VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Software Engineering Assignment

Student Smart Printing Service (HCMUT-SSPS)

Semester: 231 - Group: GROUP 12

Advisors:	Truong Thi Thai Minh	
	Bui Cong Tuan	
Student:	Nguyen Manh Khang	2152639
	Nguyen Phan Tri Duc	2152528
	Truong Quoc Hung	2153414
	Phan Quoc An	2153148
	Luu Quang Hung	2153408
	Vo Van Luan	2152744
	To Chau Hao Nhan	2152829

HO CHI MINH CITY, 2023

Contents

I Problem specification **4**

II Task solution **6**

1 Task 1: Requirement elicitation **7**
1.1 Describe the domain context of a smart printing service for students at HCMUT. Who are relevant stakeholders? What are their current needs? In your opinion, what benefits HCMUT-SSPS will be for each stakeholder? . . . 7
1.1.1 Describe the domain context of a smart printing service for students at HCMUT . 7
1.1.2 Who are relevant stakeholders? . . . 8
1.1.3 What are their current needs? . . . 9
1.1.4 In your opinion, what benefits HCMUT-SSPS will be for each stakeholder? . . . 10
1.2 Describe all functional and non-functional requirements that can be inferred from the project description. . . . 10
1.3 Draw a use-case diagram for the whole system. Choose at least one important module and draw its use-case diagram, as well as describe the use-cases using a table format . . . 15
1.3.1 Draw a general use-case diagram for the whole system . . . 15
1.3.2 Draw Smart printing service use-case diagram . . . 17
1.3.3 Describe the use-case using a table format . . . 17

2 Task 2: System modeling **32**
2.1 Draw an activity diagram to capture the business process between systems and the stakeholders for the important module(s) chosen in Task 1.3. . . . 32
2.1.1 Print Service Module Diagram: . . . 32
2.1.2 Description: . . . 32
2.1.3 Print Service Management Module Diagram: . . . 35
2.1.4 Description: . . . 35
2.2 Draw sequence diagrams for the important module(s) chosen in Task 1.3. . . . 36
2.2.1 Diagram . . . 36
2.3 Draw a class diagram of the important module(s) chosen in Task 1.3 as comprehensive as possible . . . 44
2.3.1 Diagram . . . 44
2.4 Develop MVP 1 as user interfaces of either a Desktop-view central dashboard for a particular module (the same with the module used in task 2.1). Decide yourself what to include in the view. Use a wireframe tool like Figma or Adobe XD, or Illustrator . . . 45
2.4.1 Login screen . . . 46
2.4.2 Key screens of student's flow . . . 46
2.4.3 Key screens of SPSO flow . . . 49
2.4.4 Key screens of Financial Office flow . . . 50
2.4.5 Key screens of Technical Support Office flow . . . 51

3 Task 3: Architecture design **53**

3.1 Use a layered architecture to design the HCMUT-SSPS system. Describe how will you present your User Interface. Describe how will you store your data. Describe how you will access to external services/ APIs. . . . 53
3.1.1 Use a layered architecture to design the HCMUT-SSPS system . . . 53
3.1.2 Describe how you will present your User Interface. . . . 58
3.1.3 Describe how you will store your data . . . 60
3.1.4 Describe how you will access to external services/ APIs . . . 66
3.2 Draw a component diagram for the important module(s) chosen in Task 1.3. . . . 70
3.2.1 Diagram . . . 70
3.2.2 Description . . . 71

4 Task 4: Implementation – Sprint 1 **79**
4.1 Setting up an online repository (github, bitbucket, etc) for version control. . . . 79
4.2 Adding documents, materials and folders for Requirement, System modelling and Architectural design. Use the selected version control system to report the changes to these files. . . . 80
4.3 Conduct a usability test with the user interface you developed in MVP1. . . . 83

5 Task 5: Implementation – Sprint 2 **86**
5.1 Develop MVP 2 with input from Task 2.4 and Task 4.3. You are free to choose the programming language (HTML, Javascript, Python, C#, etc). It is not required to implement a database in the backend. Data can be hard coded in code files. . . . 86
5.1.1 Tech stack choice . . . 86
5.1.2 Demonstration product . . . 87
5.2 Demonstrate the whole project from Task 1 to Task 5 . . . 87

6 Final Thoughts **88**
6.0.1 Risks and Challenges . . . 88
6.0.2 Lessons learned . . . 88

Members list & Workload

No.	Full name	Student ID	Duty in the assignment	Contribution
1	Nguyen Manh Khang	2152639	Participate, review and perfect processes and tasks. Make sure the work is consistent and of good quality. Assumed product owner. Communicate with TA for quality assurance.	100%
2	Nguyen Phan Tri Duc	2152528	Assumed Scrum master. Participate in requirement engineering. Participate in UML system modeling: activity, and class. Architectural design.	100%
3	Truong Quoc Hung	2153414	Assumed Tech lead. Participate in requirement engineering. In charge of realizing MVP#1 and MVP#2. Technical decision maker.	100%
4	Phan Quoc An	2153148	Participate in UML system modeling: sequence. Participate in requirement engineering. Architectural design	100%
5	Luu Quang Hung	2153408	Participate in UML system modeling: sequence. Participate in requirement engineering. Architectural design	100%
6	To Chau Hao Nhan	2152829	Participate in requirement engineering. Developer.	100%
7	Vo Van Luan	2152744	Participate in requirement engineering. Developer.	100%

Part I

Problem specification

The university intends to build a Student Smart Printing Service (HCMUT_SSPS) for serving students in its campuses to print their documents.

The system consists of some printers around the campuses. Each printer has an ID, brand/manufacturer name, printer model, short description, and the location (campus name, building name, and room number).

The system allows a student to print a document by uploading a document file onto the system, choosing a printer, and specifying the printing properties such as paper size, pages (of the file) to be printed, one-/double-sided, number of copies, etc. The permitted file types are limitted and configured by the Student Printing Service Officer (SPSO).

The system has to log 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.

The system allows the SPSO to view the printing history (log) of all students or a student for a time period (date to date) and for all or some printers. Of course, a student can also view his/her printing log for a time period together with a summary of the number of printed pages for each page size.

For each semester, the university gives each student a default number of A4-size pages for printing. 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 allows a student to print some number of pages when it does not exeed his/her account (page) balance. Note that one A3 page is equivalent to two A4 pages.

The SPSO has a feature to manage printers such as add/enable/disable a printer.

The SPSO also has a feature 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.

The reports of the use of the printing system are generated automatically at the end of each month and each year and are stored in the system, and can be viewed by the SPSO anytime.

All users have to be authenticated by the HCMUT_SSO authentication service before using the system.

The system is provided through a web-based app or a mobile app.

Part II

Task solution

Chapter 1

Task 1: Requirement elicitation

1.1 Describe the domain context of a smart printing service for students at HCMUT. Who are relevant stakeholders? What are their current needs? In your opinion, what benefits HCMUT-SSPS will be for each stakeholder?

1.1.1 Describe the domain context of a smart printing service for students at HCMUT

When it comes to printing documents, HCMUT students often face these issues:

- Not fully aware of the locations of copy shops when they need one the most. As a result, time and effort are wasted in finding it.
- When they need to print their files, they have to upload and access them via Facebook or Zalo at the copy shop's own desktop machine. This causes great privacy concerns.
- It is sometimes hard to calculate a fair and accurate price when the amount of printing pages is too little, for example, just printing 1 or 2 pages.
- The payment options will be limited if they print in a copy shop. For example, they want to pay via bank transfer instead of cash, but the store only accepts cash, or the total payment is not large enough for bank transfer.
- When they need to print the same file again, they have to repeat the process of going to the copy shop, sending the file to its desktop machine, and printing.
- The student may have to wait a long time for their printed documents.
- It is very likely to corrupt the files when sending via social media messaging platforms (e.g. corrupted Word file when sent via Zalo - a media employed by a lot of copy shops in Ho Chi Minh City).
- The printing settings set by the copy shop's staff may not align with the students' preferences. For instance, students might intend to print a document as single-sided, but if they forget to communicate this to the copy shop's staff, it can result in a misunderstanding of their desired printing format, e.g. the staff may assume to print as double-sided by default.

There are other challenges the product owner face when they need to implement the printing service for the students on campus:

- The product owner cannot manage all the printers effectively; they would normally rely just on old technologies like Excel, or Microsoft Access for this. Because these tools provide limited capabilities, and even outdated interfaces, it makes some management activities difficult and cumbersome.
- They would need to hire more personnel to manage the printers manually.
- They cannot know how the students use the printers.
- They would lack a dedicated interface to manage the whole printing service.

- The service might not be able to compete with the ones provided outside, making the system underused. This would cause a waste of resources.
- The operator at each printer would have a harder time serving students if there are too many coming, lining up to print.

In an attempt to address these issues, HCMUT - SSPS is a software service that is innovative, easy to use, and meets the printing demands of HCMUT students and without being overly complex:

- The software system includes all physical printers situated around HCMUT's two main campuses, each of which has their own printer ID, a short description about the model, manufacturer, and their respective location in the campus.
- Every user has to be authenticated via HCMUT-SSO before using the service. The student's information will also be mapped into this system, together with the page balance information.
- The student can upload their files onto the system and customize the printing properties. Moreover, they can customize the printing properties in bulk, meaning to customize printing properties that apply to all selected files. The uploaded files will be saved within the system for 24 hours to enhance the student's experience but also prevent unnecessary resource usage.
- What differentiates HCMUT-SSPS is the fact that the system can recommend an appropriate printer based on the time it takes to complete the printing job. The estimation of completion time for each printer will also be displayed to the students. This helps the students choose the printer that suits their needs the best. They also know when they can pick up their documents.
- The student can print their documents from anywhere, at any time without worrying about how they are going to pick up their documents because there will be operators working in office hours at each printer, who will organize the printed documents. The students confirm their student ID with the staff to obtain their documents.
- The system also provides a dedicated interface at each printer for the operator to manage printed documents by the students. They need to start an incoming printing job before the printer can print it. This helps the operator know which documents belong to whom.
- The student can view the history of printing activities.
- The system will grant the students a default amount of A4 pages at a pre-configured date. The student should print within this page balance. The student can top up their balance any time. The system will connect to BKPay to manage transactions. The system logs all purchase activities of the students. The students complete the payment via BKPay within a certain time. If one fails to do so, their misconduct will be handled according to the school regulations.
- In the profile interface, the student can view the summary of the number of printed pages for each page size A3, and A4, as well as the summary of the total number of printing activities.
- The administrative interface allows the Smart Printing Service Officer (SPSO) to manage the printers connected to the system (which includes adding, changing the active status of the printer), configure the file types students are allowed to upload, default page balance grant, and default grant date. The system also allows the SPSO to view a particular student's printing activities.
- To help the SPSO make more informed management decisions, the system will automatically generate a system usage report at the end of each month, and at the end of a year. These reports will be stored permanently. Therefore, the system will leverage an external storage to both manage them, and the files uploaded by the students.

1.1.2 Who are relevant stakeholders?

Definition of stakeholder

A stakeholder is a person, group or company that is directly or indirectly involved in the project and who may affect or get affected by the outcome of the project.

- **Students**: are the main users of the printing service.

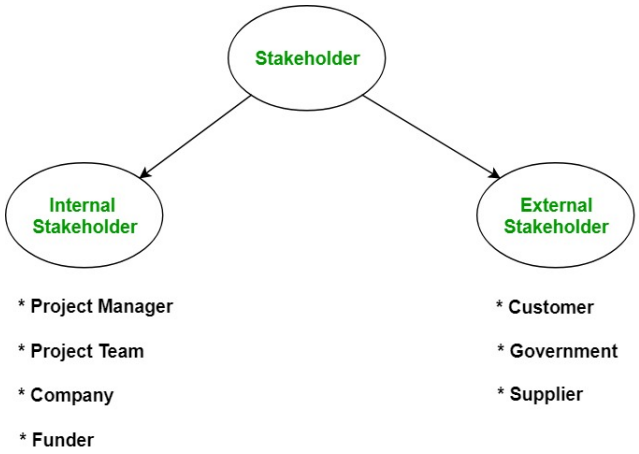


Figure 1.1: Stakeholder overview
Source: https://www.geeksforgeeks.org/software-engineering-stakeholder/

- **Student Printing Service Officers (SPSO)**: manage the service.
- **Printer operator**: employed by the school to organize the printed documents of students and confirm the student's identity when they come to pick up their documents. They also administer the ink level of printers.
- **Planning & Financial Affairs Officers (PFAO)**: manage the financial resources and activities of the printing service, such as budgeting, accounting, auditing, or reporting. They decide how much the students pay for their extra pages, and also manage BKPay - the online payment system that allows students to buy more printing pages.
- **Head of University**: provide the budget and resources for developing and maintaining the system, and communicating the system to the internal and external stakeholders. They also need to ensure the alignment and compliance of the system with the university vision and mission, or policies and standards.
- **Technical Support**: help the users of the system with any issues or questions they may have about the system, such as how to use the system, what if the system doesn't work, etc.

1.1.3 What are their current needs?

The needs may varies, based on different stakeholders. But overall:

- **Students**: They need to print their documents easily and cheaply in the university campuses.
- **Student Printing Service Officer (SPSO)**: They need a friendly system interface which allows them to manage usage and printers. They also need to observe the performance of the service.
- **Planning & Financial Affairs Officer**: They would like to know the revenue and expenses of the system. For instances, they would like to know how much money the system earns from the students who buy more printing pages, or how much money the system spends on buying and maintaining the printers, paper, ink, etc. They also need to ensure the integration and security of the system with the BKPay payment service, such as verifying and validating each payment request and response, or preventing and resolving any errors or frauds that may occur during the payment process. Therefore, they would like to access and monitor the payment records and reports of the system, such as how many payments are made by each student.
- **Head of University**: They need insights on how many students and staff use the system, or how much time and money are saved by using the system, or how well the system meets the needs and expectations of the users.
- **Technical Support**: They need to access and use the system. They should be able to test and verify the system features and functions, reproducing and resolving the user issues or errors. They also need to create and update the user manual, FAQ, or help page for the system, such as editing text, images, videos, links, etc., publishing or deleting content, etc.

1.1.4 In your opinion, what benefits HCMUT-SSPS will be for each stakeholder?

For some specific stakeholders, it is expected that they will receive different benefits based on their needs if the HCMUT-SSPS is deployed and got approved. To be more specific:

- **Students**: They gain convenience by having easy access to on-campus printing facilities, reducing the need to seek off-campus alternatives and saving time. Additionally, the availability of cost-effective printing options helps students manage their expenses while completing assignments and coursework. The service also provides flexibility, allowing students to print from various devices and locations across campus, making it easier for them to access necessary printed materials. Moreover, the streamlined processes reduce wait times and save students time, enabling them to focus on their studies more effectively.
- **Student Printing Service Officer (SPSO)**: They can ensure the service's smooth operation and responsiveness to student needs. The SPSO can provide timely user support, addressing any issues or inquiries related to the printing service, enhancing the overall user experience. Access to data analytics allows for informed decision-making, helping allocate resources effectively and make improvements where necessary. In fact, the system will help them monitor and optimize the printing service, and ensure the availability and performance of the printers.
- **Planning & Financial Affairs Officer**: The system will provide a revenue and expense tracker to monitor and manage the financial aspects of the printing service. They will be able to see how much money the system earns from the students who buy more printing pages and overdue payments of students. The system will help them budget and account for the printing service, and ensure the compliance and accuracy of the system.
- **Head of University**: The system offers a way to gauge and enhance the printing service's value and impact. They can assess how the system improves the learning and teaching experience, supports research and innovation, and contributes to sustainability and social responsibility. Additionally, they play a role in approving the system, allocating resources, promoting it, and acknowledging its achievements. The system aids them in overseeing academic and administrative matters, ensuring alignment with the university's vision, mission, policies, and regulations.

1.2 Describe all functional and non-functional requirements that can be inferred from the project description.

Requirements that are well thought through and clearly documented are essential to any successful software engineering project. There are two main different types of system requirements that should be gathered by those working on software projects. System requirements can be categorized as either functional requirements or non functional requirements.

Functional requirements and non-functional requirements

- **Functional requirements:** Specific features and functions that a system must possess in order to meet the needs and expectations of its stakeholders. They define what the system must do and are usually described in terms of inputs, processes, and outputs.
- **Non-functional requirements:** Specific attributes or qualities that a system must possess, but they are not directly related to a particular function or feature of the system.

To make it easier to control the requirements as well as proposing them later on whenever the whole system has any problems, we assign requirements with a specific **ID**. So, the requirements table will have the form **<ID> - <Problem>**

1. **Functional requirements:**
    - **For all stakeholders (system-wide functional requirements)**

ID	Requirements/Description
F-SYSTEM-0	Authenticates users using HCMUT-SSO.
F-SYSTEM-1	There are 4 access roles: Student, SPSO, Printer Operator, Technical Support.
F-SYSTEM-2	Connects to the BKPay service for payment management.
F-SYSTEM-3	Generates reports about system usage automatically at the end of each month and year and stores them.
F-SYSTEM-4	Logs all users' printing actions. The log includes these fields: – Student ID – Printer ID – File name – Printing start time – Printing end time – Number of pages for each page size. This information should be viewable by the SPSO.
F-SYSTEM-5	Generates reports about page balance purchase, which contains: – Student ID – Student name – Purchase time – Number of pages bought over a customizable time period.
F-SYSTEM-6	Automatically sorts user-reported issues based on priority and date and time created.
F-SYSTEM-7	Shows available printers on a map interface.
F-SYSTEM-8	After a number of seconds, the system will recalculate the estimated printing completion time for all available printers.

- **For students**

ID	Requirements/Description
F-STUDENT-0	For each semester, receives a default number of A4-size pages in BKNetID account.
F-STUDENT-1	Be able to upload files of permitted types.
F-STUDENT-2	Can upload multiple files at the same time.
F-STUDENT-3	For each uploaded document, the users can configure: – Paper size: A3 or A4 – Page(s) to print – Number of copies – One-sided/double-sided printing – Orientation: Landscape or Portrait – Page layout: 1, 2, 4, 6, or 9 pages per sheet
F-STUDENT-4	Can choose printers that are currently enabled by the SPSO for print request.
F-STUDENT-5	Can remove their uploaded document(s) before printing.
F-STUDENT-6	Can see the total number of pages the request takes before submitting it.
F-STUDENT-7	Can print their desired number of copies as long as it does not exceed their page balance (note: an A3 page is equivalent to two A4 pages).
F-STUDENT-8	Can see the tentative printing completion time on each printer.
F-STUDENT-9	Can purchase more A4-size pages.
F-STUDENT-10	Can access their printing logs.
F-STUDENT-11	For each printing history record, the system should display: – Request date and time – Scheduled date and time – Files being printed – Total A3 pages – Total A4 pages – Number of pages printed – Status: pending, printing, complete, error – Printer
F-STUDENT-12	Can sort the printing log entries by date, document name, page size, or total number of pages.
F-STUDENT-13	Can view a summary that shows the total number of pages printed for each page size (e.g., A4, A3).
F-STUDENT-14	Can export or print their printing logs and summaries.
F-STUDENT-15	Can upload by dragging into the interface.
F-STUDENT-16	Can upload by Google Drive.
F-STUDENT-17	Can see the status of the printer: idle, printing, or error.
F-STUDENT-18	Can see their payment history on the page purchase interface.

- **For the SPSO:**

ID	Requirements/Description
F-SPSO-0	Can add a printer. To add a printer, SPSO should specify its brand name, model name, a short description, location (campus, building, room), and activation status.
F-SPSO-1	Can remove a printer.
F-SPSO-2	Can enable a printer.
F-SPSO-3	Can disable a printer.
F-SPSO-4	Can view system usage report any time.
F-SPSO-5	Can change: – The default number of pages – The date that the system will give the default number of pages to all students – The permitted file types accepted by the system
F-SPSO-6	Can export or print the system usage report.

- **For the Planning & Financial Affairs Office (PFAO)**

ID	Requirements/Description
F-PFAO-0	Can define and manage expenses as categories
F-PFAO-1	Can generate expense reports based on categories, time periods
F-PFAO-2	Access to the expense tracking module should be restricted to authorized personnel
F-PFAO-3	Can see total revenue from students buying more printing pages
F-PFAO-4	Can see students with overdue payment

- **For the Printer Operator**

ID	Requirements/Description
F-OPRT-0	An interface to start printing job at each printer

- **For the Technical Support Office:**

ID	Requirements/Description
F-TECHSUPPORT-0	An interface where users can report issues regarding the printing service.
F-TECHSUPPORT-1	Each report comes with the detailed error message and reason.
F-TECHSUPPORT-2	Can keep track of issue report time, priority, whether it has been resolved.
F-TECHSUPPORT-3	Can inform users about scheduled maintenance/downtime through notifications.
F-TECHSUPPORT-4	Can view the status of working printers in a list interface. If a printer fails to ping back, it will be reddened for out-of-service alert. (Every 4 hours, the printers will send its health signal to confirm that it is still working properly.)

2. **Non-functional requirements:**

- **Usability**

ID	Requirements/Description
NF-USE-0	The students shall be able to make the first print request after 10 seconds of using, and use other functions after a maximum of 30 seconds of using.
NF-USE-1	All components should be available in both Vietnamese and English.
NF-USE-2	The SPSO shall be able to use all the system functions after 20 seconds of using.
NF-USE-3	Permitted file types are shown in the uploading screen, and files can be uploaded by both choosing from the device, dragging, or copying into.
NF-USE-4	The student can upload files of a maximum size of 1GB.
NF-USE-5	The system should provide precise error messages to assist in troubleshooting issues.

- **Availability**

ID	Requirements/Description
NF-AVLB-0	The system should have a maximum downtime of 1 hour per month, excluding planned maintenance.
NF-AVLB-1	Maintenance should be announced at least one day before and show estimated time to complete maintenance work.
NF-AVLB-2	The system shall be available during normal working hours (Mon-Fri, 6:00-18:00) and on the weekend (Sat, 8:00-17:00).

- **Reliability and Disaster Recovery**

ID	Requirements/Description
NF-RLBL-0	The system should have backup storage to prevent data loss.

- **Security**

ID	Requirements/Description
NF-SEC-0	The system must secure student data, including student IDs, printing logs, and payment information.
NF-SEC-1	Access to sensitive information and system functionalities should be restricted to authorized users only.
NF-SEC-2	The system should be protected against unauthorized access, data breaches, and cyber threats.

- **Performance**

ID	Requirements/Description
NF-PERF-0	The system should be able to handle up to 1000 concurrent users.
NF-PERF-1	The server can handle a maximum of 10GB of uploaded data at a time.
NF-PERF-2	Uploading documents and processing print requests should be completed within one minute.
NF-PERF-3	Response time when users interact with the system should be less than 1 second.
NF-PERF-4	Documents are expected to be delivered with a maximum delay of two minutes beyond the assigned time.

- **Compatibility**

ID	Requirements/Description
NF-COMPAT-0	The web application should be compatible with Edge, Firefox, Safari, Chrome to ensure accessibility for all users.

- **Maintainability**

ID	Requirements/Description
NF-MNBL-0	The system should be designed and implemented using well-documented code to help future maintenance and updates easily.

- **Integration**

ID	Requirements/Description
NF-ITGR-0	HCMUT-SSO's failure rate should be less than 2%
NF-ITGR-1	BKPay's failure rate should be less than 2%

1.3 Draw a use-case diagram for the whole system. Choose at least one important module and draw its use-case diagram, as well as describe the use-cases using a table format

1.3.1 Draw a general use-case diagram for the whole system

UML use-case diagram

Use-case diagrams are usually referred to as **behavior diagrams**, used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.
Use case diagrams are used to specify:

- (external) requirements, required usages of a system under design or analysis (subject) - to capture what the system is supposed to do
- the functionality offered by a subject – what the system can do
- requirements the specified subject poses on its environment - by defining how environment should interact with the subject so that it will be able to perform its services.

In system use-case diagram, there are some typical components:

- **Actor:** a graphical representation of a person, system, or external entity that interacts with a system to achieve a specific goal. In our project, there are 3 main actors: Back officer, Janitor and Collector
- **Use case:** A set of actions, services, and functions that a system provides to its users. A use-case defines a goal that a user or system has with the system being modeled, and it describes the steps involved in achieving that goal. Basic use-cases are defined quite clearly in the diagram below, including Manage tasks, assigning tasks, to sending out messages, log in, log out,...
- **Communication links:** Represents a relationship between two instances in a system. It represents the flow of information or data between the instances, and it can be used to model the interactions between objects, components, or actors in a system.
- **Boundary of system:** Represents the external limits of the system, separating it from its environment. It defines the interface between the system and the actors that interact with it, and it specifies what the system does and does not do. So, quite clearly that Authentication (Log in / Log out module) and Task assignment module are 2 main Boundary in the system.
- **Relationships:** Used to describe the connections or associations between elements in a system, such as classes, objects, use-cases, actors, components, and so on.

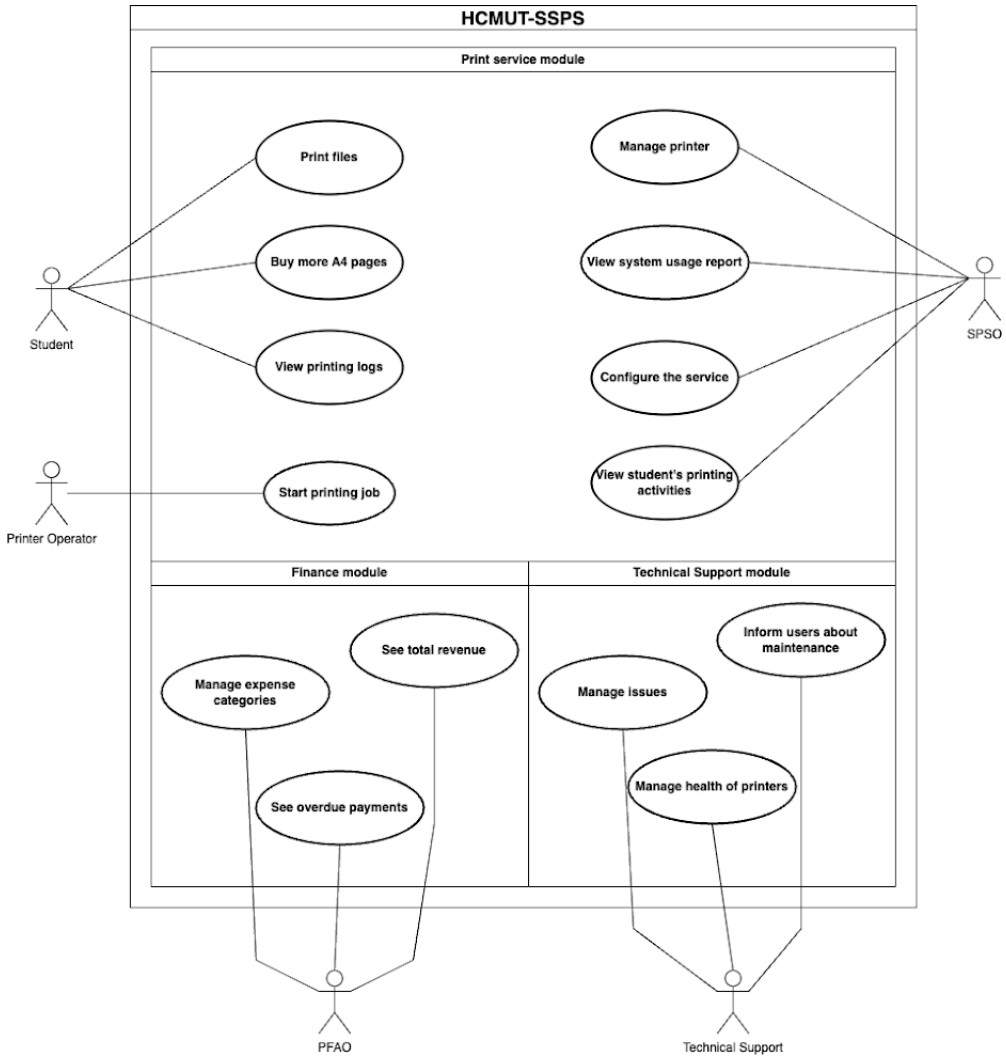


Figure 1.2: Use-case diagram for the whole system

1.3.2 Draw Smart printing service use-case diagram

Figure 1.3: Use-case diagram for the printing service

1.3.3 Describe the use-case using a table format

1.3.3.1 Upload documents by clicking

ID and Name	F-STUDENT-1 be able to upload files by clicking
Actor	HCMUT Student
Description	The student uploads their documents for printing by the clicking action.
Trigger	The student initiates the document uploading process by clicking on the file upload interface.
Pre-conditions	The student has been authenticated using their BKNetID account. The system is available. The website has been implemented and ready to execute. The student is at the "Print documents" interface. The system has access to the user's file system.
Post-conditions	The user uploads the files to the system successfully.
Normal flow	1. The system presents a designated area/button. 2. The student clicks on the designated area/button. 3. The system opens a file selection dialog. 4. The student selects files. 5. The student clicks "Upload". 6. The system closes the dialog. 7. The system displays the files being uploaded with progress bars for indicating upload status. 8. The system uploads the student's selections and updates the progress bars accordingly. 9. The system turns the progress bar into a tick symbol for every successfully uploaded file. 10. The student clicks the "Add" button to close the dialog and is back to the "Print documents" interface.
Alternate flow	1a. The system presents the option to upload files from Google Drive. 1a1. The student chooses to upload from Google Drive. 1a2. The system displays the Google log-in screen. 1a3. The student enters their Google credentials or pick their Google account to log in. 1a4. The system displays the Google Drive file selection dialog. 1a5. The student selects files from Google Drive. 1a6. The student clicks "Select" to finish selecting. 1a7. The system closes the dialog. The use case proceeds to step 9. 2a. The student drags their files onto the designated area/button. The use case proceeds to step 9.
Exceptions	• 4a. The student clicks "Cancel". 4a1. The system closes the dialog without any files selected. The use case terminates, and the student is not able to upload files • 10a. The system fails to upload one or more selected files due to a network issue or system error during the file upload process. 10a1. The system turns the progress bar into the cross symbol for the failed ones. 10a2a. The student may retry the upload for the failed files. The use case continues from step 1. 10a2b. The student may continue without the failed files. The use case continues from step 11.

1.3.3.2 Print documents

ID and Name	F-STUDENT-2 be able to print files
Actor	HCMUT Student
Description	The student prints their uploaded documents.
Trigger	The student wants to print their documents.
Pre-conditions	The student has been authenticated using their BKNetID account. The system is available. The website has been implemented and ready to execute. The student is at the "Print documents" interface. The student has uploaded their documents to the system.
Post-conditions	– The system deletes current printing session. – The system erases the uploaded files. – The system has the printing activity logged. – The student receives their printed documents.
Normal flow	1. The system creates a printing session saving files and configuration of the user. 2. The system provides a checkbox next to each uploaded file entry for selection, and quick action options: Select all, Delete. 3. The student checks a box to select one file entry. 4. The system displays the printing properties of the selection (with default values): Paper size, Page(s) to print, Number of copies, One-sided/double-sided printing, Orientation, Page layout. 5. The student inputs into the property fields. 6. The system calculates the requested pages. 7. The system updates and displays the total request pages accordingly. 8. The system presents campus choices for the user to select. 9. The student picks one campus. 10. The system estimates the printing completion time for the student request. The use case proceeds to use case "Estimate printing completion time" and return from step 10. 11. The system displays all printers on the campus's map interface, each displaying their status and having a context menu on their head showing "Printing will complete at around <time>" in which student can choose, together with an expiration timer for calculation refresh. 12. The system marks "Recommended" on the printer which completes the request fastest. 13. The student clicks on one printer from the map before the time estimation expires, then click "Select" at the bottom of the map to select the desired printer. 14. The student clicks "Print" after selecting the printer. 15. The system re-calculates the printing time in case others request the same printer while the student still decide which to use. At the same time, a new queue slot is open for contention (if it hasn't been already).

Normal flow	16. The system shows a dialog for the student to confirm their configuration and time, with 2 options "Confirm" or "Cancel". The use case repeats from step 2 to 15 until the student is satisfied with the printing preferences for all their selected file entries. 17. The student clicks "Confirm". The dialog closes. 18. The system checks the page balance. The amount requested does not exceed the page balance. 19. The system checks if the slot mentioned in step 14 is available. It is available. 19. The system sends the request to the printer controller. 20. The printer controller places the printing job in the printer queue. 21. The system logs this latest printing activity of the student. During step 17 - 21, the system displays a circular progress indicator. 22. The system removes the uploaded files from storage after 24 hours. 23. The system removes all file entries from the interface. 24. The system displays a success message to the student. 25. The system updates the status of the printing activity until it is "Complete". 26. The system updates the page balance of the student accordingly.
Alternate flow	2a. The student checks multiple boxes at once for bulk printing configuration. The use case proceeds from step 3. 2b. The student checks one or more boxes to delete. 2b1. The student clicks the "Delete" option. The use case proceeds to use case "Remove files" and goes back to step 2. Before step 7 or step 14, the student realizes they need to adjust printing properties. The use case returns to step 3. 9a. The estimated complete time is on another day, the message will be "Can take documents at <day>, <time>.". 12a. The time estimation expires, the system refreshes the calculation. The use case returns to step 8. Before step 16, the student wants to choose another printer for their request. The use case returns to proceed from step 8.
Exceptions	• 17a. The amount requested by the student exceeds their page balance. 17a1. The system shows a dialog asking the user to top up their balance. 17a2. The student clicks "Top up". The use case proceeds to "Buy more A4 pages" and proceeds to step 17. 17a2a. The student clicks "Cancel". The use case terminates, with the student unable to print their files. • 18a. If someone has secured the slot in advance, the system alerts the student that the request has failed. The use case returns to step 13. • In case there is any technical issue encountered during step 17 - 22, or at step 24, the system informs the student of the error and suggests they try again later.

1.3.3.3 Remove uploaded files

ID and Name	F-STUDENT-3 be able to remove uploaded files
Actor	HCMUT Student
Description	The student removes some unwanted uploaded documents before printing.
Trigger	Sometimes, the student needs to remove one or many uploaded documents before printing.
Pre-conditions	The student has been authenticated using their BKNetID account. The system is available. The website has been implemented and ready to execute. The student is at the "Print documents" interface. The student has uploaded their documents to the system..
Post-conditions	- The selected uploaded documents are permanently removed from the system. - Their corresponding file entries are removed from the user interface.
Normal flow	1. The quick action "Delete" is displayed but initially disabled. 2. The student selects one file entry by clicking on the corresponding checkbox. 3. The system enables the quick action "Delete". 4. The student clicks the option "Delete". 5. The system shows a confirmation dialog before the actual deletion happens. 6. The student confirms the dialog. 7. The system proceeds to permanently remove the selected file. 8. The selected file entries are removed from the user interface. 9. The system displays a success message.
Alternate flow	2a. The student selects more than one file entry by clicking on multiple checkboxes. The use case proceeds from step 3. 2b. The student clicks on the options "Select all". 2b1. All file entry checkboxes are selected. The use case proceeds from step 3.
Exceptions	• 6a. The student clicks the option "Cancel". 6a1. The system returns to the previous state without deleting any file. The use case terminates with the student unable to remove their files. • 7a. The system encounters some technical issue and the deletion cannot go on. 7a1. The system displays a message to the student informing about the error and suggesting they try again later. The use case terminates with the student unable to remove their files.

1.3.3.4 Buy more A4 pages

ID and Name	F-STUDENT-4 be able to buy more A4 pages
Actor	HCMUT Student
Description	The student purchases more A4 pages to top up their page balance.
Trigger	The student wants to purchase more A4 pages
Pre-conditions	The student has been authenticated using their BKNetID account. The system is available. BKPay is connected to the system. The website has been implemented and ready to execute. The user is in the "Buy A4 pages" interface.
Post-conditions	A payment entry is created in BKPay. The student can see their latest payment request in the interface. The page balance of the student is topped up.
Normal flow	1. The system presents an input field for the student to specify the number of A4 pages they would like to buy, a disabled button that says "Purchase", and a "Cancel" button. In addition, there is a list of 10 latest transactions of the students being displayed on the interface. 2. The student enters the desired number of A4 pages to purchase. 3. The system shows the associated cost and enable the "Purchase" button. 4. The student clicks "Purchase". 5. The system prompts a confirmation dialog with 2 options "Confirm" and "Cancel". 6. The student clicks "Confirm". 7. The system communicates with BKPay to create a payment entry. During step 8, there is a circular progress indicator to signify the process of entry creation. 8. The system tops up the page balance of the student. 9. The system logs this latest purchase activity of the student. 10. The system updates the list of 10 latest transactions on the interface. 11. The system notifies the user of the successful page purchase and clears the student's input in the field.
Alternate flow	None
Exceptions	• 4a. The student clicks "Cancel". 4a1. The system clears the input field. The use case terminates, with the student unable to make purchase. • 8a. The system could not connect to BKPay 8a1. The system informs the student of the technical issue and suggest they try again later. The use case terminates, or returns to proceed from step 4.

1.3.3.5 View printing history

ID and Name	F-STUDENT-5 be able to view own printing history
Actor	HCMUT Student
Description	The student views their printing history.
Trigger	The student needs to see their printing history.
Pre-conditions	The student has been authenticated with their BKNetID. The system and database are available. The website has been implemented and ready to execute. The student is at the "Printing history" interface.
Post-conditions	The printing history is displayed to the student.
Normal flow	1. The system queries all the records from the database. 2. The system calculates the total number of pages printed for each page size. 3. The system displays all the records in paginated batches, each of which is 10 records. 4. The student browses and clicks on one record. 5. The system displays the details of the record in a modal window. 6. The student clicks out of the modal window. 7. The modal window disappears and the student is back to the interface in its previous state.
Alternate flow	1a. The user has no log history and the system displays "No printing history found" instead.
Exceptions	• 1a. There is a technical issue with the database connection 1a1. The system informs the student of the error and suggests them try their request later.

1.3.3.6 View account

ID and Name	F-STUDENT-6 be able to view own account
Actor	HCMUT Student
Description	The student views their account.
Trigger	The student needs to see their account.
Pre-conditions	The student has been authenticated with their BKNetID. The system and database are available. The website has been implemented and ready to execute. The student is at the "My Profile" interface.
Post-conditions	The student's account information is displayed.
Normal flow	1. The system queries information from the database. 2. The system presents: the page balance, the summary of the total number of printed pages for each page size. 3. The student views their information.
Alternate flow	None
Exceptions	• 1a. There is a technical issue with the database connection 1a1. The system informs the student of the error and suggests them try their request later.

1.3.3.7 Add a printer

ID and Name	F-SPSO-1 be able to add a printer
Actor	SPSO
Description	The SPSO adds a new printer to the printers list.
Trigger	The SPSO wants to add new printers.
Pre-conditions	The SPSO has been authenticated using their BKNetID account. The system and database are available. The website has been implemented and ready to execute. The SPSO is at the "Printer Management" interface.
Post-conditions	A printer is added to the available printers list.
Normal flow	1. The SPSO can add a printer by selecting the symbol (+) from the interface. 2. The system prompts a dialog. The dialog contains input fields for **Brand**, **Model**, **Campus**, **Building**, **Description**, **Status**. 3. The SPSO enters information into these fields. 4. The system validates information input by the SPSO. 5. The SPSO submits the information. 6. The dialog closes. 7. The system updates the database of the new information.
Alternate flow	3a. If any required information is missing or incorrect, the system displays an error message "Incorrect input format", and the SPSO corrects the information before re-submitting.
Exceptions	• 1b. If the front-end fails to query the database, displays error messages on the frontend. • 5b. If the printer lose connection or fail to execute, the system automatically retries the operation after a short delay.

1.3.3.8 Remove a printer

ID and Name	F-SPSO-2 be able to remove a printer
Actor	SPSO
Description	The SPSO removes one or multiple printers from the printer list.
Trigger	The SPSO wants to remove printer(s) from the available list.
Pre-conditions	The SPSO has been authenticated using their BKNetID account. The system is available. The website has been implemented and ready to execute. The SPSO is at the "Printer Management" interface. There has to be at least one printer available in the list.
Post-conditions	Printer(s) is removed from the available list.
Normal flow	1. The system shows a list of printer entries which also includes details **(Brand, Model, Campus, Building, Description, Status)** and action buttons **(Enable/Disable, Remove)**. Each entry also has a check box. 2. SPSO clicks on the **Remove** button of one entry, which is the trash can icon. 3. The system prompts a dialog with 2 options **Confirm** and **Cancel**. 4. SPSO clicks **Confirm**. The dialog closes. 5. The system deletes the printer out of the list of printers in the database. 6. The system removes the printer entry from the interface.
Alternate flow	2c. The SPSO selects multiple checkboxes. • 2c1. The system shows the button **Delete**. • 2c2. The SPSO clicks **Delete**. The use case continues from step 4.
Exceptions	• 1b. If the front-end fails to query the database, displays error messages on the frontend. • 5b. If the printer lose connection or fail to execute, the system automatically retries the operation after a short delay.

1.3.3.9 Change status of a printer

ID and Name	F-SPSO-3 be able to change status of a printer
Actor	SPSO
Description	The SPSO toggles the status of a printer in the printer list.
Trigger	SPSO wants to toggle status of one or multiple printers from the available list.
Pre-conditions	The SPSO has been authenticated using their BKNetID account. The system is available. The website has been implemented and ready to execute. The SPSO is at the "Printer Management" interface. At least one printer has been added to the system.
Post-conditions	The status of the printer is switched.
Normal flow	1. The system shows a list of printer entries which also includes details **(Brand, Model, Campus, Building, Description, Status)** and action buttons **(Enable/Disable, Remove)**. 2. SPSO clicks on the **Enable** button of the printer entry, whose **Status** is **Disabled**. 3. The system prompts a dialog with 2 options **Confirm** and **Cancel**. 4. SPSO clicks **Confirm**. 5. The system switches the value of **Status** to **Enabled**.
Alternate flow	8a. If the SPSO declines to confirm the activation request, the system cancels the activation process. The printer remains in the disabled state. 2a. SPSO clicks on the **Disable** button of the printer entry, whose Status is **Enabled**. • 2a1. The system prompts a dialog with 2 options **Confirm** and **Cancel**. • 2a2. SPSO clicks **Confirm**. • 2a3. The system switches the value of **Status** to **Disabled**. The use case may repeats from step 1.
Exceptions	• 1b. If the front-end fails to query the database, displays error messages on the frontend. • 5b. If the printer lose connection or fail to execute, the system automatically retries the operation after a short delay.

1.3.3.10 Configure the service

ID and Name	F-SPSO-4 be able to configure the service
Actor	SPSO
Description	The SPSO configure the printing-related variables.
Trigger	SPSO needs to change the printing service configuration variables.
Pre-conditions	The SPSO has been authenticated using their BKNetID account. The system is available. The website has been implemented and ready to execute. The SPSO is at the "System configuration" interface.
Post-conditions	The system updates the new default page grant. Or, the system updates the new permitted file extensions. Or, the system updates the new page grant date.
Normal flow	1. The system shows input fields for **Default page grant**, **Grant date**, and **Permitted file extensions** already populated with previously set values. 2. SPSO inputs into the fields. 3. SPSO clicks the **Save** button. 4. The system saves those changes. 5. The system changes the color of the button and change the text to **Saved** to indicate success.
Alternate flow	3b. If the input contains invalid text or out of bound number the System shows "Invalid input" and the user recorrect the input before submitting. The use case may repeats from step 1.
Exceptions	• 5b. The system fail to adapt new config and the system record input information to exception logs.

1.3.3.11 View system usage report

ID and Name	F-SPSO-5 be able to view system usage report
Actor	SPSO
Description	The SPSO view the the system usage statistics.
Trigger	SPSO needs to view the system report.
Pre-conditions	The SPSO has been authenticated using their BKNetID account. The system is available. The website has been implemented and ready to execute. The SPSO is at the "System usage" interface. There has been at least one generated system usage report.
Post-conditions	The system displays the system usage report file on the PDF viewer. Or, the system usage report is downloaded to the SPSO's machine successfully.
Normal flow	1. On the interface, the system presents a list of generated reports, each of which has a download button. 2. SPSO clicks on one report. 3. The system fetches the report from the storage. 4. The system opens the report in the PDF viewer.
Alternate flow	2a. The SPSO clicks the **Download** button of a report. • 2a1. The system downloads the report to the SPSO's machine. The use case may repeats from step 1.
Exceptions	• 1a. The system fails to connect to the file storage system, shows "Internal Error".

1.3.3.12 View a student's printing activities

ID and Name	F-SPSO-6 be able to view a student's printing activities
Actor	SPSO
Description	The SPSO views printing activites of students.
Trigger	SPSO needs to monitor student activities.
Pre-conditions	The SPSO has been authenticated using their BKNetID account. The system is available. The website has been implemented and ready to execute. The SPSO is at the "System usage" interface.
Post-conditions	The interface loads the student's printing history successfully. The SPSO can view the printing history of a particular student.
Normal flow	1. The system shows: • A search bar with 2 options: search using student's name or student's ID. • Two date pickers labeled "From" and "To" for the SPSO to filter the results based on date range. • The option to filter results based on the printer. 2. The SPSO enters student's name, or student's ID, or pick date range, or select a printer into the search bar. 3. The system show results based on the query.
Alternate flow	None
Exceptions	• 3a. The system failed to retrieve student information and requests. • 3b. The SPSO's query did not return anything. The system shows "No matching results".

1.3.3.13 Start the printing job of a printer

ID and Name	F-OPERATOR-1 be able to start the printing job of a printer
Actor	Printer operator
Description	The campus staff resumes the printing process of the printers.
Trigger	The printer has a new pending activity. Or, the printing activity has been complete and the pending activity with the earliest request time needs to be started.
Pre-conditions	The Campus staff has been authenticated using their BKNetID account. The system is available. The website has been implemented and ready to execute. The printers are connected properly and printing properly. The Campus staff is at the "Printing Activity Hub" dashboard.
Post-conditions	The printer prints the new printing job. The printing activty is moved to the **Printing** section.
Normal flow	1. The system shows: • The system shows a list of active printers. • The campus staff clicks on a printer. • The system redirects the user to another page. This page, which has 3 sections, shows the all the pending jobs in the **Pending** section, one job currently being printed in the **Printing** section, and a list of all complete jobs within the last hour in the **Complete** section. • A new printing job is submitted to the printer. The interface prompts a dialog asking the campus staff to start this new printing job. • The campus staff presses **Start**. • The dialog disappears and the printing activity is moved to the **Printing** section. • The printer prints this new printing activity.
Alternate flow	None
Exceptions	• 3a. The system is disconnected with the printer. – 3a1. The system displays the exception message "Unable to connect to the printer". • 5b. The printer cannot print the job. – 5b1. The system displays the exception message "Unable to print".

Chapter 2

Task 2: System modeling

2.1 Draw an activity diagram to capture the business process between systems and the stakeholders for the important module(s) chosen in Task 1.3.

Activity diagram

The Unified Modeling Language includes several subsets of diagrams, including structure diagrams, interaction diagrams, and behavior diagrams. **Activity diagrams**, along with use case and state machine diagrams, are considered behavior diagrams because they describe what must happen in the system being modeled.
Activity diagrams help people on the business and development sides of an organization come together to understand the same process and behavior.
Some of the most common components of an activity diagram include:

- **Action:** A step in the activity wherein the users or software perform a given task. In Lucidchart, actions are symbolized with round-edged rectangles.
- **Decision node:** A conditional branch in the flow that is represented by a diamond. It includes a single input and two or more outputs.
- **Control flows:** Another name for the connectors that show the flow between steps in the diagram.
- **Start node:** Symbolizes the beginning of the activity. The start node is represented by a black circle.
- **End node:** Represents the final step in the activity. The end node is represented by an outlined black circle.

2.1.1 Print Service Module Diagram:

To view more clearly, follow this link: https://drive.google.com/file/d/1tQbA7iTYvqXeQ6hYyFrb6m15j_TT6zll/view?usp=drive_link

2.1.2 Description:

1. The student logs in to the system. If their provided credentials are valid, they are redirected to their profile page.
2. From their profile page, they can either choose to upload documents for printing, or purchase more A4 pages, or view their printing history, or log out of the system, at the same time.
3. If the student needs to print their documents:
   a) They first upload their documents to the system which stores them after being uploaded.
   b) The student can then configure the printing properties and choose the desired time periods that they can receive the documents, for every single document or all of them.

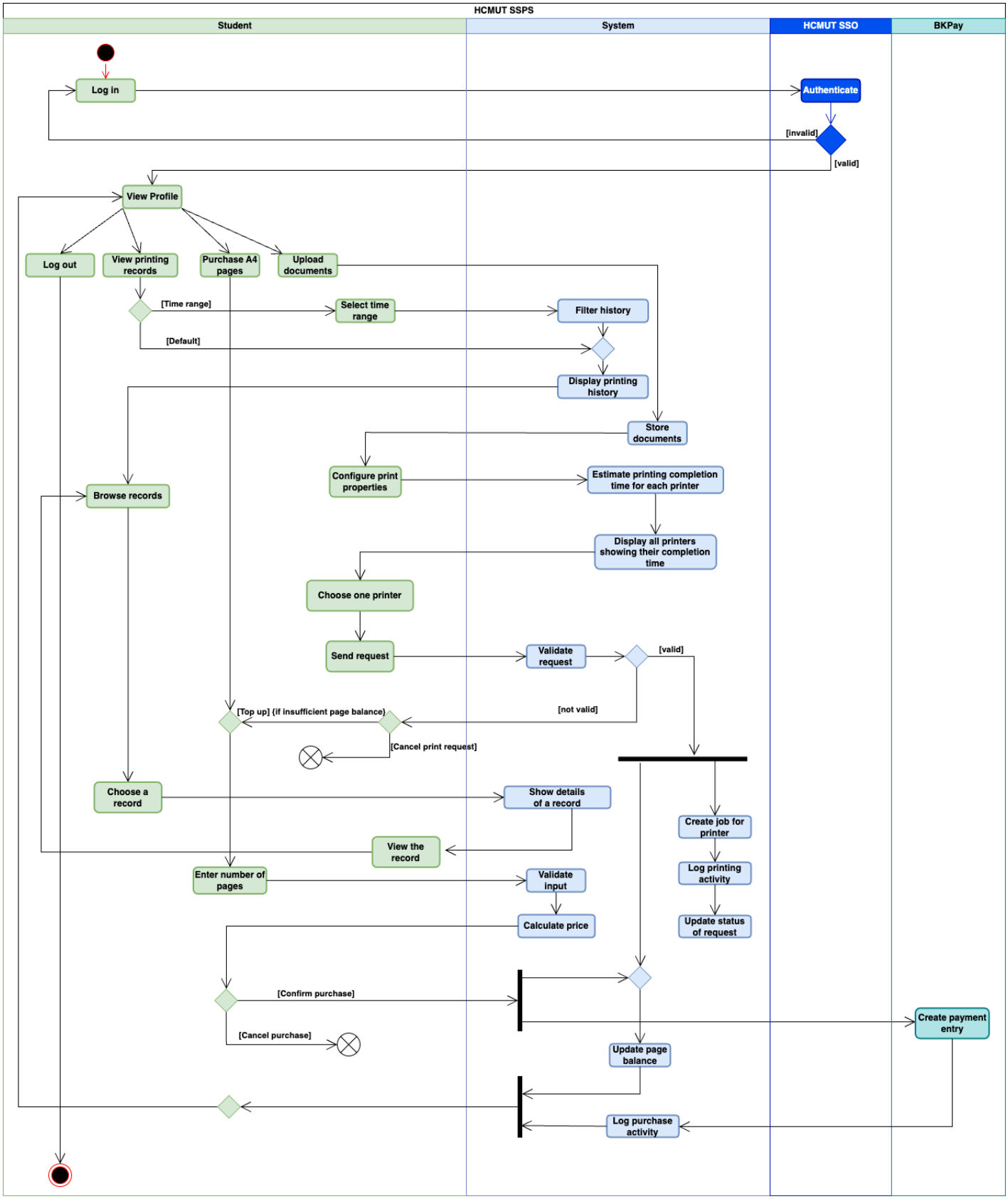


Figure 2.1: Activity diagram of the Print Service Module

c) Based on the chosen desired time periods, the system will check availability of the printers at those periods, then display them on a map interface with the printers satisfy(those are not busy by many students have scheduled the receiving time in the chosen periods) marked as green, otherwise red.

d) Furthermore, the system will calculate the most suitable green printer based on busyness and distance to mark as **Recommended** to help the student on their decision better.

e) From there, student can choose a printer to view the details schedule of the chosen printer, if some periods in the schedule is busy, the periods will be colored red. If student finds out the time that suit best, they can press that exact time on the printer schedule and confirm request, else they can exit current printer to choose the other printer to view detail schedule. (Note that the system just indicated busy printers/time periods of one printer as red, student can still choose them at their own pace).

f) After submits their print request, the system checks the requested amount of pages against the student's page balance. If it does not exceed the page balance, the request proceeds normally as the system updates the page balance of the student and the printing record, which can be view in the printing history. If the amount exceeds the page balance, they need to purchase more pages to satisfy the request, or else, cancel the request.

4. When the student needs to purchase more A4 pages:

   a) They enter the number of pages they wish to purchase

   b) The system validates the input (numeric and within a certain range of values) and calculates the corresponding price.

   c) The student confirms this price. The system creates a payment entry in BKPay and updates the page balance of the student. The student must wait until the system completes these two activities before they can turn to another activity.

5. When the student views their printing history:

   a) They can choose a time range to filter their history. Otherwise, the system displays the latest records in paginated batches.

   b) The student browses their history and can choose a record to view details. If the record status is "not yet being printed", or in other words, it is a scheduled record that not yet come to the printing time, student can edit the configuration or pick another scheduled time if current time is not right before the scheduled time(depends on the total number of pages, the system will automatically start printing a few minutes before scheduled time, so that when student arrive at the right time, all documents are ready).

2.1.3 Print Service Management Module Diagram:

To view more clearly, follow this link: https://drive.google.com/file/d/1KDpIOuziRyF2lXoE21GeL8YIB_Dofot-/view?usp=drive_link

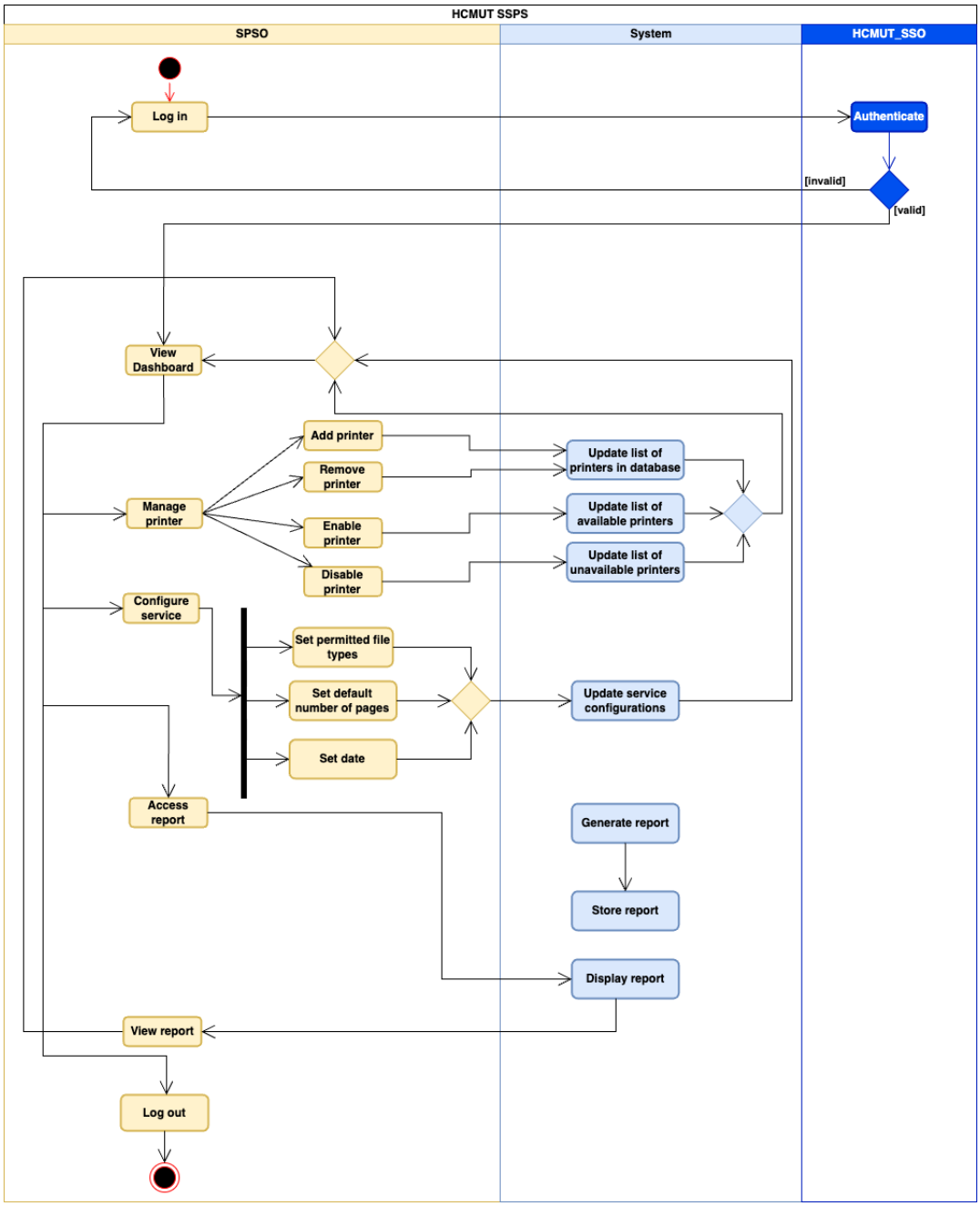


Figure 2.2: Activity diagram of the Print Service Management Module

2.1.4 Description:

1. The SPSO logs in to the system. If their provided credentials are valid, they are redirected to the administrative dashboard.
2. From the dashboard, they can either choose to manage printers, configure the print service, and view system generated reports.
3. For printer management, the SPSO can add a new printer, enable an available printer and disable an available printer. At the same time, the system updates the list of printers stored in our database.

4. The SPSO can update the print service configuration by customizing a list of permitted file types, setting the default page balance each student will receive and the date on which they do receive it.
5. The system automatically generates usage reports at the end of each month and at the end of each year and stores them in its database. When the SPSO wants to view the report, the system will display all reports, grouped by the generation year. The SPSO can choose to view a report.

2.2 Draw sequence diagrams for the important module(s) chosen in Task 1.3.

Sequence diagram

UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when. Sequence Diagrams captures:

- The interaction that takes place in a collaboration that either realizes a use case or an operation (instance diagrams or generic diagrams)
- High-level interactions between user of the system and the system, between the system and other systems, or between subsystems (sometimes known as system sequence diagrams)

2.2.1 Diagram

2.2.1.1 Upload files to the system

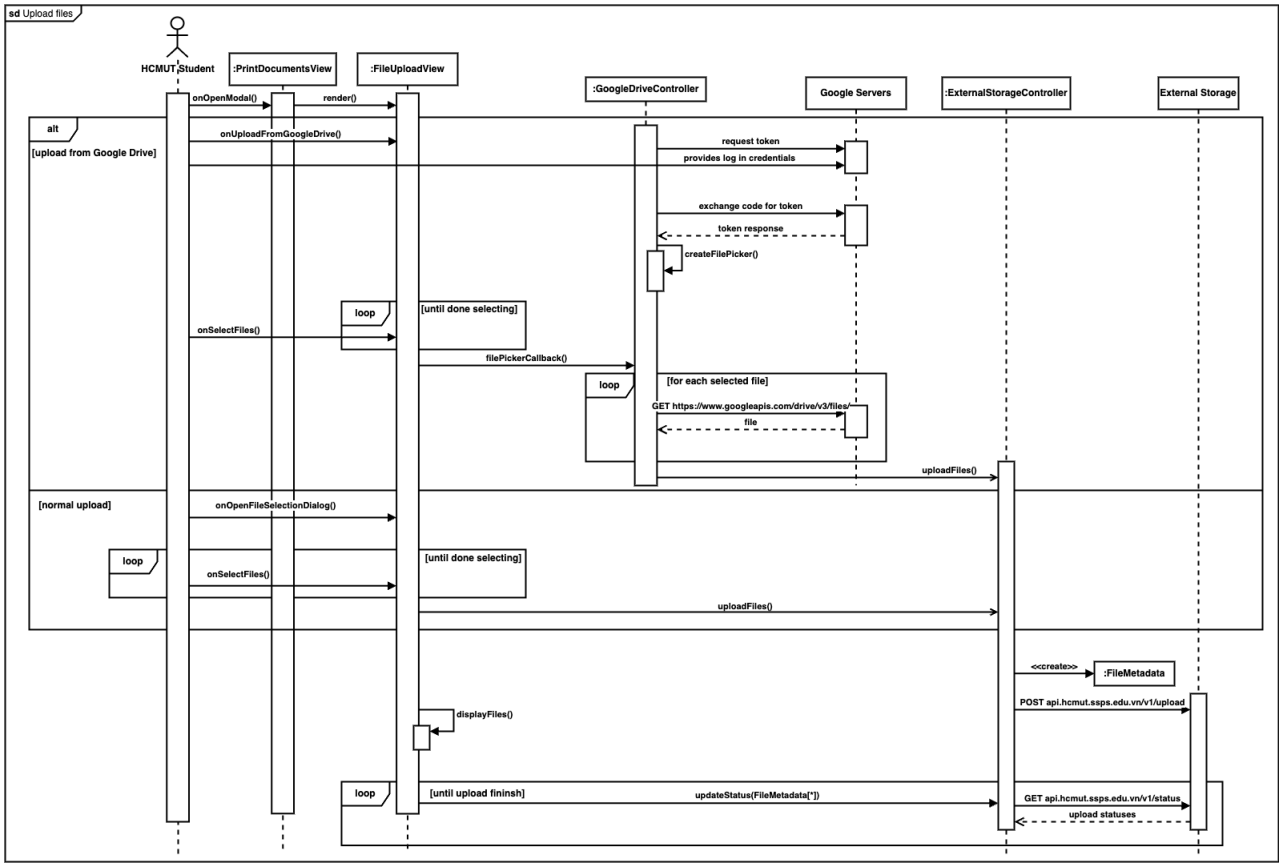


Figure 2.3: Sequence diagram of the upload use-case

To view more clearly, follow this link: https://drive.google.com/file/d/1-kZOO-fajXuKntoYIZ0tIlIDuj7oC664/view?usp=drive_link
From the Print Documents view, the student clicks the button to upload files. The callback `onOpenModal()` opens

the file upload view - rendered as a modal window. The student either chooses to upload from their Google Drive account, or upload from their local file system.

If the student chooses to upload from Google Drive, they have to provide credentials to log in their Google Account, in order for the `GoogleDriveController` object to obtain the token to call the Google Drive API to render the Google Drive file picker. In the file picker, the student browses their files until they are done and click "Select". The view will invoke the method `filePickerCallback()` in the `GoogleDriveController` to download the files from the Google Servers to the system's external storage via the method `uploadFile()` in the `ExternalStorageController`.

If the student chooses to upload from their local file system, they click on a designated area in the modal window. The view will invoke the method `openFileSelectionDialog()` to open the file selection dialog. The student will browse and select their files. The view will automatically invoke `uploadFile()` asynchronously in `ExternalStorageController` when the student is done selecting their files and the file selection dialog closes, so the student doesn't have to wait for all the files to be uploaded in order to resume interaction.

Before uploading to the external storage via the POST request, the `ExternalStorageController` will create a `FileMetadata` object for each selected file to both represent them in the view and for managing them in the storage. The view will call its `displayFiles()` method to show the files, together with a corresponding progress bar. The view will continuously invoke the `updateStatus()` method in the `ExternalStorageController` object to update the progress bars accordingly.

2.2.1.2 Print documents

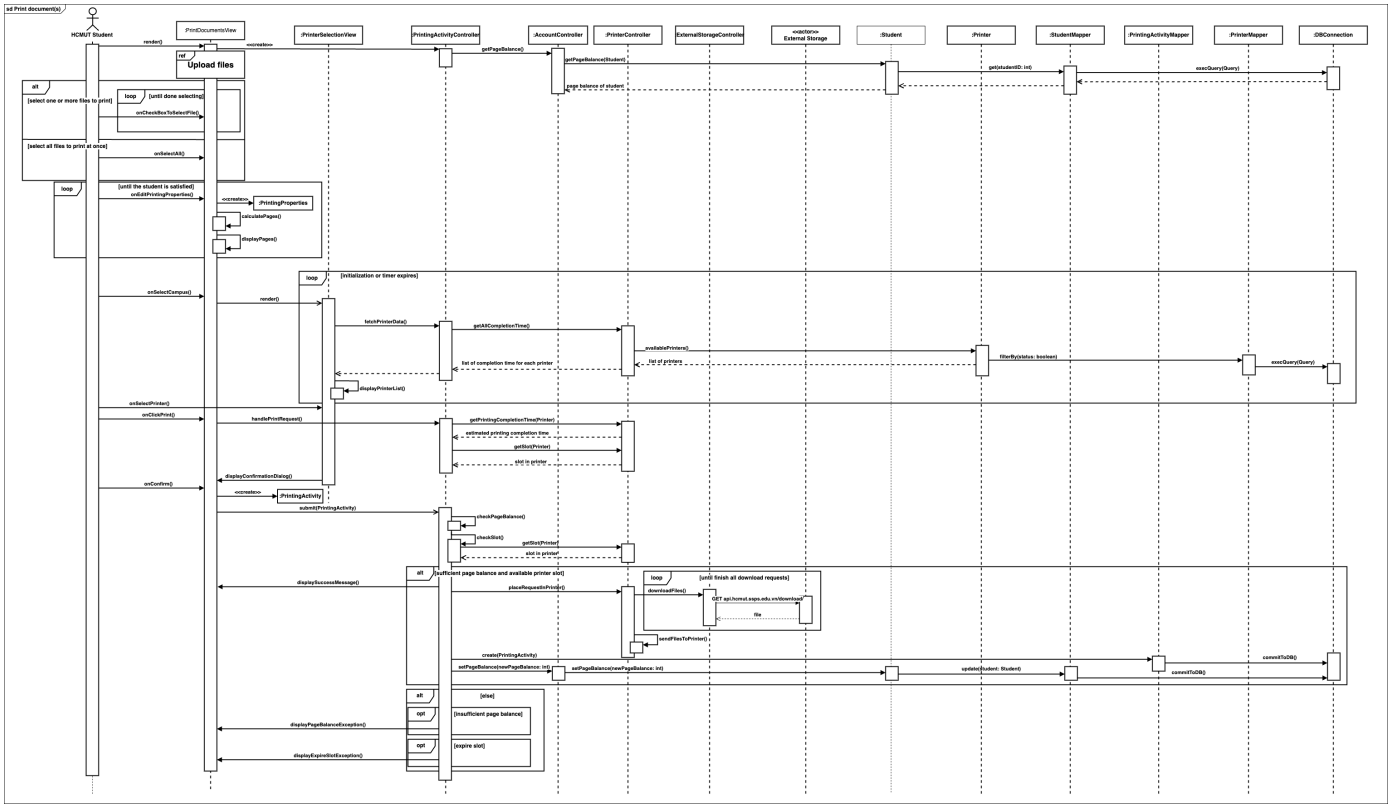


Figure 2.4: Sequence diagram of the print document use case

To view more clearly, follow this link: https://drive.google.com/file/d/13btPxgNhGqAOVenOo-8oM1QtyXok5d9s/view?usp=drive_link
The URI /print-documents redirects the students to the Print Documents View. The system fetches the page balance of the student during the rendering of the view, to help subsequent interactions of the student with the system quicker.

Firstly, the student will upload files to print to the system. Then, each files will be shown with a checkbox in an entry. The student clicks to select some files. This action will be handled by the `onCheckBoxToSelectFile()` callback of the view. Alternatively, they can opt for the **Select All** quick action button to mark all files as selected. This action will be handled by the view's `onSelectAll()` callback.

The student edits the printing properties of their documents. When they either change the number of copies, customize the page range to print, or the pages per sheet, the system will calculate and display the total request pages accordingly.

Then, on the view, the student selecting the campus will trigger the **Printer Selection** view to render. The render will be asynchronous because the student will not have to wait for it to finish rendering before they can resume their interaction. This view will call the method `fetchPrinterData()` of the `PrintingActivityController` object. This method, in turn, invokes `getAllCompletionTime()` of `PrinterController` to return the list of each printer's completion time for the print job if they submit the request within 30 seconds. Then, the list of available printers with their estimated printing completion time will be displayed. The calculation will expire after 30 seconds. On expiration, the `PrintingActivityController` will `fetchPrinterData()` again to obtain a fresh estimation.

The action of the student clicking the **Print** button to submit the request will be handled by the `onClickPrint()` callback of the view. Then, the `PrintingActivityController` object invokes the `handleRequest()` method to get the printing completion time for the printer the student has selected again to make sure the calculation is updated. The slot information will be returned so that when we call the method `checkSlot()`, this slot information will be matched against to determine if the slot has been taken by another student's request.

The view then displays a confirmation dialog for the student to confirm their print request. When the student confirms, a new `PrintingActivity` object will be created. This object will be the argument of the `submit()` method of the `PrintingActivityController`. The `PrintingActivityController` will call `checkPageBalance()` to make sure the request does not exceed the page balance, and `checkSlot()` to ensure that while the student making decision, the slot in the printer's queue has not been taken by another student's request. If all checks pass, the `PrinterController` will place the request as a new job in the printer. The `ExternalStorageController` will download the files from the external storage to the printer. The printing activity is logged to the database. The `PrintingActivityController` also updates the page balance of the student accordingly. In case any check fails, the view will display the exception message to the student.

2.2.1.3 Buy printing pages

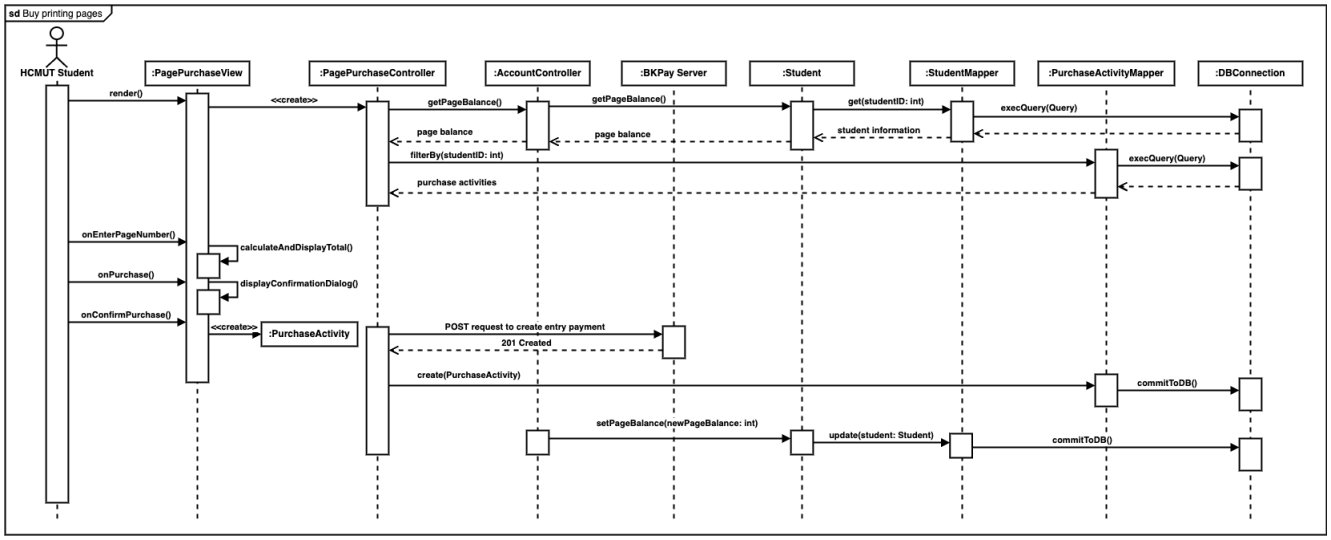


Figure 2.5: Sequence diagram of buying printing pages

To view more clearly, follow this link: https://drive.google.com/file/d/1PFpEaW82jU3ppKGlaro1nnfjS_u1nAXT/view?usp=drive_link The student accesses the Page Purchase view by the URI /buy-page. The `PagePurchaseController` object will call the `getBalance()` method of the `AccountController` object to obtain the page balance to display on the screen. The list of past purchase activities of that student will also be fetched from the database and displayed.

The student enters the quantity they want to purchase. The view captures this event, calculate, and display the total price they have to pay accordingly.

When the student clicks "Purchase", the view call `displayConfirmationDialog()` to ask the confirmation of the user. The user clicks confirm. The callback `onConfirmPurchase()` creates a new `PurchaseActivity` object and passes this to the `PagePurchaseController` to log this new activity to the database. Then, a POST request is issued from the `PagePurchaseController` to the BKPay server to creates a new transaction entry (The student will complete the payment process in BKPay). The `AccountController` object will update the page balance of the student accordingly.

2.2.1.4 View system usage

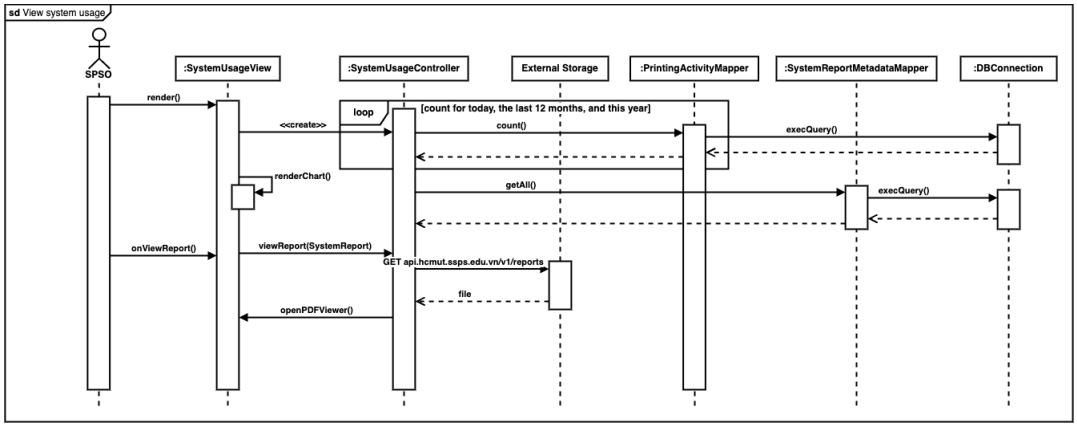


Figure 2.6: Sequence diagram of viewing system usage

To view more clearly, follow this link: https://drive.google.com/file/d/1nEnNbs-HDGuDcrYVftpeKc7DCIQAvkdS/view?usp=drive_link The SPSO accesses the `SystemUsageView` via the URI /system-usage. On rendering, the `SystemUsageController` object will obtain total printing requests made in the system by calling the method `count()` of the `PrintingActivityMapper` 3 times, once for counting the total made in the respective day, once for the total in the respective year, and another for the last 12 months. The count information will be used to render a histogram-like chart on the view.

Then, the `SystemUsageController` gets all the System Report's metadata from the database to display on the view.

The SPSO clicks on one report. The callback `onViewReport()` is called on the view. This method then makes `SystemUsageController` call `viewReport()` to issue a GET request appended with the metadata to the External Storage for the real report. The view opens a PDF viewer to display this report.

2.2.1.5 Add a printer

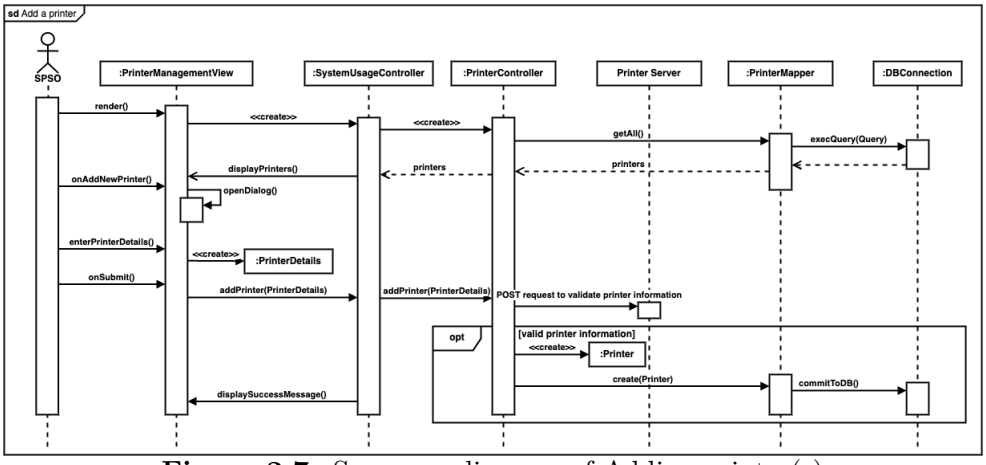


Figure 2.7: Sequence diagram of Adding printer(s)

To view more clearly, follow this link: https://drive.google.com/file/d/1GC-xiMhqdPhQ6sHByRFcb8mLjBlDM5Bb/view?usp=drive_link The SPSO accesses the `PrinterManagementView` via the URI /manage-printer. The SystemUsageController object calls the `PrinterController` to obtain the information of all printers in the database. Once the data return, the view displays the list of printer information.

When the SPSO adds a new printer, the view opens a dialog. Inside this dialog, the SPSO enter the `PrinterDetails`. This event creates a `PrinterDetails` object, which is passed to the `PrinterController` when the SPSO submits the addition request. The `PrinterController` sends a POST request with the details of the new printer to the printer server for validation. If the details are valid, the `PrinterController` creates a new Printer object and logs this to the database. Finally, it displays the success message to the view when added successfully.

2.2.1.6 Start printing job of a printer

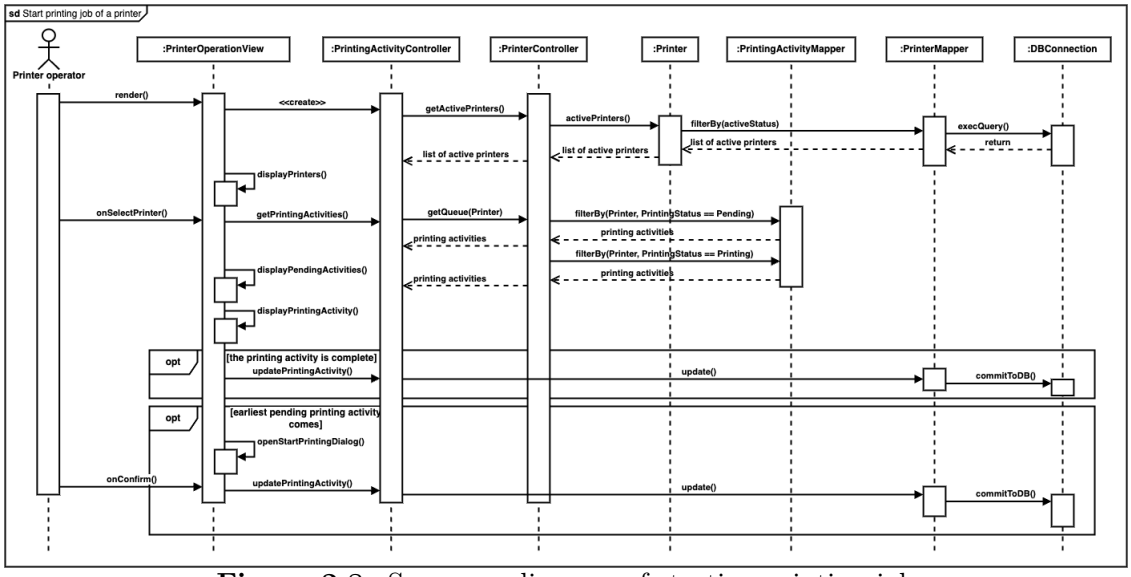


Figure 2.8: Sequence diagram of starting printing job

To view more clearly, follow this link: https://drive.google.com/file/d/13OvUQL4H7AYSer9OcfdevejQBVSLqSaz/view?usp=drive_link The operator accesses the `PrinterOperation` view. The view renders. On rendering, the view invokes the `PrintingActivityController` object to call the associative `PrinterController` object to get all active printers and displays them. The campus staff then clicks on the printer they operate on. The `PrinterController` object will call its method `getQueue` to obtain the printing activities whose status are **Pending** or **Printing** and the view displays them in designated sections. If the activity that is being printed is complete, its status will be updated in the database and the view moves it to the designated **Complete** section. The earliest pending printing activity comes next and the view opens a dialog to ask the operator to let the printer start this activity. The operator confirms and the status of this new activity changes from **Pending** to **Printing**.

2.3 Draw a class diagram of the important module(s) chosen in Task 1.3 as comprehensive as possible

Class diagram

- In software engineering, a **class diagram** in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.
- The purpose of class diagram is to:
  1. Shows static structure of classifiers in a system
  2. Diagram provides a basic notation for other structure diagrams prescribed by UML
  3. Helpful for developers and other team members too
  4. Business Analysts can use class diagrams to model systems from a business perspective
- A UML class diagram is made up of:
  - A set of classes
  - A set of relationships between classes

2.3.1 Diagram

Follow this link to have a better view: https://drive.google.com/file/d/1JAOA6y3lWRh5RBAO_pOEGCAnD-T7hRN2/view?usp=drive_link

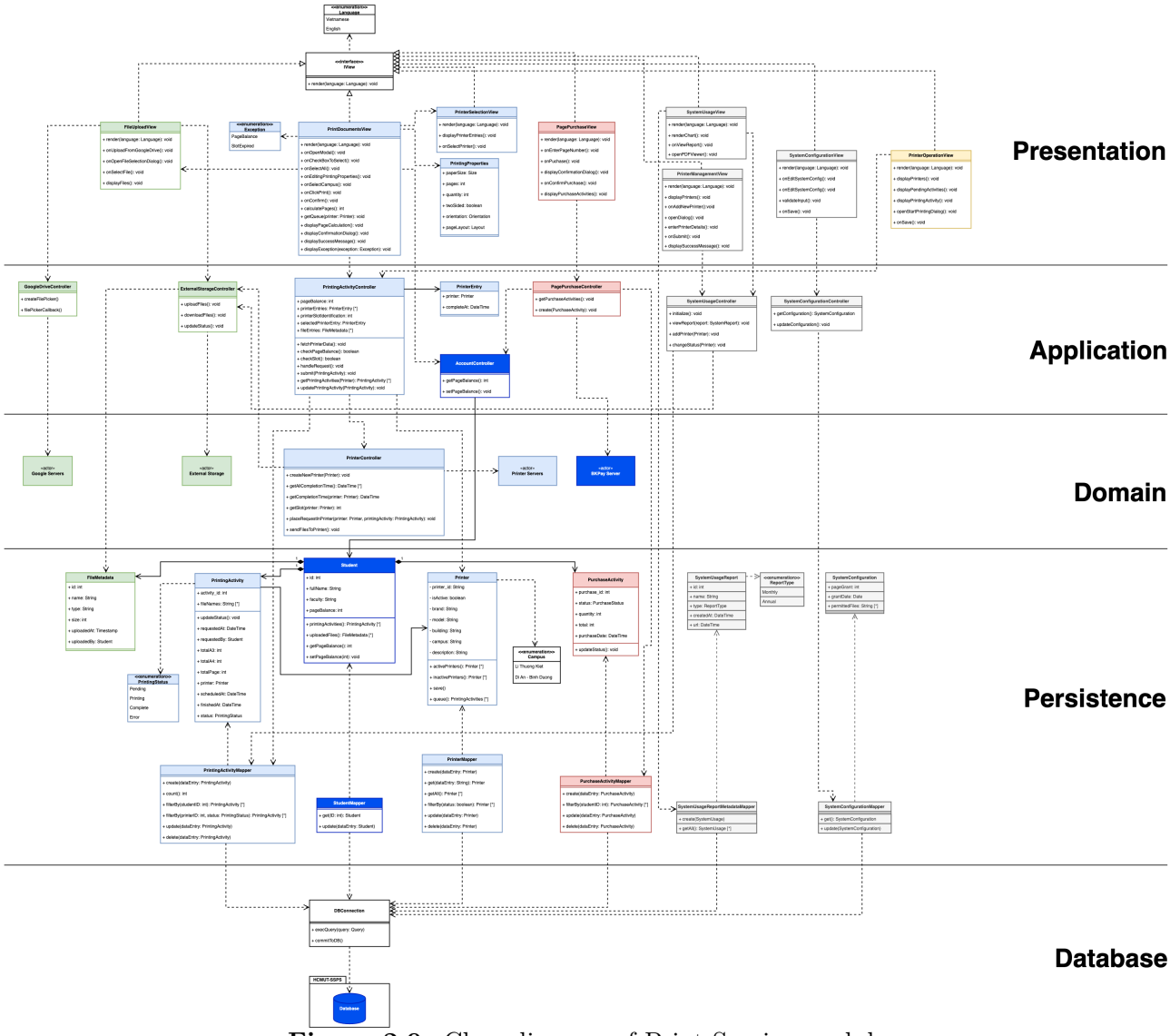


Figure 2.9: Class diagram of Print Service module

2.4 **Develop MVP 1 as user interfaces of either a Desktop-view central dashboard for a particular module (the same with the module used in task 2.1). Decide yourself what to include in the view. Use a wireframe tool like Figma or Adobe XD, or Illustrator**

Figma prototype link: https://www.figma.com/proto/GV43VeoOzvh5ibQYXL7Eib/SE-(Group-12)?type=design&node-id=1-78993&t=ZsqfQaSkr2zZuasW-0&scaling=min-zoom&page-id=1%3A2&starting-point-node-id=1%3A78993&show-proto-sidebar=1

2.4.1 Login screen

Figure 2.10: Login screen

2.4.2 Key screens of student's flow

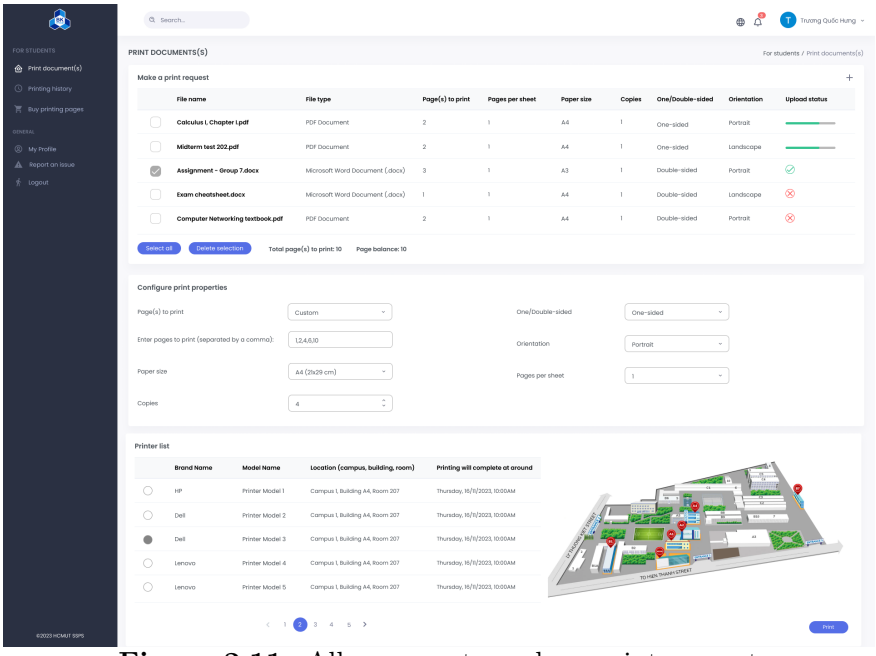


Figure 2.11: Allow users to make a print request

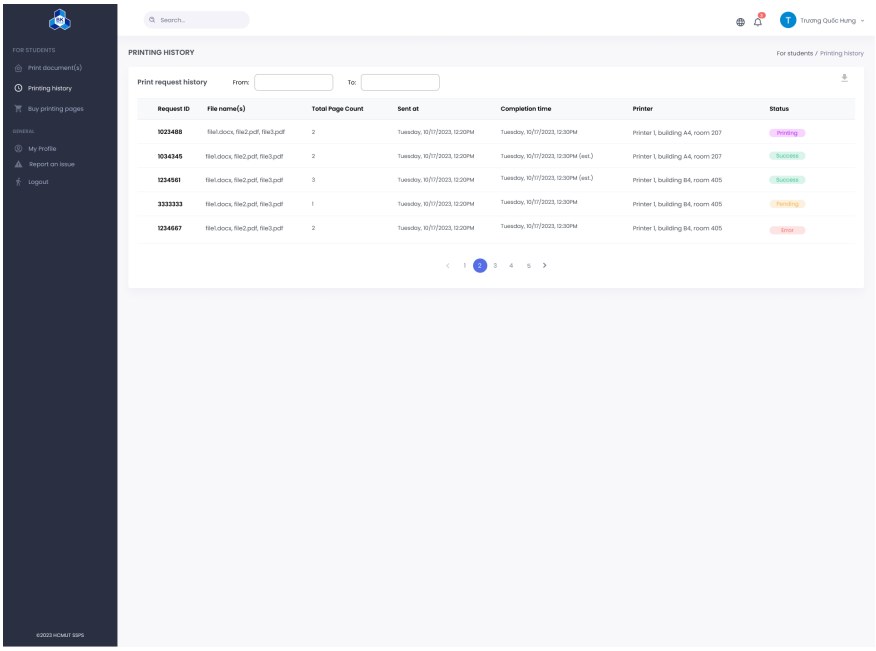


Figure 2.12: Allow users to view their printing history

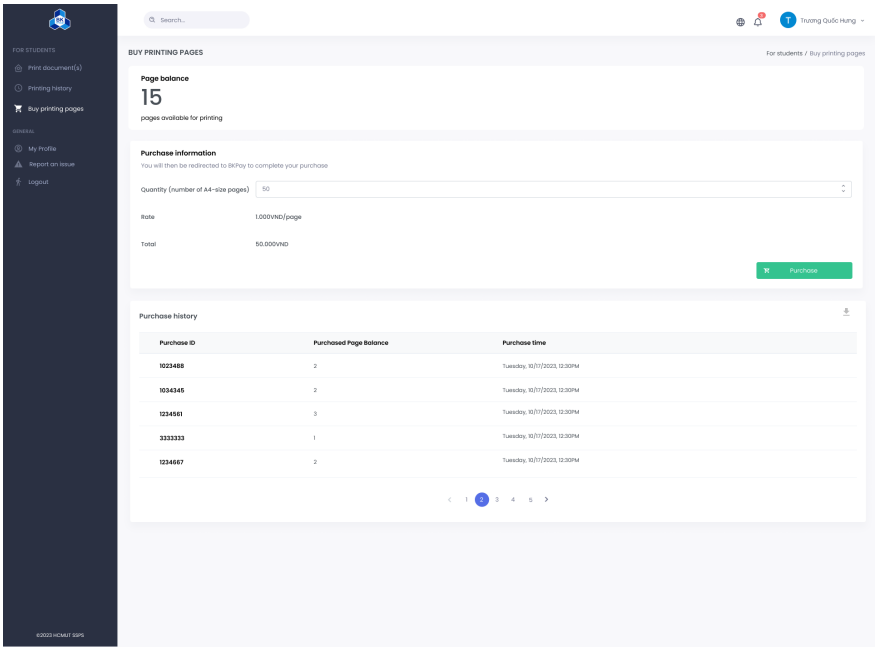


Figure 2.13: Allow users to buy printing pages

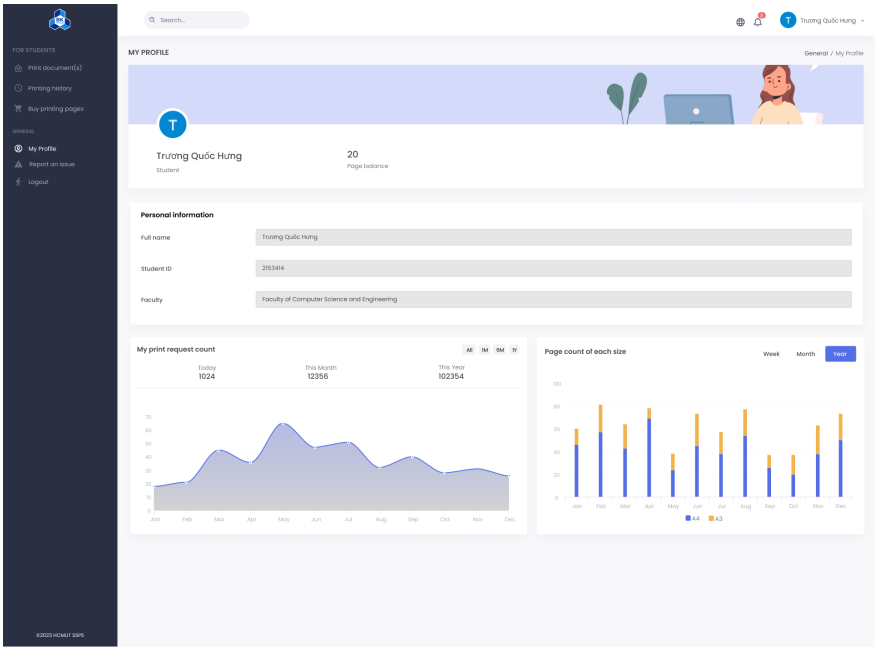


Figure 2.14: Allow users to view their profile

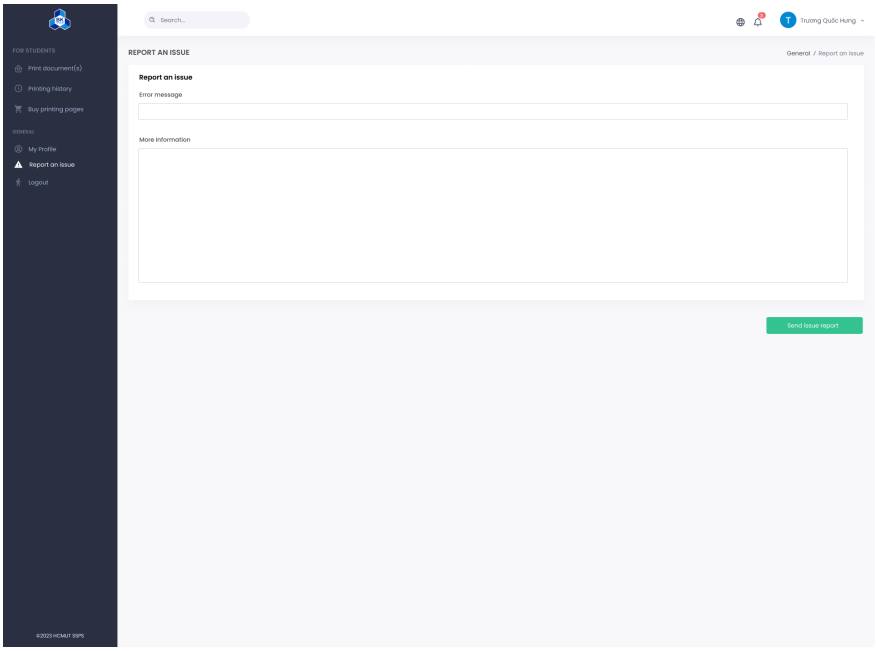


Figure 2.15: Allow users to report an issue with the SPSS service

2.4.3 Key screens of SPSO flow

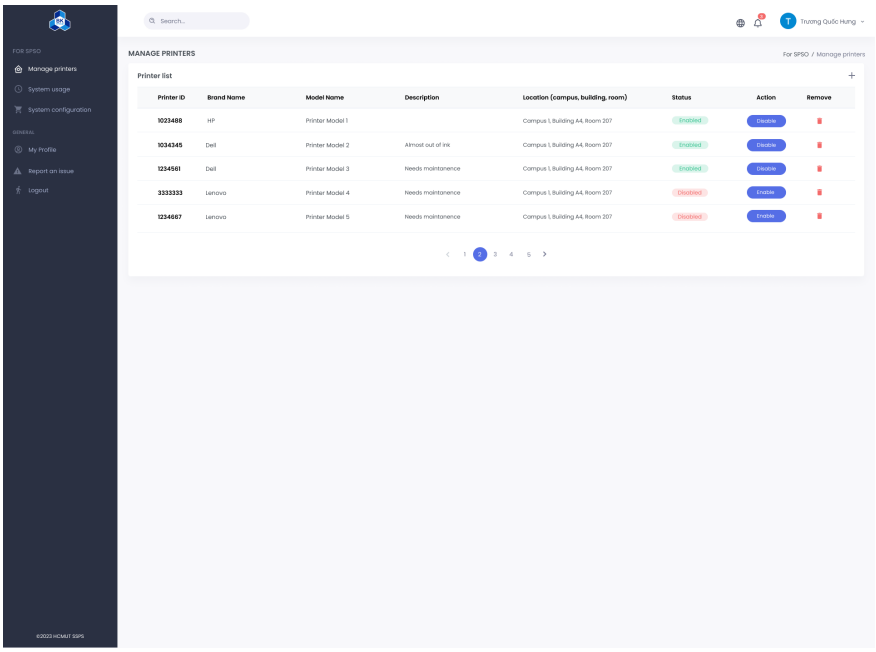


Figure 2.16: Allow SPSO to manage printers

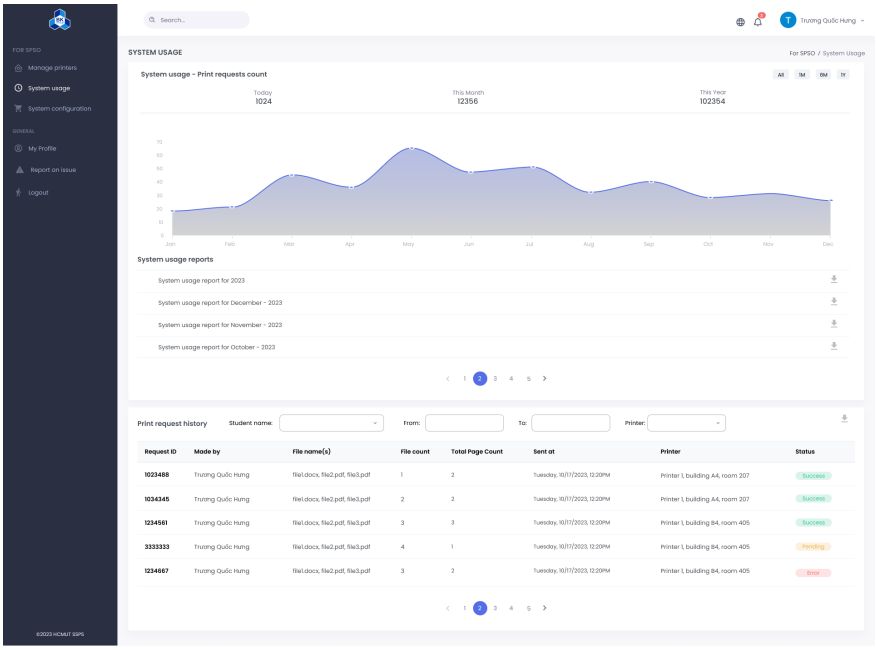


Figure 2.17: Allow SPSO to view all printer requests made

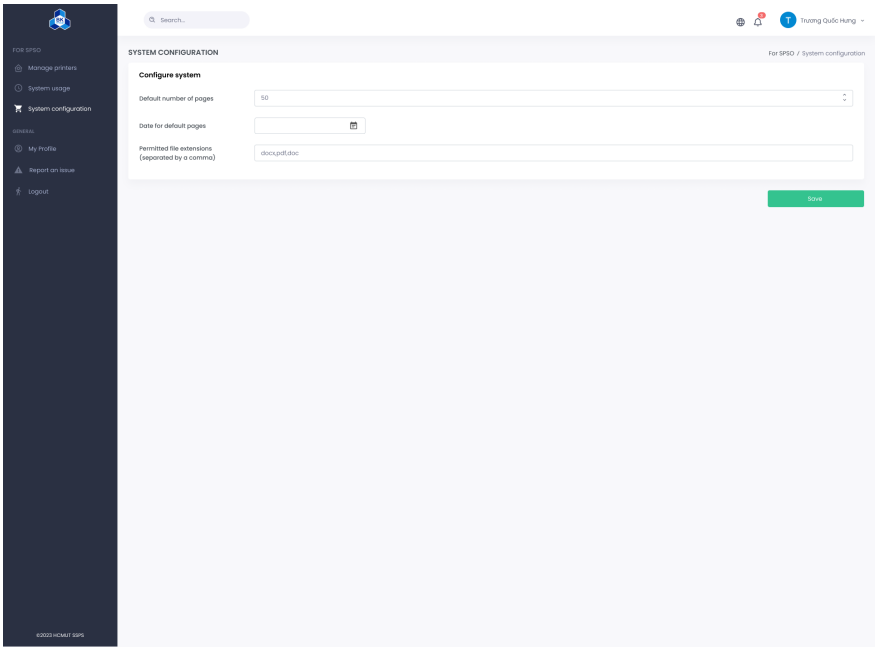


Figure 2.18: Allow SPSO to configure system properties

2.4.4 Key screens of Financial Office flow

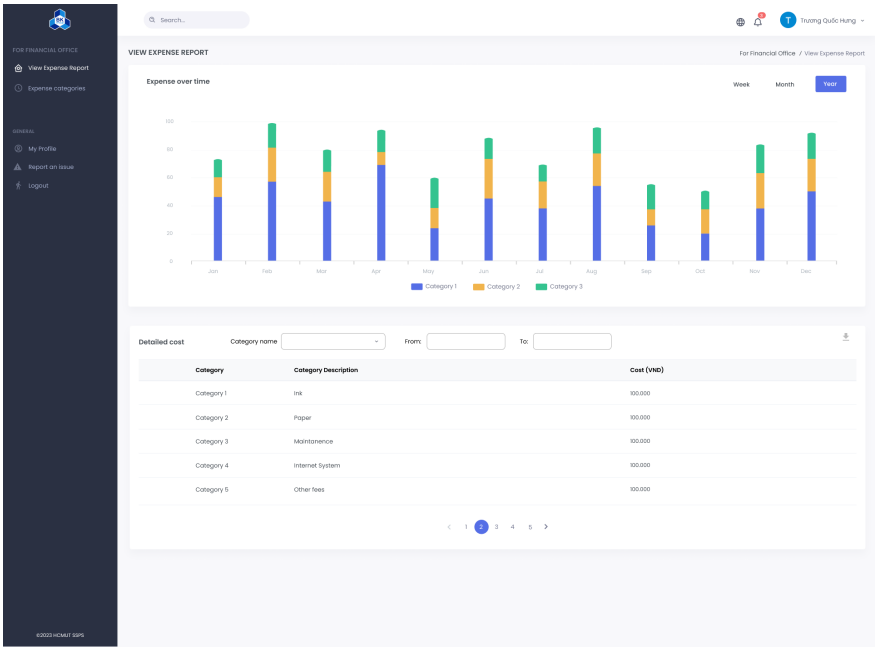


Figure 2.19: Allow the Financial Office to view financial reports

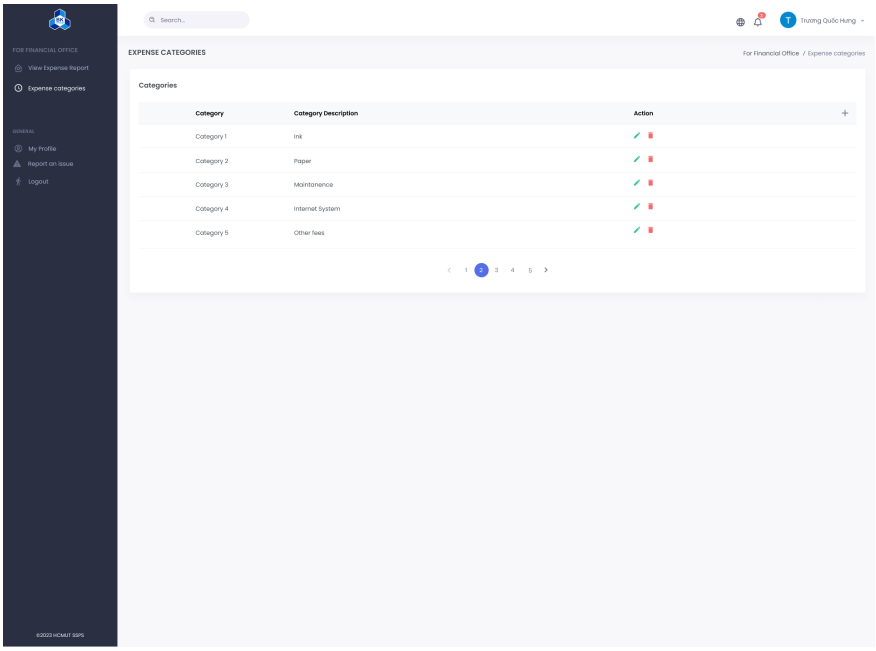


Figure 2.20: Allow the Financial Office to manage expense categories

2.4.5 Key screens of Technical Support Office flow

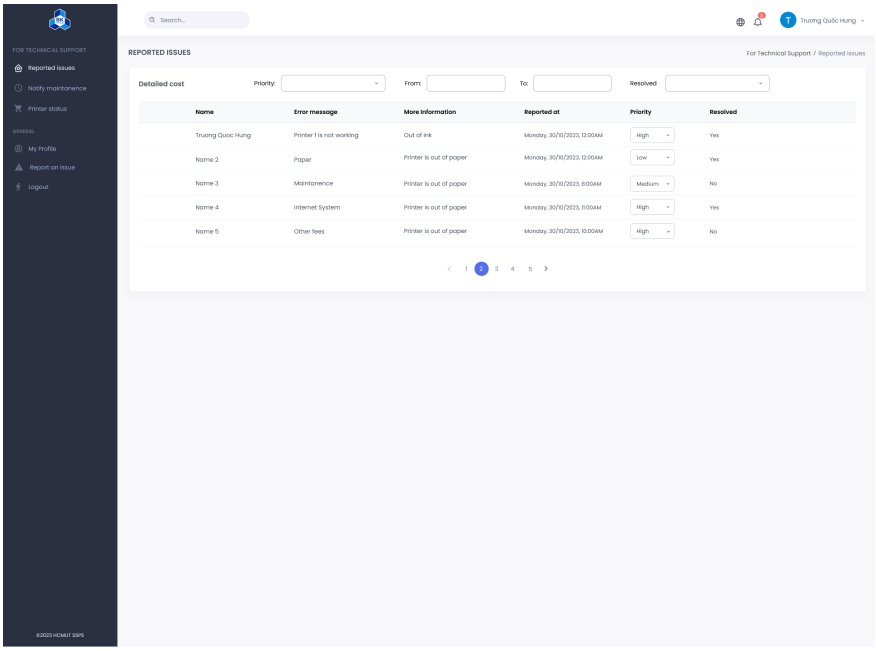


Figure 2.21: Allow the Technical Support Office to view user-reported issues

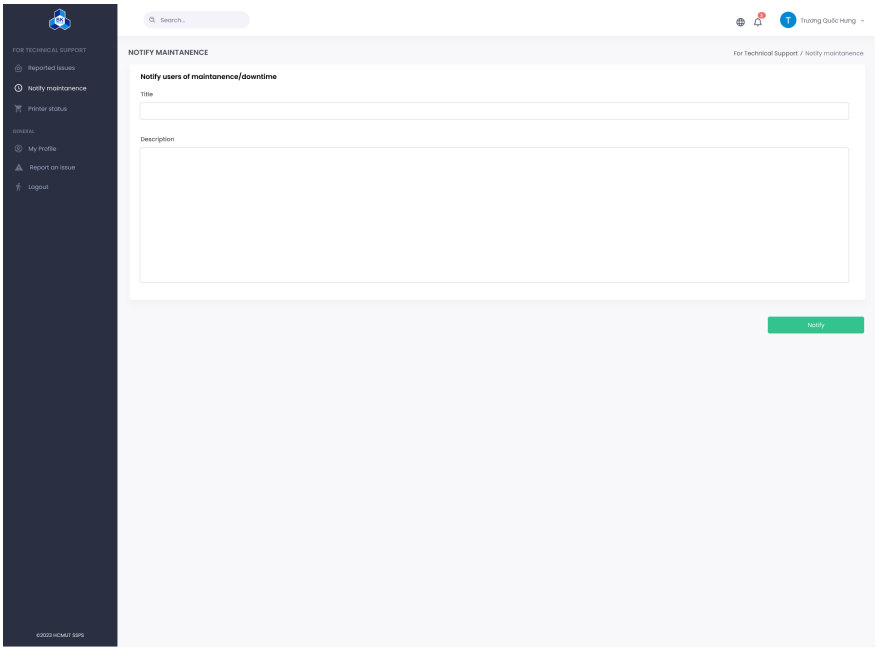


Figure 2.22: Allow the Technical Support Office to notify users about scheduled maintenance/downtime

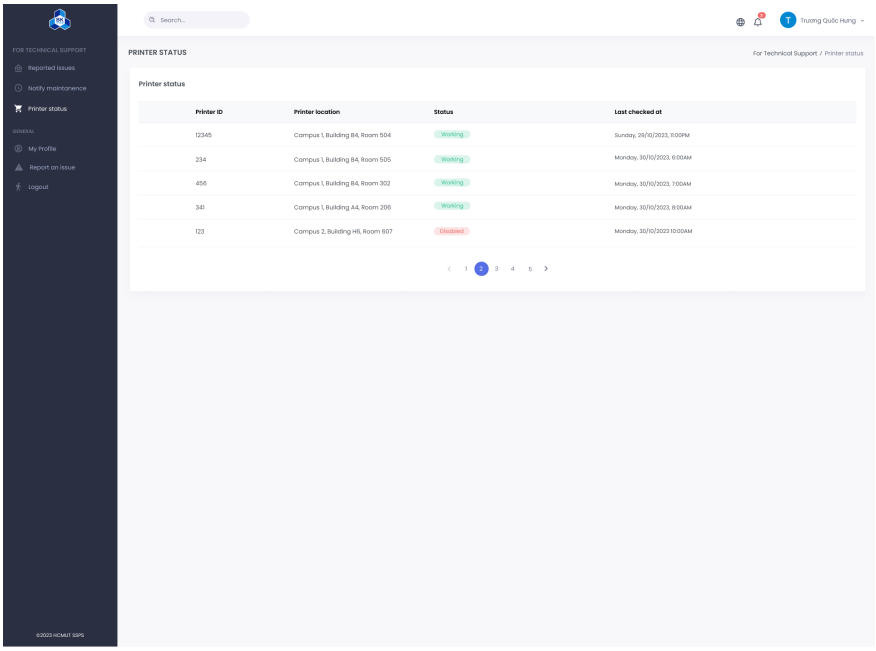


Figure 2.23: Allow the Technical Support Office to view status of printers

Chapter 3

Task 3: Architecture design

3.1 Use a layered architecture to design the HCMUT-SSPS system. Describe how will you present your User Interface. Describe how will you store your data. Describe how you will access to external services/ APIs.

3.1.1 Use a layered architecture to design the HCMUT-SSPS system

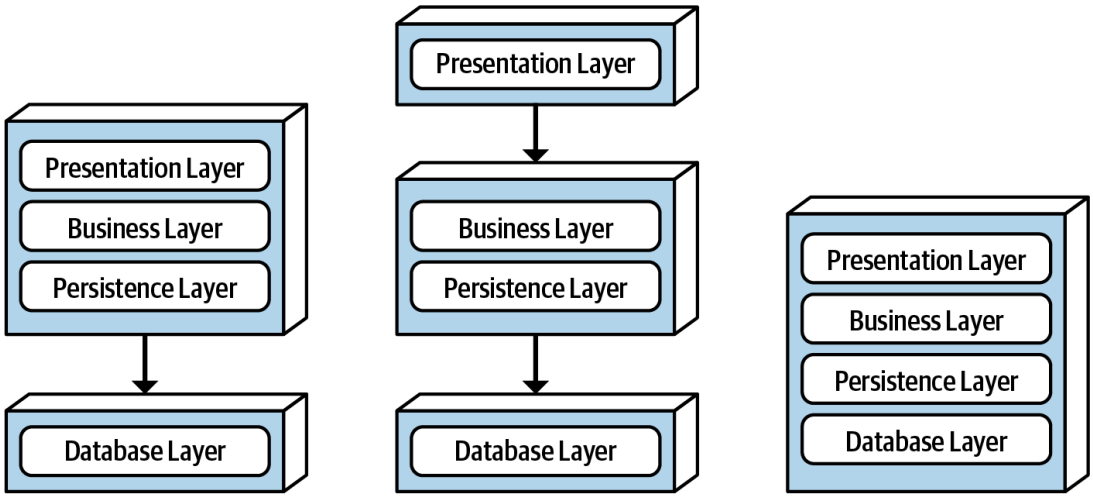
3.1.1.1 Theoretical basis: Layered Architecture Style

Definition: Layered architecture patterns are n-tiered patterns where the components are organized in horizontal layers. This is the standard method for designing most software and is intended to be self-contained. This means that all of the components are linked but not dependent on one another. This architecture has four layers, each with a connection between modularity and component within it. They are, from top to bottom:

- **Presentation layer:** It includes categories associated with the presentation layer.
- **Business (or, Application) layer:** It includes business logic.
- **Persistence layer:** It handles functions such as Object Relational Mapping's.
- **Database layer:** This is where all data are kept.

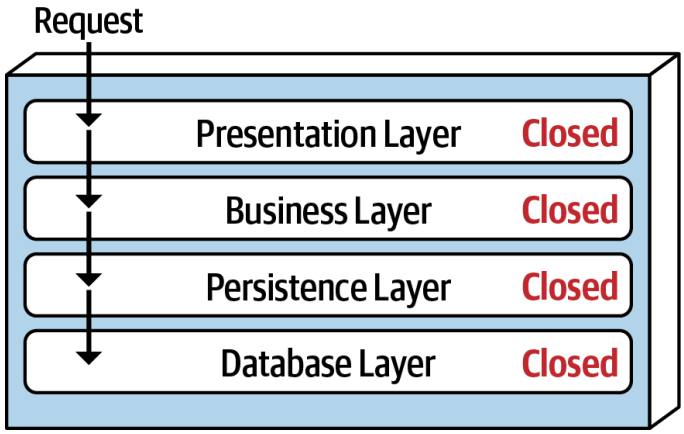
There are accepted variants of the above model.

The leftmost variant combines the presentation layer, the business layer and the persistence layer into a single



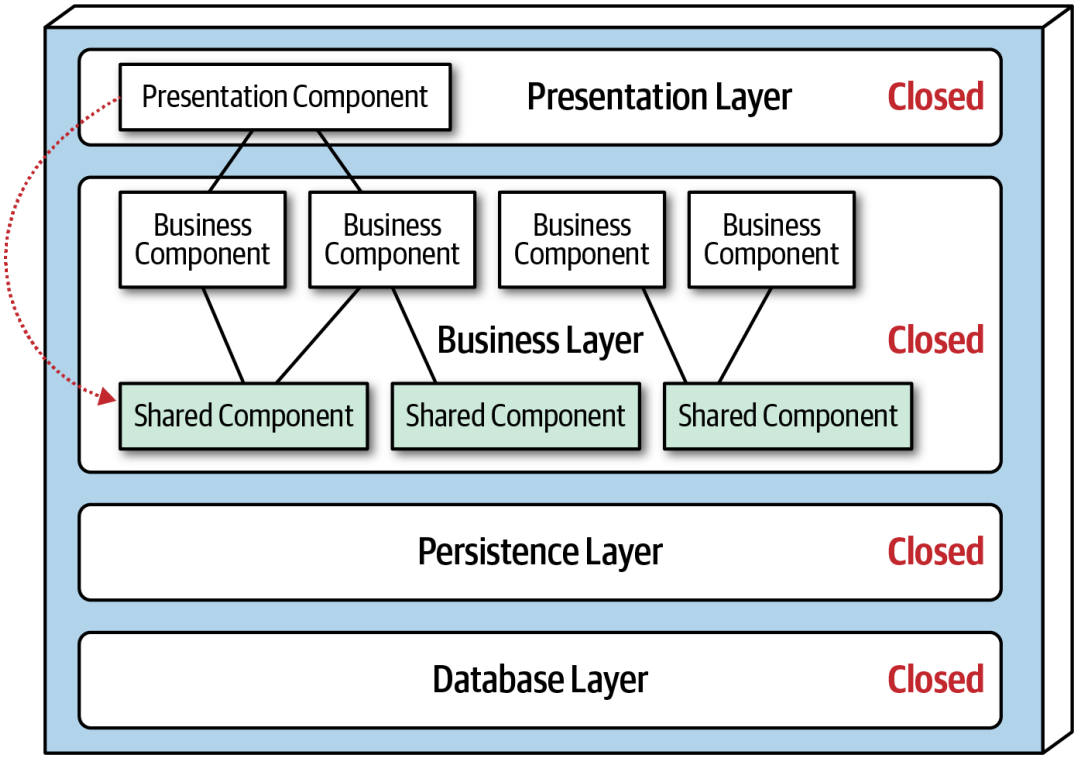
deployment unit, while the database layer is external physical database. If all layers are closed, a request must traverse all layers from top to bottom.

In this closed manner:



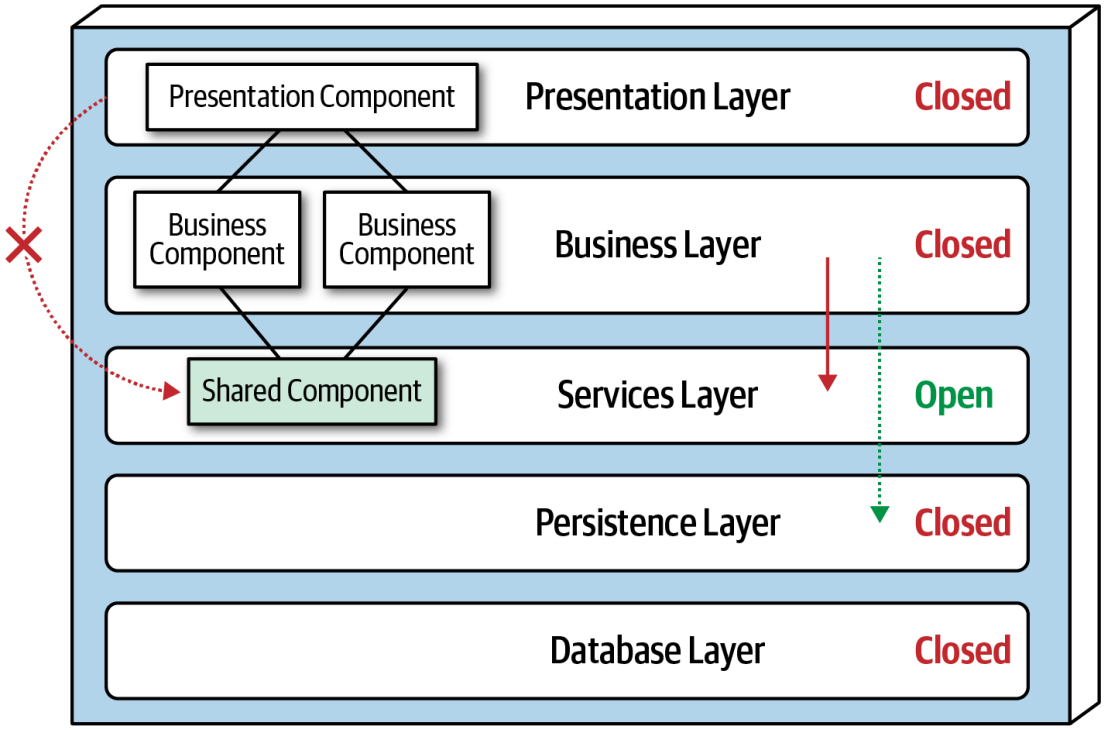
- All 'similar' components are grouped together. Having 'similar' components together means that everything relevant to a single layer remains in that single layer. This allows for a clean separation of component types and also aids in gathering similar programming code in one location.
- It provides layers of isolation. The layers become independent of one another when they are isolated. Thus, changing the database from an Oracle server to a SQL server will have a significant impact on the database layer but will have no effect on the other layers. Assume you have a custom written business layer and want to replace it with a business rules engine. If we have a well-defined layered architecture, the change will not affect other layers.

While closed layers provide layer isolation and hence assist isolate change within the design, there are situations when open layers make sense. Assume there are shared objects inside the business layer that provide common functionality for business components (for example, date and string utility classes, auditing classes, logging classes, and so on). Assume an architectural choice prohibits the presentation layer from using these common business items. This scenario is challenging to regulate and control since the presentation layer has architectural access to the business layer, and hence to the shared objects inside that layer.



To make sure the presentation part of a software system can't directly use certain shared business items, you can create a new services layer in the system's design. This new layer keeps the presentation part from getting to those shared business items because the business layer, where they are, is closed off.

But there's a catch. The new services layer has to be marked as "open." If it's not, the business layer would be forced to go through the services layer to get to the layer below, which is where data is stored. By marking the services layer as open, the business layer can choose whether to use this layer or skip it and go directly to the next layer down.



Using the idea of open and closed layers helps establish how different layers in a system's architecture connect with request flows. It gives developers the details and guidance needed to grasp the access restrictions between layers in the architecture. When there's a lack of documentation or clear communication about which layers are open or closed (and the reasons behind it), it often leads to architectures that are tightly-coupled and fragile. Such architectures become challenging to test, maintain, and deploy.

- Cost and simplicity: Monolithic architecture
  - Don't have the complexities associated with distributed architecture styles.
  - Are simple and easy to understand
  - Are relatively low cost to build and maintain.
- Deployability:
  - Don't have the complexities associated with distributed architecture styles.
  - Are simple and easy to understand
  - Are relatively low cost to build and maintain.
- Testability:
  - Can mock or stub components (or even an entire layer).
  - Even small changes can force hours of re-executing the entire regression test suite.
- Reliability:
  - No network traffic, bandwidth, and latency.
  - Monolithic deployments.
- Elasticity and scalability:
  - Monolithic deployments.

Architecture characteristic	Star rating
Partitioning type	Technical
Number of quanta	1
Deployability	★
Elasticity	★
Evolutionary	★
Fault tolerance	★
Modularity	★
Overall cost	★★★★★
Performance	★★
Reliability	★★★
Scalability	★
Simplicity	★★★★★
Testability	★★

  - No architectural modularity.
  - Applications can only scale to a certain point.
  - Selective scaling requires very complex design techniques.

- Performance: Does not lend itself to high-performance systems due to
  - Lack of parallel processing.
  - Closed layering.
  - Sinkhole architecture anti-pattern.

- Fault tolerance:
  - Due to monolithic deployments and the lack of architectural modularity: a small error can impact and crash the entire application unit.

With the characteristics thoroughly discussed, Layered Architecture is the most suitable approach for the HCMUT-SSPS as of right now. The rationales behind are:

- It is the popular choice for early development of new systems. Should new requirements appear, we can move to other architectures with ease.
- Despite certain flaws, it is still relevant in the industry. Future maintenance therefore is easier as there would be more engineers with layered architecture experience.
- The architecture's monolithic nature provide us with extremely low latency across layers in the system.

- The system are not likely to have new features apart from the core logic supporting the printing activity and the management tasks. It would likely not require frequent re-builds and deployments.

While Layered Architecture addresses most of our needs, there are aspects that the architecture under-performs in. Monolithic systems, in general, are very bad at handling faults. To satisfy the availability requirements of maximum 1 hour of downtime per month, we planned to test the system extensively to minimize bugs and failures on our end. It is not ideal, but given the benefits that this architecture will bring to the system, we think this is a good trade-off.

3.1.1.2 Design

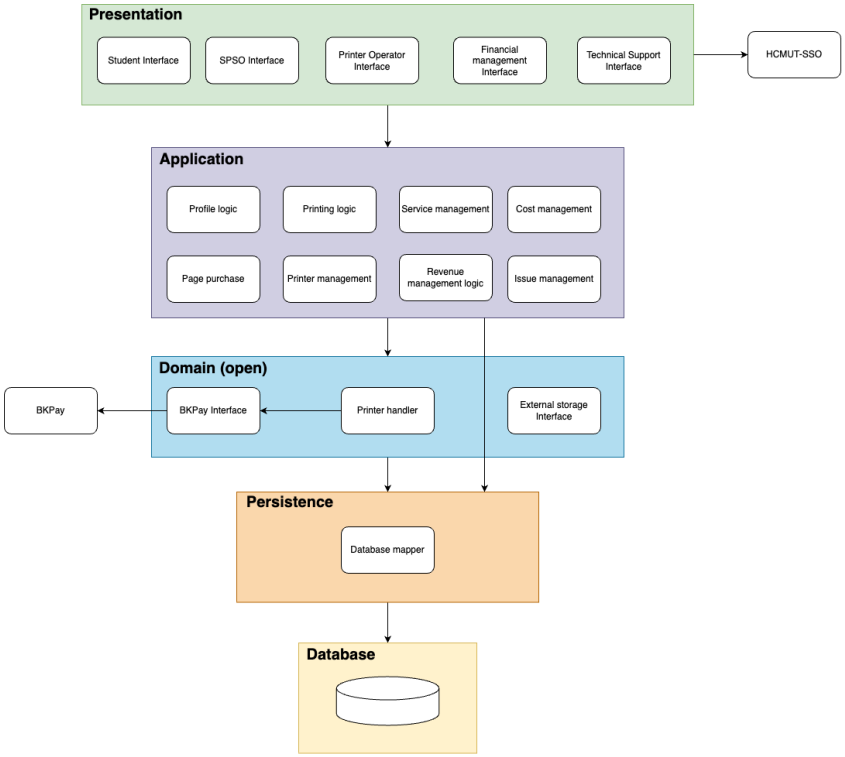


Figure 3.1: Architecture of the whole system of the HCMUT-SSPS

HCMUT-SSPS system consists of 5 layers. Each layers will contain some modules which will help functioning the features of the web. Further description on the above architecture can be emphasized as follow:

1. **Database layer:** Stores data, and a basic class to establish connections between the database and the persistence layer. Therefore, it has no module.
2. **Persistent layer:** Contains **Object Relational Mapping** (ORM) modules, which is also a technique used in creating a "bridge" between object-oriented programs and, in most cases, relational databases. Put another way, we can see the ORM as the layer that connects Object-Oriented Programming (OOP) to relational databases.
3. **Domain layer:** The printer handler module, which interfaces with the physical printers to realize the printing service. The BKPay interface that connects directly to BKPay to manage purchase activities in the system. The external storage, which stores uploaded files and system usage reports. This layer is open because there are components whose access is restricted to only the application layer.
4. **Application layer:** Contains 8 modules accounting for all the business logic in the system:
    - **Profile logic**: manages the page balance of the student.
    - **Printing logic**: handles the logic behind printing activities.
    - **Page purchase**: handles the logic for page purchase.
    - **Printer management logic:** lets the SPSO perform the management of the printers.

- **Service management logic:** lets the SPSO perform the configuration of the service.
- **Revenue management logic:** lets the PFAO see the revenue of the service over time.
- **Cost management logic:** lets the PFAO see and manage the cost of maintaining the service.
- **Issue management logic:** lets the Technical Support manage the user-reported technical issues while using the service.

5. **Presentation layer:** Contains 5 interfaces for each role that uses the system: Students, SPSO, Printer Operator, PFAO, and Technical Support. This layer is authenticated by the HCMUT-SSO system.

3.1.2 Describe how you will present your User Interface.

The SPSS service is available to users as a web application. A web application is accessible from many types of devices, such as on a desktop, laptop, or smartphone. Therefore, the design and implementation of the user interface should accommodate the unique features of each environment. For example, laptops and desktops have wider screens and is usually oriented horizontally, compared to the vertical screens of smartphones. In the demonstration, we only demonstrate the user interface on a desktop screen.

The design of the application mostly follows the same CRM design style and layout. In order to create a consistent user experience across pages, different parts of the user interface is made into components, and are reused wherever possible. This allows each component to be consistent throughout the entire application. Furthermore, all components should follow a common set of rules, called a *design system*. This allows every page to feel and behave similarly to each other.

The user interface consists of three main component: a sidebar, a header and the content.

Figure 3.2: Sidebar of student's flow

Figure 3.2 shows an example of the sidebar. A sidebar consists of clickable items to help users navigate through different functionalities of the service. In order to help users quickly find the item they want, the items on the sidebar are grouped according to which access roles should see them. For example, in figure 3.2, a student's sidebar will have two main groups: one that is only available to students and the other for all types of users (it contains navigation to the user's profile page, reporting an issue and logging out). The currently selected item is highlighted white to help users identify which page they are in. And the sidebar has a dark blue background, which helps separates it with selected items and other components, such as the header and main content.

Figure 3.3: The header of the UI

The header of the user interface can be seen in figure 3.3. This component provides access to some functionalities that should always be present on the page. On the left of the header is a search bar. This allows the user to search and navigate to the functionality they want to use. On the right of the header is the notification area and user profile. The notification area shows the user's notification (this includes information about whether a print request has finished, scheduled downtime/maintenance etc.). The user profile icon shows a quick access menu when clicked, which contains buttons to view the user's profile, or log out of the system.

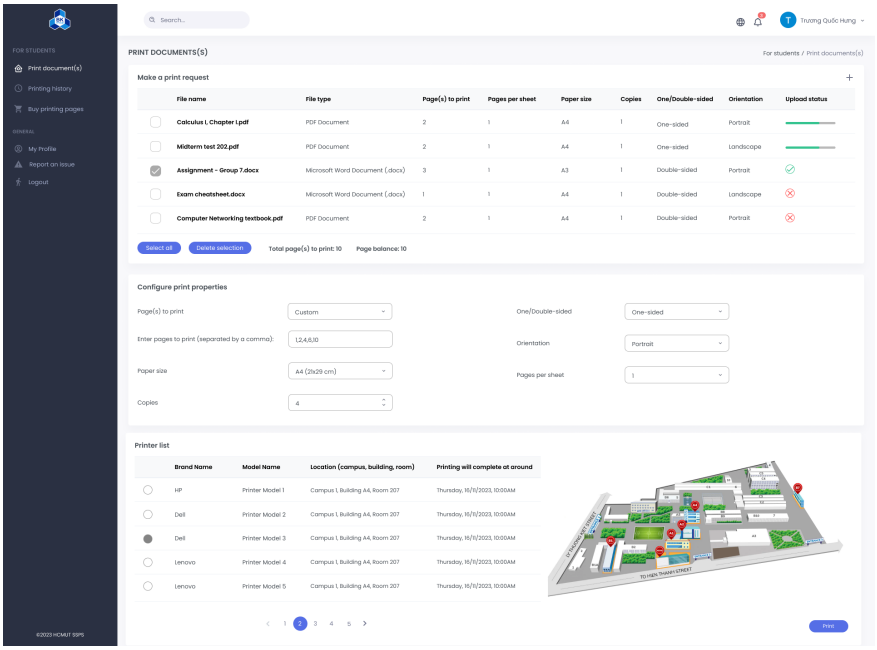


Figure 3.4: Main content of the UI

Below the header is the main content of the page. On the top left and right of every page, the title and path to the page is displayed for easy navigation. The main content is divided into multiple tiles, with each tile responsible for one function, and the layout of tiles is page-dependent.

In figure 3.5, the interface for students to make print requests is divided into 3 main tiles. The first tile is used to show files a student has uploaded to make a print request. It consists of a table of information for each file, and a button to add a new file to the list. The second tile allows the user to configure different properties for each selected files. The third tile allows the user to select a printer for printing, in the form of a table. Each table entry shows the printer's manufacturer, model, location, and estimated wait time if a request is sent to it. Tiles are given a white background, and they are on top of another light gray background, which separates them from the sidebar and header.

3.1.3 Describe how you will store your data

3.1.3.1 Data Model

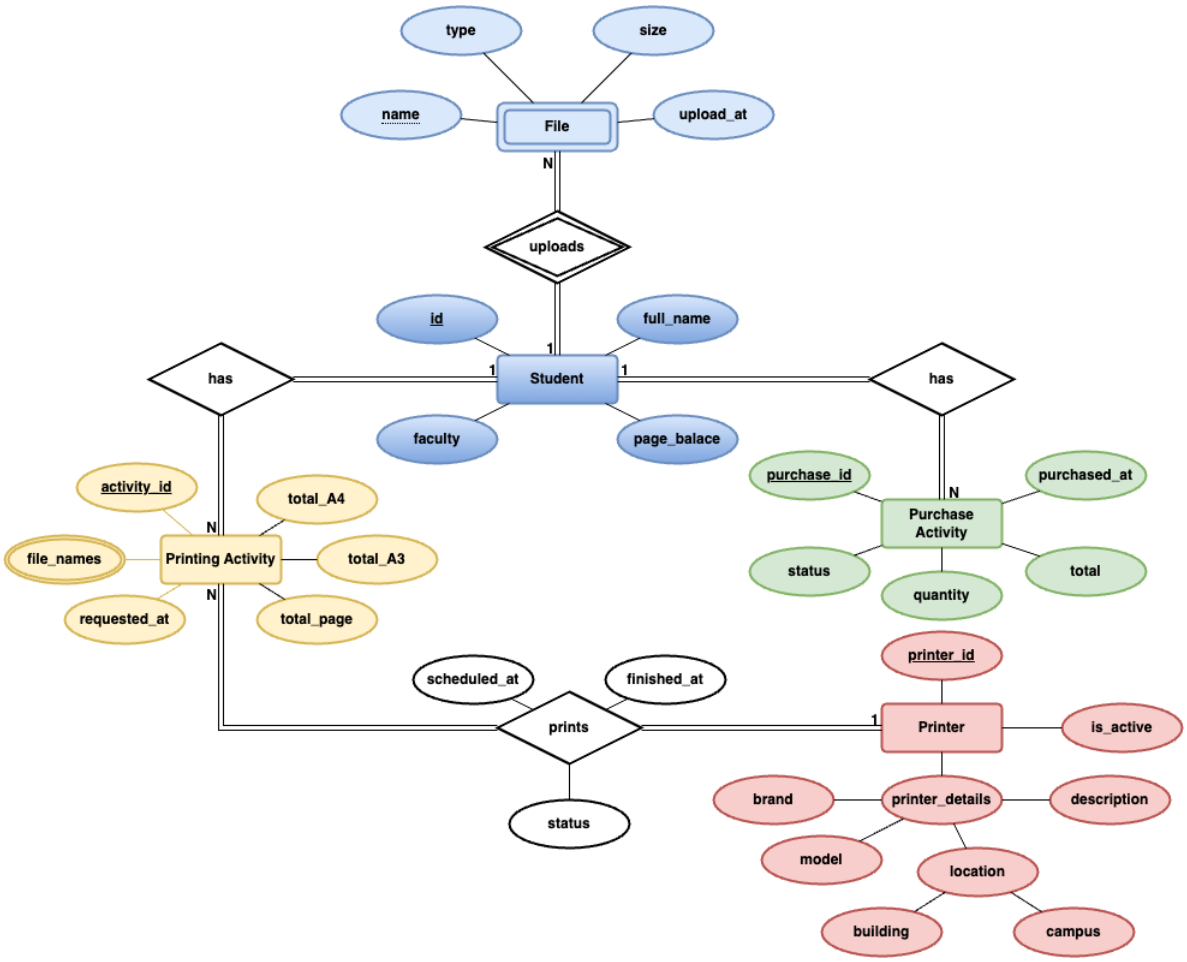


Figure 3.5: The Entity Relationship of the printing service

3.1.3.2 Use case

The system allows students to upload many files at a time. Once they print these files, the system creates an associated printing activity. Each student will have many printing activities. Each printing activity is handled by exactly one printer, and a printer handles many printing activities. The student can view the printing activity as well as the information of the printer that prints it. The student can purchase more A4 pages in order to facilitate this printing process.

3.1.3.3 Access patterns

Access Pattern	Priority	Read or Write	Description	Type (Single/-Multiple/All)	Filters	Result Ordering
Upload files	High	Write	The student uploads files to print	Multiple Items	—	—

Access Pattern	Priority	Read or Write	Description	Type (Single/-Multiple/All)	Filters	Result Ordering
Download files	High	Read	System reads metadata of uploaded files for download requests from external storage	Multiple Items	—	—
Get information of the student	High	Read	Get page balance of a particular student	Single Item	Current student	—
Get student printing activities	High	Read	Get all printing activities of a particular student	All	Student ID, date range	Latest activities first
Get printer's printing activity	High	Read	Get printing activity with status "Complete" of a printer	Single Item	Printer ID, printing status	—
Filter student's printing activities based on date range	Medium	Read	Get printing activities of a particular student based on date range	Multiple Items	Student ID, date range	Latest activities first
Filter printer's pending activities	High	Read	Get printing activities with status "Pending" of a printer	Multiple Items	Printer ID, printing status	Earliest request time first
Filter printer's complete activities	High	Read	Get all printing activities with status "Complete" within the last hour of a printer	Multiple Items	Printer ID, printing status, finish time	Earliest request time first
Get active printers	High	Read	Get all active printers	Multiple Items	Printer status	—
Get inactive printers	Medium	Read	Get all inactive printers	Multiple Items	Printer status	—
Get printers	High	Read	Get all printers	All	—	—
Add a printer	High	Write	SPSO adds a new printer into the system	Single Item	—	—

3.1.3.4 Database design

Figure 3.6: MongoDB

The data model for the SPSS service is implemented at the physical level using MongoDB. MongoDB is an open-source document database built on a horizontal scale-out architecture that uses a flexible schema for storing data. Having flexible schemas allows the team to get started easily and keep development at a fast pace. In terms of scalability, MongoDB is built on a horizontal scale-out architecture, where computing resources is organized into "blocks", and customers can choose to purchase blocks as they need. This not only allows flexible payment for the customer, but it allows the database service to efficiently auto-scale to handle large workloads, which in our opinion, is important for an application that handles the printing of all HCMUT students.

File

Attribute	Data type	Format	Description
name	String	max. 500 Unicode characters	File name
student_id	Int32	7 digits. Example: 2345678	The student who uploads
type	String	*.doc, *.docx, *.rtf, *.xls, *.xlsx, *.ppt, *.pptx, *.pdf, *.txt	The type of the uploaded file
size	Double	A real number	Size in Megabytes
upload_at	Date	Unix timestamp	The time the user upload the file.

Student

Attribute	Data type	Format	Description
_id	Int32	7-digit number. Example: 2345678	The student's identifier
full_name	String	max. 200 Unicode characters	The student's full name
faculty	String	max. 100 Unicode characters	The faculty of the student
page_balance	Int32	Non-negative number: 0 to 4.294.967.296	The amount of A4 pages available to print

Printing Activity

Attribute	Data type	Format	Description
_id	ObjectID	12 bytes. 4-byte unix timestamp, 5-byte random value, 3-byte incremental value.	The identifier of the printing activity
file_names	Array	array of strings	The file names of the printed files
total_A3	Int32	Non-negative number: 0 to 4.294.967.296	Amount of A3 pages to print
total_A4	Int32	Non-negative number: 0 to 4.294.967.296	Amount of A4 pages to print
total_page	Int32	Non-negative number: 0 to 4.294.967.296	Amount of pages subtracted from the page balance
requested_at	Date	Unix timestamp	The time the printing activity is submitted to the printer
printer_id	ObjectID	12 bytes. 4-byte unix timestamp, 5-byte random value, 3-byte incremental value.	The printer responsible for the printing activity
scheduled_at	Date	Unix timestamp	The tentative time the printing job will be complete
finished_at	Date	Unix timestamp	The actual time the printing job will be complete
status	String	pending, printing, complete, or error	The status of the printing activity

Printer

Attribute	Data type	Format	Description
_id	ObjectID	12 bytes. 4-byte unix timestamp, 5-byte random value, 3-byte incremental value.	The identifier of the printer
brand	String	max. 100 Unicode characters	The brand name of the printer
model	String	max. 100 Unicode characters	The printer model
campus	String	max. 100 Unicode characters	The campus in which the printer is located
building	String	max. 100 Unicode characters	The building in which the printer is located
description	String	max. 500 Unicode characters	The description of the printer
is_active	Boolean	true, false	The active status of the printer

Purchase Activity

Attribute	Data type	Format	Description
_id	ObjectID	12 bytes. 4-byte unix timestamp, 5-byte random value, 3-byte incremental value.	The identifier of the purchase
purchased_at	Date	Unix timestamp	The time the purchase is made
status	String	paid, unpaid	The status of the purchase
quantity	Int32	Non-negative number: 0 to 4.294.967.296	The number of A4 pages bought
total	Int32	Non-negative number: 0 to 4.294.967.296	The total amount to pay

3.1.3.5 External Storage

Figure 3.7: Google Cloud Storage

Because the system needs to store students' uploaded files and system usage reports, there is a need for external storage. For this, `Google Cloud Storage` is the choice. It is secure, has good performance, and provides easy integration with the system.

3.1.4 Describe how you will access to external services/ APIs

3.1.4.1 HCMUT SSO authentication service

For authentication, the SPSS service uses the HCMUT_SSO service, which implements the CAS (Central Authentication Service). CAS is a single sign-on (SSO) protocol and open-source software that provides authentication and authorization services for web applications and services. Its primary purpose is to streamline and enhance the authentication process for users accessing multiple web applications within an organization or across different websites.

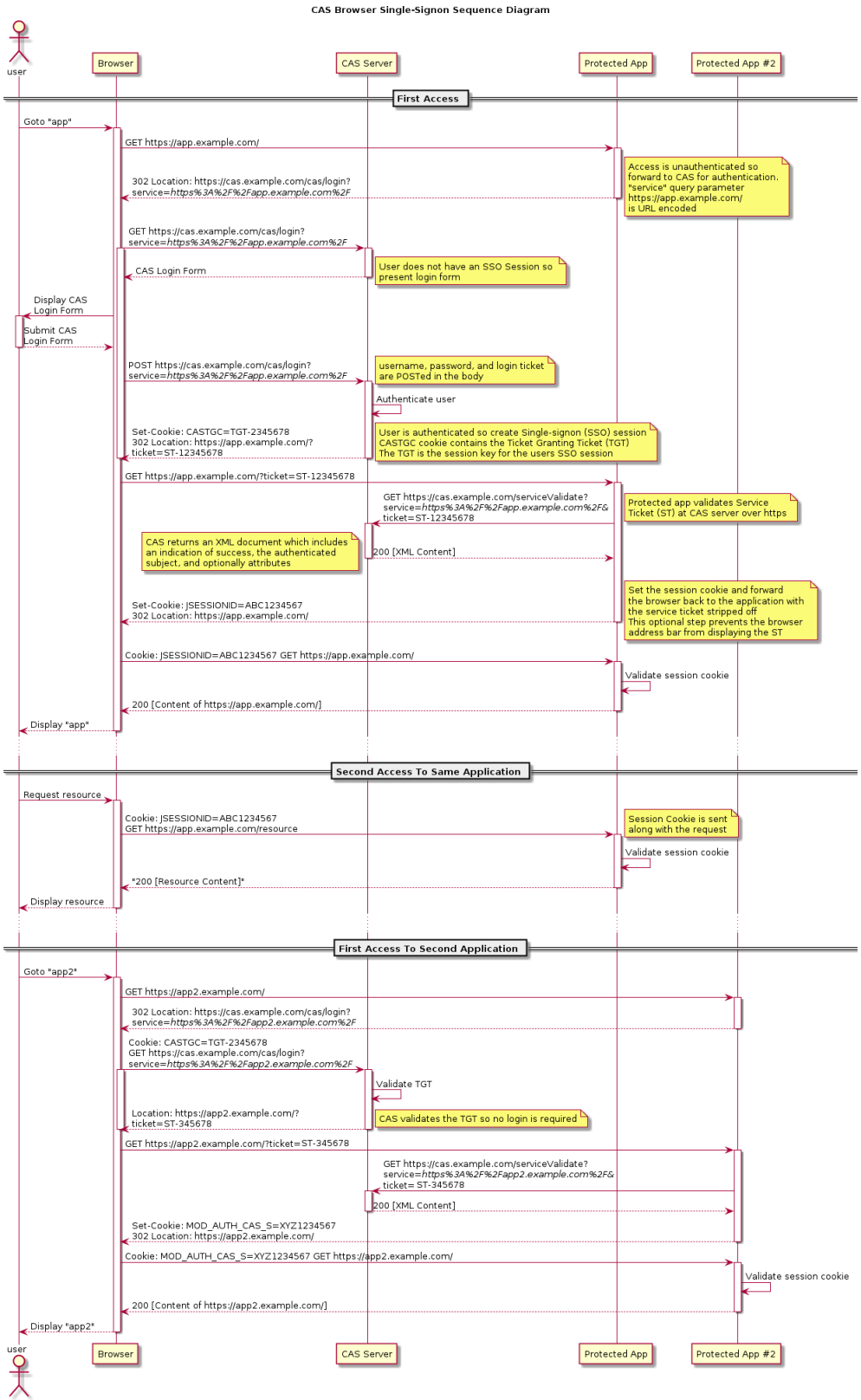


Figure 3.8: CAS work flow

CAS workflow description:

- Unauthenticated users when connected to the application will need to be navigated to GET Sample HCMUT-SSO LOGIN Page
- After login successfully, the CAS Server sends a Single-SignOn tokens to the front-end:

  - **Service Ticket:** used to confirm user identity (placed in the Query String)
  - **TGT (Ticket granting ticket):** used to get the Service Ticket without having to input username/-password. (placed in Cookie header)

- Browser will continue to redirect to the home page of the application, the application then sends the Service Ticket to the CAS Server to validate: GET Sample
- If ticket is validated, the application creates a JWT (Json Web Token) and sets it to the user cookie (JSESSIONID).

Ongoing authentication:

- If the user accesses the same application: the browser will only have to verify the JWT to authenticate without having to send any requests to the CAS Server.
- If the user accesses another CAS-enabled application, the CAS workflow continues.
- If the user accesses to another application that also need CAS:
  - The application redirects the user to CAS Login Page
  - CAS Login Page receive the TGT cookie and generate a Service Ticket for the browser
  - Browser then sends a Service Ticket to that app and that app creates a new JWT and sets it to the browser cookie.
  - That new JWT is used for authentication in the new app

3.1.4.2 Google Cloud Storage

The SPSS internally stores files that the user uploads to print. This copy of the file serves as backup in case of system failure, in which case the system can automatically restart printing job based on the uploaded files. It can also allow the user to choose a file previously uploaded, should they want to print it again.

Our first solution to file storage is to store all files on the backend server itself. However, we identify some problems with this solution:

- The solution is not scalable. The storage capacity of a server is limited, and therefore requires upgrading or attaching new storage devices
- There are no built-in security measures. Storing files on the server is a manual solution, which means that security measures must be manually implemented by database administrators. We prefer a solution where these features are built-in

The problems listed above leads us to choose Google Cloud Storage as the file storage service. Google Cloud storage is scalable, as it allows organizations to expand or reduce their data footprint depending on need. It also removes the need for us to manually implement security measures.

We authenticate between the SSPS service and Google Cloud Storage through the use of **service accounts**, which are a special type of Google account that grant permissions to virtual machines instead of end users, ensuring safe, managed connections to APIs and Google Cloud services. This authentication happens when the backend first loads up.

Our data is uploaded into buckets. Buckets are the basic data container for Google Cloud Service. Everything that is stored within the service must be contained in a bucket. Files inside a bucket are called objects. For this application, we will create one central bucket for storing, and place all our objects there.

We list the main APIs that will be used in the application:

- `POST /b`: Create a new bucket to place objects
- `POST /b/BUCKET/o`: Insert an object into the specified bucket
- `GET /b/BUCKET/o/OBJECT`: Retrieve the specified object from the specified bucket
- `DELETE /b/BUCKET/o/OBJECT`: Delete the specified object from the specified bucket

It is important to note that these APIs are called by the backend to manipulate files on the bucket. It is not secure to allow the frontend to work with these APIs, as each user will also be able to modify files not created by themselves. The backend server will perform the upload, fetching and delete operations and return the result back to the frontend.

3.1.4.3 Google Drive file selection

The SPSS service allows the user to pick one or many files for printing. The system supports picking files from two sources: either on their local machine, or through accessing their Google Drive storage. For the latter, the Google Picker API is used to show the file picker. Figure 3.9 shows the file picker, powered by the Google Picker API.

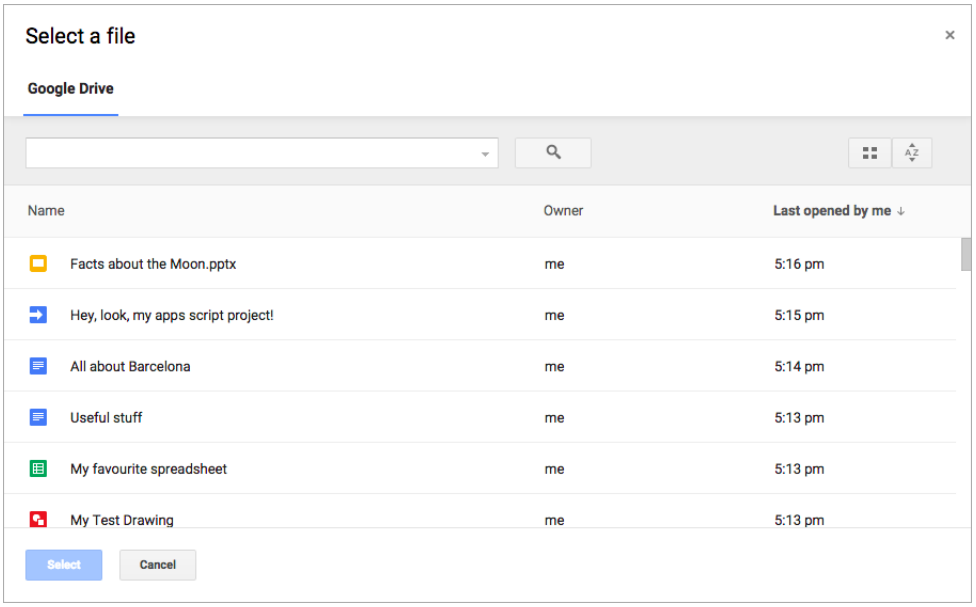


Figure 3.9: Google Drive file picker

In order to use the Google Picker API in our application, we first need to enable it. The API can be turned on in the "Enabled APIs" section of the Google Cloud Console settings for the project.

After enabling the service, we need to create an API key to use the Picker API in our application. This API key is used in conjunction with an **OAuth 2.0 Client** to authenticate a user's account. When the web application first loads up, we load the Picker API using the obtained API key. Finally, whenever the user wants to choose a file from their Google Drive storage, we simply display the selection dialog using the Picker API

3.1.4.4 Integration with BKPay

In order to print documents, the user must have enough page balance. Otherwise, they can purchase more pages using the "Purchase page balance" feature, which uses the BKPay service API. We will use the BKPay service to perform the following actions:

- Each time a student makes a print request, the system creates a payment that needs to addressed by the student. The SPSS payment system implements a "Buy Now, Pay Later" (BNPL) model, where students can purchase page balance and pay at a future date. Each payment comes with a due date for paying. The Planning & Financial Affairs Office (PFAO) is responsible for handling students that do not pay their purchases on time.
- The PFAO can retrieve a list of overdue payments, along with information of the student that made them

3.1.4.5 Integration with printers

SPSS is a printer that manages the printing activities of HCMUT students. Therefore, it is important that the system can work with printers around the campus seamlessly. The backend server will manage connections to each of the enabled printer, and use those connections to control printers. Particularly, the server will perform the following actions to printers:

- The server will forward print jobs to the corresponding printers
- The server can enable, disable each individual printer
- The server will occasionally ping each printer, in order to check if the printer is still connected and functioning properly

The most important detail when implementing the printer management module is to add support for many types of printers. For example, the school may choose to buy printers from a wide range of manufacturers (e.g HP, Dell, Lenovo), and of different models. This creates a challenge for the implementation phase, as the backend would have to know how to communicate with each printer bought by the university.

3.2 Draw a component diagram for the important module(s) chosen in Task 1.3.

Component diagram

Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.
A component represents a modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment. In UML 2, a component is drawn as a rectangle with optional compartments stacked vertically. A high-level, abstracted view of a component in UML 2 can be modeled as:

1. A rectangle with the component's name
2. A rectangle with the component icon
3. A rectangle with the stereotype text and/or icon

Two type of component interfaces include:

- **Provided interface** symbols with a complete circle at their end represent an interface that the component provides - this "lollipop" symbol is shorthand for a realization relationship of an interface classifier.
- **Required interface** symbols with only a half circle at their end (a.k.a. sockets) represent an interface that the component requires (in both cases, the interface's name is placed near the interface symbol itself).

The subsystem classifier is a specialized version of a component classifier. Because of this, the subsystem notation element inherits all the same rules as the component notation element. The only difference is that a subsystem notation element has the keyword of subsystem instead of component.
Ports are represented using a square along the edge of the system or a component. A port is often used to help expose required and provided interfaces of a component.
Graphically, a component diagram is a collection of vertices and arcs and commonly contain components, interfaces and dependency, aggregation, constraint, generalization, association, and realization **relationships**. It may also contain notes and constraints.

3.2.1 Diagram

Follow this link to view more: https://drive.google.com/file/d/1733sF0rREPmfFcZ9Uvl75bkDeD3H35iJ/view?usp=drive_link

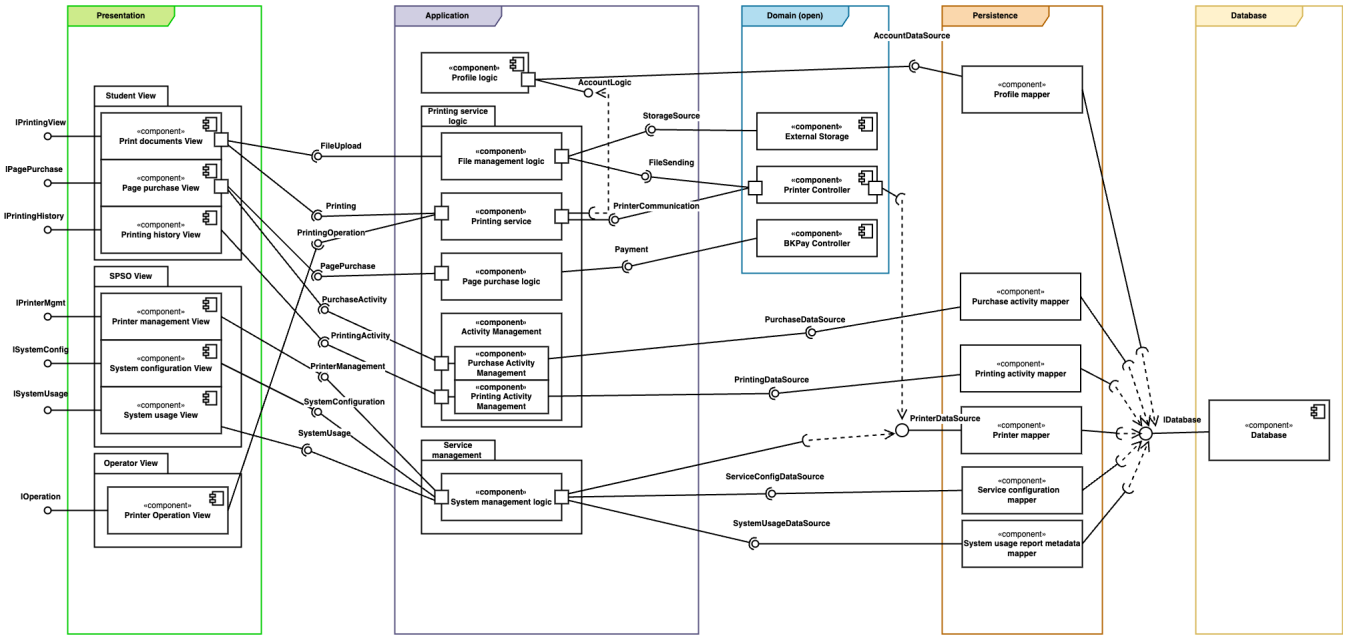


Figure 3.10: Component diagram of HCMUT-SSPS's print service

3.2.2 Description

Component	Description	Provides	Requires	Layer
Print documents View	Provides the view where the students carry out their print requests	• `IPrintingView`	• `FileUpload` • `Printing`	Presentation
Page purchase View	Provides the view where the students carry out their page purchase request	• `IPagePurchase`	• `PagePurchase` • `PurchaseActivity`	Presentation
Printing history View	Provides the view there the students view their printing history	• `IPrintingHistory`	• `PrintingActivity`	Presentation
Printer management View	Provides the view where the SPSO can manage the printers in the system	• `IPrinterMgmt`	• `PrinterManagement`	Presentation
System configuration View	Provides the view where the SPSO can configure the service settings	• `ISystemConfig`	• `SystemConfiguration`	Presentation
System usage View	Provides the view where the SPSO can monitor the system usage	• `ISystemUsage`	• `SystemUsage`	Presentation

Printer operation View	Provides the view where the Printer operator can coordinate the printing activities for each printer	• `IOperation`	• `PrintingOperation`	Application
Profile logic	Provides the page balance data from the account of the student.	• `AccountLogic`	• `AccountDataSource`	Application
File management logic	Provides the logic for managing files in the external storage	• `FileUpload` • `FileSending`	• `StorageSource`	Application
Printing service	Handling printing activities	• `Printing` • `PrintingOperation`	• `PrinterCommunication`	Application
Page purchase logic	Handling purchase activities	• `PagePurchase`	• `Payment`	Application
Purchase Activity Management	Get purchase activities data	• `PurchaseActivity`	• `PurchaseDataSource`	Application
Printing Activity Management	Get printing activities data	• `PrintingActivity`	• `PrintingDataSource`	Application
System management logic	Provides the logic for system management	• `PrinterManagement` • `SystemConfiguration` • `SystemUsage`	• `PrinterDataSource` • `ServiceConfigDataSource` • `SystemUsageDataSource`	Application
External Storage	The interface with the external storage system outside	• `StorageSource`		Domain
Printer Controller	The interface with the physical printers	• `PrinterCommunication`	• `FileSending` • `PrinterDataSource`	Domain
BKPay Controller	The interface with the BKPay service	• `Payment`		Domain
Profile mapper	The ORM module that interfaces with the database to obtain the data of the student's profile.	• `AccountDataSource`	• `IDatabase`	Persistence

Purchase activity mapper	The ORM module that interfaces with the database to obtain the data of the student's profile.	• `PurchaseDataSource`	• `IDatabase`	Persistence
Printing activity mapper	The ORM module that interfaces with the database to obtain the data of the students' profile.	• `PrintingDataSource`	• `IDatabase`	Persistence
Printer mapper	The ORM module that interfaces with the database to obtain the data of the printers.	• `PrinterDataSource`	• `IDatabase`	Persistence
Service configuration mapper	The ORM module that interfaces with the database to obtain the service configuration data.	• `ServiceConfigSource`	• `IDatabase`	Persistence
System usage report metadata mapper	The ORM module that interfaces with the database to obtain the metadata of the system usage reports that have been generated.	• `SystemUsageSource`	• `IDatabase`	Persistence
Database	The main database component of the system	• `IDatabase`		Database

Interface	Description	Methods
IPrintingView	Main interface with the Print documents View	+ onUploadFromGoogleDrive(): void + onOpenFileSelectionDialog(): void + onSelectFile(): void + displayFiles(): void + render(language: Language): void + onOpenModal(): void + onCheckBoxToSelect(): void + onSelectAll(): void + onEditingPrintingProperties(): void + onSelectCampus(): void + onClickPrint(): void + onConfirm(): void + calculatePages(): int + getQueue(printer: Printer): void + displayPageCalculation(): void + displayConfirmationDialog(): void + displaySuccessMessage(): void + displayException(exception: Exception): void
IPagePurchase	Main interface with the Buy printing pages view	+ render(language: Language): void + onEnterPageNumber(): void + onPurchase(): void + displayConfirmationDialog(): void + onConfirmPurchase(): void + displayPurchaseActivities(): void

IPrinterMgmt	Main interface with the Printer management View	+ render(language: Language): void + displayPrinters(): void + onAddNewPrinter(): void + openDialog(): void + enterPrinterDetails(): void + onSubmit(): void + displaySuccessMessage(): void
ISystemConfig	Main interface with the Service configuration View	+ render(language: Language): void + onEditSystemConfig(): void + onEditSystemConfig (): void + validatelnput(): void + onSave(): void
ISystemUsage	Main interface with the System usage View	+ render(language: Language): void + renderChart(): void + onViewReport(): void + openPDFViewer(): void
IOperation	Main interface with the Printer operation View	+ render(language: Language): void + displayPrinters(): void + displayPendingActivities(): void + displayPrintingActivity(): void + openStartPrintingDialog(): void + OnSave(): void
AccountLogic	Accessing and operating on the student's page balance	+ getPageBalance(): int + setPageBalance(int): void
FileUpload	File upload to the external storage	+ uploadFiles(): void + updateStatus(): void

Printing	Providing the main logic of the printing service	+ fetchPrinterData(): void + checkPageBalance(): boolean + checkSlot(): boolean + handleRequest(): void + submit(PrintingActivity): void
PrintingOperation	For the printer operator to coordinate the printing activities	+ getPrintingActivities(): PrintingActivity [*] + updatePrintingActivity(PrintingActivity): void
PagePurchase	Logging a purchase activity	+ create(PurchaseActivity): void
PurchaseActivity	Accessing purchase activities	+ getPurchaseActivities(): PurchaseActivities(): PurchaseActivities [*]
PrintingActivity	Accessing printing activities	+ getPrintingActivities(): PrintingActivities [*]
PrinterManagement	Managing the printers in the system	+ addPrinter(Printer): void + changeStatus(Printer): void + getAllPrinters(): void
SystemConfiguration	Configuration operations of the service	+ getConfiguration(): SystemConfiguration + updateConfiguration(): void
SystemUsage	Monitoring the system usage	+ initialize(): void + viewReport(report: SystemReport): void
StorageSource	Interacting with the external storage	+ uploadFiles(): void + downloadFiles(): void + deleteFile(FileMetadata): void

FileSending	The interface for sending file from the external storage to the printers	+ downloadFile(): void
PrinterCommunication	Communicating with the physical printers	+ getAllCompletion Time () : Date Time [*] + getCompletion Time(printer: Printer): DateTime + getSlot(printer: Printer): int + placeRequestInPrinter(printer: Printer, printingActivity: PrintingActivity): void + sendFiles ToPrinter : void
Payment	Integrating with BKPay to manage the purchase process	+ createPaymentEntry(PurchaseActivity): void
PurchaseDataSource	Provides operations on purchase activities data	+ create(dataEntry: PurchaseActivity) + filterBy(studentID: int): PurchaseActivity [*] + update(dataEntry: PurchaseActivity) + delete (dataEntry: PurchaseActivity)
PrintingDataSource	Provides operations on printing activities data	+ create(dataEntry: PrintingActivity) + count(): int + filterBy(studentID: int): PrintingActivity [*] + filterBy(printerID: int, status: PrintingStatus): PrintingActivity [*] + update(dataEntry: PrintingActivity) + delete(dataEntry: PrintingActivity)

PrinterDataSource	Provides operations on printer data	+ create(dataEntry: Printer) + get(dataEntry: String): Printer + getAlI(): Printer [] + filterBy(status: boolean): Printer [*] + update (dataEntry: Printer) + delete(dataEntry: Printer)
ServiceConfigDataSource	Provides operations on service configuration	+ get(): SystemConfiguration + update(SystemConfiguration)
SystemUsageDataSource	Provides operations on system usage data	+ create(SystemUsage) + getAll(): SystemUsage [*]
Database	Interface with the Database in the Database layer	+ commitToDB(): void + execQuery(Query): void

Chapter 4

Task 4: Implementation – Sprint 1

4.1 Setting up an online repository (github, bitbucket, etc) for version control.

Check for our official repository here: https://github.com/bluishfluid666/HCMUT-SSPS_GROUP12/

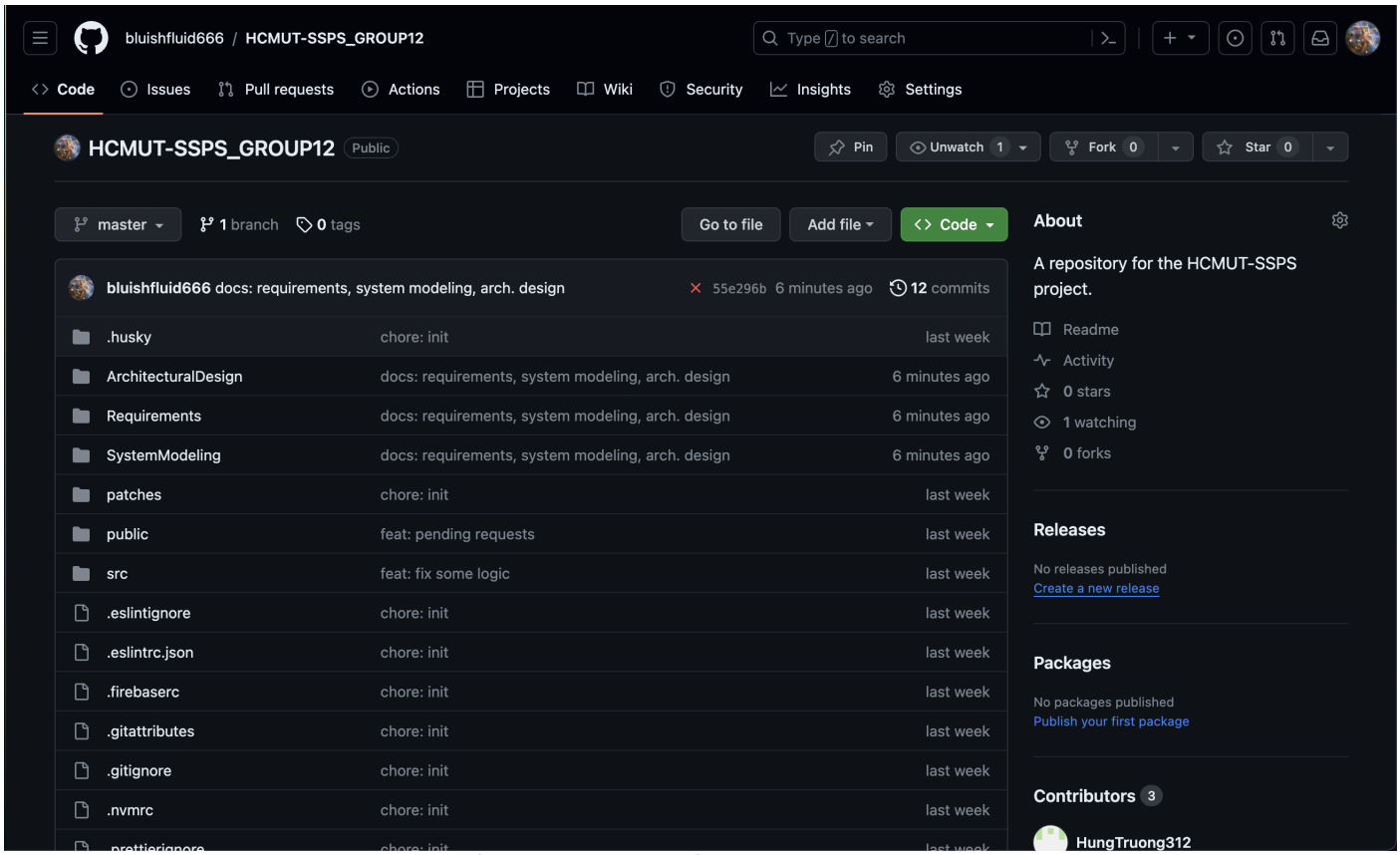


Figure 4.1: Initialize repository

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

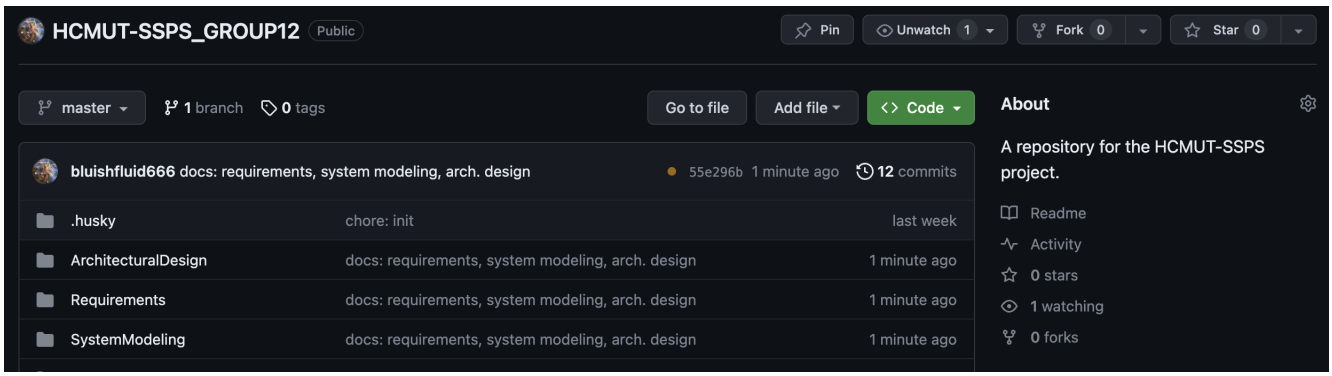


Figure 4.2: Adding documents into the repository

1. **Architectural Design:** Containing architectural diagrams of the HCMUT-SSPS, namely the overall architecture and a UML component diagram for the Print Service:

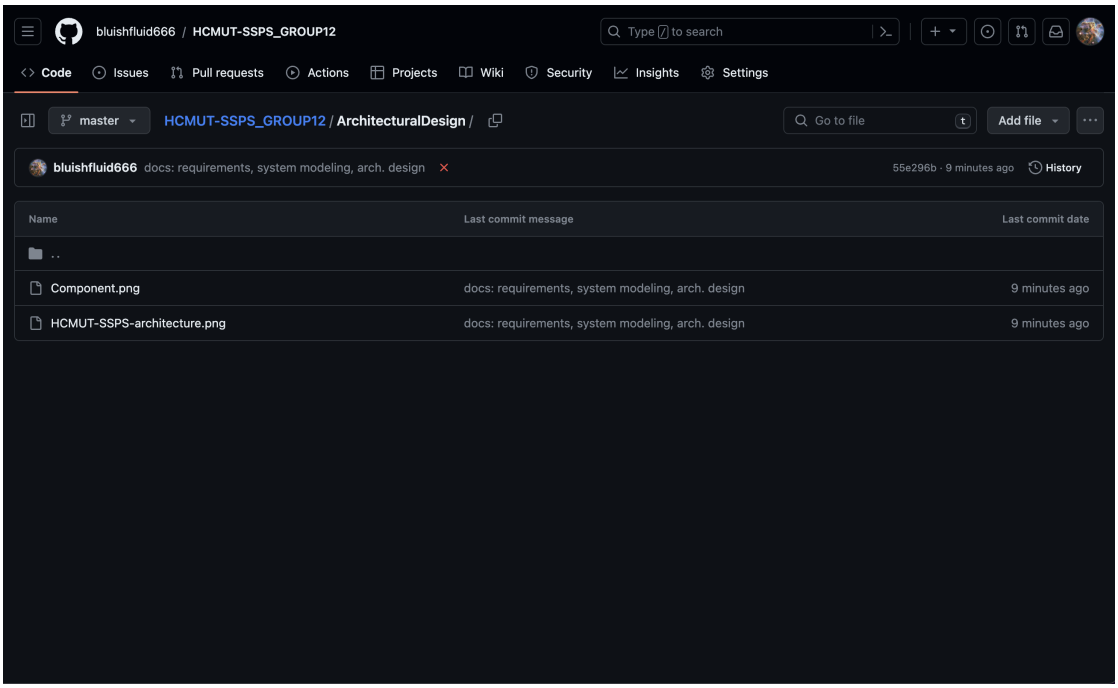


Figure 4.3: Architectural Design folder

2. **Requirements:** Containing Functional and Non Functional requirements

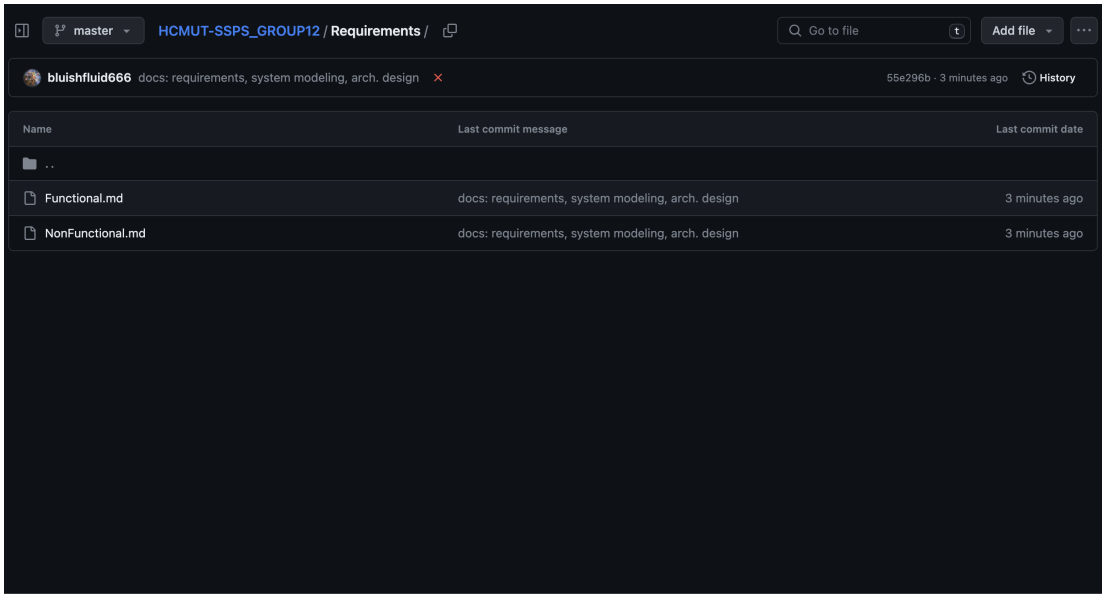


Figure 4.4: Requirements folder

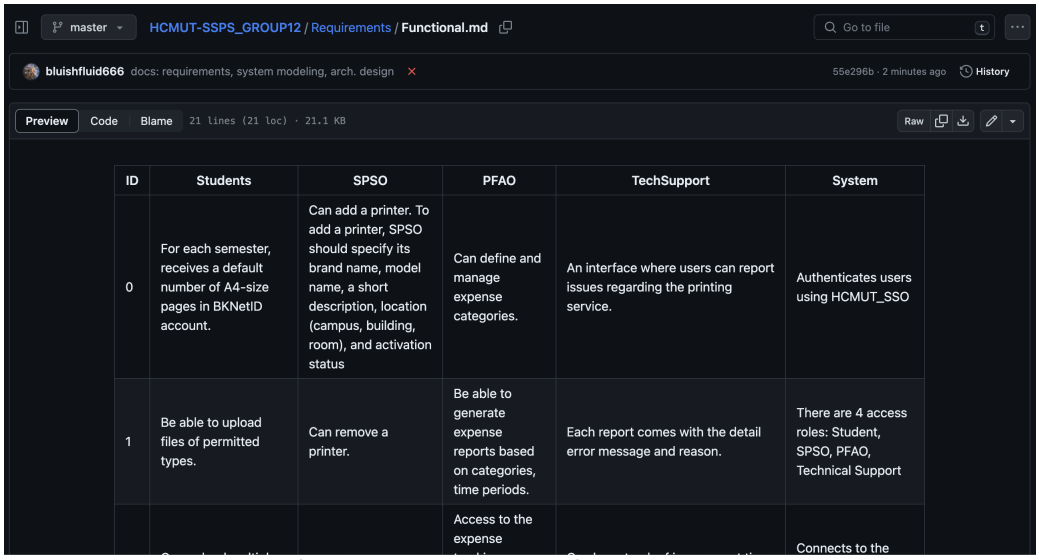
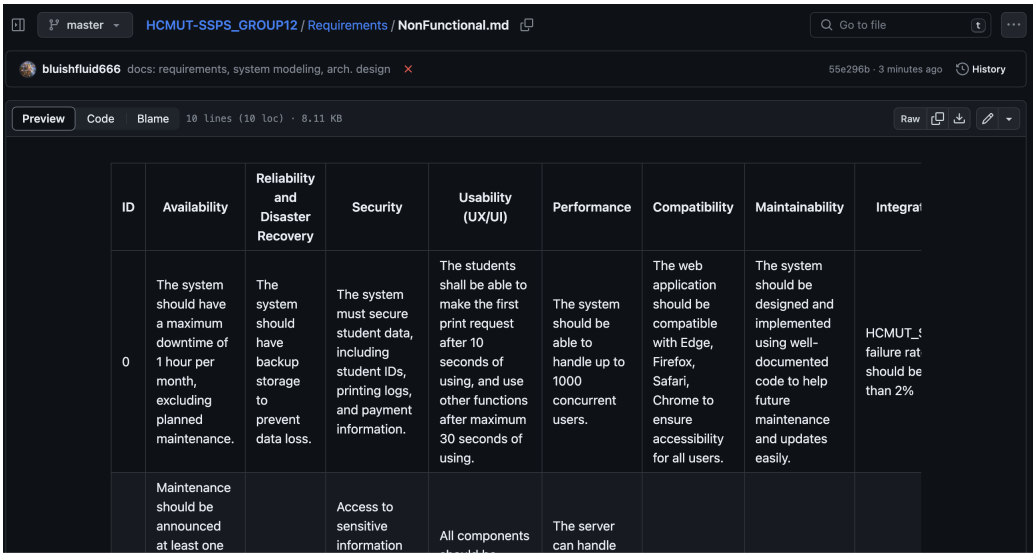


Figure 4.5: Functional Requirements



ID	Availability	Reliability and Disaster Recovery	Security	Usability (UX/UI)	Performance	Compatibility	Maintainability	Integrat
0	The system should have a maximum downtime of 1 hour per month, excluding planned maintenance.	The system should have backup storage to prevent data loss.	The system must secure student data, including student IDs, printing logs, and payment information.	The students shall be able to make the first print request after 10 seconds of using, and use other functions after maximum 30 seconds of using.	The system should be able to handle up to 1000 concurrent users.	The web application should be compatible with Edge, Firefox, Safari, Chrome to ensure accessibility for all users.	The system should be designed and implemented using well-documented code to help future maintenance and updates easily.	HCMUT_S failure rat should be than 2%
	Maintenance should be announced at least one		Access to sensitive information	All components	The server can handle			

Figure 4.6: Non-Functional Requirements

3. **System Modeling:** Activity Diagrams, Sequence Diagrams, and Class Diagram

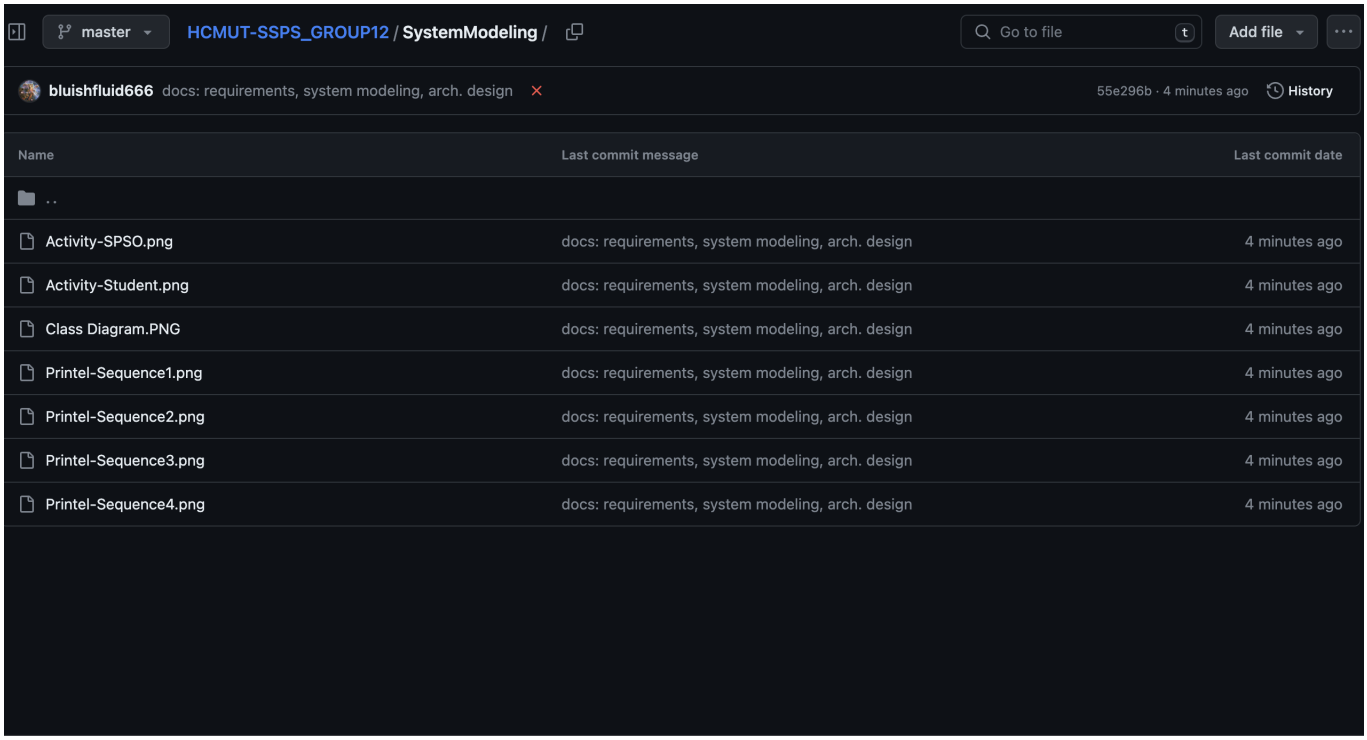


Figure 4.7: System Modeling folder

4.3 Conduct a usability test with the user interface you developed in MVP1.

Tester report: https://drive.google.com/file/d/1ahRxu8s7heblqbT6mrU-R4LnQGRiqZrC/view?usp=sharing

We also conducted a survey on our system, garnering insights from 85 responses provided by students within our university.

1. **Interfaces:** Evaluations for our Profile, Upload, Printing History, and SPSO Printer Management interfaces.

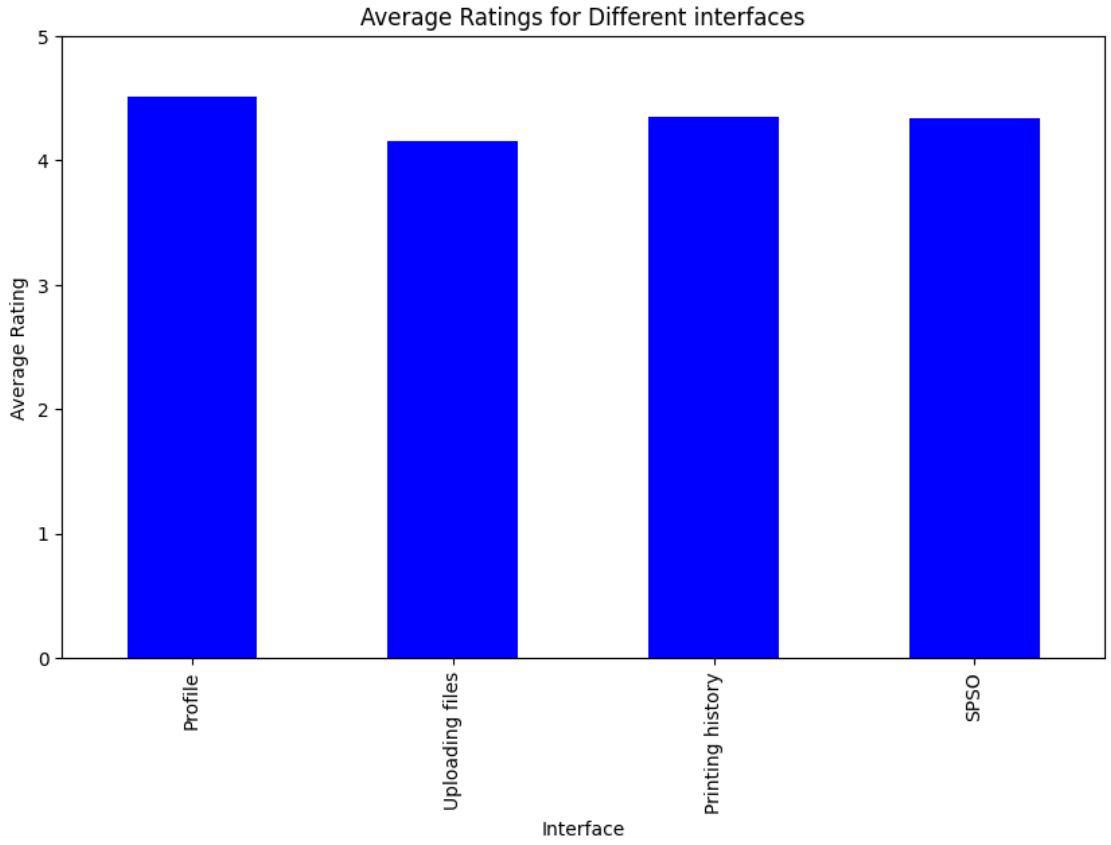


Figure 4.8: Average ratings of our interfaces

2. **Improvements:** We have received valuable feedback suggesting potential improvements to enhance the performance of our system.

	Upload improvements
8	Is it the + button for upload
21	hard to find button
29	should have review
39	Upload button too small
46	need a review of each file after print
47	Choose printer by click on the map
62	Print button very small
83	page balance needs to be bigger

	SPSO improvements
28	filter printer
35	prevent remove an enabled printer
49	It should have sorting function to find printer
57	add button is small

	Profile improvements
51	too small number
59	Bigger page balance number
71	number is small

	History improvements
9	should have sort for values
18	should have filter for finding file
43	sort is better
65	Show more files in a request
75	Filters
81	filter of many column
84	filters for status and pages count

Figure 4.9: Comments about improvements

3. **Unclear:** Some users have expressed difficulty in understanding the information and actions within our system. Analyzing this feedback can guide us in making the system more user-friendly.

	Upload Unclear
52	I don't know how to configure just one or some...

	History Unclear
0	I think printer column is a little bit long
13	What is (est.)

	SPSO Unclear
5	should have some action button to resolve thin...
66	can broken printer enabled accidently by officer?

Figure 4.10: Comments about unclear information

4. **Overall:** Overall rating of different criteria about our system.

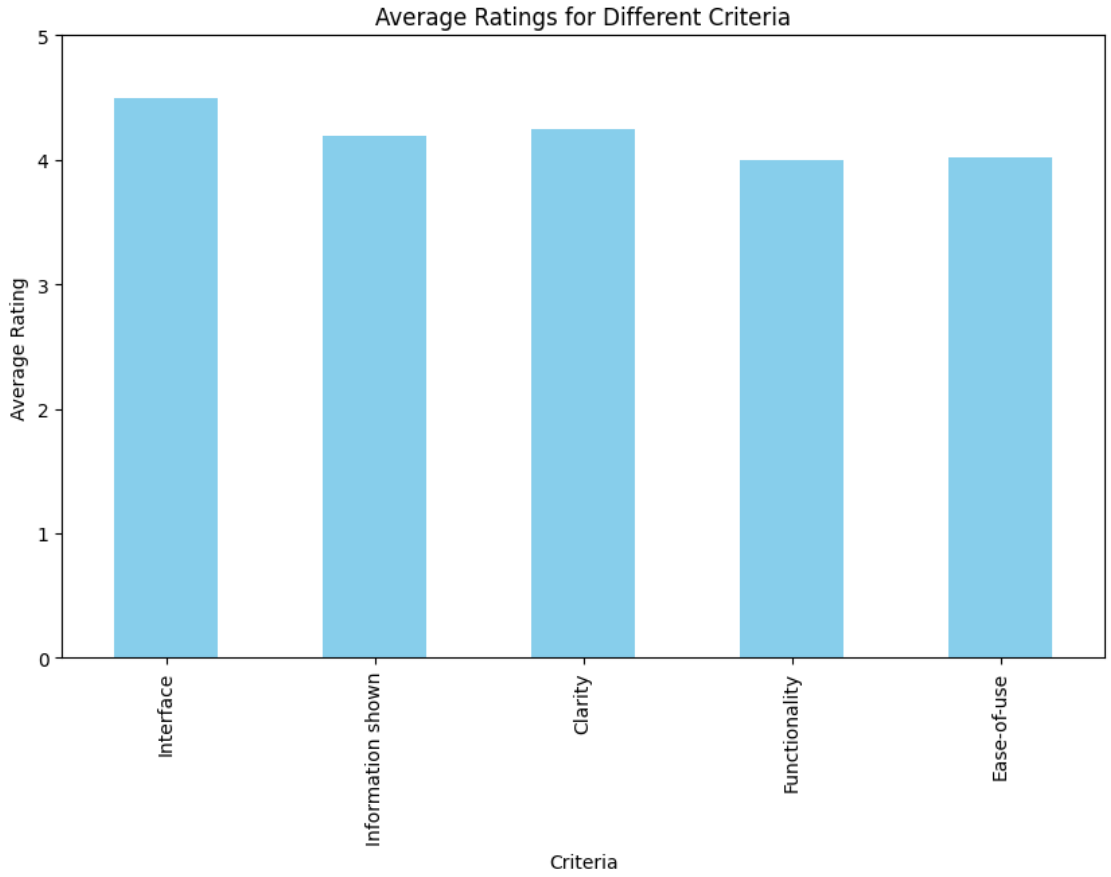


Figure 4.11: Average ratings of different criteria

The comprehensive survey findings highlight a consistently positive rating, with an average rating exceeding 4/5 across all interfaces and criteria. This favorable response underscores the overall satisfaction of users with our system's performance. However, the depth of feedback received, especially focusing on specific aspects such as the perceived diminutiveness of the 'Add' button and user experience with the printing history filters, provides insightful areas for improvement.

The meticulous analysis of user comments unveils valuable insights into the nuanced preferences and challenges encountered by our user base. The observation that some users find the 'Add' button relatively small and encounter difficulty with the printing history filters serves as a prime example of actionable feedback. These insights not only offer a qualitative understanding of user experiences but also serve as a data-driven foundation for strategic system enhancements.

Moving forward, these analytical findings will be pivotal in shaping our system's user interface and functionality. By addressing specific pain points identified in the feedback, we aim to cultivate a more intuitive and user-friendly environment that resonates with a broader audience. This iterative approach to system development ensures that our enhancements align precisely with user expectations, fostering a positive and seamless experience for all users.

Chapter 5

Task 5: Implementation – Sprint 2

5.1 Develop MVP 2 with input from Task 2.4 and Task 4.3. You are free to choose the programming language (HTML, Javascript, Python, C#, etc). It is not required to implement a database in the backend. Data can be hard coded in code files.

5.1.1 Tech stack choice

In this section, we describe the choice of technology (programming language, execution environment, libraries, frameworks) during the implementation phase of the system. We will also give rationales for these choices.

Front-end

- **React**
  - We found that React is the most popular way to make user interfaces, is easy to learn, and has a large community. This makes it easy to find documentation and get started.
  - In addition, it is easy to find libraries and tools to help develop our project (e.g., Bootstrap, Ant Design, Redux). This allows us to keep development at a fast pace and add new features quickly.
  - React allows us to write HTML-like syntax inside JavaScript code to define what the UI should look like, and it allows for separating the UI into reusable components.
  - React implements the virtual DOM, which keeps a representation of the UI in memory, and syncs this UI with the actual DOM. When syncing, React uses an algorithm to only update part of the DOM that changes. This allows our product to have good performance when faced with increased complexity.

Back-end

- **Node.js**
  - Node.js uses JavaScript, which is easy to get started, has a large community, and allows us to use the same language for writing the front-end.
  - Node.js is a single-threaded runtime environment that supports asynchronous APIs. This eliminates the need for handling various concurrency problems (e.g., race conditions, deadlocks).
  - Easy to find libraries and tools to develop the product.
  - The simplicity of Node.js event loop and its ability to handle a large amount of concurrent connections makes it easy to handle the growing complexity of the service.
- **Express.js**
  - Express.js provides a minimal layer of essential web application features, which makes creating APIs quick and easy, while taking advantage of Node.js for performance and scalability.

- Doesn't enforce a particular structure of the application. Therefore, we get the flexibility of choosing tools which best suit the problem at hand.
- Middleware system allows us to add new features, without changing existing code (e.g., middlewares for authentication, serving static files, parsing request payload).

5.1.2 Demonstration product

We have developed MVP 2 of the SPSS service and deployed it on the Internet. The demonstration product is deployed using Firebase Hosting. The demonstration website is hosted at https://spss-group-12.web.app/

5.2 Demonstrate the whole project from Task 1 to Task 5

Here are the demonstration slides we have prepared. Slide link: https://www.canva.com/design/DAF2A4rY5jI/xOBPGd29t6xYgX-pMepRjA/edit?utm_content=DAF2A4rY5jI&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Chapter 6

Final Thoughts

6.0.1 Risks and Challenges

6.0.1.1 Risks

- Even though the time is generous, there are many other school activities that overlap. Hence, we might have run the risk of not finishing up the MVPs in time.
- Our design suffers from fundamental flaws.
- Our software might not be easy to use. As software are built for users, ease of use should be the top priority.
- The software for the printing service didn't live up to its name "Smart Printing Service".

6.0.1.2 Challenges

- **Teamwork**: The schedules of members do not always agree with one another.
- **Analysis of domain context**: This is very important because to design an effective software solution, understand the context within which the software it exists, and understand the value gained by implementing the software are crucial.
- **UML Modeling**: The UML is new and sometimes cumbersome for us to effectively model the system using it.
- **Focus on the design that benefits the users the most**: This is very vague so we need a lot of input to realize it.

6.0.1.3 Resolution

- Getting as much feedback from Teaching Assistant (TA) as possible. We would be working on his suggestions to improve the current work.
- Everybody assumes a Scrum role: A product owner, A Scrum master, Developers, Testers. At least one weekly meeting for team members to be informed of others' work and catch up with newer work.
- In order to analyze the domain more appropriately and comprehensively, each team member reads chapter 1 of the book "Learning Domain-Driven Design" by Vlad Khononov, and also get Teaching Assistant's feedback on the analysis.
- We found that the book "Learning UML 2.0" by Russ Miles and Kim Hamilton helps us a ton getting onboard with UML modeling. The book is easy to read and applicable.

6.0.2 Lessons learned

6.0.2.1 Technical:

- We learn what real software engineering is about: Requirement engineering, System modeling, Architectural design, Design and Implementation, Software Testing, and System Evolution. In software engineering, the coding part is not the most important one but is heavily influenced by the design processes.

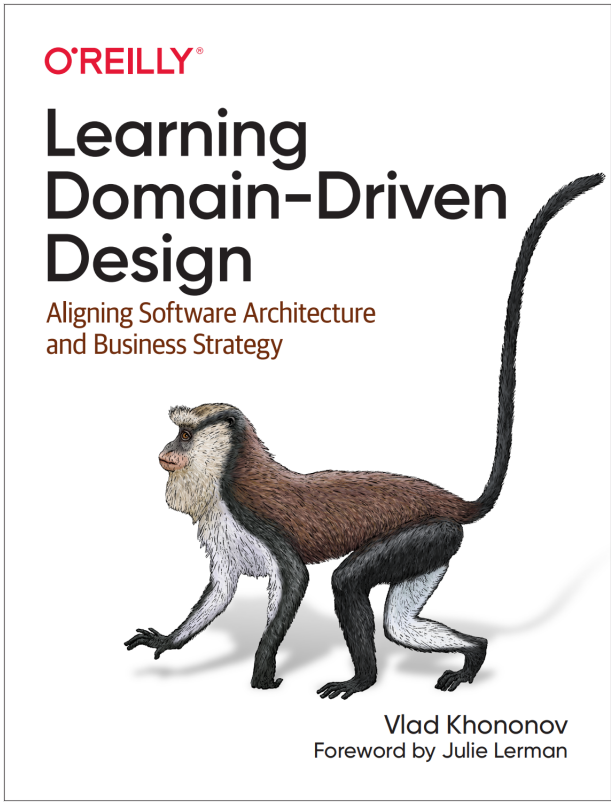


Figure 6.1: Learning Domain-Driven Design by Vlad Khononov

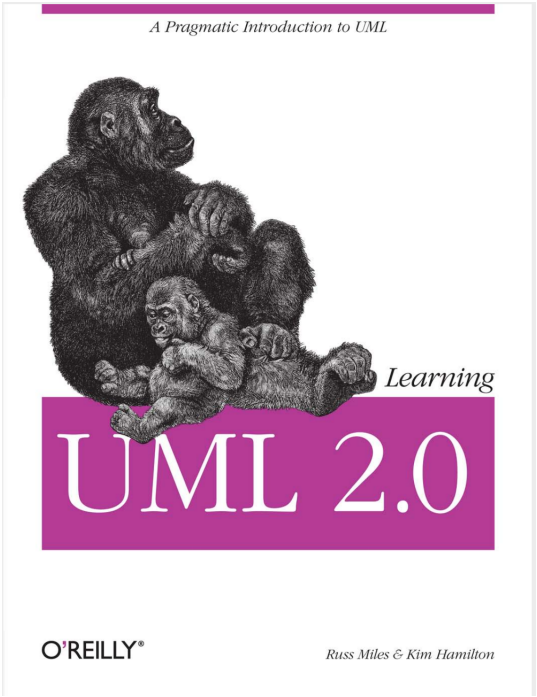


Figure 6.2: Learning UML 2.0 by Russ Miles and Kim Hamilton

- During our system modeling process with UML, we learn that the diagrams complement each other. For example, when we revise the sequence diagram, we realize in class diagrams there are some methods and classes that needed to be added. So, we should be drawing those diagram concurrently to more effectively model the system.
- We get to learn about the techniques as well as the structures of a web application.

6.0.2.2 Non-Technical:

- Not try to be perfectionists. Applying the Agile mindset.
- The people factor is the most important. The technicality isn't itself complicated.
- Problems during teamwork should be raised as soon as possible. This is because these problems may cause changes in previous work. For example, difficulties in implementations may cause the specification to change to fit the time constraint.