Software Requirements Specification for

A smart printing service for students at HCMUT

Version 1.0 approved

Prepared by:

1. **Nguyễn Hoàng Dung** **2252109**
2. **Nguyễn Hữu Cường** **2252098**
3. **Nguyễn Hữu Thanh** **2252742**
4. **Nguyễn Lê Duy Khang** **2252303**
5. **Nguyễn Minh Tiến** **2150033**

Department of Software Engineering

Faculty of Computer Science and Engineering

Ho Chi Minh City University of Technology – VNU-HCM

<17 September 2024>

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Nguyễn Lê Duy Khang	18/09/2024	Add content for Requirement elicitation	1.1
Nguyễn Hữu Thanh	19/09/2024	Edit and add more information	1.2
Nguyễn Minh Tiến	24/09/2024	Edit and add more information	1.3
Nguyễn Minh Tiến	24/09/2024	Add diagram for whole system	1.4
Nguyễn Lê Duy Khang	24/09/2024	Add diagram for print document module	1.5
Nguyễn Hoàng Dung	25/09/2024	Add description table for module	1.6
Nguyễn Lê Duy Khang	26/09/2024	Edit diagram for print document module	1.7
Nguyễn Hoàng Dung	26/09/2024	Edit description table for module	1.8
Nguyễn Hữu Thanh	27/09/2024	Edit diagram for whole system	1.9
Nguyễn Hữu Thanh	28/09/2024	Edit diagram for whole system	1.10
Nguyễn Hữu Cường	05/10/2024	Add activity diagram	2.1
Nguyễn Minh Tiến	07/10/2024	Add sequence diagram	2.2
Nguyễn Lê Duy Khang	07/10/2024	Add sequence diagram	2.2
Nguyễn Hữu Thanh	10/10/2024	Add class diagram	2.3
Nguyễn Hoàng Dung	10/10/2024	Add class diagram	2.3
Nguyễn Hữu Cường	12/10/2024	Add MVP	2.4
Nguyễn Hữu Thanh	15/10/2024	Add additional use case for module	2.5
Nguyễn Hữu Cường	16/10/2024	Edit activity diagram and MVP	2.6
Nguyễn Minh Tiến	17/10/2024	Edit sequence diagram	2.7
Nguyễn Lê Duy Khang	17/10/2024	Edit sequence diagram	2.7
Nguyễn Hoàng Dung	18/10/2024	Edit class diagram	2.8
Nguyễn Hữu Cường	25/10/2024	Add architecture diagram	3.1
Nguyễn Hoàng Dung	26/10/2024	Add paragraph for architecture diagram	3.2
Nguyễn Hữu Thanh	27/10/2024	Edit paragraph	3.3
Nguyễn Lê Duy Khang	27/10/2024	Add component diagram	3.4
Nguyễn Minh Tiến	28/10/2024	Add paragraph for component diagram	3.5
Nguyễn Hữu Thanh	31/10/2024	Conduct a survey for MVP 1	4.0
Nguyễn Lê Duy Khang	02/11/2024	Analysis the survey	4.1
Nguyễn Minh Tiến	02/11/2024	Analysis the survey	4.1
Nguyễn Hoàng Dung	02/11/2024	Analysis the survey	4.1

1. Requirement elicitation

a. Domain Context

The HCMUT Smart Student Printing Service (HCMUT-SSPS) aims to provide a convenient and efficient document printing solution for students at Ho Chi Minh City University of Technology (HCMUT). The service will operate through a network of printers distributed across university campuses, allowing students to upload documents online and print them using various customizable settings like paper size, page range, number of copies, and print mode (single- or double-sided). This service will primarily cater to students needing to print academic materials, assignments, and other documents. With online document submission and printing management features, HCMUT-SSPS will reduce the need for manual operations and streamline the printing process across the campus.

With the default number of pages for each student, students are able to print some documents for free. If they exceed the default number of pages, they have to pay for some additional pages of printing. Besides that, the application provides reports of each month, and each year for staff to keep track.

b. Stakeholders and Needs

- *There are several key stakeholders involved in this system.*
- The primary stakeholders are students, who will use the system to print documents with ease, manage their printing quotas, and view their printing history.
- Another key stakeholder is the Student Printing Service Officer (SPSO), who oversees system configuration, printer management, and monitor usage logs to ensure smooth operation.
- The university administration is another stakeholder responsible for supporting and maintaining the infrastructure, such as providing default printing pages per semester and managing payments through the BKPay system.
- *Each of these stakeholders has unique needs*

- Students need a reliable and user-friendly printing service, efficient way to print documents without long wait times, and they require easy access to their printing history and account balance
- The SPSO requires efficient tools to manage printers and monitor printing logs,
- The university needs transparency and accountability in managing student printing activities.

c. Benefits of the System

HCMUT-SSPS provides significant benefits to all stakeholders.

- For students, the system offers convenience and flexibility, allowing them to manage their printing requirements from any campus location and monitor their printing history and balance. It also reduces the hassle of waiting in long queues for printers.
- For the SPSO, the system streamlines printer management by allowing them to enable/disable printers, configure file types, and view detailed printing logs. This enhances overall operational efficiency.
- Lastly, for the university, the system ensures better control over printing resources, reducing waste, and providing detailed reports on system usage. It also aligns with the university's goal of embracing digital solutions to improve campus services.

d. Functional Requirements

For all Stakeholders:

1. The system allows users to check if there are any available printers to use.
2. The system allows users to register any time and print in office hours.
3. The system allows users to change passwords through their registered email.

For Students:

1. The system allows students to register their account through their school emails (each student has a unique email).
2. The system must allow students to upload document files for printing.
3. The system must enable students to choose a printer from the list of available printers.

4. The system must allow students to configure printing properties, such as paper size, number of copies, and single-/double-sided printing.
5. The system must allow students to view their printing history for a selected time period, including a summary of printed pages by page size.
6. The system must enable students to purchase additional printing pages through an online payment system like BKPay and update their page balance accordingly.
7. The system allows students to choose a method of payment: cash or bank transfer (through BKPay).

For Student Printing Service Officer (SPSO):

1. The system must allow the SPSO to add, enable, or disable printers in the system.
2. The system must enable the SPSO to view detailed printing logs of all students or a specific student for a selected time period and specific printers.
3. The system must allow the SPSO to configure the permitted file types that students can upload for printing.
4. The system must allow the SPSO to restart the request of printing pages if there are errors when printing.
5. The system must generate monthly and yearly reports on system usage, which the SPSO can view and manage.
6. The system allows SPSO to create a new account for students.
7. The system allows SPSO staff to check if students already paid for their use of printers.

For University (Administration):

1. The system must automatically assign a default number of A4-size printing pages to each student at the beginning of each semester.
2. The system must ensure that the university can restrict student printing to their page balance, preventing any printing when their balance is insufficient.
3. The system must integrate with the university's BKPay system to handle online payments made by students for purchasing additional printing pages.

4. The system must provide real-time monitoring and control of the printers across campuses to ensure operational efficiency.
5. The system must ensure data security, auditing, and compliance with university policies regarding student information and printing activity.
6. The system allows admins to create, add or delete accounts of students, SPSO or another admin.
7. The system allows admins to check and refund if there is a problem during the money transfer.

e. Non- Functional Requirements

For Students:

1. The system must be accessible to students via web and mobile applications with a consistent user interface across platforms.
2. The system must ensure high availability, allowing students to upload documents and print them in office hours.
3. The system ensures that students can schedule for printing if at that moment they cannot print (as there are no more left available printers or out of office hours).
4. The system must be user-friendly and intuitive, ensuring minimal effort for students to manage printing tasks.
5. The system must process printing requests quickly to minimize delays between file upload and the printing task being executed.
6. The system must ensure data privacy, so students' personal data and printing history are securely stored and only accessible by authorized users.
7. Students can collect printing papers at school during office hours.

For Student Printing Service Officer (SPSO):

1. The system must ensure secure authentication for the SPSO, restricting access to administrative functions such as disabled printers (when they cannot work) and configuration advanced settings (such as the size paper, default number of pages).

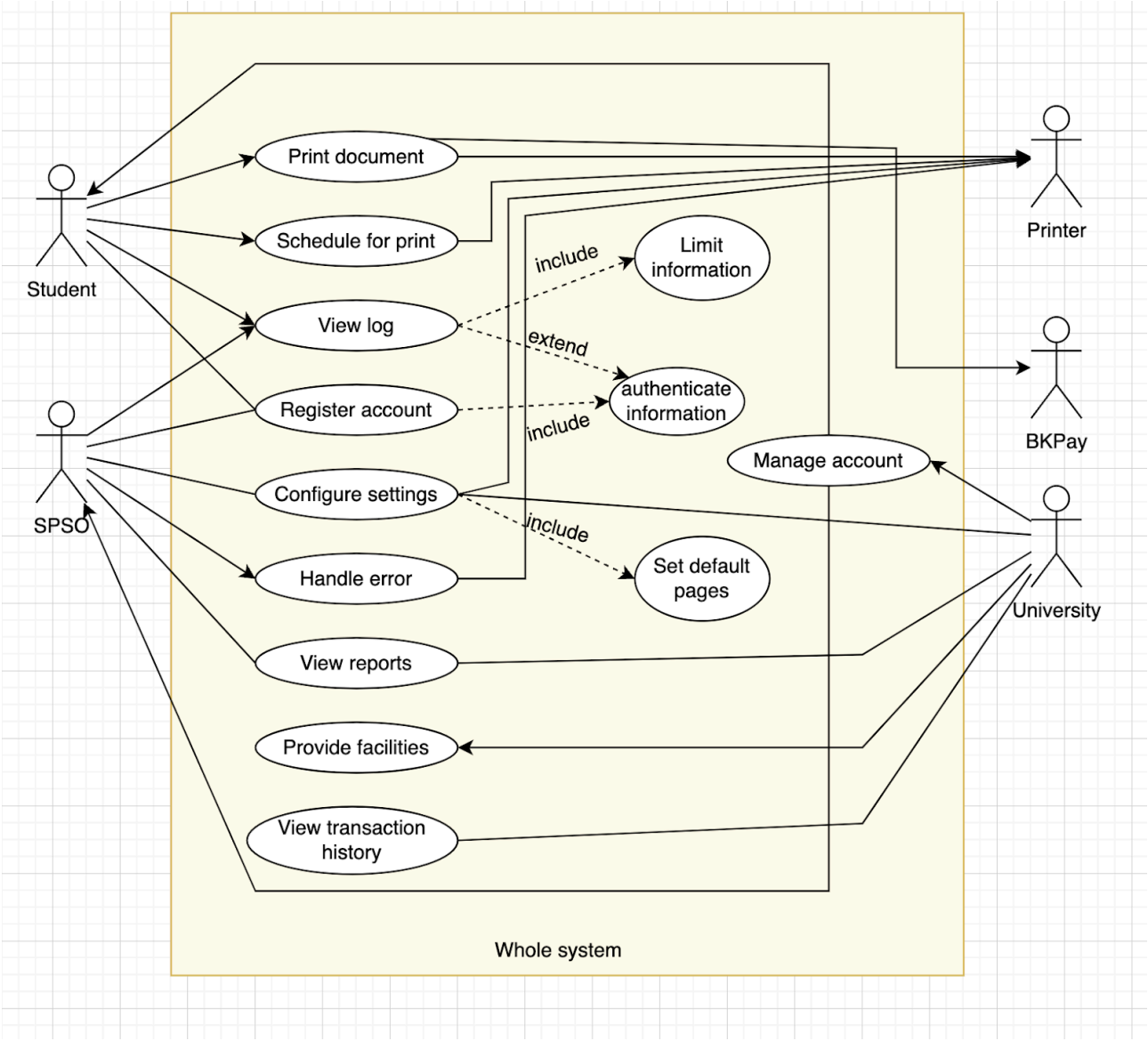
2. The system must provide real-time updates and notifications to the SPSO regarding printer status (e.g., offline, low ink, out of paper) and other critical system events (printer cannot print as struck paper).
3. The system must support scalability, allowing the SPSO to manage an increasing number of printers and growing student population without performance degradation.
4. The system must generate and store monthly and yearly reports automatically, ensuring the SPSO can access them anytime without manual intervention.
5. The system must have a robust logging mechanism, tracking all administrative changes made by the SPSO for accountability and audit purposes.

For University (Administration):

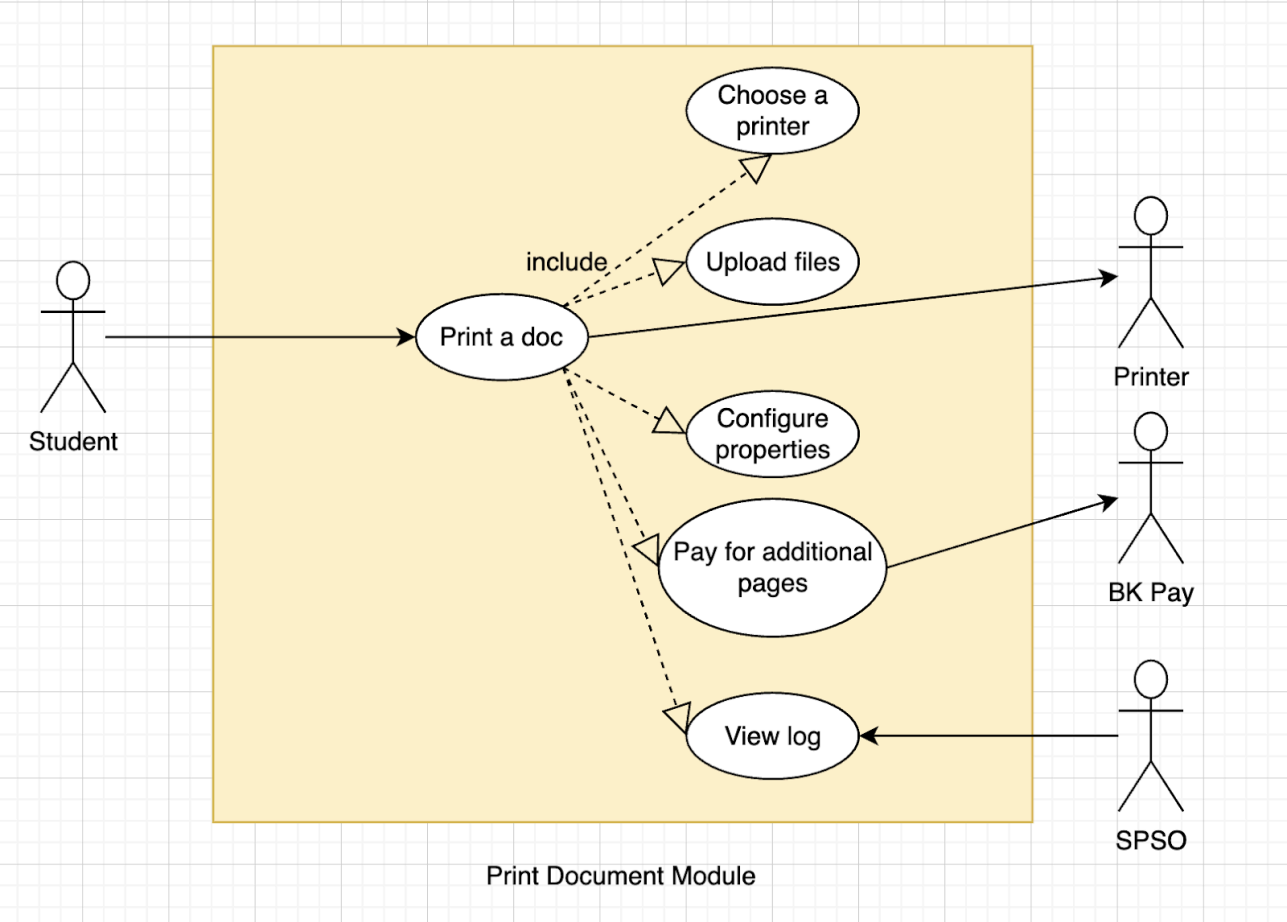
1. The system must integrate securely and efficiently with the university's HCMUT_SSO authentication service to ensure compliance with university-wide access control policies.
2. The system must be reliable, handling large volumes of printing requests and payments during peak academic periods without compromising performance.
3. The system must ensure that all financial transactions (e.g., payments for additional pages) through BKPay are processed securely and in real-time.
4. The system must follow strict data security policies, protecting sensitive information like student IDs, payment details, and printing logs from unauthorized access.
5. The system must provide audit trails and reports that are compliant with university regulations and policies, ensuring transparency and traceability of all activities within the system.

2. Use-case Diagrams

a. Use-case Diagram for the Whole System



b. Use-case Diagram for Print Document Module



c. The Details of Use-Cases in Print Document Module

1. Use Case *Choose a printer*

Use-case name	**Choose a printer**
Created By	Group 7 **Date Created:** **27/09/2024**
Primary Actor	Student
Secondary Actor	Printer
Description	A student chooses an available printer for printing.

Trigger	Student wants to print a document.
Preconditions	PRE-1. The student has a valid HCMUT account. PRE-2. The printer is available to print.
Postconditions	POST-1. The student chooses the printer successfully. POST-2. The other users cannot access that printer if already chosen by one user. POST-3. Each printer is chosen by only one user at once time.
Normal Flow	1.0 Choose a printer directly through application. 1. Log in with the authenticated account. 2. System gives the list of printers that are available at the time. 3. Student chooses one printer in the list to print.
Exceptions	**E.1.** There are no available printers at the moment. **E.2.** There are two students who suddenly chose 1 printer at the moment, the system will notify the error then allow students to choose again.
Alternative Flow	**None**

2. Use case *Upload a file*

Use-case name	**Upload a file**
Created By	Group 7 **Date Created: 27/09/2024**
Primary Actor	Student
Secondary Actor	Printer
Description	A student chooses a printer which is available for printing.
Trigger	Student wants to print a document.

Preconditions	PRE-1. The student chooses the printer successfully. PRE-2. The file student upload must have the correct format that system allows.
Postconditions	POST-1. The student uploads files successfully. POST-2. No one except admin and user could access the printer.
Normal Flow	2.0 Upload the files 1. Choose the files from the user's local disk to upload. 2. Make sure that the files are in the correct format that system allows, if not, change their format. 3. Upload the files to the system 4. The system will notify if the process is successful.
Exception	None
Alternative Flow	None

3. Use Case *Configure properties*

Use-case name	**Configure properties**
Created By	Group 7 **Date Created: 27/09/2024**
Primary Actor	Student
Secondary Actor	Printer
Description	A student configures properties of the printer for printing.
Trigger	Student wants to print a document.
Preconditions	PRE-1. The student has chosen the printer and uploaded the files successfully.

	PRE-2. The printer allows the user to configure the settings. PRE-3. The student configures the printer's settings which do not violate the constraint of the printer. PRE-4. The user doesn't make any change in settings that lead to conflict in the printer.
Postconditions	POST-1. The printer is ready to print POST-2. The system asks the student for the confirmation.
Normal Flow	3.0 Configure properties 1. Configure the properties if there are any default properties that do not match with the user's purpose. 2. The system will notify the confirmation box of the properties for the user. 3. The system allows users to print.
Exceptions	**E.1.** Student configures the settings which are not allowed to change due to the role (which settings are just modified by granted user) **E.2.** Students configure the settings that are not suitable for the printer (the paper size is not available for the printer).
Alternative Flow	None

4. Use Case *Pay additional pages*

Use-case name	**Pay additional pages**
Created By	Group 7 **Date Created: 27/09/2024**
Primary Actor	Student
Secondary Actor	Printer, BKPay
Description	A student pays for additional pages for printing.

Trigger	Student wants to print a document with lots of pages.
Preconditions	PRE-1. The student has money in an online banking account. PRE-2. Student need to log in to BKPay. PRE-3. BKPay is ready for the transaction process.
Postconditions	POST-1. The system returns the bill of transaction. POST-2. The system increases the allowed pages for printing.
Normal Flow	4.0 Pay additional pages 1. Log in BKPay system 2. The system will generate the QR Code for scanning to get access to BKPay. 3. Using the user's bank account to pay. 4. The system notified that user has paid success for additional pages.
Exceptions	**E.1.** If a student cannot get to the QR Code or cannot pay, the system will allow the QR Code for 10 minutes, if it is over 10 minutes, the transaction process will be terminated.
Alternative Flow	None

5. Use Case *View log*

Use-case name	**View log**
Created By	Group 7 **Date Created: 03/10/2024**
Primary Actor	Student, SPSO
Secondary Actor	Printer,System
Description	A student wants to view him/her history of printing. The staff of SPSO wants to see the history of printing for all students.

Trigger	Students who want to view their history of printing. SPSO who wants to check the history of printing of all students at a specific time.
Preconditions	PRE-1. The student must have printed before or their history will be empty PRE-2: The student has an account PRE-2. SPSO who want to view must have the grant from the University. PRE-3. The time that the user wants to see must be valid.
Postconditions	POST-1. The system returns the history of printing for himself/herself if the user is a student, otherwise for all of the students at a specific time. POST-2. The system returns the empty if the data has been deleted by admin or there is no data at that time.
Normal Flow	5.0 View log 1. When you go to the printing section, you'll see a button to check your past print jobs. Clicking this button will show you a list of everything you've printed, with details like what you printed and when. 2. You can decide if you want to see all your past print jobs, or just the ones from a specific time. 3. The system will make sure the date and time you entered are correct (if you chose to view a specific time) 4. If you choose a time and there's nothing printed at that time, the system will show you that there are no results.
Exceptions	**E.1.** If users choose a specific time and that time is not valid (for ex: exceed the real time, too far in the past), the system will notify that the time was not valid and ask users to re-enter again.
Alternative Flow	None

3. System modeling

3.1. The activity diagrams

3.1.1. Use Case Choose a printer

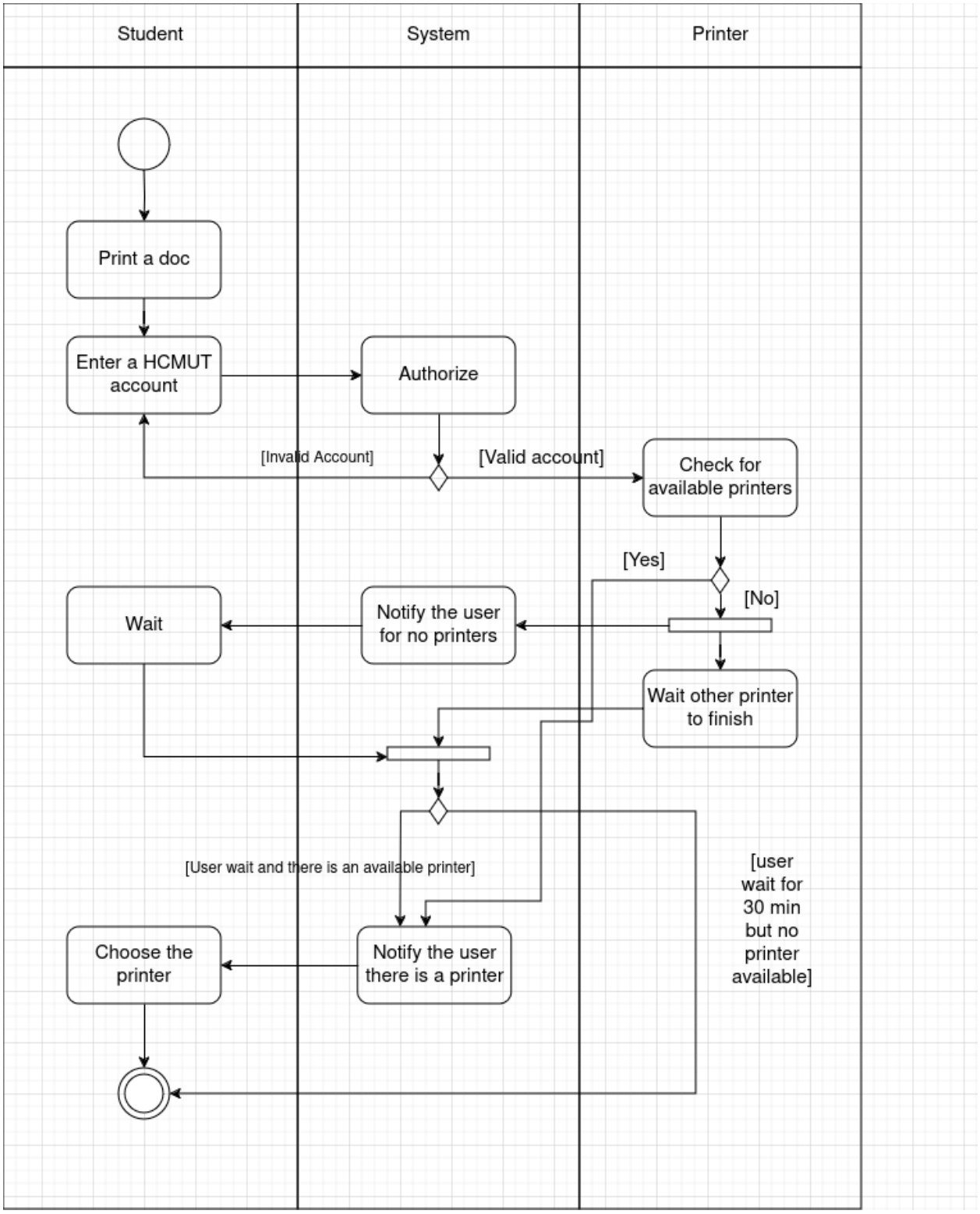


Figure 3.1. Activity diagram for choosing a printer

Students need to log into their HCMUT accounts to print documents. The system verifies the account, then checks for available printers. If one is found, students can select it. If not, they wait up to 30 minutes. If no printer becomes available within that time, the printing session ends.

3.1.2. Use case Upload a file

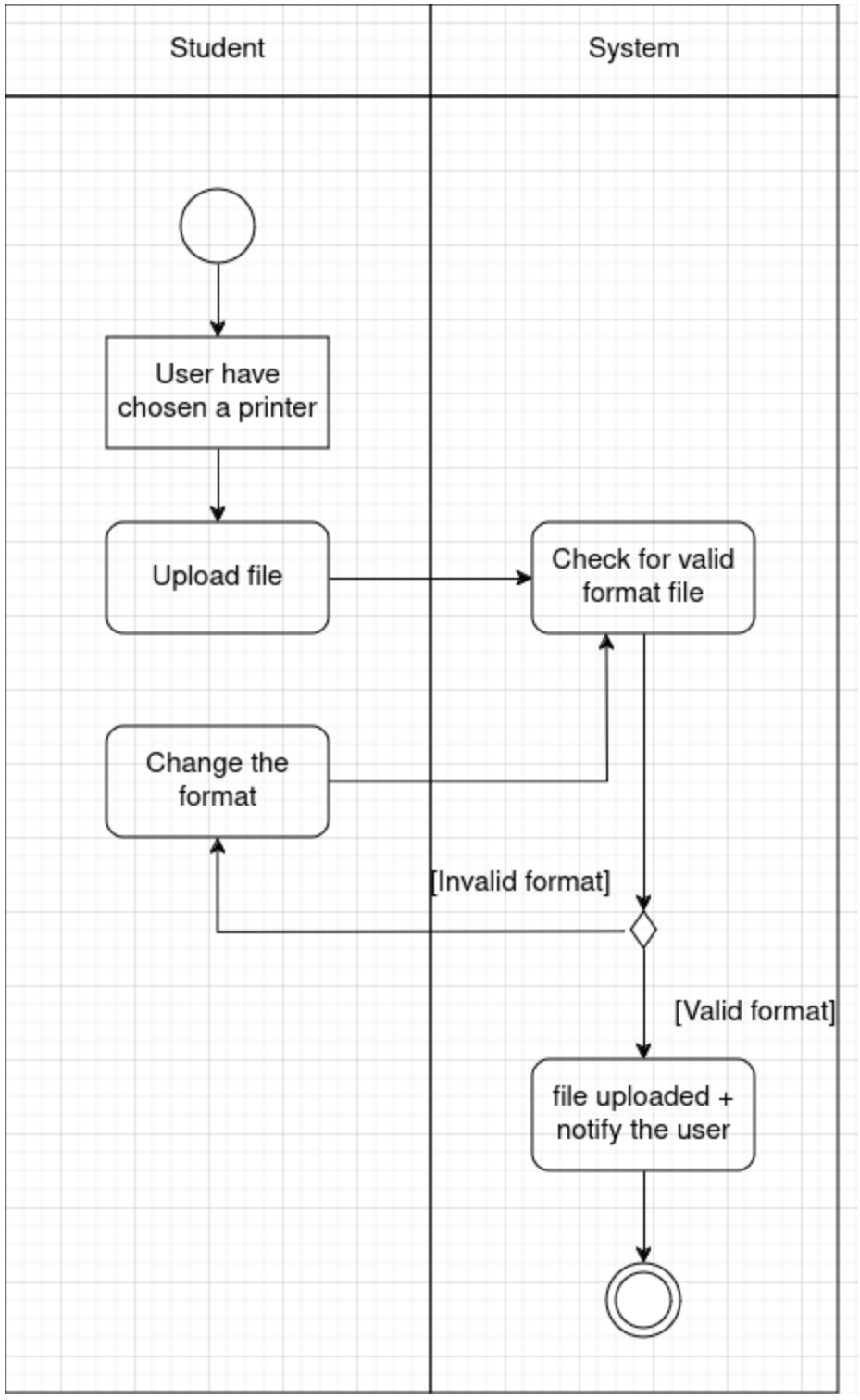


Figure 3.2. Activity diagram for uploading a file

After the user has chosen the printer, he can upload the file to the system. The system will check if that the format of type of the file is valid, if not, the student has to change the format of type. If the file is uploaded successfully, the system will notify the student.

3.1.3. Use Case Configure properties

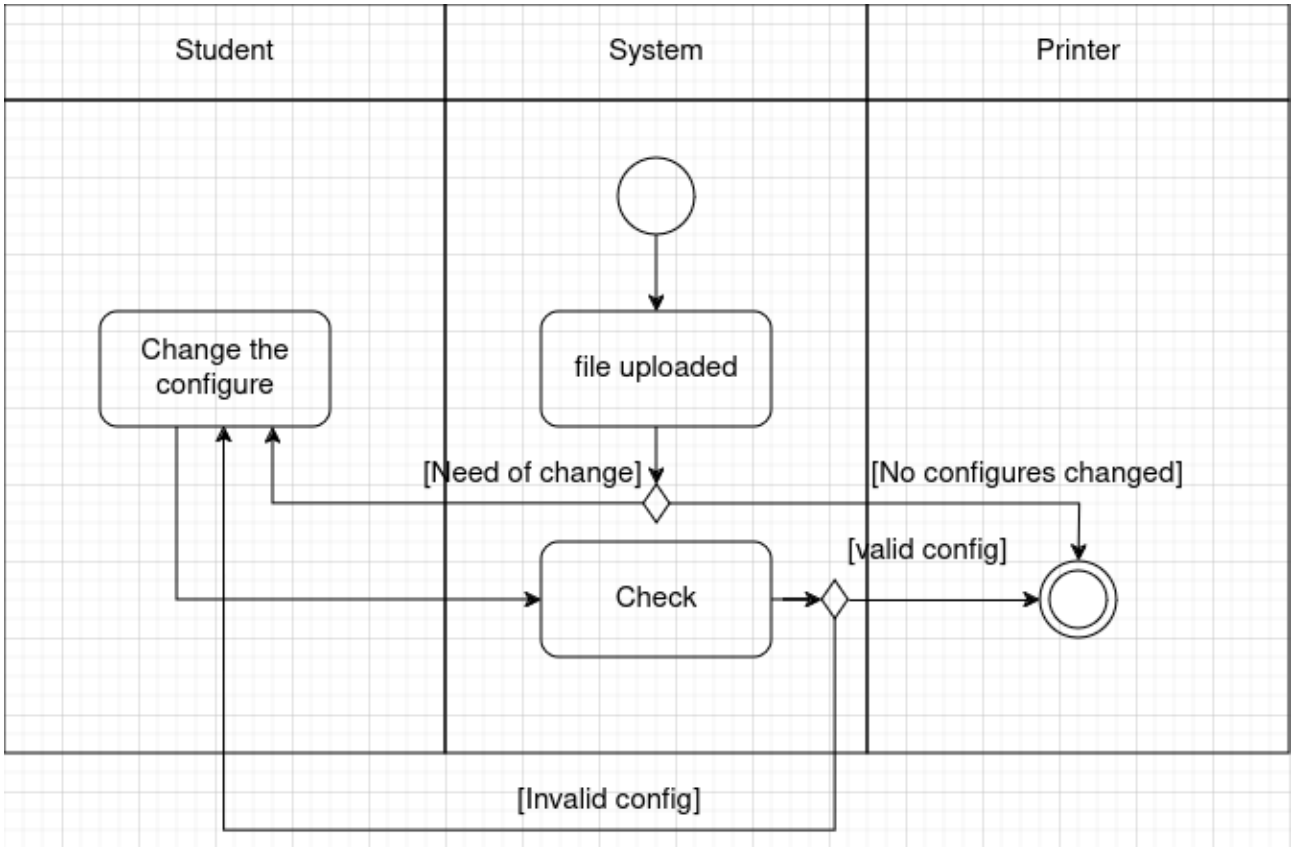


Figure 3.3. Activity diagram for Configuring properties

After the file is uploaded, if the student can choose to change the configuration. If the configured settings are not suitable or conflict, the system informs the user of the error and lets the user configure again.

3.1.4 Use Case Pay additional pages

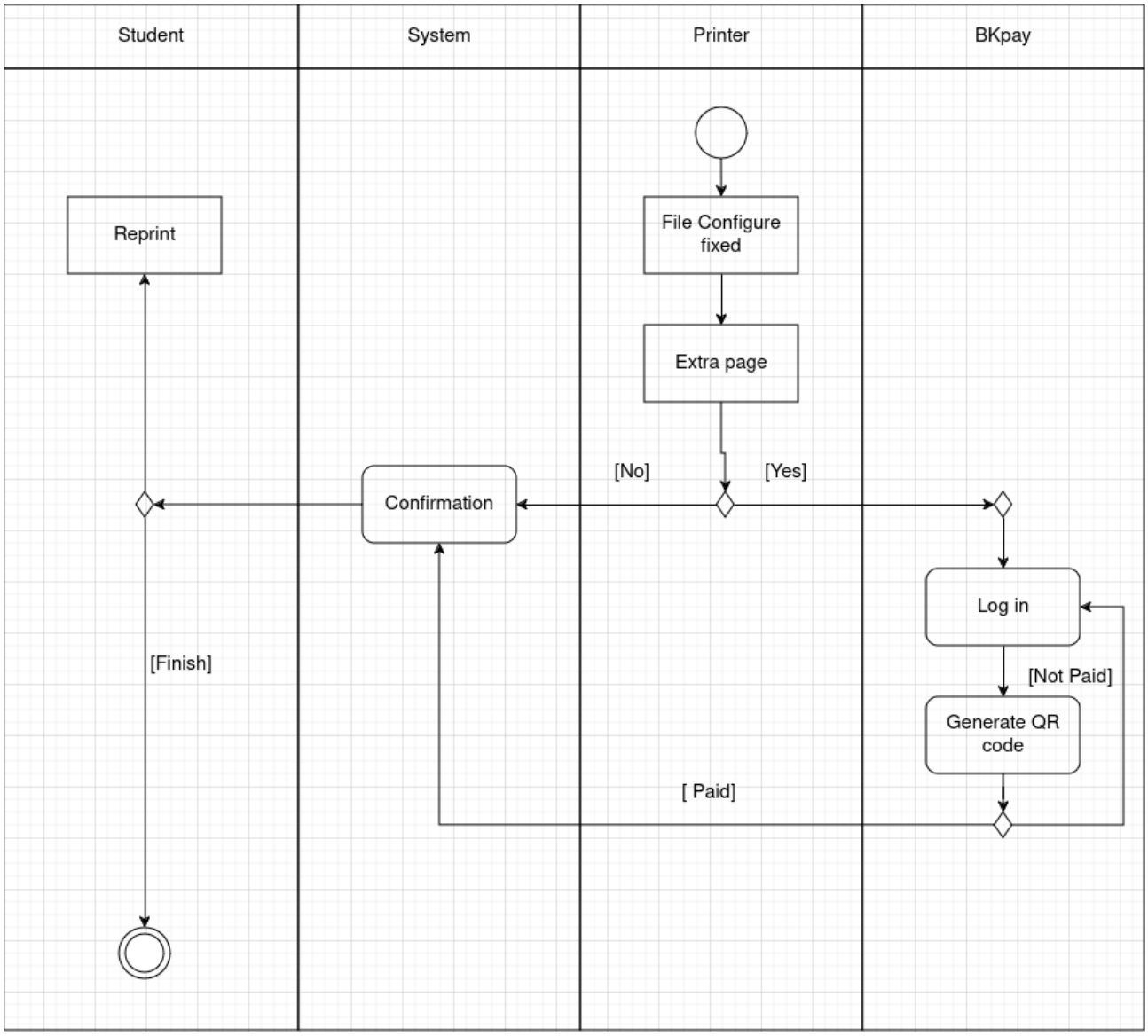


Figure 3.4. Activity diagram for paying addition pages and printing

After all the configuration is fixed, the printer will check if there are any extra pages. If there are any, the student will log in to his BKpay account. Then BKpay generates a QR Code for the student to pay for those extra pages, then the system will notify a confirmation if the student has paid. After that the student can choose to reprint or end the session

3.1.5. View log

Figure 3.5. Activity diagram for view log of student

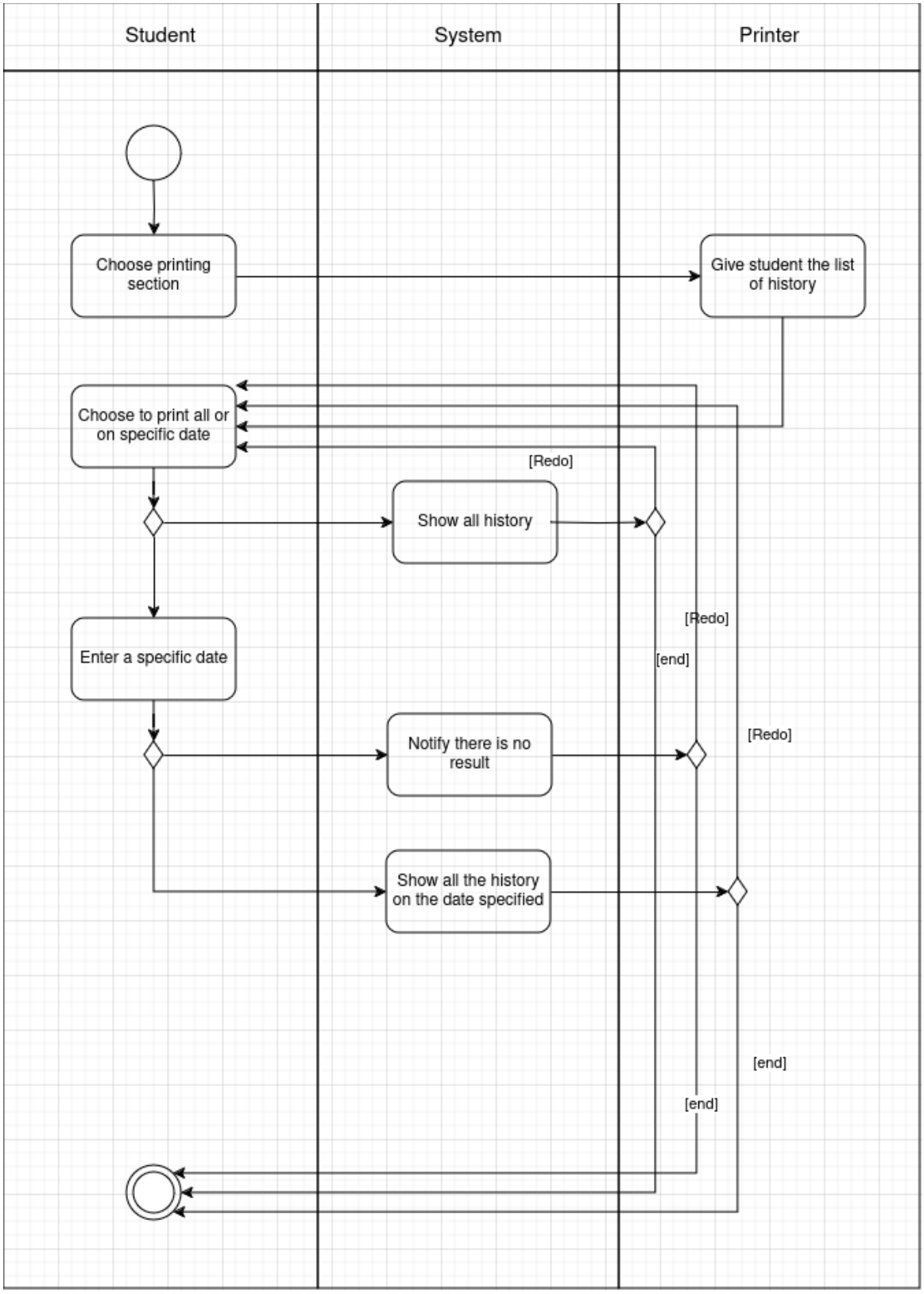
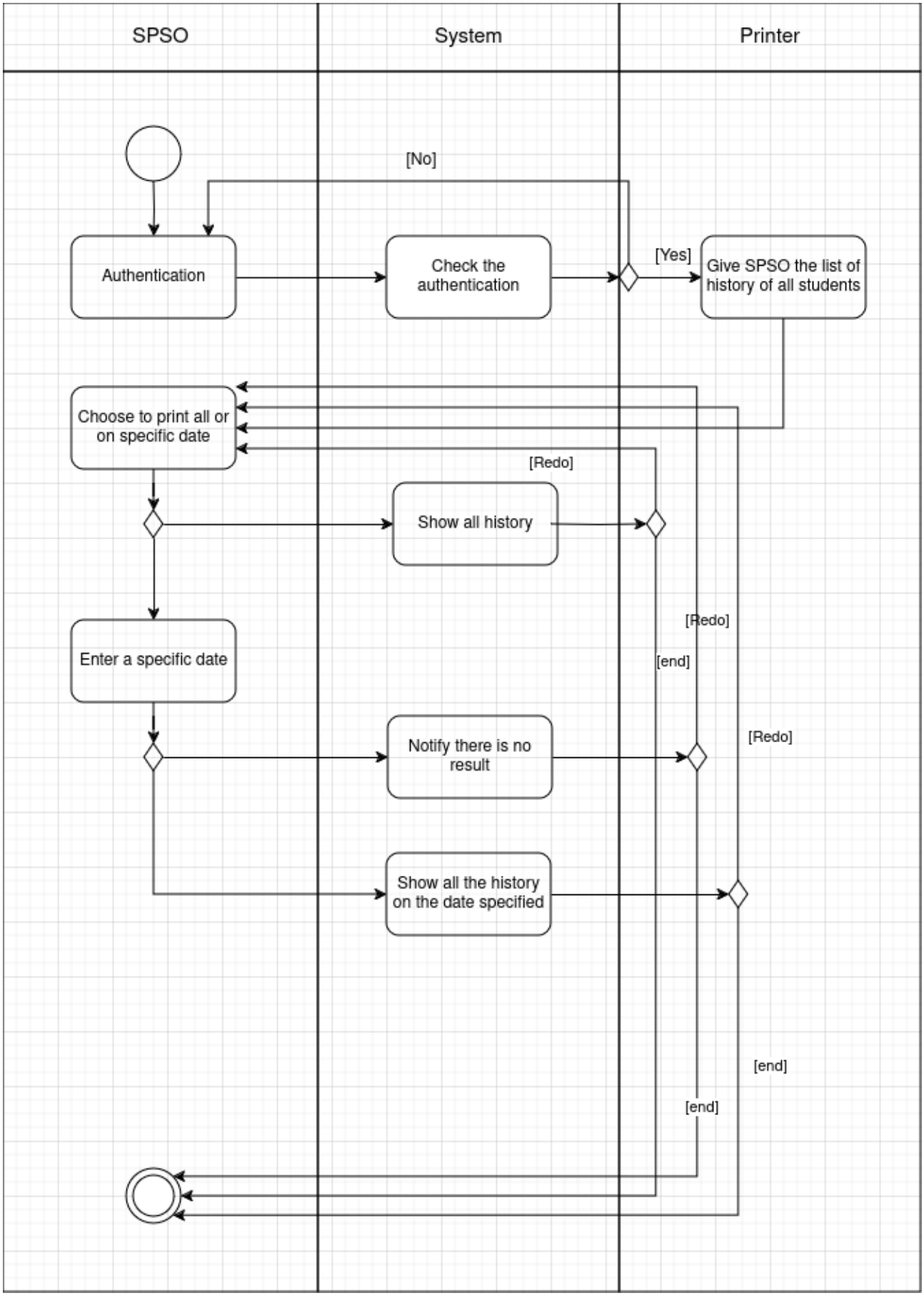

Student
System
Printer
Choose printing section
Give student the list of history
Choose to print all or on specific date
[Redo]
Show all history
[Redo]
Enter a specific date
[end]
Notify there is no result
[Redo]
Show all the history on the date specified
[end]
[end]

Figure 3.6. Activity diagram for view log of SPSO


SPSO
System
Printer
[No]
Authentication
Check the authentication
[Yes]
Give SPSO the list of history of all students
Choose to print all or on specific date
[Redo]
Show all history
[Redo]
Enter a specific date
[end]
Notify there is no result
[Redo]
Show all the history on the date specified
[end]
[end]

After authentication, the student/SPSO can check for the history. For the student, he can only check the history of his own, while the SPSO can check for the history of all students. User can choose to show all the history, or enter a specific date to check.

3.1.6. Overall activity diagram of print document module

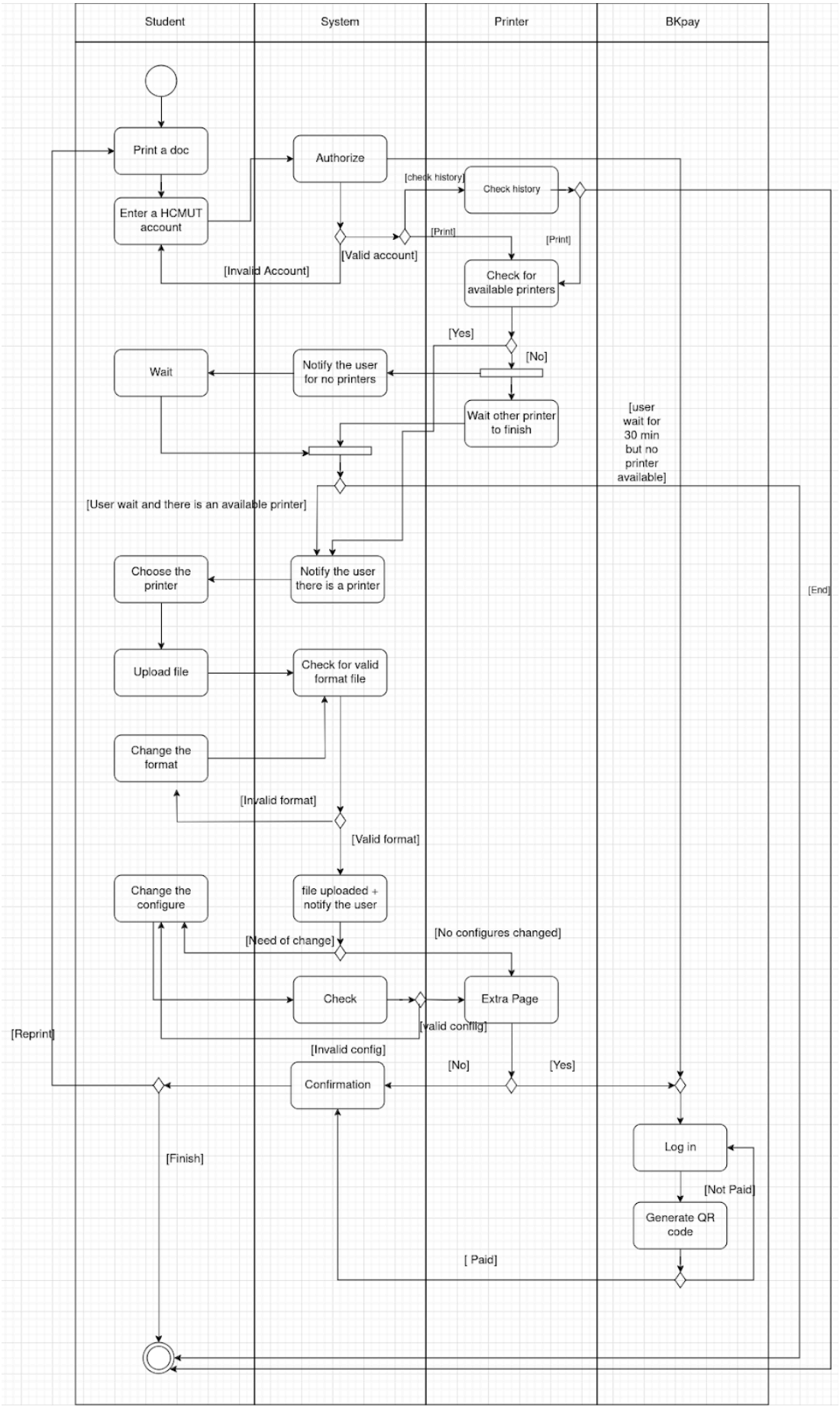

Student
System
Printer
BKpay
Print a doc
Authorize
Enter a HCMUT account
[check history]
Check history
[Print]
[Print]
[Valid account]
[Invalid Account]
Check for available printers
[Yes]
[No]
Wait
Notify the user for no printers
Wait other printer to finish
[user wait for 30 min but no printer available]
[User wait and there is an available printer]
Choose the printer
Notify the user there is a printer
[End]
Upload file
Check for valid format file
Change the format
[Invalid format]
[Valid format]
Change the configure
file uploaded + notify the user
[Need of change]
[No configures changed]
Check
Extra Page
[valid config]
[Reprint]
[Invalid config]
[No]
[Yes]
Confirmation
Log in
[Finish]
[Not Paid]
Generate QR code
[Paid]

Figure 3.7. Activity diagram for printing document

3.2. The sequence diagram

3.2.1. Use Case Choose a printer

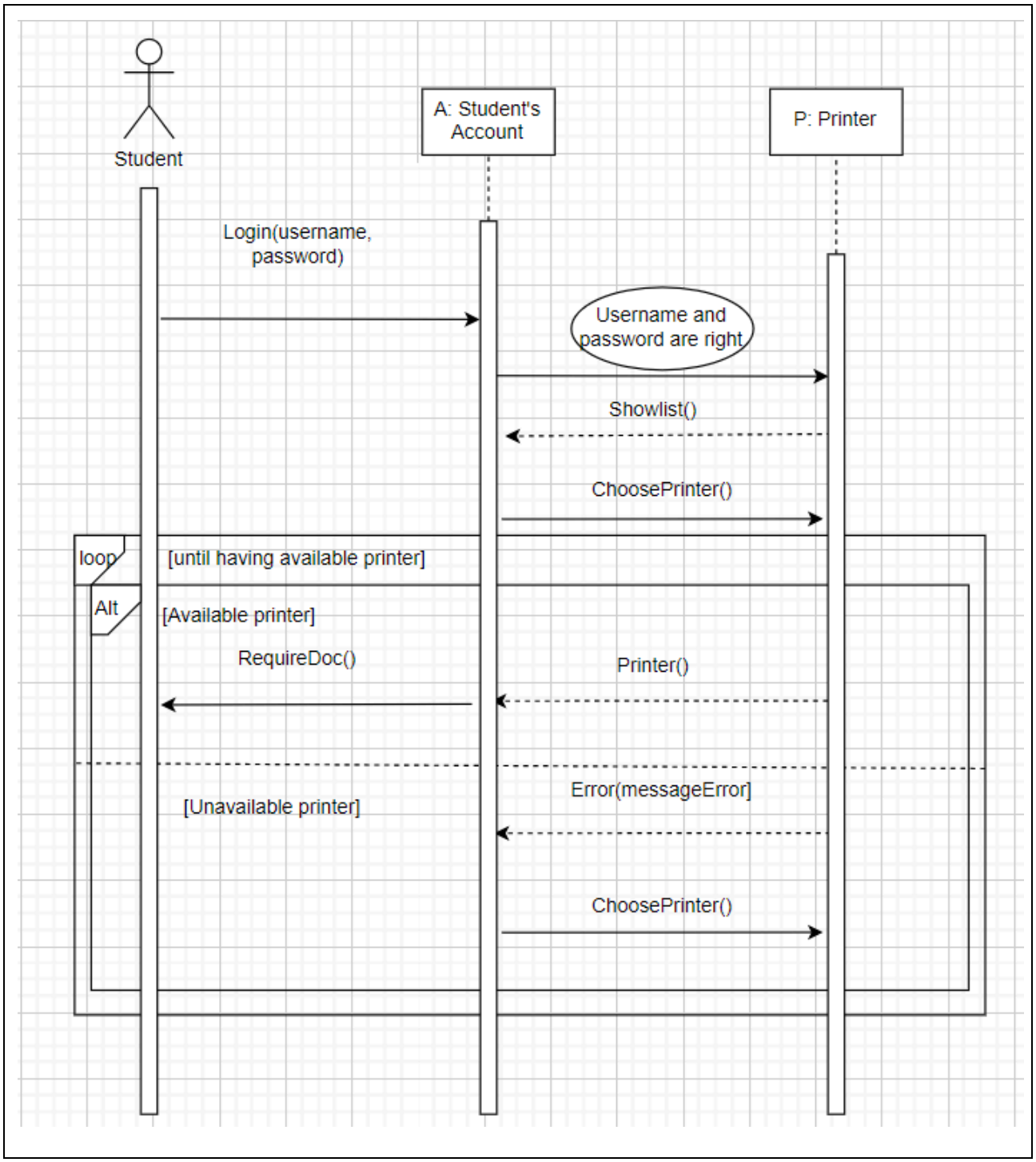


Figure 3.8. Sequence diagram for choosing a file

According to Use Case Choose a Printer, we have an actor: Student, and two instances: A: Student's Account and P: Printer. Firstly, the student must login the account by entering username and password, then, if those are right, the student's account will show the list of printers sent by the

software and the student will choose one of them. Secondly, if the one student chooses is available, the system will send back the successful signal for him/her to continue the step of inputting the document, else it will display the message error and require to choose again until having an available printer.

3.2.2. Use case *Upload a file*

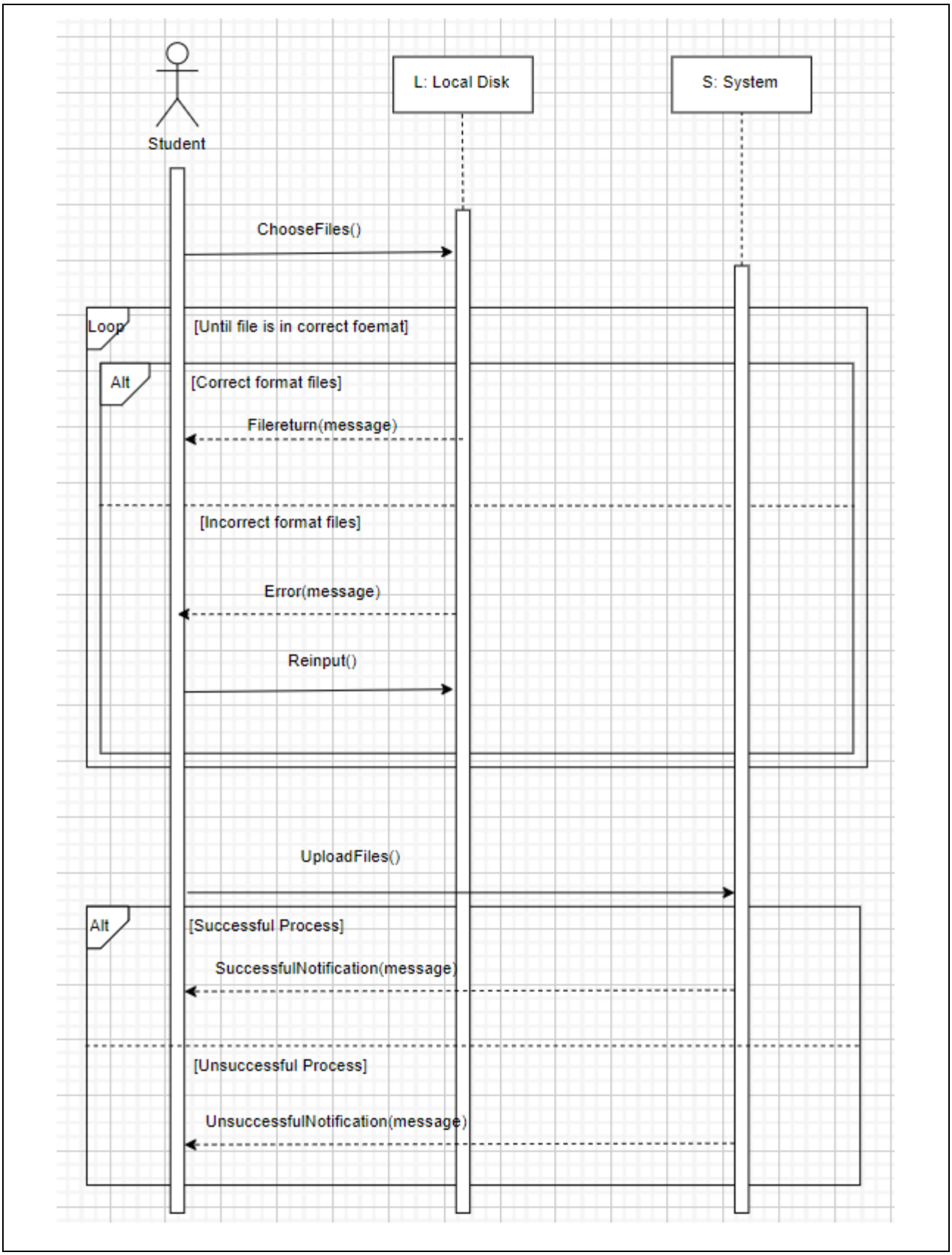

Student
L: Local Disk
S: System
ChooseFiles()
Loop
[Until file is in correct foemat]
Alt
[Correct format files]
Filereturn(message)
[Incorrect format files]
Error(message)
Reinput()
UploadFiles()
Alt
[Successful Process]
SuccessfulNotification(message)
[Unsuccessful Process]
UnsuccessfulNotification(message)

Figure 3.9. Sequence diagram for uploading a file

According to Use Case Upload a File, we have actor: Student, and two instances: L: Local Disk and S: System. Firstly, students should choose a file from a local disk, then if the format files are correct, the system will send the successful signal and the student can upload the file into the system, else the system will send the error message and require the student to re-input until the format is correct. Secondly, if the process of uploading the file is successful, the system will send the successful notification, if not it will send the failed one.

3.2.3 Use Case Configure properties

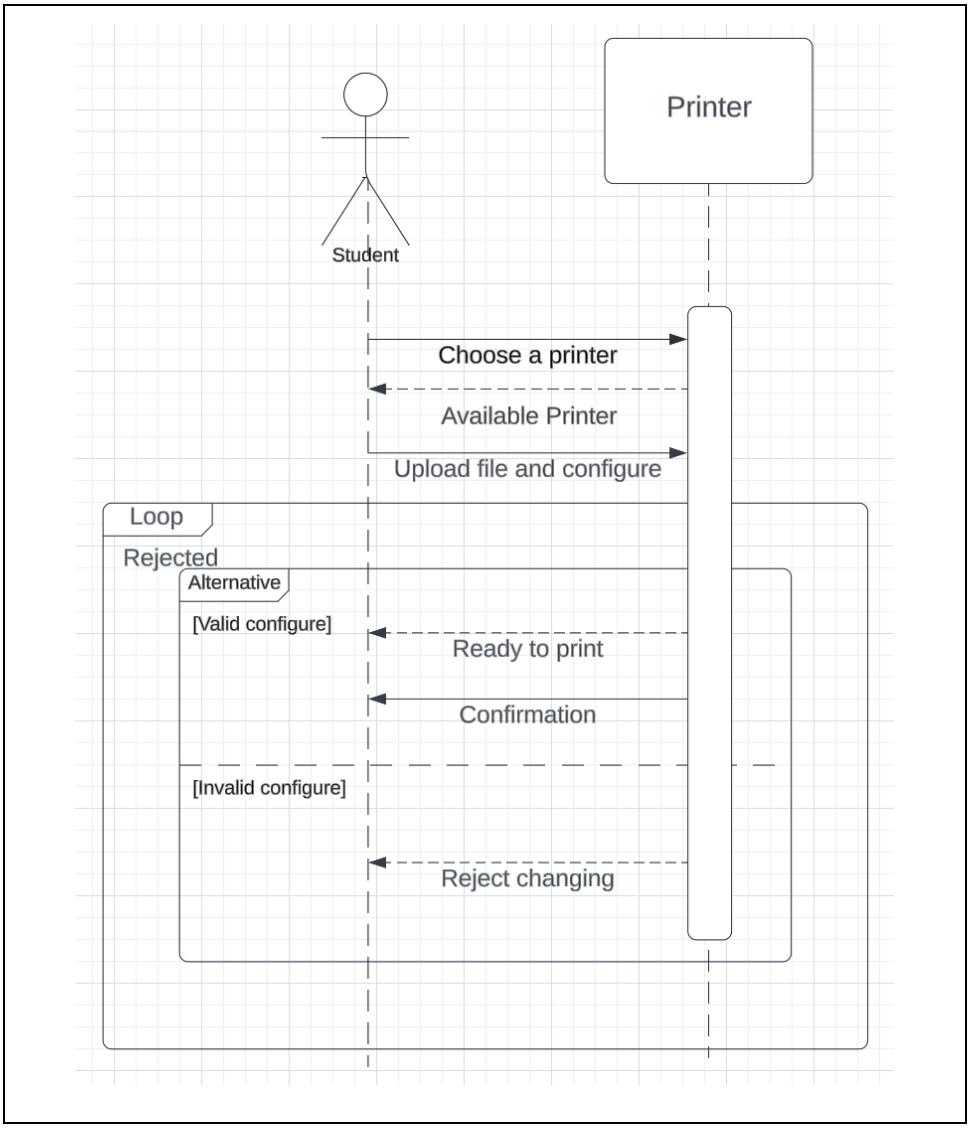


Figure 3.10. Sequence diagram for configuring properties

According to Use Case Configure properties, we have an actor: Student, and an instance: Printer. Firstly, the student must choose an available printer and upload their files. If the student configures the printer's settings which do not violate the constraint of the printer or the user makes any change in settings that lead to conflict in the printer, the student's configuration will be rejected, else their files are ready to print.

3.2.4 Usecase Pay additional pages

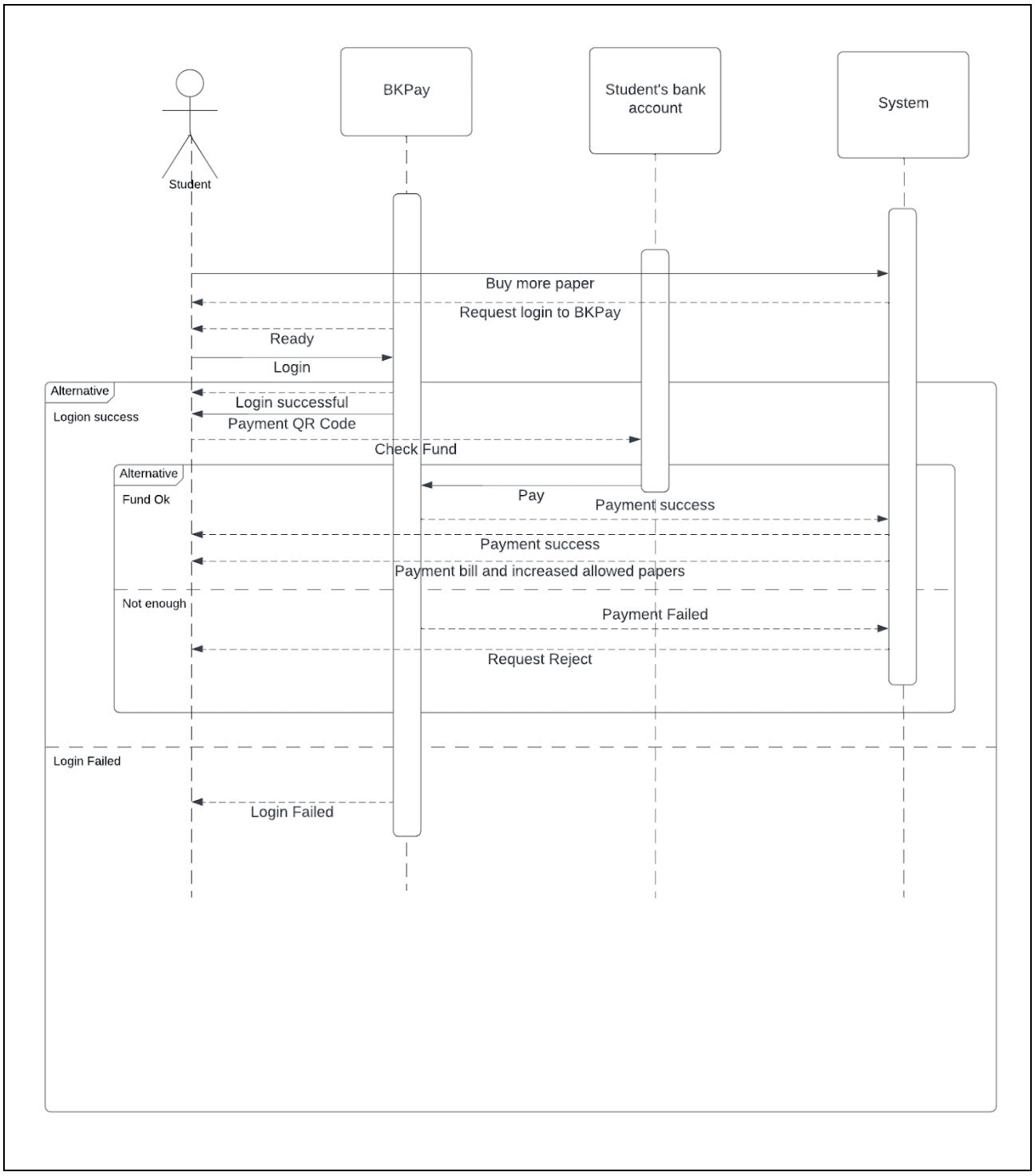


Figure 3.11. Sequence diagram for paying additional pages

In this use case scenario, a student wants to print a document with many pages and needs to pay for additional pages. First, the student logs into the BKPay system, where the system generates a QR code for the student to scan and complete the transaction. After scanning the QR code, the

student uses their bank account to make the payment. Upon successful payment, the system notifies the student that they have successfully paid for the additional pages. Preconditions for this process include the student having money in their online banking account and BKPay being ready for the transaction. If the student is unable to access the QR code or complete the payment, the system allows the QR code to remain valid for 10 minutes; if this time elapses, the transaction process will be terminated. After completion, the system returns the transaction bill and increases the allowed pages for printing.

3.2.5 Use Case View Log

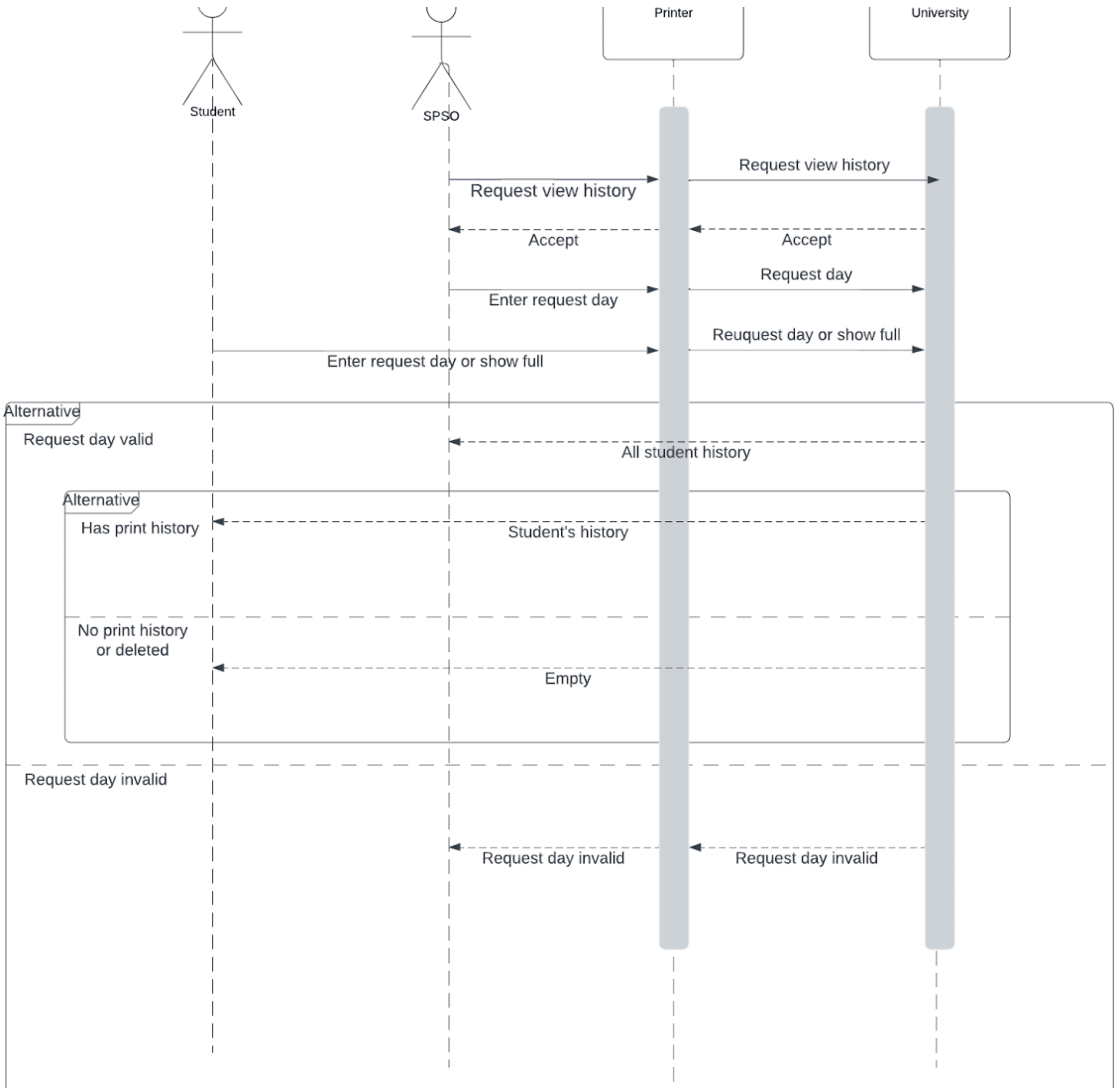


Figure 3.12. Sequence diagram for paying additional pages

According to Use Case View Log, we have two main actors: Student and SPSO (System Printing Service Officer). Firstly, the student or SPSO accesses the printing section in the system and clicks the button to view the print history. The system checks if the student has printed before or, for SPSO, whether they have the necessary permissions. If the time period selected is valid, the system retrieves the relevant print logs. If the format or time is incorrect, the system sends an error message and prompts the user to re-enter valid details. Secondly, if the log retrieval is successful, the system displays the printing history. If not, it sends a notification indicating that there are no results or the selected time is invalid.

3.3. Class Diagram

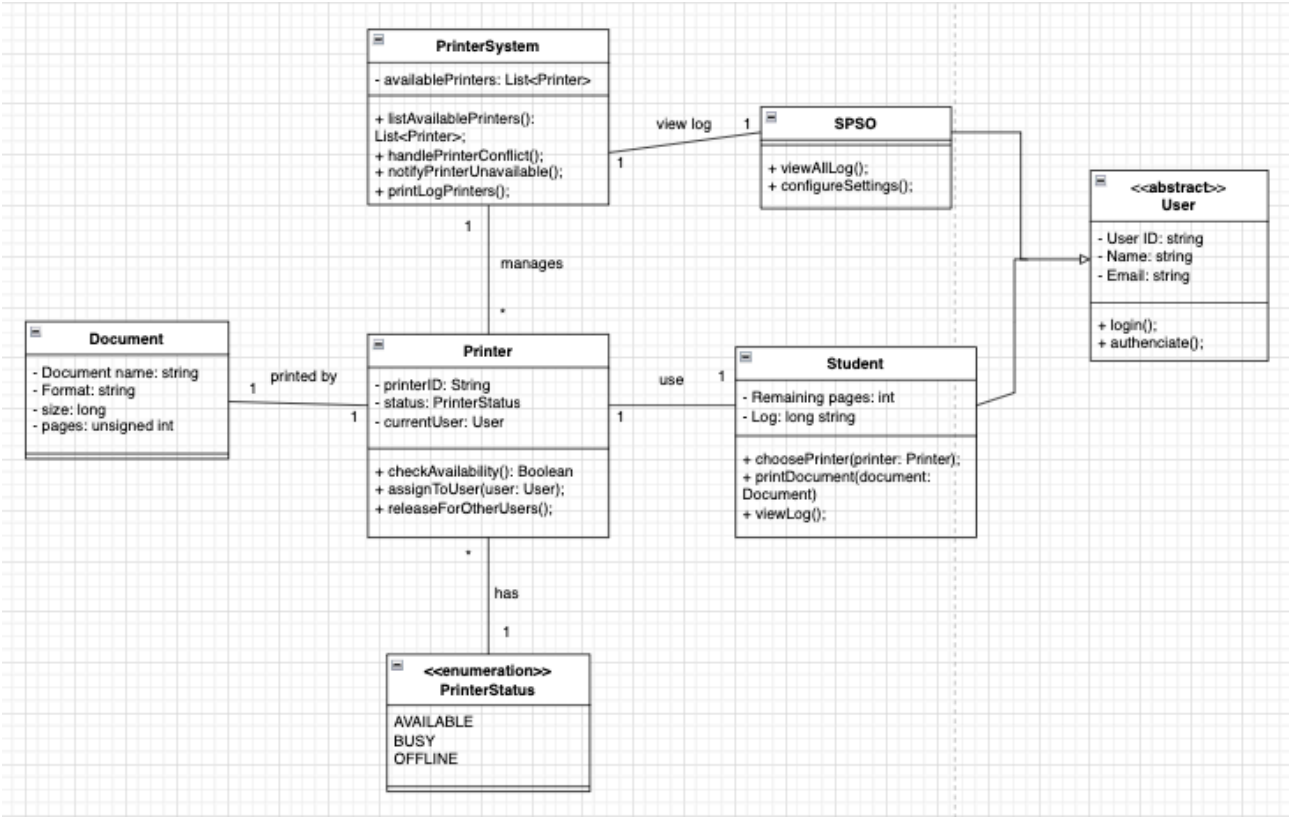


Figure 3.13. Class Diagram for Printing Document

This diagram encompasses primary classes such as Student, SPSO, Document, Printer, and Payment, representing the key entities involved in the printing process. Each class encapsulates attributes and methods specific to its function. For instance, the Student class possesses attributes like student ID, student name, remaining print pages, and methods enabling students to log in,

upload documents, select printers, configure print settings, and make payments. The relationships between these classes are clearly depicted in the diagram; for example, the one-to-many relationship between Student and Document signifies that a student can upload multiple documents for printing

3.4. MVP

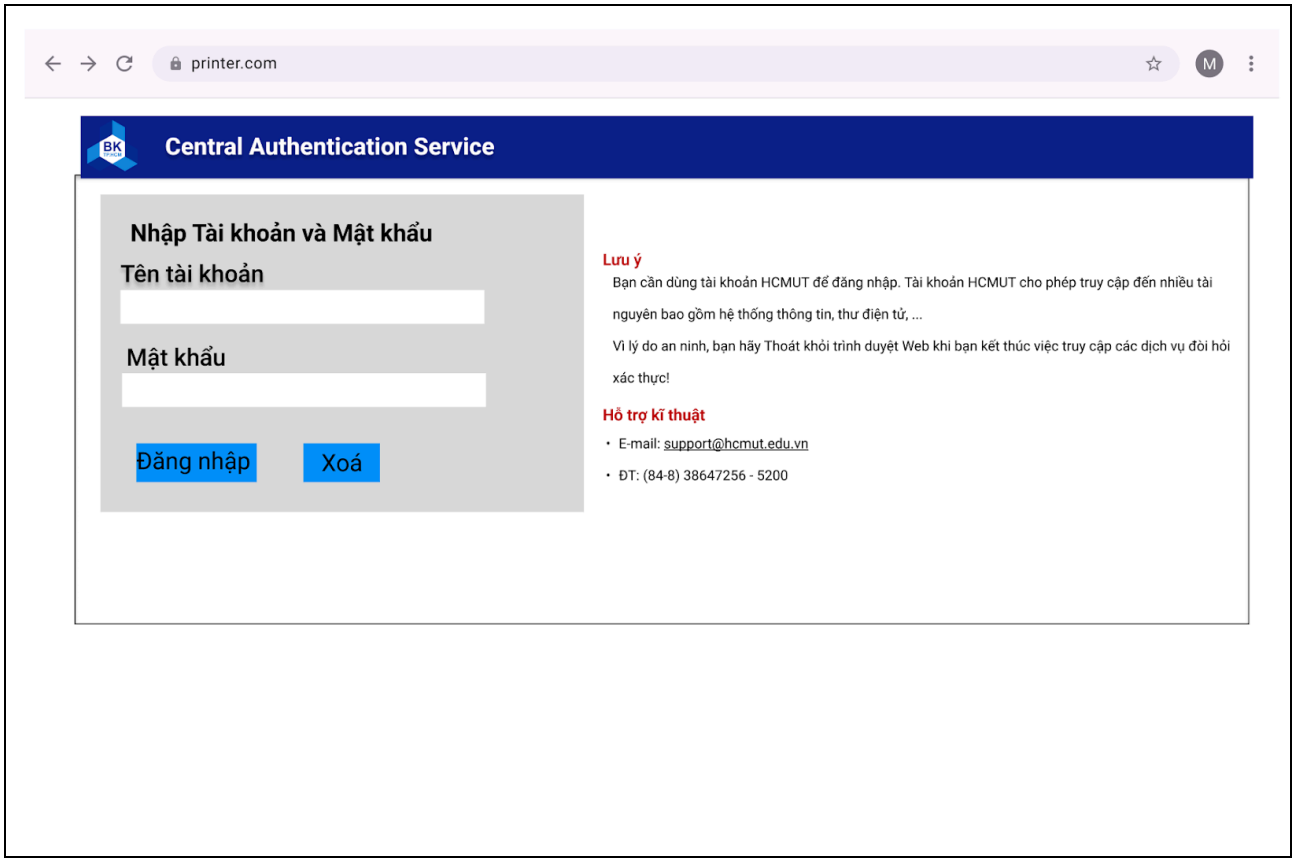


Figure 3.14. Illustrate for login page

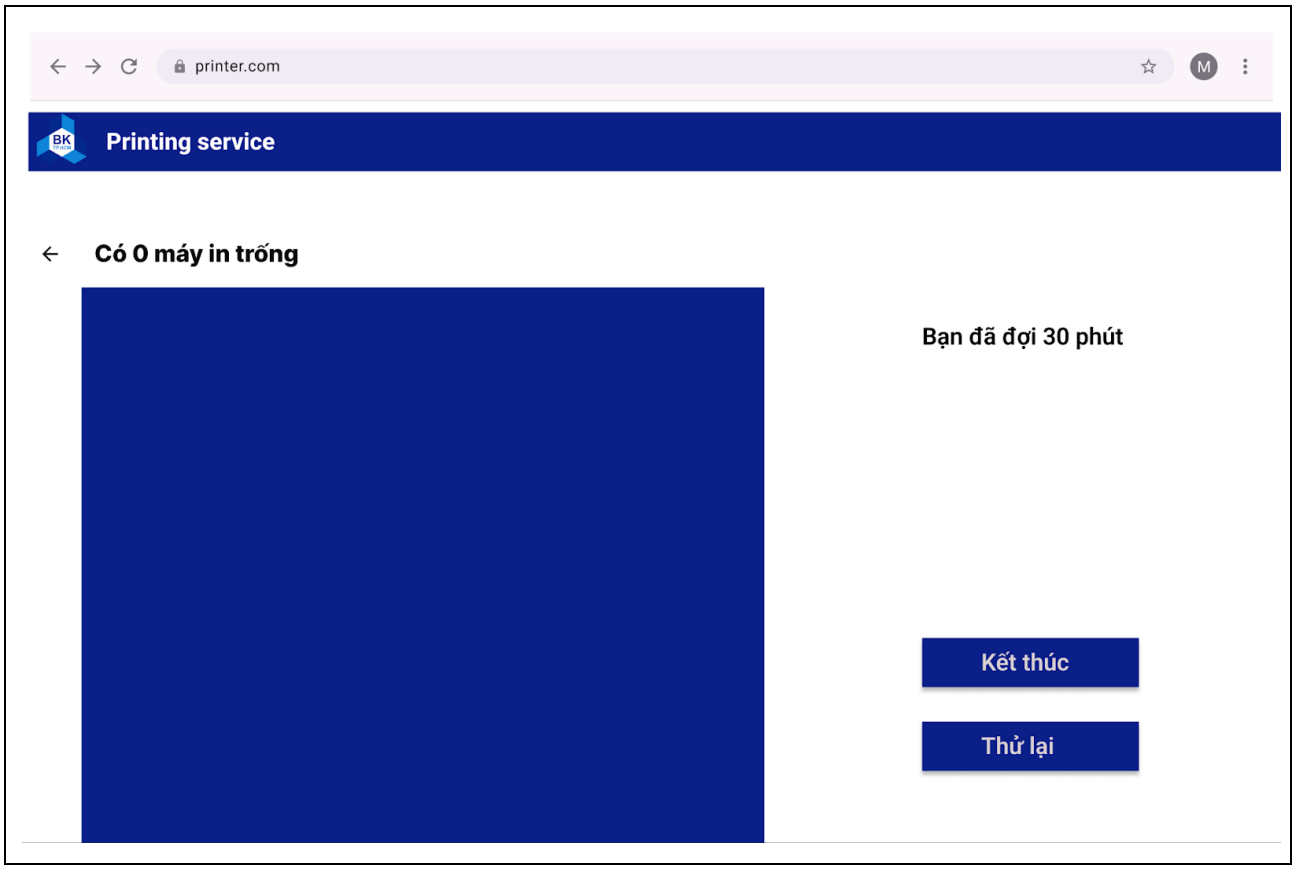


Figure 3.14. Illustrate for waiting for printer page

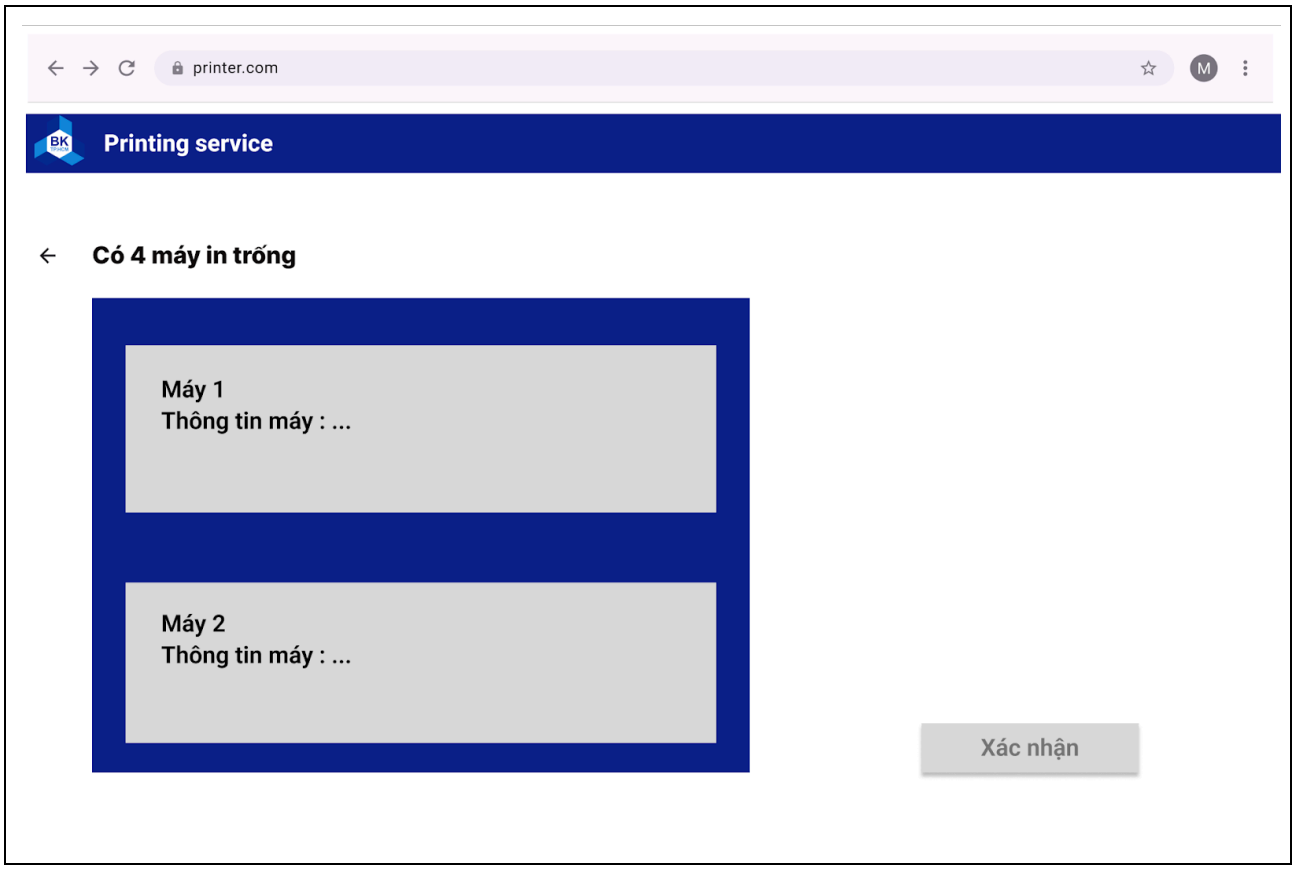


Figure 3.14. Illustrate for list of available printer page

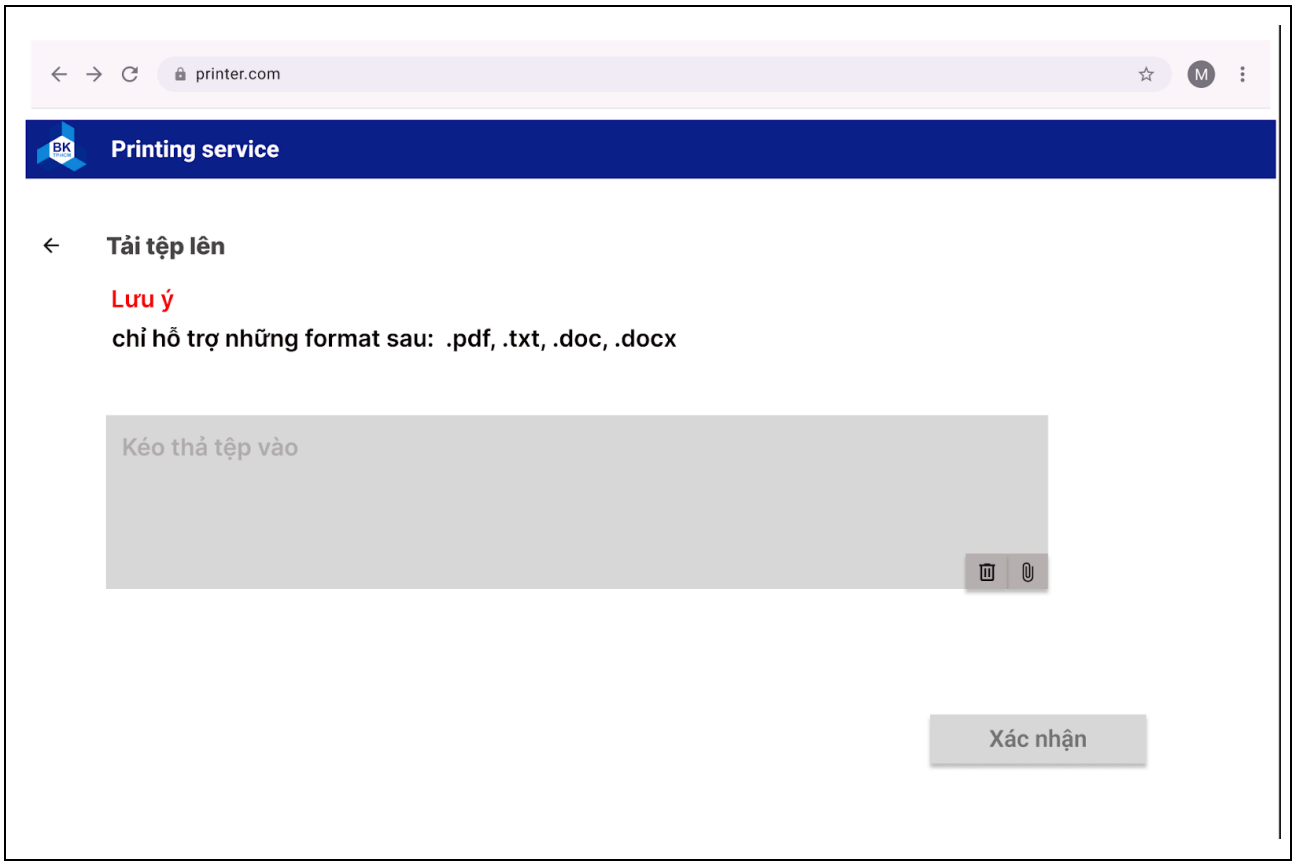


Figure 3.14. Illustrate for upload page

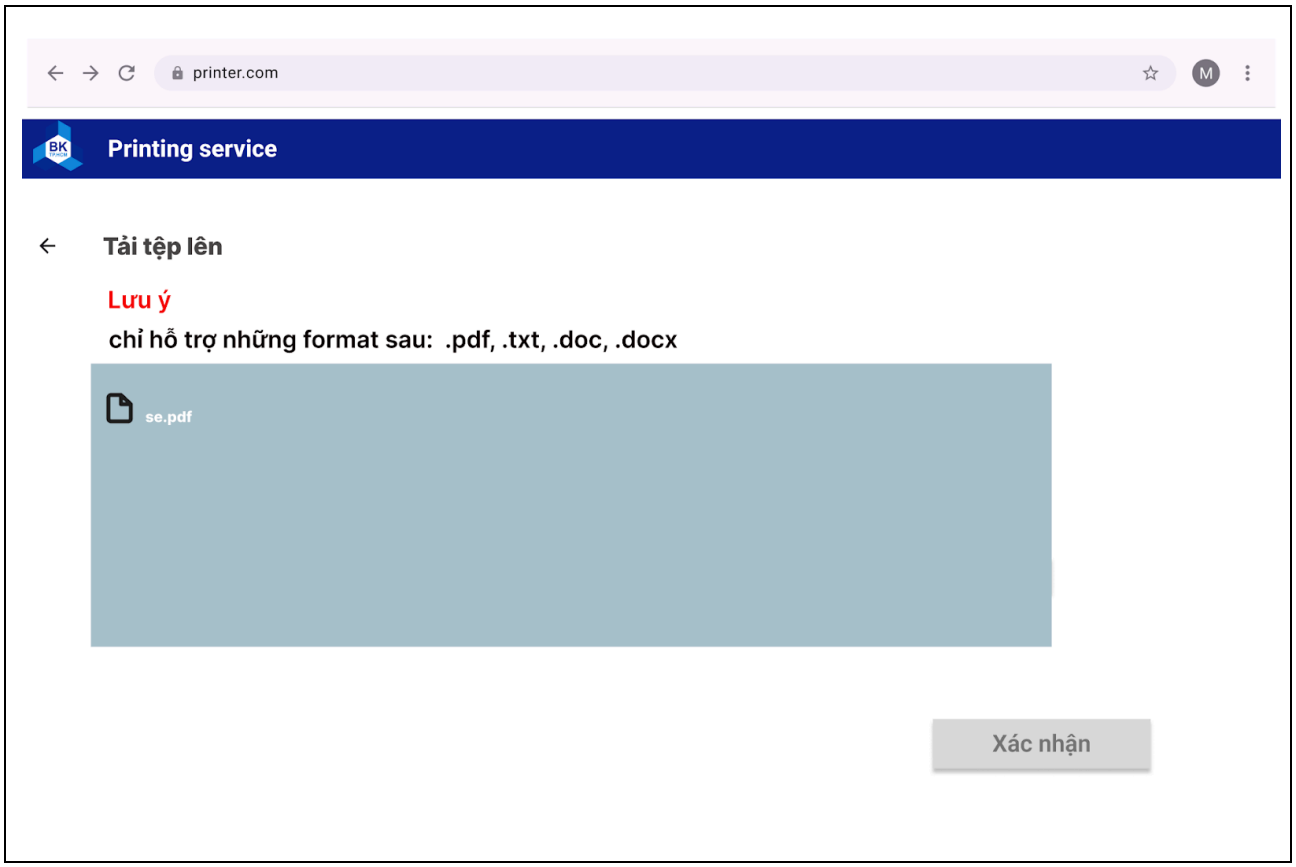


Figure 3.14. Illustrate for upload page 2

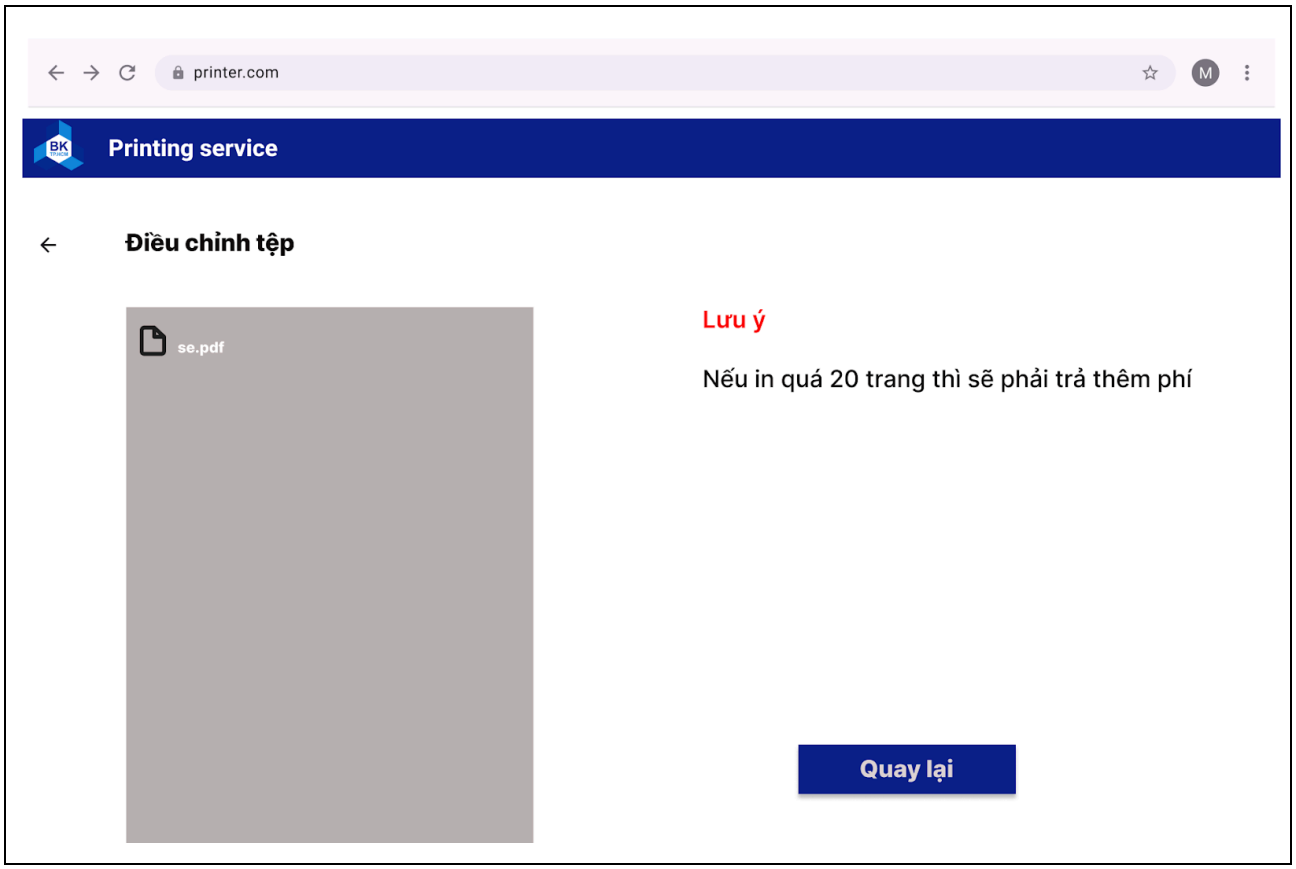


Figure 3.14. Illustrate for warning for pay fee to print more pages page

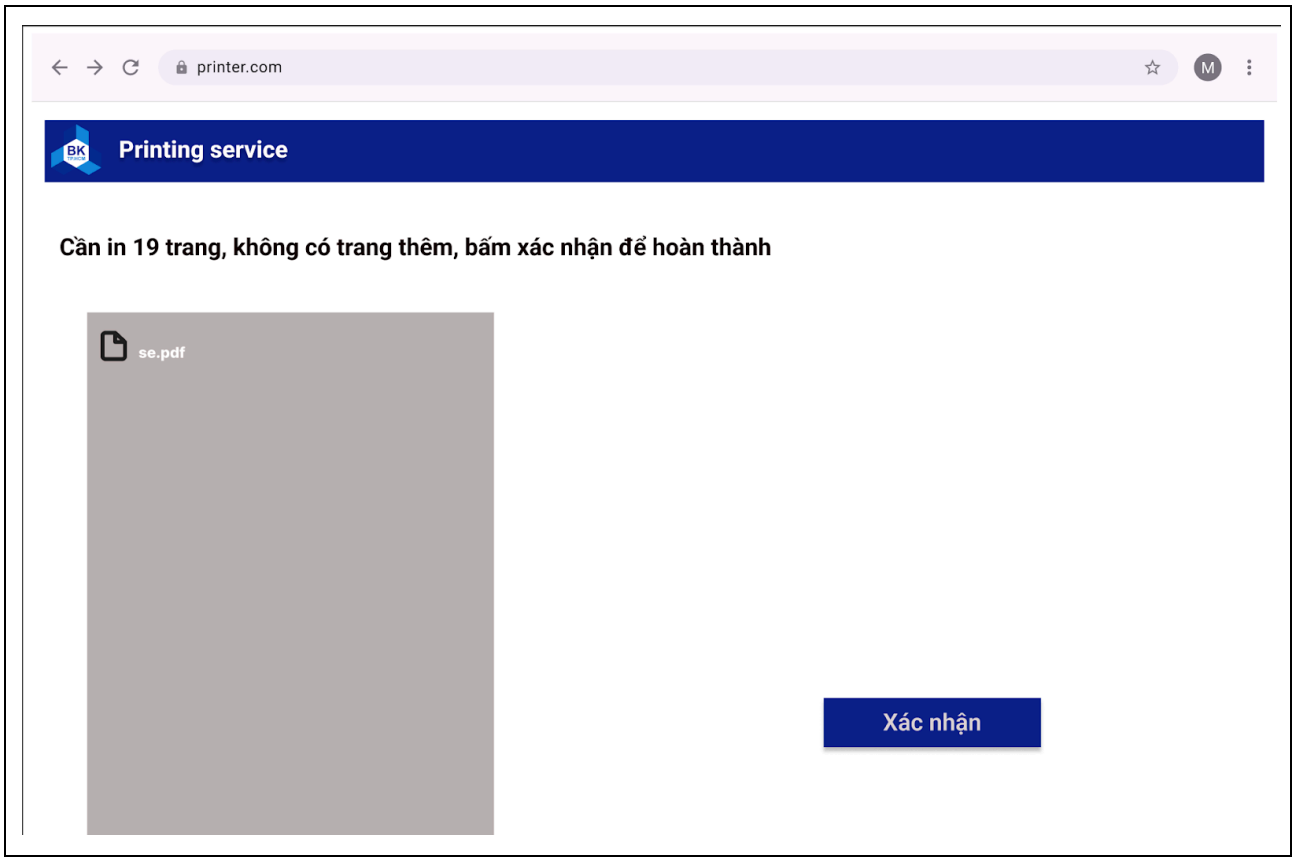


Figure 3.14. Illustrate for confirmation of printing page

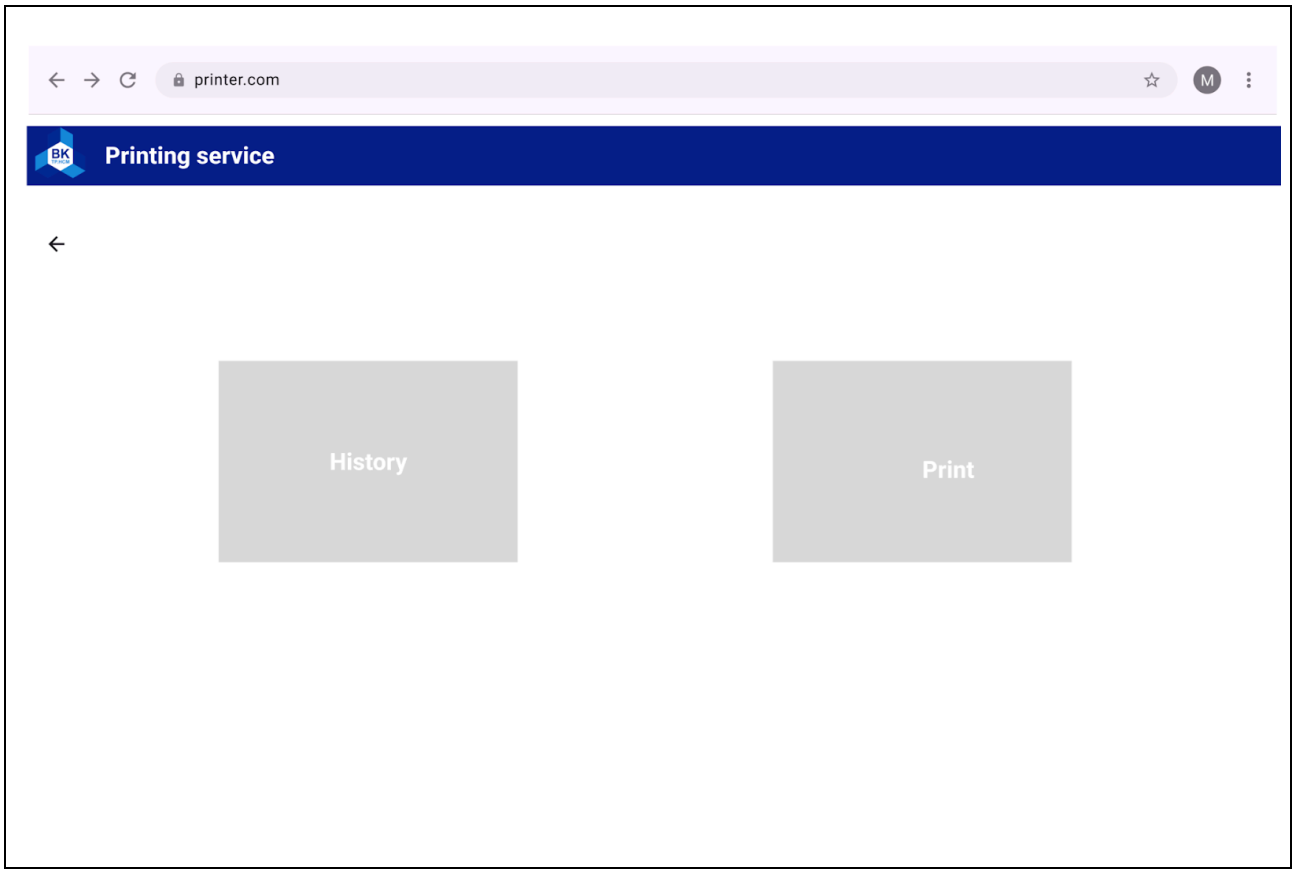


Figure 3.14. Illustrate for view log page

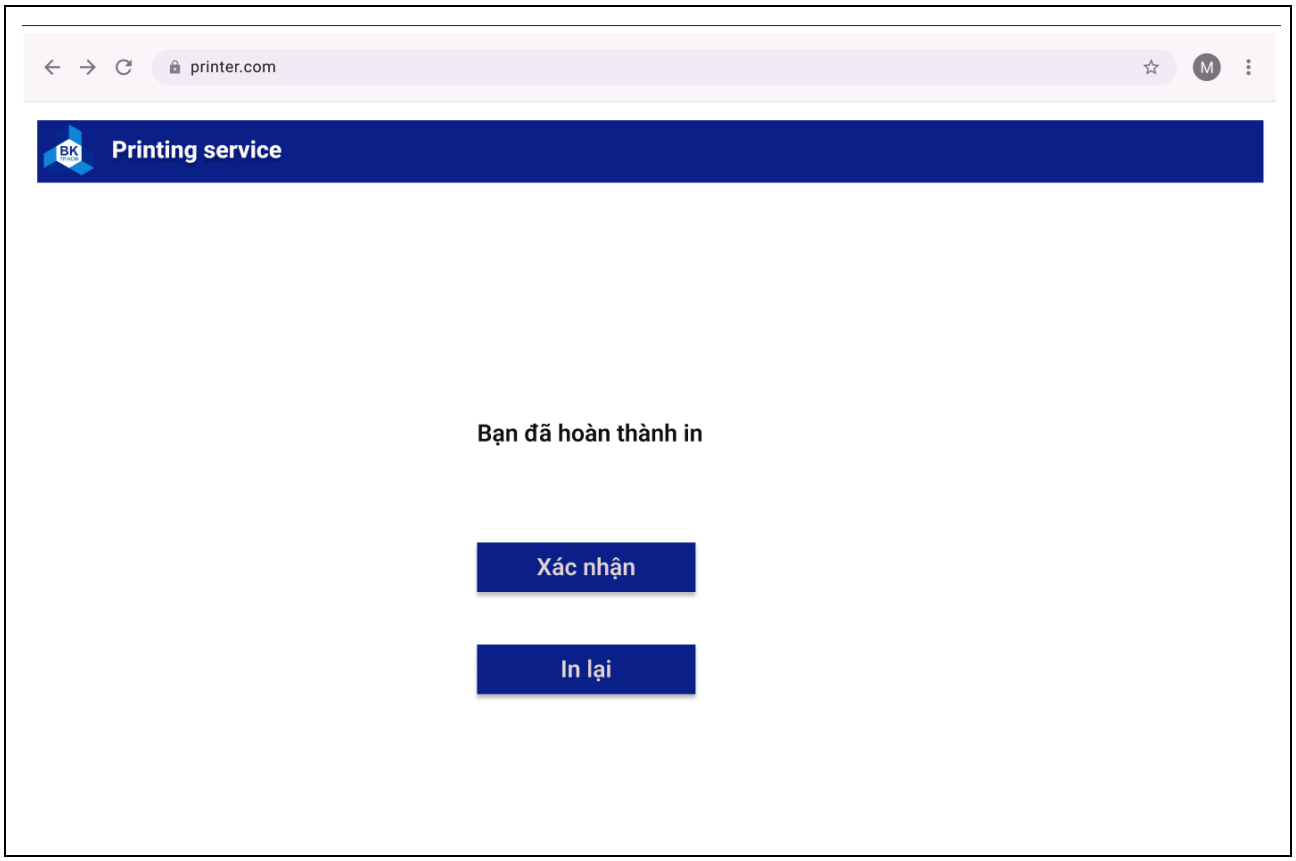
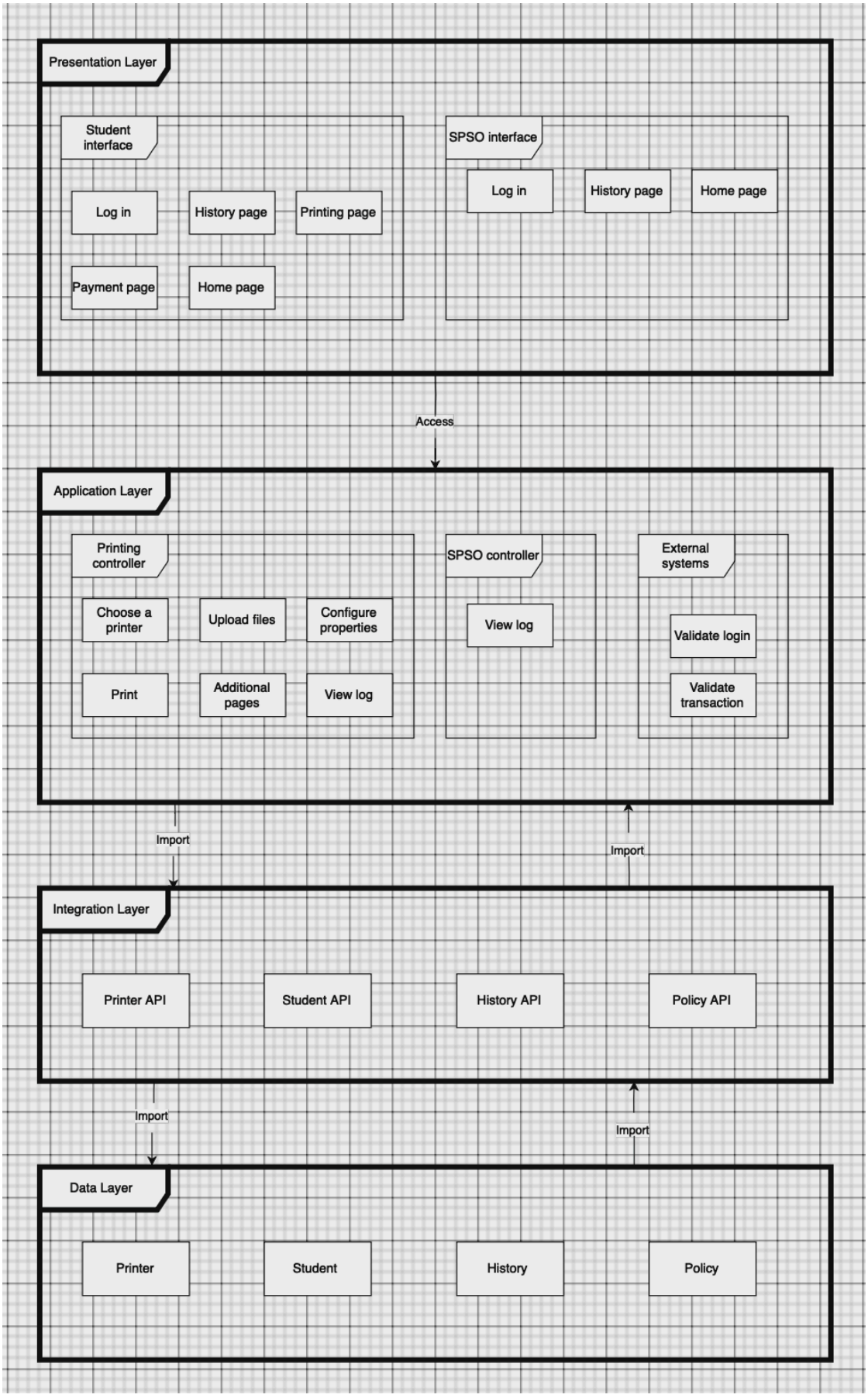


Figure 3.14. Illustrate for notification of complete priting page

Figure 3.14. Illustrate for QR code use to pay page

4. Architecture design

4.1. Layered Architecture


Presentation Layer
Student interface
Log in
History page
Printing page
Payment page
Home page
SPSO interface
Log in
History page
Home page
Access
Application Layer
Printing controller
Choose a printer
Upload files
Configure properties
Print
Additional pages
View log
SPSO controller
View log
External systems
Validate login
Validate transaction
Import
Import
Integration Layer
Printer API
Student API
History API
Policy API
Import
Import
Data Layer
Printer
Student
History
Policy

4.1.1. Presentation strategy

This layer represents how users (including SPSO and student) interact with the system.

1. Student Interface:
    - Login
    - Home page
    - Printing page
    - History page
    - Payment page
    - Error page
2. SPSO Interface:
    - Login
    - Home page
    - History page

Both interfaces share **login** and **home page** elements. **Login page** is used for validation, depending on this, the user will be directed to either SPSO **home page** or Student **home page**. The home page is the initial point for both types of user, having information about the user. Both users also share a **history page**, but the difference is that the student can only watch his own history, while the SPSO can watch the entire history.

For the student interface specifically, we have a **printing page** and **payment page**. In the **printing page,** we allow users to upload, choose, configure and print, while the payment page works as a pop-up window to create QR code for students to pay if there are any extra pages. In the event of a user error, such as incorrect printer configuration, **an error page** will be generated to alert the user and show the error(s). The system will automatically resolve the error by resetting the affected settings.

4.1.2. Application layer

1. **Printer controller**

- ***Choose the printer:*** The controller will call **Printer API** to return the list of available printers for the user. After the student chooses the printer through **Student API**, the controller will update the value in the **Printer Database** accordingly.
- ***Upload files:*** The controller will access via **Policy API** to get the supported files type. Then the student will upload file accordingly via **Student API**
- ***Configure properties:*** The user can change the configuration of the file via **Student API**, and the new configuration have to be coincide with the one called from **Policy API**
- ***Extra pages:*** The controller will access **Policy API** to get the supported pages number, if the required pages number is bigger, the system will direct to the payment page.
- ***Print:*** The student will call **Student API** to print, then the controller will access **Printer API** to print the documents, and modify the **History Database** via **History API**
- ***History***: The user will call **History API** to get the list of history

2. **SPSO controller**

- ***View log:*** The SPSO will access the list of entire history via **History API**

3. **External system**

Transaction validate: belong to BKpay system

Login validate: belong to authentication services

4.1.3. Data Layer

1. **Printer**

Attribute	Data type	Description
Printer ID	Int	Unique number for each printer
Printer Brand	String	Brand of the printer
Printer Location	String	Location of the printer
Status	Boolean	Status of the printer (active or in-used)

Attribute	Data type	Description
Settings	List	Comprise of settings for printer

2. Student

Attribute	Data type	Description
Student Id	Int	Unique number to identify between students
Student name	String	Name of the student
Faculty	String	Faculty of the student
Remaining pages	Int	Number of pages that student has left to printing
Bills	String	Bills that student has paid for extra pages

3. History

Attribute	Data type	Description
Print ID	Int	Unique number to identify printing object
Student ID	Int	To identify which student do the action
Printer Id	Int	Identify which printer was used
Datetime	Datetime	Datetime that student printed
Page size	String	Size of page (A0, A1, A2, ...)
Page number	Int	Number of page that was printed
File name	String	Name of the file printed
Location	String	Location to acquire the paper(s)

4. System policy

Attribute	Data type	Description
Modified date	Date	Most recently policy change date
Maximum file size	Int	Maximum size for student to upload
Supported file type	List	List of supported file for uploading
Supported Page	Int	Number of pages that student don’t have to pay

4.1.4. API management (Integrated layer)

1. Student API

Method	Passing parameter	Description	Return
Upload	File	Upload file to the server for printing	Notify for success
Configure	Page Size : String (A0,A1,..) File Name: String Page number: Int	Change the configuration of the file accordingly	Confirmation/Error if conflict the default settings
Print	None	Student print the file	Notify for success

2. Printer API

Method	Passing parameter	Description	Return
GetAvailablePrinter	None	Find all the available printer in database (one with status value is True)	List of Printer (Printer brand, Printer ID, Printer Location)
ChoosePrinter	Printer ID : Int	Change the Status of Printer with Printer ID to False (indicating unavailability)	None
PrinterFinish	Printer ID : Int	Change the Status of Printer with Printer ID to True (indicating availability)	None

3. History API

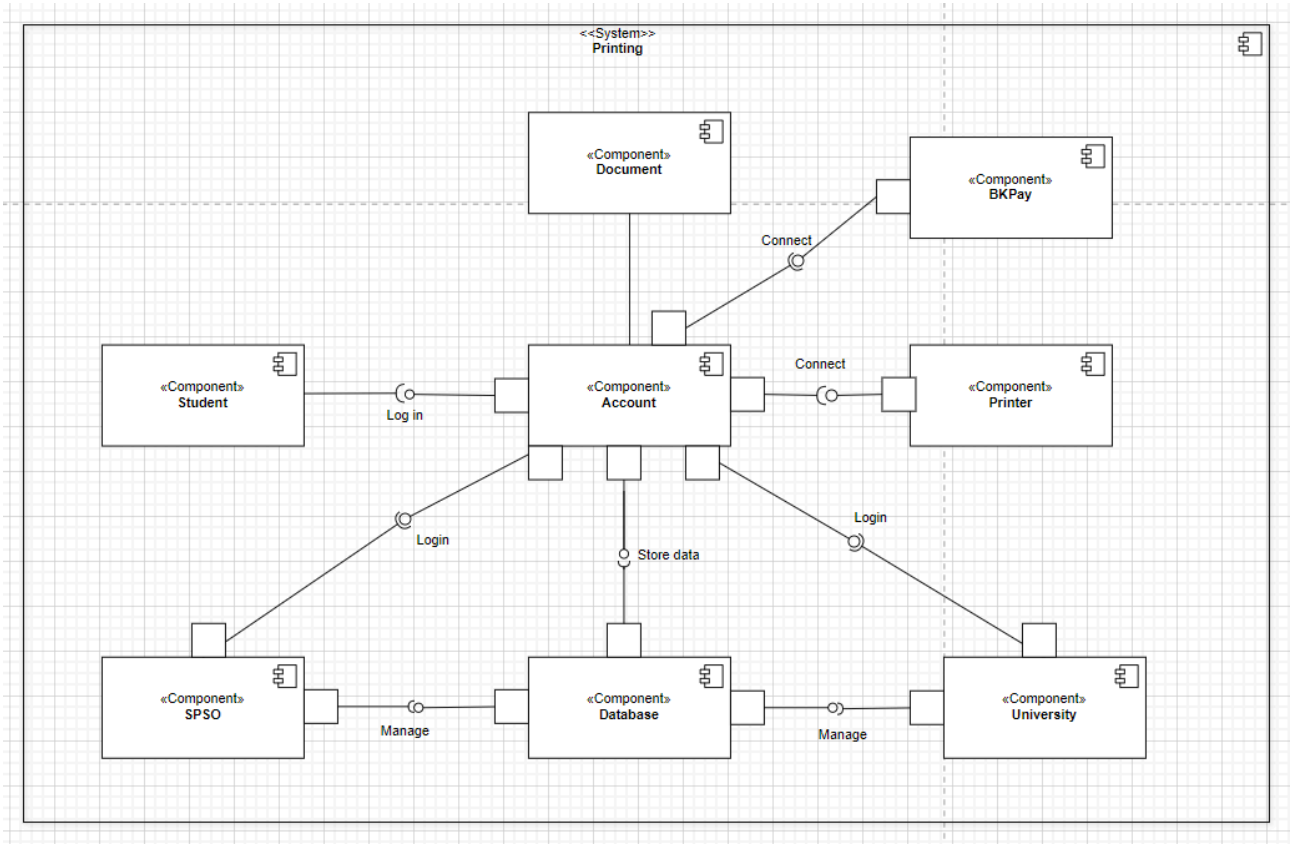
Method	Passing parameter	Description	Return
GetHistory	StudentID: Int	Find the history from from the database of Student with Student Id value	History(Print Id, PrinterID, Date, Page size, Page number, File name, Location,)
GetAllHistory	None	Return all the history of all student	History(StudentID,Print Id, PrinterID, Date, Page size, Page number, File name, Location,)
Add	StudentID: Int Print Id: Int PrinterID: Int Date: Date Page size: String Page number: Int File name: String Location: String	Student successfully print, add into history	None

4. Policy API

Method	Passing parameter	Description	Return
SupportFileType	None	Student access to get the list of supported file type saved in the database	List of Supported file type
ExtraPage	None	Access to get the number of pages supported	Page number: Int

Configure	None	Supported Configuration	Constraint on file name, page number and page size

4.2. Component Diagram



This component diagram represents a printing system that involves several interconnected components. At the center of the system is the **Account** component, which serves as the core hub interacting with multiple other components.

1. **Student** and **SPSO** (possibly representing a user management system) components are connected to the **Account** component through login interactions. This suggests that both students and system administrators authenticate through this central **Account** component.

2. The **Account** component also stores data in the **Database**, indicating that user and system data related to accounts, transactions, or printing activities are saved there.
3. There are connections between the **Account** component and other parts of the system like **Document**, **Printer**, and **BKPay**. The connection with the **Document** suggests document management functionality, possibly uploading or selecting documents for printing. The **Printer** component connects to **Account**, suggesting printing actions are authorized through the user's account. **BKPay** is likely involved in payment processing, enabling users to pay for their printing services via the account.
4. The **University** component is responsible for managing system resources, indicating some administrative or integration role, perhaps managing user permissions or providing system-wide data integration.

In summary, this diagram outlines a structured system where the **Account** component acts as a central point of access for users (students and administrators), allowing interaction with printers, payment systems, documents, and the database, all managed under the larger university system framework.

5. Implementation first time

5.1. Setting up Github and using Github

- Link Github for our group: https://github.com/LancerHihi/SE-Group-7

5.2. Conducting a usability test

This report summarizes the findings from the usability testing conducted on the HCMUT Smart Student Printing Service (HCMUT-SSPS). The testing aimed to evaluate the system's usability, identify potential issues, and gather user feedback to inform future improvements.

- ***Goals***

The usability testing sought to achieve the following goals:

+ Identify usability issues: The testing aimed to uncover any challenges users encounter while interacting with the system.

+ Evaluate effectiveness and efficiency: The testing assessed the system's ability to meet user needs, including the ease and speed of task completion.
+ Measure user satisfaction: The testing gauged overall user satisfaction with the system.

- ***Methodology***

A mixed-methods approach was employed, combining both qualitative and quantitative data collection techniques:

+ Qualitative Methods: Think-aloud protocols and semi-structured interviews were used to gather in-depth insights into user experiences.
+ Quantitative Methods: Task completion rates, task completion times, error rates, and the System Usability Scale (SUS) were measured to obtain objective data.

- ***Participants***

A total of 10 participants were recruited, consisting of 5 students and 5 SPSO staff (just random person). Participants were selected to represent a diverse range of technical expertise and usage scenarios.

- ***Testing Setup, Moderation & Post-Test Questions***

Testing Setup: Usability testing took place in both in-person and remote settings. In-person testing was conducted in a controlled lab environment at HCMUT, while remote testing allowed participants to participate using their own devices.

Moderation: Testing sessions were moderated by trained researchers who guided participants through tasks, observed interactions, and conducted post-test interviews.

Post-Test Questions:

1. *What are your overall impressions of the system?*
2. *Did you find the system easy to use? Why or why not?*
3. *Were you able to complete the tasks efficiently?*
4. *What aspects of the system did you find most helpful?*

5. *What aspects of the system did you find most challenging?*
6. *What suggestions do you have for improvement?*

- ***Findings***
    - Qualitative Findings

Common Issues:

+ Lack of Clarity in the Printing Credit System: Participants frequently expressed confusion regarding the mechanics of the printing credit system, including the accrual, usage, and replenishment processes.
+ Incomprehensible Error Messages: Users encountered difficulty understanding error messages, which often lacked specificity and actionable guidance.
+ Inefficient Navigation: Navigating through the system's interface proved challenging for some participants, particularly when performing complex tasks or accessing specific functionalities.

Positive Feedback:

+ Mobile Web Convenience: Participants appreciated the convenience of accessing the printing service through the mobile app, enabling them to monitor their printing history, check credit balances, and submit print jobs remotely.
+ Printing History Tracking: The ability to track printing history was well-received, as it allowed users to review past print jobs, identify trends, and optimize their printing behavior.
+ Visually Appealing Interface: The system's visually appealing interface was generally praised, with participants noting that it was easy on the eyes and contributed to a positive user experience.

    - Quantitative Findings

Task Completion Rates:

+ Task 1 (Account Registration): 100%
+ Task 2 (Document Upload): 90%

+ Task 3 (Pay for additional pages): 60%
+ Task 4 (Configure settings for printing): 80%

Task Completion Times:

+ Task 1: 10 seconds
+ Task 2: 3 seconds
+ Task 3: 180 seconds
+ Task 4: 60 seconds

Error Rates:

+ Task 1: 0%
+ Task 2: 10%
+ Task 3: 40%
+ Task 4: 20%

SUS Score: 75%

- ***Discussion***

The usability testing revealed several key findings:

Key Findings from the Usability Testing:

+ System Efficiency: While the system demonstrated reasonable efficiency, certain tasks, including Account Registration and Document Upload, were identified as potential areas for improvement. Streamlining these processes could significantly reduce user effort and enhance overall system performance.
+ User Satisfaction: Participants generally expressed satisfaction with the system, particularly commending its convenience and user-friendly interface. These positive perceptions indicate that the system effectively meets the needs of its target users.

Areas for Improvement:

+ Clarity of Information: The clarity of error messages and instructional text should be enhanced to minimize user confusion and frustration.
+ Navigation Structure: The navigation structure could be improved to make it more intuitive and efficient, reducing the time required to complete tasks.
+ Printing Credit System: Clearer explanations and visualizations of the printing credit system should be provided to improve user understanding and alleviate concerns.

5.3. Demonstration of the application

- In this application, our group implemented a successful backend and frontend using just HTML, CSS and JavaScript, and no framework is used.
- First of all, the testers should run the file ***account_signin1.html*** in the folder 'Thanh' first.
- In the demonstration, we register an account for student with

Họ và tên	Nguyễn Hữu Thanh
Số điện thoại	123456789
Khoa	Khoa học máy tính
Email	thanh@hcmut.edu.vn
Tên đăng nhập	thanh
Mật khẩu	123
Xác nhận mật khẩu	123

- Then we use the account has just been registered to sign in the application
- If you want to sign in the BKPay simulator, you just need to sign in with these informations

Thông tin cá nhân	
Tên người dùng	Nguyễn Hữu Thanh
Email	thanh@hcmut.edu.vn
Thông tin OCB	

Tên người dùng	thanh
Mật khẩu	123

- Then after you have done with the student view, you should enter the information of administrator account with:

Tên đăng nhập	admin
Mật khẩu	123

6. Future Improvement

- In the future, we plan to enhance our application by integrating modern frameworks and tools to streamline development and improve performance. Currently built using plain HTML, CSS, and JavaScript, we aim to adopt Node.js to enable real-time functionalities, such as live updates and instant user interactions, which will significantly elevate the user experience. Additionally, transitioning to Tailwind CSS will allow us to design and style the application more efficiently by leveraging its utility-first approach, saving development time and ensuring consistency across components. These upgrades will not only improve maintainability and scalability but also prepare the application for more complex and dynamic use cases.

7. References

1. Sommerville, I. (2015). Software Engineering, Global Edition.
2. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/
3. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/