Software Requirements Specification

for

Student Smart Printing Service

Version 1.0 approved

Prepared by:

1. Nguyen Trinh Tien Dat - 2252147

2. Nguyen Viet Hoang – 2252235

3. Nong Thuc Khue – 2252385

4. Pham Doan Bao Trung – 2252858

5. Pham Le Huu Hiep – 2252223

Department of Software Engineering

Faculty of Computer Science and Engineering

Ho Chi Minh City University of Technology – VNU-HCM

September 20, 2024

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. REQUIREMENT ELICITATION

1.1. Domain Context

Every year, there is a huge demand for printing study materials from thousands of students of Ho Chi Minh City University of Technology (HCMUT). A student must print a large number of papers each semester and the students who do not have their own printers must go to a print shop to print documents, which takes lots of their time and money. Therefore, the Student Smart Printing Service (HCMUT_SSPS) is designed to facilitate efficient printing for HCMUT students, which will enhance student accessibility to printing resources as well as address the above mentioned students' time and finance problems.

Each student is given a default number of A4-size pages for printing every semester and allowed to buy more pages by using feature on SPSS website/app and paying payment through online payment system. The students must upload the document file onto the system and only the one which is permitted by the Student Printing Service Officer (SPSO) is printed. SPSO has the permission to view the printing history of all students and the students can also see their printing history. This feature helps students keep track of money they spent on printing as well as helps the SPSO manage students' printing activities and the number of printed papers, which is related to the sustainability issue and the SPSS has to propose solutions for reducing the paper waste in the range of school. By storaging the printing activities, the system can analyze and generate report automatically.

For securing SPSS system and tracking printing activities, all users have to be authenticated by the HCMUT_SSO authentication service before using the SPSS system. In general, the SPSS system is a user-friendly software of providing efficient and convenient service to fulfill the huge printing demand of HCMUT students. For more detailed description, we would like to explain and implement in this report.

1.2. Stakeholders and Needs

Stakeholders of the system may be students (end users), SPSOs (system managers), and the university.

As students, they need efficient ways to print many materials associated with their studies. Student Smart Printing Service (SSPS) therefore allows remote control over printers across the campus through a web-based app and a mobile app, which can be used to locate their locations fast and conveniently. There are many file types related to their study such as PDF, DOCX, PPTX. Students should be able to upload and print a variety of them. In addition, the materials may be printed to their preference. As a result, they need customizable printing properties such as paper size, page selection, number of copies, etc. There is limitation to the number of pages that are allowed to print, however, so students also need usage tracking. The system should allow them to view their printing history and remaining page balance. Of course, there is purchasable usage extension such as additional allowed pages, so students can make use of that. They would want a secure, quick, and seamless payment experience such as online transaction through BKPay system. Since there are already a lot of options in SSPS system, students need an intuitive user interface design as a consequence of handling such complexity.

As Student Printing Service Officers (SPSOs), they need to manage printers effectively. They would want remote control over the printers across the campus conveniently and rapidly through a web based app and a mobile app like the students. They are in charge of adding, enabling, disabling, or configuring the printers like file type permission, active page limit, etc in order to ensure optimal resource utilization or expected operations. Additionally, they need transparency and accountability for audit and troubleshooting the operations. As a result, access to log and history to view and manage student usage of printers is critical. They can view the number of active users and printers in real time to make decisions as needed. To manage student usage, they may have the authority over customizable system settings such as upper page limit, upper printer limit, time limit, etc. They may warn (or notify), suspend, ban, delete, or create any student printing account as necessary. At the end of each month or each year, they may generate usage reports as needed for identifying trends and planning future improvement. Like the

students, SPSOs need intuitive user interface to handle all the management complexity above.

As the system owner, the university need to oversee larger operational and financial aspects of SSPS system. First, they need to ensure security for the system. As a consequence, they need to integrate and manage authentication service for all users through HCMUT_SSO authentication system. Additionally, online payment system such as BKPay for student usage extension purchase is essential and should be managed carefully. The university need detailed record and automated monthly and yearly reports to keep track of all purchases. In addition, system maintenance cost such as paper cost, ink cost, printer maintenance cost, etc must be fully accounted for.

As IT department, their primary concern is ensuring system reliability and security. First, they need to ensure seamless integration of the system with the university's HCMUT_SSO authentication service and online payment system such as BKpay. Second, they need to maintain system scalability and reliability that can handle high usage automatically during peak times, such as the end of a semester when students typically print final assignments. Additionally, the team may provide regular system monitoring and updates for smooth operations on various devices and platforms through the web-based app and mobile app. Finally, they may implement automated backups and manage associated data security, including protecting sensitive information such as student IDs, payment details, printing logs, etc.

As Financial department, they need to manage system maintenance cost effectively. Therefore, they need monthly or yearly generated reports on system usage such as paper usage, ink usage, printer maintenance, etc and their cost for budget planning and resource allocation. They also need online payment records associated with system usage extension for auditing purposes.

1.3. Benefits of the System

The HCMUT-SSPS delivers significant advantages to its primary users, the students, faculty, and the university administration.

In the first place, for students, this system provides a seamless, user-friendly printing experience for their academic needs, such as coursework, assignments, and reports. It enables quick submission of documents, selection of optimal printing options, and easy tracking of their usage. This will lead to time saving, reducing frustration, and enhancing the overall academic experience, which will allow students to give more concentration on their studies without the burden of inefficient printing processes.

In addition, faculty members and instructors also benefit from the system's accessibility in terms of simplifying the preparation of lesson materials, assignments, and resources. The convenience of on-demand printing ensures that they can easily provide materials to students, supporting a more efficient and effective teaching environment. With a reported high paper usage for educational purposes, this system helps manage printing demands in an organized manner. A study conducted in 2020 found that a typical school in the United States uses an average of 2,000 sheets of paper per day. Over a full school year of 160 days, this amounts to over 320,000 sheets of paper per year. With approximately 100,000 schools in the U.S., the total paper consumption in schools reaches around 32 billion sheets per year (Record Nations, 2020).

Besides, for the SPSO, the system offers enhanced administrative control, safety, and streamlined management of the printing infrastructure. By enabling easy adjustments to system settings - such as file type permissions, page allocations, and printing dates, the system ensures smooth operation. It also facilitates effortless monitoring and maintenance of the printer fleet, ensuring continuous availability for students.

Finally, the university administration gains valuable insights from the system's reporting capabilities. Informative data on resource usage, costs, and printing trends helps administrators make informed decisions regarding budgeting, resource allocation, and

system improvements. In line with global studies showing the value of data-driven decision-making, the HCMUT-SSPS supports future improvements for both students and the university. According to a scientific publication of Harvard Business Review, organizations that leverage data for high-impact business decisions are three times more likely to report significant improvements in decision-making compared to those that rely less on data (Harvard Business Review, 2012).

1.4. Functional Requirements

ID	STAKEHOLDERS	REQUIREMENT
1	Students	- The system lets students choose a printer by showing all the printers' properties such as ID, brand name, printer model, short description and location, and let the student choose one. - The system allows a student to upload a document file onto the system. - After choosing printer, the system allows student to specifying the printing properties such as paper size, pages (of the file) to be printed, one/double-sided, number of copies, etc. - Students can send the printing actions by clicking button "Accept", then students can track status of the action (the status is "In process" means it has not done yet, "Done" means it has been done and students can go to the site that they choose the printer and get the printing papers). - The system has to log and store the printing actions for all students, including student ID, printer ID, file name, printing start and end time, number of pages for each page size so that students can view his/her printing log for a time period.

		- Each semester the system must deliver a number of papers for each student. - Students are allowed to buy some more using the feature Buy Printing Pages of the system and pay the amount through some online payment system like the BKPay system of the university. - The system only allow a student to print some number of pages when it does not exceed his/her account (page) balance. - After each printing operation, present a real-time account balance and remaining paper update to ensure transparency.
2	Student Printing Service Officer (SPSO)	- The system allows the SPSO to view the printing history (log) of all students or a student for a time period (date to date) that include the student ID, date sent the action, date done the action, number of papers used,… - The system also allows the SPSO to view the printing history of all or some printers include: Date done the action, ID of the student that requests action, number of paper used,… - The SPSO has a feature to manage printers such as add/enable/disable a printer. - Provide the feature to SPSO to manage other configuration of the system 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.

3	University Administration	- All users have to be authenticated by the HCMUT_SSO authentication service before using the service. - Generate reports summarizing printing trends monthly or annually. - Access to dashboard displaying real-time printing statistics and system performance metrics. - Support export for reports in many reports (CSV, PDF,…).
4	Finance Department	- Provide data to BKPay to process payment securely. - Maintain a transparent transaction log, associating. payments with student accounts for auditing purposes. - Must provide financial reporting, summarizing revenue from printing services on a monthly basis.
5	IT Department	- Provide a real-time monitoring dashboard for IT staff to track server performance and network connectivity, and also to fix bug. - Automated system backups, with backups stored. securely and regularly tested for restoration. - Automated server maintenance and updates scheduled during off-peak hours.

1.5. Non-Functional Requirements

ID	STAKEHOLDERS	REQUIREMENT
1	Students/Instructors	- Allow 1000 students to use the service concurrently during peak periods.

		- The response time of processing a request should be less than 2 seconds. - Instructors and students can use the printing service after 1 hour training.
2	Student Printing Service Officer	- Backup user data and print job queues to prevent data loss when the system crash. - In case of hardware failure, the system must rearrange print jobs to available printers.
3	University administrator	- The system must provide easy – to – use user interface that allows user to navigate quickly between dashboards, reports. - The system must be available 100% time during office hours and 24/7 for specific services.
4	Financial Department	- All information about transaction must be encrypted using strong encryption protocols. - Every transaction or accessing financial resources must be logged. Audit logs should include actions, timestamps and people who accessed.
5	IT Department	- The system must follow a modular architecture that allows easy updates without disrupting the entire system. - The system must be compatible with different operating systems.

2. USE-CASE DIAGRAMS

2.1. Use-case Diagram for the Whole System

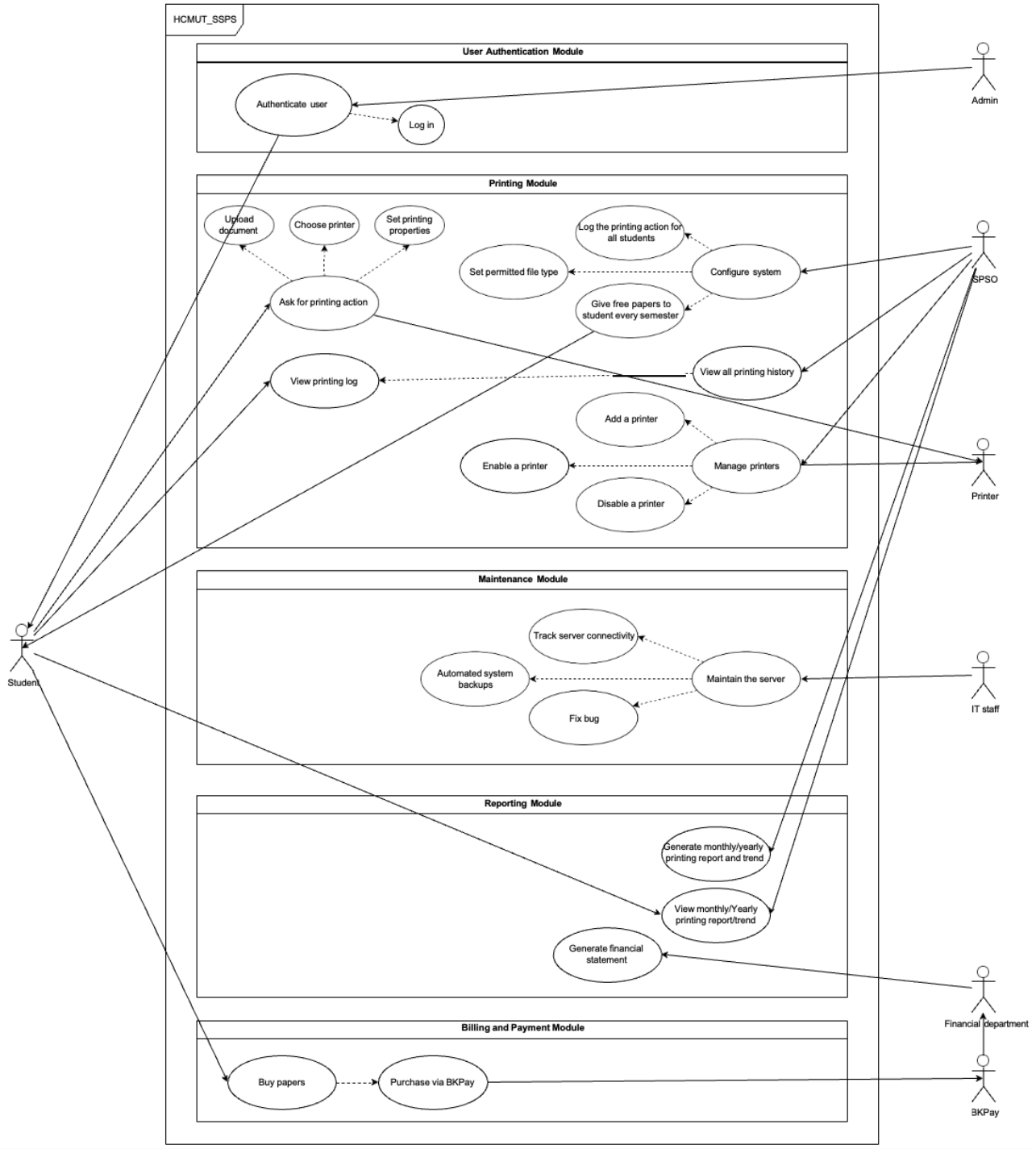

Figure 2.1. Use-case diagram for the Whole System

2.2. Use-case Diagram for Printing Module

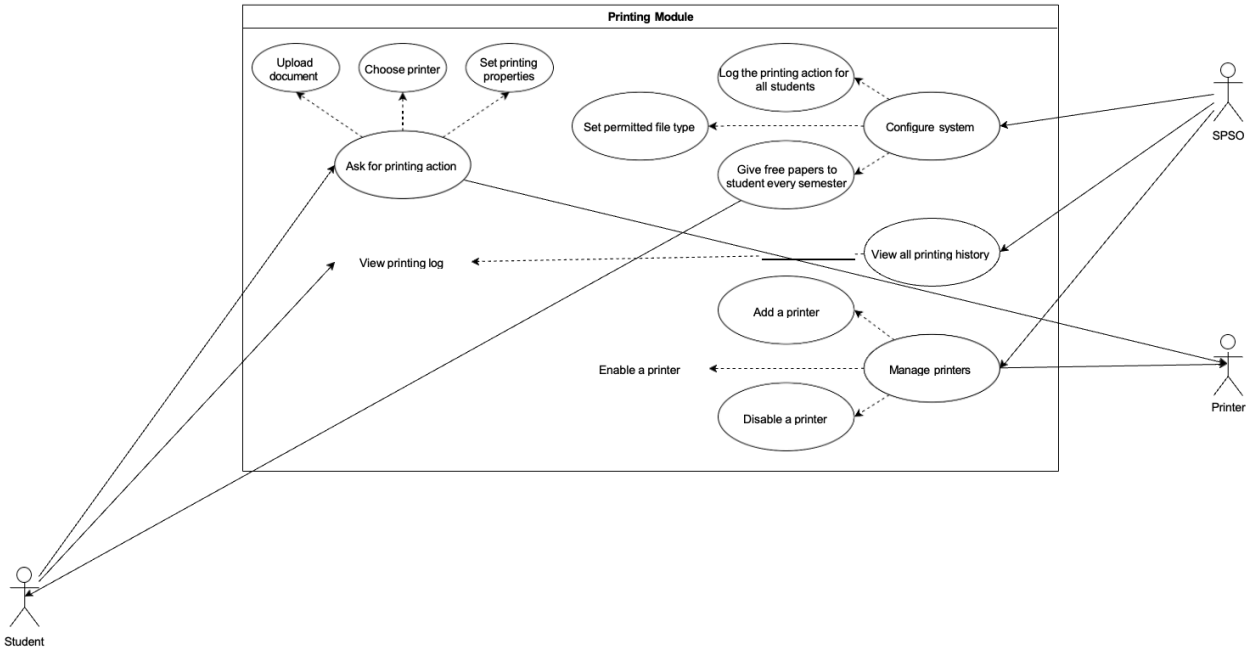

Figure 2.2. Use-case diagram for Printing Module

2.3. The Details of Use cases in Printing Module

2.3.1. Use case upload document

Use case name	Upload document
Actor	Students, Lecturers, and Faculty
Description	Giving users the permission to upload documents for printing
Pre-conditions	Users have HCMUT accounts and are logged into the system
Post-conditions	The documents are submitted and ready for printing
Trigger	Users press the submitting button
Normal Flow	1. Users navigate to the upload section. 2. Users set up and select the document file. 3. Users clicks "Upload" and receive the confirmation pop-up 4. System confirms successful upload.

	5.The preview version can be seen clearly
Exceptions	*Exception at step 2*: If the format is violated, the error pop-up appear *Exception at step 3*: If there is server error during the uploading phase, the system displays the error code
Alternative Flows	At step 2: 2a. The users have to adjust the format and continue with step 3 2b. If the users do not want to print anymore, they can cancel the phase and return the main page At step 3: 3a. They may contact the help desk

2.3.2. Use case print document

Use case name	Print document
Actor	Students, Lecturers, and Faculty
Description	Allowing users to review the remaining paper and decide to start the printing phase based on the suitable properties and paper
Pre-conditions	The documents have been uploaded
Post-conditions	The documents are printed successfully without any issue
Trigger	The documents are uploaded
Normal Flow	1. Users are able to available printers lists with their information including locations, types, status, etc. 2. Users select the appropriate printer from the list. 3. The system displays a dialog box with the printing properties.

	4. The users set up the metrics: number of copies, page range, page settings (single/double sided, page size, orientation, margin) and print handling (multiple pages per sheet). 5. The users confirm the selected properties. 6. The system updates the print job with the chosen properties. 7. The users press the send request button. 8. Users receive the confirmation pop-up. 9. The request is queued up to the printing phase
Exceptions	*Exception at step 1:* If all the printers are busy, the users need to wait. *Exception at step 4:* If the selected page range is out of bounds, the system corrects to the valid page selections. *Exception at step 4:* If there's running out of page, it will show in the UI and the send request button will be unclickable. There will be a notification that requires the user to change printing property or purchase more pages.
Alternative Flows	*At step 1:* 1a. If the users decide not to proceed, they can cancel the upload and return. *At step 4:* 4a. If the user's paper balance is insufficient to cover the print job cost, they are prompted to purchase additional pages before proceeding.

	4b. If the user decides to adjust the printing properties, they can do so by clicking an “Edit Properties” option and returning to the property selection step.

2.3.3. Use case view printing log

Use-case name	View printing log
Actor	Students
Description	Allows users to view their own record of printing actions for a time period, including student ID, printer ID, file name, printing start and end time, number of pages for each page size.
Preconditions	Users log in the HCMUT_SPSS system succesfully.
Postconditions	Users view their printing log.
Trigger	Users click the “View printing log” section.
Normal Flow	1. Users log in the HCMUT_SPSS system successfully. 2. Users click the “View printing log” section. 3. Users filter the criteria of start and end date, printer selection. 4. The system displays the printing log after filtering the criteria.
Exceptions	*Exception at step 4:* The system displays message “No result” if there is no data match the criteria.
Alternative Flows	None.

2.3.4. Use case view printing history

Use-case name	View printing history
Actor	Student Printing Service Officer (SPSO)

Description	Allows SPSOs to view printing history of one or all students for a time period and for some or all printers.
Preconditions	SPSOs log in the HCMUT_SPSS system successfully.
Postconditions	SPSOs view the printing history of all students or a student as they want.
Trigger	SPSOs click the “View printing log” section.
Normal Flow	1. SPSOs log in the HCMUT_SPSS system successfully. 2. SPSOs click the “View printing log” section. 3. SPSOs choose to view all students’ printing history or a particular student’s history. 4. SPSOs filter the criteria of start and end date, printer selection. 5. The system displays the printing history as SPSOs’ desire.
Exceptions	*Exception at step 4:* The system displays message “No result” if there is no data match the criteria.
Alternative Flows	None.

2.3.5. Use case buy printing pages

Use case name	Buy printing pages
Actor	Primary actor: Student Secondary actor: BKPay system
Description	A module that allows students to buy pages through online payment system
Trigger	The student clicks the “Buy Printing Pages” button
Pre-conditions	PRE-1. Students have logged into the system

	PRE-2. Student must have enough account balance
Post-conditions	POST-1. Balance fluctuations and success transaction must be notified to the student POST-2. The number of pages increase
Normal Flow	1. The system displays a form for student to input the type of page and number of pages to purchase 2. Students specify the type and number of pages 3. Students click “Buy” button 4. The system add pages and decrease money in students’ account balance 5. The system displays successful message and return to homepage
Alternative Flow	None.
Exceptions	3.E. The account balance is not sufficient

2.3.6. Use case manage printers

Use case name	Manage printers
Actor	Primary actor: SPSO
Description	A module that allows SPSO to manage their printers
Trigger	SPSO clicks the “Manage Printer” button
Pre-conditions	PRE-1. SPSO identity is authenticated PRE-2. SPSO is authorized to manage printers
Post-conditions	POST-1. The printer is modified according to SPSO
Normal Flow	1. The system displays an interface that has 3 options: add, enable, disable

	2. SPSO chooses 1 of 3 options (See 2.1, 2.2, 2.3) 2.1. Add 2.1.1. SPSO specifies information of the printers (ID, brand name, printer model, short description, location) 2.1.2. The system adds printer to the list 3. SPSO clicks “Save” button 4. System saves the changes 5. The system displays a successful message to SPSO
Alternative Flows	Alternative options in step 2: 2.2. Enable 2.2.1. SPSO specifies ID of the printer and clicks “Enable” button 2.2.2. The system matches ID with the existing printers 2.3. Disable: 2.3.1. SPSO specifies ID of the printer and clicks “Disable” button 2.3.2. The system matches ID with the existing printers
Exceptions	2.2.2.E. The system cannot find the printer 2.3.2.E. Same error as 2.2.2.E

2.3.7. Use case manage configuration

Use-case name	Manage configuration
Actor	SPSO

Description	Users choose system settings such as page limit, printer limit, time limit, file type permission, etc., and the applied date for the system.
Precondition	Users authenticated by HCMUT SSO and authorized in system settings
Postcondition	System settings are applied and recorded.
Trigger	Users choose to set system configuration.
Normal flow	1. System lists a number of settings for users to choose. 2. Users choose and set necessary system configuration and the applied date. 3. The system saves, records the changes, and applies them on the specified date.
Alternative flow	Users decide to cancel the process or undo the changes to previous save in the system.
Exception	System settings are not available (because of physical constraints like page limit, printer limit, file limit, etc).

2.3.8. Use case generate report

Use-case name	Generate report
Actor	SPSO
Description	Users choose a monthly or yearly report for generation.
Precondition	Users authenticated by HCMUT SSO and authorized in report access
Postcondition	Report generated and recorded
Trigger	Users request report.

Normal flow	1. Users choose monthly or yearly report. 2. The system records and generates chosen report.
Alternative flow	Users generate available report within a month or a year.
Exception	No report available

3. SYSTEM MODELLING

3.1. Activity diagrams

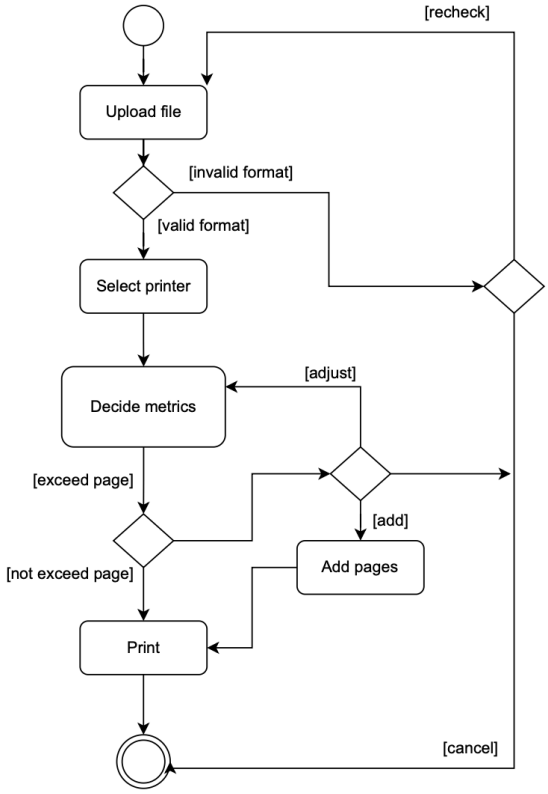

Figure 3.1. Activity diagram - Student printing

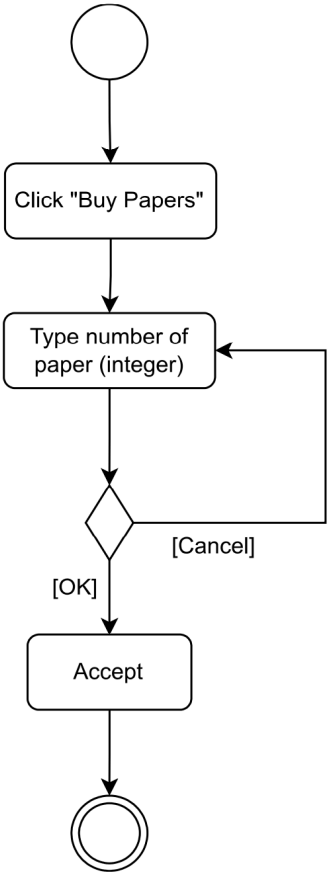

Figure 3.2. Activity diagram - Student buys pages

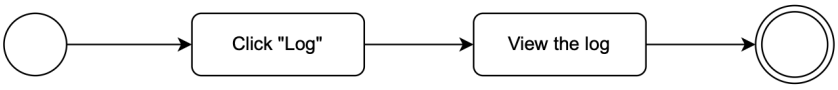

Figure 3.3. Activity diagram - Student views printing log

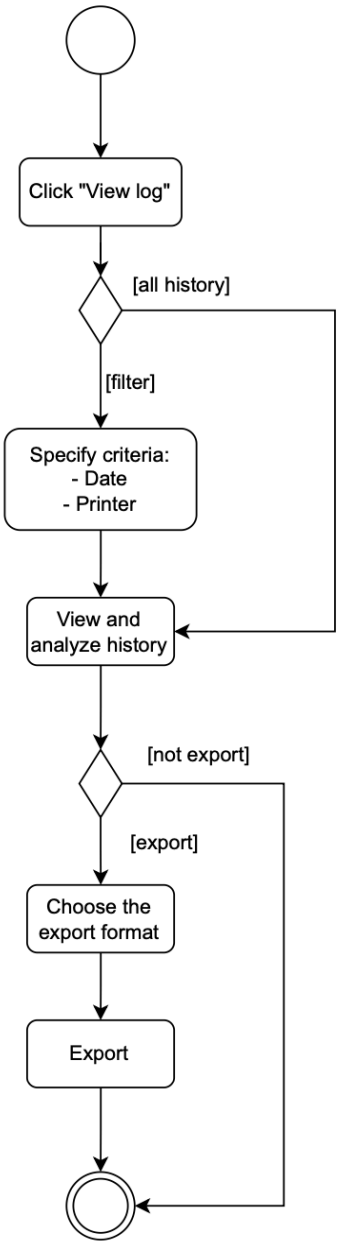

Figure 3.4. Activity diagram - SPSO views printing history

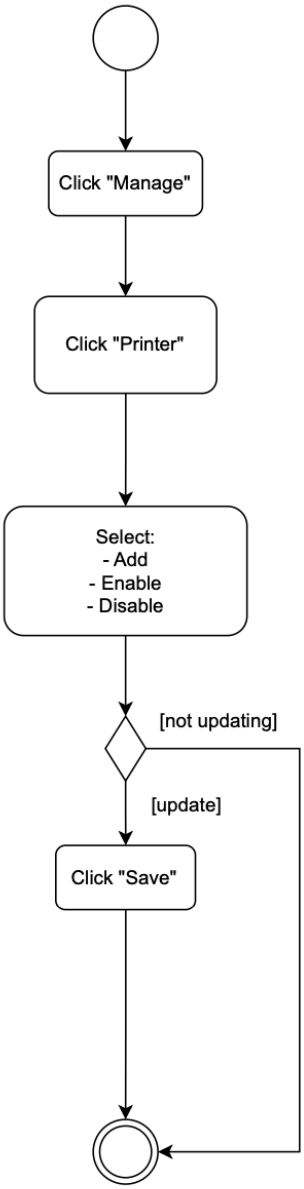

Figure 3.5. Activity diagram - SPSO manages printers

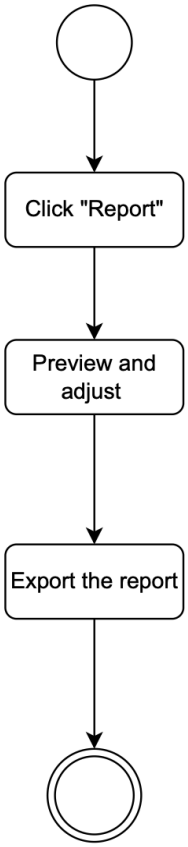

Figure 3.6. Activity diagram - SPSO generates report

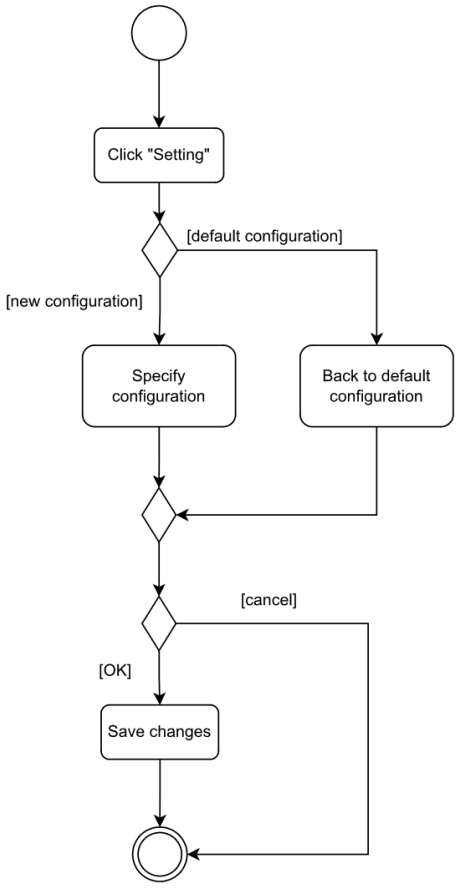

Figure 3.7. Activity diagram - SPSO manages configuration

3.2. Sequence diagrams

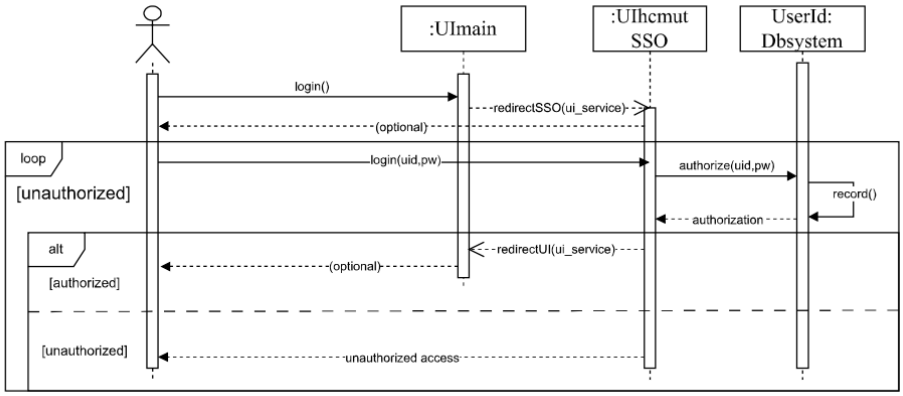

Figure 3.8. Sequence diagram - Login

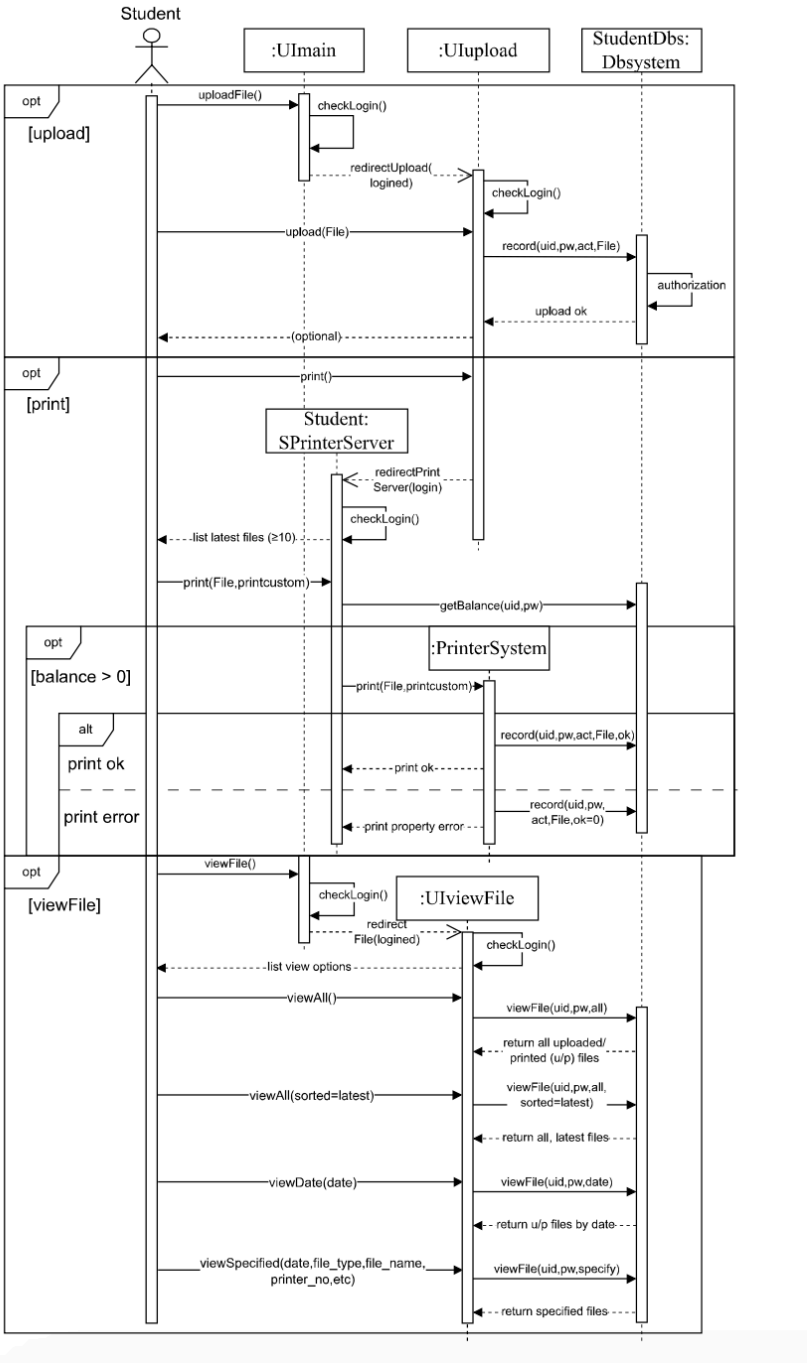

Figure 3.9. Sequence diagram - Student printing

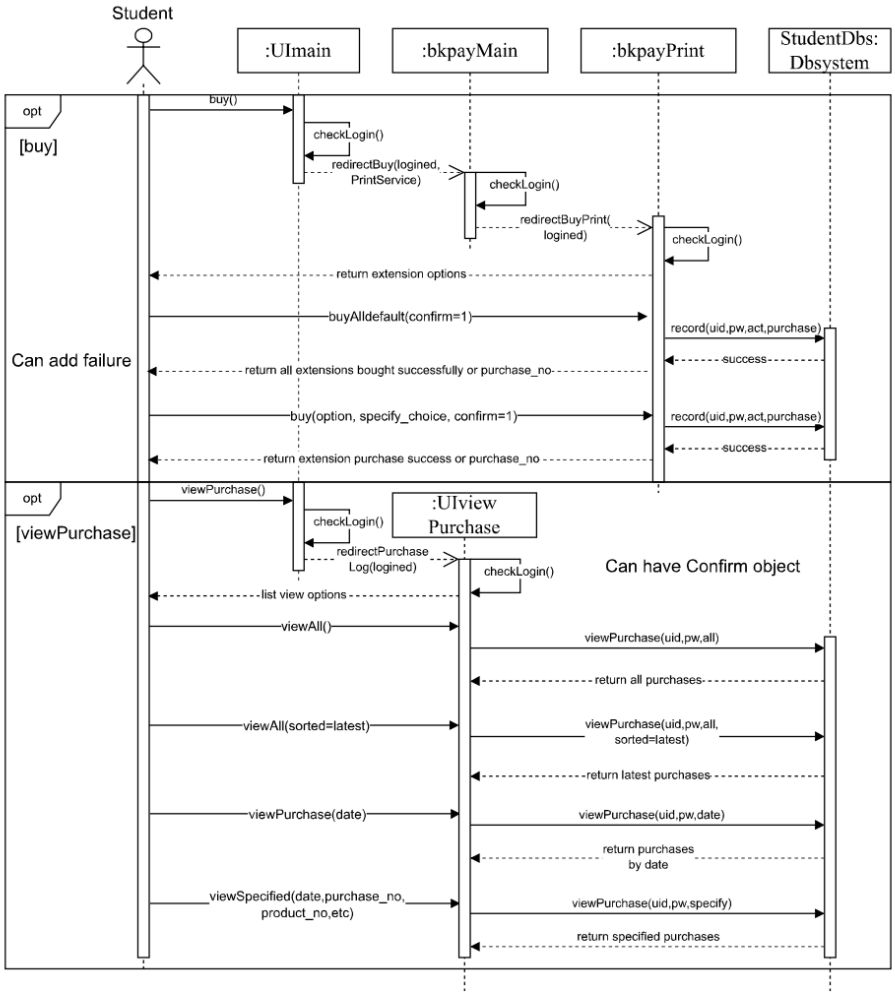

Figure 3.10. Sequence diagram - Student buys pages

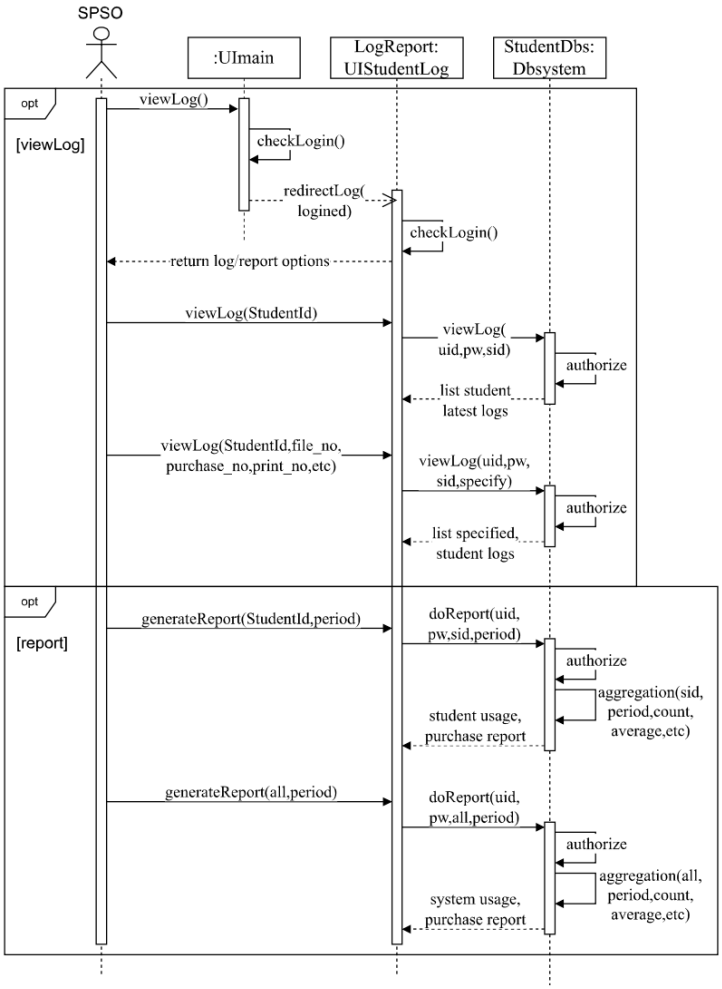

Figure 3.11. Sequence diagram - SPSO views printing history

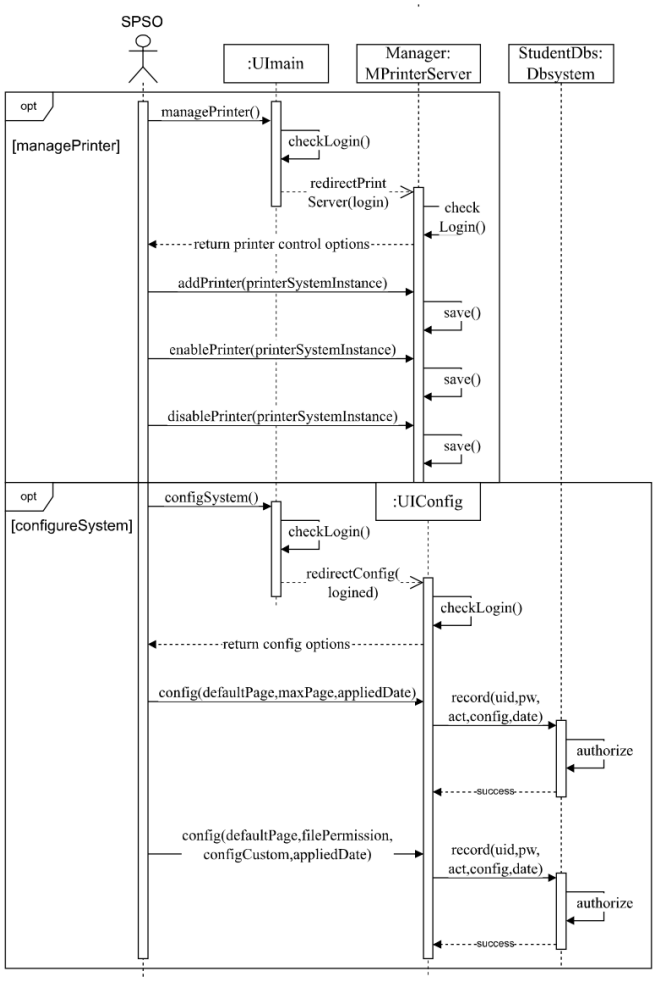

Figure 3.12. Sequence diagram - SPSO manages printers

3.3. Class diagrams

3.3.1. View layer

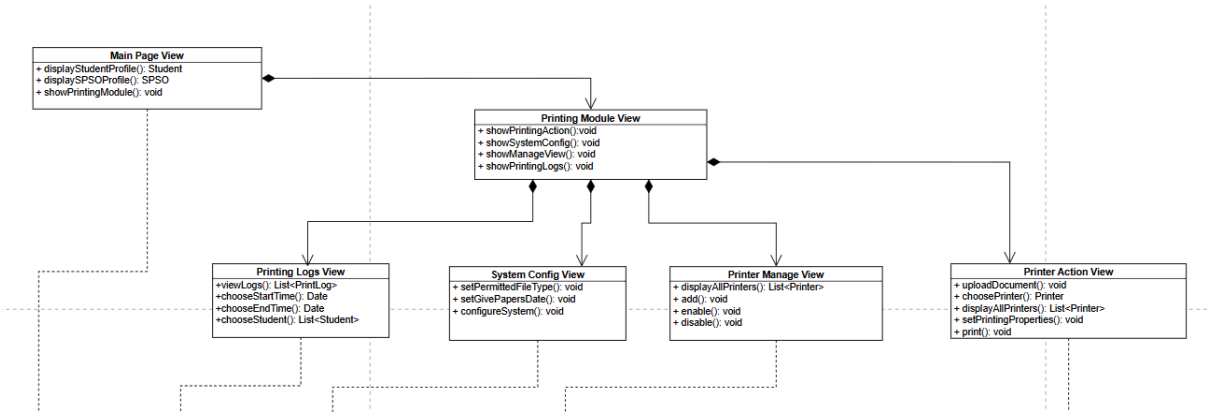

Figure 3.13. Class diagram - View layer

1. displayStudentProfile():

- Retrieves authenticated student's data and render it with user-friendly interface.

- Provides an overview of the student's profile.

2. displaySPSOProfile():

- Retrieves authenticated officer's data and render it with user-friendly interface.

- Offers insights into the SPSO overseeing the printing service.

3. showPrintingModule():

- Provides a comprehensive view of modules with list of functionalities.

- Changes from main page to printing module page.

4. showPrintingAction(): Provides a user-friendly interface for students to choose configuration and upload file in order to print.

5. uploadDocument():

- Allows a student to upload file for printing.

- Initiates the document upload process to be used in printing requests.

6. choosePrinter():

- Assists the student in selecting a printer from the available ones.

- Enhances the user experience by simplifying the printer selection process.

7. displayAllPrinters():

- Assists students when they need to choose a printer.

- Supports choosePrinter() function.

8. setPrintingProperties():

- Allows student to choose configuration for printing (number of pages, number of copies, paper type, paper size,...).

- Simplifies the process of choosing properties with easy-to-use interface.

9. print(): After uploading, setting properties and choosing printer, student can now enable to print their document.

10. showManageView():

- Provides management interface for Smart Printing Service Officer (SPSO) to view and manage available printers.

- Allows officers to perform administrative tasks such as adding, removing, or updating printer details within the system.

11. add():

- Allows officer to add a printer to the system.

- Provides the officer with a form to input information about the printer.

12. enable():

- Allows officer to enable a printer from disabled state.

- If the printer has already been enabled, an unchanged message will appear.

13. disable():

- Allows officer to disable a printer from disabled state.

- If the printer has already been disabled, an unchanged message will appear.

14. showSystemConfig():

- Opens the configuration settings for system, allowing officer to view and modify its operational parameters.

- Enables officers to adjust system-specific settings such as number of default papers and date to give papers.

15. setPermittedFileType():

- Defines and sets the file types permitted by the system for upload or processing.

- Allows administrators to configure which file formats are accepted, ensuring compatibility and security within the system.

16. setGivePaperDate():

- Defines and sets the date for automatically give papers to all students.

- Allows administrators to configure the date to give papers to all students.

17. configureSystem():

- Saves all configuration of the system.

- Passes all data input from officer to the controller.

18. showPrintingLogs():

- Displays a view of the system's printing logs, allowing officers to track and review print activity.

- Provides detailed records of past printing jobs, including time, user, printer used, and file details for audit and monitoring purposes.

19. viewLogs():

- Retrieves and returns a list of print logs from the system for review or analysis.

- Provides officer or student with a detailed list of past print jobs, including relevant information such as timestamps, users, and printer details.

- Optional choice for officer to choose see print logs of a student or all students.

20. chooseStartDate():

- Chooses a start date to retrieve print logs.

- Allows student and officer to set a start date for making lower bound to get print logs.

21. chooseEndDate():

- Chooses an end date to retrieve print logs.

- Allows student and officer to set an end date for making upper bound to get print logs.

22. chooseStudents():

- Retrieves a list of students in order to support viewing print logs of those students.

- Allows officers to choose students for targeted retrieval of their printing history and logs for monitoring or reporting purposes.

3.3.2. Controller layer

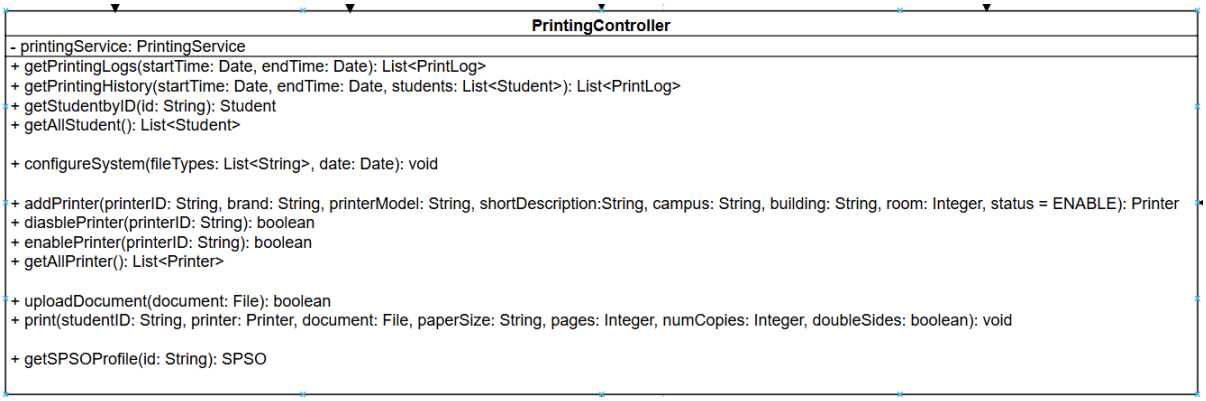

Figure 3.14. Class diagram - Controller layer

1. getPrintingLogs(startTime: Date, endTime: Date):

- Retrieves and returns a list of print logs within a specified time range, from startTime to endTime.

- Acts as a controller-layer function to filter and provide print logs based on the given date range for auditing or reporting purposes.

2. getPrintingHistory(startTime: Date, endTime: Date, students: List<Student>):

- Returns a filtered list of print logs for the specified students within a given time range from startTime to endTime.

- A controller-layer function used by officers to retrieve printing history for selected students during a specific period, enabling targeted review and monitoring.

3. getStudentByID(id: String):

- Retrieves and returns a student's details based on the provided student ID.

- A controller-layer function called by the chooseStudents() function in the view layer, enabling officers to select a student for retrieving their printing history using getPrintingHistory().

- This function is also useful when the student login which will return a comprehensive profile for the interface.

4. getAllStudent():

- Retrieves and returns a list of all students from the database for further processing.

- A controller-layer function called by the chooseStudents() function in the view layer, allowing officers to retrieve all printing history using getPrintingHistory().

5. configureSystem(fileTypes, List<String>, date: Date):

- Sends the provided file types and date to the service layer and saves the system configuration in the SystemConfig class.

- A controller-layer function that handles system configuration, ensuring the allowed file types and date are updated and persisted within the system configuration.

6. addPrinter(printerID: String, brand: String, printerModel: String, shortDescription:String, campus: String, building: String, room: Integer, status = ENABLE):

- Sends the provided printer details to the service layer and saves the printer object in the database.

- A controller-layer function that adds a new printer to the system with details like ID, brand, model, location, and status, ensuring it is stored in the database for future use.

7. diasblePrinter(printerID: String):

- Finds the printer by its ID and disables it, marking it as unavailable for use.

- A controller-layer function triggered by the officer from the view layer, passing the printer ID to the service layer to update its status to "DISABLED" in the system.

8. enablePrinter(printerID: String):

- Locates the printer by its ID and enables it, making it available for use.

- A controller-layer function called by the officer from the view layer, passing the printer ID to the service layer to update its status to "ENABLED" in the system.

9. getAllPrinter():

- Retrieves and returns a list of all printers from the repository for display in the view layer.

- A controller-layer function that provides officers with a list of printers, enabling them to select and manage specific printers through the interface.

10. uploadDocument(document: File):

- Passes the uploaded document to the service layer for validation against permitted file types.

- A controller-layer function that handles the student's file upload, ensuring the document complies with the system's allowed file types before proceeding.

11. print(studentID: String, printer: Printer, document: File, paperSize: String, pages: Integer, numCopies: Integer, doubleSides: boolean):

- Sends the specified printing parameters to the service layer to initiate the printing of the student's document after validating the file type.

- A controller-layer function that processes the print request, ensuring all necessary parameters, such as printer selection, document, paper size, number of copies, and double-sided printing are provided before executing the print job.

12. getSPSOProfile(id: String):

- Retrieves the profile information of a Smart Printing Service Officer (SPSO) by their ID from the database via the service layer.

- A controller-layer function that fetches the SPSO's details and returns them to the view layer for display or further actions.

3.3.3. Service layer

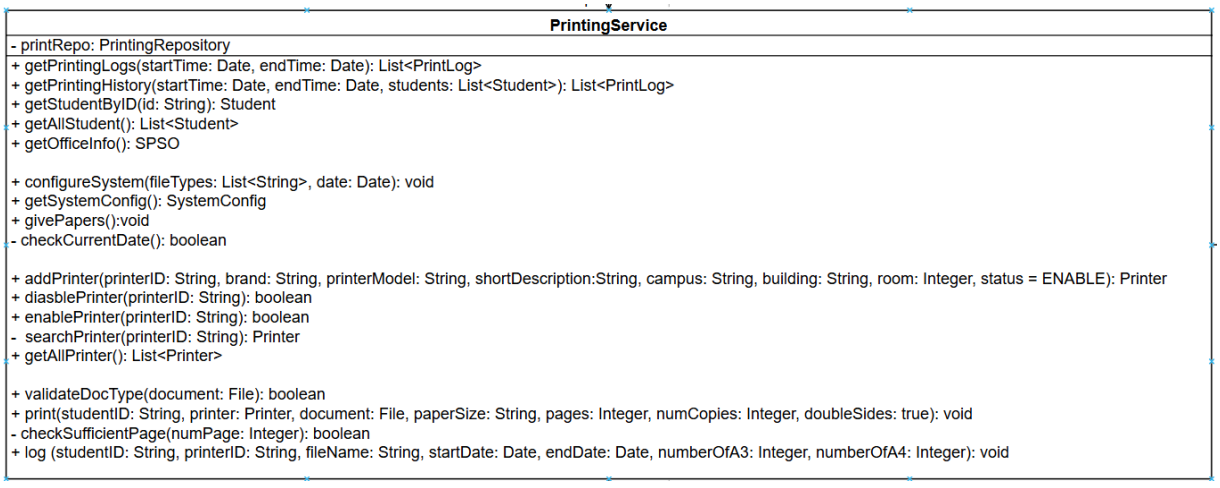

Figure 3.15. Class diagram - Service layer

1. getPrintingLogs(startTime: Date, endTime: Date):

- Fetches print logs from the database within the given startTime and endTime, as provided by the controller.

- A service-layer function that retrieves and returns the filtered print logs to the controller for further actions or display.

2. getPrintingHistory(startTime: Date, endTime: Date, students: List<Student>):

- Processes the provided student list and date range, then retrieves the corresponding print logs from the database.

- A service-layer function that gathers printing history for the specified students within the given time period and returns the results to the controller.

3. getStudentByID(id: String):

- Retrieves student information from the repository based on the provided student ID.

- A service-layer function that processes the ID and returns the corresponding student's details to the controller.

4. getAllStudent(): A service-layer function that fetches and returns the complete list of students to the controller for further use.

5. configureSystem(fileTypes: List<String>, date: Date): A service-layer function that processes the input and ensures the new configuration is saved to the SystemConfig class for future use.

6. getSystemConfig(): A service-layer function that fetches and returns the system's configuration data, allowing it to be modified or referenced in the configureSystem function.

7. givePapers():

- Automatically grants the default number of papers to students based on system configuration when the scheduled giving date arrives.

- A service-layer function that allocates the specified amount of paper to students, following the system's settings, once the giving date is triggered.

8. checkCurrentDate():

- Checks if the current date matches the giving date specified in the system configuration and returns a boolean value.

- A service-layer function that verifies the system's configured giving date, returning true if it matches the current date, enabling automatic paper distribution to students.

9. addPrinter(printerID: String, brand: String, printerModel: String, shortDescription:String, campus: String, building: String, room: Integer, status = ENABLE): A service-layer function that takes the printer's information from the controller layer and stores it in the database for future use within the system.

10. diasblePrinter(printerID: String):

- Locates the printer by its ID and disables it, throwing an exception if the printer is not found.

- A service-layer function that updates the printer's status in the database to "DISABLED," ensuring it is no longer available for use.

- Useful for maintenance or resolving issues with specific printers.

11. enablePrinter(printerID: String):

- Searches for the printer by its ID and enables it, throwing an exception if the printer cannot be found.

- A service-layer function that updates the printer's status in the database to "ENABLED," making it available for use once again.

12. searchPrinter(printerID: String):

- Locates the printer using its ID, returning the corresponding Printer object or null if the printer is not found.

- A service-layer function that provides the necessary information for the enable and disable operations, allowing these functions to handle cases where the printer does not exist.

13. getAllPrinter(): A service-layer function that facilitates the management of printers by providing the controller with up-to-date printer information for further processing.

14. validateDocType(document: File):

- Processes the uploaded document from the controller layer and checks if its file type matches any of the permitted types in the system configuration.

- A service-layer function that returns true if the document's type is valid, ensuring compliance with the system's file upload requirements.

15. print(studentID: String, printer: Printer, document: File, paperSize: String, pages: Integer, numCopies: Integer, doubleSides: boolean):

- Processes the print request by taking necessary parameters from the controller, including printer details and document specifications.

- A service-layer function that invokes the print method of the Printer object to execute the printing of the document with the specified settings.

16. checkSufficientPage(numPage: Integer):

- Verifies if there are enough pages available for printing based on the specified number of pages required.

- A service-layer function that checks the current paper supply and returns a boolean value indicating whether the printing request can be fulfilled without running out of pages.

17. log (studentID: String, printerID: String, fileName: String, startDate: Date, endDate: Date, numberOfA3: Integer, numberOfA4: Integer):

- Records the details of a successful printing operation, including the student ID, printer ID, file name, and the number of pages printed in A3 and A4 sizes.

- A service-layer function that creates a new log entry and stores it in the database, ensuring all relevant details of the print job are saved for future reference and analysis.

3.3.4. Repository

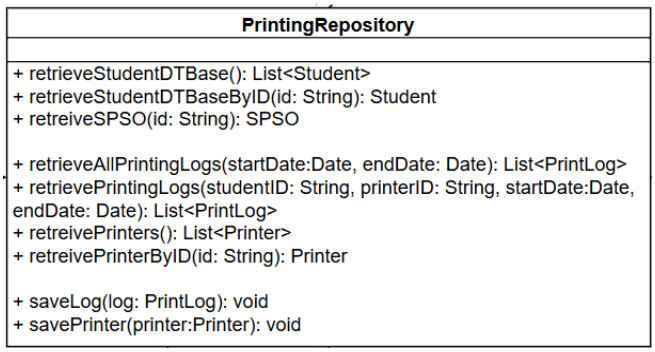

Figure 3.16. Class diagram - Repository

1. retrieveStudentDatabase(): Gets information of all students from the database, then return the list to service layer for processing.

2. retrieveStudentDatabaseByID(id: String): Finds student with ID from the database and then return the object to the service

3. retreiveSPSO(id: String): Finds the officer's information from the database and return the object to upper layer

4. retrivePrintingLogs(studentID: String, printerID: String, startDate: Date, endDate: Date): querying printing logs based on specified criteria: a specific student, printer, and within a specified date range.

5. retrieveAllPrintingLogs(startDate: Date, endDate: Date): query all printing logs of all students within a specified date range.

6. retrievePrinters(): query all records of printers from the database.

7. retrievePrinterByID(id: String): query record of printer base on id of printer.

8. saveLog(log: PrintLog): save the print log to the database.

9. savePrinter(printer: Printer): save new printer to the database.

3.3.5. Entity classes

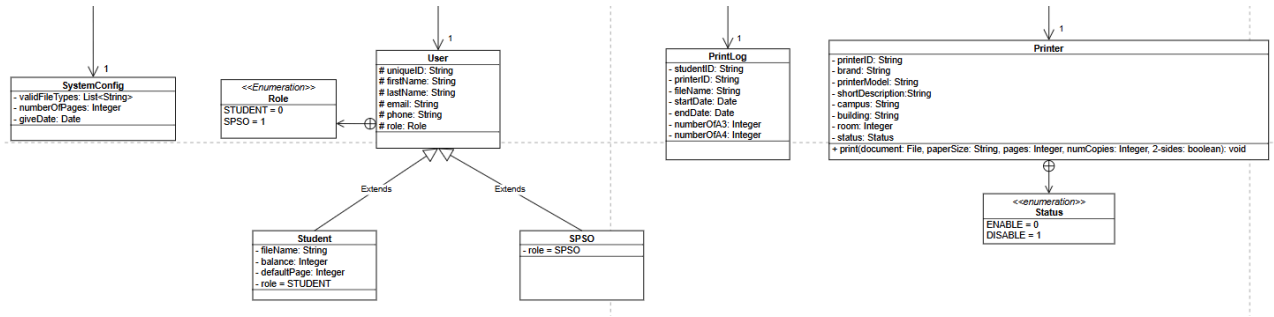

Figure 3.17. Class diagram - Entity classes

3.4. MVP Wireframe

3.4.1. Homepage

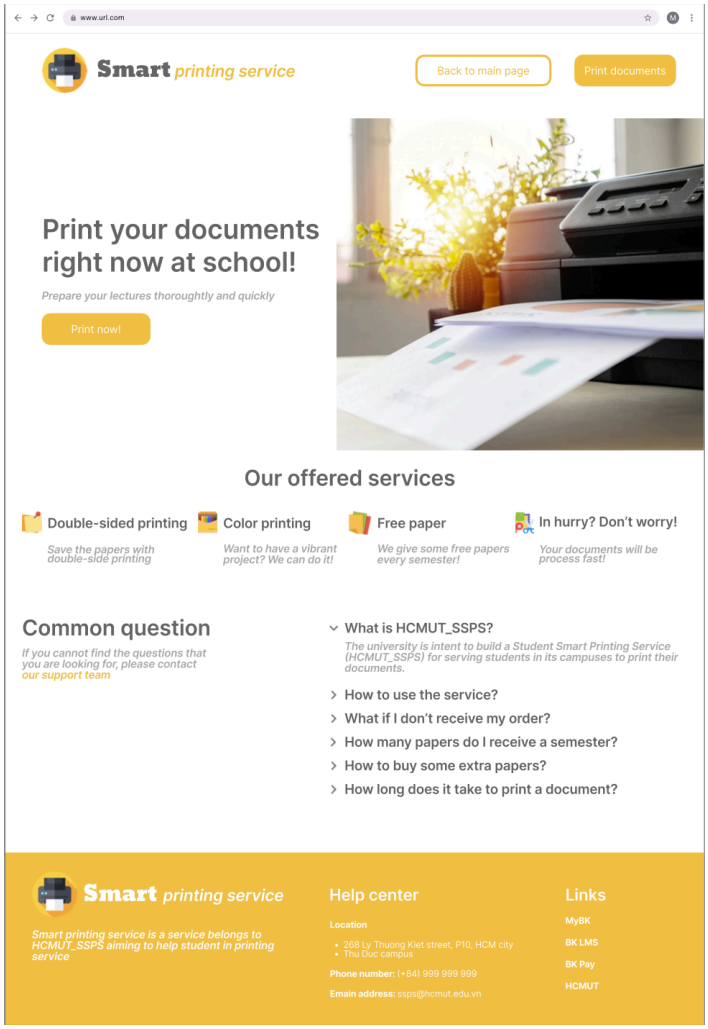

Figure 3.18. Homepage

3.4.2. Login

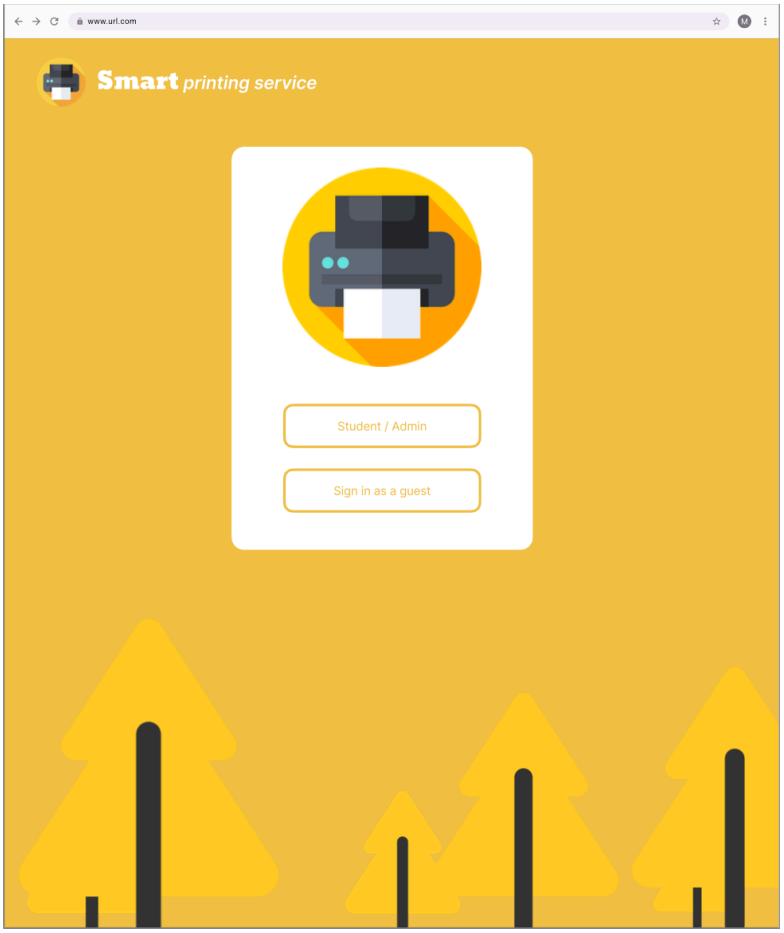

Figure 3.19. Login - Page 1

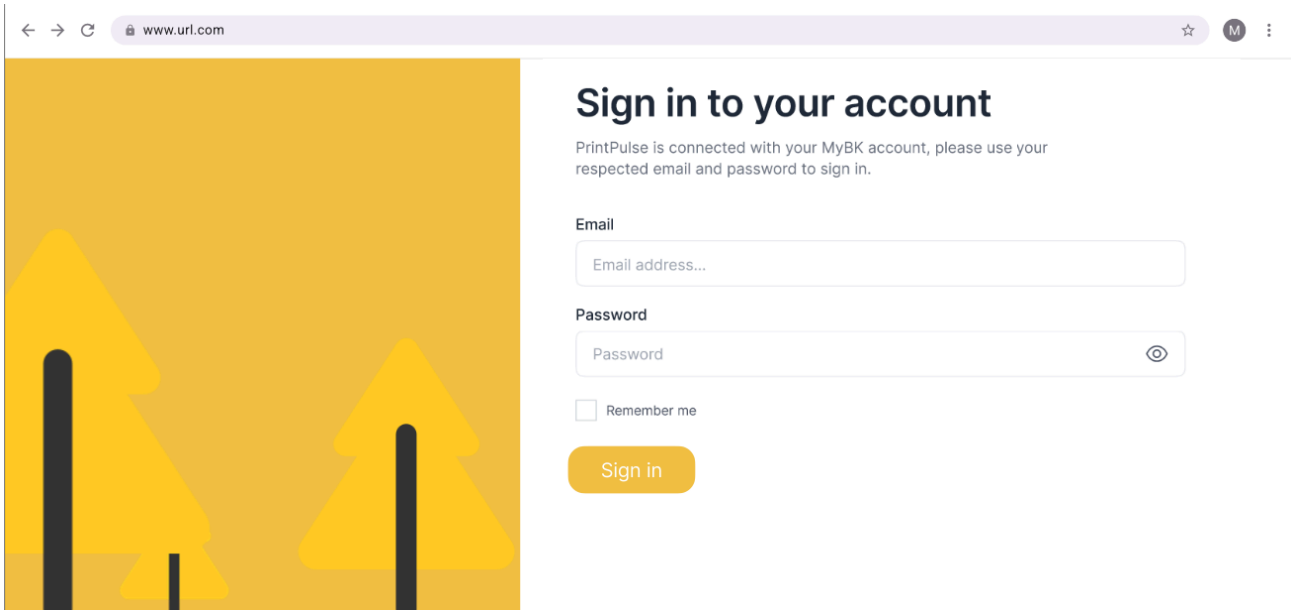

Figure 3.20. Login - Page 2

3.4.3. Notification

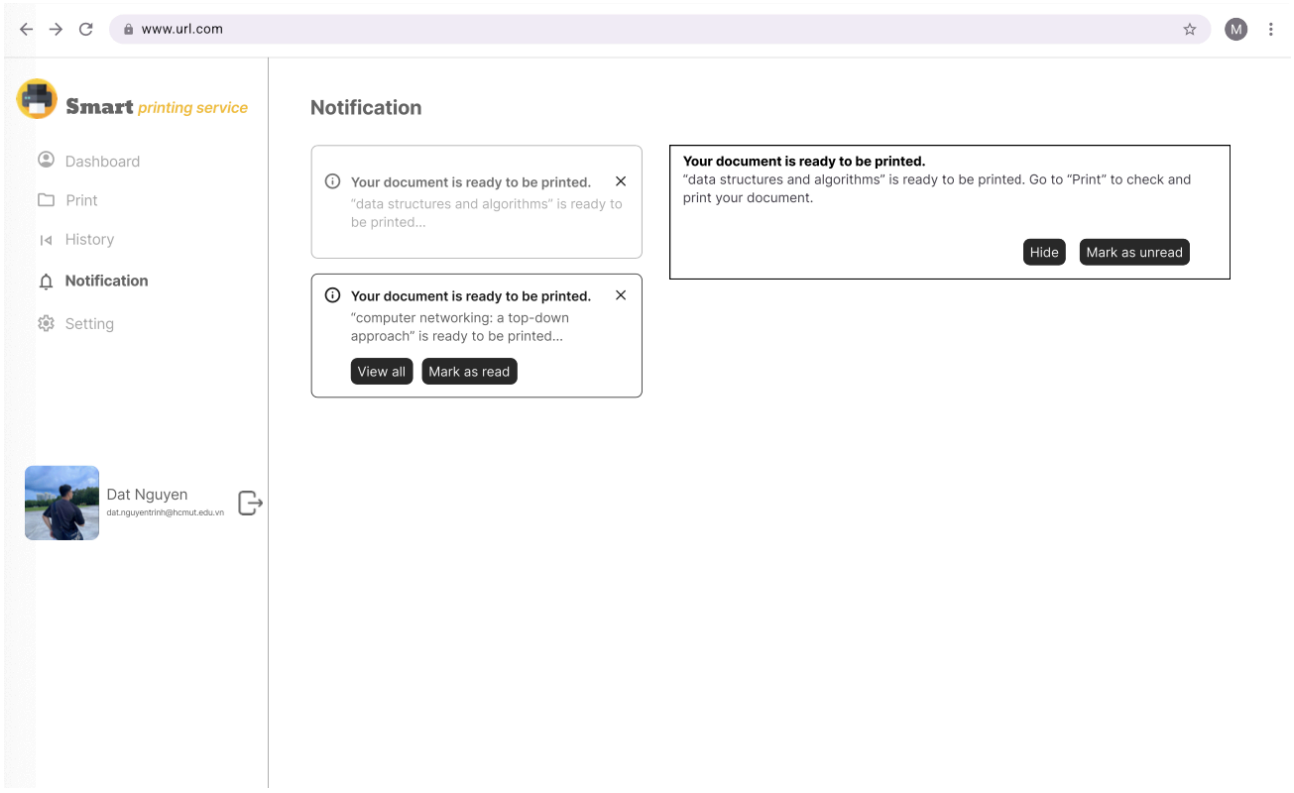

Figure 3.21. Notification

3.4.4. Student Dashboard (Student view)

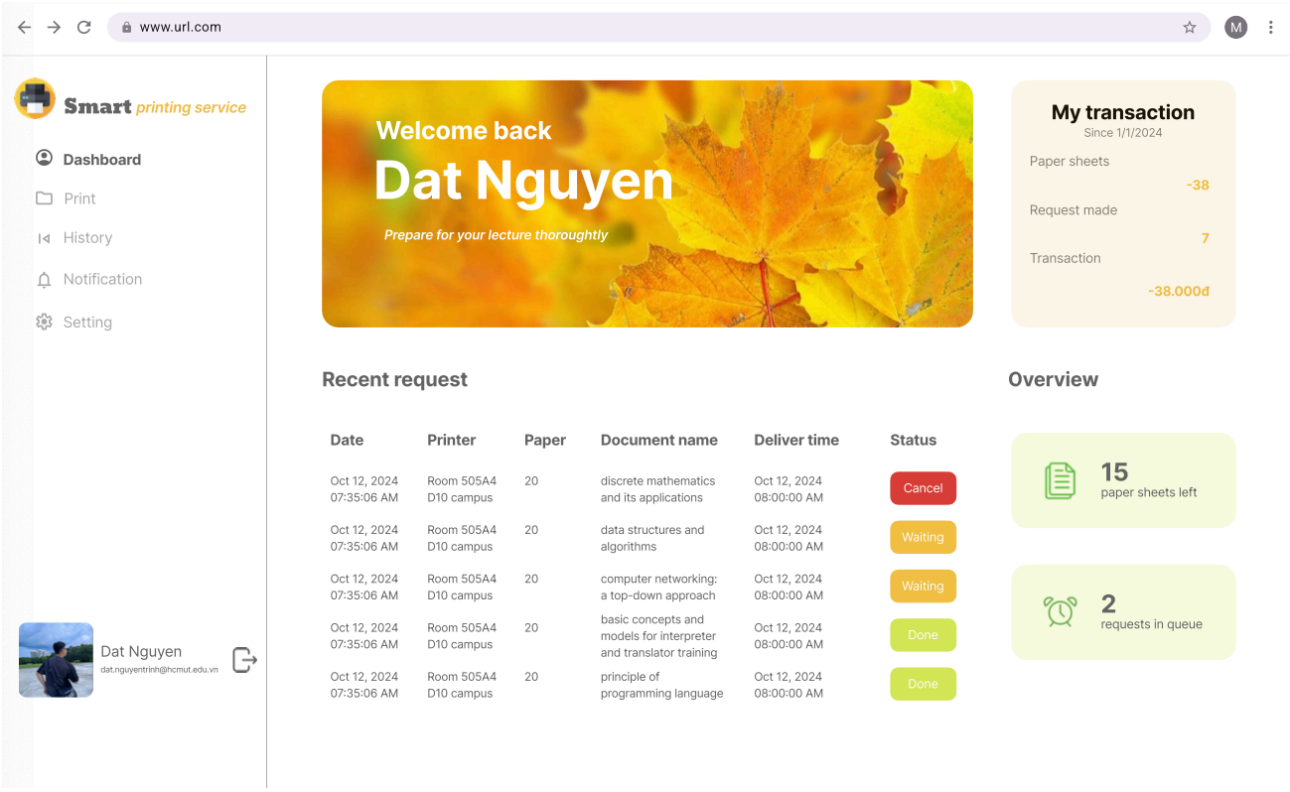

Figure 3.22. Student Dashboard

3.4.5. Print service (Student view)

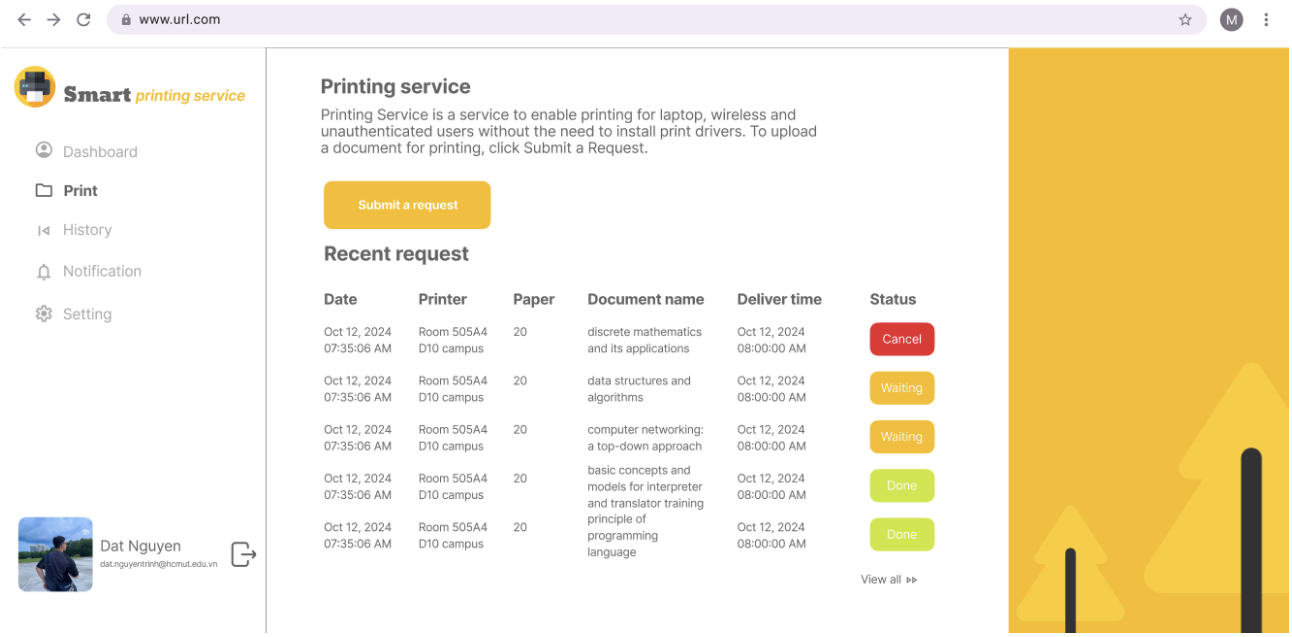

Figure 3.23. Print service - Summary

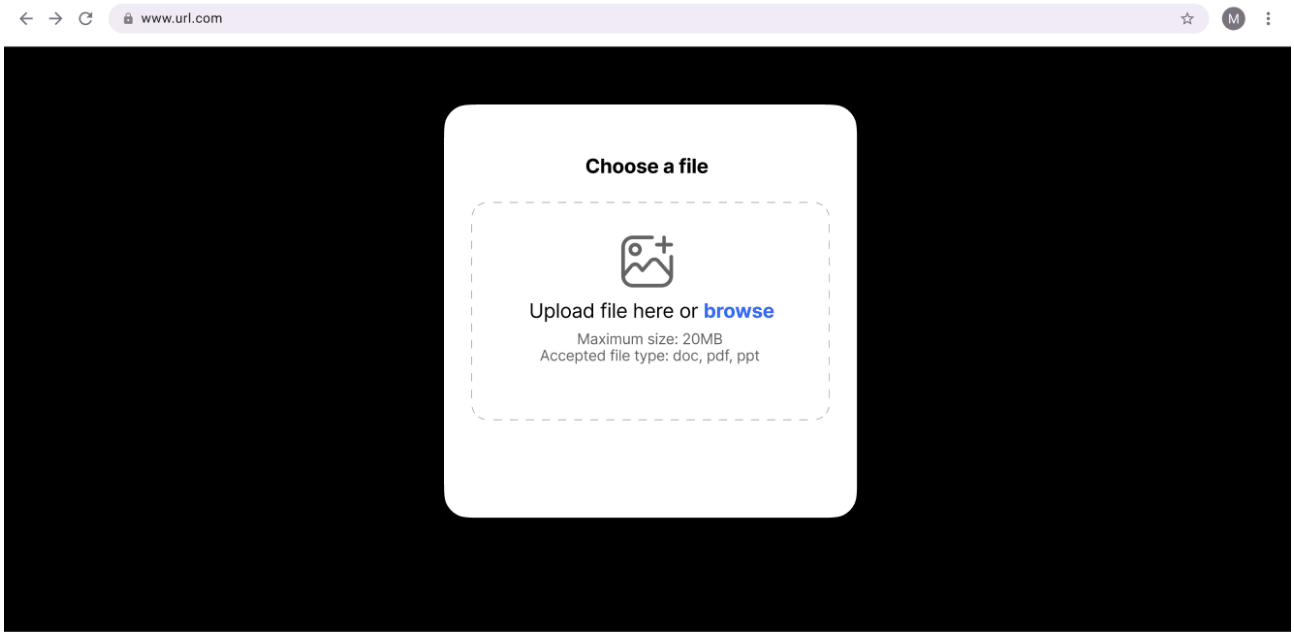

Figure 3.24. Print service - Upload document

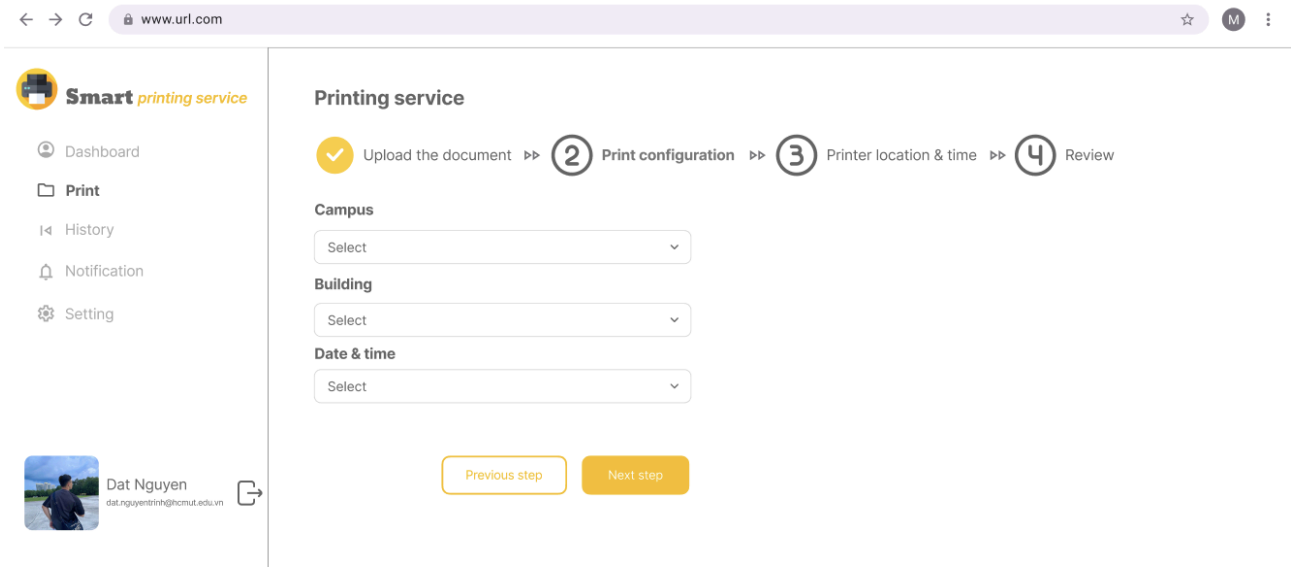

Figure 3.25. Print service - Print configuration

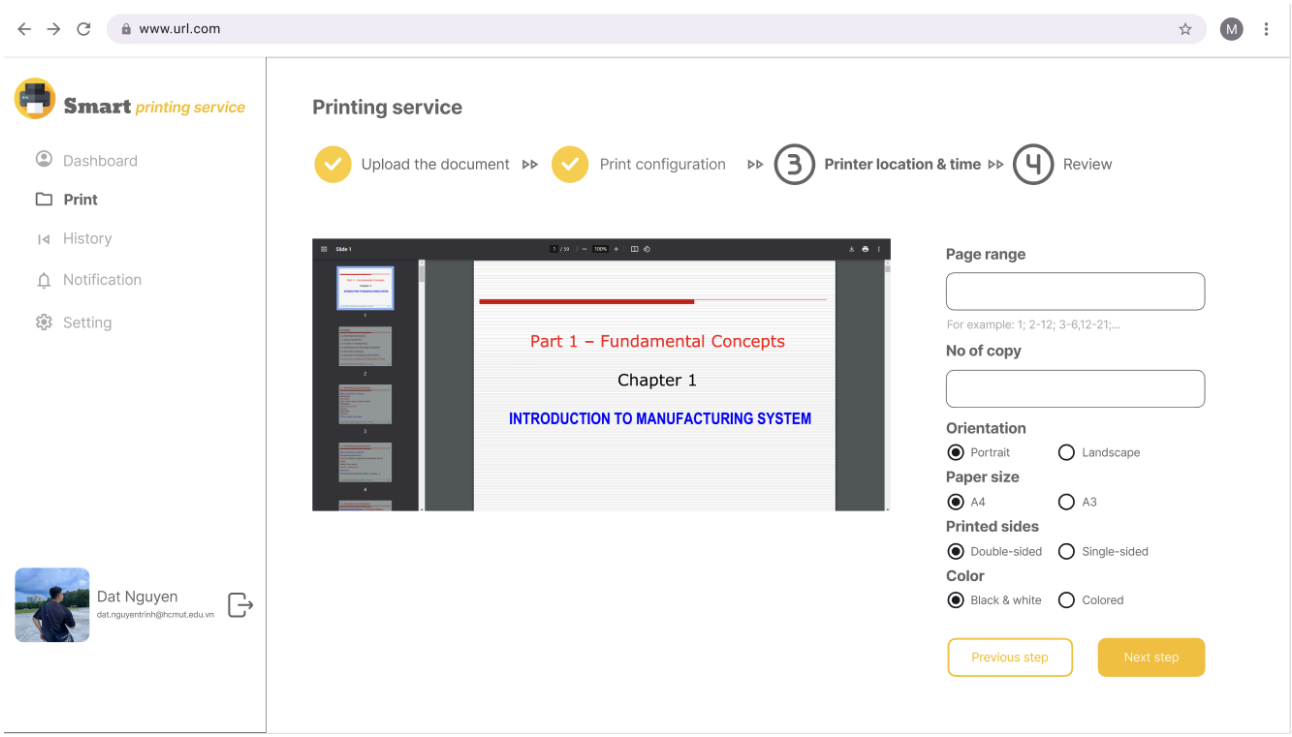

Figure 3.26. Print service - Printer location & time

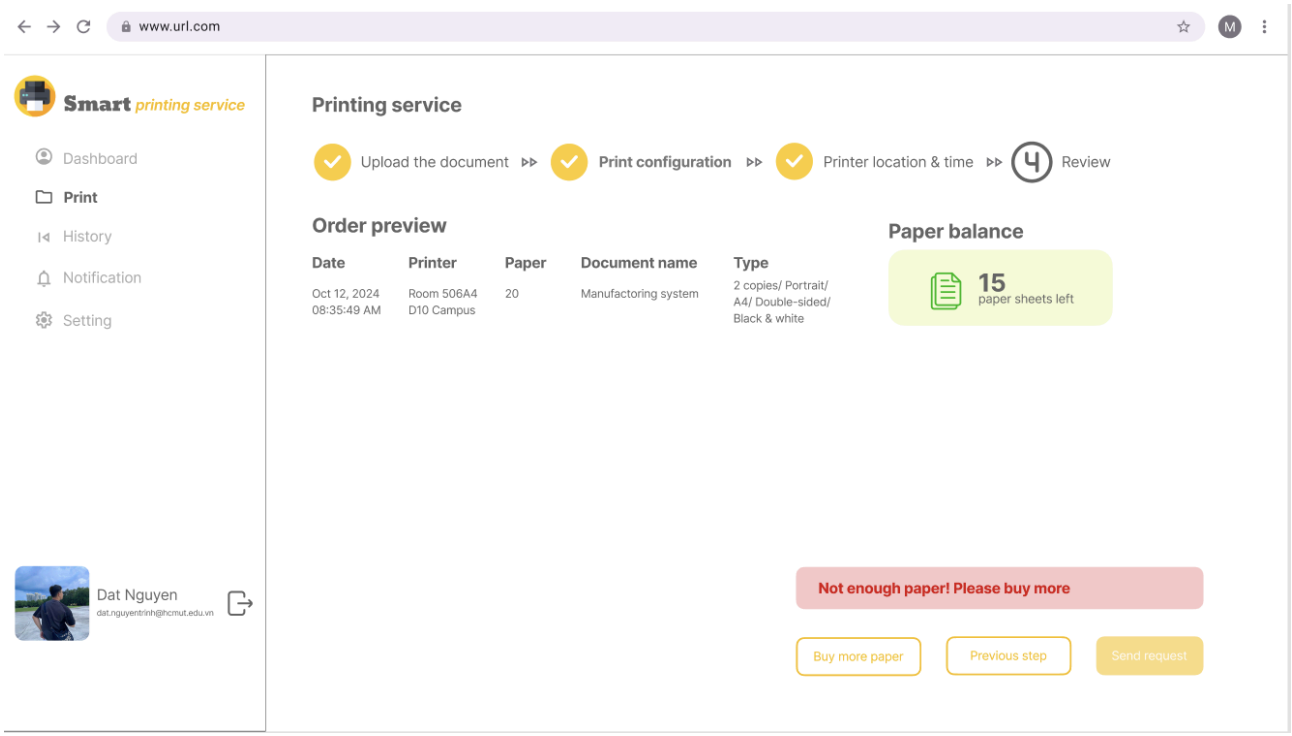

Figure 3.27. Print service - Review

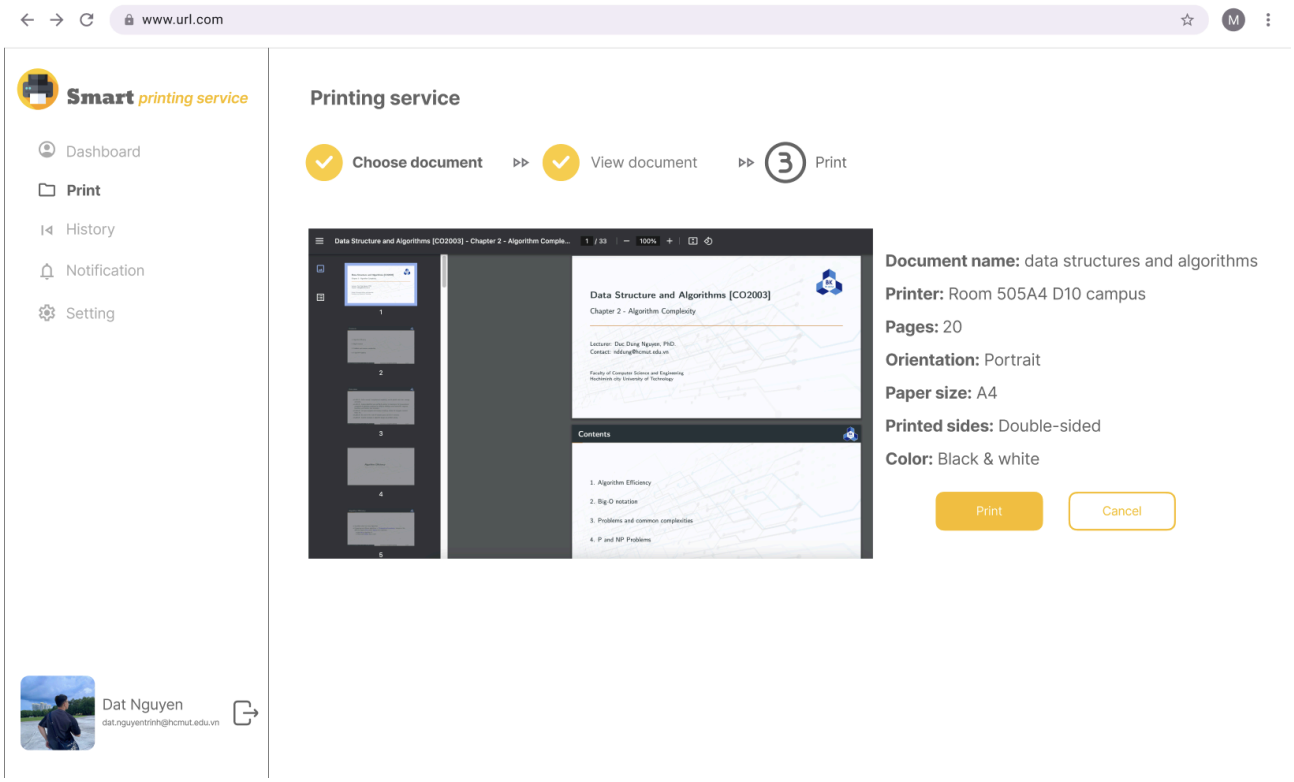

Figure 3.28. Print service - Choose a document

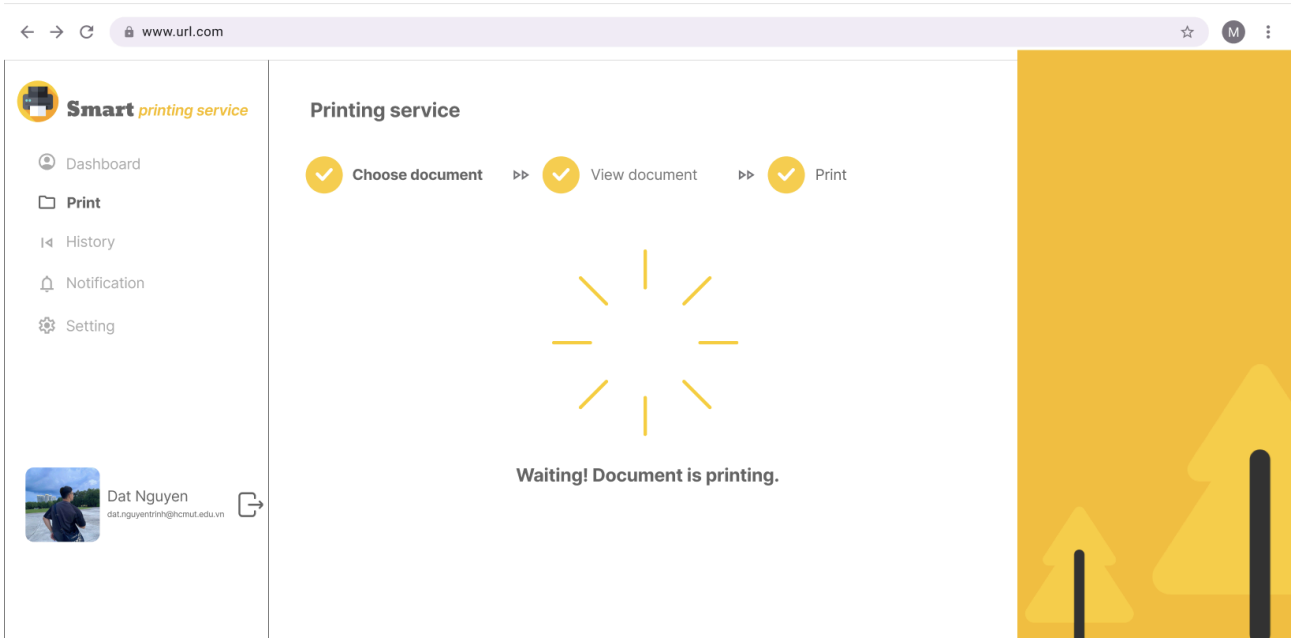

Figure 3.29. Print service - Waiting for print

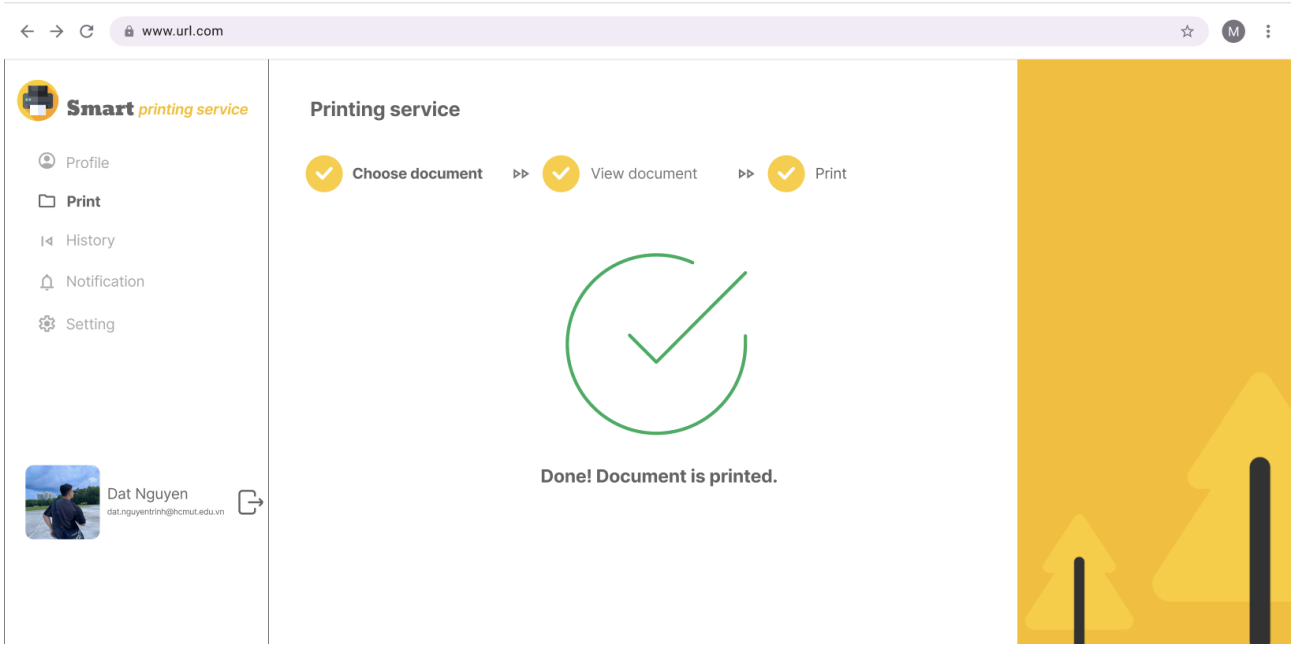

Figure 3.30. Print service - Finish print

3.4.6. Printing log (Student view)

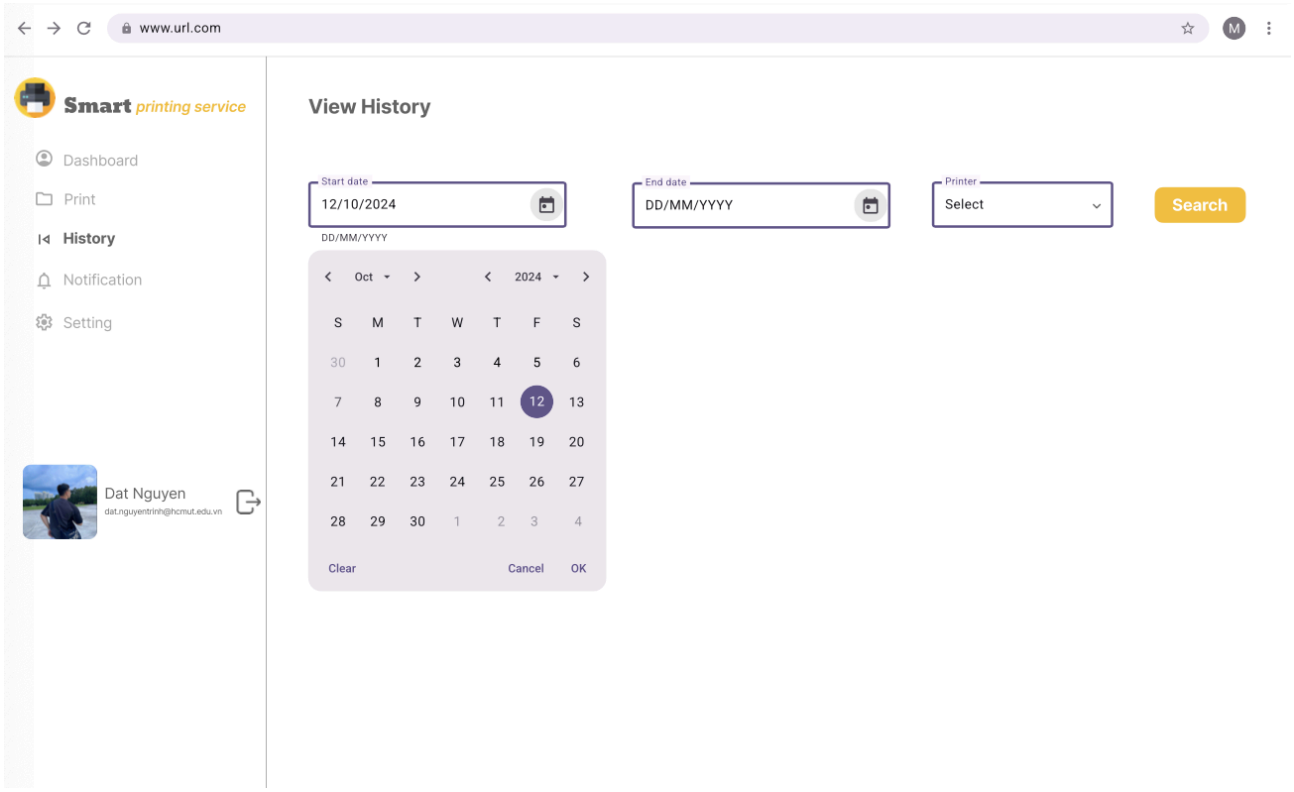

Figure 3.31. Printing log - Filter criteria

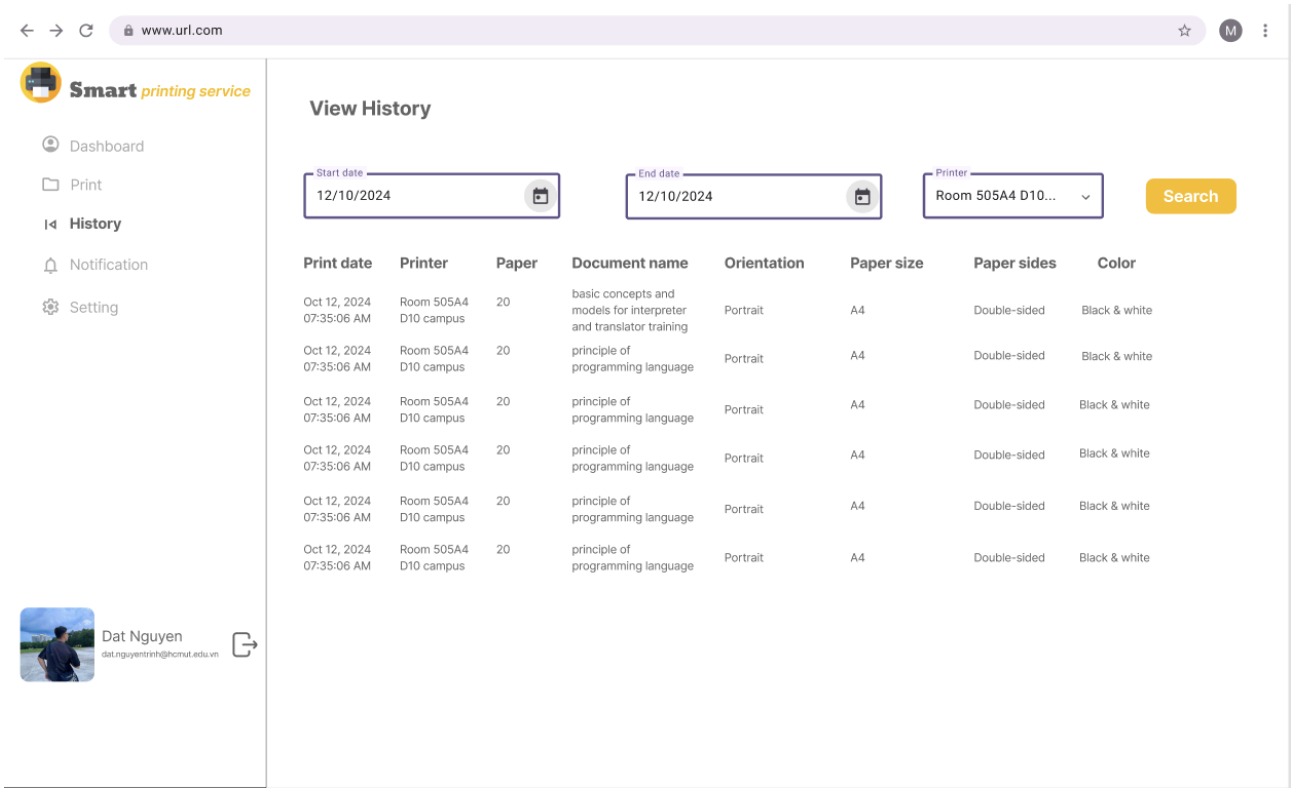

Figure 3.32. Printing log - View printing log

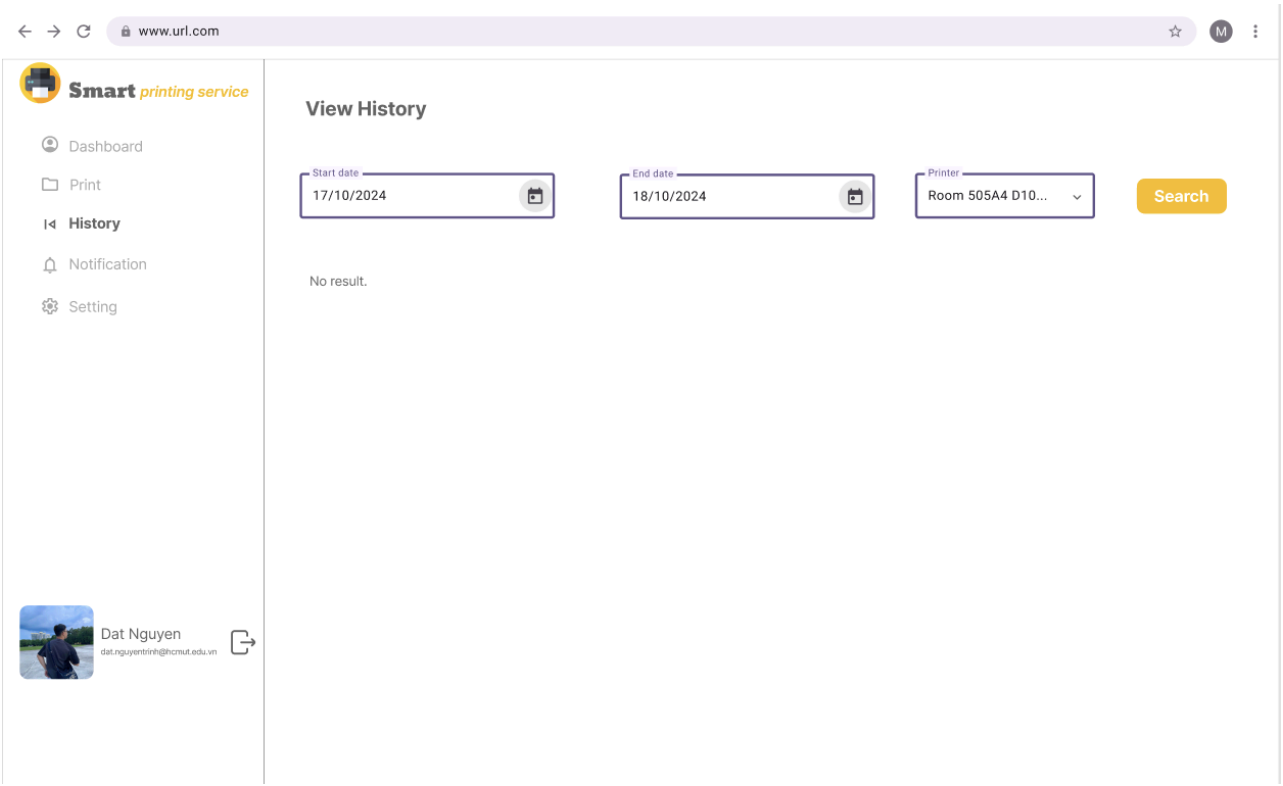

Figure 3.33. Printing log - No data match the criteria

3.4.7. Setting (Student view)

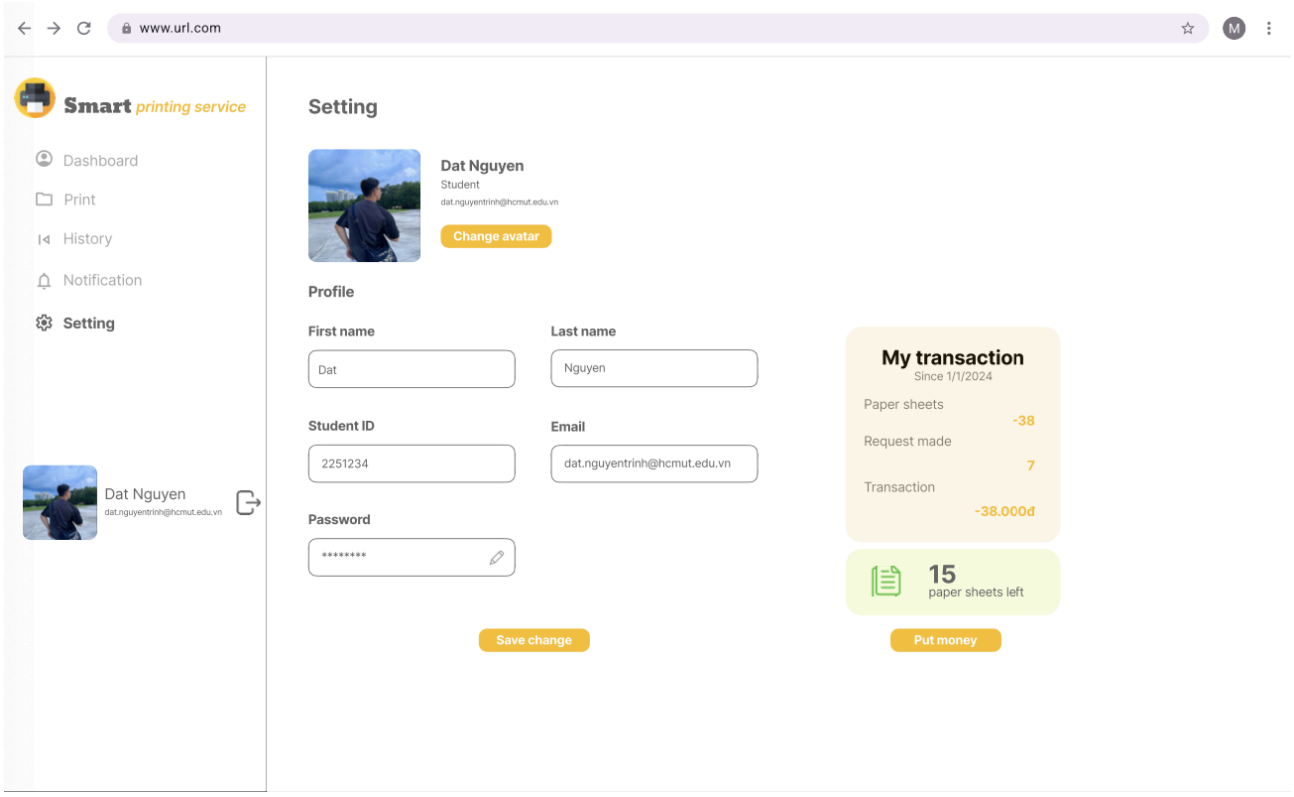

Figure 3.34. Setting - Summary

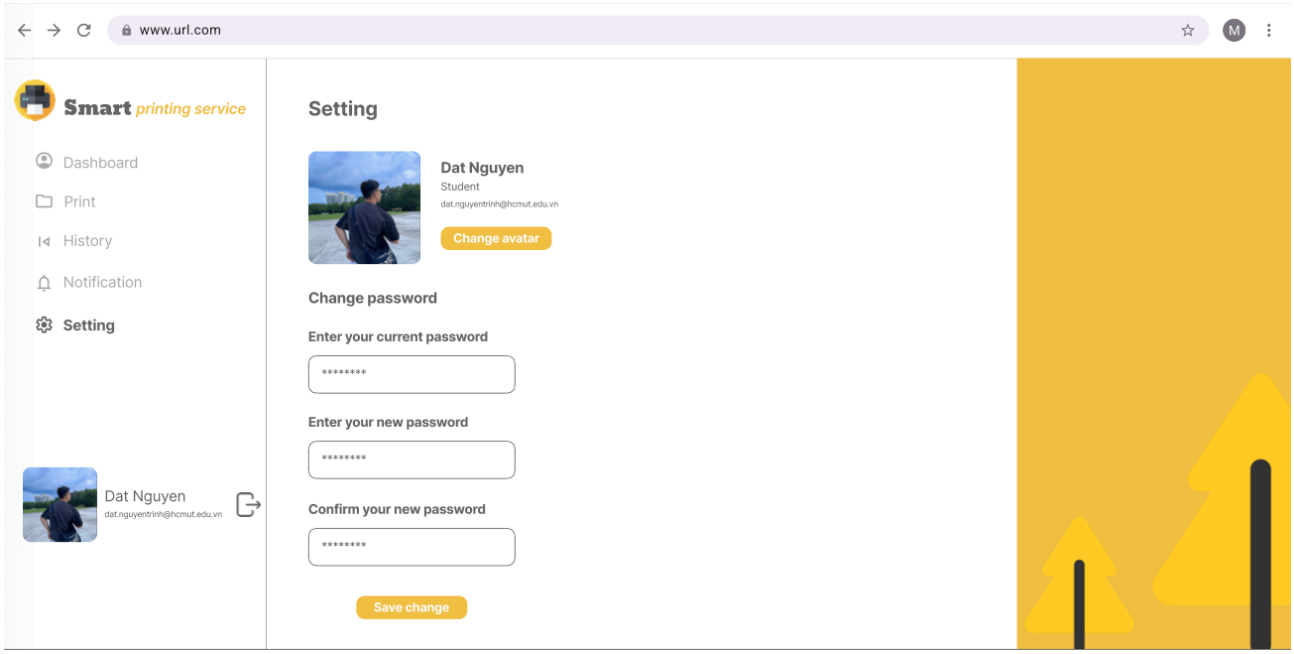

Figure 3.35. Setting - Change password

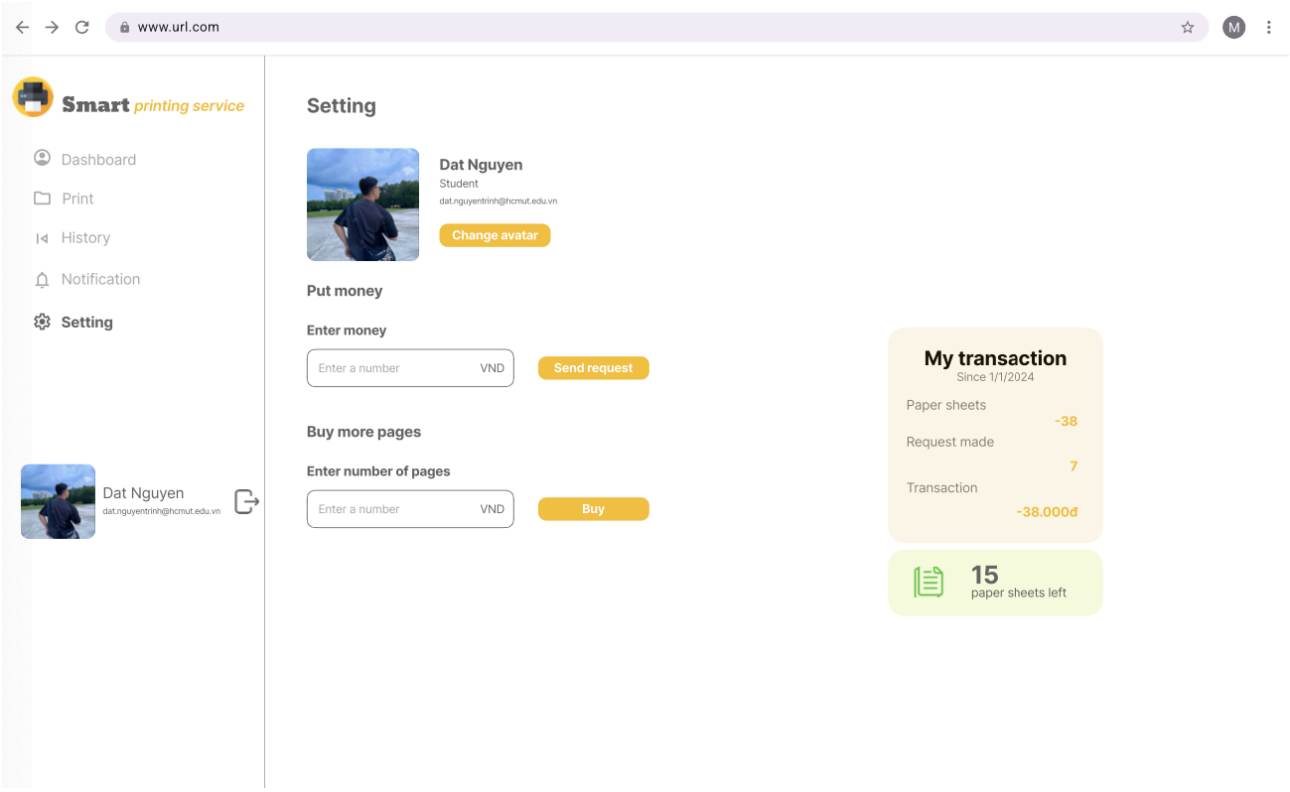

Figure 3.36. Setting - Put money & Buy pages

3.4.8. SPSO Dashboard (SPSO view)

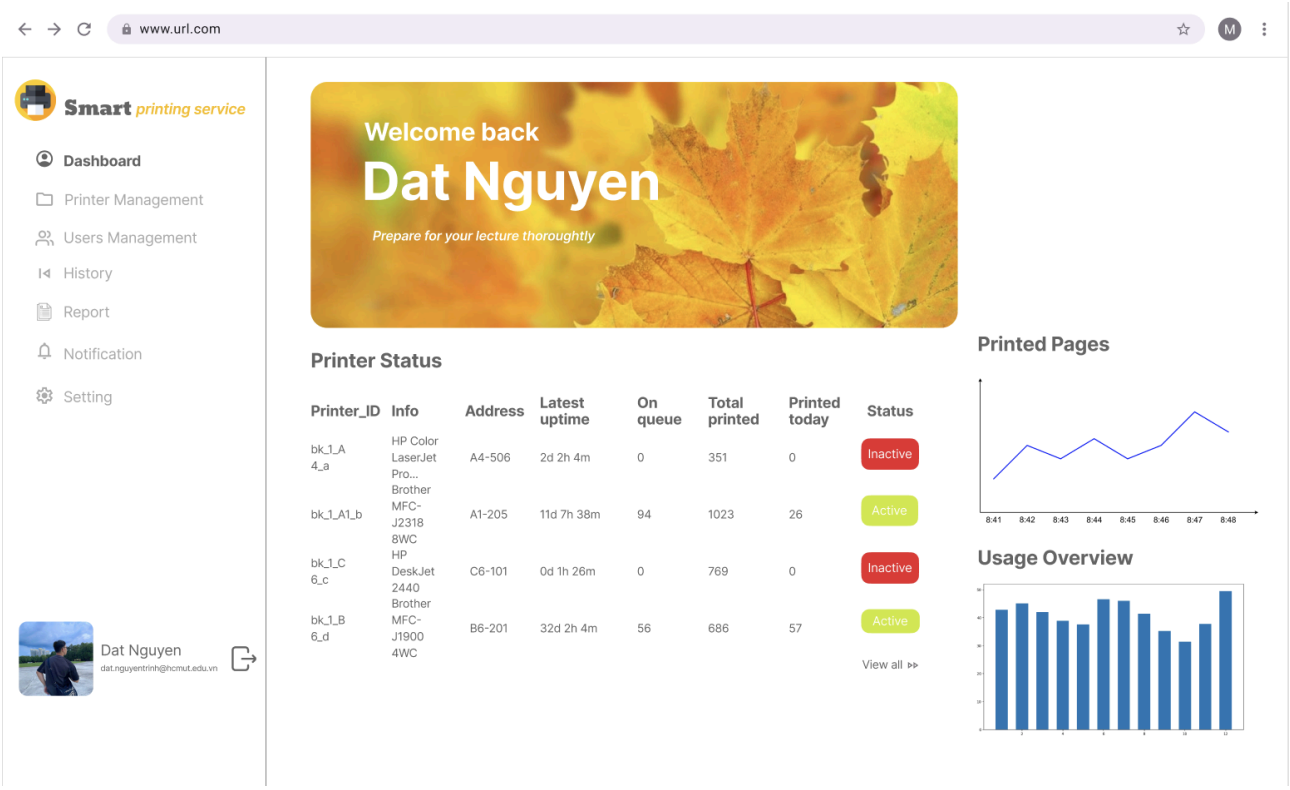

Figure 3.37. SPSO Dashboard

3.4.9. Printers management (SPSO view)

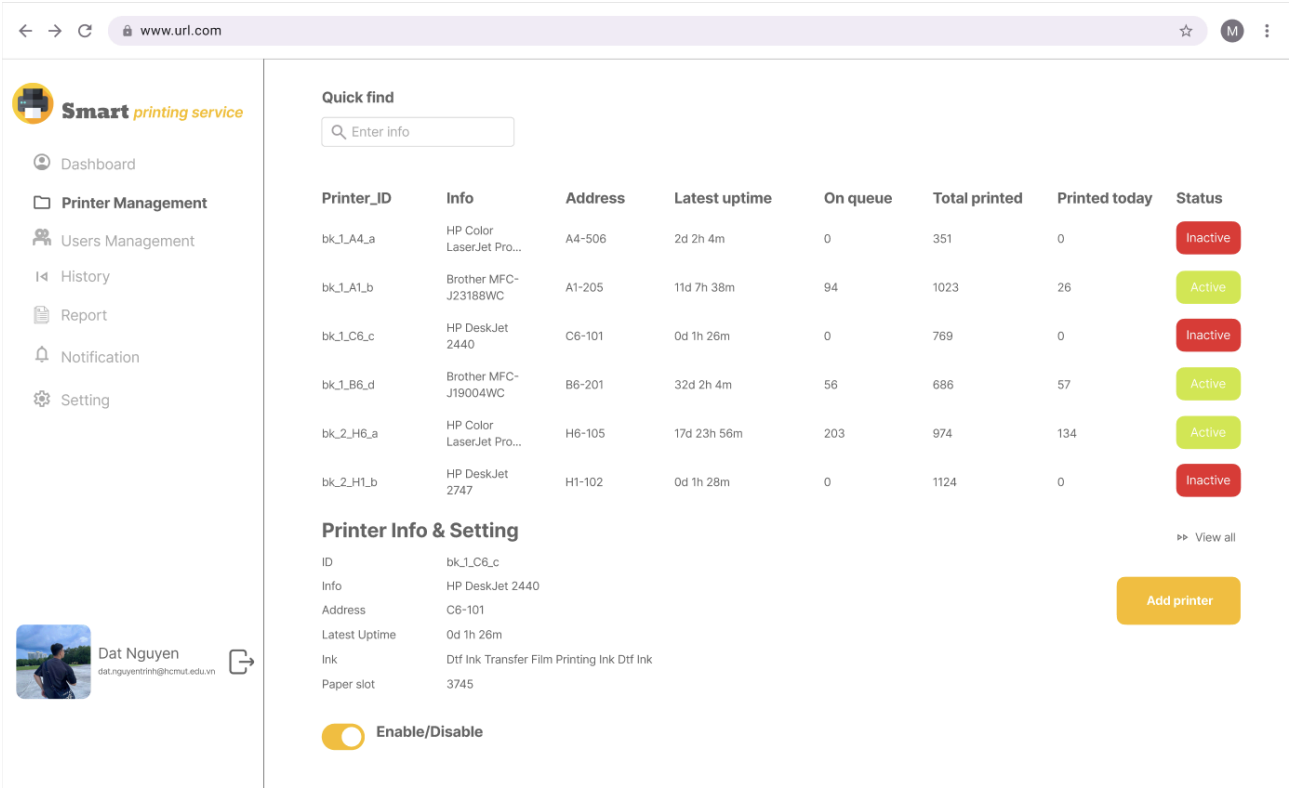

Figure 3.38. Printers management - Summary

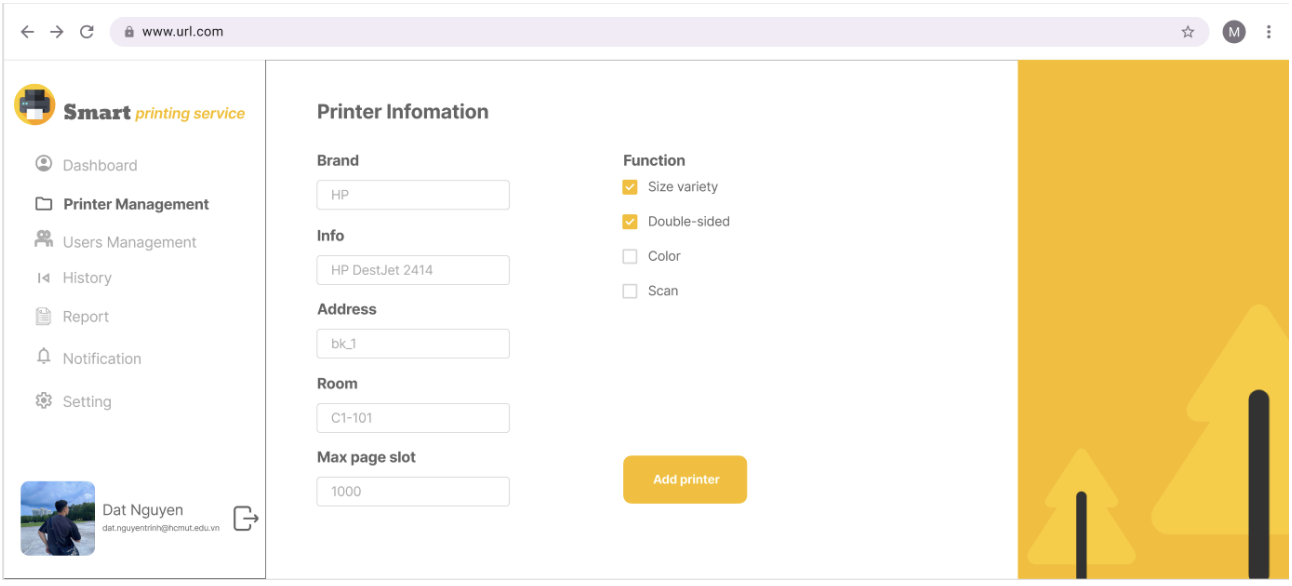

Figure 3.39. Printers management - Add a printer

3.4.10. Users management (SPSO view)

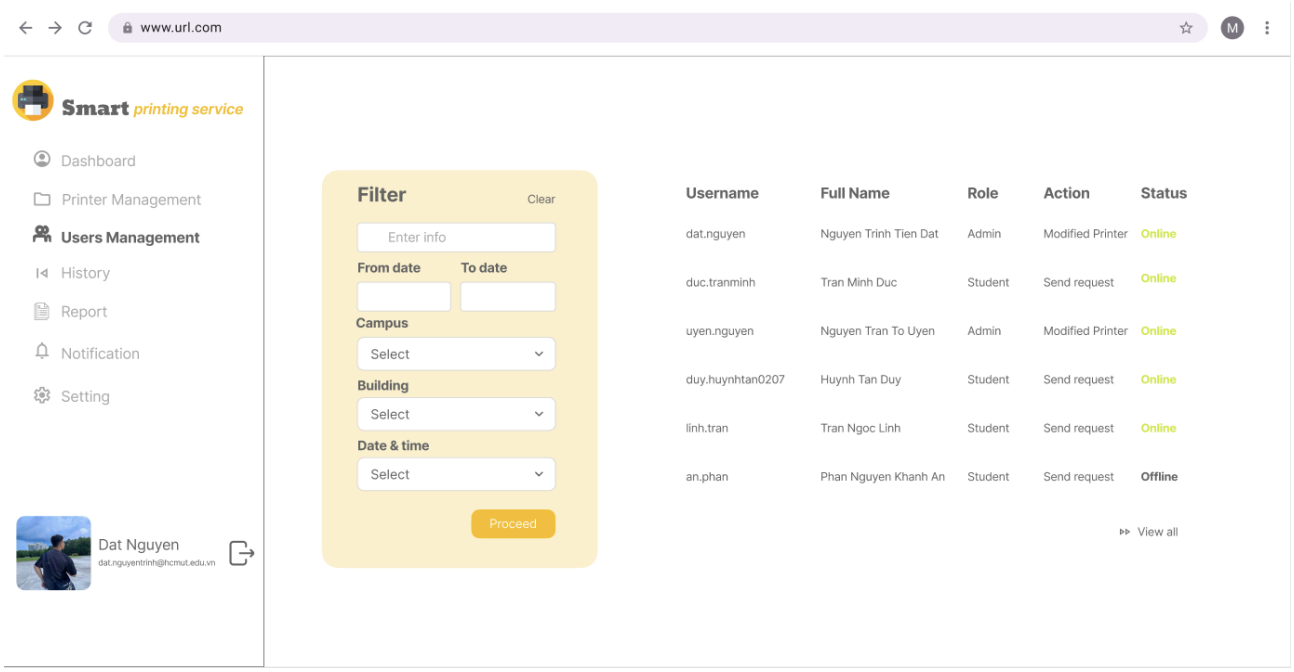

Figure 3.40. Users management

3.4.11. Printing history (SPSO view)

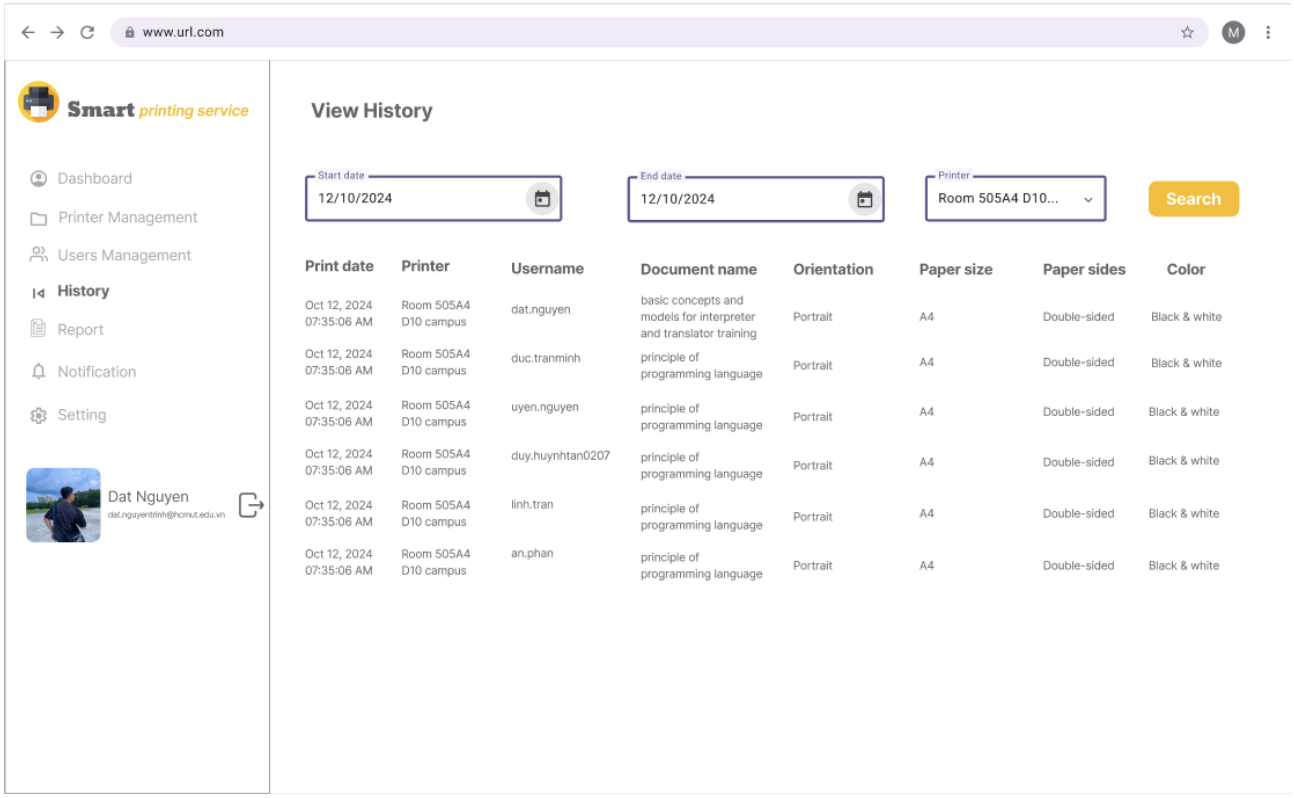

Figure 3.41. View printing history

3.4.12. Generate report (SPSO view)

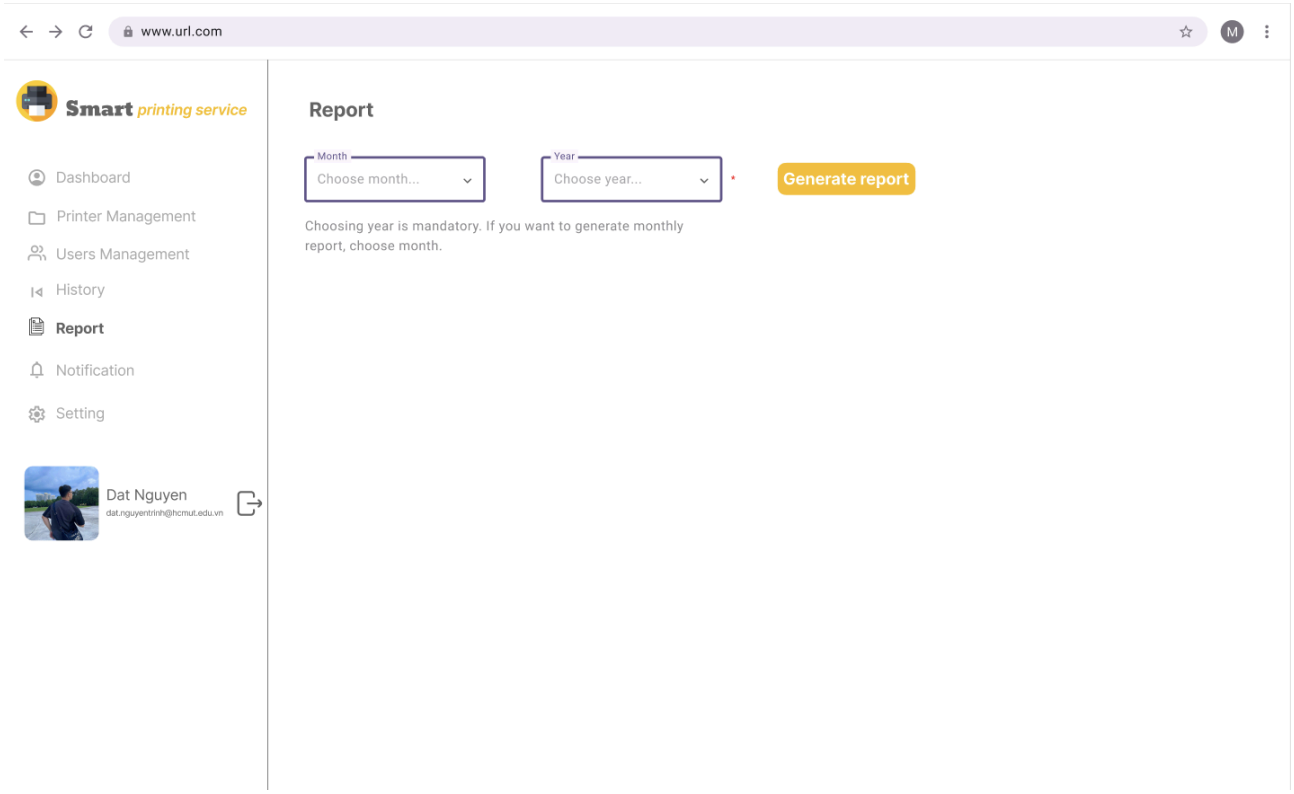

Figure 3.42. Generate report

4. ARCHITECTURE DESIGN

4.1. Layered architecture definition

4.1.1. Architectural diagrams

Integration with external services such as Google Drive, and BKPay enriches the functionality and utility of the system, providing users with an enhanced and comprehensive experience.

The layered architecture of the SSPS can be presented as follow:

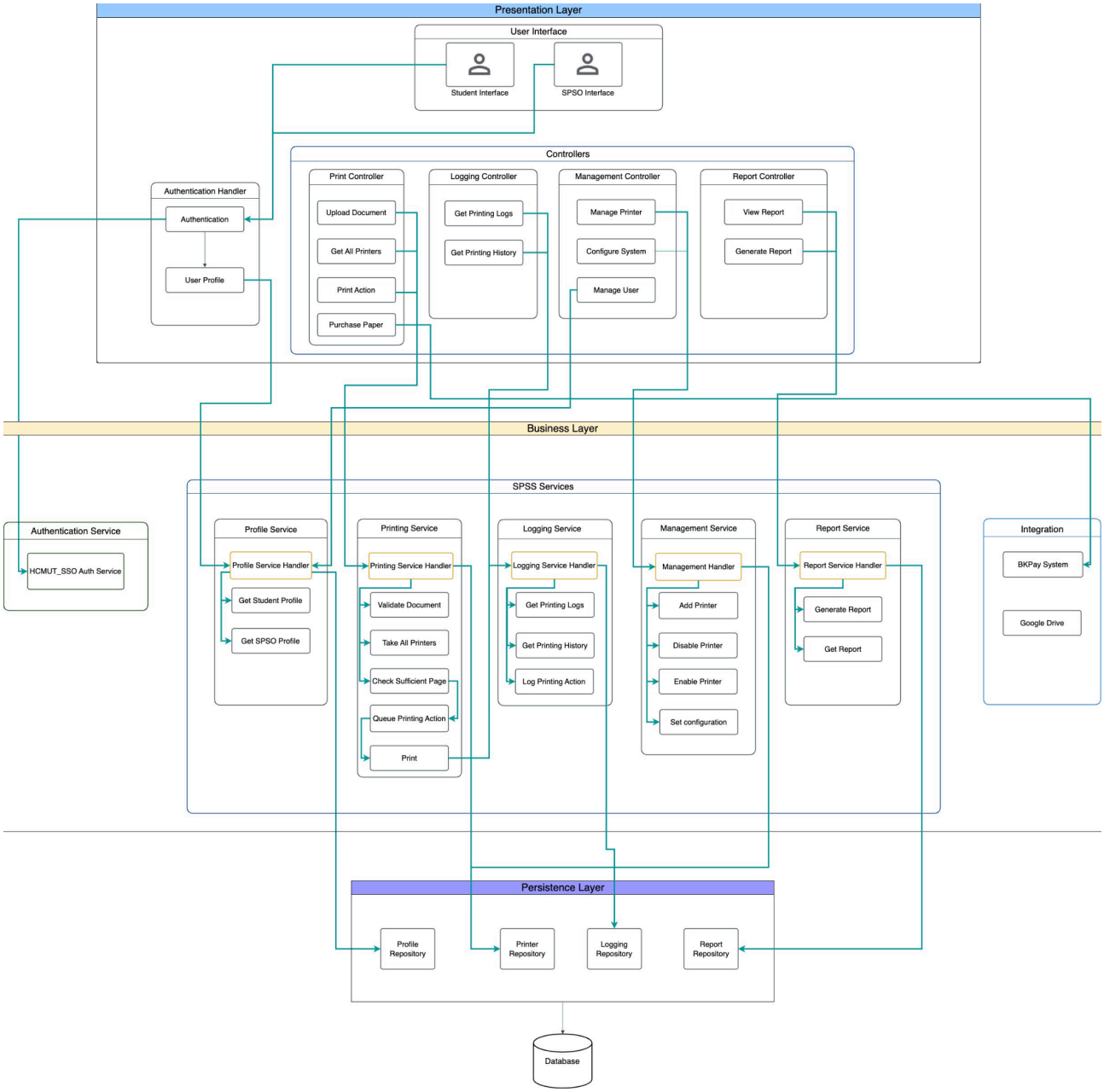

Figure 4.1. Architectural diagram

Our diagram will consist of 4 layers: Presentation, Business, Persistence, and Database layer.

Presentation layer

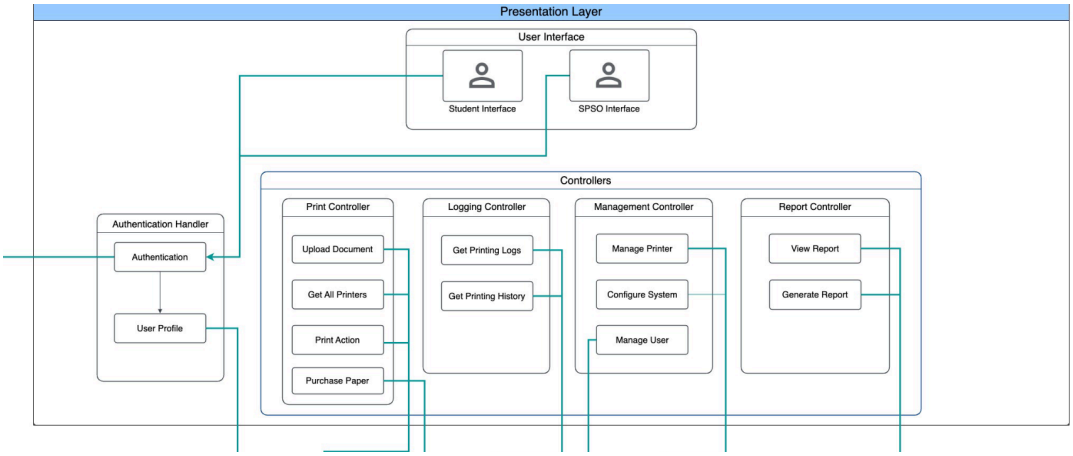

Figure 4.2. Presentation layer

The User Interface will contain 2 specific interfaces:

- Student Interface: allows students to interact with the printing system.

- SPSO Interface: allows officers to manage the infrastructure of the printing system.

Authentication handler will handle the event when Student or SPSO login. The controller is responsible for handling requests from the users. In the diagram above, there are 4 types of controllers according to the functionality:

- Print controller: receive requests related to printing view.

- Logging controller: receive requests related to the history of printing and the dashboard of users.

- Management controller: receive requests from SPSO to manipulate and configure the printing system.

- Report controller: receive requests from SPSO to view and generate report.

Business layer

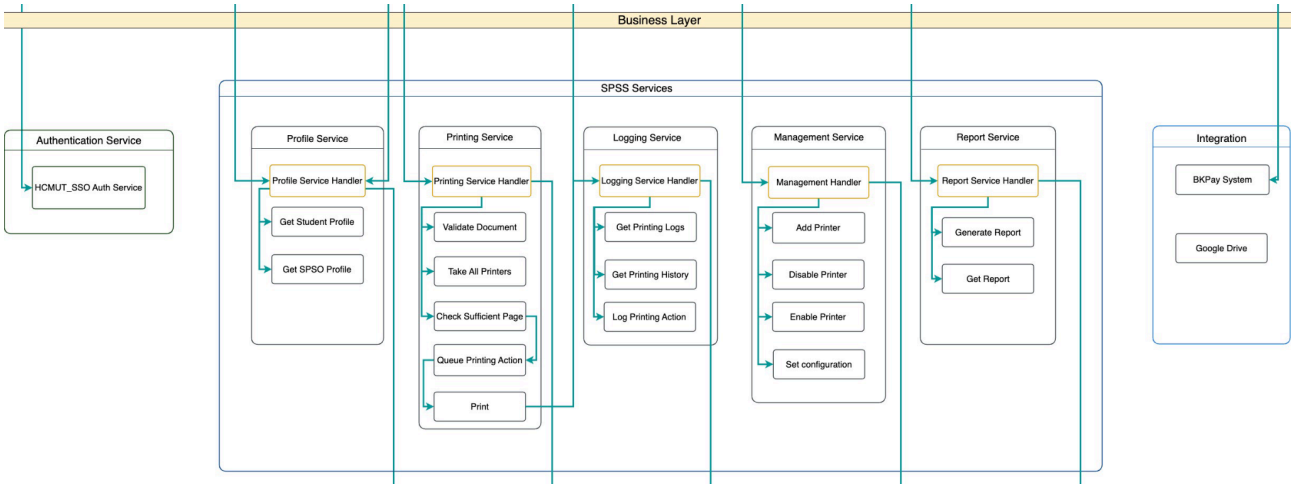

Figure 4.3. Business layer

The business layer in a layer architecture diagram handles the core logic and processing of application data, bridging the communication between the presentation and persistence layers.

- Authentication service: manages user authentication request from authentication handler.

- Profile Service: processes requests from the authentication handler after user authentication has completed.

- Printing Service: manages action related to printing, including validation, printer retrieval, job queue, printing tasks.

- Logging Service: is responsible for printing logs retrieval and processing requests from logging controller and printing service.

- Management Service: manages actions regarding printer manipulation and other system configurations.

- Report Service: processes requests from report controller in order to retrieve and generate new reports.

Persistence layer and Database

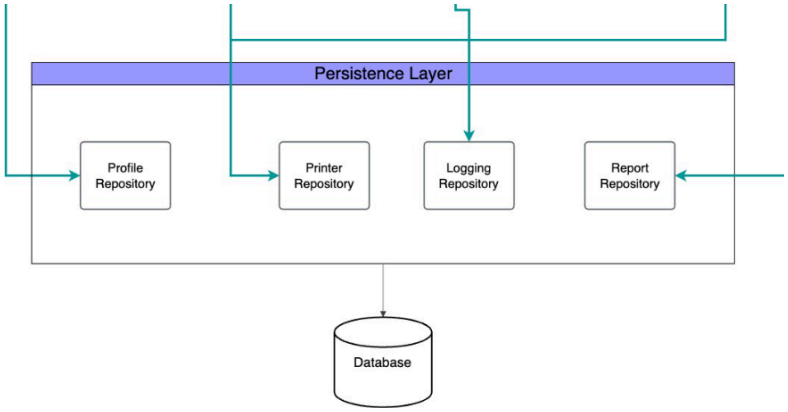

Figure 4.4. Persistence layer and Database

This layer acts as a bridge between the database layer and business layer of SSPS.

- Profile Repository: connects and maps user's profile to the database system.

- Printer Repository: ensures data of printer be retrieved and stored properly.

- Logging Repository: maintains data related to system logs and activities.

- Report Repository: handles the retrieval and storage of report-related data.

4.1.2. Presentation strategy

We created a website for SPSS as the graphical user interface. After successfully logging in, the user always sees the sidebar on all pages. Each item on the sidebar represents a feature of the system. By clicking on any item on the sidebar, the user will be redirected to the corresponding page. There are 2 types of sidebar based on the role of the user: student's sidebar and SPSO's sidebar (Figure 4.5).

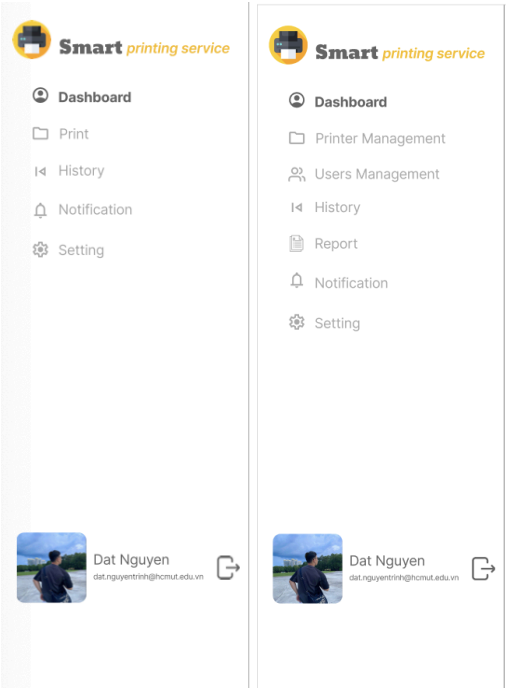

Figure 4.5. Student's sidebar and SPSO's sidebar

Student interface

The dashboard page displays a summary of recent printing requests, the user's transactions, and a paper overview (Figure 4.6).

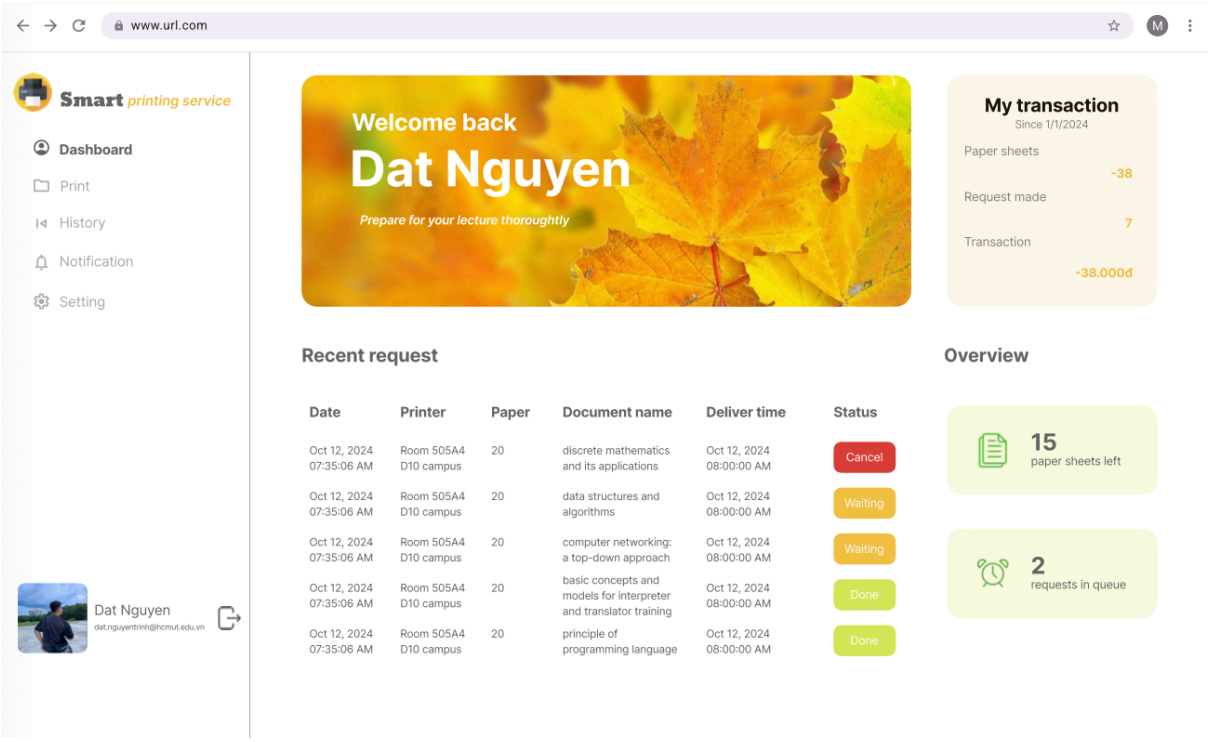

Figure 4.6. Student Dashboard

The print page shows data on recent printing requests (Figure 4.7). If the user wants to upload more documents, they click the “Submit a request” (Figure 4.7) and then, choose the document they want to upload (Figure 4.8).

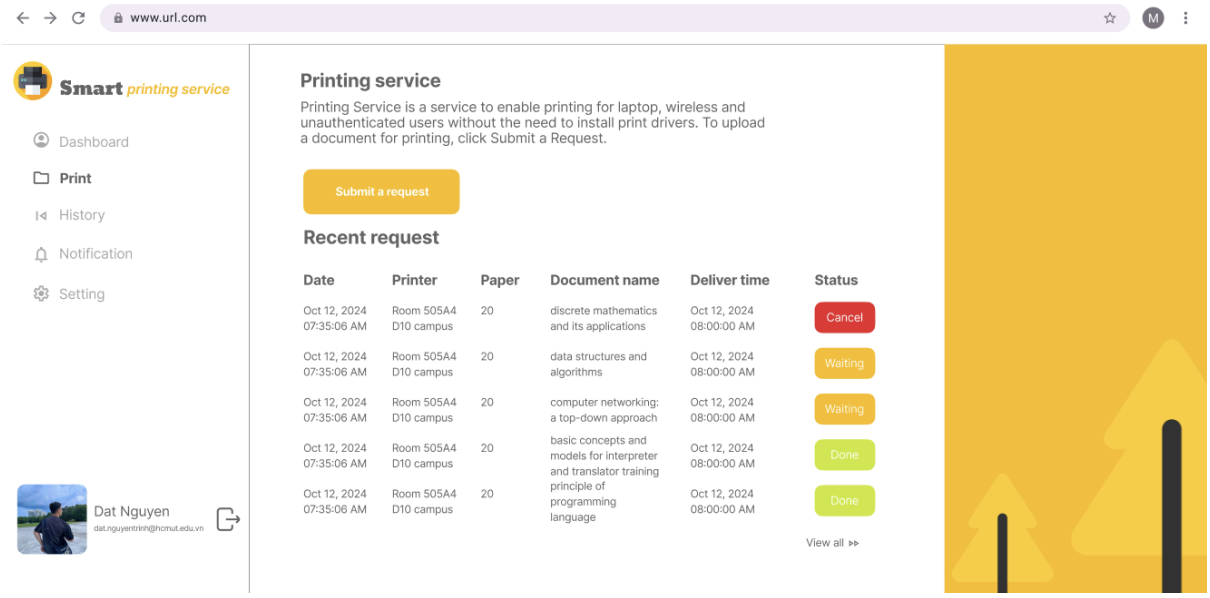

Figure 4.7. Print service - Summary

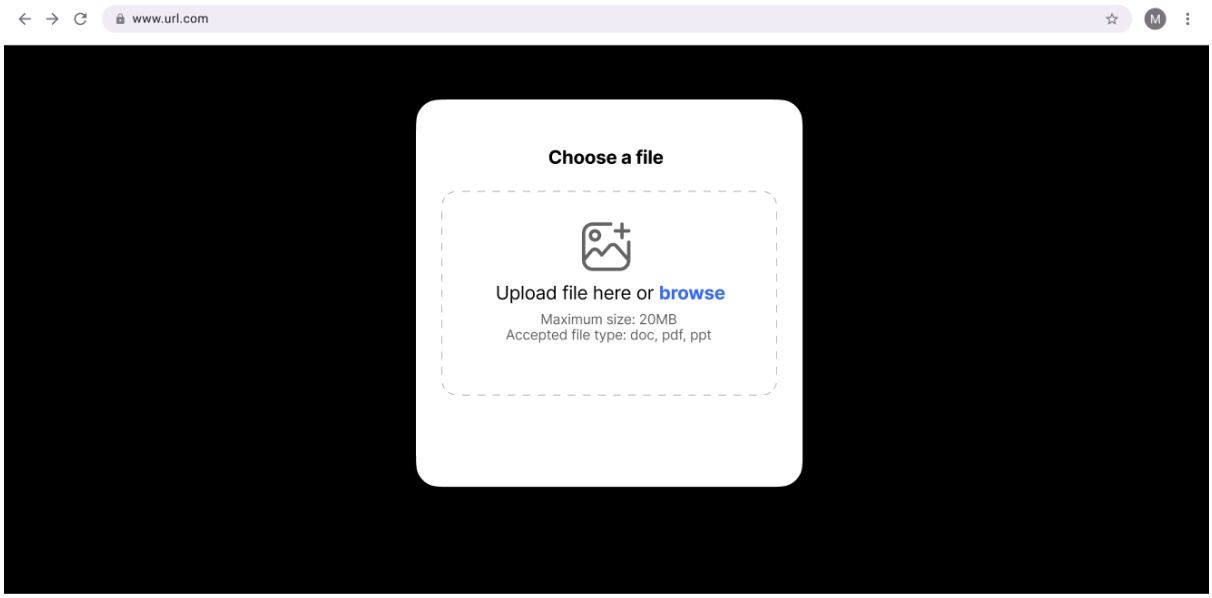

Figure 4.8. Print service - Upload document

After uploading the document, the user the following steps to submit the printing request (Figure 4.9).

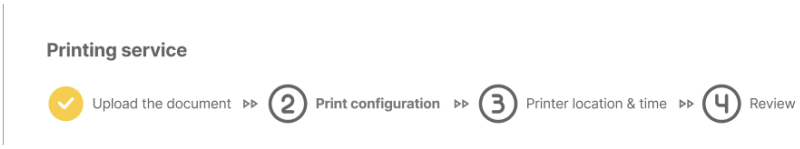

Figure 4.9. Print service - Steps to submit printing request

To print a document, the user chooses the document that they want to print in the Print - Summary page (Figure 4.7) and follows the steps to finish the printing action (Figure 4.10).

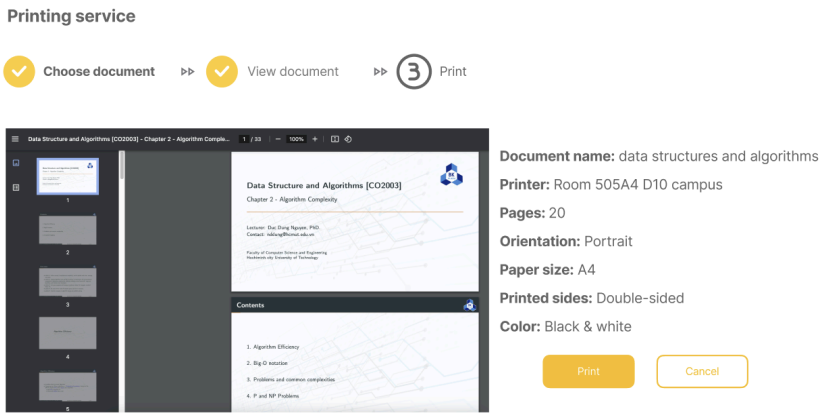

Figure 4.10. Print service - Printing action

On the history page, the user can see their printing log by choosing the start date, end date, and printer. The printing log will be displayed as in Figure 4.11.

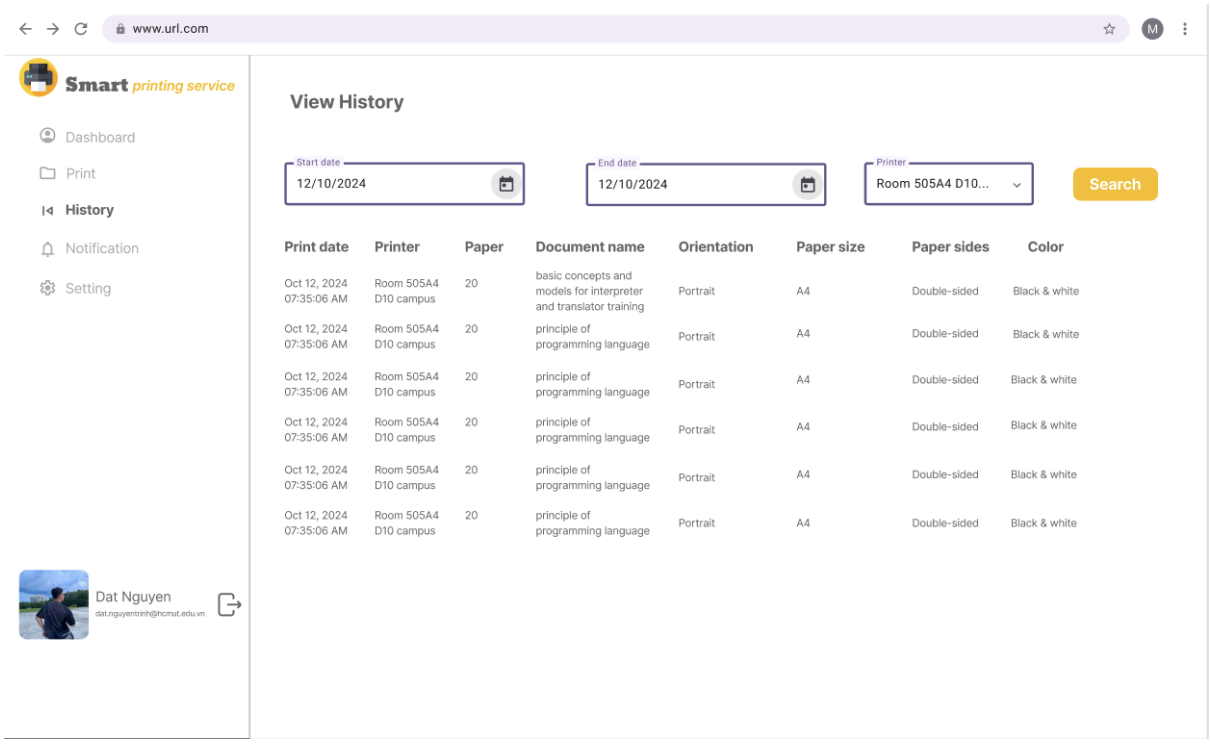

Figure 4.11. Printing log - View printing log

The Notification page includes all the notifications of the user's account and service (Figure 4.12).

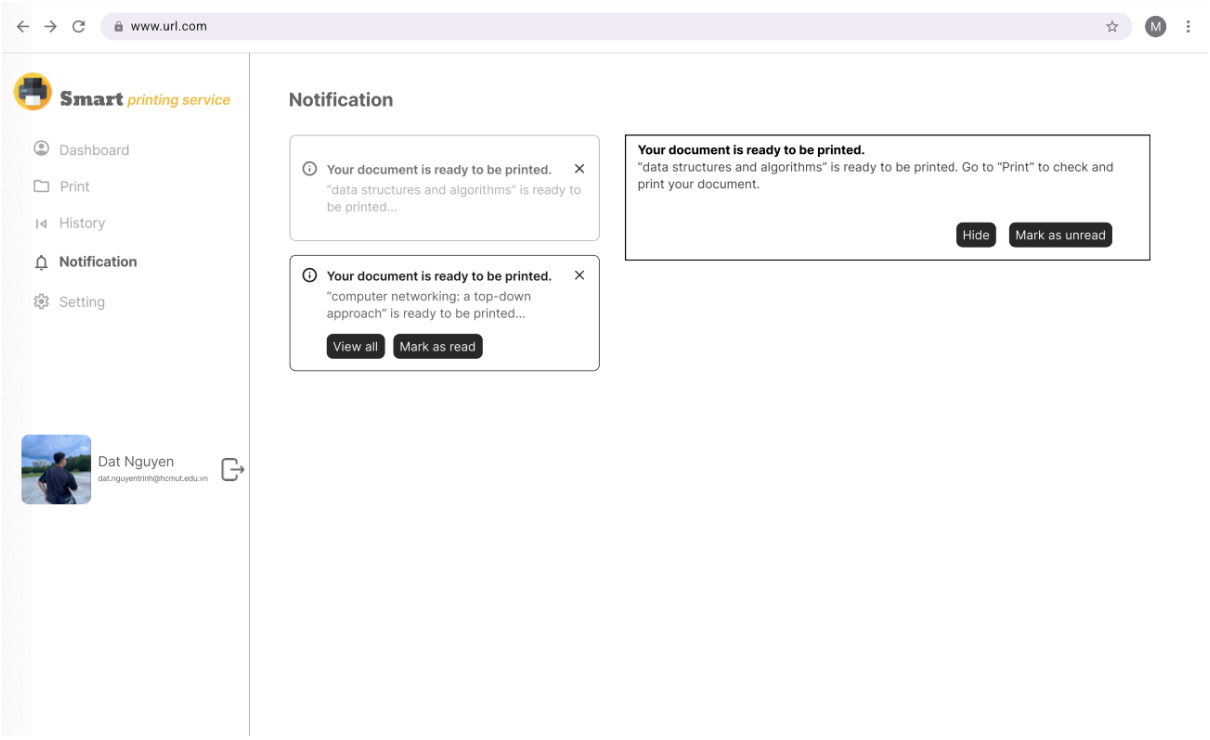

Figure 4.12. Notification

The settings page shows the personal information of the user. The user can change the password or put more money into their transaction (Figure 4.13).

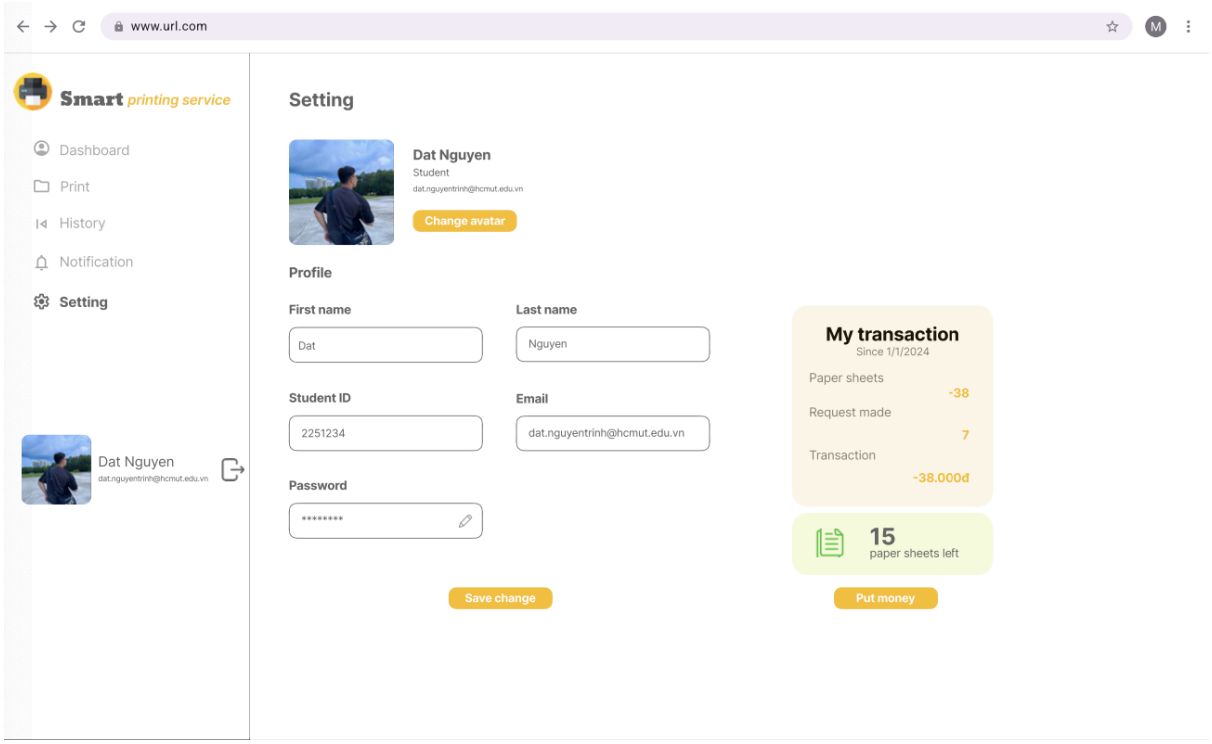

Figure 4.13. Setting - Summary

SPSO interface

The dashboard page of SPSO shows the summary of printer status, graphs of printed pages, and usage overview (Figure 4.14).

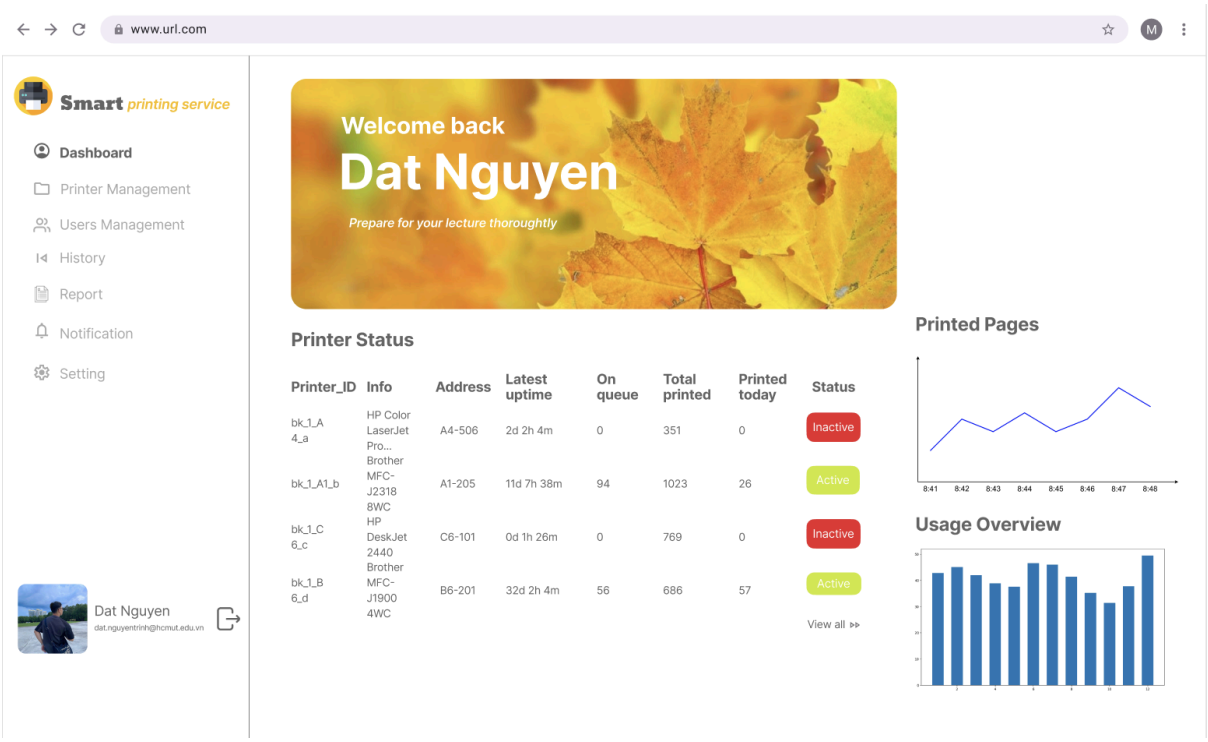

Figure 4.14. SPSO Dashboard

On the printer management page, the SPSO can view the printer status, activate or deactivate any printer (Figure 4.15), and add a printer (Figure 4.16).

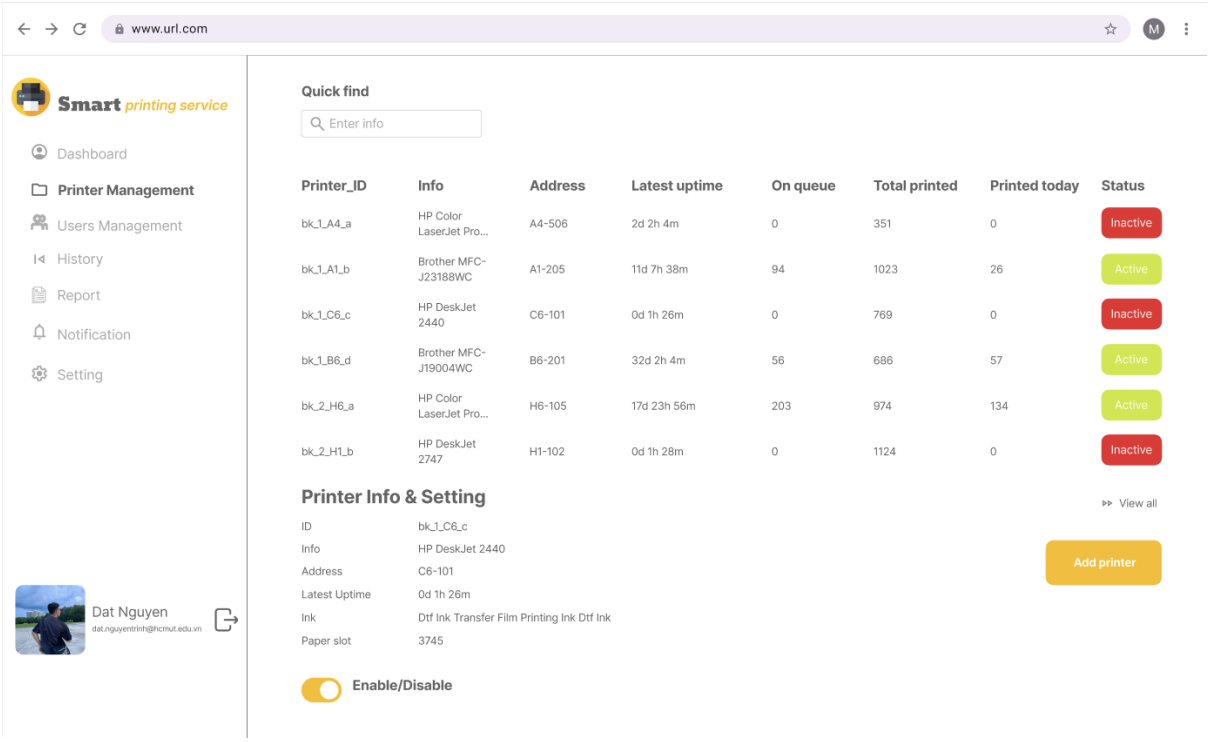

Figure 4.15. Printers management - Summary

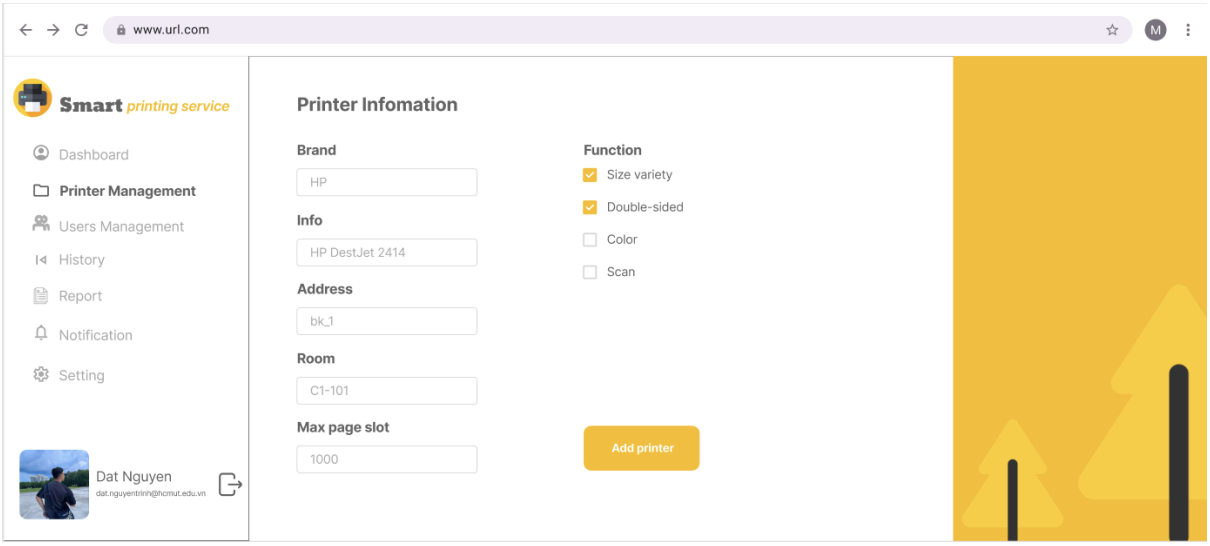

Figure 4.16. Printers management - Add a printer

On the users management page, the SPSO can view all students' information and status as well as manage them (Figure 4.17).

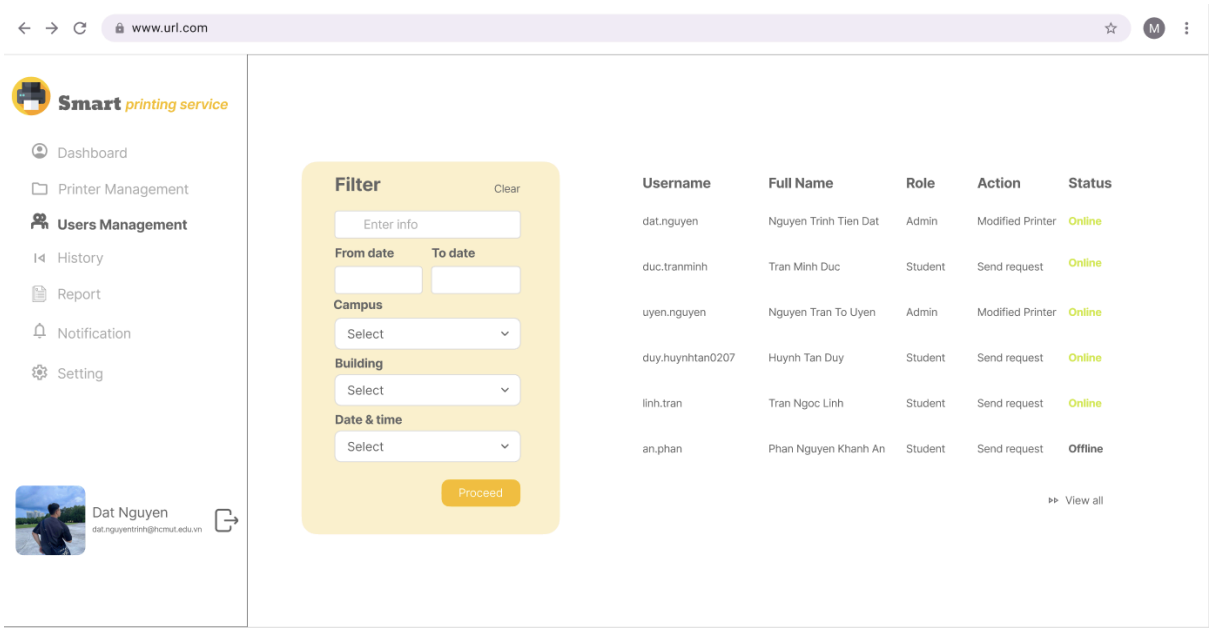

Figure 4.17. Users management

The history page displays the information on all printer's activities in the period of time that the SPSO chooses (Figure 4.18).

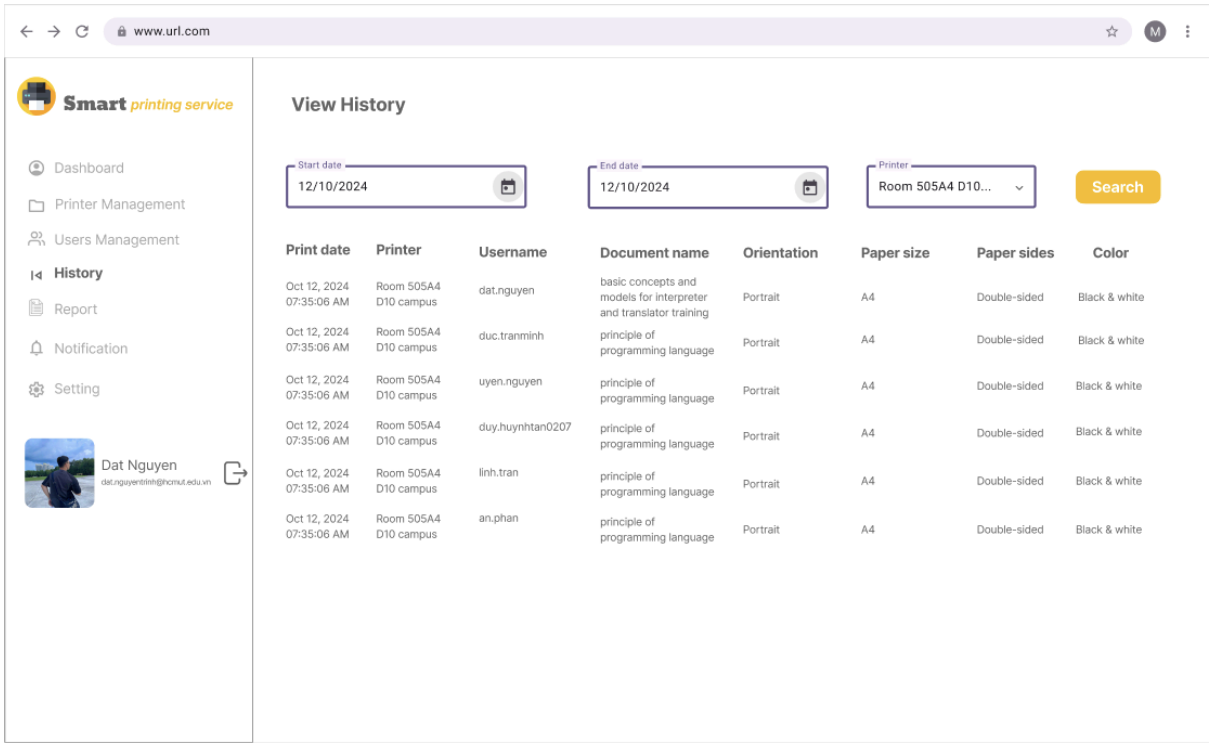

Figure 4.18. View printing history

On the report page, SPSO can let the system generate the reports. The SPSO must select the year, the system will generate a yearly report. They can also select the month to retrieve the monthly report (Figure 4.19).

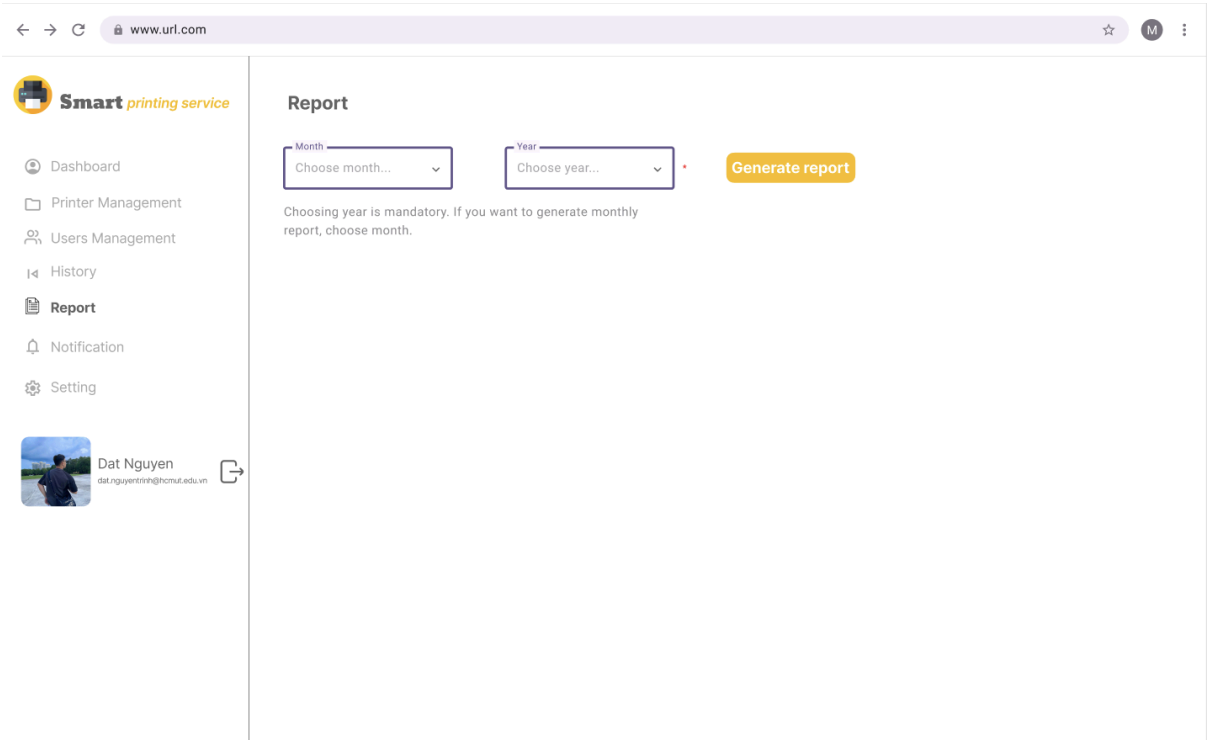

Figure 4.19. Generate report

The notification page of the SPSO is the same as the student's notification page. The setting page in the SPSO interface is similar to the setting page in the student interface, but it does not have the "Put money" feature.

4.1.3. Data Storage/Design

For data storage, our choice is the MongoDB database. MongoDB Atlas serves as the optimal solution for deploying, operating, and scaling MongoDB in the cloud environment. With the capabilities of Atlas, setting up and running a MongoDB database becomes a hassle-free process, achievable with just a few clicks and within a matter of minutes.

Our data structures we will be using are key-values and JSON-formatted. JSON, which stands for JavaScript Object Notation, is a lightweight data interchange format that is easy for humans to read and write, and easy for machines to parse and generate. It is primarily

used to transmit data between a server and a web application. JSON is text-based, making it language-independent, and it consists of key-value pairs arranged in a simple.

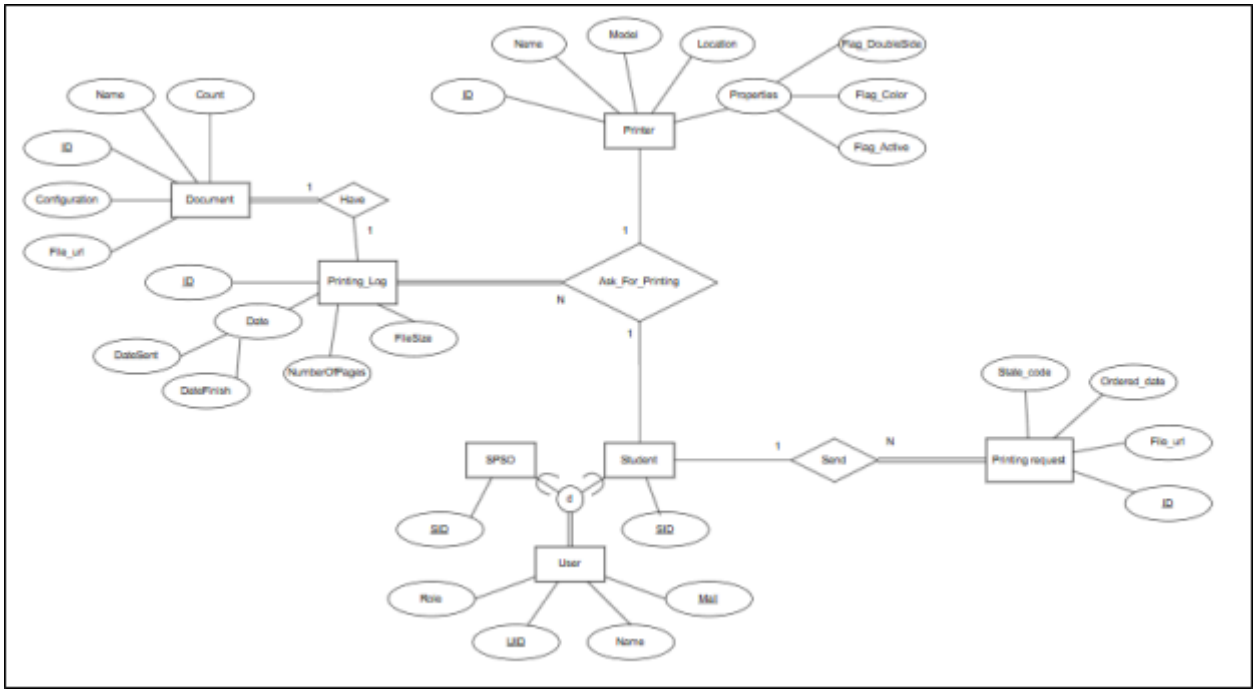

Figure 4.20. SEQ Figure * ARABIC 1: ERD diagram

User: the schema contains the following attributes

Attribute	**Data type**	**Description**
UID	Integer	Primary key, Not NULL
Name	String	Not NULL
Mail	String	Primary key, Not NULL
Role	Boolean	0 is student, 1 is SPSO

Table 4.1. User schema

Printing log: the schema contains the following attributes

Attribute	**Data type**	**Description**

ID	Integer	Primary key, Not NULL
DateSend	Date	Not NULL
DateFinish	Date	Not NULL
NumberOfPages	Integer	Not NULL
FileSize	String	Not NULL

Table 4.2. Printing log schema

Printer: the schema contains the following attributes

Attribute	**Data type**	**Description**
ID	Integer	Primary key, Not NULL
Name	String	Not NULL
Model	String	Not NULL, configuration of the printer
Location	String	Not NULL, location of the printer
Flag_DoubleSide	Boolean	0: it cannot print double side, 1 is else
Flag_Color	Boolean	0: it cannot print color, 1 is else
Flag_Active	Boolean	0: it is unable, 1 is else

Table 4.3. Printer schema

Printing request: the schema contains the following attributes

Attribute	**Data type**	**Description**

ID	Integer	Primary key, Not NULL
File_url	String	Not NULL
Order_date	Date	Not NULL
State_code	String	Not NULL

Table 4.4. Printing request schema

Document: the schema contains the following attributes

Attribute	**Data type**	**Description**
ID	Integer	Primary key, Not NULL
Name	String	Not NULL
Count	Integer	Not NULL
Configuration	String	Not NULL
File_url	String	Not NULL

Table 4.5. Document schema

The data types supported by JSON include objects, arrays, strings, numbers, booleans, and null.

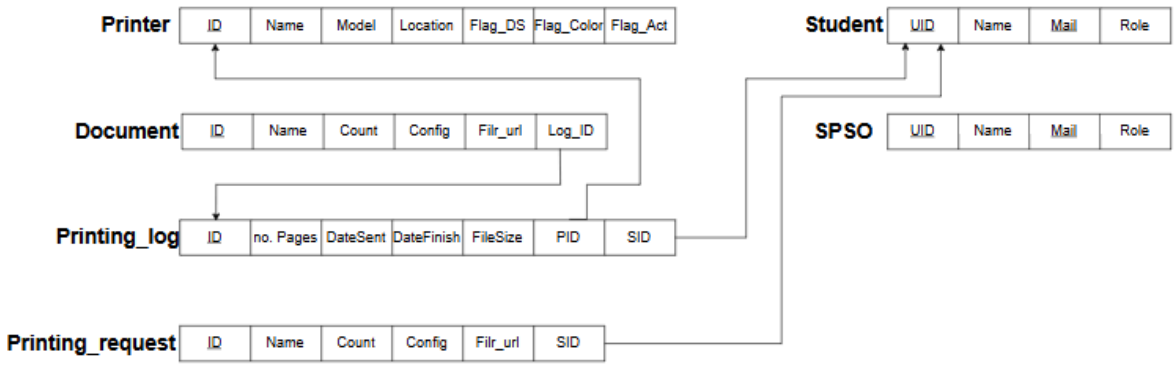

Figure 4.21. ERD mapping

User's JSON Structure:

```
{
    "first_name": "",
    "last_name": "",
    "email": "",
    "id": "",
    "phone": "",
    "city": "",
    "description": "",
    "isAdmin": boolean,
}
```

Printing Request's JSON Structure:

```
{
    "id": "",
    "state_code": "req_state_uploaded", "req_state_new",
"req_state_processing", "req_state_proceeded",
    "order_date": "2017-02-01 11:22:33",
    "email": "customer1@hcmut.edu.vn",
    "items": [
        {
            "id": "",
```

```
            "name": "book_hardcover_21x21",
            "count": "1",
            "configurations": [
                {
                    "option": "",
                    "value": ""
                },
                {
                    "option": "",
                    "value": ""
                },
            ],
            "files": [
                {
                    "file_id": "",
                    "url": ""
                },
                {
                    "field_id": "",
                    "url": ""
                }
            ]
        }
    ]
}
```

Document's JSON Structure:

```
{
    "name": "",
    "id": "",
    "count": "1",
    "configurations": [
        {
            "option": "",
            "value": ""
        },
        {
            "option": "",
            "value": ""
        }
    ],
```

Printing Log:

```
{
    "student_id": "",
    "printer_id": "",
    "files": [
        {
            "file_id": "",
            "url": ""
        },
        {
            "field_id": "",
            "url": ""
        }
    ],
    "started_at": "2017-02-01 11:22:33",
    "ended_at": "2017-02-01 11:22:33",
    "configurations": [
        {
            "option": "",
            "value": ""
        },
        {
            "option": "",
            "value": ""
        },
    ],
    "number_of_a3": 3,
    "number_of_a4": 5
}
```

Printer:

```
{
    "printer_id": "",
    "manufacturer": "",
    "model": "",
    "description": "",
    "status": ["DISABLED", "ENABLED"],
    "campus": "",
    "building": "",
    "room": ""
}
```

4.1.4. API Management

4.1.4.1. User management

- Register user:


```
{
  "User Management": {
    "Register User": {
      "method": "POST",
      "endpoint": "/api/users/register",
      "description": "Registers a new user with the system.",
      "request": {
        "headers": {
          "Content-Type": "application/json"
        },
        "body": {
          "first_name": "string",
          "last_name": "string",
          "email": "string",
          "password": "string",
          "phone": "string",
          "city": "string",
          "isAdmin": "boolean"
        }
      },
      "response": {
        "201": {
          "description": "User successfully registered.",
          "body": {
            "user_id": "string",
            "message": "User registration successful."
          }
        },
        "400": {
          "description": "Invalid input or missing fields."
        }
      }
    },
```


- Login user:

```
"Login User": {
  "method": "POST",
  "endpoint": "/api/users/login",
  "description": "Logs in an existing user.",
  "request": {
    "headers": {
      "Content-Type": "application/json"
    },
    "body": {
      "email": "string",
      "password": "string"
    }
  },
  "response": {
    "200": {
      "description": "User successfully logged in.",
      "body": {
        "user_id": "string",
        "token": "string",
        "message": "Login successful."
      }
    },
    "401": {
      "description": "Unauthorized access - invalid credentials."
    }
  }
},
```


- Get user profile:

```
"Get User Profile": {
  "method": "GET",
  "endpoint": "/api/users/{user_id}",
  "description": "Retrieves the profile of a specific user by user_id.",
  "request": {
    "headers": {
      "Content-Type": "application/json",
      "Authorization": "Bearer <token>"
    }
  },
  "response": {
    "200": {
      "description": "User profile retrieved successfully.",
      "body": {
        "user_id": "string",
        "first_name": "string",
        "last_name": "string",
        "email": "string",
        "phone": "string",
        "city": "string",
        "isAdmin": "boolean"
      }
    },
    "404": {
      "description": "User not found."
    }
  }
},
```

- Update user profile:

```
"Update User Profile": {
  "method": "PUT",
  "endpoint": "/api/users/{user_id}",
  "description": "Updates the profile information of a specific user.",
  "request": {
    "headers": {
      "Content-Type": "application/json",
      "Authorization": "Bearer <token>"
    },
    "body": {
      "first_name": "string",
      "last_name": "string",
      "phone": "string",
      "city": "string",
      "description": "string (optional)"
    }
  },
  "response": {
    "200": {
      "description": "User profile updated successfully.",
      "body": {
        "user_id": "string",
        "message": "Profile update successful."
      }
    },
    "400": {
      "description": "Invalid input or missing fields."
    },
    "404": {
      "description": "User not found."
    }
  }
},
```

- Delete user:

```
    "Delete User": {
      "method": "DELETE",
      "endpoint": "/api/users/{user_id}",
      "description": "Deletes a specific user by user_id (Admin only).",
      "request": {
        "headers": {
          "Content-Type": "application/json",
          "Authorization": "Bearer <token>"
        }
      },
      "response": {
        "200": {
          "description": "User successfully deleted.",
          "body": {
            "user_id": "string",
            "message": "User deletion successful."
          }
        },
        "403": {
          "description": "Forbidden – only admins can delete users."
        },
        "404": {
          "description": "User not found."
        }
      }
    }
  }
}
```


4.1.4.2. Printer

- Add new printer:

```
{
    "Printer Management API": {

      "Add New Printer": {
        "method": "POST",
        "endpoint": "/api/printers",
        "description": "Adds a new printer to the system.",
        "request": {
          "headers": {
            "Content-Type": "application/json",
            "Authorization": "Bearer <token>"
          },
          "body": {
            "printer_id": "string",
            "manufacturer": "string",
            "model": "string",
            "status": "string (e.g., 'ENABLED', 'DISABLED')",
            "campus": "string",
            "building": "string",
            "room": "string"
          }
        },
        "response": {
          "201": {
            "description": "Printer added successfully.",
            "body": {
              "printer_id": "string",
              "message": "Printer successfully added."
            }
          },
          "400": {
            "description": "Invalid input or missing fields."
          }
        }
      },
```

- Get all printers:

```
"Get All Printers": {
  "method": "GET",
  "endpoint": "/api/printers",
  "description": "Retrieves a list of all printers in the system.",
  "request": {
    "headers": {
      "Content-Type": "application/json",
      "Authorization": "Bearer <token>"
    }
  },
  "response": {
    "200": {
      "description": "Successfully retrieved list of printers.",
      "body": [
        {
          "printer_id": "string",
          "manufacturer": "string",
          "model": "string",
          "status": "string",
          "campus": "string",
          "building": "string",
          "room": "string"
        }
      ]
    },
    "404": {
      "description": "No printers found in the system."
    }
  }
},
```

- Get printer details:

```
"Get Printer Details": {
  "method": "GET",
  "endpoint": "/api/printers/{printer_id}",
  "description": "Retrieves details of a specific printer.",
  "request": {
    "headers": {
      "Content-Type": "application/json",
      "Authorization": "Bearer <token>"
    }
  },
  "response": {
    "200": {
      "description": "Successfully retrieved printer details.",
      "body": {
        "printer_id": "string",
        "manufacturer": "string",
        "model": "string",
        "status": "string",
        "campus": "string",
        "building": "string",
        "room": "string"
      }
    },
    "404": {
      "description": "Printer not found."
    }
  }
},
```

- Enable or disable printer:

```
"Enable or Disable Printer": {
  "method": "PATCH",
  "endpoint": "/api/printers/{printer_id}/status",
  "description": "Toggles the status of a specific printer between 'ENABLED' and 'DISABLED'.",
  "request": {
    "headers": {
      "Content-Type": "application/json",
      "Authorization": "Bearer <token>"
    },
    "body": {
      "status": "string (e.g., 'ENABLED', 'DISABLED')"
    }
  },
  "response": {
    "200": {
      "description": "Printer status updated successfully.",
      "body": {
        "printer_id": "string",
        "status": "string",
        "message": "Printer status updated."
      }
    },
    "400": {
      "description": "Invalid status code."
    },
    "404": {
      "description": "Printer not found."
    }
  }
},
```


- Delete printer:

```
    "Delete Printer": {
      "method": "DELETE",
      "endpoint": "/api/printers/{printer_id}",
      "description": "Deletes a specific printer from the system.",
      "request": {
        "headers": {
          "Content-Type": "application/json",
          "Authorization": "Bearer <token>"
        }
      },
      "response": {
        "200": {
          "description": "Printer deleted successfully.",
          "body": {
            "printer_id": "string",
            "message": "Printer successfully deleted."
          }
        },
        "404": {
          "description": "Printer not found."
        }
      }
    }
  }
```


4.1.4.3. Printer requests

- Submit print request:

```
{
  "Print Requests": {
    "Submit Print Request": {
      "method": "POST",
      "endpoint": "/api/printing/request",
      "description": "Allows a user to submit a new printing request.",
      "request": {
        "headers": {
          "Content-Type": "application/json",
          "Authorization": "Bearer <token>"
        },
        "body": {
          "user_id": "string",
          "items": [
            {
              "id": "string",
              "name": "string (document name)",
              "configurations": [
                {
                  "option": "string",
                  "value": "string"
                }
              ],
              "files": [
                {
                  "file_id": "string",
                  "url": "string"
                }
              ]
            }
          ],
          "order_date": "string (ISO date format)"
        }
      },
```

```
      "response": {
        "201": {
          "description": "Print request submitted successfully.",
          "body": {
            "request_id": "string",
            "message": "Print request submitted."
          }
        },
        "400": {
          "description": "Invalid input or missing fields."
        }
      }
    },
```


- Get all print requests for user:

```
"Get All Print Requests for User": {
  "method": "GET",
  "endpoint": "/api/printing/requests/{user_id}",
  "description": "Retrieves a list of all printing requests for a specific user.",
  "request": {
    "headers": {
      "Content-Type": "application/json",
      "Authorization": "Bearer <token>"
    }
  },
  "response": {
    "200": {
      "description": "Successfully retrieved printing requests.",
      "body": [
        {
          "request_id": "string",
          "order_date": "string (ISO date format)",
          "state_code": "string (e.g., 'req_state_uploaded')",
          "items": [
            {
              "id": "string",
              "name": "string",
              "count": "integer",
              "configurations": [
                {
                  "option": "string",
                  "value": "string"
                }
              ],
```

```
              "files": [
                {
                  "file_id": "string",
                  "url": "string"
                }
              ]
            }
          ]
        }
      ]
    },
    "404": {
      "description": "User has no print requests or user not found."
    }
  }
},
```


- Get print request information:

```
"Get Print Request Information": {
  "method": "GET",
  "endpoint": "/api/printing/request/{request_id}",
  "description": "Retrieves details for a specific printing request.",
  "request": {
    "headers": {
      "Content-Type": "application/json",
      "Authorization": "Bearer <token>"
    }
  },
  "response": {
    "200": {
      "description": "Successfully retrieved print request details.",
      "body": {
        "request_id": "string",
        "order_date": "string (ISO date format)",
        "state_code": "string",
        "user_id": "string",
        "items": [
          {
            "id": "string",
            "name": "string",
            "count": "integer",
            "configurations": [
              {
                "option": "string",
                "value": "string"
              }
            ],
            "files": [
              {
                "file_id": "string",
                "url": "string"
              }
            ]
          }
        ]
      }
    },
    "404": {
      "description": "Print request not found."
    }
  }
},
```

- Cancel print request:

```
"Cancel Print Request": {
  "method": "DELETE",
  "endpoint": "/api/printing/request/{request_id}",
  "description": "Cancels an existing printing request.",
  "request": {
    "headers": {
      "Content-Type": "application/json",
      "Authorization": "Bearer <token>"
    }
  },
  "response": {
    "200": {
      "description": "Print request canceled successfully.",
      "body": {
        "request_id": "string",
        "message": "Print request has been canceled."
      }
    },
    "404": {
      "description": "Print request not found."
    },
    "403": {
      "description": "User does not have permission to cancel this request."
    }
  }
},
```

- Update print request state:

```
    "Update Print Request State": {
      "method": "PATCH",
      "endpoint": "/api/printing/request/{request_id}/state",
      "description": "Updates the state of a printing request (e.g., 'processing', 'completed').",
      "request": {
        "headers": {
          "Content-Type": "application/json",
          "Authorization": "Bearer <token>"
        },
        "body": {
          "state_code": "string (e.g., 'req_state_processing', 'req_state_completed')"
        }
      },
      "response": {
        "200": {
          "description": "Print request state updated successfully.",
          "body": {
            "request_id": "string",
            "state_code": "string",
            "message": "Print request state updated."
          }
        },
        "400": {
          "description": "Invalid state code."
        },
        "404": {
          "description": "Print request not found."
        }
      }
    }
  }
}
```


4.1.4.4. Logging

- Add printing log entry:

```
"Logging API": {
  "Add Printing Log Entry": {
    "endpoint": "/api/logs/printing",
    "description": "Adds a new log entry for a printing transaction. This is typically used by the system to record each print action.",
    "request": {
      "headers": {
        "Content-Type": "application/json",
        "Authorization": "Bearer <token>"
      },
      "body": {
        "student_id": "string",
        "printer_id": "string",
        "start_time": "string (ISO date format)",
        "end_time": "string (ISO date format)",
        "number_of_a3": "integer",
        "number_of_a4": "integer",
        "configurations": [
          {
            "option": "string",
            "value": "string"
          }
        ]
      }
    },
    "response": {
      "201": {
        "description": "Printing log entry added successfully.",
        "body": {
          "log_id": "string",
          "message": "Printing log entry created."
        }
      },
      "400": {
        "description": "Invalid input or missing fields."
      }
    }
  },
```


- Get user printing logs:

```
"Get User Printing Logs": {
  "method": "GET",
  "endpoint": "/api/logs/printings/{user_id}",
  "description": "Retrieves a list of printing log entries for a specific user within a specified date range.",
  "request": {
    "headers": {
      "Content-Type": "application/json",
      "Authorization": "Bearer <token>"
    },
    "query_parameters": {
      "from": "string (start date in ISO format)",
      "to": "string (end date in ISO format)"
    }
  },
  "response": {
    "200": {
      "description": "Successfully retrieved printing logs.",
      "body": [
        {
          "log_id": "string",
          "printer_id": "string",
          "start_time": "string",
          "end_time": "string",
          "number_of_a3": "integer",
          "number_of_a4": "integer",
          "configurations": [
            {
              "option": "string",
              "value": "string"
            }
          ]
        }
      ]
    },
    "404": {
      "description": "No logs found for the specified user or date range."
    }
  }
```


- Get printer activity logs:

```
"Get Printer Activity Logs": {
  "method": "GET",
  "endpoint": "/api/logs/printers/{printer_id}",
  "description": "Retrieves all activity logs for a specific printer within a date range.",
  "request": {
    "headers": {
      "Content-Type": "application/json",
      "Authorization": "Bearer <token>"
    },
    "query_parameters": {
      "from": "string (start date in ISO format)",
      "to": "string (end date in ISO format)"
    }
  },
  "response": {
    "200": {
      "description": "Successfully retrieved printer activity logs.",
      "body": [
        {
          "log_id": "string",
          "student_id": "string",
          "start_time": "string",
          "end_time": "string",
          "number_of_a3": "integer",
          "number_of_a4": "integer",
          "configurations": [
            {
              "option": "string",
              "value": "string"
            }
          ]
        }
      ]
    },
    "404": {
      "description": "No logs found for the specified printer or date range."
    }
  }
```

- Get system activity logs:

```
"Get System Activity Logs": {
  "method": "GET",
  "endpoint": "/api/logs/system",
  "description": "Retrieves system-wide activity logs, filtered by date range.",
  "request": {
    "headers": {
      "Content-Type": "application/json",
      "Authorization": "Bearer <token>"
    },
    "query_parameters": {
      "from": "string (start date in ISO format)",
      "to": "string (end date in ISO format)"
    }
  },
  "response": {
    "200": {
      "description": "Successfully retrieved system activity logs.",
      "body": [
        {
          "log_id": "string",
          "activity": "string (e.g., 'print', 'update')",
          "actor": "string (e.g., 'student', 'admin')",
          "actor_id": "string",
          "timestamp": "string",
          "details": {
            "description": "string",
            "additional_info": "string"
          }
        }
      ]
    },
    "404": {
      "description": "No logs found for the specified date range."
    }
  }
```

4.2. Component diagram

The component diagram (in figure 4.22 below) depicts basic functionalities of the printing system in relationship with backend database systems for usage storage, including student storage separated from (SPSO) manager storage. In general, there are three main components namely User, Printing system, and Database system. First, it is important that all users must be authenticated by the HCMUT_SSO system, which is verified by the backend database system called UserDbs. UserDbs may only store user ids for

identification. Log-ins may also be recorded for audit and logging. Then, printing system service may begin. To support printing, the system allows various functionalities provided by many subsystems as seen below, including Upload system, (student) View system, (student) printer server system, and Delete system. Each system functions as its name suggests, and it may provide more specific operations as needed. For example, the Upload system is for uploading files specifically which may be expanded to support file encryption for security... Upload system will then send files to the student database through a database system called StudentDbs. Similarly, the Delete system will delete files from StudentDbs. Printer server system allows students to print files through the printer system. All operations above except for student viewing will be recorded to StudentDbs. To support the managing role, system allows (SPSO) manager access to View system, printer system, printer server system, report system, and configuration system. Printer system and printer server system will be configured by managers to add, enable, or disable printers as needed. Report system is for usage report generation. Configuration system (ConfigSystem) is to control system usage, which in this case is student usage. All manager actions except viewing will be recorded to a manager database system called ManagerDbs.

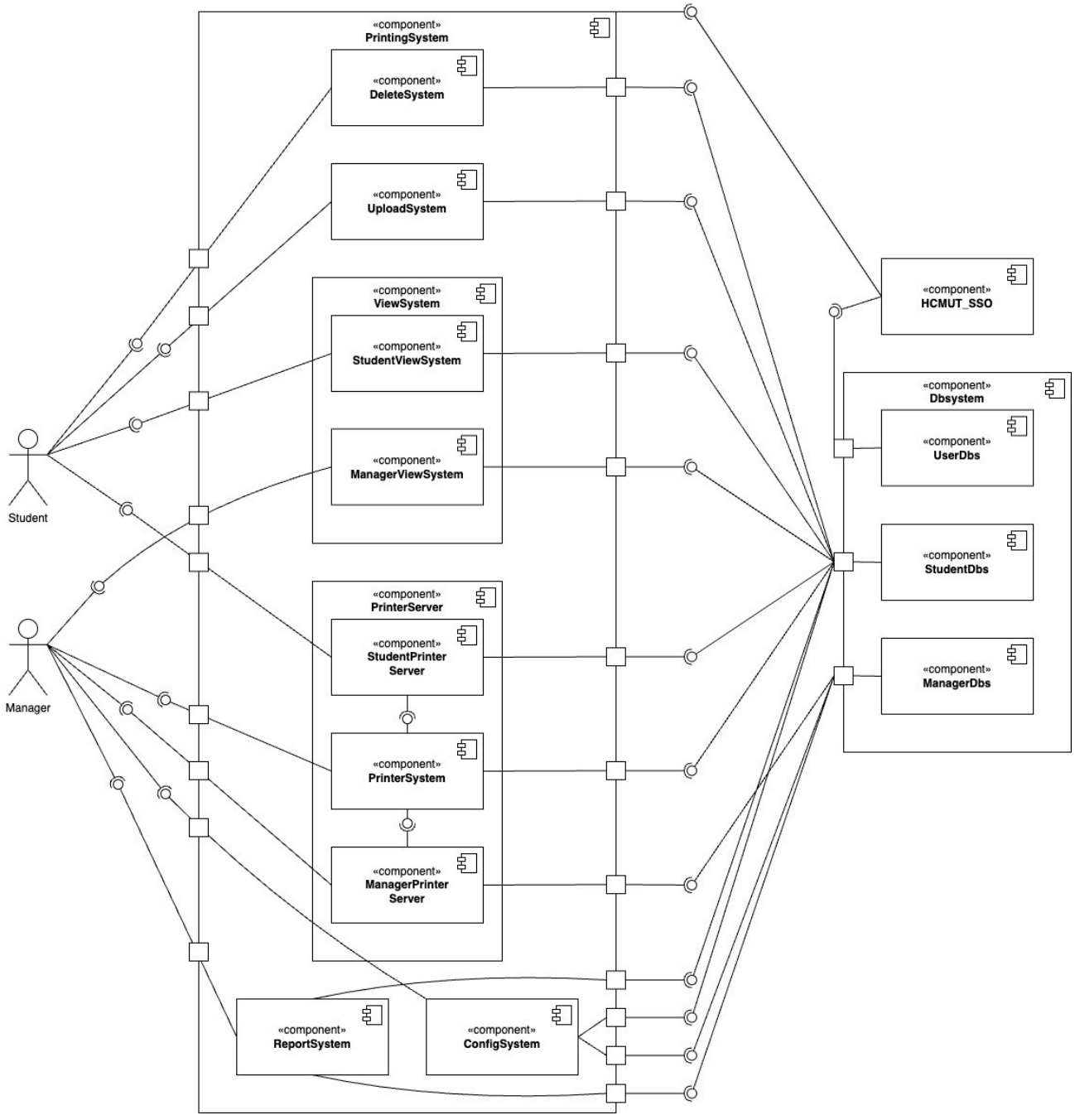

Figure 4.22. Component diagram

5. IMPLEMENTATION

5.1. GitHub version control

We use GitHub to manage the SPSS project. The documents, materials, and folders for Requirement, System modeling, and Architectural design are added to the same repository. Our GitHub repository can be accessed via Software Engineering GitHub.

5.2. Usability testing

5.2.1. Overview

We use the Figma prototype for usability testing. The testers access the Figma prototype through Software Engineering - Web demo and give feedback via the Feedback form. In this part, we will present the process of usability testing: recruiting participants, defining tasks for participants, giving the test strategy, and showing the results of usability testing.

5.2.2. Participants

The first step in conducting testing and gathering feedback is recruiting participants, focusing on university students as the core user base. The testing phase lasts for 5 days, providing time for participants to interact with the system and give their feedback. The feedback form and demo will be shared with the team members' friends for testing.

5.2.3. Participants' tasks

Participants must give their scores on the following screens:

- Homepage
- Sign in/Sign up
- [Student] dashboard
- [Student] Print Services

After that, they must give the overall score and their feedback.

5.2.4. Test strategy

The test strategy outlines both qualitative and quantitative methods used in usability testing. The qualitative approach emphasizes collecting insights and user anecdotes to identify issues in the user experience. In contrast, the quantitative approach focuses on evaluating the UI through scoring, tracking task completion times, and analyzing navigation efficiency.

During qualitative testing, we will ask participants:

- about their initial impressions of the interface,

- how to enhance their overall user experience,

- whether they would recommend our product to others, offering valuable insights into the system's usability and its potential for positive recommendations.

With the quantitative testing, users will be asked to rate the user interface on a scale from 0 to 10, where 0 indicates “need major improvements” and 10 indicates “good”. This evaluation will cover an overall score for the entire UI. Additionally, the results will be displayed visually using bar charts for easy and effective presentation.

5.2.5. Feedback

There are 37 participants in this usability testing. All of them are university students and 86.5% are third-year students.

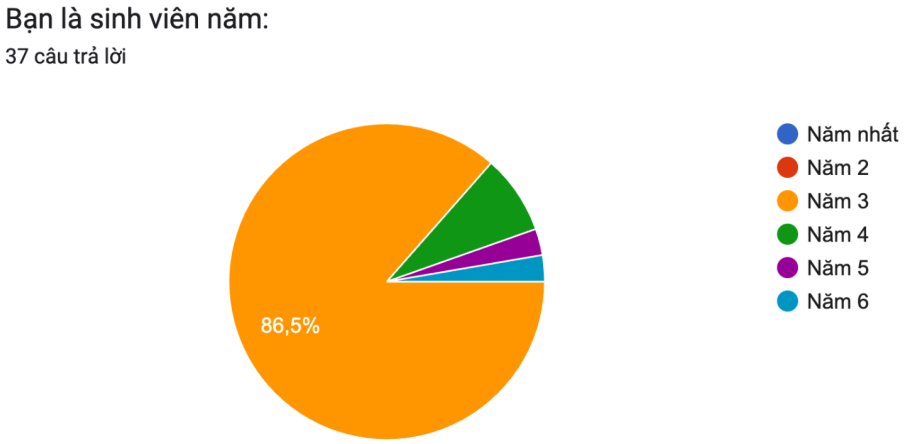

Figure 5.1. Participants’ demographics

The participants come from different universities in Ho Chi Minh City.

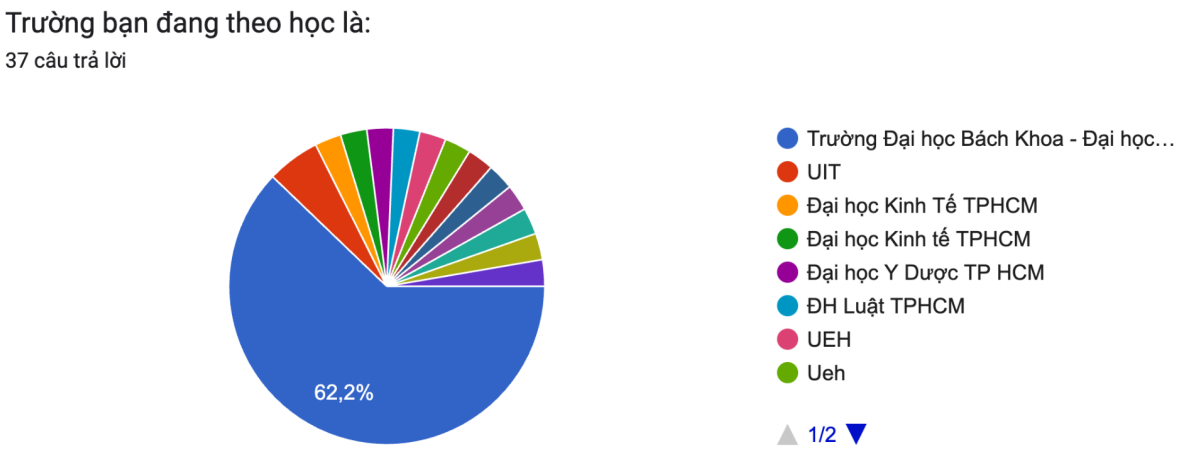

Figure 5.2.1. Participants' workplaces

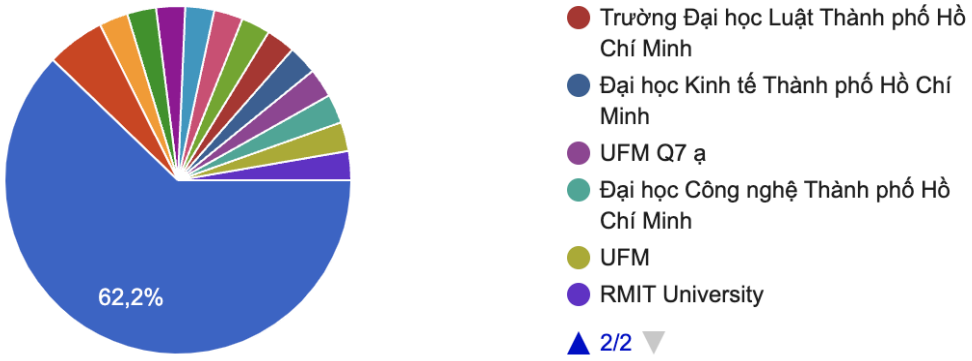

Figure 5.2.2. Participants' workplaces

5.2.5.1. UI Evaluation

Overall, most participants are satisfied with the UI.

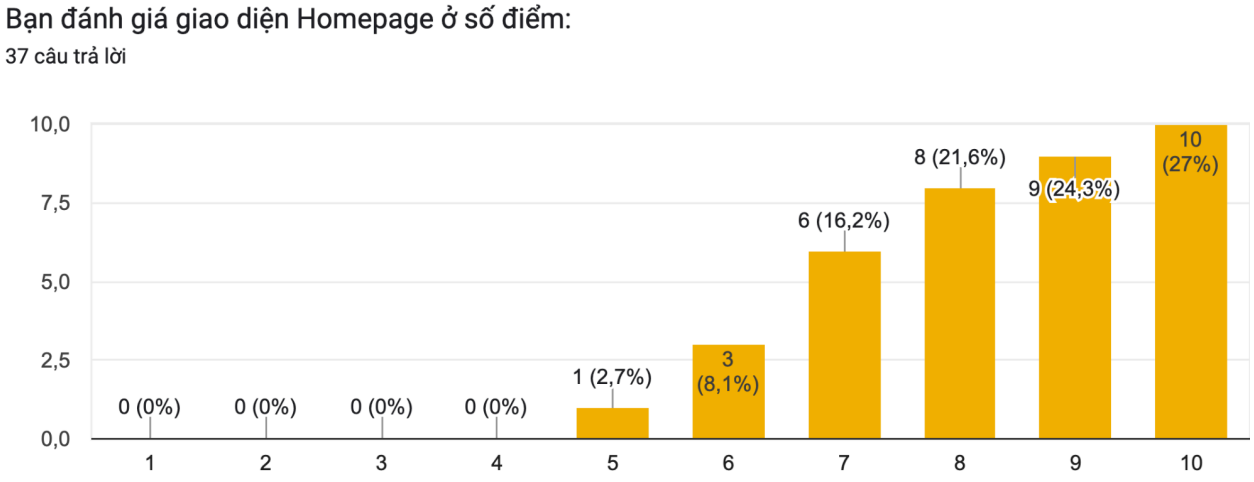

Figure 5.3. Homepage evaluation

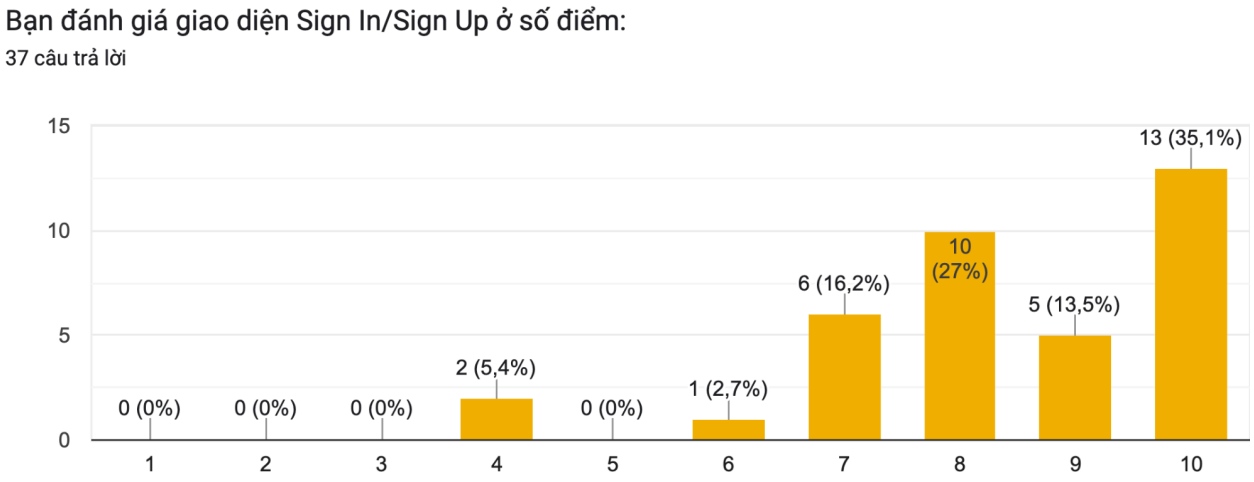

Figure 5.4. Sign in/Sign up evaluation

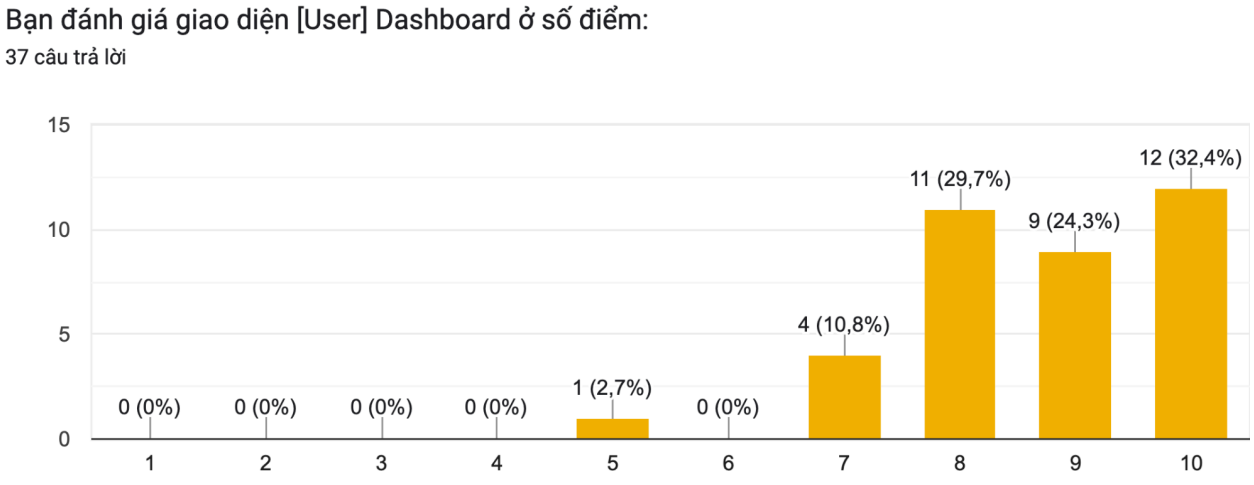

Figure 5.5. [Student] Dashboard evaluation

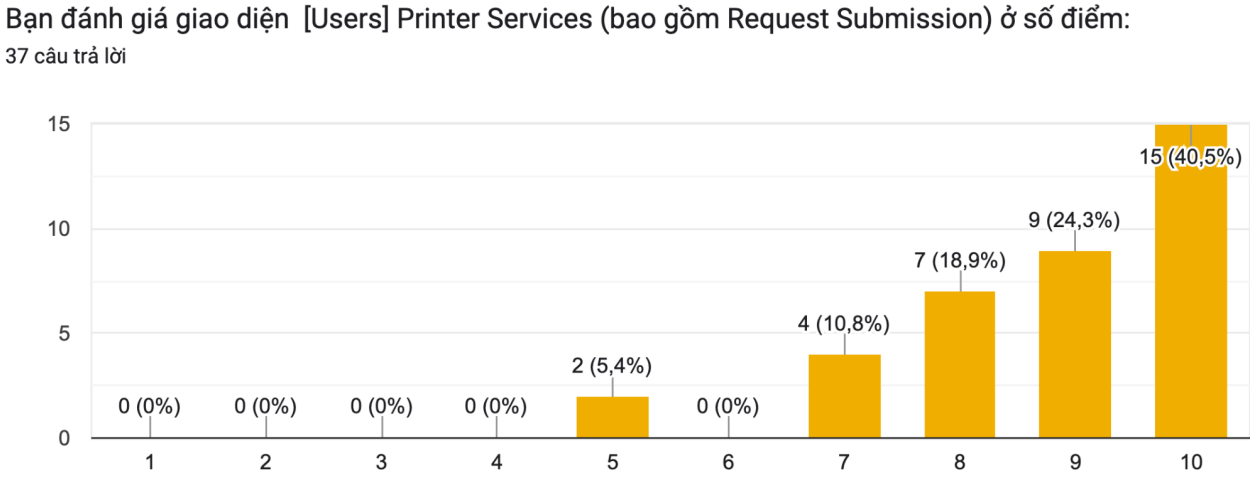

Figure 5.6. [Student] Dashboard evaluation

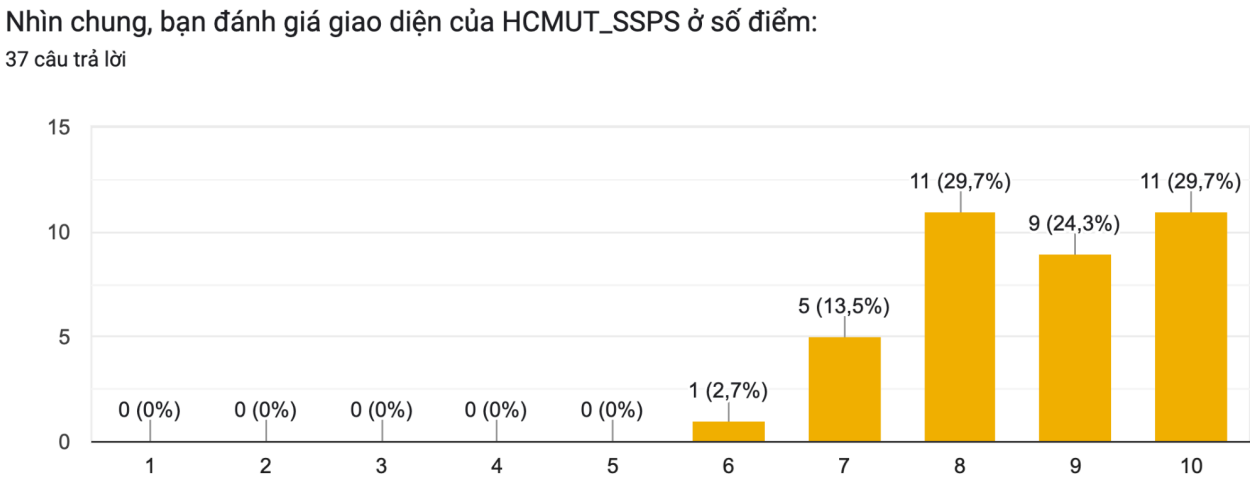

Figure 5.7. [Student] Overall evaluation

5.2.5.1. Feedback

The participants gave many valuable feedback, which we can take into consideration to improve our website as well as user experience.

Giao diện nhìn sang xịn mịn quá shop, nhưng mà nếu được thì cho màu xanh để đồng bộ nhà trường
Đoạn Sign in hơi nhiều màu trắng, mình thấy hơi khó nếu nhìn vô để nhập tên đăng nhập, nên làm viền tên và nhập mk đặm hơn, hoặc làm full khung 1 màu.
Màu hơi chói
Homepage thì 2 cái khung [back to main page] và [print document] nên dùng font to hơn (Bold) cho chữ. Phần Q&A nên thay màu chữ nội dung thành màu đen, dùng font nhỏ hơn và thanh hơn (giảm độ dày/bold của chữ). Chữ vàng trên nền trắng sẽ hơi nhói đó, nên là chỉ nên dùng với những cụm nhỏ nhỏ và dễ đọc thôi nhé.

Phần Status ở user dashboard có màu dễ nhầm lẫn, nhất là status waiting và done

Layout, chữ của giao diện khá lớn, đọc hơi mỏi mắt , nên vừa trong một khung hình.

Nhìn chung giao diện của mn làm quá ổn rồi, bravo:::
1. mình chỉ có một chút góp ý là notification, bên cạnh việc để trong giao diện chính để user click vào, thì team có thể add thêm một icon ở góc màn hình trên bên phải, tương tự như giao diện các trang mxh. Điều này sẽ dễ giúp người dùng đang ở một page như print khi đã order xong thì ngay lập tức noti ở góc hiện ra, indicate là order nó hoàn tất hay chưa, hay cả update về cái gì đó.
2. Bên cạnh đó, team có thể thêm một mục như là Project cho phần user để họ có thể tích hợp thêm cả tính năng storage, thông qua việc tích hợp Gdrive/onedrive option cho họ có thể trực tiếp truy cập kho của bản thân. Còn nếu muốn kiếm thêm chút tiền nữa thì có thể coi như là cho thuê storage của bản thân :)). Bản thân mình nếu in tài liệu thì thường tạo folder cho các dự án để khi cần mình có thể in nhiều bản và lặp đi lặp lại. Cái này sẽ mở rộng cho các bạn tệp người sử dụng là dân kinh doanh như mình :)) thay vì chỉ là one-off hay adhoc project.
3. Cuối cùng là sau khâu order successful thì thay vì trả thẳng về màn hình dashboard, các bạn có thể trả về lại chỗ print để user có thể trực tiếp order thêm, thay vì phải click lại vào mục print. Cá nhân mình thì nhiều lúc multi-tasking nhiều nên mình thấy sẽ có lúc user sẽ hơi frustrated nhẹ nếu phải thêm một step như thế.
4. Mở rộng cho ý thư 3, đó là team có thể expand print option từ một file at one time thành multiple at one time, như v người dùng có thể tiết kiệm tgian hơn nữa.
đó là toàn bộ những gì mình cảm thấy từ góc nhìn là một dân kinh doanh :))))). Hope u guys earn flying colors with this project.....

6. PROJECT CONCLUSION

6.1. Result

- Prototype: https://shorturl.at/4SgxQ

- Front-end and back-end implementation: https://shorturl.at/qP28f

- User experience survey: https://shorturl.at/DsoHj

- Project’s phases:

    + Task 1 (1 week): The team determined the requirements.

    + Task 2 (1 week): The team drew use case diagrams.

+ Task 3 (2 week): The team conducted system modeling and designed the architecture.

+ Task 4 (2 week): The team implemented front-end and back-end of the HCMUT_SPSS website.

+ Task 5 (1 week): The team reviewed the project as well as presented the project to the Lecturer and classmates.

6.2. Lessons

We gained a thorough education from the Smart Printing Service System project. Our comprehension of front-end and back-end interaction taught us how to connect user interfaces and server-side logic, which is one of the most important areas of development. The user experience involved managing data flow, responding to API requests, and guaranteeing real-time updates. As we came across and fixed numerous problems, we greatly improved our debugging abilities. In terms of design, we learned how to create the wireframe, mockup, and prototype by using Figma.

Beside technical advancement, the project promoted our critical communication and cooperation skills as we worked well together to assign assignments, handle conflicts, and guarantee the project objectives. Furthermore, we learned how to handle Feedback from peers and mentors, which improves our capacity to take constructive criticism.

In general, all of these experiences together have given us a growth-oriented mindset that emphasizes ongoing learning and development, as well as equipped us for future software development problems.