Software Requirements Specification

for

SMART PRINTING SERVICE FOR STUDENTS AT HCMUT

Version 1.0 approved

Prepared by:

1. Phan Thanh Son – 2252718
2. Tran Dang Hien Long – 2252449
3. Tran Hoang Phuc – 2252647
4. Tran Nguyen Anh Khoa – 2252364
5. Tran Thi Hong Hanh – 2252193
6. Tran Thien Phu – 2252623

Department of Software Engineering

Faculty of Computer Science and Engineering

Ho Chi Minh City University of Technology – VNU-HCM

September, 2024

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Requirement elicitation (1.1, 1.2)	September 22nd, 2024	Complete domain context, stakeholders and needs, benefits of each stakeholder of the system, functional requirements and non-functional requirements.	1.0
Use Case Diagram (1.3)	September 30th, 2024	Use case diagram for the whole system, use case diagram and the detail of each use case of Printing Documents Module.	1.1
System Modelling	October 27th, 2024	Create activity diagram, sequence diagram, class diagram and user interface using figma for Printing Document Module	1.2
Architecture Design	November 13th, 2024	Create architectural diagram, component diagram and their descriptions	1.3

1. Requirement elicitation

1.1 Domain Context

Printing study materials at Ho Chi Minh City University of Technology (HCMUT) has long been difficult due to the absence of an on-campus printing system. Instead, students are dependent on off-campus vendors, which introduces significant inconvenience and unpredictability. Not only can the pricing be uncertain, but the physical distance to these vendors often means students face unexpected costs and additional travel time, further compounding the frustration.

The HCMUT Student Smart Printing Service (SSPS) addresses these issues by offering a dependable and straightforward alternative by:

- **Enabling Digital Uploads:** Students can upload files and configure print settings through a user-friendly web and mobile application.
- **Providing Clear Options***:* Offers straightforward choices for paper size, colour, and quantity, with transparent pricing.
- **Managing Print Quotas:** Allows students to easily monitor and manage their print quotas, ensuring no unexpected costs.

With SSPS, students benefit from a reliable and efficient printing service, minimizing the uncertainties and inconvenience associated with off-campus vendors and enhancing their overall academic experience.

1.2 Stakeholders and Needs

1.2.1 Student Printing Service Officer (SPSO)

- Description: SPSO is the one who manages and monitors the printing activities in the system
- Needs:
    + Manage all printers across the campus.
    + Configure system-wide settings about files, printing pages, policies.
    + Access logs and reports of all students using printing service.
    + Monitor the status of the printing system.
    + Resolve students' complaints and issues related to printing.

+ Maintenance requirements, refilling paper/ink.

1.2.2 University Administration (HCMUT)

- Description: Manages the overall operation and efficiency of the printing service, ensures student satisfaction, and monitors financial and operational performance.
- Needs:
  + Keep track and manage the resources and financial efficiency of the service.
  + Ensure a reliable printing service is provided for students.
  + Ensure data security and privacy.
  + Control over system policies, such as page limits, semester-based allocations, and pricing.
  + Generate reports on overall system usage for budget planning and service improvements.

1.2.3 Students

- Description: Main users of the system's printing services. They are the ones who create printing and download requests, upload documents, select printers, and make payments
- Needs:
  + Easy access to print documents via both web and mobile platforms, user-friendly and reliable.
  + Ability to create requests to print documents quickly and can access affordable prices for students.
  + Ability to upload files, select printer locations, and configure print settings.
  + View and keep track printing history, page summary and available balance.
  + Can buy more printing pages through the BK-Pay system.

1.2.4 Student Management System

- Description: This system is responsible for storing and managing student data, such as student ID, name, major, faculty, and status (active, graduated, on leave,...). It provides this information to the HCMUT_SSPS to facilitate smooth authentication, tracking, and management of student-related activities within the printing service.
- Needs:
  + Seamless integration with the HCMUT_SSPS to ensure accurate and up-to-date student information.

+ A secure and reliable connection to transfer sensitive student data without compromising privacy.
+ Regular updates to student records that reflect any changes in student status (graduation, withdrawal, etc.) in real-time.
+ The ability to log access requests for auditing purposes to ensure data security.

1.2.5 Payment System (BKPay system)

- Description: processes online payments for students buying additional printing pages, ensuring secure and reliable payment transactions.
- Needs:
  + Secure and process online payments made by students to buy more printing pages.
  + Ensure the reliability and uptime of the payment platform.

1.2.6 Finance Department

- Description: Manages the financial aspects of the printing service, including tracking payments made by students and generating financial reports on the service's usage.
- Needs:
  + Integration with the BKPay for student transactions.
  + Track and manage payments made by students when purchasing additional printing pages.
  + Generate financial reports related to printing usage and income from printing services.

1.2.7 Guests

- Description: Guests are users who do not log in to the system but can view general information about the HCMUT Student Smart Printing Service
- Needs:
  + Access to basic information about the printing service, such as available printer locations, operating hours, and pricing details.
  + Information on how to use the system and any frequently asked questions
  + Security to ensure that while viewing general information, no sensitive data is exposed or accessed.

1.2.8 Development Team

- Description: This team is responsible for maintaining the technical infrastructure of the system, ensuring secure authentication, integration with payment systems, and resolving any technical issues.
- Needs:
  + Develop the printing system.
  + Maintain system security and data integrity, including student data, printing logs, and payment details.
  + Ensure system scalability and reliability for large numbers of students and printers.
  + Handle any system issues, security threats, or technical problems quickly.

1.2.9 Printer Manufacturers

- Description: Supply and maintain the printers used in the system, ensuring they are compatible and functioning properly, and providing technical support as needed.
- Needs:
  + Provide, install and maintain the printer across the campus.
  + Ensure technical support related to the printers and repair when needed.

1.3 Benefits of the System

1.3.1 Student Printing Service Officer (SPSO)

- Streamlined Printer Management: Simplifies the management of printers across campus with centralized controls and monitoring.
- Policy Configuration: Provides flexibility to set printing policies such as page limits, file types.
- Efficient Issue Resolution: Quickly address student complaints and resolve printing issues.
- Access to Reports: Automatically generated monthly and yearly reports for improvement.
- Maintenance Tracking: Helps manage paper and ink refill needs, ensuring minimal downtime for printers.

1.3.2 University Administration (HCMUT)

- Operational Efficiency: Automates student page allocations and usage tracking, allowing better resource management and cost control.

- Data Security: Ensures secure handling of sensitive student data with HCMUT_SSO authentication, preventing unauthorized access.
- Financial Planning: Provides insights through usage reports, which help in budgeting and resource allocation for the printing service.
- Service Reliability: Increases student satisfaction by maintaining a stable, reliable printing service across campuses.

1.3.3 Students

- Convenience and Accessibility: Easy access to printing services via both web and mobile platforms, offering flexible usage from anywhere on campus.
- User-Friendly Interface: Allows students to upload files, select printers and have personalized settings.
- Flexible Payment Choices: The ability to purchase additional pages through BKPay ensures affordability and flexibility in managing printing needs.
- History Tracking: Students can view their printing logs, track usage, and monitor their available balance, promoting accountability.

1.3.4 Student Management System

- Seamless Integration: Results in up-to-date student data, facilitating accurate and smooth authentication and tracking within the HCMUT_SSPS.
- Data Privacy: The secure transfer of sensitive information protects student data, following privacy guidelines.
- Real-Time Updates: Reflects any changes in student status (e.g., graduation, leave) instantly, resulting in system accuracy.
- Audit Logs: Logs access requests for auditing purposes, ensuring data security and regulatory compliance.

1.3.5 Payment System (BKPay System)

- Secure Payments: Facilitates secure, real-time online payments for students purchasing additional printing pages.
- Reliability: Ensures high availability and uptime, which is crucial for uninterrupted transactions and student convenience.
- Integration: Streamlined with the HCMUT-SSPS to provide seamless payment experiences.

1.3.6 Finance Department

- Transaction Tracking: Tracks payments made by students for additional printing pages, providing accurate financial records.
- Report Generation: Generates financial reports on the system's usage and income, aiding in financial planning and analysis.
- BKPay Integration: Facilitates easy integration with the BKPay system for reliable tracking and management of financial data related to printing services.

1.3.7 Guests

- Access to Information: Provides basic information like printer locations, pricing, and operational hours, ensuring transparency.
- Guides and FAQs: Offers helpful resources on how to use the system and common queries, making it easier for newcomers to understand.
- Security: Results in no sensitive data is accessible while browsing general information, protecting the integrity of the system.

1.3.8 Development Team

- Streamlined System Development: Enables efficient development, updates, and customization of the printing system, reducing development time and improving system functionality.
- Security and Data Integrity: Ensures the protection of student data, payment details, and printing logs through secure authentication (HCMUT_SSO) and data management.
- Scalable and Reliable Infrastructure: Provides a scalable system that can support a large number of students and printers, reducing the need for major overhauls and ensuring long-term stability.
- Proactive Issue Resolution: Enables faster detection and resolution of system issues or security threats, minimizing downtime and improving overall system reliability.
- Seamless System Integration: Ensures smooth integration with third-party systems like BKPay and the Student Management System, simplifying technical maintenance and improving user experience.

1.3.9 Printer Manufacturers

- Maintenance and Support: Enables manufacturers to maintain and provide technical support for printers effectively, ensuring ongoing service quality.
- System Compatibility: Ensures that printers are fully compatible with the system, leading to a smoother operation and fewer issues.

- Repair Services: Streamlines the process for identifying and addressing technical problems, resulting in quick turnaround times for repairs.

1.4 Functional Requirements

1.4.1 Student Printing Service Officers (SPSOs)

- SPSOs can access and view student account information by using the system's administrative interface.
- SPSOs can view the status and information of all printers in the system by accessing the printer management dashboard.
- SPSOs can add, activate, or deactivate printers by configuring the system's printer settings.
- SPSOs can manage system configurations, including adjusting the default number of pages allocated to students and selecting file formats for upload through the system's settings panel.
- The system must generate monthly and yearly statistical reports, which only SPSOs can access through the reporting function.

1.4.2 University Administration (HCMUT)

- The university administration can access analytics on printing resource usage through a reporting dashboard for budget planning and resource allocation.
- The system must enforce user authentication through HCMUT_SSO by integrating the university's Single Sign-On (SSO) system.
- The university administration can set policies regarding print quotas and pricing for students by accessing the system's policy configuration tools.
- The system must implement data security measures to ensure the protection of user information and printing logs by using encryption protocols.
- The system should facilitate communication between students and SPSOs by enabling a feedback mechanism within the platform.

1.4.3 Students

- Students can access and view personal information by logging into the system through their account.
- Students can select a file from their device and upload it to the system by using the file upload feature.

- Students can select a specific printer based on their desired location by using the printer selection tool within the system.
- Students can choose a time to receive their print jobs by scheduling through the system's time management feature.
- The system must notify students via email when their print job is completed by sending an automated email.
- Students can adjust printing settings (e.g., page size, number of copies, single or double-sided) by configuring options within the system.
- The system must store all print history, including student ID, printer ID, start and end times, and page counts by logging each print activity.
- Students can access and review their previous print jobs by using the system's history feature and filtering based on time or printer location.
- The system must allocate a specific number of printing pages to each student at the beginning of each semester.
- Students can purchase additional printing pages through the integrated electronic payment system (BKPay).

1.4.4 Student Management System

- The student management system can provide real-time access to student data for authentication purposes by integrating with HCMUT-SSPS.
- The student management system must support secure data transfer for student information by using encrypted communication protocols.
- The student management system should automatically update student records (e.g., graduation or withdrawal) by synchronizing data with HCMUT-SSPS.
- The student management system must log access requests by maintaining detailed logs of who accessed student data and when for auditing.
- The student management system must allow efficient retrieval of student information by HCMUT-SSPS for operational tasks through a secure API.

1.4.5 Payment system (BKPay)

- The payment system can securely process payments for students purchasing additional printing pages by using the integrated payment gateway.
- The payment system must ensure uninterrupted transaction processing by maintaining high availability and reliability.
- The payment system should provide instant confirmation of successful transactions by sending notifications through the integrated user interface.

- The payment system must support refunds and transaction history retrieval by enabling access to payment logs within the system.
- The system must send email receipts to students after successful payments through the automated email function.

1.4.6 Finance Department

- The finance department can access detailed reports on student payments for printing services by generating financial reports from the system.
- The finance department can track and manage all financial transactions related to the printing service by accessing system transaction logs.
- The finance department should generate financial summaries for budget planning by using the system's financial reporting tools.
- The finance department must ensure compliance with financial regulations by auditing student transactions through the system.
- The finance department must provide financial oversight by conducting regular audits of the printing service's operations through detailed reports.

1.4.7 Guest

- The system provides a homepage containing general information for guests by displaying system details.
- Guests can view information about the system by accessing the homepage.
- Guests can review key features of the system by navigating through the homepage.
- Guests can view usage instructions by accessing the dedicated help section on the system.
- Guests can contact support by using the contact information provided on the system.
- Guests can log in to the system by following the authentication process to access services.
- The system must assign roles (students, SPSO, staff) upon successful login through the authorization module.

1.4.8 Development Team

- The development team can deploy new system features and updates by using a continuous integration/continuous deployment (CI/CD) pipeline.
- The system must allow the development team to monitor system performance and detect errors in real-time by using integrated logging and monitoring tools.
- The maintenance team can schedule system maintenance windows by configuring downtime periods within the system's maintenance module.

- The development team must implement security patches and updates by following secure coding standards and best practices.
- The system must allow the maintenance team to back up and restore data by using an automated backup solution.
- The development team can track and resolve bugs by using an integrated issue-tracking system linked to the development environment.
- The system must notify the development and maintenance team of critical system issues through automated alerts via email or messaging platforms.
- The development team must document all system changes and updates by maintaining version control and update logs.
- The maintenance team can roll back system updates or changes by using the version control system when issues arise.
- The system must facilitate communication between development, maintenance, and other stakeholders by providing a ticketing and communication platform (BKSI) for issue reporting.

1.4.9 Printer Manufacturers

- Printer manufacturers must provide installation and setup support for all printers deployed on campus by dispatching technical teams as needed.
- Printer manufacturers must ensure all printers are compatible with HCMUT-SSPS by providing technical documentation and system integration support.
- Printer manufacturers must supply technical specifications for each printer by including documentation within the system for reference.
- Printer manufacturers must offer ongoing technical support and maintenance services by providing a dedicated support team for troubleshooting and repairs.
- Printer manufacturers must supply updates or patches for printer firmware by releasing them, when necessary, through the system's update management tool.
- Printer manufacturers must ensure that parts and supplies are readily available for maintenance and repairs by maintaining stock at local service centers.

1.5 Non - Functional Requirements

1.5.1 Performance Requirements

- Response Time: System should process requests within 2 seconds under normal load.
- Scalability: Support up to 10 000 concurrent users without performance degradation.

- Throughput: Handle up to 100 print job submissions per minute.
- System Load: Maintain optimal performance for up to 100 000 active users daily.

1.5.2 Safety Requirements

- Data Backup: Automated daily backups to ensure data safety.
- Data Recovery: Restore lost data within 30 minutes in case of failure.
- Failure Handling: Ensure safe shutdown to prevent data loss during system failures.

1.5.3 Security Requirements

- Authentication: Integrate with HCMUT's Single Sign-On (SSO) for secure user authentication.
- Data Encryption: Use SSL/TLS for data transmission and AES-256 for data storage.
- Access Control: Implement role-based access control (RBAC) for system functionalities.
- Logging: Log access and data changes for auditing, retaining logs for 2 years.

1.5.4 Software Quality Attributes

- Usability: Provide a user-friendly interface requiring minimal training.
- Reliability: Ensure 99.9% uptime to minimize service disruptions.
- Maintainability: System updates and patches should be easy to deploy with minimal service impact.
- Compatibility: Ensure cross-browser support (Chrome, Firefox, Safari) and mobile compatibility.

1.5.5 Business Rules

- Print Quotas: Students receive 100 pages per semester, with additional pages available for purchase.
- Refunds: System failures will trigger automatic refunds or equivalent solutions.

2. Use-case Diagrams

2.1 Use-case Diagram for the Whole System

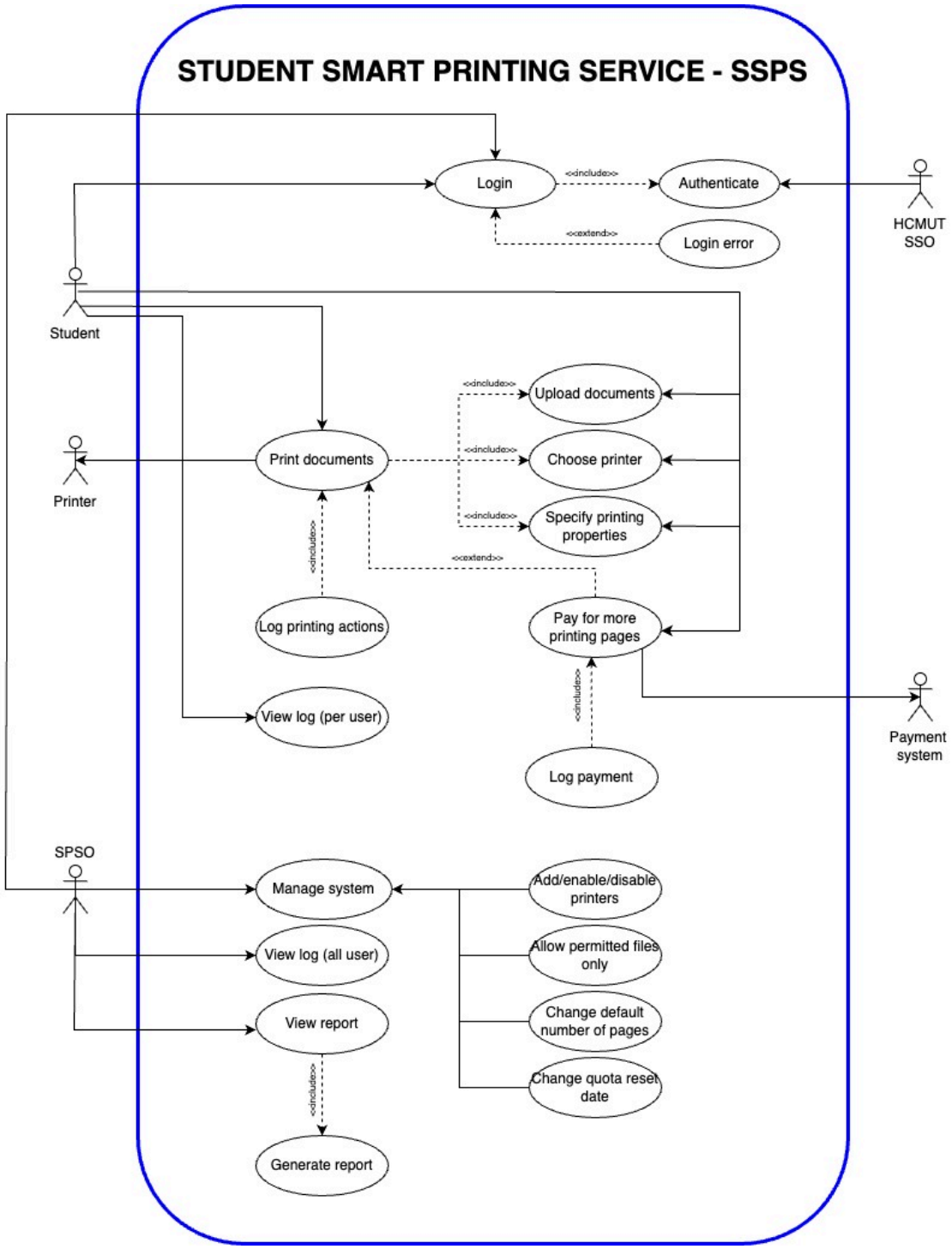

Figure 1: Use Case Diagram for The Whole System

2.2 Use-case Diagram for Printing Documents Module

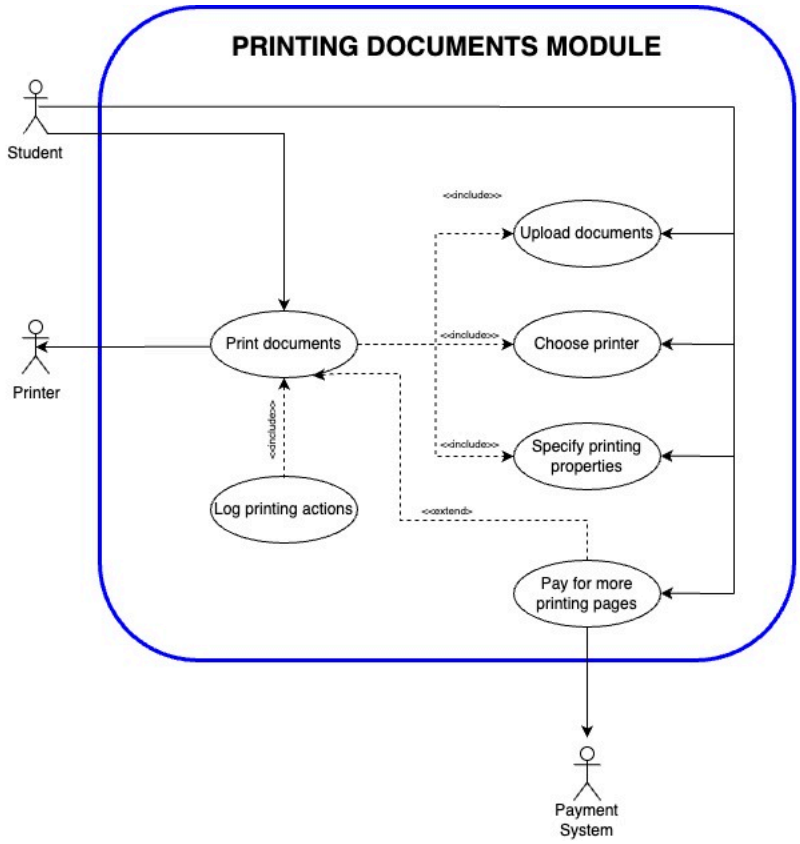

Figure 2: Use Case Diagram for Printing Documents Module

2.3 The Details of Use Cases in Printing Documents Module

2.3.1 Use case Print Document

ID and Name	**UC-1 Print Document**
Created By	Tran Thi Hong Hanh
Date Created	September 28th, 2024
Primary Actor	Student
Secondary Actors	Printer
Description	Student prints documents uploaded into HCMUT_SPSO

Trigger	After completing all printing configuration, students clicks to **Print** button to trigger this use case
Preconditions	PRE-1: The device has to be connected to the internet. PRE-2: The student has to log in to the system and be authenticated by HCMUT_SSO. PRE-3: The student has to upload documents, identify printing properties, and choose the printer. PRE-4: The student has enough available pages to print. PRE-5: Printer is available.
Postconditions	POST-1: Documents are printed successfully. POST-2: Page balance is updated. POST-3: Printing history is recorded.
Normal Flow	1. The student has to be authenticated by HCMUT_SSO. 2. The student uploads documents to the system. 3. Student configs printing properties. 4. Student chooses the printer that is available. 5. Student presses the **Print** button to trigger this action. 6. The system accepts requests and sends the request to the chosen printer. 7. Printer starts printing documents. 8. The system logs the printing activities.
Alternative Flow	None
Exceptions	E1. Invalid login credentials. E2. Invalid file type or file size limit. E4. No printer is available. E6a. The student's remaining printing pages are not enough. E6b. Printer rejects the request due to an error (e.g., printer jam, no paper).

Table 1: Detail of Use Case Print Document

2.3.2 Use case Upload Documents

ID and Name	**UC-2 Upload Documents**
Created By	Phan Thanh Son

Date Created	September 28th, 2024
Primary Actor	Student
Secondary Actors	None
Description	The student uploads a document to the HCMUT_SPSO system for printing. The system validates document format and the document is available for printing if it passes the validation.
Trigger	When a student decides to print a document by uploading it into the HCMUT_SPSO system.
Preconditions	PRE-1: The student must be authenticated through the HCMUT_SSO service. PRE-2: The document must be in a permitted file format. PRE-3: The system must be connected to the internet.
Postconditions	POST-1: The document is successfully uploaded and stored in the system. POST-2: The student can view the preview of uploaded documents. POST-3: The document is available for further actions (to be sent to printers for printing).
Normal Flow	1. The student has to be authenticated by HCMUT_SSO. 2. The student selects the "Upload Document" option. 3. The system displays the document upload interface. 4. The student selects and uploads a document from their device. 5. The system validates the document's file type and size. 6. The system confirms that the upload is successful. 7. The student can proceed with printing if all the requirements are satisfied.
Alternative Flow	None
Exceptions	E5a: The document is in an unsupported file format. E5b: The document exceeds the file limit.

Table 2: Detail of Use Case Upload Documents

2.3.3 Use case Specify Printing Properties

ID and Name	**UC-3 Specify Printing Properties**

Created By	Tran Hoang Phuc
Date Created	September 28th, 2024
Primary Actor	Student
Secondary Actors	None
Description	Student specifies printing properties for a document to be printed, involves paper size, pages (of the file) to be printed, one-/double-sided, number of copies, etc
Trigger	The student needs to print the document with their preferred configuration and the student clicks the **Edit Properties** button.
Preconditions	PRE-1: The student must be authenticated via the HCMUT_SSO authentication service. PRE-2: The student must have sufficient page balance in their account. PRE-3: The document has been chosen or uploaded and must be of a permitted file type.
Postconditions	POST-1: The document is configured with the specified properties. POST-2: The properties are displayed to the screen.
Normal Flow	1. The student has to be authenticated by HCMUT_SSO. 2. The student uploads documents to the storage. 3. The student selects the printing document option including paper size, pages (of the file) to be printed, one-/double-sided, number of copies and decides to choose to save the properties or not. 4. System validates if the page balance is sufficient. 5. Student confirms selected printing properties.
Alternative Flow	3.1. The student chooses printing properties that have been selected and saved before.
Exceptions	E2. The uploaded files are not valid in terms of size and file type. E4. Insufficient page balance.

Table 3: Detail of Use Case Identify Printing Properties

2.3.4 Use case Choose Printer

ID and Name	**UC-4 Choose Printer**
Created By	Tran Dang Hien Long
Date Created	September 28th, 2024
Primary Actor	Student
Secondary Actors	None
Description	The student selects a printer from the available printers connected to the system to proceed with printing the document.
Trigger	The student needs to select a printer to initiate the document printing process by clicking the **Choose Printer** button.
Preconditions	PRE-1: The student must be authenticated via the HCMUT_SSO authentication service. PRE-2: The student has to choose or upload documents to print. PRE-3: The student has to select the printing properties. PRE-4: At least one printer must be available and connected to the network. PRE-5: The printer must be enabled and ready to accept print jobs.
Postconditions	POST-1: The selected printer is assigned for the printing job. POST-2: The printer details are saved and linked to the printing job in the system.
Normal Flow	1. The student has to be authenticated by HCMUT_SSO. 2. The student uploads the document that needs to be printed. 3. The student chooses printing properties. 4. The system displays a list of available printers. 5. The student selects the desired printer from the list. 6. The system confirms the selection and updates the print job with the chosen printer. 7. The student proceeds to configure the print properties and finalize the printing process.
Alternative Flow	None
Exceptions	E4. If no printers are available, the system will notify the student and prompt them to try again later.

	E5. If the selected printer encounters an error (e.g., offline or malfunctioning), the system will prompt the student to choose a different printer.

Table 4: Detail of Use Case Choose Printer

2.3.5 Use case Pay More Printing Pages

ID and Name	**UC-5 Pay More Printing Pages**
Created By	Tran Nguyen Anh Khoa
Date Created	September 29th, 2024
Primary Actor	Student
Secondary Actors	Online Payment System
Description	The student purchases additional printing quota when their printing quora is insufficient.
Trigger	When a student's A4 paper provided by the university is insufficient and they want to buy more by clicking the **Buy More Page** button.
Preconditions	PRE-1: The student has to be authenticated by HCMUT_SSO. PRE-2: The student chooses an online payment system like BKpay or other similar systems. PRE-3: The student must have an existing page balance from their default allocation. PRE-4: The student's printing quota is insufficient.
Postconditions	POST-1: The student's printing quota is increased by the amount of A4 pages purchased. POST-2: The transaction is recorded in the system, including details of the purchases. POST-3: A payment confirmation is sent to the student, indicating the successful transaction. POST-4: The updated page balance is reflected in the student's account. POST-5: The payment log (history) will be updated and the student can view their printing history.
Normal Flow	1. The student has to be authenticated by HCMUT_SSO.

	2. The student selects the buy printing quota option. 3. The system displays the printing quota price. 4. The student selects the number of A4 papers to purchase. 5. The system processes the payment through the payment system (BKPay). 6. Upon successful payment, the system updates the student's printing quota. 7. The system sends a confirmation of the transaction to the student. 8. The system will save the details of the transaction and can be viewed by SPSO anytime.
Alternative Flow	6.1. The system processes the payment through VNPay.
Exceptions	E1: If the student is not authenticated, the system denies access to the purchase option. E5a: If the payment system encounters an error, the system notifies the student and suggests trying again later. E5b: If the student's account is inactive or blocked, the system informs them that they cannot make purchases until the issue is resolved. E5c: If the requested purchase amount exceeds the allowed quota, the system informs the student to adjust the number of pages they wish to buy.

Table 5: Detail of Use Case Pay For More Printing Pages

2.3.6 Use case Log Printing Actions

ID and Name	**UC-6 Log Printing Actions**
Created By	Tran Thien Phu
Date Created	September 28th, 2024
Primary Actor	Student
Secondary Actors	None
Description	After the printers successfully print the document, the system logs each printing event initiated by students, including the details of the document printed, the printer used, and the print properties specified.

Trigger	After the printer prints the documents successfully, prompting the system to log the action.
Preconditions	PRE-1: Student must be authenticated via the HCMUT_SSO authentication service PRE-2: The document is successfully uploaded and print properties are specified.
Postconditions	POST-1: The print action is recorded in the system log. POST-2: Relevant information (e.g., student ID, document name, printer ID, and timestamp) is saved for future reference.
Normal Flow	1. The print job is successfully sent to the printer. 2. The system records the details of the print job. 3. The system waits for the completion of the print job. 4. The system saves the print action in the log with a timestamp. 5. The student and HCMUT_SPSO can view the printing history.
Alternative Flow	None
Exceptions	E3. Job failed due to printer errors.

Table 6: Detail of Use Case Log Printing Actions

3. System Modelling

3.1 Activity Diagram

3.1.1 Upload Document

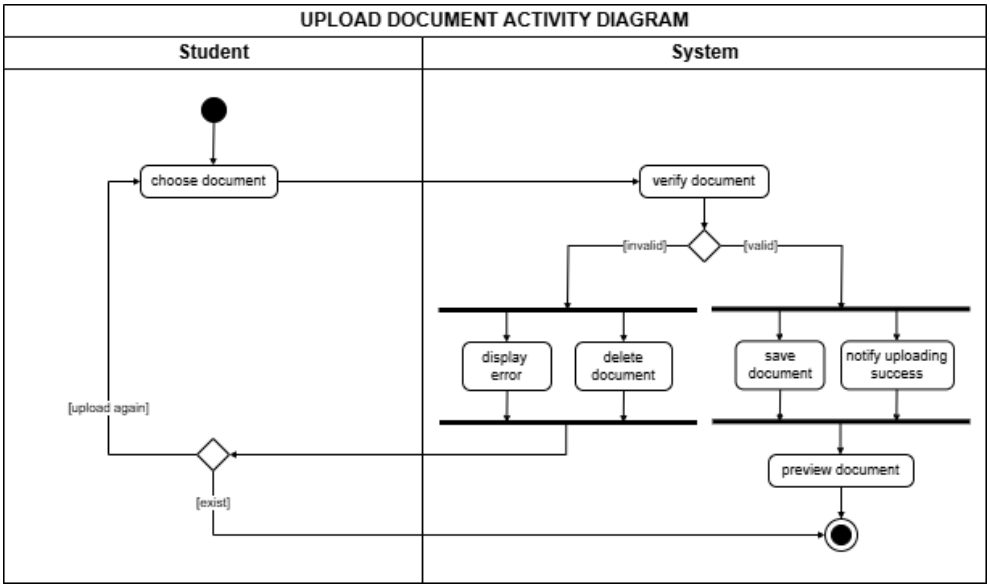

Figure 3: Activity Diagram of Upload Document Use Case

The activity diagram outlines the process of uploading a document by a student through a system. It begins with the student selecting a document to upload. Once chosen, the system verifies the document's validity. If the document is invalid, an error is displayed, and the document is deleted, prompting the student to upload a new document. If the document is valid, the system proceeds to save the document and notifies the student of the successful upload. Following the successful upload, the system offers the student a preview of the uploaded document. The process completes once the document is either successfully uploaded and saved, or an error occurs and the document must be re-uploaded.

3.1.2 Specify Printing Properties

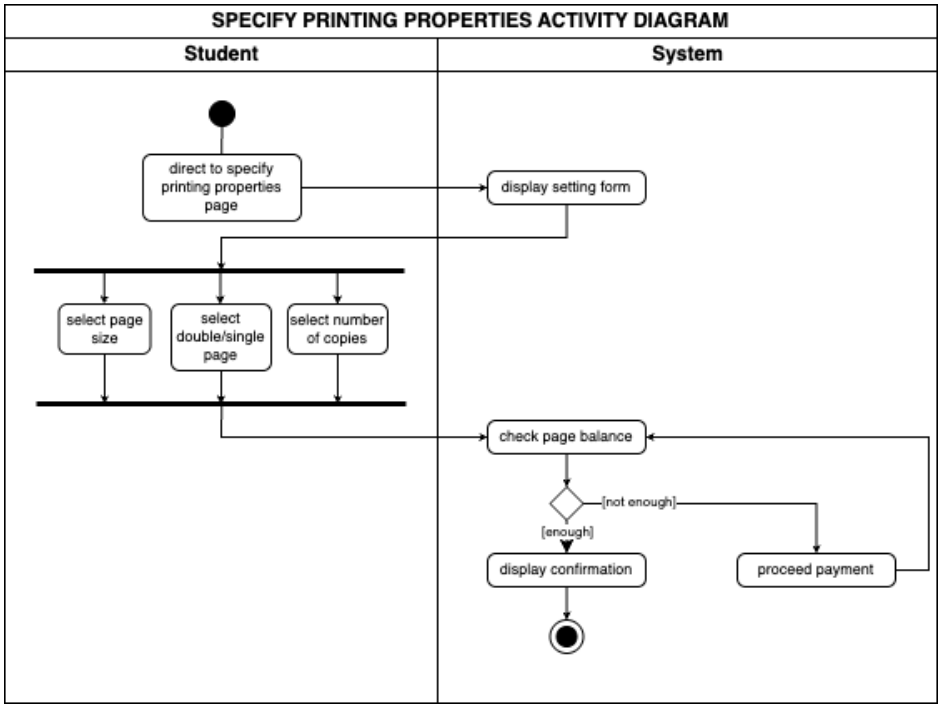

Figure 4: Activity Diagram of Specify Printing Properties Use Case

The activity diagram illustrates the process of specifying printing properties for a student using a system. The activity begins with the student being directed to a page where they can specify printing options. The system displays a form that allows the student to input these settings. The student then chooses from various printing settings, including page size, double or single-sided printing, and the number of copies. Once the selections are made, the system checks the student's page balance to determine enough pages to complete the printing request. If the balance is insufficient, the system prompts the student to proceed with payment. After payment is made, the system checks page balance again to guarantee that the student has enough pages to print. If the balance is sufficient, a confirmation is displayed to the student and the process is completed.

3.1.3 Buy Pages

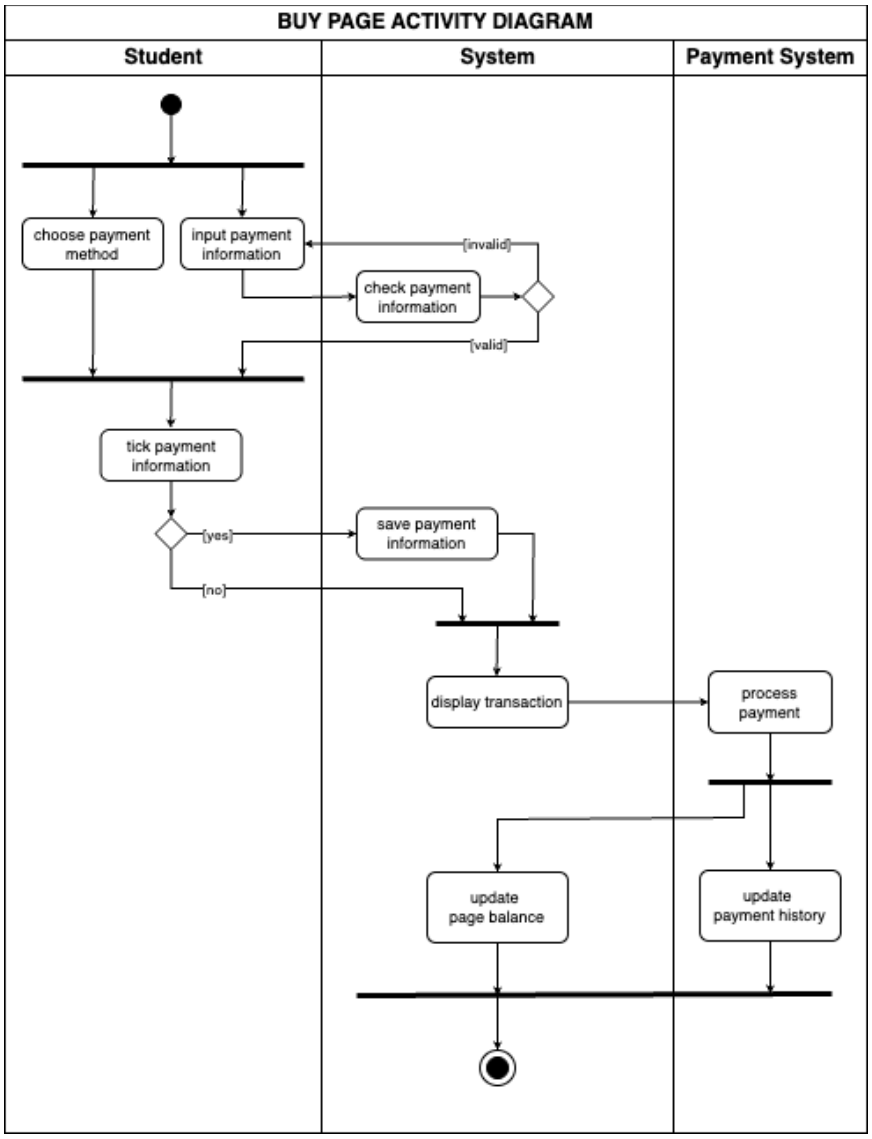

Figure 5: Activity Diagram of Buy Pages Use Case

The activity diagram represents the process of a student purchasing pages through a system that interfaces with a payment system. The process starts with the student choosing a payment method and inputting their payment information. The system then checks the validity of the payment details. If the information is invalid, the process terminates, requiring the student to input the correct details again. If valid, the student is prompted to confirm the payment information. Upon confirmation, the system saves the payment details and displays the transaction information. Once the transaction is successful, the system updates the student's page balance, reflecting the newly purchased credits and also updates payment history. The activity ends after the balance and the payment history are updated.

3.1.4 Choose Printer

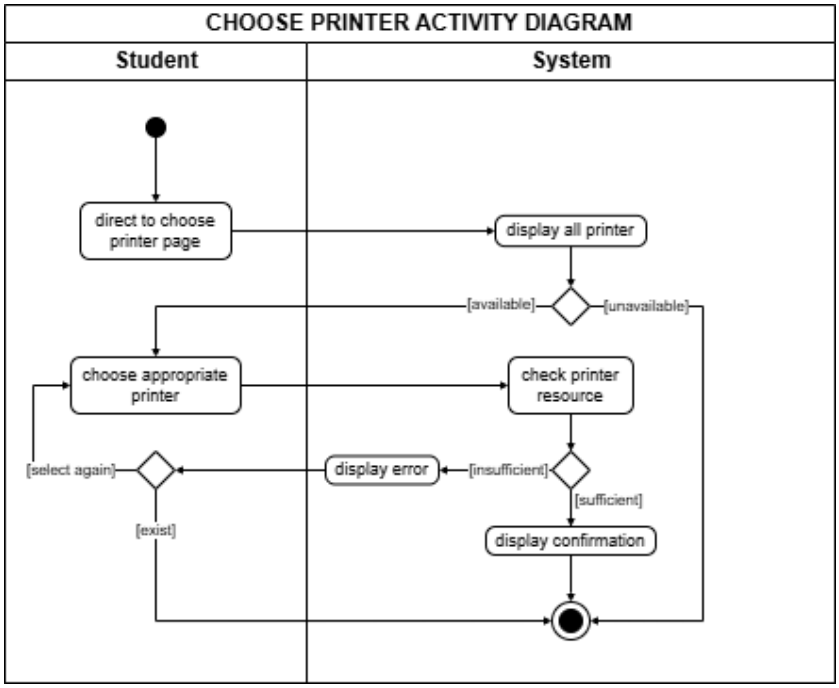

Figure 6: Activity Diagram of Choose Printer Use Case

The activity diagram outlines the process a student follows to choose an appropriate printer for their printing needs. The student is directed to the "Choose printer" page, where the system displays all available printers. The system checks the printer's availability, and if the printer is unavailable, the system will terminate the choice. Once an available printer is selected, the system checks the printer's resources to ensure it has sufficient capacity to handle the printing job. If the printer's resources are insufficient, an error is displayed, prompting the student to select again. However, if the printer has enough resources, a confirmation message is displayed, marking the completion of the process.

3.1.5 Print Document

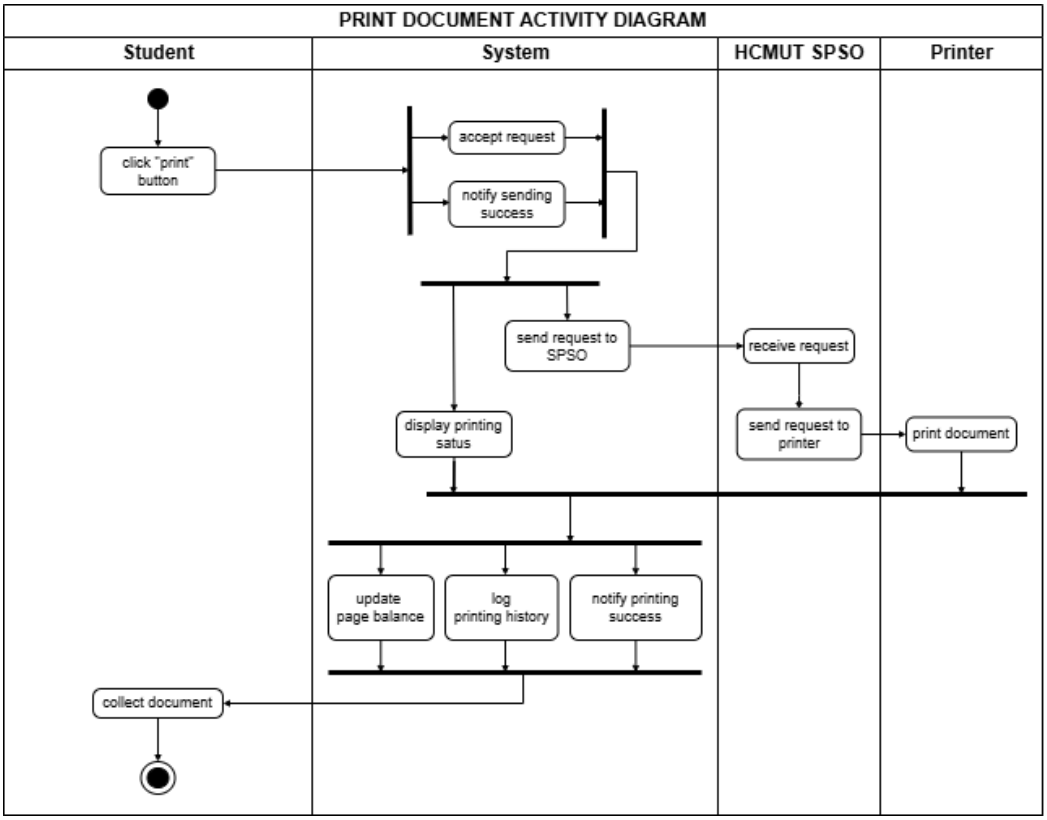

Figure 7: Activity Diagram of Print Document Use Case

The activity diagram outlines the process of printing a document initiated by a student. The student begins by clicking the "print" button, which sends a request to the system. The system accepts the request and notifies the student that the request was successfully sent. The system then forwards the request to the HCMUT SPSO, which receives the request and passes it to the designated printer. Once the printer receives the request, it proceeds to print the document. The system concurrently displays the printing status at regular intervals, allowing the student to track the progress of their print job. After printing successfully, the system updates the page balance for the student, logs the printing history, and notifies the student of the successful printing. The final step involves the student collecting the printed document.

3.2 Sequence Diagram

3.2.1 Upload Document

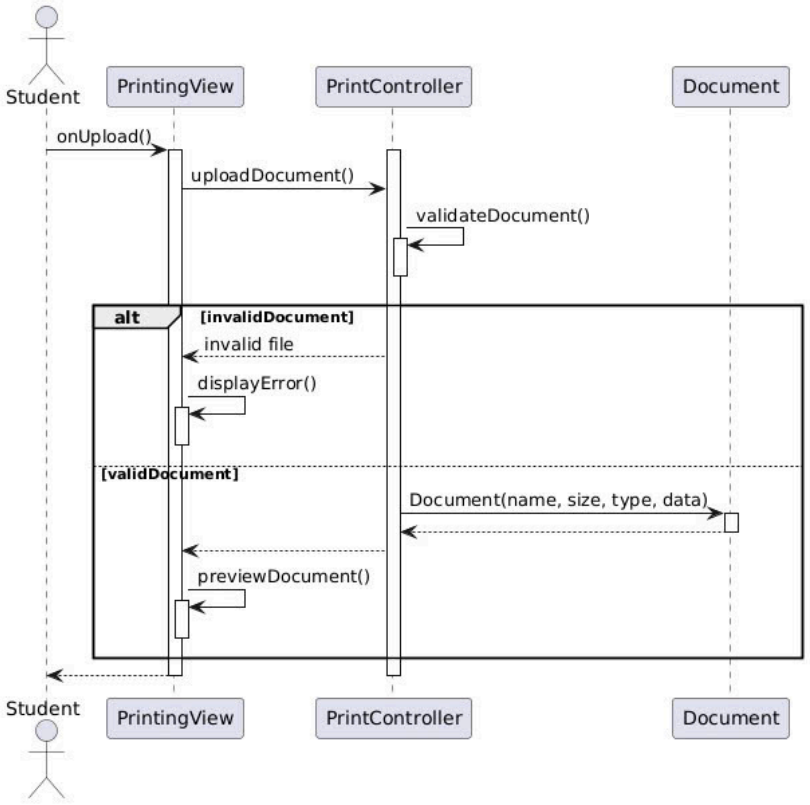

Figure 8: Sequence Diagram of Upload Document Use Case

The student triggers the upload action by interacting with the PrintingView via the onUpload() method. The PrintingView sends an uploadDocument() message to the PrintController, initiating the document upload process. The PrintController calls validateDocument() to ensure the uploaded document meets the required file type, size, and other constraints. This validation is handled by the system.

Conditional handling (alt block):

- *Invalid document path:* If the document is invalid (e.g., wrong format, type, size,...), the system marks the document as invalidDocument. The PrintController sends an invalid file response, and the PrintingView calls displayError() to show an error message to the student.
- *Valid document path:* If the document is valid, the system sends back a Document(name, size, type, data) response, and the PrintingView proceeds to call previewDocument() to display a preview of the document for the student to review.

3.2.2 Specify Printing Properties

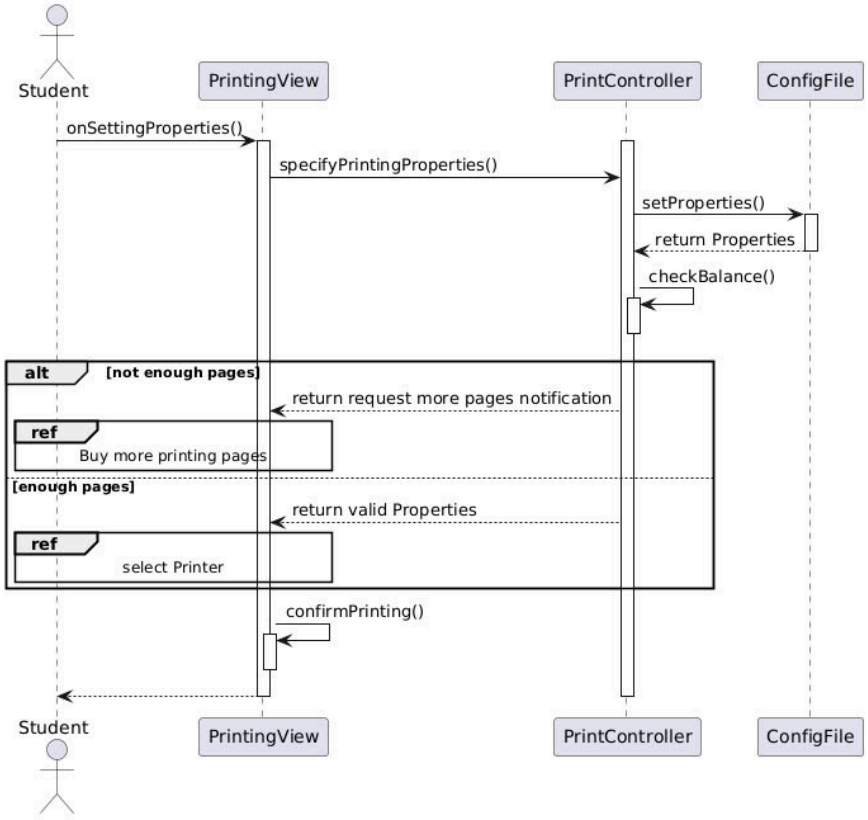

Figure 9: Sequence Diagram of Specify Printing Properties Use Case

The student triggers the process by interacting with the PrintingView via the onSettingProperties() method. The PrintingView sends a message specifyPrintingProperties() to the PrintController to capture the student's desired printing settings (e.g., page size, number of copies, single/double page). The PrintController calls setProperties() to configure the printing properties. It retrieves the appropriate configurations from the ConfigFile. The PrintController then invokes the checkBalance() method to verify whether the student has enough balance to print the document with the specified properties.

Conditional handling (alt block):

- *Not enough pages:* If the student does not have enough balance, the system returns a notification (return request more pages notification) prompting the student to purchase more pages. The student can then follow a reference (Buy more printing pages) to acquire additional quota.
- *Enough pages:* If the student has sufficient quota, the system returns the valid properties (return valid Properties). After the system validates the printing properties, the student proceeds to select the printer by following a reference (Select printer).

Finally, the system shows confirmation of printing properties by invoking confirmPrinting(), completing the process.

3.2.3 Buy Pages

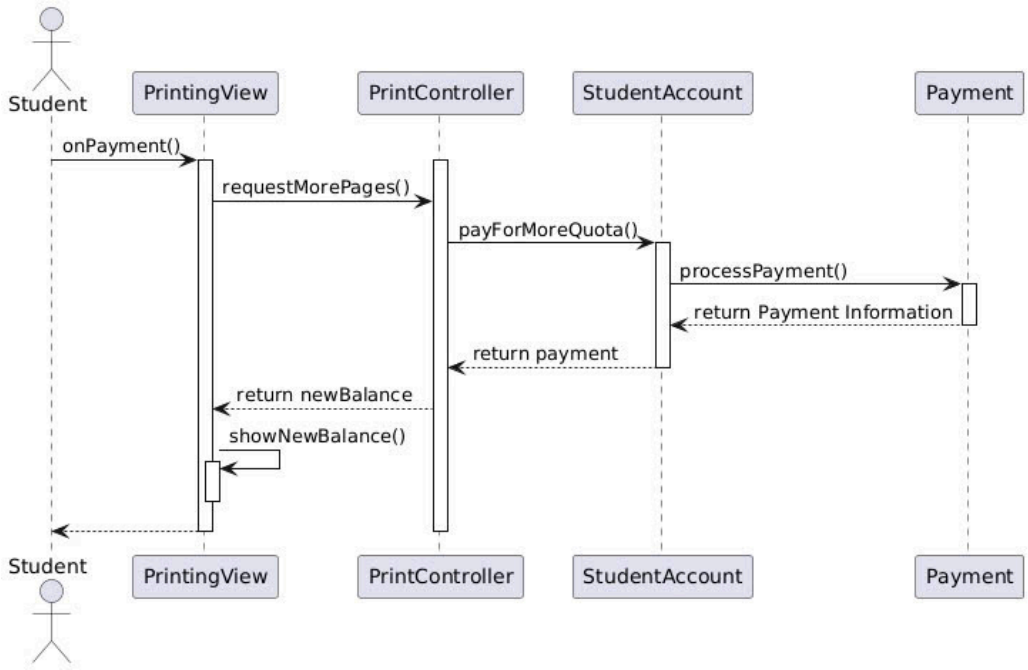

Figure 10: Sequence Diagram of Buy Pages Use Case

The student starts the process by triggering the onPayment() method from the PrintingView, indicating they want to pay for more printing pages. Then the PrintingView sends a requestMorePages() request to the PrintController. The PrintController forwards the request to the StudentAccount with a payForMoreQuota() message, initiating the payment for additional pages. The StudentAccount calls the processPayment() function from the Payment component to process the student's payment. The Payment component processes the payment and returns the payment information to the StudentAccount. The StudentAccount returns the payment result to the PrintController, updates and sends the new balance update to the PrintingView. The PrintingView then displays the updated balance to the student using the showNewBalance() method. The updated page balance is displayed to the student, and the transaction is completed.

3.2.4 Choose Printer

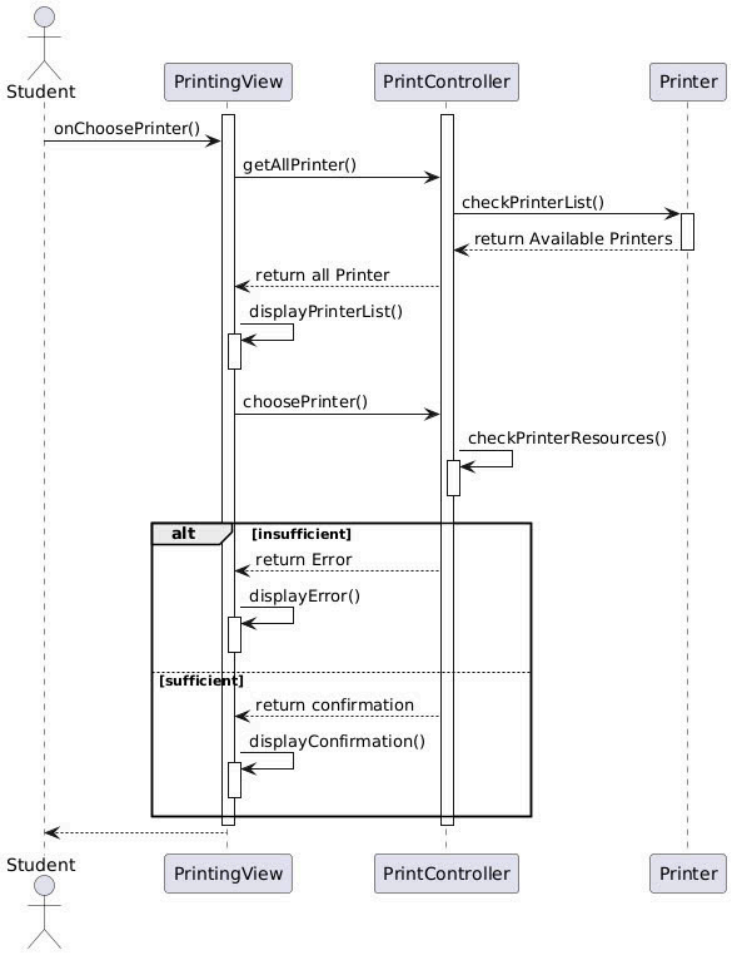

Figure 11: Sequence Diagram of Choose Printer Use Case

The student starts the process by interacting with the PrintingView via the onChoosePrinter() method, indicating they want to select a printer. The PrintingView calls the getAllPrinter() method to retrieve a list of available printers. The PrintController checks the printer list by calling checkPrinterList() from Printer and returns the list of available printers to controller. Then, the PrintingView displays the list of available printers to the student using displayPrinterList(). The student chooses a printer from the displayed list by calling the choosePrinter() method. The PrintController checks whether the chosen printer has sufficient resources (e.g., enough paper, not offline,..) by invoking checkPrinterResources().

Conditional paths (alt block):

- *Insufficient resources:* If the printer lacks the necessary resources, the system returns an error (return Error) and displays it to the student via the displayError() method.
- *Sufficient resources:* If the printer has enough resources, the system confirms the printer selection (return confirmation), and the confirmation is displayed to the student using displayConfirmation().

3.2.5 Print Document

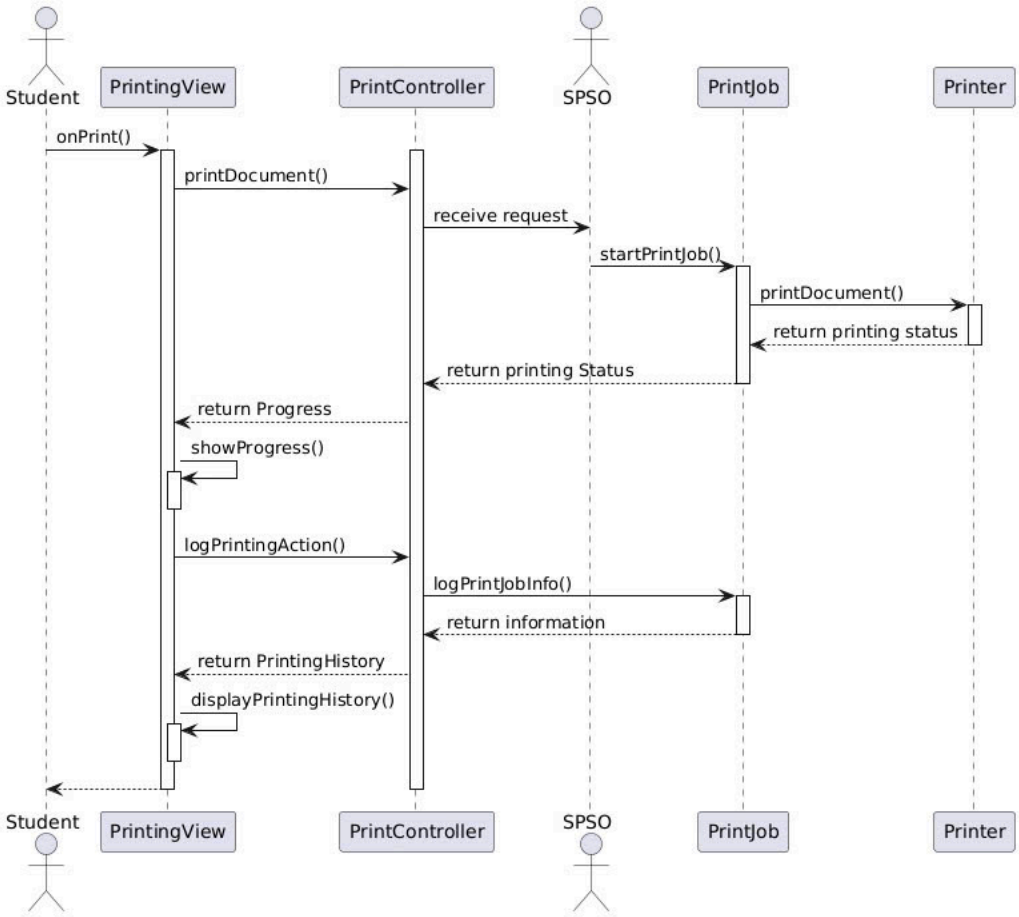

Figure 12: Sequence Diagram of Print Document Use Case

The student starts the process by triggering the onPrint() method from the PrintingView, indicating they want to print the document. The PrintingView sends a printDocument() request to the PrintController. After that, the PrintController forwards the request to the SPSO. The SPSO initiates the print job by calling startPrintJob() to begin the process of sending the document to the printer. The PrintJob receives the request and calls the printDocument() to print the document via the selected Printer. The PrintJob returns the current printing status of the document to the controller, indicating the progress of the print job. Next, the controller sends the progress update (return Progress) to the PrintingView. The PrintingView displays the progress to the student using the showProgress() method. After the document is printed, the PrintController logs the student's printing action with logPrintingAction(). The controller calls logPrintJobInfo() from PrintJob to get the printing information. The logged data is then returned to the PrintController with a return information message. Finally, the PrintController returns the student's printing history (return Printing History), which is displayed to the student via the displayPrintingHistory() method in the PrintingView.

3.3 Class Diagram

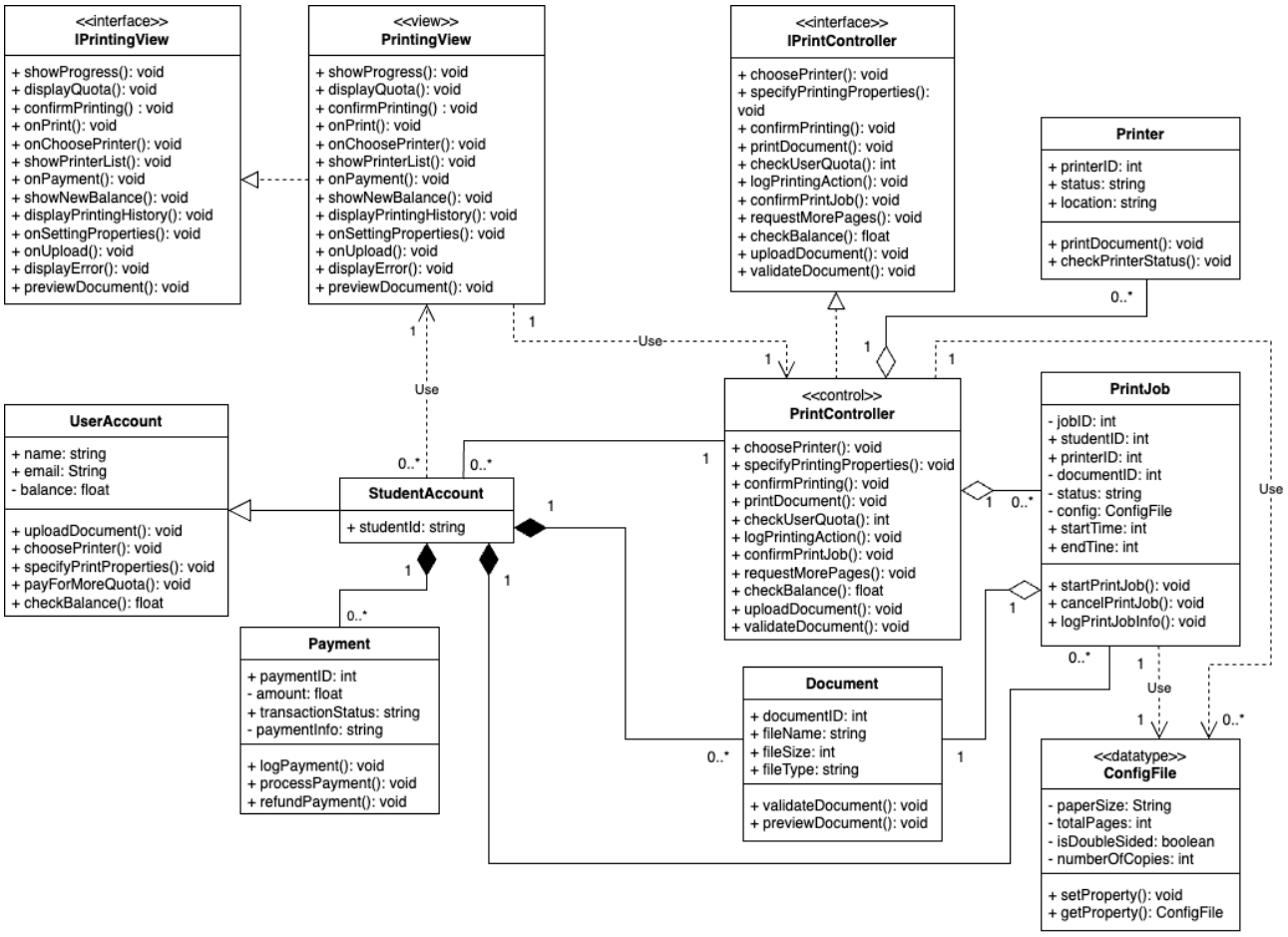

Figure 13: Class Diagram of Printing Document Module

The class diagram represents HCMUT_SSPS using the Model-View-Controller (MVC) pattern, with defined relationships between the key components. The diagram involves interfaces, controllers, views, and models that interact to manage the printing process for students.

First, the **inheritance relationship** is demonstrated between StudentAccount and UserAccount. StudentAccount inherits attributes and methods from UserAccount, such as the user's name, email, and balance. It also adds a new attribute, studentId, specific to student users.

The **realization relationship** connects PrintController and IPrintController, as well as PrintingView and IPrintingView. PrintController implements the methods defined in IPrintController, such as selecting printers and specifying print properties,… while PrintingView implements the interface IPrintingView, providing essential actions like displaying quotas, confirming printing, showing errors,…

The **aggregation relationship** is used in several places. PrintController aggregates multiple PrintJob and Printer objects. This means PrintController manages the lifecycle of PrintJob and

Printer, but they can still exist independently of the controller. A single PrintController can manage multiple print jobs and printers, but each job or printer belongs to one specific controller. Similarly, PrintJob also aggregates with Document to manage all documents uploaded by students.

The **composition relationship** illustrates a stronger connection. For instance, StudentAccount has a composition relationship with PrintJob, meaning that when a StudentAccount is deleted, its related PrintJob instances are also removed. Additionally, StudentAccount compositions exist with Payment and Document to ensure all related payments and documents are removed when the account is deleted.

The **dependency relationships** are shown between the controller and view components. PrintingView depends on the PrintController to fetch and display necessary data (e.g., showing the available printers or updating the quota after a print job). There is also a dependency between StudentAccount and PrintingView, where the student account uses the view to display the user's printing history and account balance. In addition, PrintController uses ConfigFile to manage and store print job settings that need to be applied when executing the printing task. The same is applied for PrintJob which accesses this configuration to ensure the printing follows the user's preferences. This ensures that every print job adheres to the specific settings provided by the user and stored in ConfigFile.

The one-to-many relationship exists between StudentAccount and Document, PrintJob, and Payment, indicating that each student account can be associated with multiple documents, print jobs, and payments, while each of these entities is linked to a single student account. Similarly, PrintController and PrintingView each have a one-to-many relationship with StudentAccount, allowing them to manage and display information for multiple student accounts. Furthermore, the PrintController has a one-to-many relationship with both ConfigFile and PrintJob, enabling the controller to manage multiple configurations and print jobs simultaneously. On the other hand, the Document and PrintJob entities are connected through a one-to-one relationship, ensuring that each document corresponds to a specific print job. Additionally, PrintJob also establishes a one-to-one relationship with ConfigFile, ensuring that each print job adheres to a unique set of configuration settings tailored to the user's preferences. These relationships collectively illustrate how the various components within the MVC architecture interact to facilitate the printing process efficiently and effectively.

3.4 User Interface

3.4.1 Student's Home Page

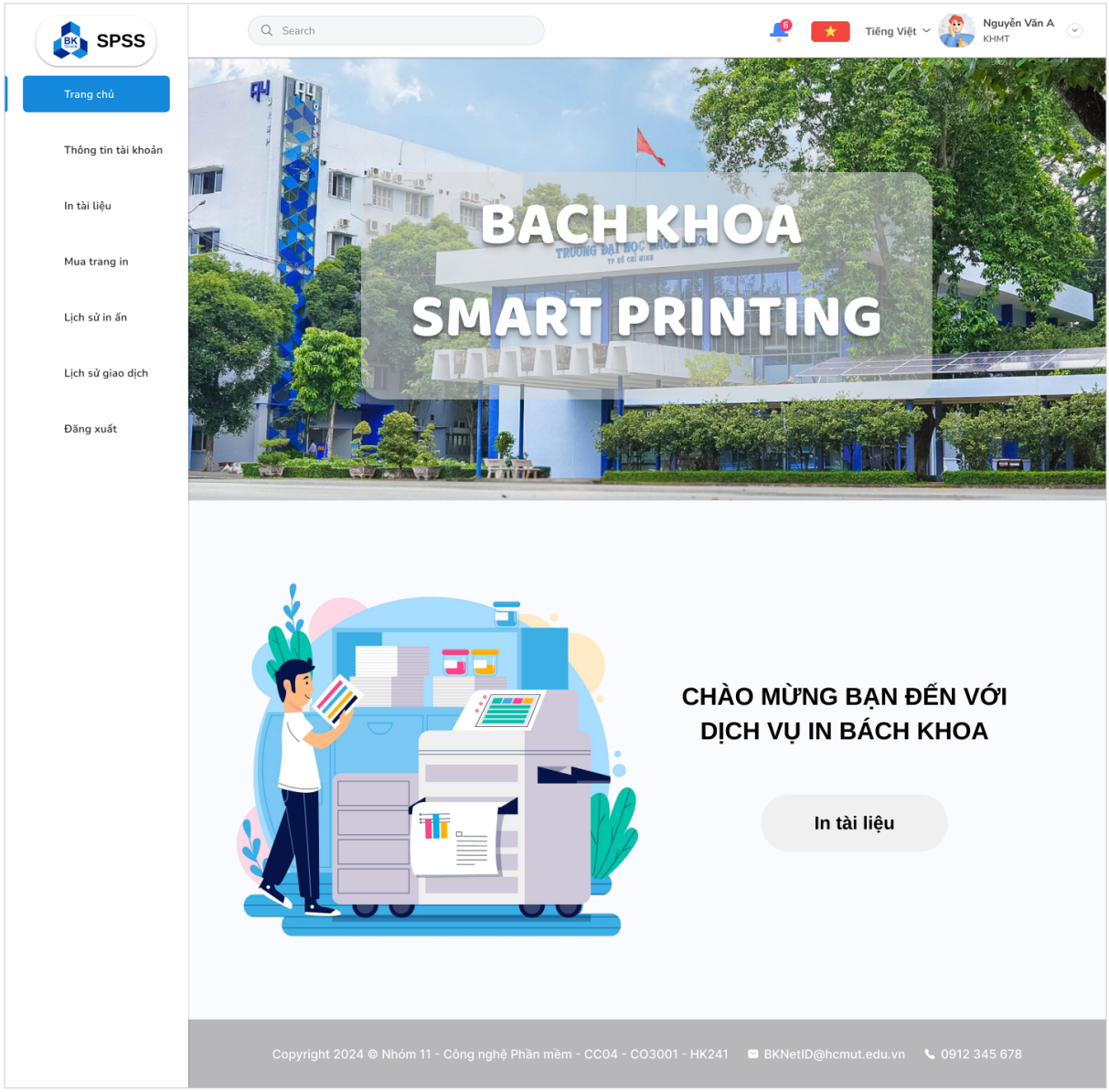

Figure 14: Student's Home Page (picture source: HCMUT)

3.4.2 Admin's Home Page

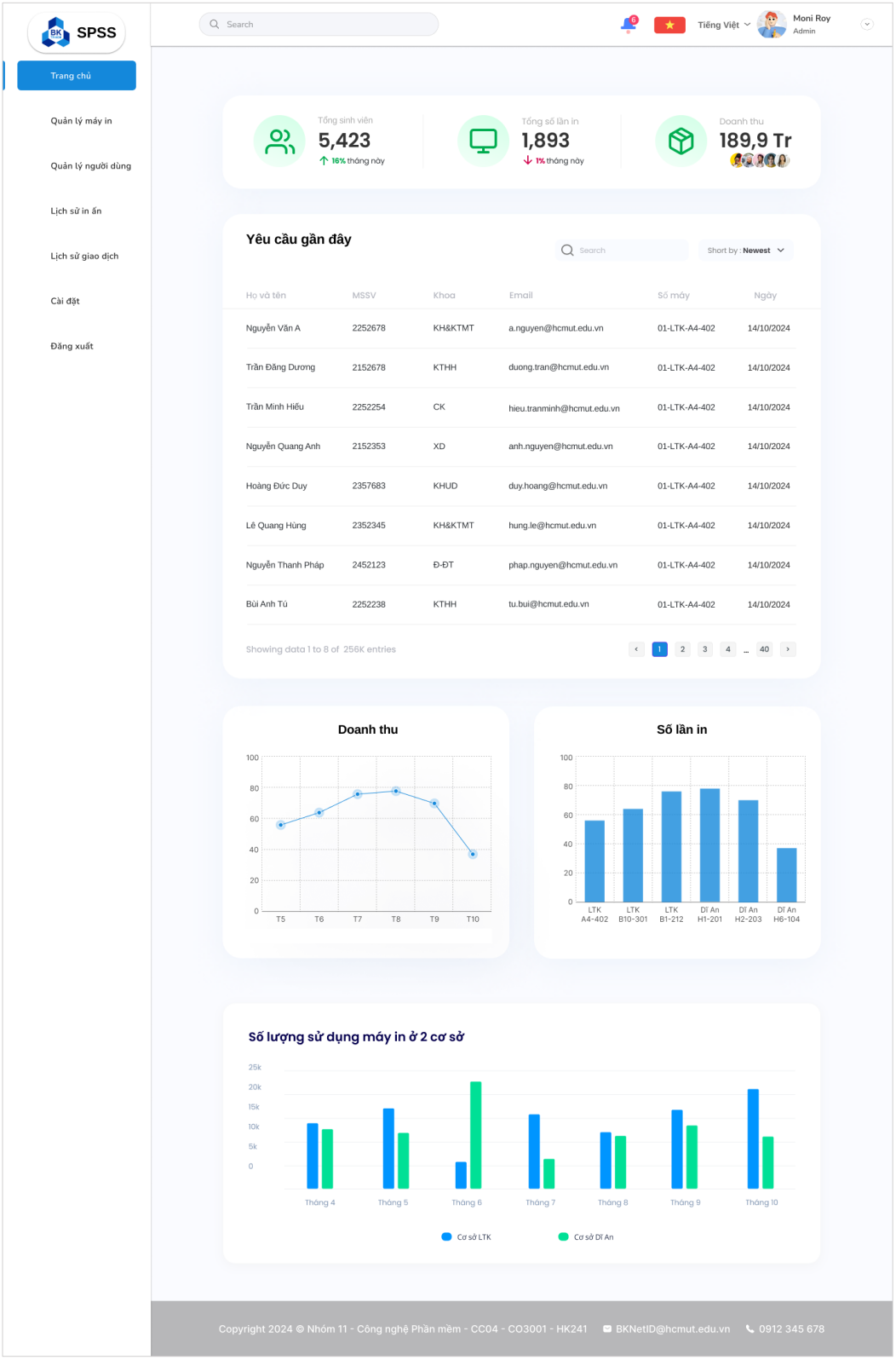

Figure 15: Admin's Home Page

3.4.3 Printing Module

3.4.3.1 Upload Document Page

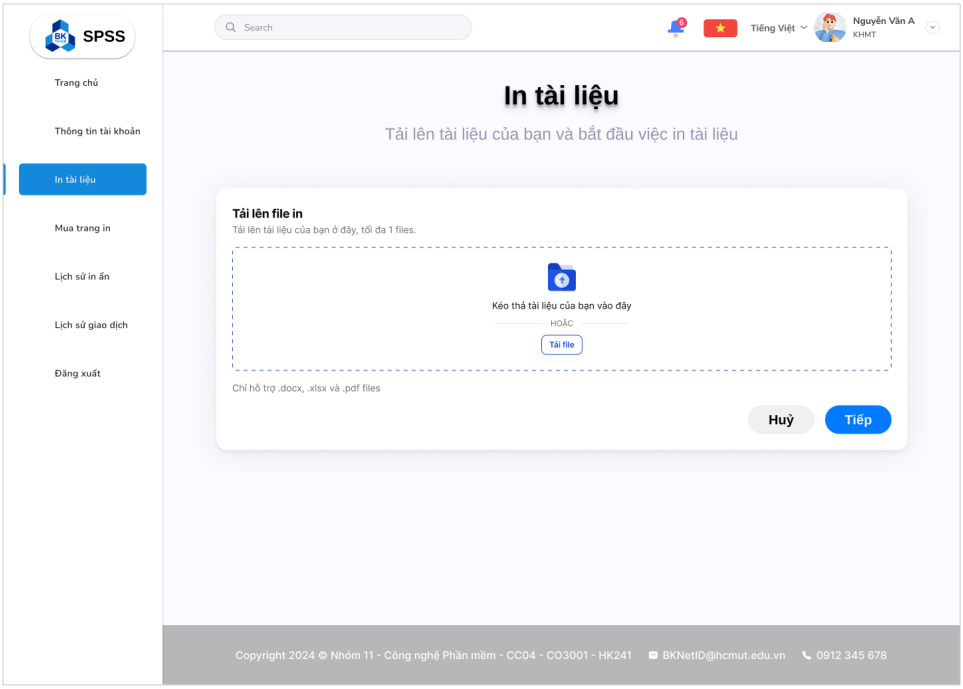

Figure 16: Upload Document

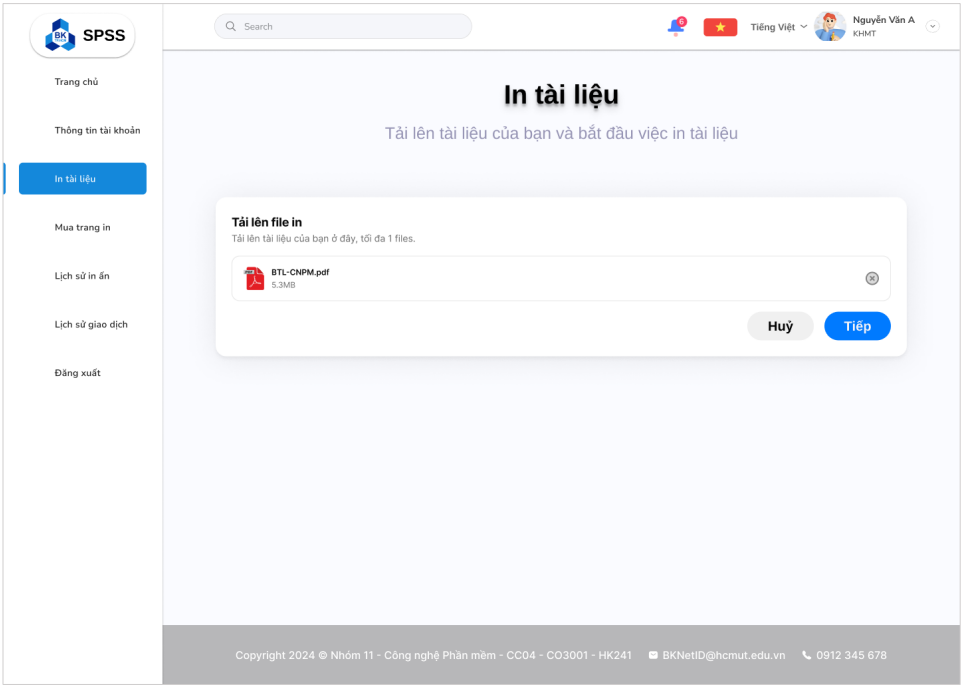

Figure 17: Document Preview

3.4.3.2 Specify Printing Propery Page

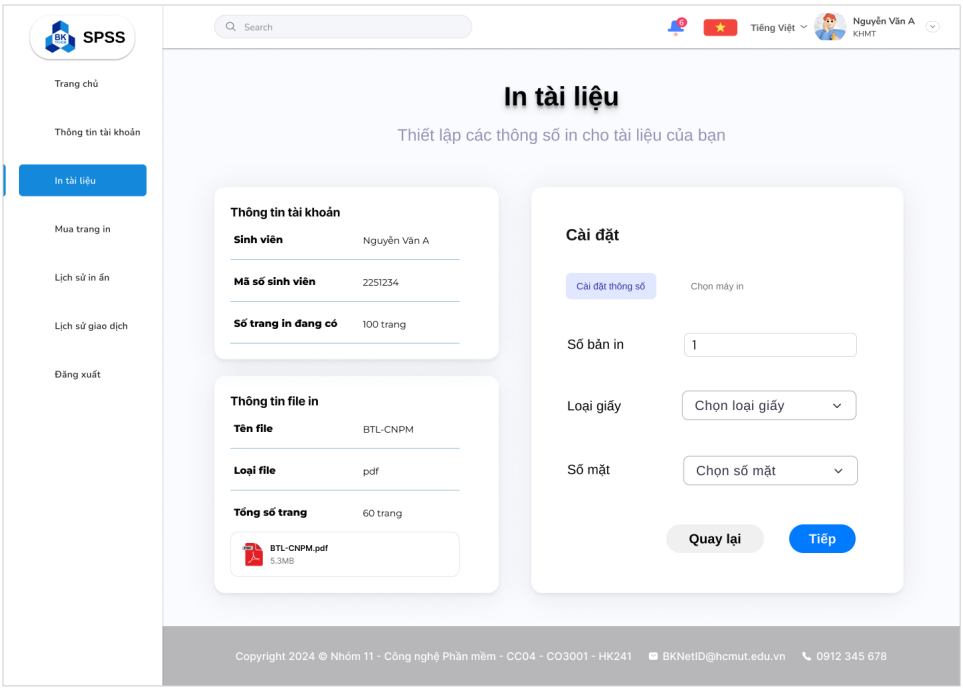

Figure 18: Specify Printing Property

3.4.3.3 Choose Printer Page

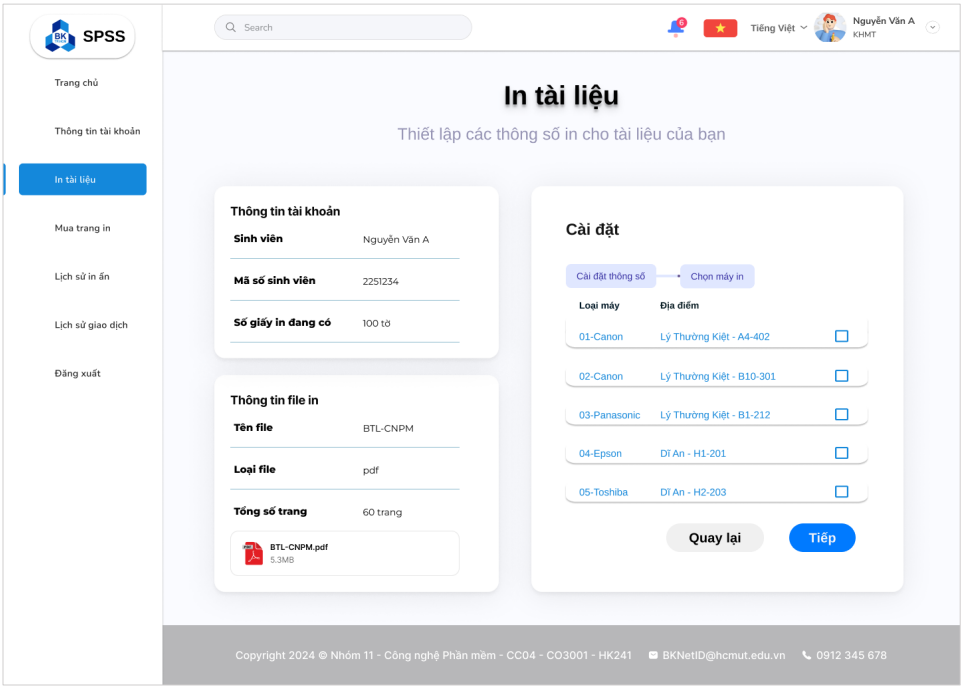

Figure 19: Choose Printer

3.4.3.4 Confirmation Printing Request Page

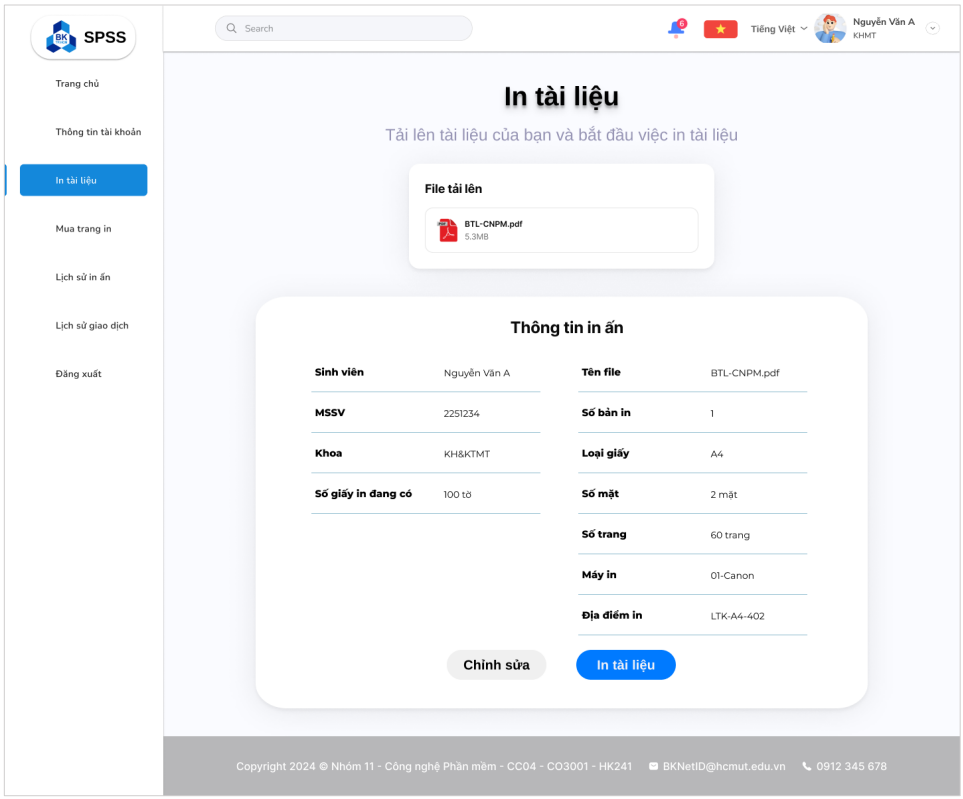

Figure 20: Printing Information Preview

3.4.3.5 Sending Request Successfully Page

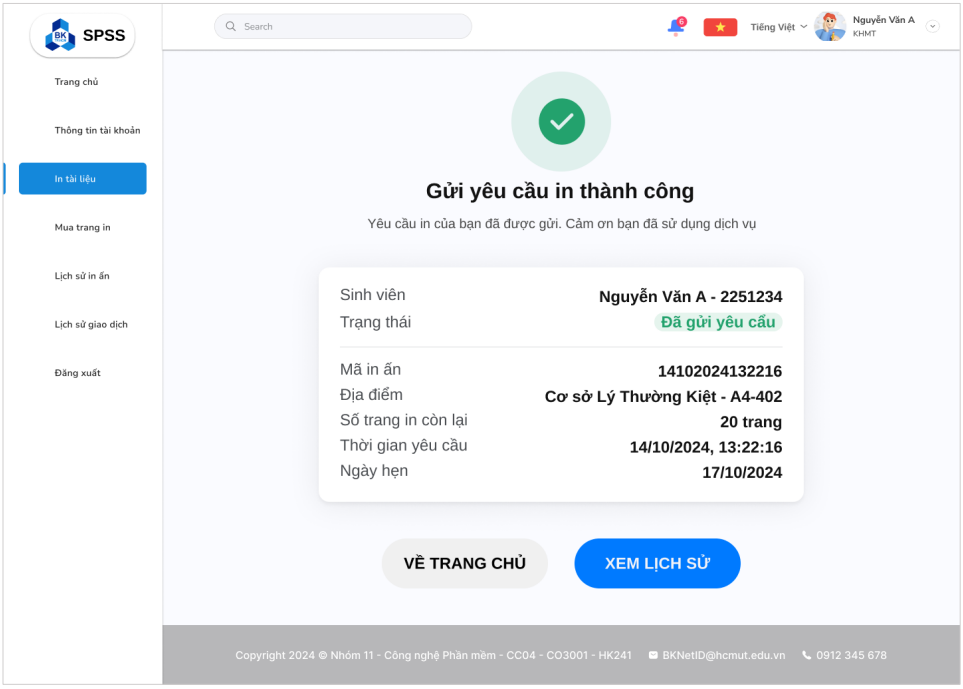

Figure 21: Sending Printing Request Successfully

3.4.3.6 Printing History Page

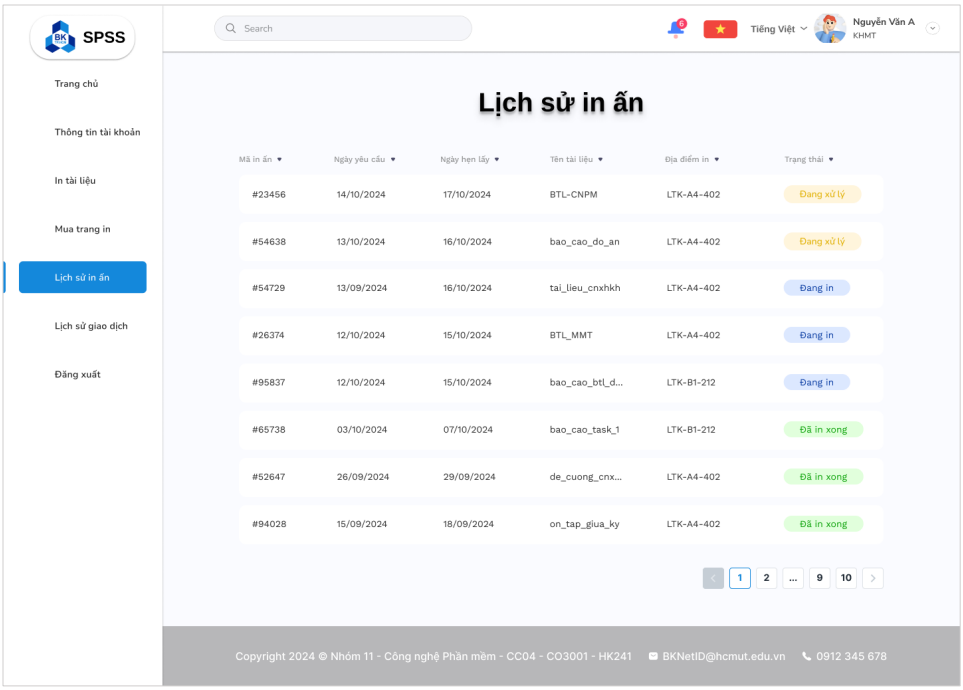

Figure 22: Printing History

3.4.3.7 Pay More Printing Balance Page

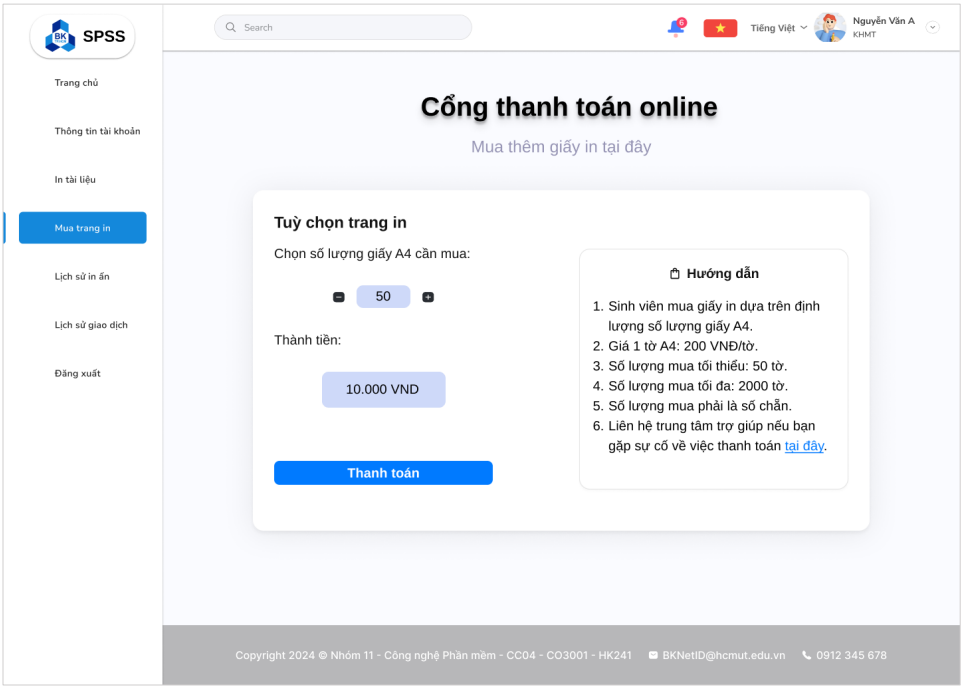

Figure 23: Choosing Number of Pages to Purchase

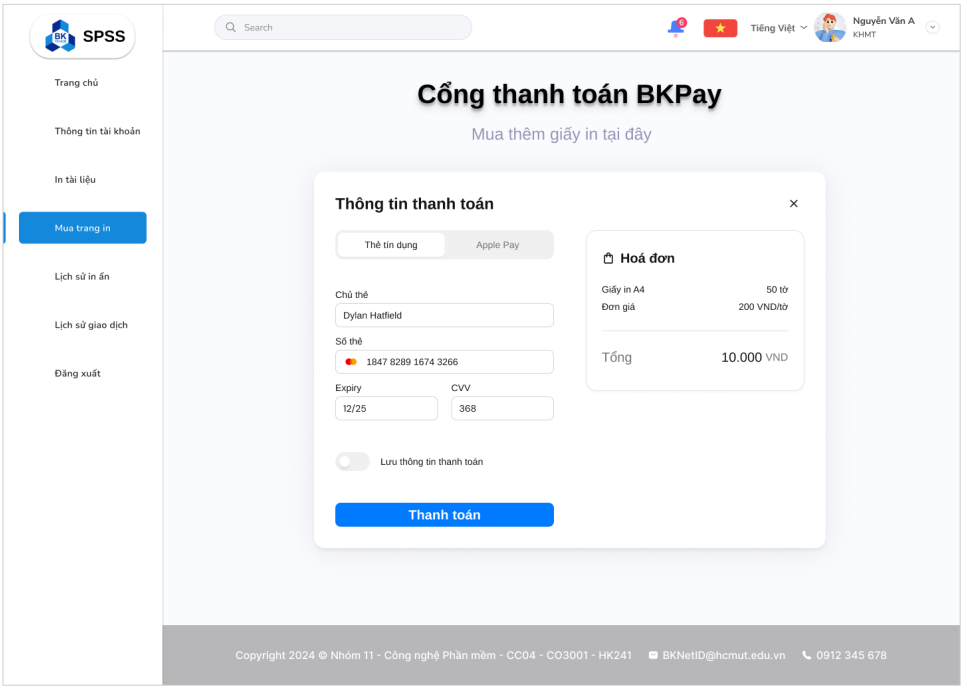

Figure 24: Payment Information Form

3.4.3.8 Payment Success Page

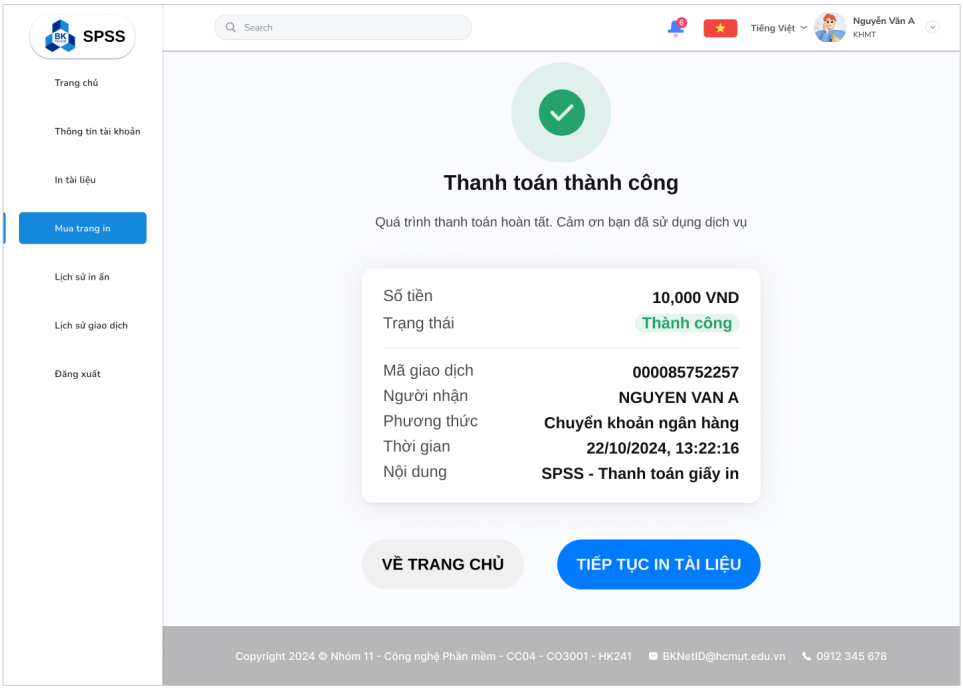

Figure 25: Payment Success

3.4.3.9 Payment History Page

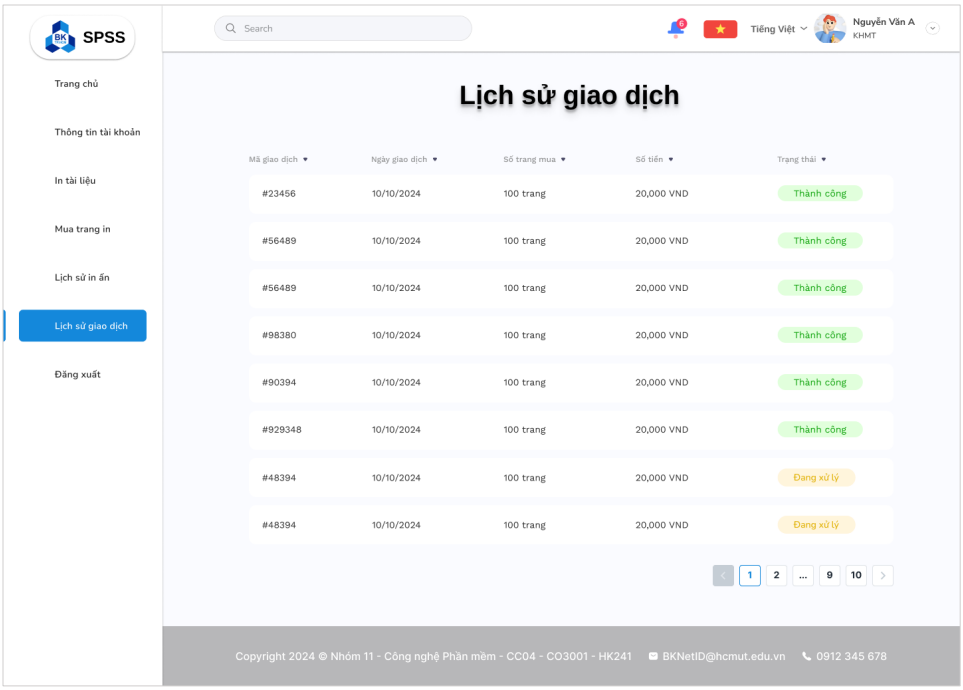

Figure 26: Payment History

4. Architecture Design

4.1 Architectural Diagram

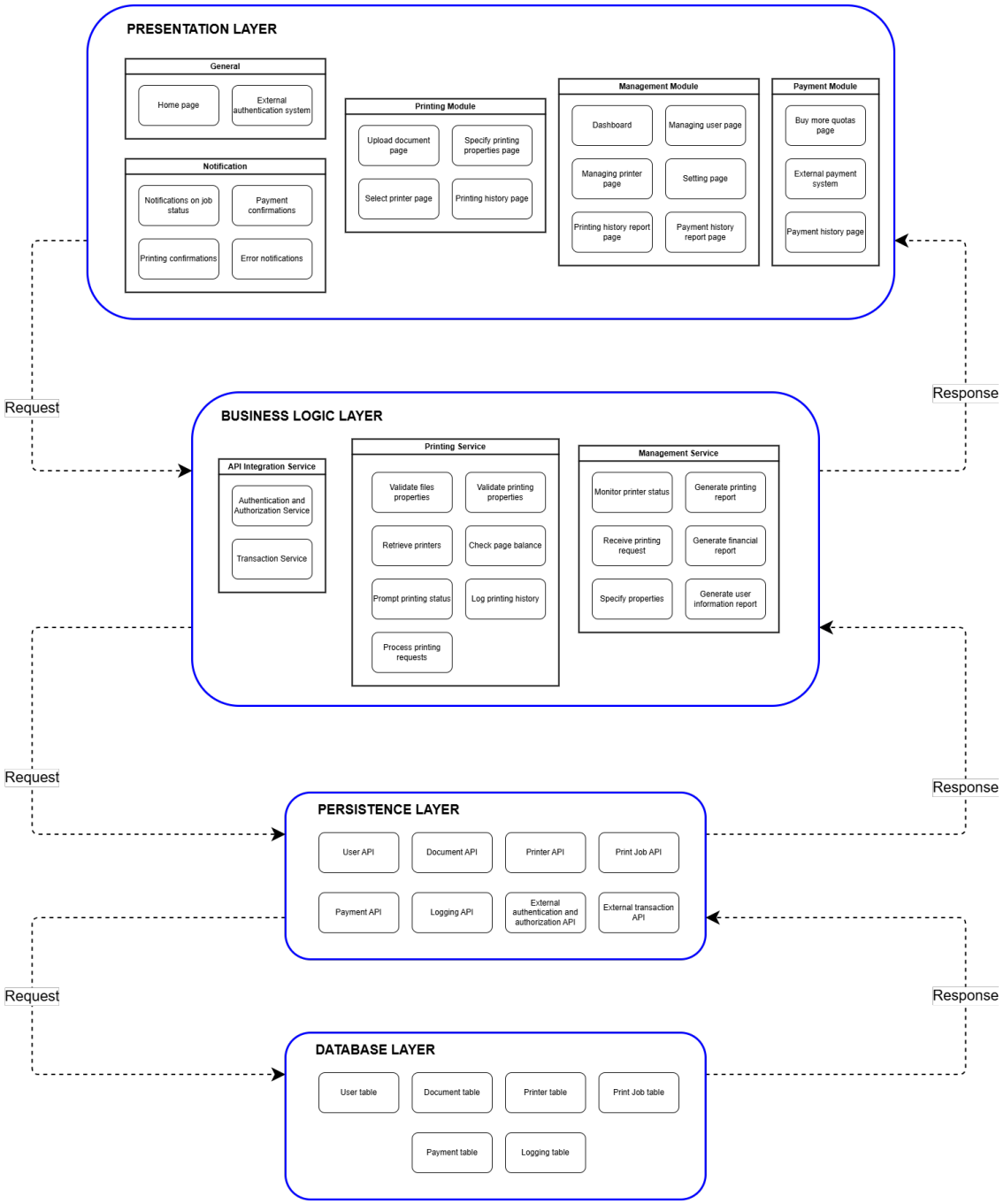

Figure 27: Architectural Diagram of Printing Module

4.1.1 Presentation Strategy

The presentation layer of this software system is organized into several modules, each designed to handle specific user-facing functionalities within the application. This layer is structured to deliver a seamless user experience by organizing components logically according to their functionalities. Here's a breakdown of each module and its purpose:

4.1.1.1 General Module

Contains the essential entry points and authentication mechanisms for the application.

Components:

- Home Page: The main landing page of the application.
- External Authentication System: Integrates with external authentication providers to allow secure user login.

4.1.1.2 Notification Module

Manages notifications related to various system events, providing users with real-time updates on their actions and system status.

Components:

- Notifications on Job Status: Alerts users about the current status of their jobs.
- Payment Confirmations: Notifies users upon successful payments.
- Printing Confirmations: Provides confirmation after print jobs are completed.
- Error Notifications: Alerts users about any errors encountered during processing.

4.1.1.3 Printing Module

The core module dedicated to managing the printing functionality, guiding users through document upload to specifying print properties.

Components:

- Upload Document Page: Allows users to upload documents for printing.
- Specify Printing Properties Page: Enables users to set specific printing options for their documents.
- Select Printer Page: Allows users to choose a printer for their job.
- Printing History Page: Displays the history of previous print jobs.

4.1.1.4 Management Module:

Provides administrative tools for managing various aspects of the application, including user and printer settings.

Components:

- Dashboard: A centralized interface for monitoring key metrics and data.

- Managing User Page: Manages user profiles and access permissions.
- Managing Printer Page: Administers printer settings and availability.
- Setting Page: Allows customization of various application settings.
- Printing History Report Page: Generates reports based on the printing history.
- Payment History Report Page: Provides a record of past payments.

4.1.1.5 Payment Module:

Manages the purchasing and payment processes within the application.

Components:

- Buy More Quotas Page: Allows users to purchase additional print quotas.
- External Payment System: Integrates with external payment processors for secure transactions.
- Payment History Page: Shows a history of completed payments.

4.1.2 Data Storage Approach

4.1.2.1 User table

Attribute	Type	Description
UserID	String	Unique identifier for the user.
Username	String	Full name of the user.
Email	String	Email address of the user.
RemainingPages	Integer	Count of remaining pages the user can print.

The User contains basic information about the users of the system (students or other personnel). The key interaction points for the User collection are:

- Documents: A User uploads documents, which are stored in the Document collection. The UserID is used as a foreign key in the Document collection to link the uploaded documents to the user.
- Print Jobs: A User initiates print jobs, which are recorded in the Print Job collection. The UserID is used to associate a print job with the user who initiated it.
- Payments: A User can make payments for additional pages, which are recorded in the Payment collection. The UserID is used to track which user made the payment.
- Logging: Every action performed by the User is recorded in the Logging collection, such as document uploads, print job executions, and payment actions. The UserID links the log entry to the specific user.

4.1.2.2 Document table

Attribute	Type	Description
DocumentID	String	Unique identifier for the document.
UserID	String	ID of the user who uploaded the document (foreign key).
Filename	String	Name of the document file.
FileType	String	File format of the document (e.g., PDF, DOCX).
FileSizeMB	Float	Size of the document in MB.

The Document stores details about documents uploaded by users. Each document has the following interactions:

- User: Each document is uploaded by a User, and the UserID (foreign key) associates the document with that user.
- Print Jobs: When a User initiates a print job, the DocumentID is used to link the document being printed with the specific print job. This helps track which document was printed, how many pages were printed, and where the document was printed.

4.1.2.3 Printer table

Attribute	Type	Description
PrinterID	String	Unique identifier for the printer.
PrinterBrand	String	Brand name of the printer.
PrinterName	String	Model name of the printer.
Location	Object	Embedded object detailing the printer's location.
Location.Facility	String	Facility where the printer is located.
Location.Building	String	Building where the printer is located.
Location.Room	String	Room number where the printer is located.
Status	Boolean	Printer's operational status (enabled/disabled).
PrintedPages	Integer	Total number of pages printed by this printer.

The Printer stores information about the printers available in the system. The interactions for the Printer collection include:

- Print Jobs: Each Print Job references a PrinterID, which identifies the specific printer used to print the document. This allows tracking of which printer was used for each print job.
- Status and Printed Pages: The Status of a printer (enabled/disabled) and the PrintedPages count help the system monitor printer availability and usage.

4.1.2.4 PrintJob table

Attribute	Type	Description
PrintJobID	String	Unique identifier for each print job.
UserID	String	ID of the user initiating the print job (foreign key).
DocumentID	String	ID of the document being printed (foreign key).
PrinterID	String	ID of the printer used (foreign key).
PrintingTime	Date	Date and time when the print job was executed.
PrintedPages	Integer	Number of pages printed in this job.
PaperType	String	Type of paper used for printing (e.g., A4, A3).

The Print Job records every print job initiated by users. Each print job contains the following interactions:

- User: Each print job is linked to a User by the UserID, indicating which user initiated the job.
- Document: Each print job references the document being printed through the DocumentID, helping track which document was printed.
- Printer: Each print job references the PrinterID to specify which printer was used for the print job. This interaction also helps to update the PrintedPages count on the corresponding printer.
- Logging: Print jobs are logged in the Logging collection to track the action, success, or failure, as well as the timestamp and any other relevant details.

4.1.2.5 Payment table

Attribute	Type	Description
PaymentID	String	Unique identifier for each payment transaction.
UserID	String	ID of the user making the payment (foreign key).

Time	Date	Date and time when the payment was made.
Method	String	Method used to pay for additional pages (e.g., credit card, PayPal).
Amount	Float	Total amount paid for purchasing additional pages.
PurchasedPages	Integer	Number of additional pages purchased in the transaction.

The Payment records payments made by users for purchasing additional printing pages. The interactions for the Payment collection are as follows:

- User: Each payment is linked to a User through the UserID, showing which user made the payment for extra pages.
- Logging: Payment transactions are logged in the Logging collection to track when and how much the user paid, and the status of the payment.

4.1.2.6 Logging table

Attribute	Type	Description
LogID	String	Unique identifier for each log entry.
UserID	String	ID of the user associated with the log entry (foreign key).
ActionType	String	Type of action performed (e.g., print, payment, upload).
Description	String	Detailed description of the action performed.
Timestamp	Date	Date and time when the action was logged.
Status	String	Status of the action (e.g., successful, failed, pending).

The Logging tracks various actions performed by users and the system, allowing for auditing and monitoring. These are the main interactions for the Logging collection:

- User: The UserID associates each log entry with a specific user, helping identify who performed the action (e.g., printing a document, making a payment).
- ActionType and Description: The ActionType describes what action was taken (e.g., document upload, print job initiation, payment) and the Description provides more details about the action.
- Timestamp and Status: The Timestamp captures when the action occurred, and the Status indicates whether the action was successful, failed, or still pending.

4.1.3 API Management

The Persistence Layer in this system handles communication between the Business Logic Layer and the Database Layer via APIs, providing standardized interfaces for accessing and manipulating data. Below are descriptions of each API within this layer

4.1.3.1 User API

Purpose: Manages user-related information in the database, supporting authentication, authorization, and user profile management.

Methods:

Method	Return Type	Description
getUserDetails(userID: String)	User	Retrieves user information (e.g., name, email, role) for a specified userID.
updateUserInfo(userID: String, userInfo: Object)	Boolean	Updates specific details in the user's profile (e.g., email or faculty), based on the provided userInfo object.
getUserPermissions(userID: String)	List<String>	Retrieves a list of permissions associated with the user to verify access rights for various system modules.
getUserInfo(userID: int)	UserAccount	Retrieves the account details (name, email, balance) of a specified user.
addUser(user: UserAccount)	int	Adds a new user to the system, returning the unique userID for the user.
updateUserBalance(userID: int, balance: float)	Boolean	Adjusts the user's balance, typically after a payment or quota purchase.
deleteUserAccount(userID: int)	Boolean	Deletes a user account from the system

Use Cases:

- Used during login to fetch user details and permissions.
- Called when administrators update user profiles or permissions.

4.1.3.2 Document API

Purpose: Handles storage, retrieval, and management of document data for printing operations.

Methods:

Method	Return Type	Description
getDocumentDetails(docID: String)	Document	Retrieves metadata (e.g., file name, type, size) for a document identified by docID.
storeDocument(document: Object)	String	Saves a new document in the database and returns a unique document ID.
deleteDocument(docID: String)	Boolean	Deletes a document after it has been printed or reaches a specified retention period.
updateDocument(docID: String, documentData: Object)	Boolean	Updates the properties of an existing document, such as file name or print settings.

Use Cases:

- Called when a user uploads a document for printing.
- Allows updates or deletions based on retention policies or user actions.
- Enables administrators to retrieve document details for auditing or troubleshooting.

4.1.3.3 Printer API

Purpose: Provides access to printer information, including status, location, and configuration settings.

Methods:

Method	Return Type	Description
getPrinterStatus(printerID: String)	String	Returns the current status of a printer (e.g., "Available", "Busy", "Out of Order").
getPrinterLocation(printerID: String)	String	Retrieves the building and room information for a specific printer.

updatePrinterConfig(printerID: String, config: Object)	Boolean	Updates printer configuration details such as paper type, default print quality, and operational status.
getAvailablePrinters()	List<String>	Retrieves a list of all available printers, optionally filtered by location or printer type.
resetPrinter(printerID: String)	Boolean	Resets a printer's status to "Available" in case of error or failure.

Use Cases:

- Used by students to check printer availability before sending print jobs.
- Called by administrators to monitor printer locations and configurations.

4.1.3.4 Print Job API

Purpose: Manages print job lifecycle, from submission to completion, and provides status tracking.

Methods:

Method	Return Type	Description
submitPrintJob(jobDetails: Object)	String	Submits a new print job with details such as userID, docID, and printerID, and returns a unique job ID.
getPrintJobStatus(jobID: String)	String	Retrieves the current status of a specific print job (e.g., "In Queue", "Printing", "Completed").
cancelPrintJob(jobID: String)	Boolean	Cancels a print job if it has not yet started, freeing up resources.
updatePrintJob(jobID: String, newDetails: Object)	Boolean	Updates details of a print job, such as changing the printer or priority level.

getPrintHistory(userID: String)	List<PrintJob>	Retrieves the printing history for a specific user, showing details of past print jobs.

Use Cases:

- Used by the Printing Module when a user initiates a print request.
- Allows students or administrators to check job status or cancel jobs if needed.

4.1.3.5 Payment API

Purpose: Facilitates payment processing for students to purchase additional printing quotas.

Methods:

Method	Return Type	Description
processPayment(userID: String, amount: Double)	TransactionID	Processes a payment transaction and returns a TransactionID upon success.
getPaymentHistory(userID: String)	List<Transaction>	Retrieves the history of all payment transactions made by a user.
refundTransaction(transactionID: String)	Boolean	Initiates a refund for a specified transaction if applicable.
verifyPayment(transactionID: String)	Boolean	Verifies if a payment was processed successfully.

Use Cases:

- Called when students purchase additional pages or quotas.
- Administrators can review payment history or process refunds when required.

4.1.3.6 Logging API

Purpose: Records system activities and user actions for tracking and auditing purposes.

Methods:

Method	Return Type	Description
logActivity(userID: String, action: String, details: Object)	Boolean	Logs a specific user action or system event, including details such as timestamps and action type.
getActivityLogs(filterParams: Object)	List<LogEntry>	Retrieves logs based on filter criteria (e.g., by userID, date range, or action type).
deleteOldLogs(retentionPeriod: Integer)	Boolean	Periodically deletes logs older than a specified retention period.

Use Cases:

- Called whenever a significant user action occurs, such as document printing or system changes.
- Allows administrators to access activity logs for security and troubleshooting.

4.1.3.7 External Authentication and Authorization API

Purpose: Integrates with an external authentication system to validate user credentials and retrieve user roles.

Methods:

Method	Return Type	Description
validateUserCredentials(username: String, password: String)	Boolean	Authenticates user login credentials with an external system.
fetchUserRoles(userID: String)	List<String>	Retrieves roles and permissions for a user to enable role-based access control.
resetPassword(userID: String)	Boolean	Initiates a password reset for a user via the external authentication system.

Use Cases:

- Called during login to authenticate and authorize users, particularly for those with roles requiring restricted access.

4.1.3.8 External Transaction API

Purpose: Interfaces with an external payment service to handle secure transactions for purchasing additional pages or printing services.

Methods:

Method	Return Type	Description
initiateTransaction(userID: String, amount: Double)	TransactionID	Starts a new transaction and returns a TransactionID for tracking.
validateTransaction(transactionID: String)	Boolean	Confirms the transaction's success with the payment service before granting additional printing pages.
getTransactionStatus(transactionID: String)	String	Checks the current status of a payment (e.g., "Pending", "Completed", "Failed").
cancelTransaction(transactionID: String)	Boolean	Cancels a transaction if it's still pending.

Use Cases:

- Used by the Payment Module to securely handle transactions for purchasing print credits.
- Allows the system to verify and record completed transactions before updating user quotas.
- Each API in the Persistence Layer serves as a bridge between the Business Logic Layer and the Database Layer, ensuring modular and secure interactions with the underlying data. These APIs allow for clear separation of functionalities, making the system scalable and easier to maintain.

4.2 Component Diagram

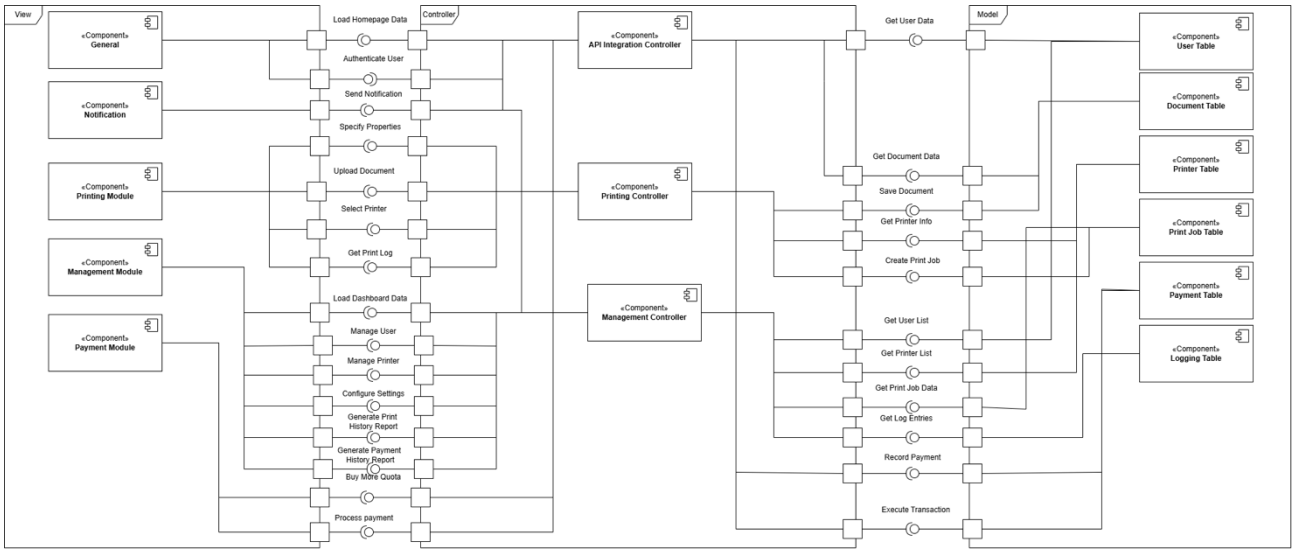

Figure 28: Component Diagram for Printing Module

In the component diagram, these different modules, controllers and tables are connected, showing how they relate to other system parts. Left, five main modules including **General**, **Notification**, **Printing Module**, **Management Module** and **Payment Module** respectively perform user-oriented functionalities. With that, the General module has functions that include **Load Homepage Data** and **Authenticate User** so it establishes how the system is going to talk to the system. The Notification module consists of the **Send Notification** which sends updates to the user. **Upload Document**, **Select Printer**, and **Get Print Log** are features of this Printing Module, allowing document management and printing operations. **Load Dashboard Data**, **Manage User**, **Manage Printer** and **Configure Settings** are all functions found in the Management module which is an administrative module. It, also, contains reporting functions entitled **Generate Print History Report** and **Generate Payment History Report**. User quota and payment operations handled by the Payment module include **Buy More Quota** and **Process Payment**.

Across from the center are a few controllers connecting modules to the database table on the right. It provides functions such as **Get User Data**, **Get Document Data** and **Save Document** to handle interactions between systems with external or internal data sources. The printing controller has the function of creating a **Print Job** and a **Get Printer Info** which implies that it is responsible for managing and executing printing tasks. Functions such as **Get User List**, **Get Printer List**, and **Get Print Job Data**, are imperative for system management and monitoring, and are what the Management Controller is responsible for, retrieving various system lists and data. Financial transactions are performed and consist of **Record Payment**, **Execute Transaction** etc. to support and log payment activities within the system via the Payment Controller.

Several tables on the right store and organize system data. The User Table contains the User List & User Data while Document Table is linked with saved documents. Information on the printers and the individual print jobs are managed by the Printer Table and Print Job Table, respectively. Supporting functions Record Payment and Execute Payment in beta, the Payment Table to log payment-related data. The Logging Table documents system activities, and is updated to record in the Logging Table a record of system activities, which is useful in the Get Log Entries function.