

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



## SOFTWARE ENGINEERING (CO3001)

---

### **Software Requirements Specification For:**

# **A smart printing service for students at HCMUT**

---

Instructor(s): Trương Tuấn Anh, CSE-HCMUT

Group: CC03-09

Student(s):

Trịnh Anh Minh - 2252493

Trần Xuân Hào - 2252191

Vũ Quỳnh Hương – 2252286

Vũ Minh Quân – 2212828

Võ Cao Nhật Minh – 2252495

Trần Quốc Bảo Long – 2252453

HO CHI MINH CITY, NOVEMBER 2024



## Table Of Content

<b>Revision History .....</b>	<b>2</b>
<b>1. Task 1: Requirement elicitation (1.1, 1.2) .....</b>	<b>3</b>
1.1 Domain Context .....	3
1.2 Stakeholders and Needs .....	3
1.3 Benefits of the System.....	4
1.4 Functional Requirements.....	5
1.5 Non- Functional Requirements .....	7
<b>2. Task 1: Requirement elicitation (1.3) .....</b>	<b>9</b>
2.1 Use case Diagram for the Whole System .....	9
2.2 Use case Diagram for Authentication .....	10
2.3 Use case Diagram for Printer Management.....	14
2.4 Use case Diagram for Purchasing Paper .....	18
2.5 Use case Diagram for Management Configuration .....	22
2.6 Use case Diagram for System Report.....	29
2.7 Use case Diagram for Management Account .....	33
2.8 Use case Diagram for Printing Service .....	38
<b>3. Task 2: System modelling.....</b>	<b>46</b>
<b>3.1 Activity Diagram.....</b>	<b>46</b>
3.1.1 Change configuration for SPSO.....	46
3.1.2 Display Uploading Files Page .....	47
3.1.3 Uploading File for Student.....	48
3.1.4 Printing Configuration for Student.....	49
3.1.5 Printing Service for Students.....	50
<b>3.2 Sequence Diagram .....</b>	<b>51</b>
3.2.1 Change configuration for SPSO.....	51
3.2.2 Display Uploading Files Page .....	53
3.2.3 Sequence Diagrams for Uploading Files.....	54
3.2.4 Sequence Diagram for Printing Service .....	55
3.2.5 Sequence Diagram for Printing Configuration for Students .....	57
<b>3.3 Class Diagram .....</b>	<b>60</b>
3.3.1 PrintServiceView() .....	61



3.3.2 PrintServiceController() .....	62
3.3.3 Student() .....	62
3.3.4 Printer() .....	63
3.3.5 Printer 1, 2, 3, 4, 5() - inheritance from the Printer.....	63
3.3.6 Document().....	63
3.3.7 PrintConfiguration() .....	63
3.3.8 PrintConfigurationView().....	64
3.3.9 PrintingConfigurationModel() .....	65
3.3.10 PrintingConfigurationController() .....	65
3.3.11 SPSO().....	65
3.3.12 Database().....	65
<b>3.4 User Interfaces using Figma .....</b>	<b>67</b>
3.4.1 Main screen .....	67
3.4.2 Pre-login screen.....	68
3.4.3 Login page.....	68
3.4.4 Main menu for the student .....	69
3.4.5 Print document page.....	70
3.4.6 Uploading file page .....	71
3.4.7 Uploaded file page .....	71
3.4.8 Confirm file to go to the configuration .....	72
3.4.9 Configuration file format page.....	72
3.4.10 Invoice page .....	73
3.4.11 Popup successfully print page .....	74
3.4.12 System configuration page .....	74
3.4.13 System config in use .....	75
3.4.14 Popup successfully save page .....	76



<b>4. Task 3: Architecture design</b>	<b>77</b>
<b>4.1 Layered Architecture:</b>	<b>77</b>
<b>4.2 Present User Interface:</b>	<b>78</b>
<b>4.3 API management:</b>	<b>78</b>
<b>4.4 Store Data:</b>	<b>79</b>
4.4.1 SPSO:	80
4.4.2 Student:	80
4.4.3 Printer:	80
4.4.4 Document:	80
4.4.5 Paper_Transaction:	81
<b>4.5 Deployment Diagram:</b>	<b>81</b>
<b>4.6 Component Diagram:</b>	<b>82</b>
4.6.1 View:	83
4.6.2 Controller:	83
4.6.3 Model:	84
4.6.4 Database:	84
<b>5. Task 4: Implementation – Sprint 1</b>	<b>85</b>
<b>5.1 Setting up an online repository (github, bitbucket, etc) for version control</b>	<b>85</b>
<b>5.2 Adding documents, materials and folders for Requirement, System modelling and Architectural design. Use the selected version control system to report the changes to these files.</b>	<b>85</b>
<b>5.3 Conducted a usability test with the user interface.</b>	<b>86</b>
5.3.1 Recruit participants/ testers.	86
5.3.2 Define tasks.	87
5.3.3 Define test strategy (qualitative vs. quantitative, remote vs. inperson.)	87
5.3.4 Conduct the test.	87
5.3.5 Document the feedback from testers.	88
5.3.6 Improve the project	89



## Revision History

Name	Date	Reason For Changes	Version
Printing Service	September 22 <sup>nd</sup> 2024	Update task 1.1, 1.2	1.0
Printing Service	October 1 <sup>st</sup> 2024	Update task 1.3	1.1
Printing Service	October 27 <sup>th</sup> 2024	Update task 2	1.2
Printing Service	November 15 <sup>th</sup>	Update task 3	1.3



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

### 1.1 Domain Context

The **Student Smart Printing Service (HCMUT\_SSPS)** is a system designed to help students print documents on campus through a web-based and mobile app. The key entities of this system are **Students, University Administrators, School IT Administrators, Printer Manufacturer and the Student Printing Service Officer (SPSO)**. The system allows students to upload documents, select printers, and specify printing properties such as paper size, single or double-sided printing, and the number of copies. Each print job records details like student ID, printer ID, file name, start and end times, and the number of pages printed. The print history is automatically deleted after a set period, and printing activity is restricted based on a student's department and managed by department-specific administrators or the central manager.

The system includes a feature where students can purchase additional printing pages through BKPay once their balance is depleted. Every print job checks the remaining page balance, and page allocation is managed by the SPSO, who configures limits such as 1 A3 page equaling 2 A4 pages. The SPSO also controls the default number of pages issued to students each semester, manages file types allowed for printing, and can view monthly and yearly usage reports generated by the system. The system is highly customizable, enabling the SPSO to activate or deactivate printers and adjust the printing configurations as needed.

To ensure security, all users, including students and administrators, must authenticate through the HCMUT\_SSO service before accessing the system. This maintains the integrity and privacy of the system, ensuring that only authorized users can access printing services. The system's design ensures efficient management of resources while making it easy for students to print documents across campus.

### 1.2 Stakeholders and Needs

The HCMUT-SSPS (Student Smart Printing Service) gives services and is maintained to support users. Therefore, there are stakeholders in this system corresponding to their needs.

The first stakeholder is the **Students**. They are the main users of the system: uploading documents, selecting printers, specifying printing properties, etc. They can also view their printing log for a time period together with a summary of the number of printed pages for



each page size and require the printing and BKPay app to be easy to user-friendly, flexible and reliable.

The second stakeholder is the **Student Printing Service Officer (SPSO)**, who is in charge of permitting and configuring certain file types, traversing and monitoring the printing history of students for a time period (date to date) and for all or some printers. The officer is provided the permission to manage printers such as add, enable, disable a printer. The officer also manages other configurations of the systems such as changing the default number of pages, the dates that the system will give the default number of pages to all students, the permitted file types accepted by the system. The reports of the use of the printing system is viewed by SPSO. A user-friendly and reliable managing app is also required by the SPSO.

The third stakeholder is the **University Administrators**, which decides the default number of A4-size pages each student has.

**The Printer Manufacturer** is another stakeholder. The manufacturer is responsible for providing printers, maintenance services if such needs arise.

The last stakeholder is the **School IT Administrators**. The IT administrators manage the HCMUT\_SSO authentication service. They will provide access to users via the authentication service and are also responsible for the maintenance and development of the system. The admins must make sure that the web-based app and mobile app works correctly and available at all times and without errors.

### **1.3 Benefits of the System**

The **HCMUT-SSPS (Student Smart Printing Service)** offers multiple benefits to various stakeholders. For **students**, the system provides convenience and efficiency by allowing them to print documents from any campus location, track their printing history, and manage their page quota. This service reduces the need to locate specific printers or manually handle print jobs, which streamlines their academic workflow. Additionally, students have flexibility to purchase extra pages when needed, which ensures uninterrupted printing service. According to studies on self-service systems in education, such functionalities increase student satisfaction and reduce administrative burden.

For the **Student Printing Service Officer (SPSO)**, HCMUT-SSPS offers a centralized management system, allowing real-time monitoring of printer performance and usage logs. The ability to manage printer configurations and access detailed reports helps the



SPSO maintain efficiency and optimize resource distribution. Automated reporting tools reduce the workload of manual data collection, while configurable settings—such as page quotas and file types—ensure adaptability to evolving student demands. Effective management of such services has been shown to increase operational efficiency in university environments.

For **University administrators**, the system generates valuable insights through periodic reports on printer usage, which aid in decision-making processes, such as budgeting and resource allocation. By supporting seamless printing services, the system enhances the university's infrastructure and promotes a tech-enabled campus environment. This aligns with the broader trend of universities adopting smart technologies to improve service delivery and student satisfaction. Furthermore, integrating online payment systems like BKPay streamlines revenue collection for additional printing services, providing a financial model that supports both sustainability and growth.

For **Printer Manufacturers**, manufacturers benefit from the system's centralized printer management, which allows early detection of issues such as low ink levels or paper shortages. This ensures prompt maintenance and servicing, extending the lifespan of printers and reducing the risk of downtime. The system's usage data also helps manufacturers optimize their support and service offerings.

For **School IT Administrators**, IT administrators play a critical role in ensuring the HCMUT-SSPS system's availability and security. By integrating the HCMUT\_SSO authentication system, IT administrators ensure that only authorized users access the system. Additionally, they oversee system maintenance and troubleshooting, ensuring that both the web-based and mobile apps function efficiently. This ensures a smooth and secure printing experience for all users.

## **1.4 Functional Requirements**

### **Students:**

1. Upload documents to print.
2. Choose between black-and-white printing and color printing.
3. View printing history, including usage details like number of papers, files, etc.
4. Access a predefined quota of free prints per semester.





5. Pay for additional prints beyond the free quota via the BKPay system.

**University Administrators:**

1. Manage accounts - they can create, suspend or delete student accounts.
2. Check the printer status, like ink and paper levels.
3. Adjust default number of A4-size pages for printing.
4. Adjust (decrease/increase) prices for the number of pages.
5. Generate usage statistics reports (number of students, amount of paper used, revenue...)

**School IT Administrators:**

1. Grant students access to the system through the HCMUT\_SSO authentication service.
2. Receive feedbacks about the system.
3. Ensure that user data and system logs are securely stored and protected from unauthorized access.
4. Troubleshoot system-related issues, ensuring all system functionalities operate correctly (e.g., file uploads, payment processing).
5. Update and maintain system infrastructure, including software updates for security and compatibility with new devices.

**Printer Manufacturer:**

1. Provide updates to ensure compatibility between the system and new printer models.
2. Monitor and receive alerts for printer maintenance needs such as low ink levels, paper jams, or hardware failures.
3. Conduct periodic system maintenance to optimize printer performance.
4. Provide remote troubleshooting assistance when printers encounter issues.
5. Provide instructions and installation manuals.

**Student Printing Service Officer (SPSO):**

1. Manage printers such as add, enable, or disable a printer.



2. Manage configurations of the system (changing number of pages, the dates that the system will give the default number of pages to all students, the permitted file types accepted by the system).
3. Permit file types for printing (.jpg, .png, .pdf, .doc...)
4. View the reports of the use of the printing system.
5. Generated automatically at the end of each month and each year, stored in the system.
6. View the printing history (log) of all students or a student for a period (6 months) and for some/all printers.

## **1.5 Non- Functional Requirements**

### **Performance Requirements:**

1. The system shall display printing logs within 2 seconds.
2. The system shall respond to printing requests within 2 seconds.

### **Security Requirements:**

1. Students shall authenticate themselves using the school account granted to them by the university.
2. The system shall ensure that only authenticated and authorized students and SPSO have access to their respective functionalities.
3. The system shall limit access to printer management and configuration settings to the SPSO only.

### **Usability Requirements:**

1. The system's user interface shall be intuitive and user-friendly, both on the web-based app and mobile app.
2. The system shall provide clear error messages and instructions when user actions fail (e.g., insufficient page balance, unsupported file format).

### **Reliability and Availability Requirements:**

1. The SSPS shall be available for printing to all students during normal working hours (Mon–Fri, 08:30–17:30).
2. Downtime within normal working hours shall not exceed five seconds in any one day.
3. The log and reports shall be available for viewing at any time.



4. The system shall save the logs of all students for up to 6 months.

#### **Scalability Requirements:**

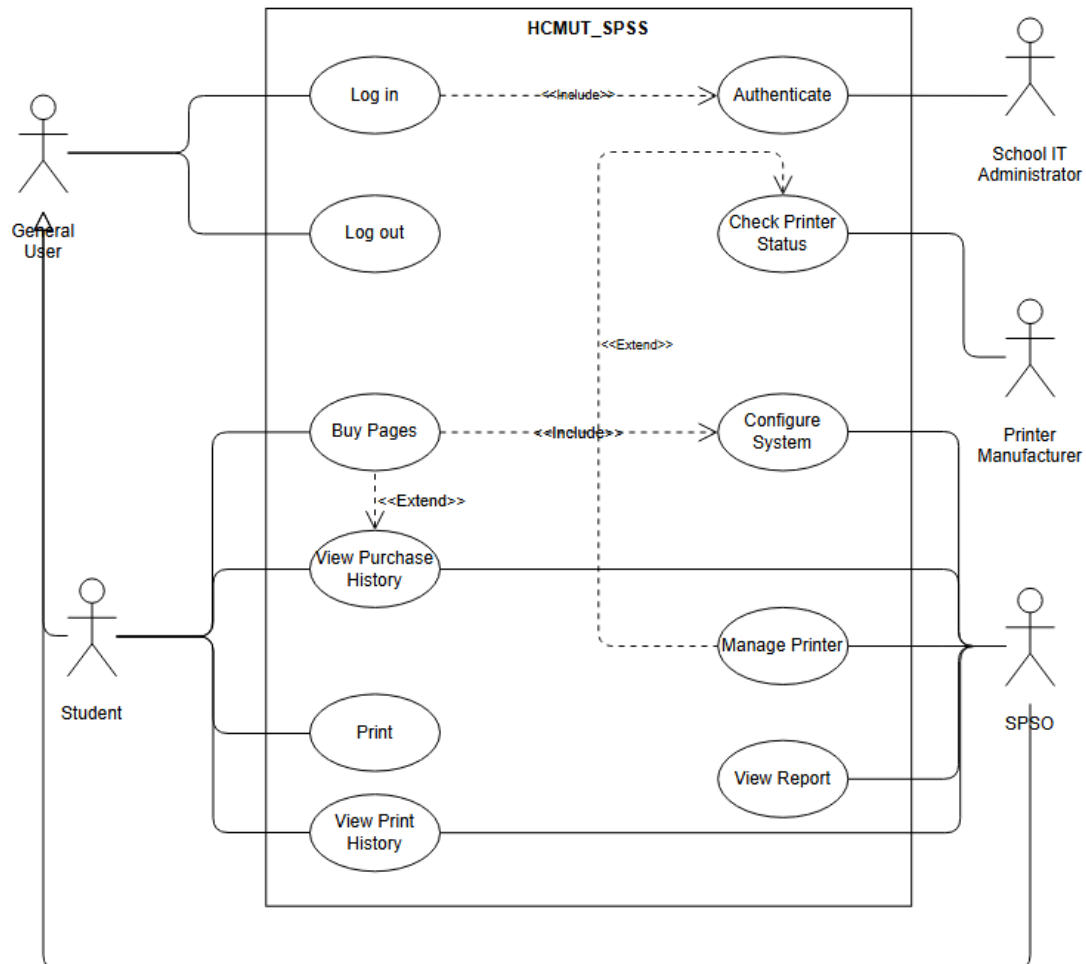
1. The system shall be scalable to support up to 10 additional printers as needed without degradation of performance.
2. The system shall be able to handle up to 1,000 simultaneous users, including both students and SPSO.

#### **Compatibility Requirements:**

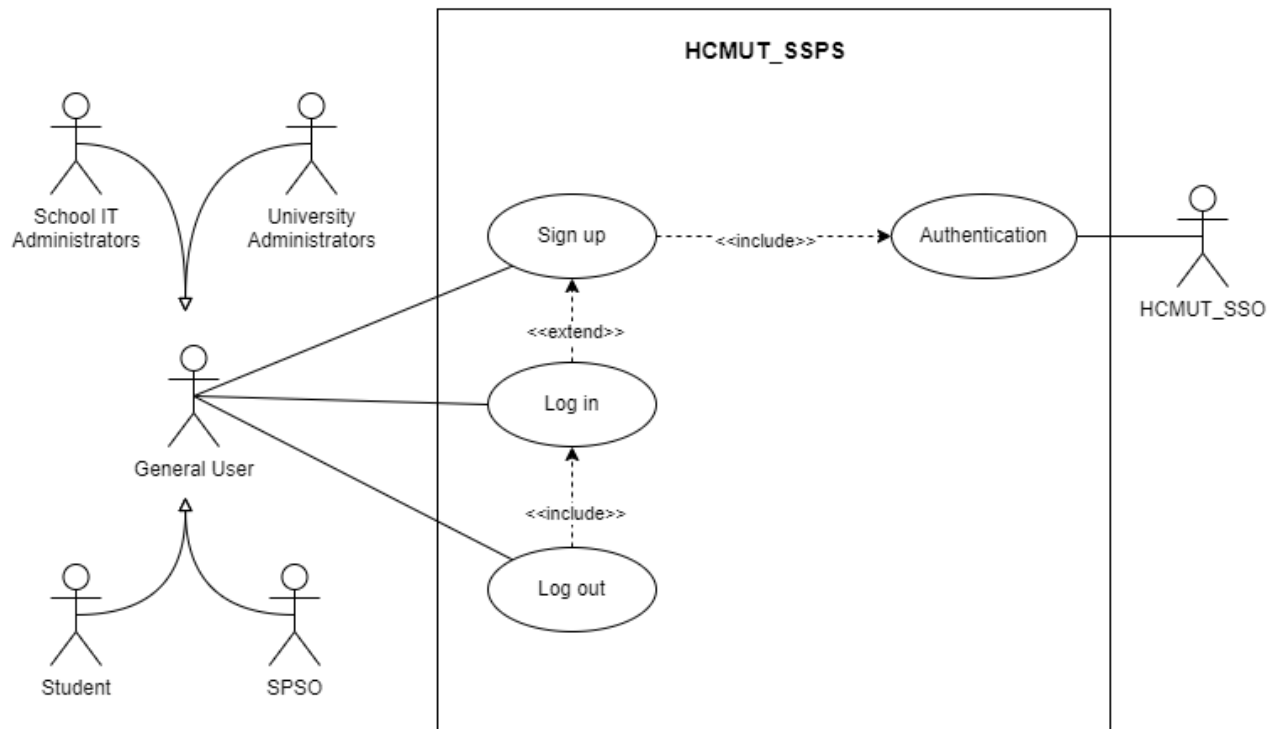
1. The system shall be compatible with all major operating systems, including Windows, macOS, Linux, iOS, and Android, for both the web and mobile applications.
2. The web-based app shall support and function correctly on all modern web browsers, including Chrome, Firefox, Safari, and Edge, in their most recent and at least two previous versions.
3. The system shall support the most commonly used document file formats for uploading, such as PDF, DOCX, and PPTX, as configured by the SPSO.
4. The system shall be compatible with printers from multiple manufacturers and support common printer models that can handle printing requests based on specified properties like paper size, color, and duplex printing.

## 2. Task 1: Requirement elicitation (1.3)

### 2.1 Use case Diagram for the Whole System



## 2.2 Use case Diagram for Authentication



<b>Name</b>	Sign Up
<b>ID</b>	
<b>Actor</b>	Student
<b>Description</b>	A page for the student to sign up new account
<b>Trigger</b>	Click on the button “Sign Up”

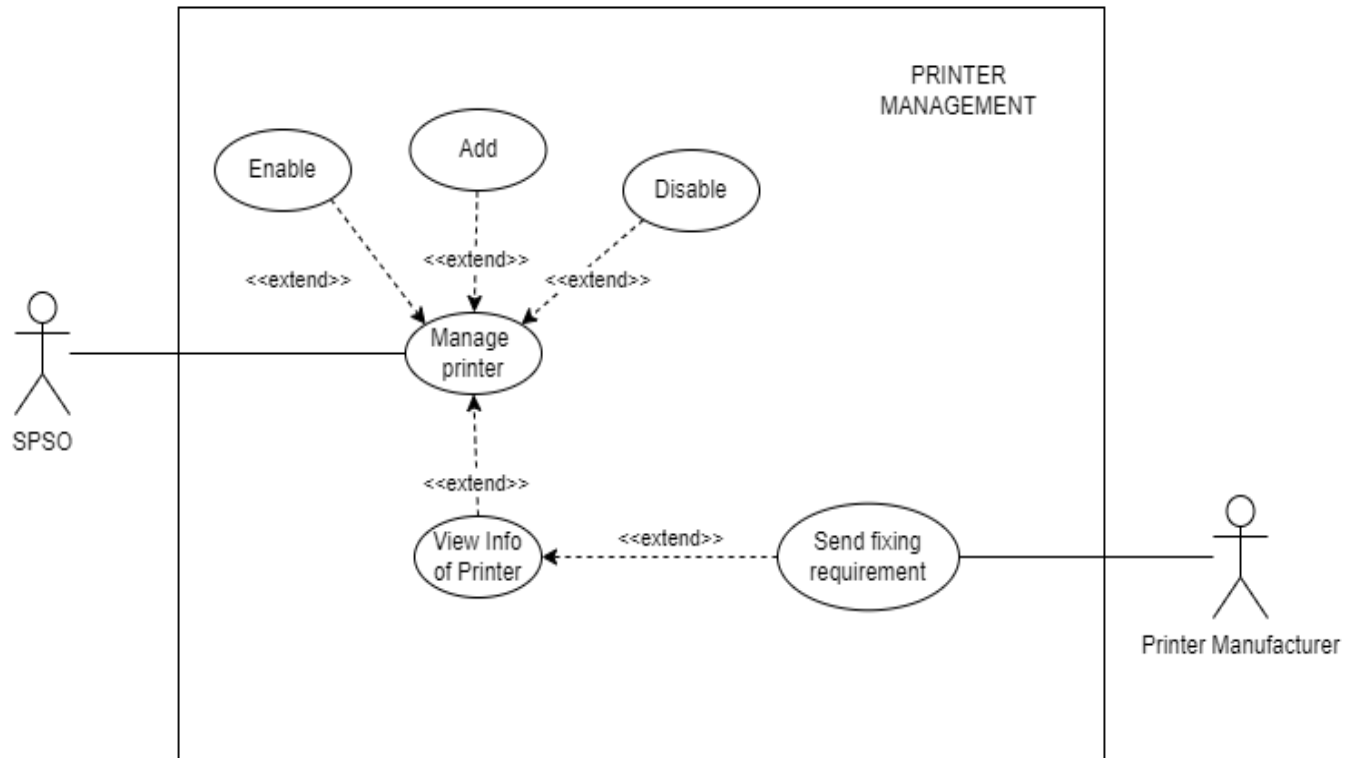
<b>Precondition</b>	Student does not have an account
<b>Postcondition</b>	Student successfully created a valid account.
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. Student inputs in “ID” section</li> <li>2. Student inputs in “Email” section</li> <li>3. System checks in HCMUT_SSO database to make sure the given ID and email is correct</li> <li>4. System sends a 6-digits code to student’s email</li> <li>5. Student enters the given code in the “Enter given code” sections</li> <li>6. Student enters the password in the “Enter password” section</li> <li>7. Student re-enters the password in the “Confirm password” section and press “Confirm”</li> <li>8. The student has successfully created an account, directed to “Log in”</li> </ol>
<b>Alternative Flow</b>	
<b>Exception Flow</b>	<p>At step 3, if the given ID and email do not map anything in the HCMUT_SSO</p> <ol style="list-style-type: none"> <li>3.1 Trigger an error log</li> <li>3.2 Student return to step 1</li> </ol> <p>At Step 4, if student does not receive the email</p> <ol style="list-style-type: none"> <li>4.1 Press “Send code again” to receive another code</li> <li>4.2 Re-enter the given code</li> </ol> <p>At step 6, if the “confirm password” does not match the “password”</p> <ol style="list-style-type: none"> <li>6.1 Trigger an error log</li> <li>6.2 Re-enter password</li> </ol>

<b>Name</b> <b>ID</b>	Log In
<b>Actor</b>	General User
<b>Description</b>	A page for the user to log in
<b>Trigger</b>	When user open the web/app
<b>Precondition</b>	User that has an account
<b>Postcondition</b>	Student successfully logged in the web/app.
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. Student inputs in “ID” section</li> <li>2. Student inputs in “Email” section</li> <li>3. Student inputs in “Password” section</li> <li>4. Student inputs in “Captcha” section</li> <li>5. Student presses “Log In”, then successfully enter the web/app</li> </ol>
<b>Alternative Flow</b>	No
<b>Exception Flow</b>	<p>At step 5, if the ID or Email or Password or Captcha is invalid</p> <ol style="list-style-type: none"> <li>5.1 Trigger a respective error log</li> <li>5.2 User return to step 1</li> </ol>

<b>Name</b> <b>ID</b>	Log Out
<b>Actor</b>	General User
<b>Description</b>	User gets out of web/app functional pages and gets back to log in page
<b>Trigger</b>	Click on the button “Log Out”
<b>Precondition</b>	Student has logged in the web/app
<b>Postcondition</b>	Student has successfully logged out system will not provide any functionality until re-logging in
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. Student press “Log Out” button</li> <li>2. System triggers a pop up has a button “Confirm Log Out”, then logs user out</li> <li>3. User gets back to log in page</li> </ol>
<b>Alternative Flow</b>	No
<b>Exception Flow</b>	<p>At step 2, system gets an error while logging out</p> <ol style="list-style-type: none"> <li>2.1 System trigger an error log</li> <li>2.2 System remains status and does not log user out</li> </ol>



## 2.3 Use case Diagram for Printer Management



<b>Name</b>	Adding/Removing Printer
<b>ID</b>	
<b>Actor</b>	SPSO

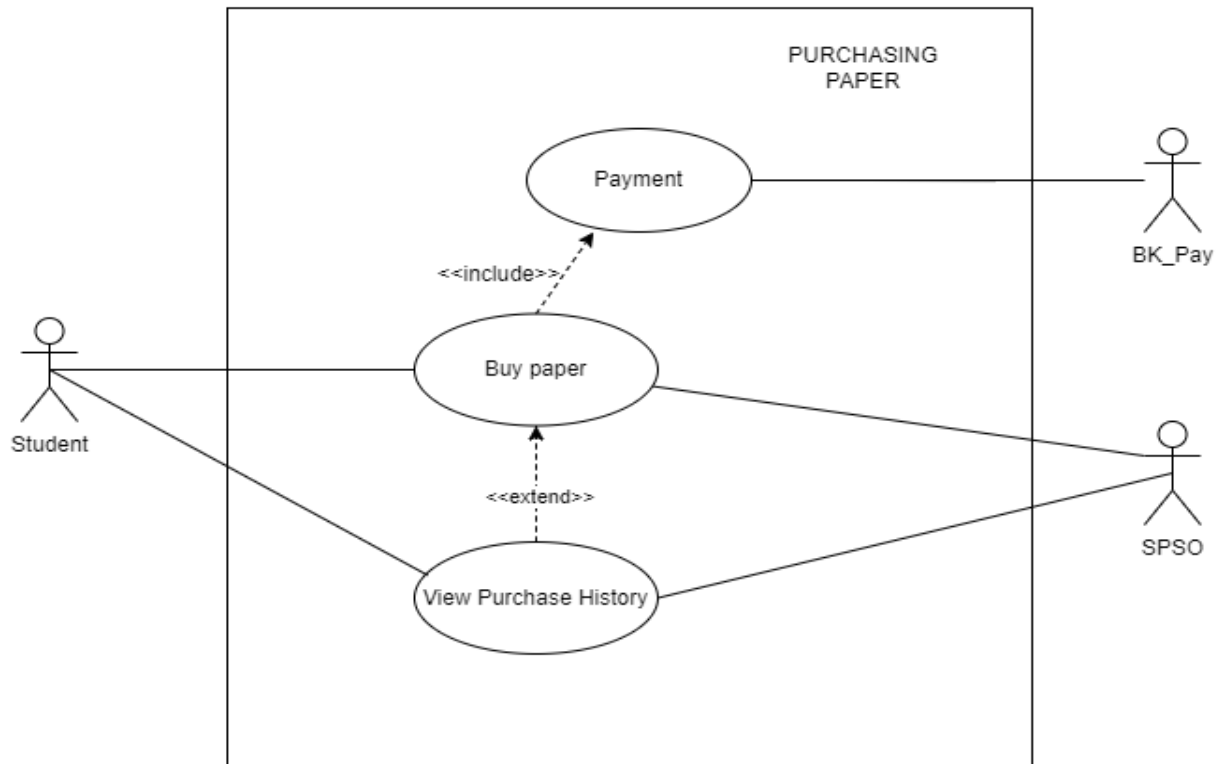
<b>Description</b>	SPSO adds/removes printers to/from the system
<b>Trigger</b>	Click on the button "Add Printer"
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. Logged in as SPSO account successfully</li> <li>2. SPSO click on the button "Manage Printer"</li> </ol>
<b>Postcondition</b>	Add/remove printers without errors
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The system displays "Add/Remove Printer" page.</li> <li>2. SPSO selects the desired printer(s) they want to add/remove.</li> <li>3. SPSO clicks confirm to apply the changes.</li> <li>4. The system confirms that the changes have been applied successfully.</li> </ol>
<b>Alternative Flow</b>	<p>At step 3: If "Cancel" is selected instead of selecting the changes.</p> <ol style="list-style-type: none"> <li>3.1 The system returns to step 1 (the add/remove printer page).</li> </ol>
<b>Exception Flow</b>	<p>At step 3: If the system encounters an error.</p> <ol style="list-style-type: none"> <li>3.1. The system displays the reasons for the error.</li> <li>3.2. The system prompts for a fix or adjustment to resolve issue.</li> <li>3.3. The system returns to step 3 to attempt the change again.</li> </ol>

<b>Name</b>	Disabling/Enabling Printer
<b>ID</b>	

<b>Actor</b>	SPSO
<b>Description</b>	SPSO disables/enables printer(s) in the system
<b>Trigger</b>	Click on the button "Add Printer"
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. Logged in as SPSO account successfully</li> <li>2. SPSO click on the button "Manage Printer"</li> </ol>
<b>Postcondition</b>	Printers disabled/enabled without errors
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The system displays "Enable/Disable Printer" page.</li> <li>2. SPSO selects the desired printer(s) they want to enable/disable.</li> <li>3. SPSO clicks confirm to apply the changes.</li> <li>4. The system confirms that the changes have been applied successfull</li> </ol>
<b>Alternative Flow</b>	<p>At step 3: If "Cancel" is selected instead of selecting the changes.</p> <ol style="list-style-type: none"> <li>3.1 The system returns to step 1 (the enable/disable printer page).</li> </ol>
<b>Exception Flow</b>	<p>At step 3: If the system encounters an error.</p> <ol style="list-style-type: none"> <li>3.1. The system displays the reasons for the error.</li> <li>3.2. The system prompts for a fix or adjustment to resolve issue.</li> <li>3.3. The system returns to step 3 to attempt the change again..</li> </ol>

<b>Name</b> <b>ID</b>	Viewing Printer's Info
<b>Actor</b>	SPSO
<b>Description</b>	SPSO checks the printer's information
<b>Trigger</b>	Click on the button "View Printer Info"
<b>Precondition</b>	1. Logged in as SPSO account successfully 2. SPSO click on the button "Manage Printer"
<b>Postcondition</b>	Printer's information is viewed without errors
<b>Main Flow</b>	1. The system displays "Printer's Info" page. 2. SPSO selects the desired printer they want to view info. 3. SPSO selects "Exit" to exit view
<b>Alternative Flow</b>	At step 2: If a printer encounters an error 2.1. Get fixing requirement from printer manufacturer
<b>Exception Flow</b>	No

## 2.4 Use case Diagram for Purchasing Paper



<b>Name ID</b>	Buy paper
<b>Actor</b>	Student
<b>Secondary Actor</b>	SPSO
<b>Description</b>	The Student enters the amount and size of additional pages to

	purchase.
<b>Trigger</b>	The Student clicks on the button “Buy paper”.
<b>Precondition</b>	User logs in as a Student successfully.
<b>Postcondition</b>	The Student finishes entering the amount and size of additional pages to purchase without errors.
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The system displays the “Buy paper” page.</li> <li>2. The Student chooses the page size to buy.</li> <li>3. The Student chooses the amount of chosen page size to buy.</li> <li>4. The Student clicks “Confirm” to confirm the chosen page size and amount.</li> <li>5. The system confirms that the page size and amount have been chosen successfully and goes to the payment option.</li> </ol>
<b>Alternative Flow</b>	<p>At step 4: If "Cancel" is selected instead of confirming the action.</p> <ol style="list-style-type: none"> <li>4.1. The system returns to step 1 (the “Buy paper” page).</li> </ol>

<b>Name ID</b>	Payment
<b>Primary Actor</b>	Student
<b>Secondary Actor</b>	BK_Pay
<b>Description</b>	The Student pays for additional pages.

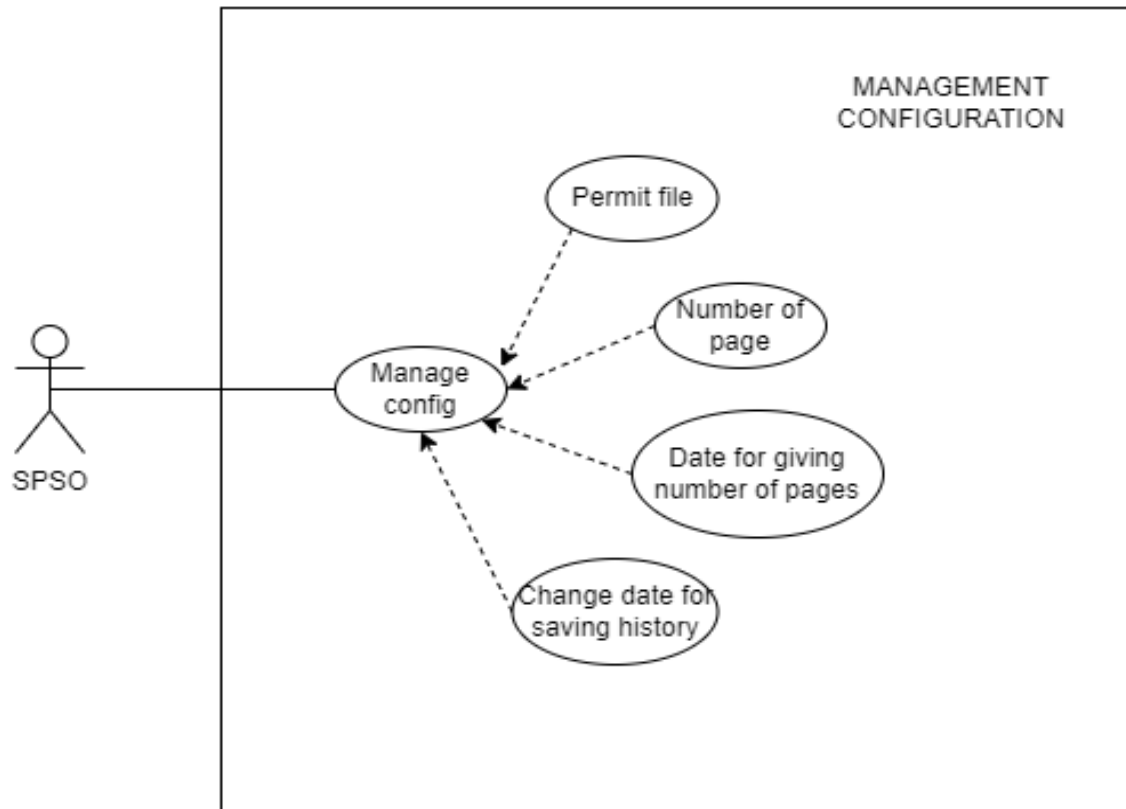
<b>Trigger</b>	The Student clicks on the button “Confirm” in the “Buy paper” page.
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. User logs in as a Student successfully.</li> <li>2. The Student clicks on the button “Buy paper”.</li> </ol>
<b>Postcondition</b>	The Student finishes purchasing additional papers without errors.
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The system directs the Student to the BK pay page.</li> <li>2. The Student logs into the BK pay page to complete the purchase</li> <li>3. The system confirms the pages have been successfully paid for.</li> <li>4. The system updates the Student’s account page balance.</li> </ol>

<b>Name ID</b>	View purchase history
<b>Primary Actor</b>	Student, SPSO
<b>Description</b>	The Student or SPSO views the Student’s purchase history.
<b>Trigger</b>	The Student or SPSO clicks on the button “View purchase history”.
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. User logs in as a Student successfully.</li> </ol> or 1.1. User logs in as SPSO successfully.
<b>Postcondition</b>	The Student or SPSO finishes viewing the Student’s purchase history without errors.

<b>Main Flow</b>	<ol style="list-style-type: none"><li>1. The system displays the Student's purchase history with purchase date and amount.</li><li>2. The Student or SPSO can scroll through the purchase history to view it.</li><li>3. The Student or SPSO clicks "Finish" after done viewing.</li><li>4. The system closes down the purchase history.</li></ol>
<b>Alternative Flow</b>	<p>At step 2: The Student or SPSO can search for a specific purchase date instead of scrolling through the purchase history</p> <ol style="list-style-type: none"><li>2.1. The Student or SPSO enters a specific date to view.</li><li>2.2. The system displays the purchase history of the entered date.</li></ol>
<b>Exception Flow</b>	<p>At step 2.1: If the date entered is invalid:</p> <ol style="list-style-type: none"><li>2.1.1. The system displays: "Invalid date"</li><li>2.1.2. The system returns to step 2.1 for the user to enter again.</li></ol>



## 2.5 Use case Diagram for Management Configuration



<b>Name</b>	Manage configuration
<b>ID</b>	
<b>Actor</b>	SPSO

<b>Description</b>	SPSO manage the configuration of the system
<b>Trigger</b>	Click on the button “Change Configuration”
<b>Precondition</b>	Log in as SPSO account successfully
<b>Postcondition</b>	Change the configuration without error
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The system displays “the Configuration Change” page.</li> <li>2. SPSO selects the configuration they want to change.</li> <li>3. SPSO clicks confirm to apply the changes.</li> <li>4. The system confirms that the changes have been applied successfully.</li> </ol>
<b>Alternative Flow</b>	<p>At step 3: If "Cancel" is selected instead of confirming the change.</p> <ol style="list-style-type: none"> <li>3.1. The system returns to step 1 (the configuration change page).</li> </ol>
<b>Exception Flow</b>	<p>At step 4: If the system encounters an error.</p> <ol style="list-style-type: none"> <li>4.1. The system displays the reason for the error.</li> <li>4.2. The system prompts for a fix or adjustment to resolve the issue.</li> <li>4.3. The system returns to step 3 for the SPSO to attempt the change again.</li> </ol>

<b>Name</b> <b>ID</b>	Permit file types
<b>Actor</b>	SPSO
<b>Description</b>	SPSO permit the types of files that students can upload for printing in the system
<b>Trigger</b>	Click on the button “Permit file types”
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. Log in as SPSO account successfully</li> <li>2. SPSO click on the button “Change Configuration”</li> </ol>
<b>Postcondition</b>	Permit file types without error
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The system displays “the file type permission” page.</li> <li>2. SPSO selects the file type they accept.</li> <li>3. SPSO clicks confirm to apply the changes.</li> <li>4. The system confirms that the changes have been applied successfully.</li> </ol>
<b>Alternative Flow</b>	<p>At step 2: If the SPSO wants to change the memory limit for those files:</p> <ol style="list-style-type: none"> <li>2.1. The SPSO types in the maximum memory size they accept for file uploads.</li> </ol> <p>At step 3: If the SPSO selects "Cancel" instead of confirming the</p>

	<p>change:</p> <p>3.1. The system returns to step 1 (file type permission page).</p>
<b>Exception Flow</b>	<p>At step 4: If the memory limit for the file type exceeds 1GB:</p> <p>4.1. The system displays an error: "The memory exceeds the limit."</p> <p>4.2. The system prompts the SPSO to input a valid memory size.</p> <p>4.3. The system returns to step 2.1 for the SPSO to correct the memory size.</p>

<b>Name</b>	Number of default pages
<b>ID</b>	
<b>Actor</b>	SPSO
<b>Description</b>	SPSO permit the number of default that student can receive per month
<b>Trigger</b>	Click on the button “Change default page”
<b>Precondition</b>	<p>1. Log in as SPSO account successfully</p> <p>2. SPSO click on the button “Change Configuration”</p>
<b>Postcondition</b>	Change default pages without error

<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The system displays “Change default pages” page.</li> <li>2. SPSO selects the number of default pages they accept.</li> <li>3. SPSO clicks confirm to apply the changes.</li> <li>4. The system confirms that the changes have been applied successfully.</li> </ol>
<b>Alternative Flow</b>	<p>At step 3: If the SPSO selects "Cancel" instead of confirming the change:</p> <ol style="list-style-type: none"> <li>3.1. The system returns to step 1 (“Change default pages” page).</li> </ol>
<b>Exception Flow</b>	No

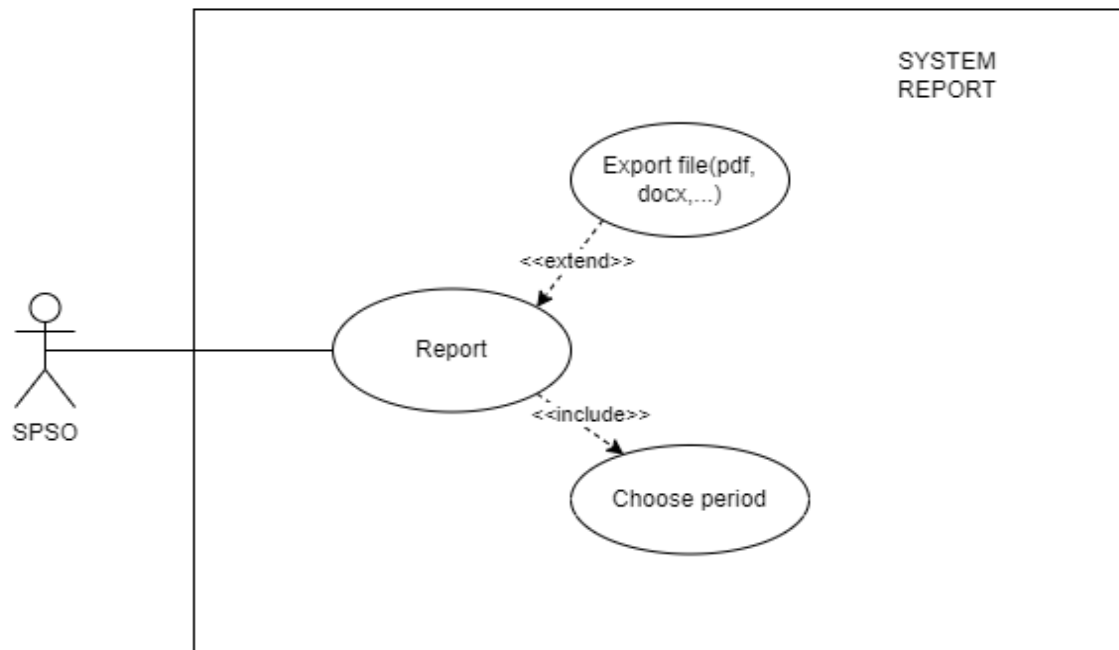
<b>Name</b> <b>ID</b>	Change date for saving history
<b>Actor</b>	SPSO
<b>Description</b>	SPSO change date for saving history
<b>Trigger</b>	Click on the button “Change saving date”
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. Log in as SPSO account successfully</li> <li>2. SPSO click on the button “Change Configuration”</li> </ol>

<b>Postcondition</b>	Change saving date
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The system displays “chang saving date” page.</li> <li>2. SPSO selects the month.</li> <li>3. SPSO clicks confirm to apply the changes.</li> <li>4. The system confirms that the changes have been applied successfully.</li> </ol>
<b>Alternative Flow</b>	<p>At step 3: If the SPSO selects "Cancel" instead of confirming the change:</p> <ol style="list-style-type: none"> <li>3.1. The system returns to step 1 (“chang saving date” page).</li> </ol>
<b>Exception Flow</b>	<p>At step 4: If choosing month exceed 12:</p> <ol style="list-style-type: none"> <li>4.1. The system displays an error: "The month is invalid."</li> <li>4.3. The system returns to step 2.1 for the SPSO to choose again.</li> </ol>

<b>Name</b>	Date for giving number of pages
<b>ID</b>	
<b>Actor</b>	SPSO
<b>Description</b>	SPSO change date for giving number of pages

<b>Trigger</b>	Click on the button “Change giving date”
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. Log in as SPSO account successfully</li> <li>2. SPSO click on the button “Change Configuration”</li> </ol>
<b>Postcondition</b>	Change saving date successfully if the last changing is exceeding 12 months.
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The system displays “chang giving date” page.</li> <li>2. SPSO selects the month.</li> <li>3. SPSO selects the appropriate with that month.</li> <li>4. SPSO clicks confirm to apply the changes.</li> <li>5. The system confirms that the changes have been applied successfully.</li> </ol>
<b>Alternative Flow</b>	<p>At step 3: If the SPSO selects "Cancel" instead of confirming the change:</p> <ol style="list-style-type: none"> <li>3.1. The system returns to step 1 (“chang giving date” page.).</li> </ol>
<b>Exception Flow</b>	<p>At step 4: If the last change was made less than 12 months ago:</p> <ol style="list-style-type: none"> <li>4.1. The system displays an error: "You cannot change the date within 12 months of the last change."</li> <li>4.2. The system prompts the SPSO to wait until 12 months have passed since the last change.</li> </ol>

## 2.6 Use case Diagram for System Report



<b>Name ID</b>	Report
<b>Actor</b>	SPSO
<b>Description</b>	SPSO manages the report of the printing system.
<b>Trigger</b>	SPSO clicks on the button “Report”.
<b>Precondition</b>	User logs in as SPSO successfully.



<b>Postcondition</b>	SPSO finishes managing the report without errors.
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The system displays the “Report management” page.</li> <li>2. SPSO choose the action they want to perform (period to view or export report file)</li> <li>3. SPSO clicks “Done” after finishing managing the report.</li> <li>4. The system closes down the “Report management” page.</li> </ol>
<b>Alternative Flow</b>	No
<b>Exception Flow</b>	No

<b>Name ID</b>	Choose period
<b>Actor</b>	SPSO
<b>Description</b>	SPSO chooses the report period of the printing system to view.
<b>Trigger</b>	SPSO clicks on the button “Choose period”.
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. User logs in as SPSO successfully.</li> <li>2. SPSO clicks on the button “Report”.</li> </ol>
<b>Postcondition</b>	SPSO finishes viewing the report period without errors.
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. SPSO enters the month and year period of the report to view.</li> <li>2. The system displays the report of the chosen month and year.</li> </ol>

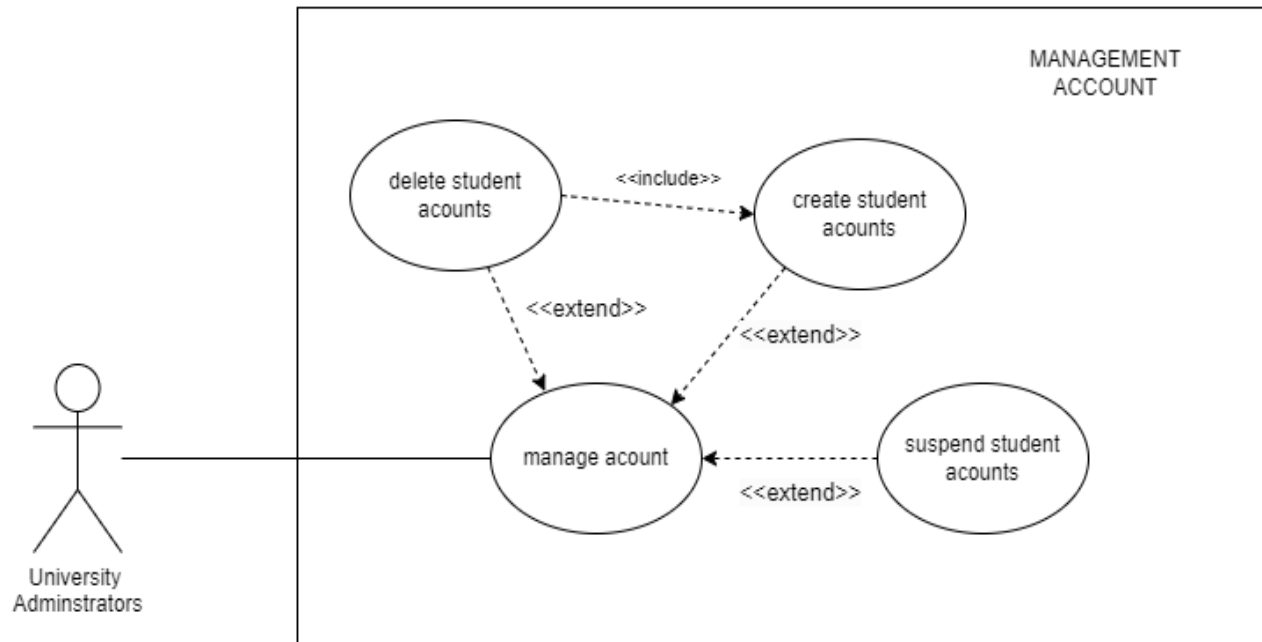
	<p>3. SPSO clicks “Done” after finishing viewing.</p> <p>4. The system closes down the report.</p>
<b>Alternative Flow</b>	
<b>Exception Flow</b>	<p>At step 1: If the chosen month and year is invalid or after the previous month:</p> <p>1.1 The system displays: “Invalid time period” or “The time period cannot be after the previous month”.</p> <p>1.2 The system returns to step 1 for SPSO to enter again.</p>

<b>Name ID</b>	Export file
<b>Actor</b>	SPSO
<b>Description</b>	SPSO chooses the file type to export the report.
<b>Trigger</b>	SPSO clicks on the button “Export”.
<b>Precondition</b>	<p>1. User logs in as SPSO successfully.</p> <p>2. SPSO clicks on the button “Report”.</p>
<b>Postcondition</b>	The report file is exported in the chosen file type for the SPSO.
<b>Main Flow</b>	<p>1. SPSO chooses the desired file type for the report.</p> <p>2. SPSO clicks “Confirm” to export the report in the chosen file type.</p>



	3. The system confirms the file type to export. 4. The system exports the report in the chosen file type for SPSO.
<b>Alternative Flow</b>	No
<b>Exception Flow</b>	No

## 2.7 Use case Diagram for Management Account



<b>Name ID</b>	Manage account
<b>Actor</b>	University Administrators
<b>Description</b>	University Administrators manage the accounts of all students.
<b>Trigger</b>	University Administrators click on the button “Manage accounts”.
<b>Precondition</b>	Users log in as University Admins successfully.

<b>Postcondition</b>	University Administrators finish managing student accounts without errors.
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The system displays the “Account Management” page.</li> <li>2. University Admins choose the action they want to perform (create, delete or suspend an account).</li> <li>3. University Admins click “Confirm” to carry out the action.</li> <li>4. The system confirms that the action has been carried out successfully.</li> </ol>
<b>Alternative Flow</b>	<p>At step 3: If "Cancel" is selected instead of confirming the action.</p> <ol style="list-style-type: none"> <li>3.1. The system returns to step 1 (the “Account Management” page).</li> </ol>
<b>Exception Flow</b>	<p>At step 4: If the system encounters an error.</p> <ol style="list-style-type: none"> <li>4.1. The system displays the reason for the error.</li> <li>4.2. The system returns to step 3 for the University Admins to attempt the action again.</li> </ol>

<b>Name ID</b>	Create student account
<b>Actor</b>	University Administrators
<b>Description</b>	University Administrators create a student account.
<b>Trigger</b>	University Administrators click on the button “Create student account”.
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. Users log in as University Admins successfully.</li> <li>2. University Admins click on the button “Manage accounts”.</li> </ol>

<b>Postcondition</b>	University Administrators finish creating a student account without errors.
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The system displays the “Account Creation” page.</li> <li>2. University Admins fill in the necessary information to create an account (Username, password).</li> <li>3. University Admins click “Confirm” to carry out the action.</li> <li>4. The system confirms that the action has been carried out successfully.</li> </ol>
<b>Alternative Flow</b>	<p>At step 3: If "Cancel" is selected instead of confirming the action.</p> <ol style="list-style-type: none"> <li>3.1. The system returns to step 1 (the “Account Creation” page).</li> </ol>
<b>Exception Flow</b>	<p>At step 4: If the username and password contains invalid characters:</p> <ol style="list-style-type: none"> <li>4.1. The system displays: “Invalid characters used in username or password”</li> <li>4.3. The system returns to step 1 for the University Admins to attempt the action again.</li> </ol>

<b>Name ID</b>	Delete student account
<b>Actor</b>	University Administrators
<b>Description</b>	University Administrators delete a student account.
<b>Trigger</b>	University Administrators click on the button “Delete student account”.
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. Users log in as University Admins successfully.</li> </ol>

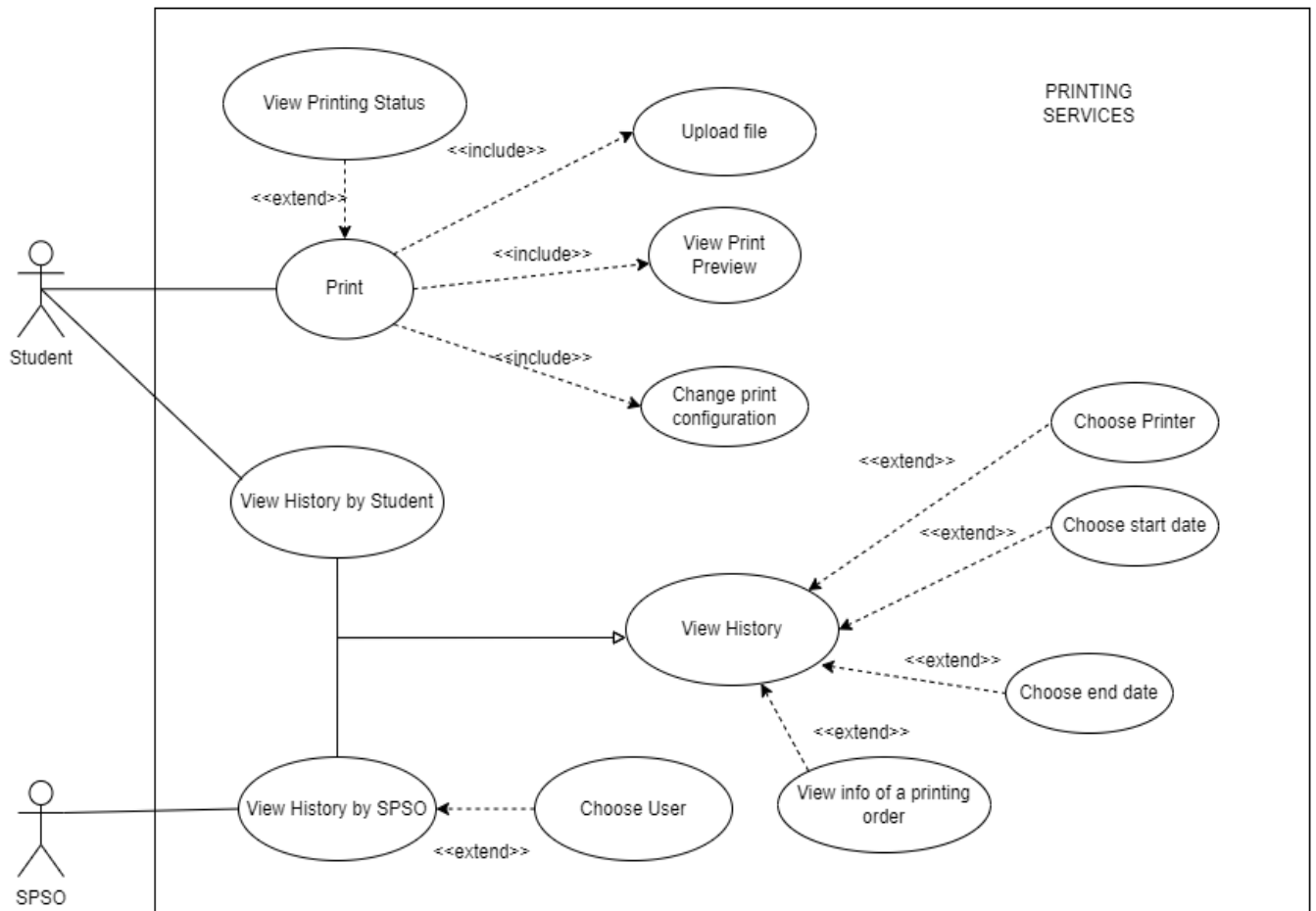
	2. University Admins click on the button “Manage accounts”.
<b>Postcondition</b>	University Administrators finish deleting a student account without errors.
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The system displays the “Account Deletion” page.</li> <li>2. University Admins choose a student account to delete</li> <li>3. University Admins click “Confirm” to carry out the action.</li> <li>4. University Admins click “Yes” when the system displays “Are you sure you want to delete this account?”</li> <li>5. The system confirms that the action has been carried out successfully.</li> </ol>
<b>Alternative Flow</b>	<p>At step 3: If "Cancel" is selected instead of confirming the action.</p> <p>3.1. The system returns to step 1 (the “Account Deletion” page).</p> <p>At step 4: If "No" is selected instead of “Yes”..</p> <p>4.1. The system returns to step 1 (the “Account Deletion” page).</p>
<b>Exception Flow</b>	No

<b>Name ID</b>	Suspend student account
<b>Actor</b>	University Administrators
<b>Description</b>	University Administrators suspend a student account.
<b>Trigger</b>	University Administrators Click on the button “Suspend student account”.

<b>Precondition</b>	<ol style="list-style-type: none"><li>1. Users log in as University Admins successfully.</li><li>2. University Admins click on the button “Manage accounts”.</li></ol>
<b>Postcondition</b>	University Administrators finish suspending a student account without errors.
<b>Main Flow</b>	<ol style="list-style-type: none"><li>1. The system displays the “Account suspension” page.</li><li>2. University Admins choose a student account to suspend</li><li>3. University Admins enter the reason for suspension</li><li>4. University Admins click “Confirm” to carry out the action.</li><li>5. The system confirms that the action has been carried out successfully.</li></ol>
<b>Alternative Flow</b>	<p>At step 4: If "Cancel" is selected instead of confirming the action.</p> <p>4.1. The system returns to step 1 (the “Account Deletion” page).</p>
<b>Exception Flow</b>	No



## 2.8 Use case Diagram for Printing Service



<b>Name</b> <b>ID</b>	Print
<b>Actor</b>	Student
<b>Description</b>	Student prints document
<b>Trigger</b>	Click on the button “Print file”
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. Log in as Student account successfully</li> <li>2. Upload file</li> <li>3. Have enough page for printing</li> <li>4. Choose their configuration (choose printer, ...)</li> </ol>
<b>Postcondition</b>	Print file successfully
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. Student clicks on the button “Print file”.</li> <li>2. The system displays “Printing...”.</li> <li>3. The system confirm “Print successfully”.</li> </ol>
<b>Alternative Flow</b>	<p>At step 1: If student choose “View Printing Status”</p> <ol style="list-style-type: none"> <li>1a.1. The system display “Printing Status”.</li> <li>1a.2. Return the “Printing” page.</li> </ol>

<b>Exception Flow</b>	<p>At step 1: If the printer have any problems:</p> <p>1b.1. The system display “Failure”.</p> <p>1b.2. Return to “Printing page”</p> <p>1b.3. Send inform to SPSO.</p>
-----------------------	---

<b>Name</b>	Upload file
<b>ID</b>	
<b>Actor</b>	Student
<b>Description</b>	Student uploads file for printing
<b>Trigger</b>	Click on the button “Printing”
<b>Precondition</b>	<p>1. Log in as Student account successfully</p> <p>2. Display “Printing” page</p>
<b>Postcondition</b>	Upload file for printing successfully
<b>Main Flow</b>	<p>1. Student uploads file</p> <p>2. Student clicks “Print”.</p>
<b>Alternative Flow</b>	No

<b>Exception Flow</b>	<p>At step 2: If the file is not permitted.</p> <p>4.1. The system displays the permitted file.</p> <p>4.2. Return step 1.</p>
-----------------------	--

<b>Name</b>	View printing review
<b>ID</b>	
<b>Actor</b>	Student
<b>Description</b>	Student see printing review
<b>Trigger</b>	Click on the button “Printing”
<b>Precondition</b>	<p>1. Log in as Student account successfully</p> <p>2. Display “Printing” page</p>
<b>Postcondition</b>	Student can see printing review
<b>Main Flow</b>	<p>1. Student clicks “Print”.</p> <p>2. The system display printing review.</p> <p>3. Student confirms.</p> <p>4. Start printing.</p>

<b>Alternative Flow</b>	At step 3: If Student does not confirm  3.1. Return “Printing” page
<b>Exception Flow</b>	No

<b>Name</b> <b>ID</b>	Change print configuration
<b>Actor</b>	Student
<b>Description</b>	Student can change the configuration for printing
<b>Trigger</b>	Click on the button “Printing”
<b>Precondition</b>	1. Log in as Student account successfully 2. Display “Printing” page
<b>Postcondition</b>	Print document by user’s configuration
<b>Main Flow</b>	1. The system display configuration(area,...) 2. Student confirms. 3. Student clicks “Print”.

<b>Alternative Flow</b>	At step 2: If student clicks “cancel”  2.1. Return “Printing” page
<b>Exception Flow</b>	At step 2: If the printer in that area is not ready(out of paper,...)  2.1. Return step 1.

<b>Name ID</b>	Viewing History
<b>Actor</b>	SPSO
<b>Description</b>	Check the printer(s)’ printing history by SPSO
<b>Trigger</b>	SPSO click on the button “View History” to see the information on the system.
<b>Precondition</b>	1. User has successfully logged in to the system 2. User selects “View History”.
<b>Postcondition</b>	The printing history is viewed successfully
<b>Main Flow</b>	1. The system displays “the Printing History” page. 2. The system display and SPSO user can view the printing history and filters the information: <ul style="list-style-type: none"> <li>• Select user: Choosing which user name of student to view</li> <li>• Select printer: Select the printer to view</li> <li>• Select start date: Select the start date of historical statistics. If the user do not select the start date, all previous time will be retrieved.</li> </ul>

	<ul style="list-style-type: none"> <li>Select end date: Select the end date of historical statistics.</li> <li>If you do not select the end date, the current time will be retrieved.</li> </ul>
<b>Alternative Flow</b>	<p>At step 2, SPSO wants to filter orders based on user name, printer, processing time</p> <p>2.1. SPSO proceeds to select the desired attributes</p> <p>2.2. The system displays orders based on the selected information</p>
<b>Exception Flow</b>	<p>At step 1, the system failed to retrieve data</p> <p>1.1. The system displays an error message to SPSO</p> <p>At step 2, there is no print order in the system</p> <p>2.1. The system displays a message that there is no print order</p>

<b>Name ID</b>	Viewing User's Personal Printing History
<b>Actor</b>	Student
<b>Description</b>	User check personal printing history
<b>Trigger</b>	Click on the button "View History" to see the information.
<b>Precondition</b>	<p>1. User has successfully logged in to the system</p> <p>2. User selects "View History"</p>
<b>Postcondition</b>	The printing history is viewed successfully

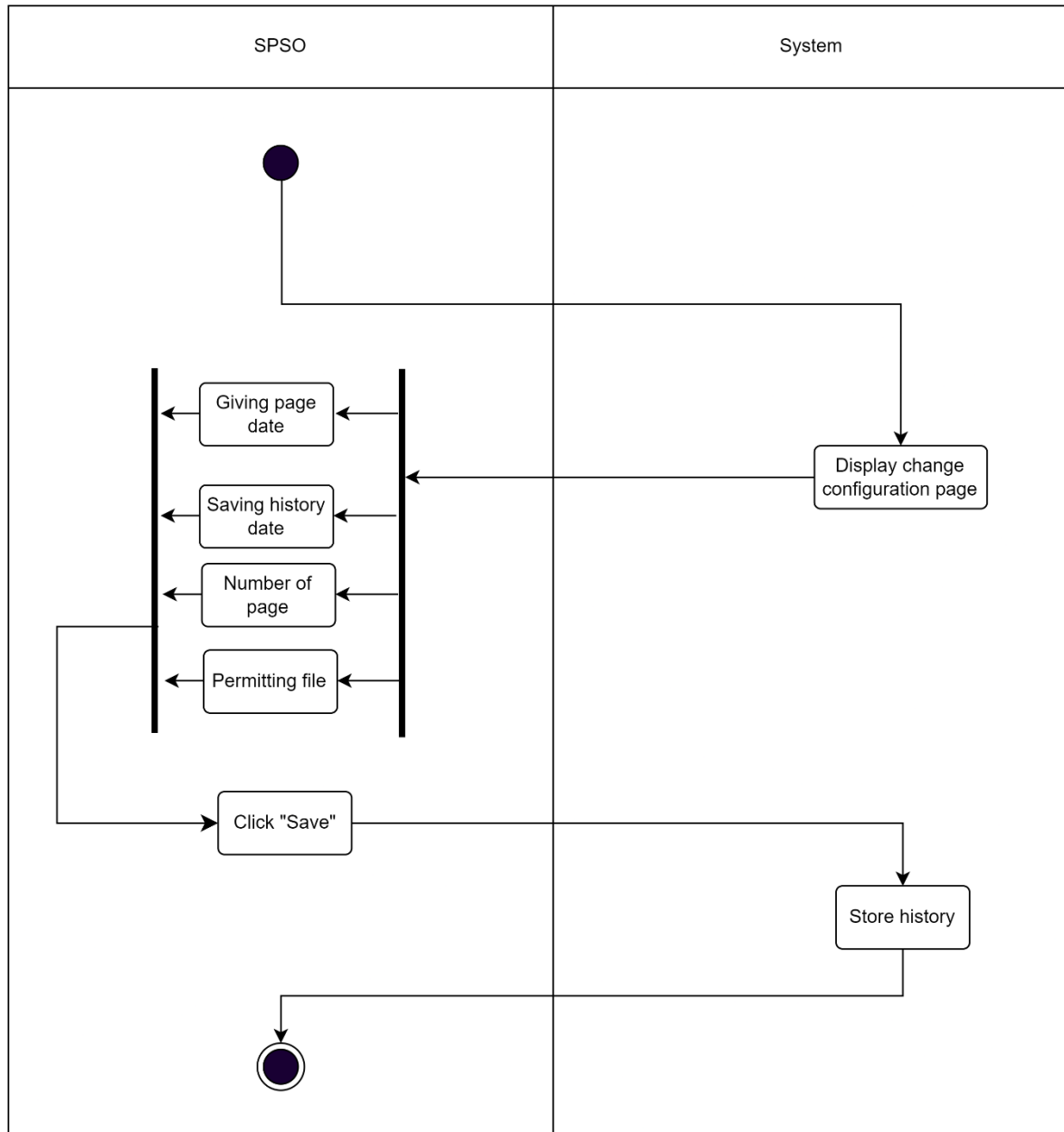
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The system retrieves printing info</li> <li>2. The system shows a interface to user and sort out information such as: <ul style="list-style-type: none"> <li>• Select printer: choose the desired printer</li> <li>• Select start date: choose the desired starting date of the printing record. If none is selected, it will retrieve all the previous records.</li> <li>• Select end date: choose the desired date that the printing record ends. If none is selected, it will retrieve all the records up to the current date starting from the starting date.</li> </ul> </li> </ol>
<b>Alternative Flow</b>	<p>At step 2, if user want to sort records based on printers, time executed</p> <ol style="list-style-type: none"> <li>2.1. User selects desired attributes</li> <li>2.2. The system shows information based on the selected attributes.</li> </ol>
<b>Exception Flow</b>	<p>At step 1, the system fails to retrieve data</p> <ol style="list-style-type: none"> <li>1.1 The system notifies the error for the user</li> </ol> <p>At step 2, there are no records in the system</p> <ol style="list-style-type: none"> <li>2.1. The systems notifies that there are no records.</li> </ol>



## 3. Task 2: System modelling

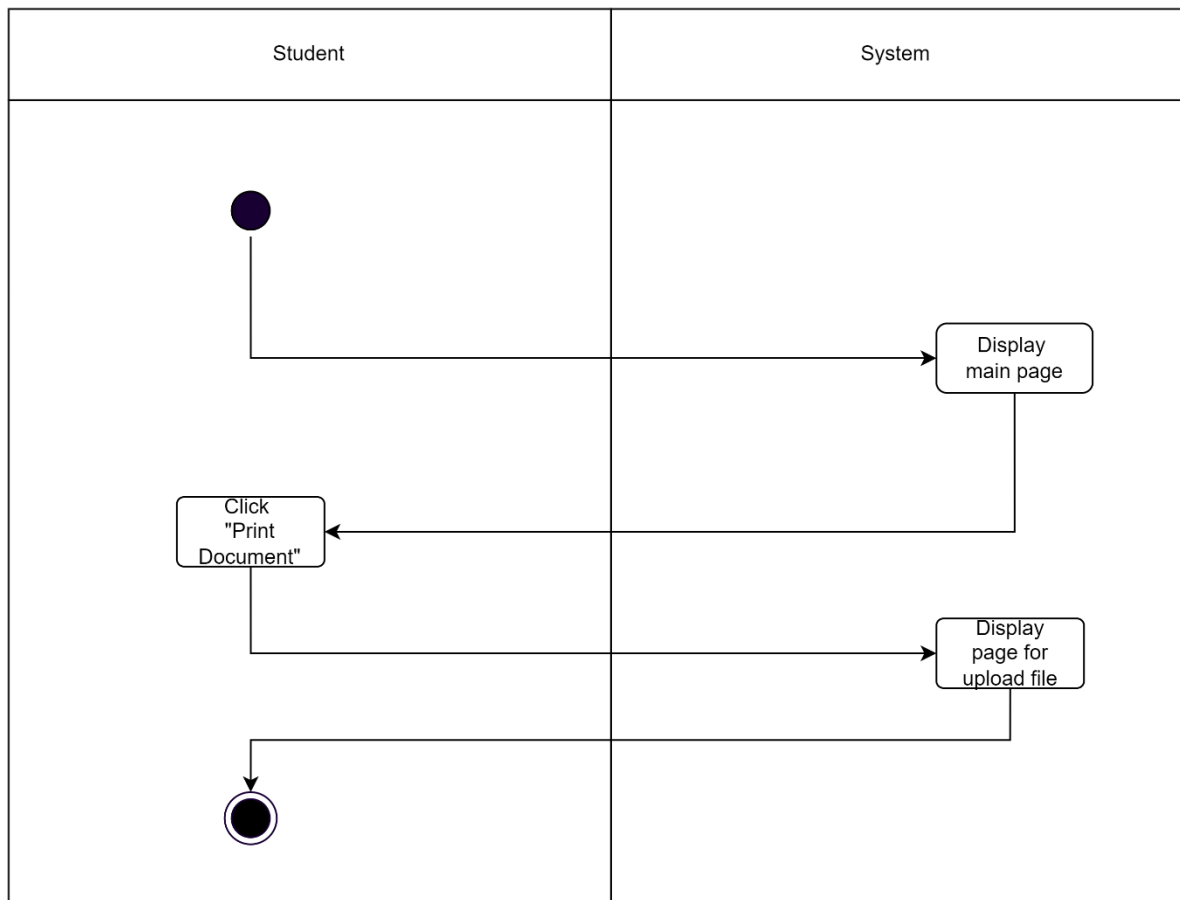
### 3.1 Activity Diagram

#### 3.1.1 Change configuration for SPSO



Link Draw.io of the diagram: [here](#)

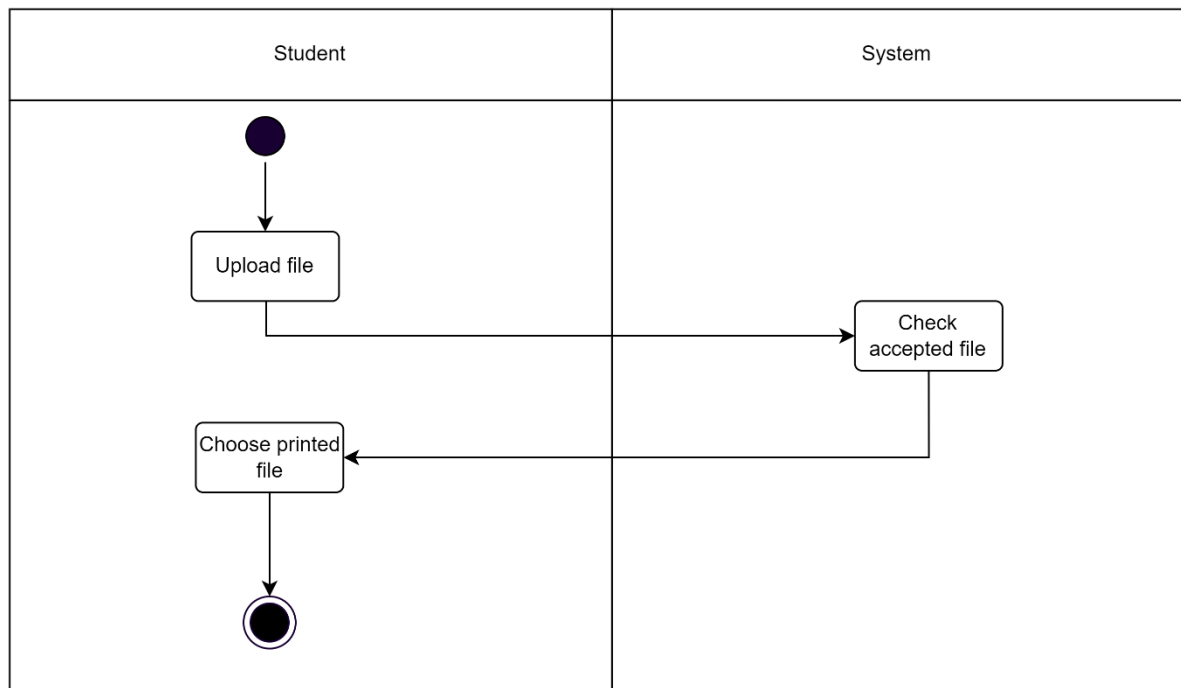
### 3.1.2 Display Uploading Files Page



Link Draw.io of the diagram: [here](#)

After successful log in, to print documents, users need to click on "In Tài Liệu" displayed on the main page screen. The system then proceeds to display the upload documents page.

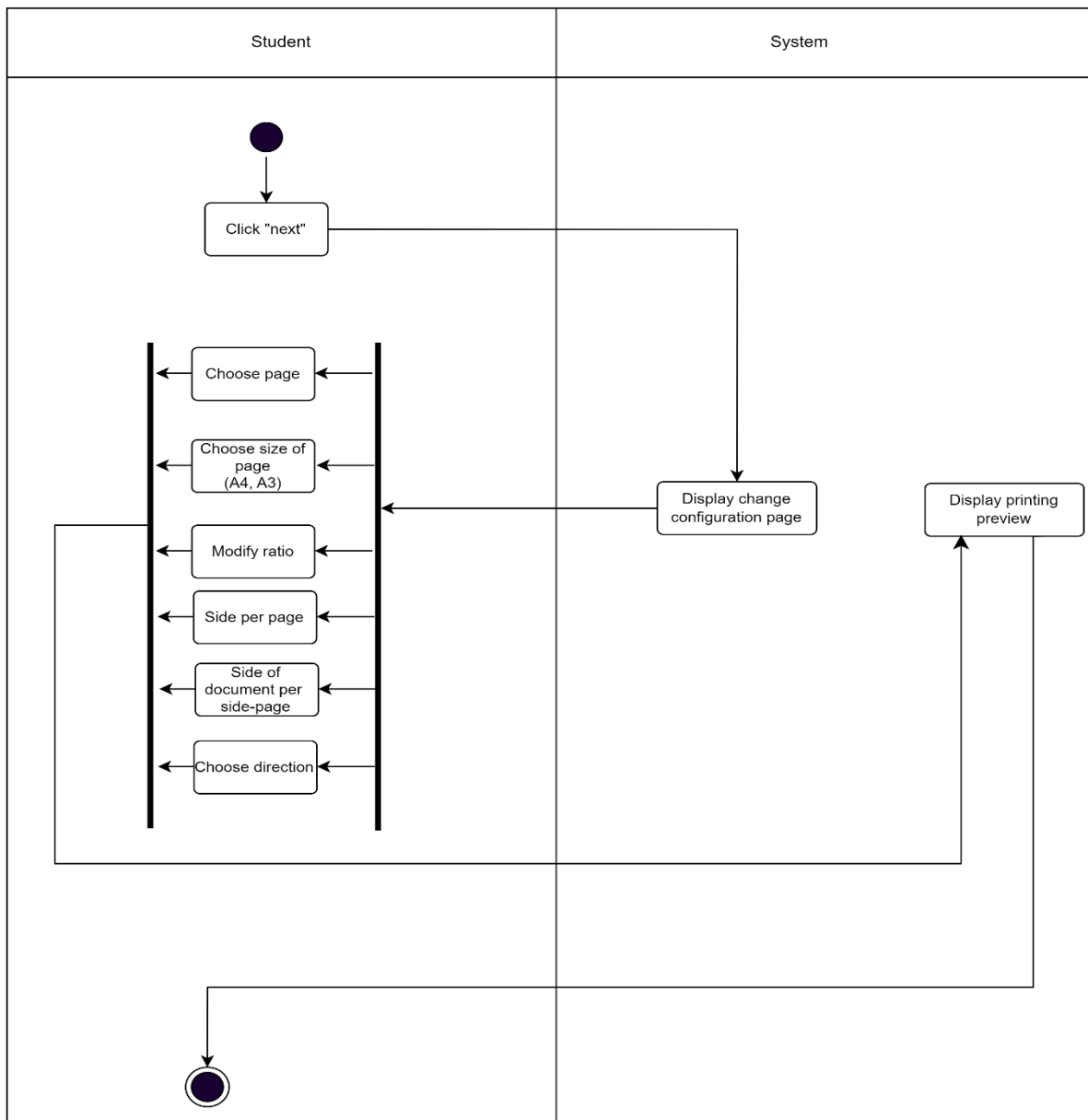
### 3.1.3 Uploading File for Student



Link Draw.io of the diagram: [here](#)

Students upload the files into the system. With each file is uploaded, the System will check the file type and return result. With the file has “Hoàn tất”, that file is acceptable. With another result as “Lỗi kích thước”, “Lỗi định dạng” will be blur. Students can choose one and just one file which has “Hoàn tất” to print.

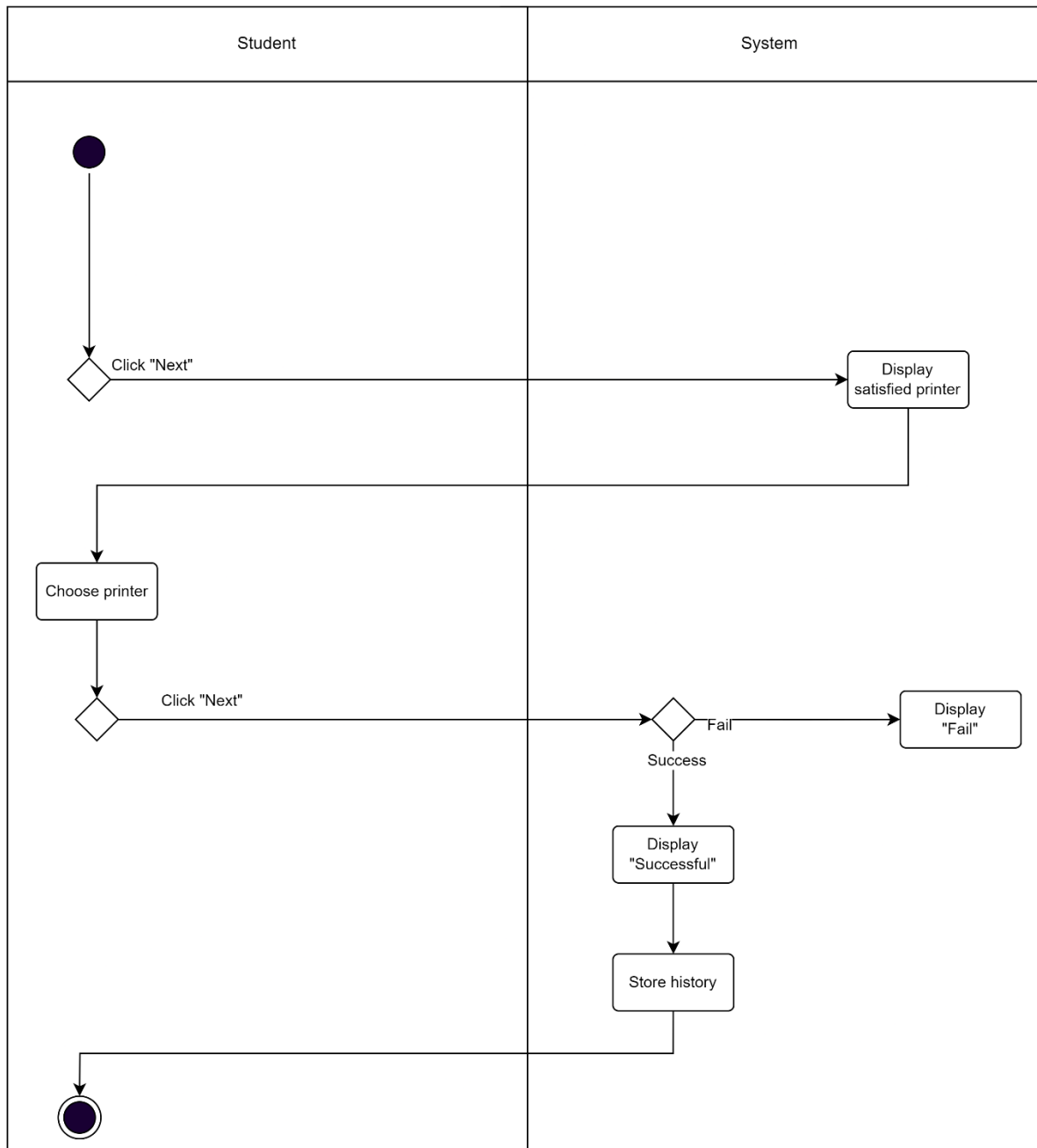
### 3.1.4 Printing Configuration for Student



Link Draw.io of the diagram: [here](#)

Clicking “Next” will display the “Configuration Page” for the student who can then choose the page sequence to print, choose the size for the pages, modify the ratio, choose the side per page, choose side of document per side-page and choose page direction. After that the printing preview will be displayed.

### 3.1.5 Printing Service for Students

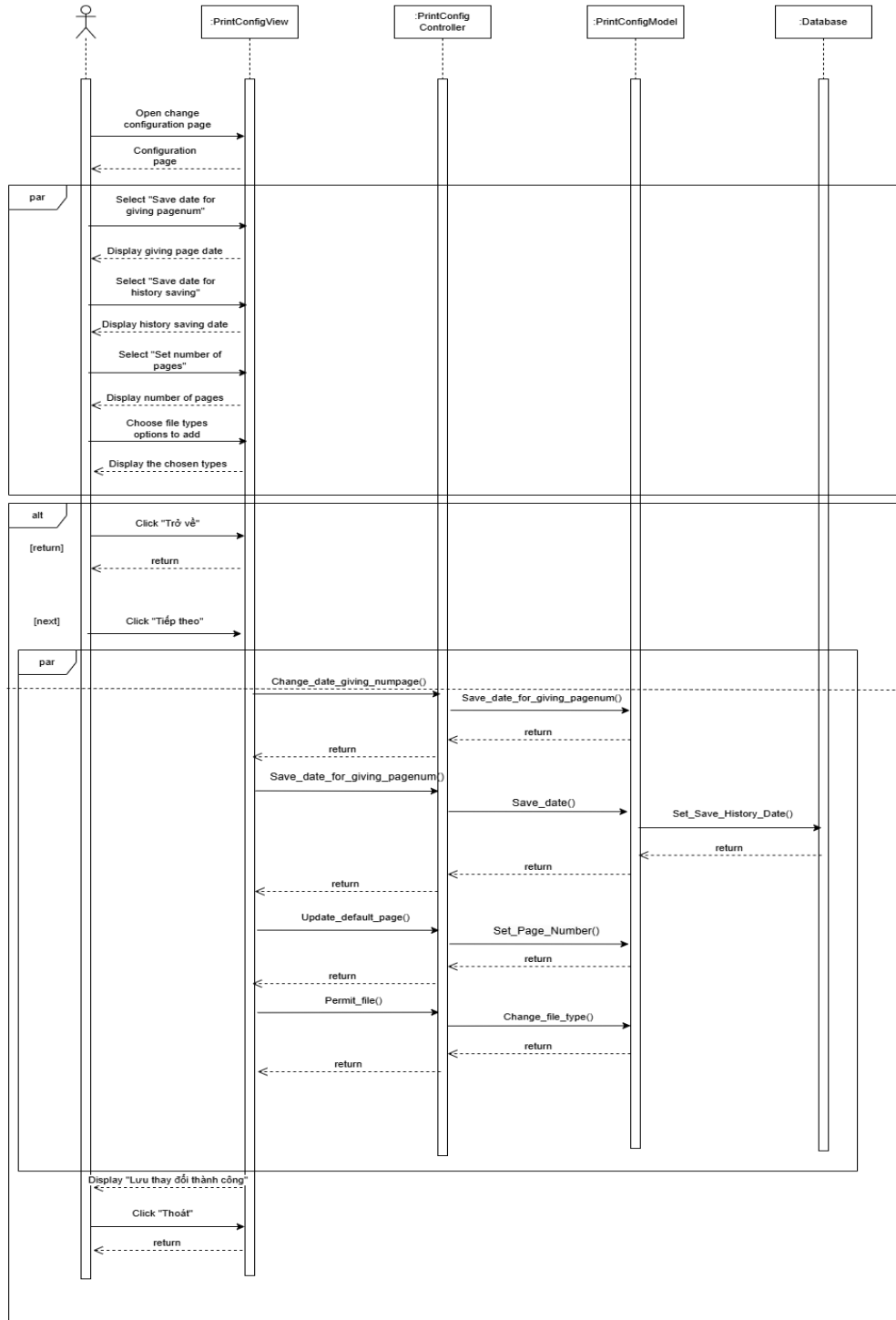


Link Draw.io of the diagram: [here](#)

Clicking “Next” will display the satisfied printers that can be used and the student will be able to choose one of those printers to print their uploaded file. If the file is printed successfully the “Successful” page will be displayed and the printing will be stored into the database. Otherwise, if the printer fail to print the “Fail” page will be displayed.

## 3.2 Sequence Diagram

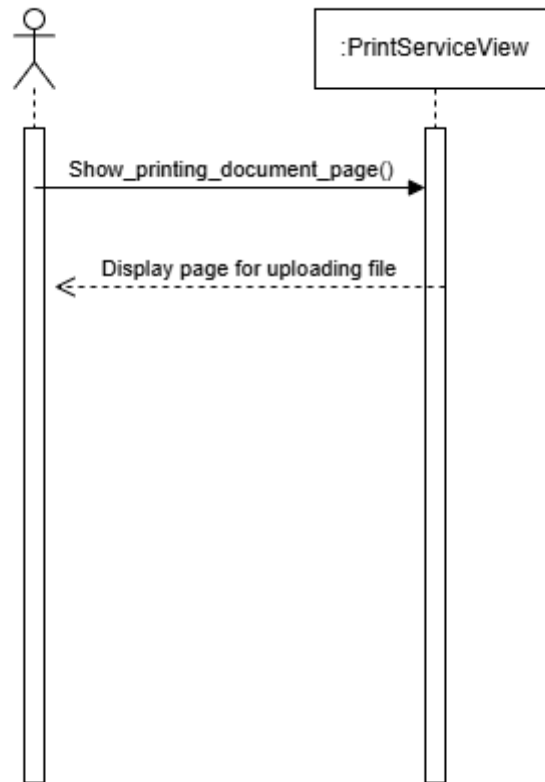
### 3.2.1 Change configuration for SPSO



Link Draw.io of the diagram: [here](#)

After the student has clicked “Open change configuration page”, the PrintConfigView will display the “Configuration page”. The student will then be able to carry out the following actions in parallel with each other: Select “Save date for giving pagenum” - the PrintConfigView will display giving page date, Select “Save date for history saving” - the PrintConfigView then will display history saving date, Select “Set number of pages” - the PrintConfigView will display number of pages, click the checkbox multiple select file types to add will display the chosen file types allowed for students. Then if the student clicks “Trở về”, it will return to the previous page, else if student clicks “Tiếp theo”, it will do some parallel following actions: the PrintConfigView calls the Change\_date\_giving\_numpage() from PrintConfigController which will then call Save\_date\_for\_giving\_pagenum() function from PrintConfigModel which will return to PrintConfigController then return to PrintConfigView; the PrintConfigView calls the Save\_date\_for\_giving\_numpage() from PrintConfigController which will then call Save\_date() function from PrintConfigModel which will call the Set\_Save\_History\_Date() from Database and return respectively to the PrintConfigModel, PrintConfigController, PrintConfigView; the PrintConfigView calls the Update\_default\_page() from PrintConfigController which then calls the Set\_Page\_Number() function from PrintConfigModel and return respectively to the PrintConfigController, PrintConfigView; the PrintConfigView calls the Permit\_file() function from the PrintConfigController which then calls the Change\_file\_type() function from PrintConfigModel, then return to the PrintConfigController and return to PrintConfigView. After all those parallel actions, a pop up with an announce “Lưu thay đổi thành công” will appear together with a button “Thoát”. When the student clicks “Thoát” then returns.

### 3.2.2 Display Uploading Files Page

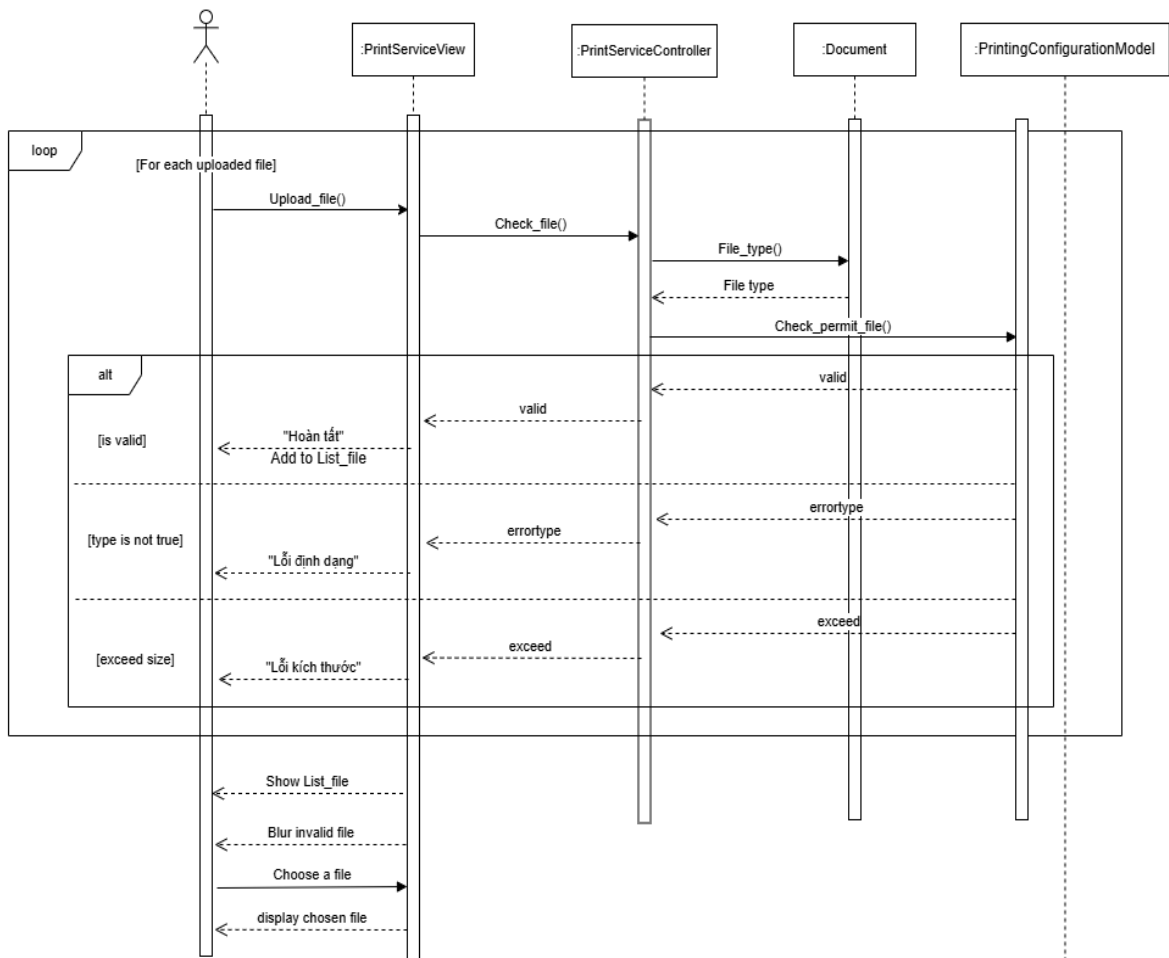


Link Draw.io of the diagram: [here](#)

After student clicks on "In Tài Liệu" displayed on the main page screen. The PrintServiceView activates the function Show\_printing\_document\_page() and display the upload documents page to User Interface.



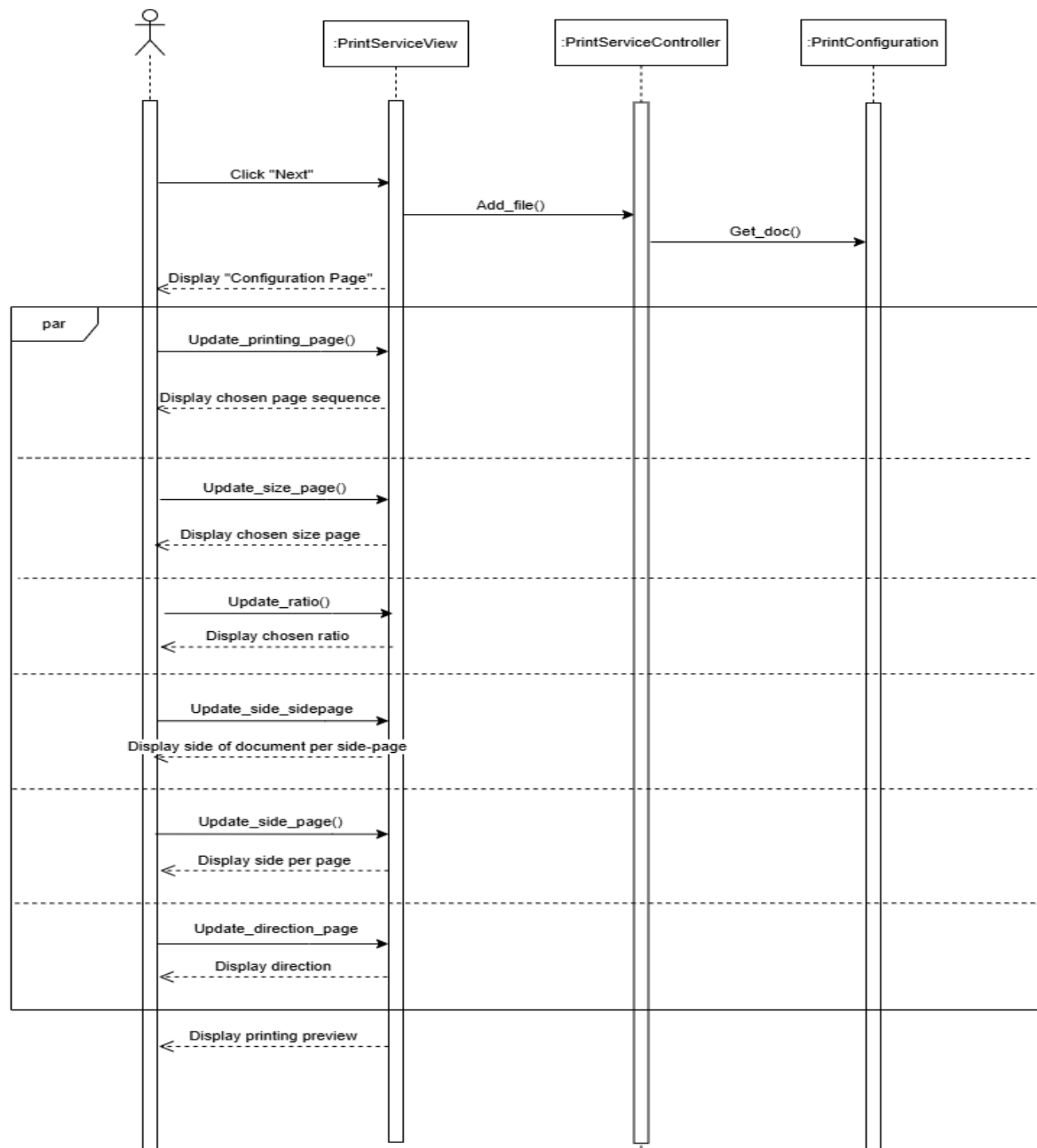
### 3.2.3 Sequence Diagrams for Uploading Files



Link Draw.io of the diagram: [here](#)

The user can upload multiple files at once. Each time a file is uploaded, PrintServiceView will call the Check\_file() function from PrintServiceController(). PrintServiceController calls File\_type() from Document to obtain the type of file. PrintServiceController then uses this type to call the Check\_permit\_file() function from PrintingConfigurationModel(). If the result returned is valid, PrintServiceView() will display the screen as “Hoàn tất” indicating that the file upload is complete. Otherwise, it will display either “Lỗi định dạng” or “Lỗi kích thước” Files without errors will be added to List\_file, while files with errors will be blurred. The user will select one of the error-free files to proceed to the next step.

### 3.2.4 Sequence Diagram for Printing Service



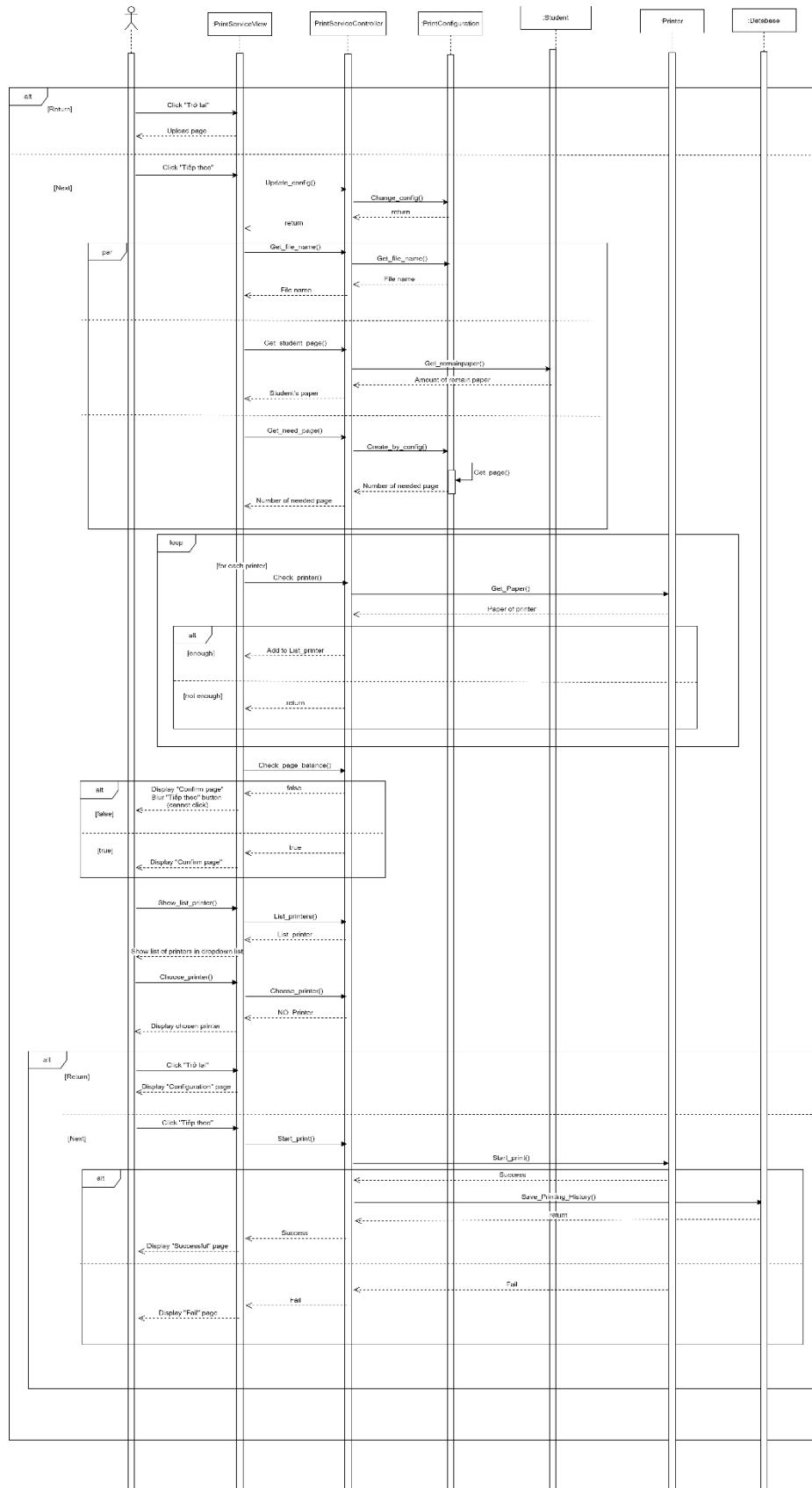
Link Draw.io of the diagram: [here](#)

After the student has clicked “Next”, `PrintServiceView` will call the `Add_file()` function from `PrintServiceController` which will then call `Get_doc()` from `PrintConfiguration` for the `PrintServiceView` to display the “Configuration page”. The student will then be able to carry out the following actions in parallel with each other: Choose page sequence to print



(Update\_printing\_page()); choose page size A3 or A4 (Update\_size\_page()); choose ratio for printed pages (Update\_ratio()), choose side per page (Update\_side\_page()); choose side of document per side-page (Update\_side\_sidepage()); and choose page direction (Update\_direction\_page()). PrintServiceView will subsequently display the chosen printing configuration for each action after they are carried out and finally display the printing preview.

### 3.2.5 Sequence Diagram for Printing Configuration for Students



Link Draw.io of the diagram: [here](#)

If the student clicks “Return” then PrintServiceView will display the “Upload page”. Otherwise, if the student clicks “Next” PrintServiceView will call Update\_config() from PrintServiceController which will then change the configuration to the chosen one by calling Change\_config from PrintConfiguration. PrintServiceView will then carry out 3 parallel: Get the file name by calling the Get\_file\_name() function from PrintServiceController which will get the file name from PrintConfiguration by calling its Get\_file\_name() function; get the remaining paper left in the student’s account by calling Get\_student\_page() from PrintServiceController which will call Get\_remainpaper() in Student to return amount of student’s remain paper; get the number of needed papers to print by calling function Get\_need\_page() in PrintServiceController which will call Create\_by\_config() in PrintConfiguration and it will return the needed number by calling its own Get\_page() function.

PrintServiceView will check for each available printer by calling the function Check\_printer() from PrintServiceController which will then call Get\_Paper() from Printer to get the number of papers left in each printer. If the number is enough to print the printer will be added to List\_printer.

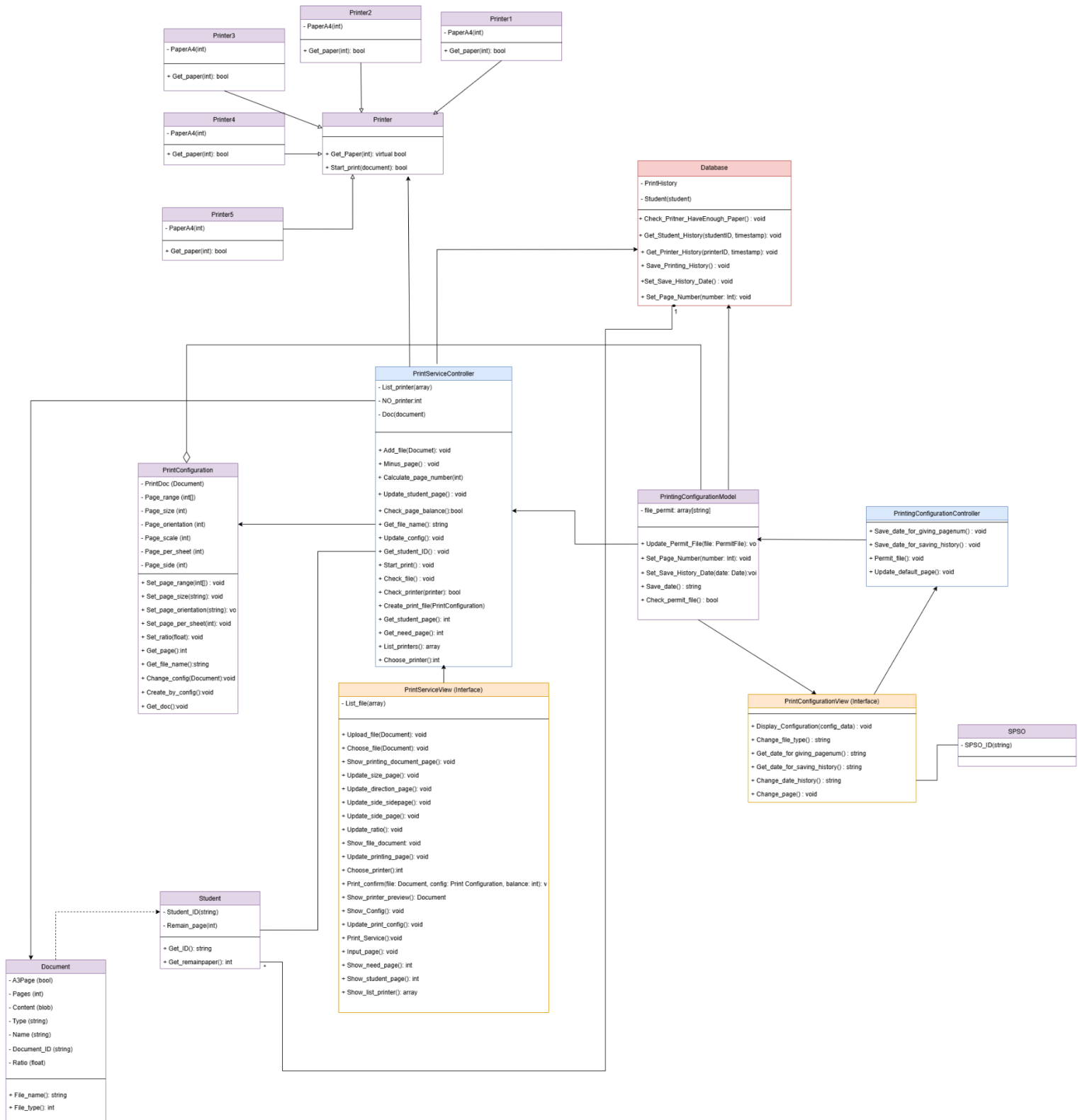
PrintServiceView will then check the page balance of the student by calling Check\_page\_balance() from PrintServiceController. If it’s not enough then PrintServiceView will display the “Confirm page” with the “Next” button blurred out which means it cannot be clicked. Otherwise, PrintServiceView will display the “Confirm page” normally. The student can then view the list of satisfied printers in a dropdown list by calling the Show\_list\_printer() function from PrintServiceView which will then get the List\_printer from PrintServiceController its List\_printers() function. The student will then choose a printer to use from this list by calling the function Choose\_printer() from PrintServiceView which will get the number of the chosen printer (No\_Printer) from PrintServiceController after calling its Choose\_printer() function. The student can then see PrintServiceView displaying the chosen printer.

Finally the student gets the option to click “Return” which PrintServiceController will then take them back to the “Configuration” page. Otherwise, the student can click “Next” to start printing. PrintServiceView will call the Start\_print function from PrintServiceController which will call another Start\_print function from Printer. If the printer successfully prints the



file PrintServiceController will save the printing history into Database by calling its function Save\_Printing\_History() and PrintServiceView will display the “Success” page. Otherwise, PrintServiceView will display the “Fail” page.

### 3.3 Class Diagram



Link Draw.io of the diagram: [here](#)

### 3.3.1 PrintServiceView()

Displays the printing interface, supports entering printing parameters, selecting a printer, and confirming printing.

#### Attribute

- List\_file : the list of the file that student upload and want to print

#### Method

- Upload\_file(Document): upload the file want to print on the website
- Choose\_file(Document): choosing what type of file want to print (doc,pdf,...)
- Show\_printing\_document\_page(): display the pages that have the information of printing service for students
- Update\_size\_page(): student enters the size of page
- Update\_direction\_page(): choose the direction of page (left, right, upside-down,...)
- Update\_side\_sidepage(): choose how many page for 2-side page printing
- Update\_side\_page(): choose how many page for 1-side page printing
- Update\_ratio(): choose the ratio of page want to print
- Show\_file\_document: display the file document on the screen with information
- Update\_printing\_page(): update the default of printing page
- Choose\_printer(): choose printer want to print
- Print\_confirm(): confirm to print after enter all the necessary information
- Show\_printer\_preview(): show the information of the printer
- Show\_Config(): show the configuration of the printer for student to choose and modify
- Update\_print\_config(): update the printing page configuration after edit the information of printing page
- Print\_Service(): print the page by request
- Input\_page(): input the pages that have to print
- Show\_need\_page(): show the page that need more to print
- Show\_student\_page(): show the default page of student has
- Show\_list\_printer(): show the list of printer working or not



### 3.3.2 PrintServiceController()

Function of entering printing parameters into the system, counting the number of papers, and selecting printers suitable for printing needs

#### Attribute

- List\_printer : the list of the printer
- NO\_printer: number of the printer in service
- Doc: document that student upload

#### Method

- Add\_file(): add the file on the website
- Minus\_Page(): delete the page printing
- Calculate\_page\_number(): calculate the page that student want to print
- Update\_student\_page() : update the student page printing
- Check\_page\_balance(): check the present page that student has to print
- Get\_file\_name(): get the name of the file printing
- Update\_config(): update the printing configuration
- Get\_student\_ID() : get the student ID
- Start\_print() : start to print the document
- Check\_file() : check the type, format of the file
- Check\_printer(printer): check the available printer to print
- Create\_print\_file(PrintConfiguration): create to print after edit the print configuration
- Get\_student\_page(): get the student page available
- Get\_need\_page(): get the page that student need more to print
- List\_printers(): list the number and available printer
- Choose\_printer(): choose the printer to print the document

### 3.3.3 Student()

User who uses the printing service

#### Attribute

- Student\_ID: the ID of the student to log in to the website
- Remain\_page: Page that student has to print the document



### **Method**

- Get\_ID(): get the ID of the student
- Get\_remainpaper(): get the remaining page that the student has

#### **3.3.4 Printer()**

Print the document on demand

### **Method**

- Get\_Paper(): get the paper to print the document
- Start\_print(): start to print the document when receive the request.

#### **3.3.5 Printer 1, 2, 3, 4, 5() - inheritance from the Printer**

Get the paper and send to the **Printer** to start print

### **Attribute**

- PaperA4: page A4 in document that need to print more

### **Method**

- Get\_paper(): get the A4 paper then send to Printer to print

#### **3.3.6 Document()**

Student uploads the document to print

### **Attribute**

- A3Page: page A3 in document that need to print
- Pages
- Content
- Type: type of the document (word, PDF,...)
- Name: the title of the document
- Document\_ID: the ID of the document
- Ratio: ratio, size of the document

### **Method**

- File\_name(): get the file name of the document to print
- File\_type(): get the type of the document

#### **3.3.7 PrintConfiguration()**

Print configuration function for student to edit and enter information to print the document

#### **Attribute**

- PrintDoc (Document): document to print
- Page\_range (int[]): the range of the document
- Page\_size (int): size of the paper
- Page\_orientation (int): orientation of paper (portrait, landscape,...)
- Page\_scale (int): scale of the paper
- Page\_per\_sheet (int): 1 page per sheet, 2 page per sheet ....
- Page\_side (int): print 1 page 1 side or 1 page 2 side

#### **Method**

- Set\_page\_range(int[]): set the range of the paper to print
- Set\_page\_size(): set the size of the paper to print
- Set\_page\_orientation(): set the orientation of paper to print
- Set\_page\_per\_sheet(): set the paper per sheet (1 paper per sheet, 3 paper per sheet,...)
- Set\_ratio(): set the ratio for paper
- Get\_page(): get the page that the student has to print
- Get\_file\_name(): get the file name of the document student want to print
- Change\_config(document): change the printing configuration by edit the information
- Create\_by\_config(): create parameters for print configuration
- Get\_doc(): get the document to print

### **3.3.8 PrintConfigurationView()**

Display the interface for printing configuration settings

#### **Method:**

- Display\_Configuration(config\_data): Displays the given configuration data for printing settings.
- Change\_file\_type(): Changes the permit file type for printing.
- Get\_date\_for\_giving\_pagenum(): Modify the date for giving page numbers.
- Get\_date\_for\_saving\_history(): Get the date for the printing history stored.



- `Change_date_history()`: Modify the date for the printing history stored.
- `Change_page()`: Changes the number of giving pages

### 3.3.9 **PrintingConfigurationModel()**

Manage and validate printing permissions, page settings, and historical records

**Attribute:**

- `file_permit (array[string])`: Stores a list of permitted file types

**Method:**

- `Update_Permit_File(file: PermitFile)`: Updates the list of permitted file types by adding or modifying a specific permit file type.
- `Set_Page_Number(number: Int)`: Changes the number of giving pages
- `Set_Save_History_Date(date: Date)`: Modify the date for the printing history stored.
- `Save_date()`: Modify the date for the printing history stored.
- `Check_permit_file()`: Checks if a file is permitted based on the list of allowed file types, returning boolean

### 3.3.10 **PrintingConfigurationController()**

Manage and control the settings and permissions related to printer configuration,

**Method:**

- `Save_date_for_giving_pagenum()`: Changes the number of giving pages
- `Save_date_for_saving_history()`: Modify the date for the printing history stored.
- `Permit_file()`: Updates the list of permitted file types
- `Update_default_page()`: Changes the number of giving pages

### 3.3.11 **SPSO()**

User who manages the configuration

**Attribute:**

- `SPSO_ID`: determine user

### 3.3.12 **Database()**

Responsible for directly accessing and retrieving data from the database



### **Attribute**

- PrintHistory: history of printing information
- Student

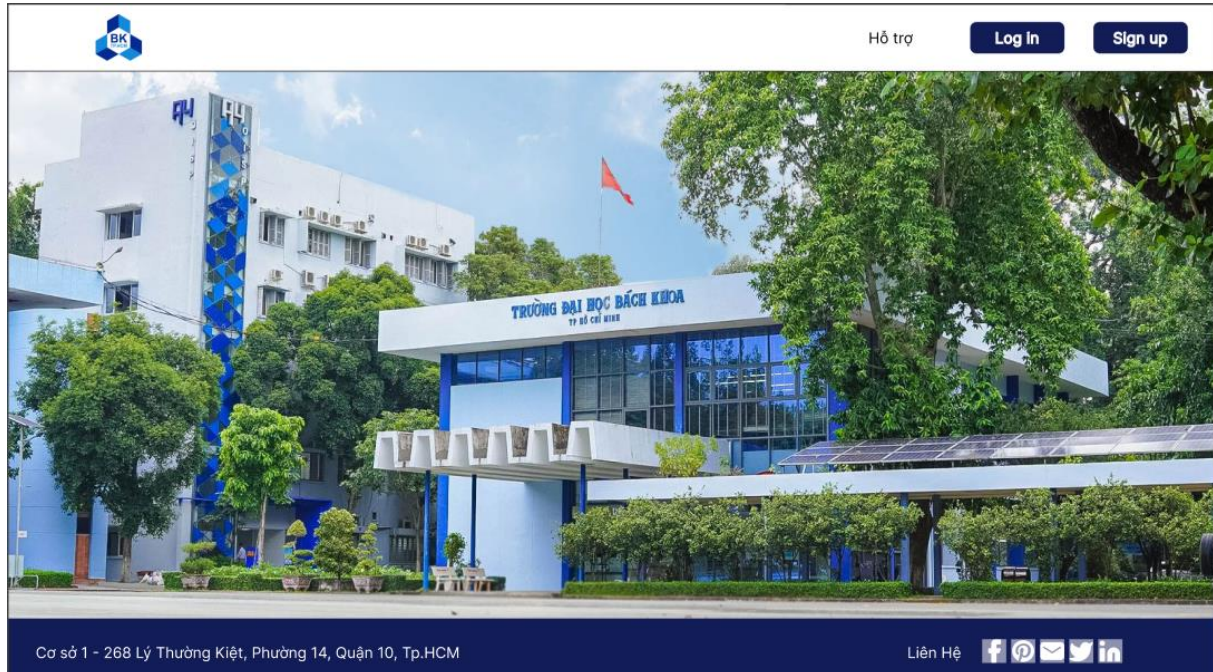
### **Method**

- Check\_Pritner\_HaveEnough\_Paper() : check if the printer has enough of paper to print or not
- Get\_Student\_History(studentID, timestamp): get the student printing information from timestamp
- Get\_Printer\_History(printerID, timestamp): get the printer printing information from timestamp
- Save\_Printing\_History() : save the printing history information
- Set\_Save\_History\_Date() : set the timestamp that use the printing service
- Set\_Page\_Number(number: Int): set the number of pages that have to print

### 3.4 User Interfaces using Figma

Link Figma: [here](#)

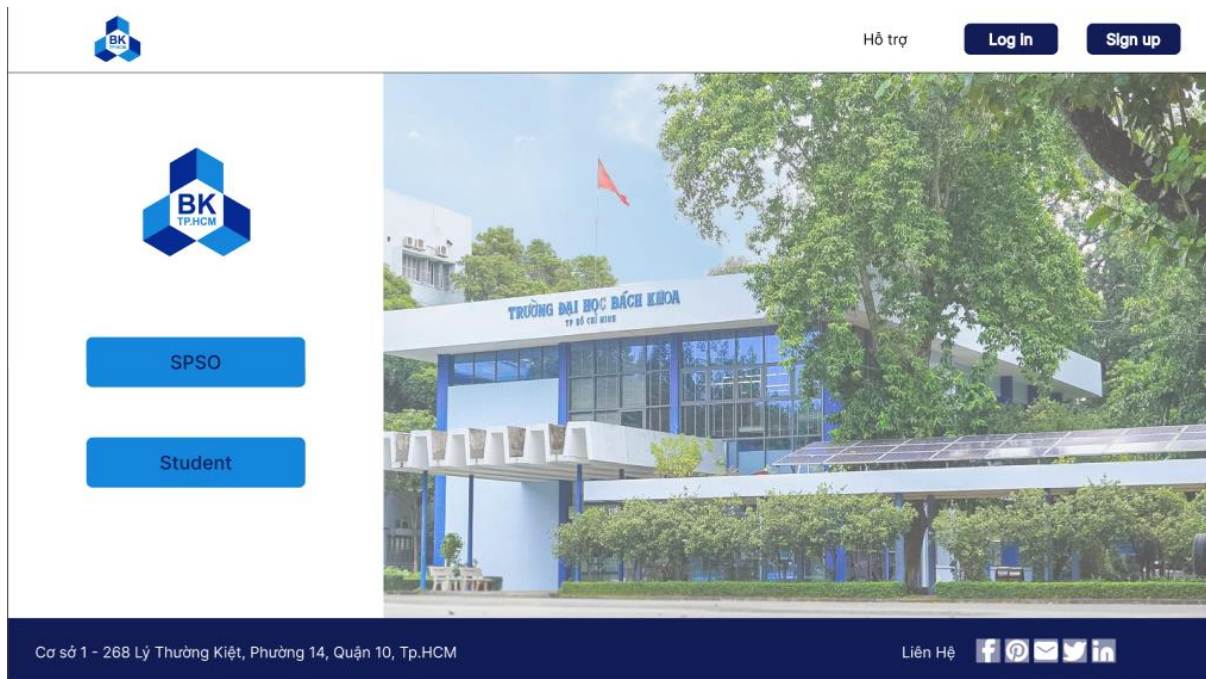
#### 3.4.1 Main screen



This is the first look of the web page which includes 3 sections: the header which then serves as a navigation bar - has 2 buttons “Log in” and “Sign Up”, the body section is HCMUT background, the footer contains the address together with contact information of HCMUT.



### 3.4.2 Pre-login screen



Before using system functionality, the user must login. Click on “Log in” button will then open a sidebar contain 2 button “SPSO” and “Student” based on the user job and domain.

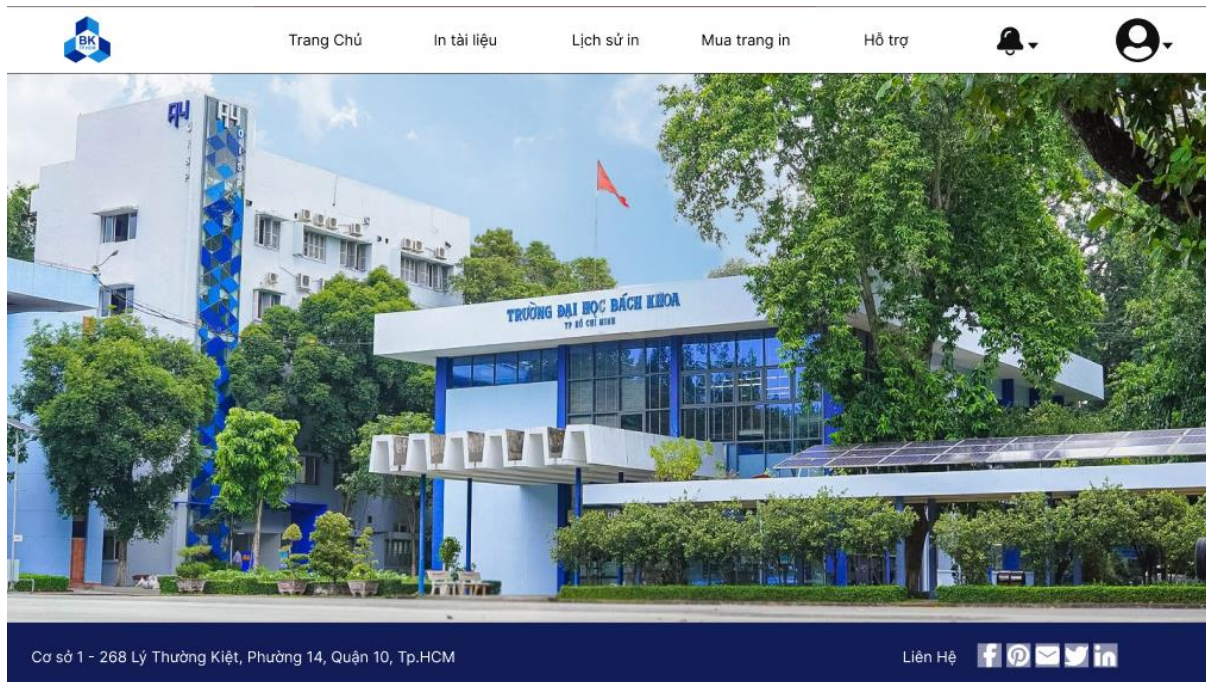
### 3.4.3 Login page



If the user is a student, click on the “Student” button, the sidebar will have new content. The student enters “Username” and “Password” and press Login to check validation

of the account. If correct, then directs the student to the next page else an announcement will occur to inform wrong account information. The “Sign up” button serves the same as the one in the navigation bar, which will then redirect users to the sign-up area.

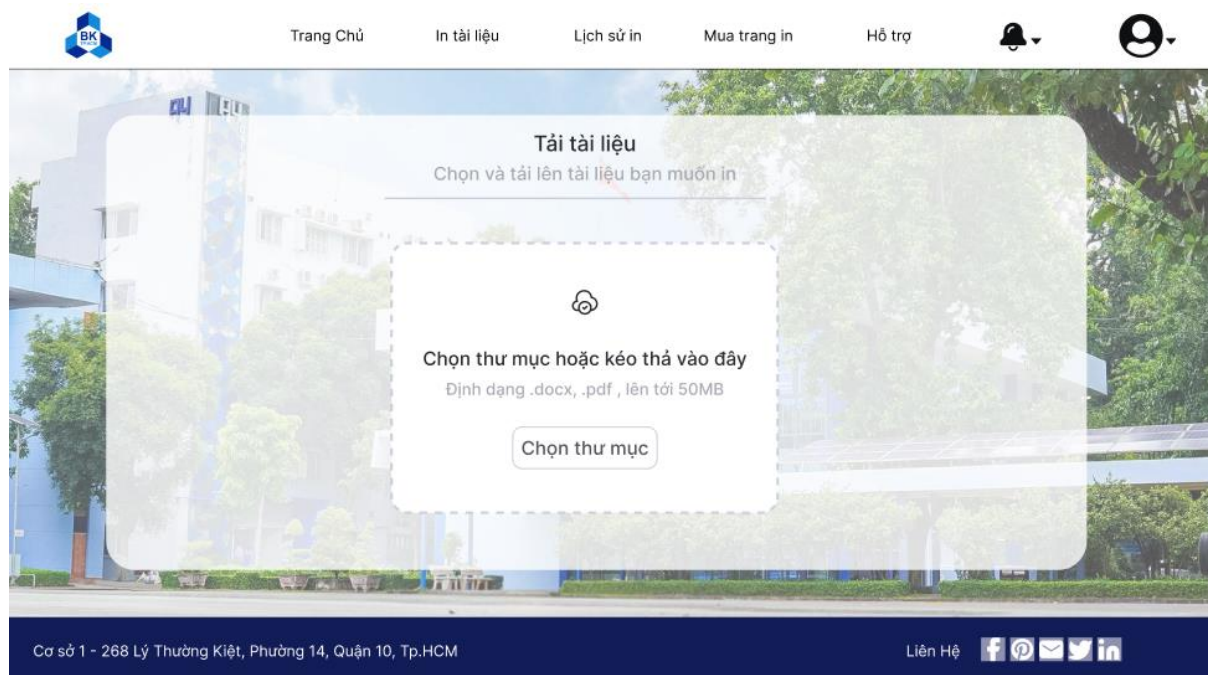
### 3.4.4 Main menu for the student



When the account is correct, the web will go to the main page for student purposes. The navigation bar will now fill with new link: “Trang chủ”, “In tài liệu”, “Lịch sử in”, “Mua trang in”, “Hỗ trợ”, a bell icon represents notifications, an avatar icon is link to user’s information.

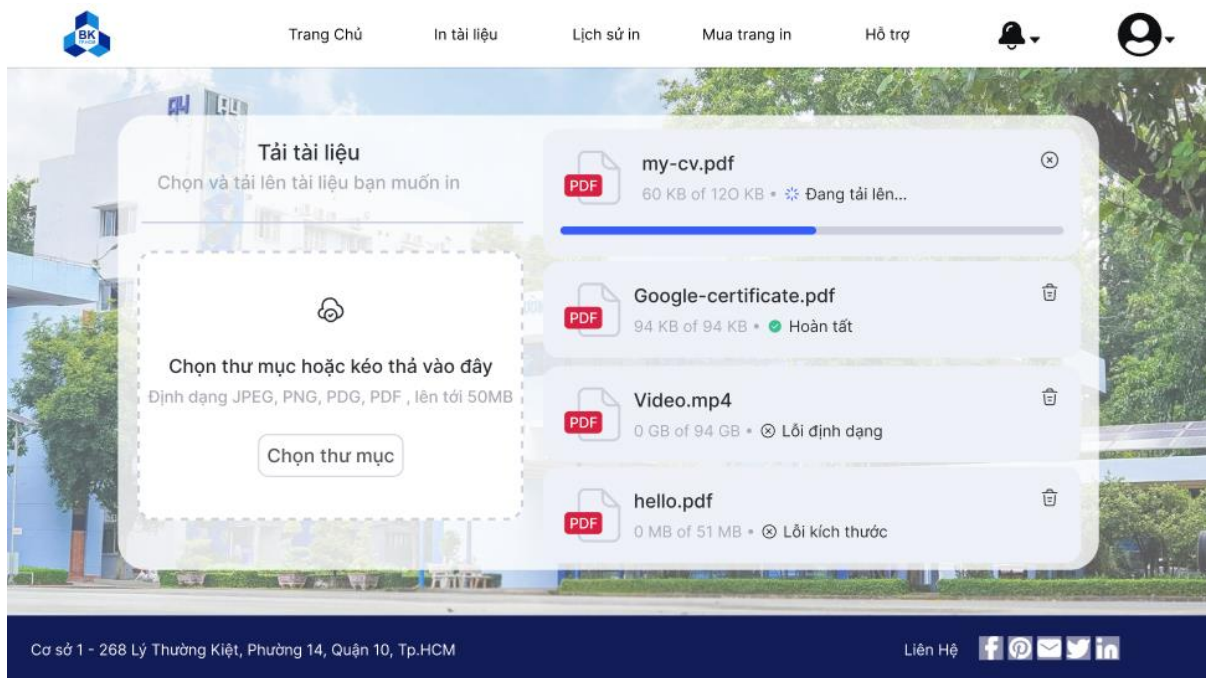


### 3.4.5 Print document page



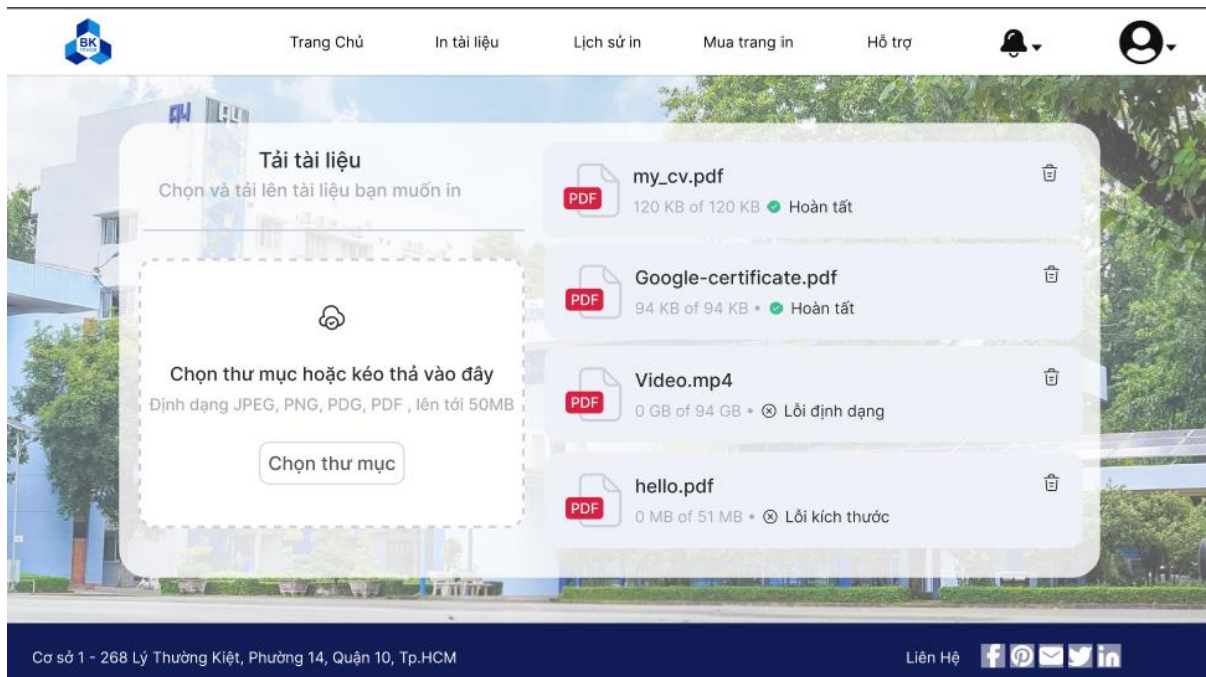
Click the “In tài liệu” will direct the student to the page for pushing the document that student wants to print. Here, students have 2 options, drag and drop file or choosing file from a local database such as computer, smartphone, Warning! the file type has already been mentioned, if it does not meet the file type or exceeds 50MB, the file will be served as an error and can’t be printed.

### 3.4.6 Uploading file page



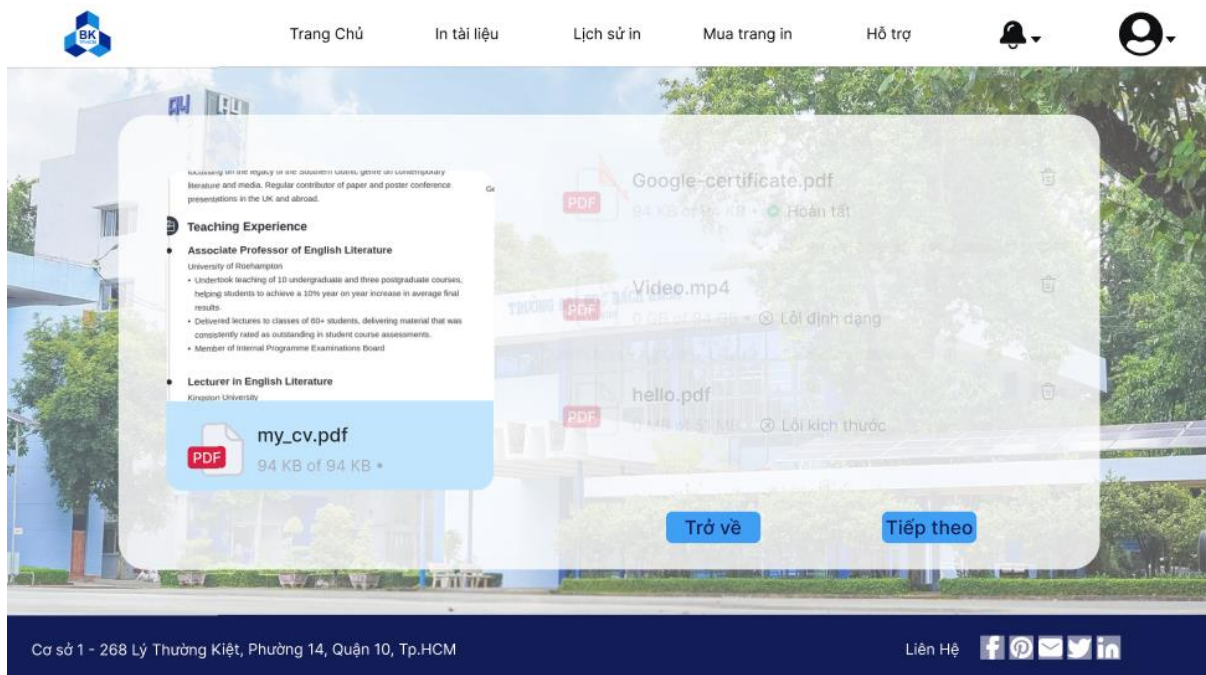
If the file meets all the requirements, it will be uploaded to the server for the following actions.

### 3.4.7 Uploaded file page



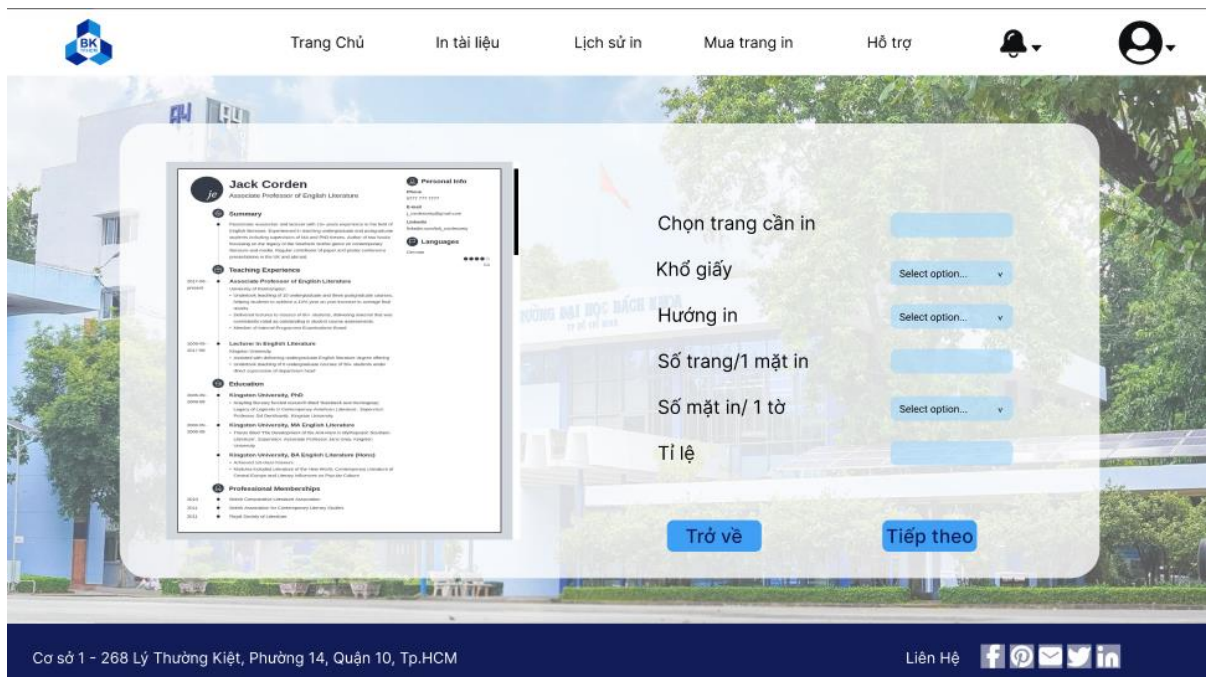
Here is an example of a file uploaded successfully.

### 3.4.8 Confirm file to go to the configuration



Then the flow will automatically move to the next action, which is to confirm the file that was uploaded. Click the button “Trở về” will redirect user to the previous interface. Click the button “Tiếp theo” to move to the next step of flow.

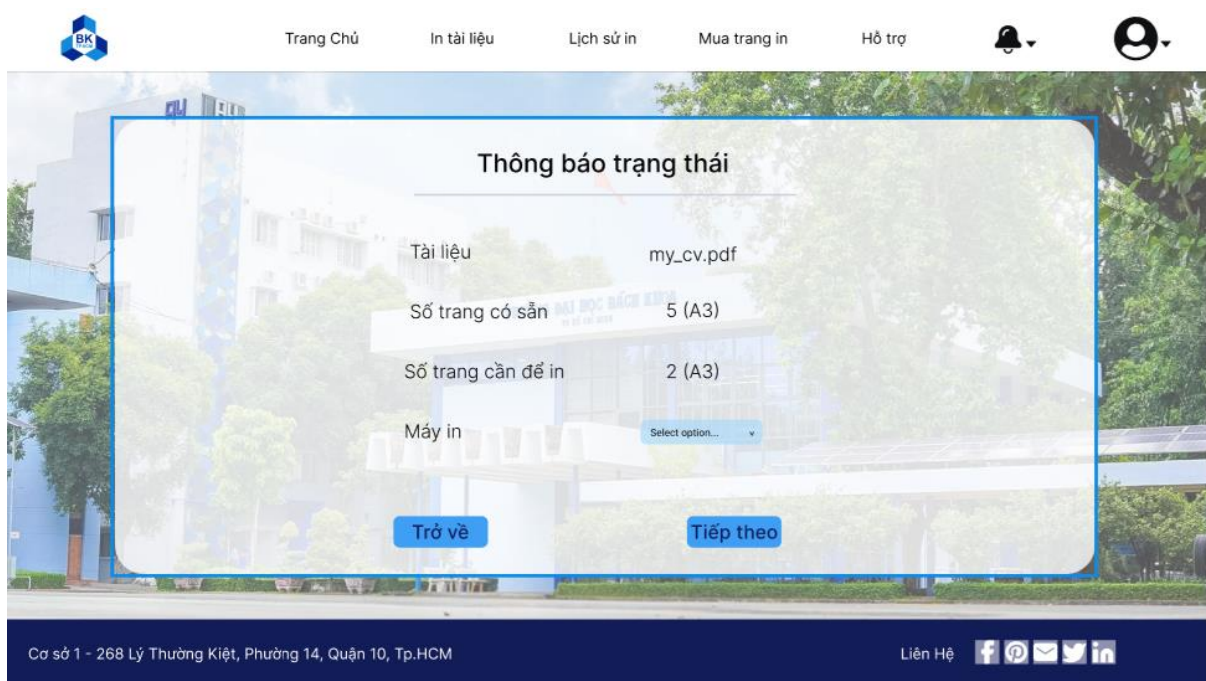
### 3.4.9 Configuration file format page





The left side is displaying the uploaded file, the right side is configuration for it. The student has 6 fields to complete, “Chọn trang cần in” will let user choose pages to print (default is print all pages), “Khổ giấy” will let user choose page size: A3 or A4, “Hướng in” will let user choose page’s direction: top or bottom, “Số trang/1 mặt in” will let you enter prefer number of slide in 1 pages, “Số mặt in/1 tờ” will let user choose between 1 and 2. The same with previous frame, it has 2 buttons to direct user “Trở về” and “Tiếp theo”.

### 3.4.10 Invoice page



**Thông báo trạng thái**

Tài liệu	my_cv.pdf
Số trang có sẵn	5 (A3)
Số trang cần để in	2 (A3)
Máy in	Select option...

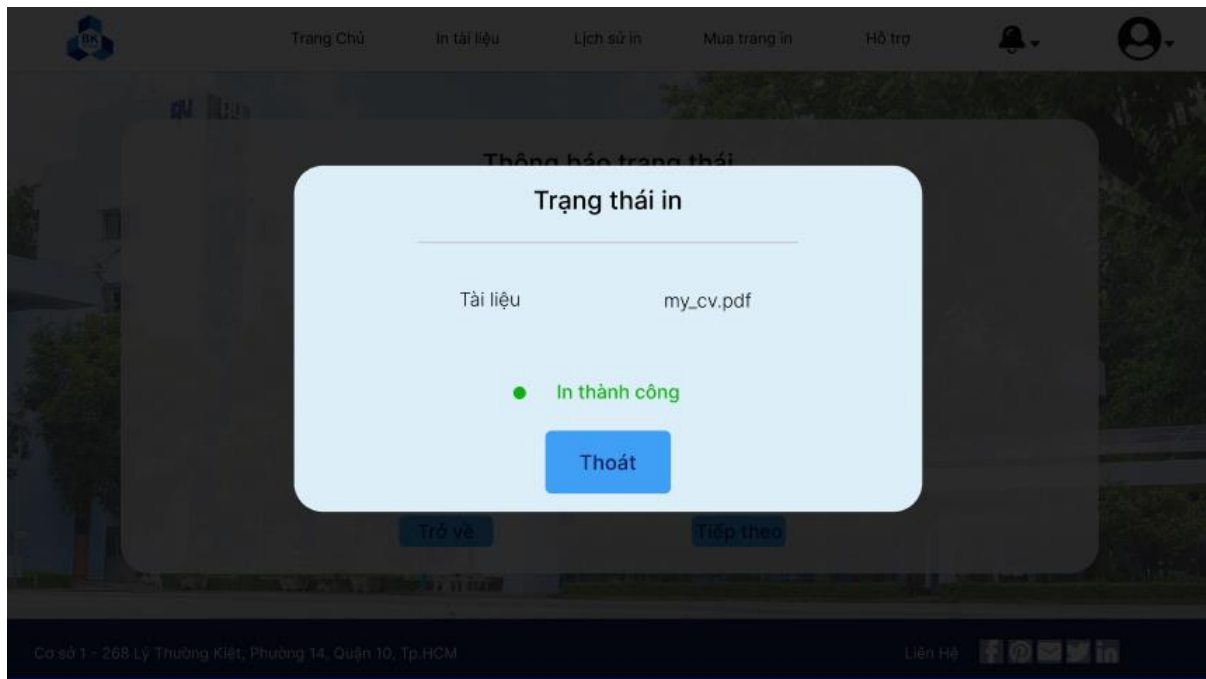
[Trở về](#) [Tiếp theo](#)

Cơ sở 1 - 268 Lý Thường Kiệt, Phường 14, Quận 10, Tp.HCM

Liên Hệ [f](#) [p](#) [e](#) [t](#) [in](#)

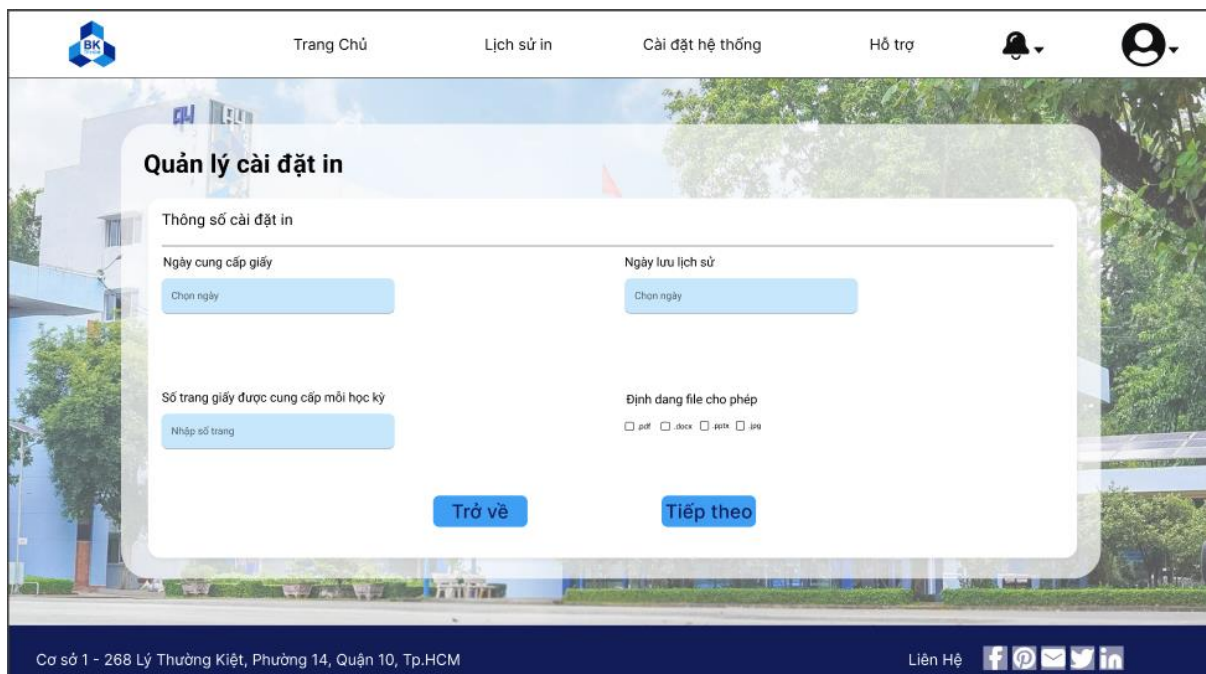
After that, the server will create a kind of invoice that displays some crucial information: File name, Number of page available, Number of pages needed to print, Printer - can choose from a couple of available printers. Button “Trở về” to return, button “Tiếp theo” to forward.

### 3.4.11 Popup successfully print page



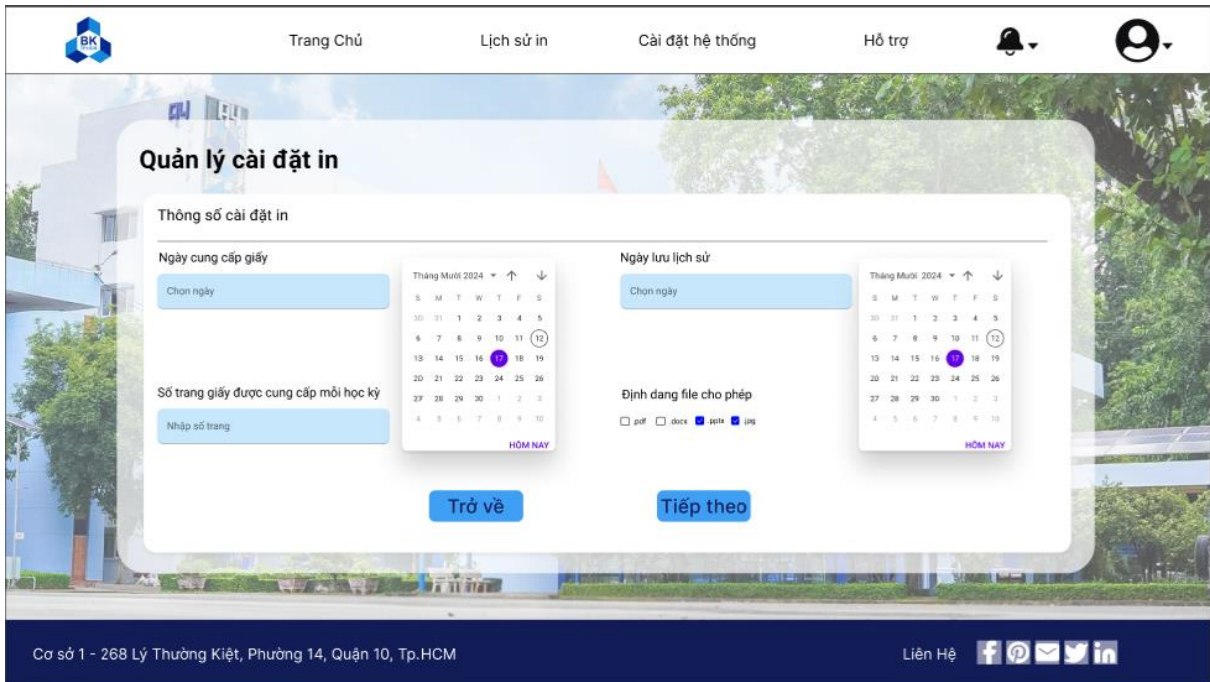
A pop up appears to announce a user that has printed successfully. Click “Thoát” to get out of the flow.

### 3.4.12 System configuration page



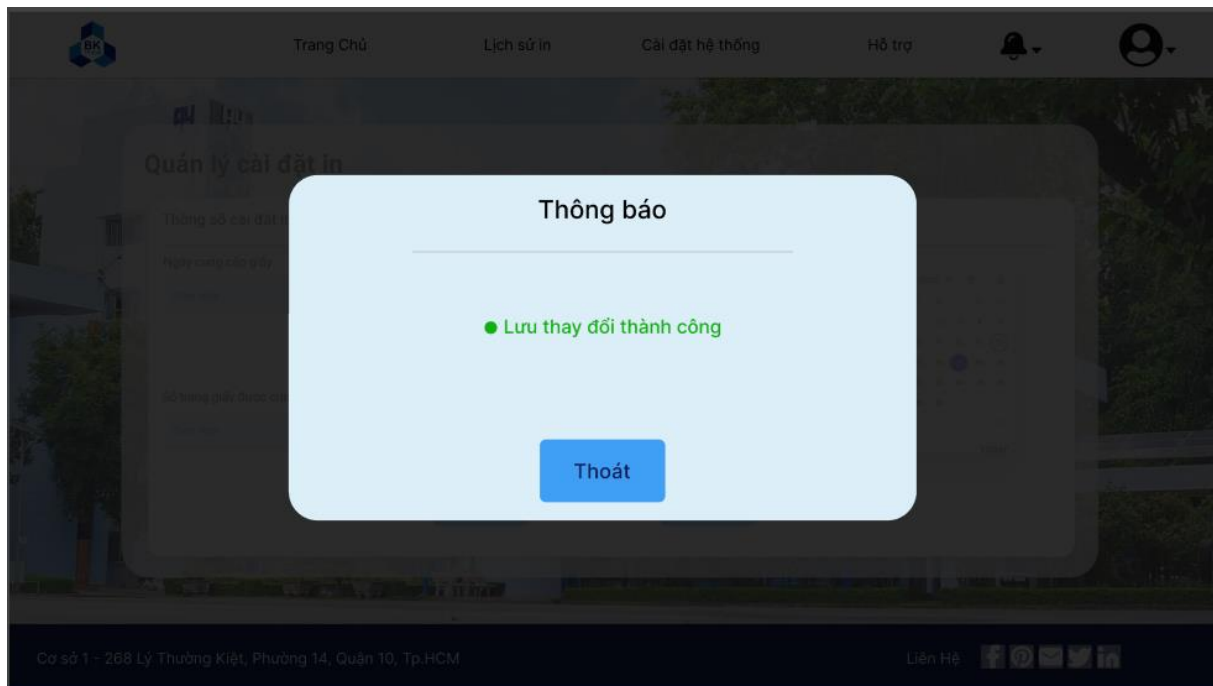
When user navigate through the “Cài đặt hệ thống” in the navigate bar, there will be the user interface (UI) for SPSO to config the default set up of the system.

### 3.4.13 System config in use



Here, user can select date for box “Ngày cung cấp giấy” and box “Ngày lưu lịch sử” using a date picker with the appearance of a small calendar, user can choose date, month, year and the output will be displayed with this format “DD/MM/YYYY”. With the box “Số trang giấy được cung cấp mỗi học kỳ”, user will be able to input an integer number as needed. The box “Định dạng file cho phép” is a multiple-choice checkbox, it will display supported only file types for users to tick. After ticking the box, from a blank square it will turn into a checkbox with a blue background surrounding a white check sign. At the bottom of the frame, there will be 2 buttons which is “Trở về” and “Tiếp theo” for navigating purpose. Button “Trở về” will return user back to the previous page.

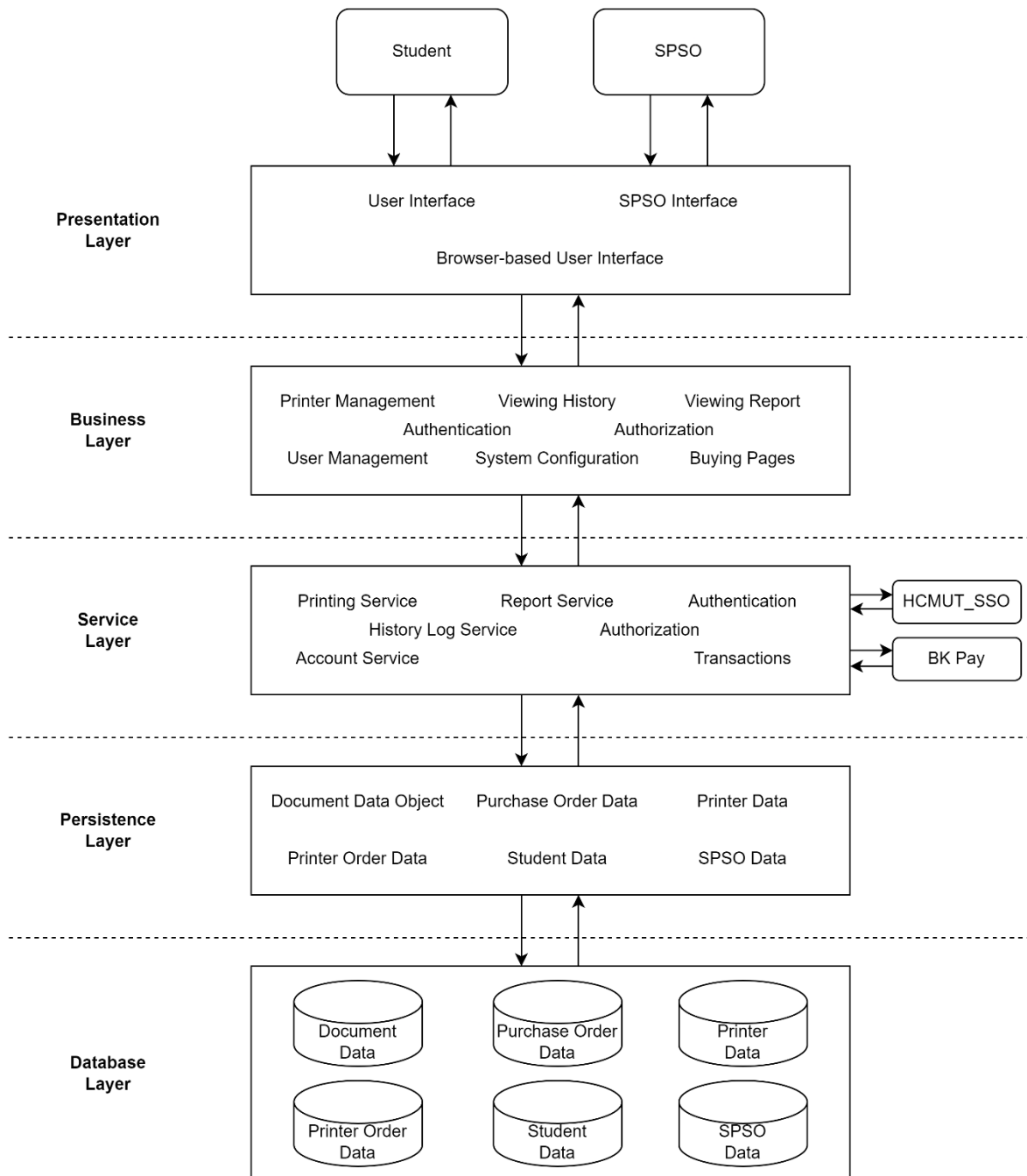
### 3.4.14 Popup successfully save page



When user clicks Button “Tiếp theo” there will be a pop up to inform user that has changed successfully with a text box “Lưu thay đổi thành công”, and a button named “Thoát” to get user out of the flow.

## 4. Task 3: Architecture design

### 4.1 Layered Architecture:



Link Draw.io of the diagram: [here](#)

The system is mainly used by two types of users: Students and SPSO. Therefore, the user interface is also divided into Students' interface and SPSO's interface.



The student's interface has main functions such as Login, Print, Purchase, View History. The logging function will access the HCMUT\_SSO service for user authentication. The printing document function accesses the printing service and interacts with student data through the Persistence layer. The viewing printing history function accesses the history service, it reads data about printing orders and documents from the database. The purchasing printed page's function will access the BKPay payment service and record the transaction results into the purchase order database.

The SPSO interface has main functions such as Login, viewing printing history of customers, viewing periodic reports, managing printers and editing system configuration. Some functions operate similarly to those of the students. In addition, there are the following functions: The periodic report viewing function accesses the report creation service, thereby reading data from related databases. The printer management function accesses the printer management service to read printer information, printer list from the printer database, and update, add, delete, printer information, enable and disable printers. The system configuration function accesses and edits configuration files from the system.

## 4.2 Present User Interface:

Our team choose React for some reason:

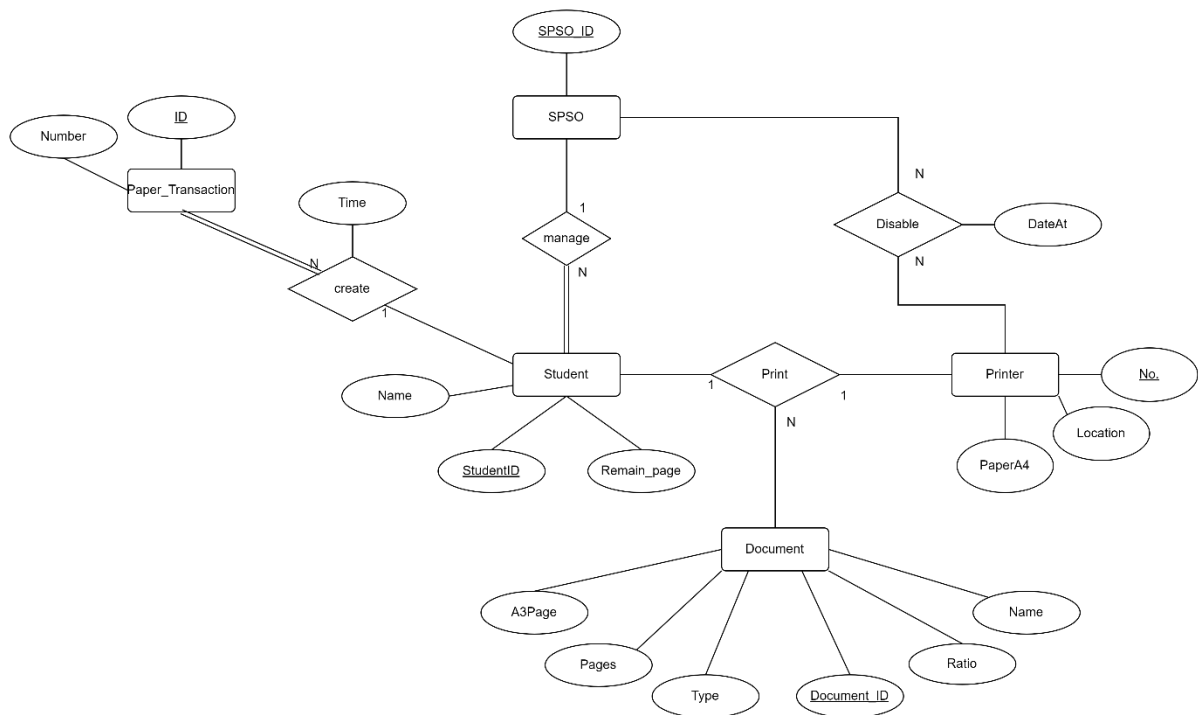
- **Reusable Components:** Create parts of UI (like buttons, print status indicators, and settings) once and reuse them throughout the app, saving time and ensuring everything looks consistent.
- **Smooth Experience:** React makes the app more responsive, so users will have a smoother experience when interacting with things like print jobs or settings.
- **Scalability:** As developpe app and add more features, React's structure makes it easy to scale without making the code messy.
- **Easy Integration:** React can easily work with backend services and third-party tools, add new features quickly.

## 4.3 API management:

- **API for Authentication:** Uses HCMUT\_SSO (HCMUT Single Sign-On) for user authentication, ensuring only authorized users can access the system.

- **API for Document Pre-processing:** When a user uploads a file, this API checks if the file format is acceptable (e.g., PDF, DOCX). It ensures compatibility with the printing system.
- **API for Document Formatting:** After successful upload, this API creates a file in a default format. If the user modifies the format, this API updates the document accordingly to provide a preview before printing.
- **API for System Management:** Allows the SPSO (Smart Printing Service Operator) to manage and modify the system settings as needed.
- **API for Payment:** Facilitates transactions through BKPay, allowing users to make payments for printing services or purchase additional print pages.
- **API for Printing History and Reports:** Generates reports periodically and enables admins and users to view print history, allowing for easy monitoring and tracking of print activities.

#### 4.4 Store Data:



Link Draw.io of the diagram: [here](#)

We chose MySQL for the Smart Printing Service database because it's a reliable, scalable, and secure relational database system, ideal for handling large datasets and enforcing relationships between entities like students, printers, and documents.



#### 4.4.1 SPSO:

##### Attributes:

- SPSO\_ID: A unique ID for each operator managing the printers.

##### Relationships:

- Manages: The SPSO oversees each printer, ensuring it's stocked and working.
- Disables: The SPSO can turn off a printer if it needs maintenance or isn't available.

#### 4.4.2 Student:

##### Attributes:

- StudentID: A unique ID for each student.
- Name: The student's full name.
- Remain\_page: Number of Pages the student has left to print.

##### Relationships:

- Creates: The student buy paper, called a Paper\_Transaction.
- Prints: The student can send a Document to the printer to be printed.

#### 4.4.3 Printer:

##### Attributes:

- No.: A unique ID for each printer.
- Location: Where the printer is located.
- PaperA4: The amount of A4 paper currently in the printer.

##### Relationships:

- **Disabled by SPSO:** The SPSO can disable the printer when needed.
- **Prints:** This printer is used to print documents sent by students.

#### 4.4.4 Document:

##### Attributes:

- Document\_ID: A unique ID for each document printed.
- Name: The title of the document.
- Pages: The number of pages in the document.
- Type: The file type, like .pdf or .docx.
- A3Page: Shows if the document includes any A3-sized pages.
- Ratio: Indicates the print size, such as fit to page or scale.

### Relationships:

- Printed by Student: The student who requested to print this document.
- Printed on Printer: The printer that was used for printing this document.

### 4.4.5 Paper\_Transaction:

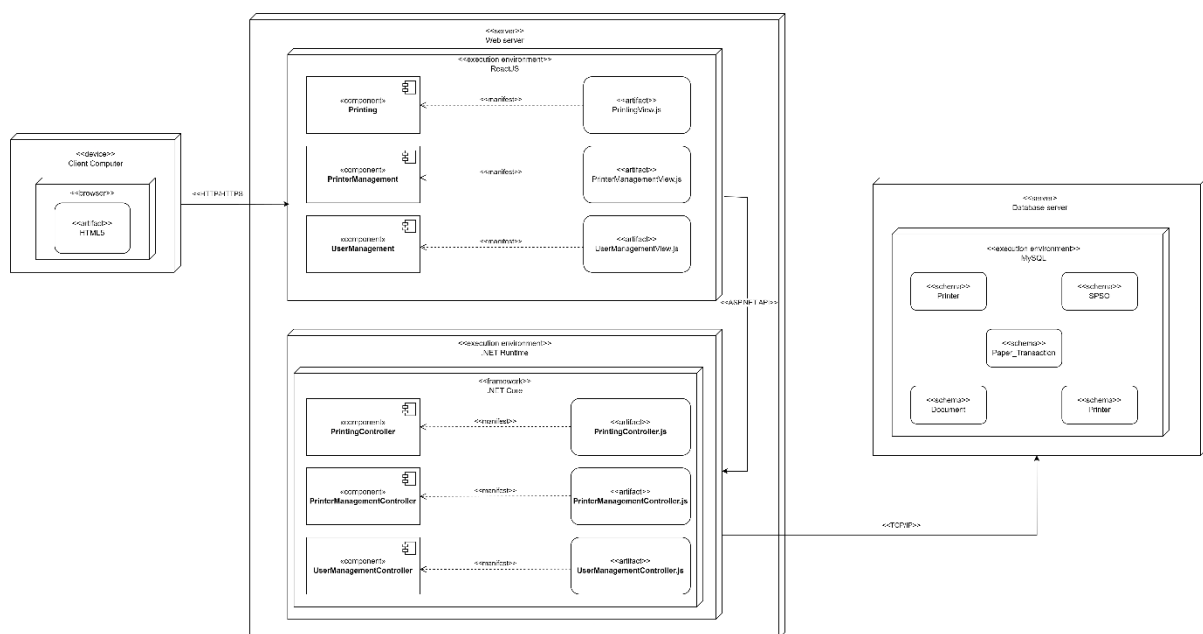
#### Attributes:

- ID: A unique ID for each buying paper transaction.
- Number: The amount of buying paper.
- Time: When the transaction making.

#### Relationships:

- Created by Student: Each transaction is created by specific student.

### 4.5 Deployment Diagram:



Link Draw.io of the diagram: [here](#)

The Client will connect with the server using HTTPS protocol, the data received from the Server are the files that execute the interface and will be displayed on users' browser website via HTML5.

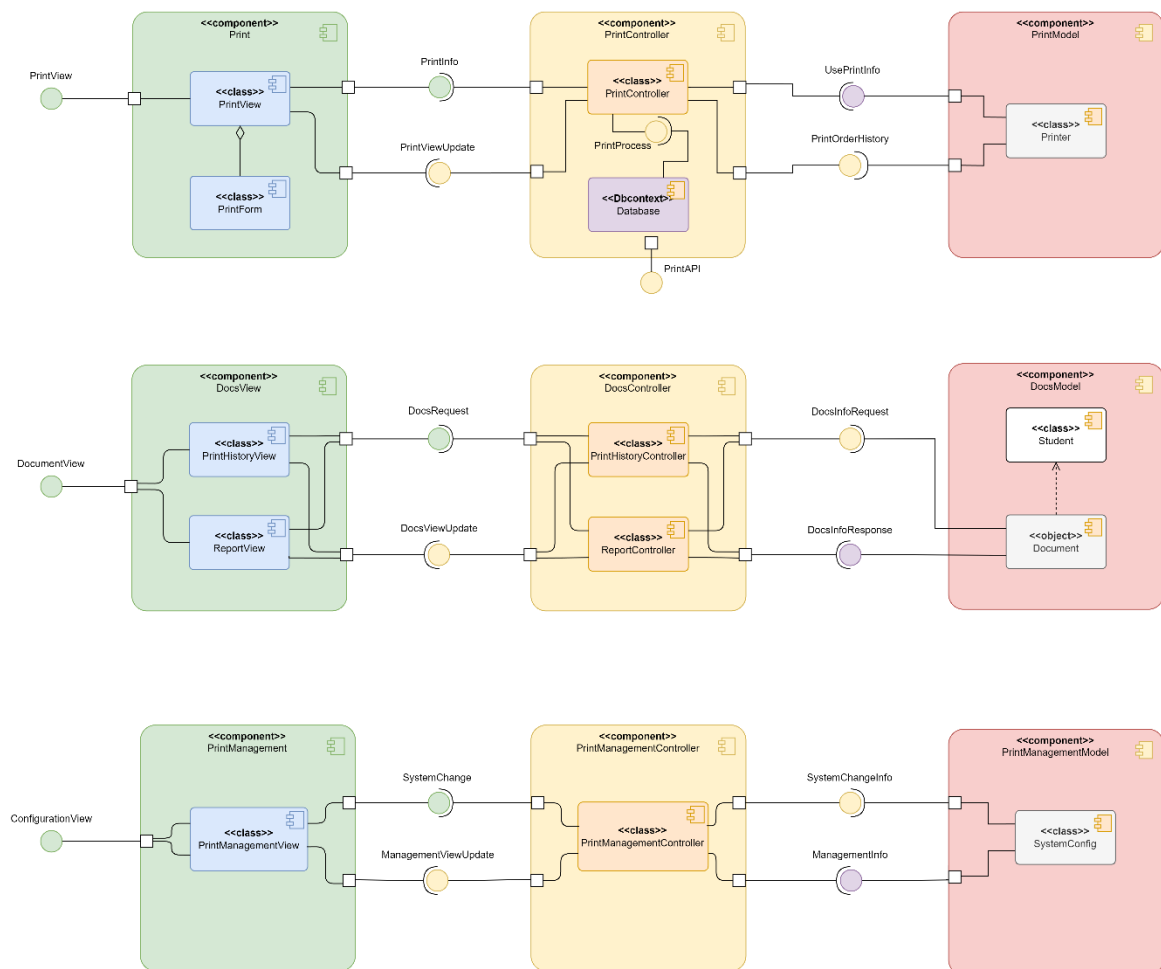
The backend execution environment is NodeJS, using .NET framework, consisting of 3 main components. The frontend execution environment is ReactJS, also consists of 3 main

components. Through ASP.NET API, the frontend and backend sides can communicate with each other to update, display interface, and receive responses from the users.

The Server will connect to the Database Server through TCP/IP protocol, to access data displayed on the interface or update the data whenever the user interacts with the application.

The Database Server includes MySQL DMBS. The Database Server stores data in structured Relational Schemas.

## 4.6 Component Diagram:



Link Draw.io of the diagram: [here](#)

#### 4.6.1 View:

##### 4.6.1.1 Print:

**Main function:** implements interface management classes for printing support.

**Includes classes:**

- PrintView: class provides page interface to perform file upload, printer selection and printing operations.
- PrintForm: class provides form interface to fill in information about printing parameters.

**Relationship between components:** PrintView class relies on information entered from PrintForm to receive values about printing parameters and then send them to the controller for processing.

PrintView implements PrintView, PrintInfo interface.

All components require PrintViewUpdate interface.

##### 4.6.1.2 DocsView:

**Main function:** Implements classes that provide interfaces for viewing print history and periodic reports.

**Includes classes:** PrintHistoryView, ReportView

- Class PrintHistoryView provides an interface for viewing student print history (and printers if logged in by SPSO).
- Class ReportView provides an interface for viewing periodic reports.

Both components implement the DocumentView and DocsRequest interfaces.

Both components require the DocsViewUpdate interface.

##### 4.6.1.3 PrintManagement:

**Main function:** Implements the printer management interface for SPSO.

Both components implement the PrintManagementView, SystemChange interfaces.

Both components require the ManagementViewUpdate interface.

#### 4.6.2 Controller:

##### 4.6.2.1 PrintController:

**Main function:** implements a class that provides event handling methods and information about user input print parameters, print balance accounts, and the print process. It will get the PrintAPI from printer also.

Database control the printController to print the process.

Both Component implements the PrintViewUpdate, PrintOrderHistory, PrintProcess

Both components require the UsePrintInfo interface.



#### 4.6.2.2 DocsController:

**Main function:** implements classes that provide methods to process timestamps (or add printer codes if SPSO) that users enter.

**Includes classes:** PrintHistoryController, ReportController.

- PrintHistoryController class provides methods to process information about timestamps to view student printing history (and printer if SPSO).
- ReportController class provides methods to process timestamp information to view periodic reports.

Both components implement the DocsViewUpdate and DocsInfoRequest interfaces.

Both components require DocsInfoResponse interfaces.

#### 4.6.2.3 PrintManagementController:

**Main function:** Implements a class that provides methods for handling information streams that require updating information about printer management.

Component implements the ManagementViewUpdate, SystemChangeInfo interfaces.

Component requires the ManagementInfo interfaces.

### 4.6.3 Model:

#### 4.6.3.1 PrintManagementModel:

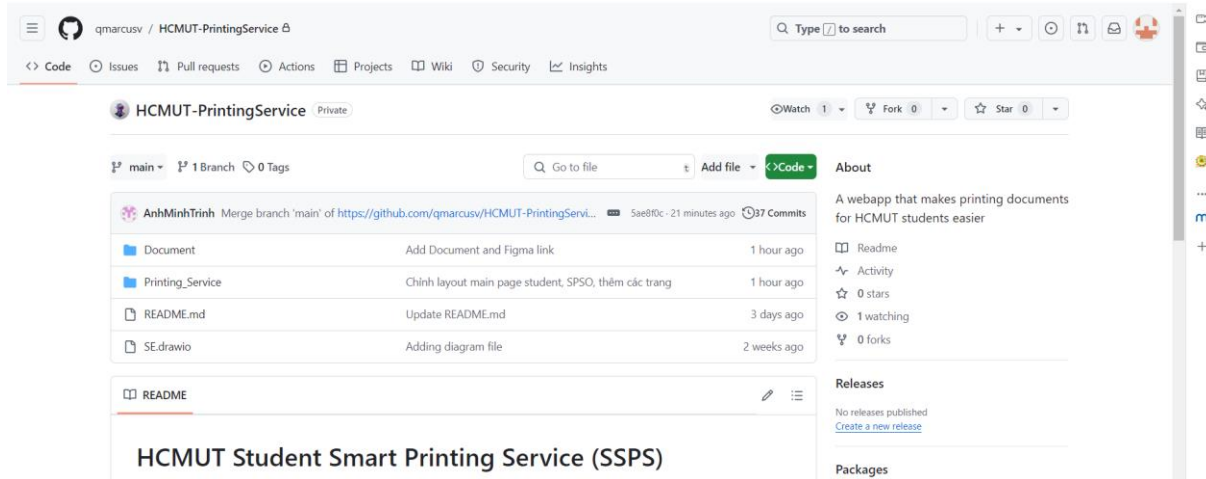
**Main function:** implements a class that provide information of System. It is not included in Database.

### 4.6.4 Database:

Schema Printer, Student, Document are generated in Database for the main purpose store data and retrieve the history.

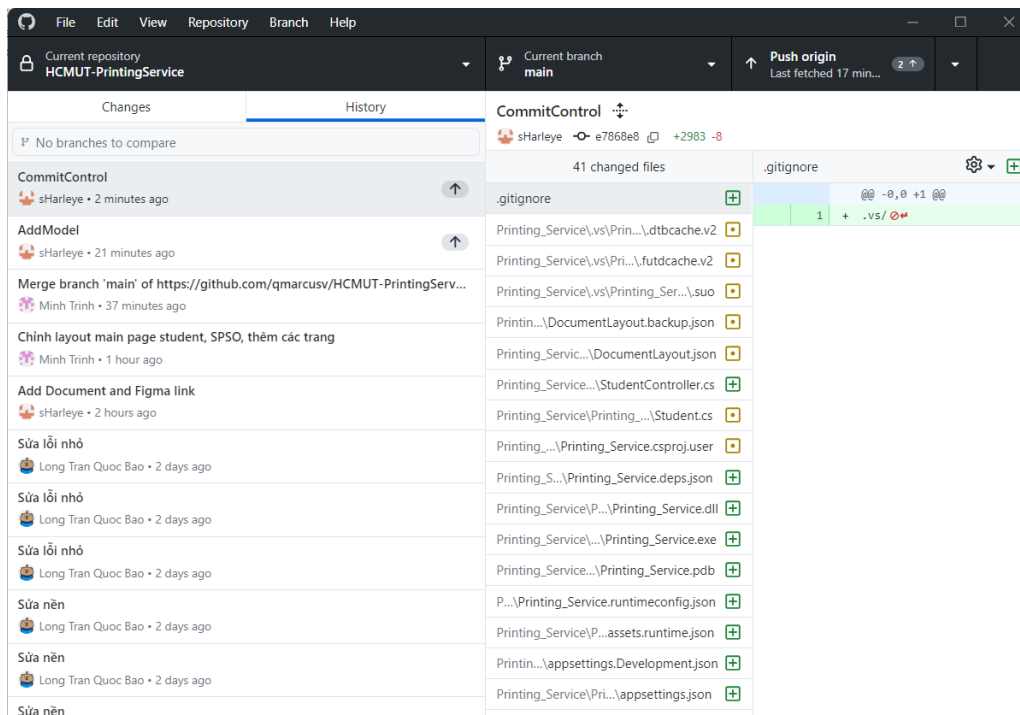
## 5. Task 4: Implementation – Sprint 1

### 5.1 Setting up an online repository (github, bitbucket, etc) for version control



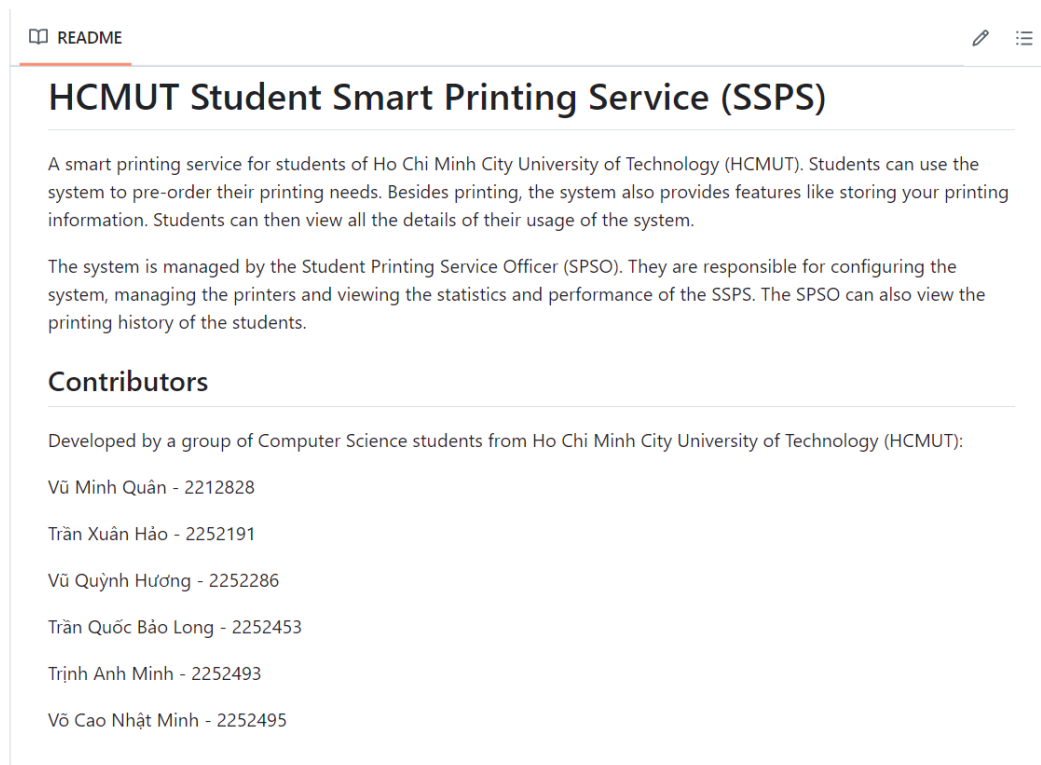
We set up GitHub and push latest-report, source code...

### 5.2 Adding documents, materials and folders for Requirement, System modelling and Architectural design. Use the selected version control system to report the changes to these files.



This is our history so far, where we begin coding our app and more.





This is the initial version of our ReadMe.md. As we continue developing the app, we will update and expand this document to provide more detailed information.

## 5.3 Conducted a usability test with the user interface.

### 5.3.1 Recruit participants/ testers.

The tester (also known as the facilitator or supervisor) plays the role of guiding participants through the testing steps, answering any questions they may have during the process, recording the testing procedure, and analyzing the records to improve the product. The tester sets tasks and requires participants to perform them. Participants in the test are individuals hired to evaluate the product by completing the tasks assigned by the tester. These tasks can be communicated to participants in two forms: verbal and written. For both forms, participants are asked to voice their thoughts and explain how they perform the tasks, allowing the tester to understand the participants' behaviors, desires, thoughts, and motivations, as well as those of the broader user group. They serve as representatives of the target user group.

### 5.3.2 Define tasks.

Our team has divided the 'Print for Student' module into smaller tasks, such as:

- HomePageStudent:
- UploadFile
- PrintingConfig
- Printing

Our team has divided the 'Configure System for Student' module into smaller tasks, such as:

- HomePageSPSO
- SystemConfig


The purpose of breaking it down into smaller tasks is so that the tester can focus on each page individually, analyze it, and identify issues that need to be addressed, both in terms of the interface and user satisfaction with the features.

### 5.3.3 Define test strategy (qualitative vs. quantitative, remote vs. inperson.)

After the team held a meeting and thoroughly analyzed the project requirements, it was decided to adopt remote testing as the preferred method. This approach would not only provide greater flexibility but also allow the team to reach a more diverse group of participants, ensuring that feedback is representative of a wider range of users. The team will utilize qualitative testing methods to gain a deeper understanding of user experiences. By focusing on observations, interviews, and detailed feedback, the goal is to uncover insights into users' motivations, behaviors, and challenges. This will help identify potential areas of improvement in the product and ensure it meets the needs and expectations of its target audience.

### 5.3.4 Conduct the test.

To carry out the testing, our team created a Google Form that includes all the tasks participants need to complete. Additionally, the form contains a detailed guide on how to use the product, ensuring that users have a clear understanding of the process and can have the best possible experience during the test.



## User Testing for UI

Here is some information to use this.  
Link to use: [Here](#)  
Thank you.

huong.vuquynh0611@hcmut.edu.vn [Chuyển đổi tài khoản](#)

Không được chia sẻ

\* Biểu thị câu hỏi bắt buộc

Your name \*

Câu trả lời của bạn

Tester Role

☐ SPSO

☐ Student

### 5.3.5 Document the feedback from testers.

HomePageSPSO	SystemConfig	HomePageStudent	UploadFile	PrintingConfig	Printing	Rating	UsedTimes	
5	10						8	1
		8	8	8	8	8	8	3
9	10						10	1
8	10						9	1
9	9						9	2
		7	7	8	6	7	7	1
		8	9	7	8	8	8	2
		8	7	7	7	7	7	1

This is the rating result for each defined task. In addition, we also gather feedback to continuously improve our project.

#### Feedbacks

I think you should improve the HomePage, it is not friendly.

Improve on the layout of the mainpage of student

Everything is ok

It will be nice if you change HomePage.

Change the ratio of some button for better interface

Useful app, but not-so-friendly interface.

The student's main page is OK, but some improvements are needed

The printing configuration page should have an option for printing specific

Many feedback responses suggested that we should improve the homepage to make it more aesthetically pleasing and user-friendly. We have taken this feedback into account and have designed a homepage that best fits the app.

### 5.3.6 Improve the project

After gathering feedback from several people, we have revised the HomePageSPSO and the HomePageStudent. Additionally, we have adjusted in Figma to ensure the project progresses more smoothly.

