability. Additionally, caching solutions like Redis and database query optimizations will be implemented to ensure faster data retrieval and system responsiveness.

7.4 Final Thoughts

In conclusion, this project provided a comprehensive and hands-on experience in building a real-world application. It combined software development practices with stakeholder-focused design principles, allowing the team to deliver a functional MVP while addressing user needs. The experience equipped the team with critical technical and project management skills, preparing them for future endeavors in software development and system design. Moving forward, the planned enhancements will further strengthen the system, ensuring it meets scalability, usability, and performance requirements.

Appendix

Appendix A: Team Members and Roles

No.	Fullname	Student ID	Task(s)	Percentage
1	Nguyễn Trương Thái Bảo	2210251	Tasks 1.1, 1.3.1, 2.3, 3.1, Task 5	100%
2	Nguyễn Truyền Tuấn	2213795	Task 1.2, 1.3.1, 2.4, 3.2, Task 5	100%
3	Nguyễn Tuấn Huy	2211253	Task 1.1, 1.3.2, 2.2, 3.1, Task 5	100%
4	Nguyễn Văn Nhật Huy	2211254	Task 1.1, 1.3.2, 2.2, 3.1, Task 4, Task 5.2	100%
5	Phạm Bá Nhật Khang	2211460	Task 1.2, 1.3.1, 2.1, 3.2, Task 5	100%