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

SOFTWARE ENGINEERING

Report

A Smart Printing Service For Students At HCMUT

Advisor:	Trương Thị Thái Minh		
Students:	Phạm Thành Nam	-	2153603
	Trần Nhật Tân	-	2112259
	Trịnh Hoàng Duy	-	2152473
	Đinh Lê Nhật Anh	-	2052370
	Nguyen Anh Khang	-	2053097
	Phạm Thiên Đăng	-	1952653

HO CHI MINH CITY, NOVEMBER 2023

Contents

1 Requirement elicitation

1.1 General Descriptions

HCMUT Student Smart Printing Service (HCMUT_SSPS)

The Ho Chi Minh City University of Technology (HCMUT), being at the forefront of technological and educational advancements, understands the pivotal role that accessible and efficient resources play in enhancing the academic experience of students. This understanding has led to the conceptualization of the Student Smart Printing Service (HCMUT_SSPS), a novel initiative aimed at revolutionizing how students utilize printing services across the expansive university campuses.

Traditional printing services often involve cumbersome processes. Students typically have to transfer files onto a USB drive, physically go to a print station, and manually set up their print jobs, not to mention the long waiting times during peak hours. With the introduction of HCMUT_SSPS, these inefficiencies are poised to become a thing of the past. The service promises a seamless digital integration, allowing students to remotely access the system, upload their documents, and set their printing preferences, all from the comfort of their own devices.

The HCMUT_SSPS system is meticulously designed to ensure ease of use and flexibility. Once a document is uploaded, students can select from various printers scattered around the campus, ensuring they can always find a nearby machine. This is particularly significant considering HCMUT's sprawling layout with two main campuses and around ten buildings in each. To further the commitment to accessibility, the university has decided to install at least one printer in every building, making sure no student is ever too far from a printing facility.

Beyond mere printing, the system offers a range of customizable settings. Whether a student wishes to specify the paper size, select certain pages, opt for double-sided printing, or decide on the number of copies, HCMUT_SSPS provides these options at their fingertips. Such features not only promote a personalized printing experience but also contribute to sustainable practices by reducing paper waste.

Stakeholders and needs

- Students: Need to use the system to print documents.
- Student Printing Service Officer (SPSO): Need to manage what and how much each student uses the system to print documents.
- IT Staff: Keep track and maintain, upgrade the system.

Benefits

- Students: Students can print documents autonomously for free (unless they use more than the number of default pages), see all of their uploaded files and how much they have printed up to a certain amount of time (i.e a month).
- SPSO: Easily keep track of printing activity in order to have immediate response to any issues that occur and be able to gain data for statistical and other purposes (warranty pur).

1.2 Functional and Non-functional requirements

Functional requirements

- <u>Students:</u>
  + A map of available printers in the system can be seen by all students.
  + Each student has their personal drive space in the system so that they can upload their files.
  + Students are able to choose a printer and specify the printing properties before printing.
  + Each student can see their page balance.
  + Each student can access the BKPay system to pay for extra pages and choose one of the available payment options.
  + Students are able to see their printing history up to one month before.
  + Students are able to cancel during the printing process.
- <u>SPSO:</u>
  + Is able to know which printer runs out of paper or ink.

+ Is able to add, enable, disable a printer.
+ Is able to manage other configurations of the system.
+ Is able to view the reports of the printing system anytime.
+ Is able to control who accesses the printer.
+ Is able to track the cost of printing to help cost control.

- IT staff:
  + The system should be well organized and developed.
  + The system should keep detailed logs of any errors.
  + The system should have clear documentation on how it works and how to maintain it.
  + The system must provide mean to contact the system distributor for support.
  + The system must provide tool for update to add more features or to fix bugs.

Non-Functional requirements

- Performance:
  + The web-based app and the mobile app takes no more than 5 seconds to load.
  + Response time for each action is less than 1 second.
  + The app must be able to handle multiple print jobs simultaneously at most 5 seconds delay during peak times such as exams.
- Usability:
  + The system interface will be in English, with the ability to switch to Vietnamese in future versions.
  + All students and staff should be able to operate the system under 1 hour of usage.
  + The app server is able to work well in office hours and recover quickly from any failures.
  + The system should be compatible with a wide range of devices (Phone, PC) and operating systems (MacOS, Window).
- Security:

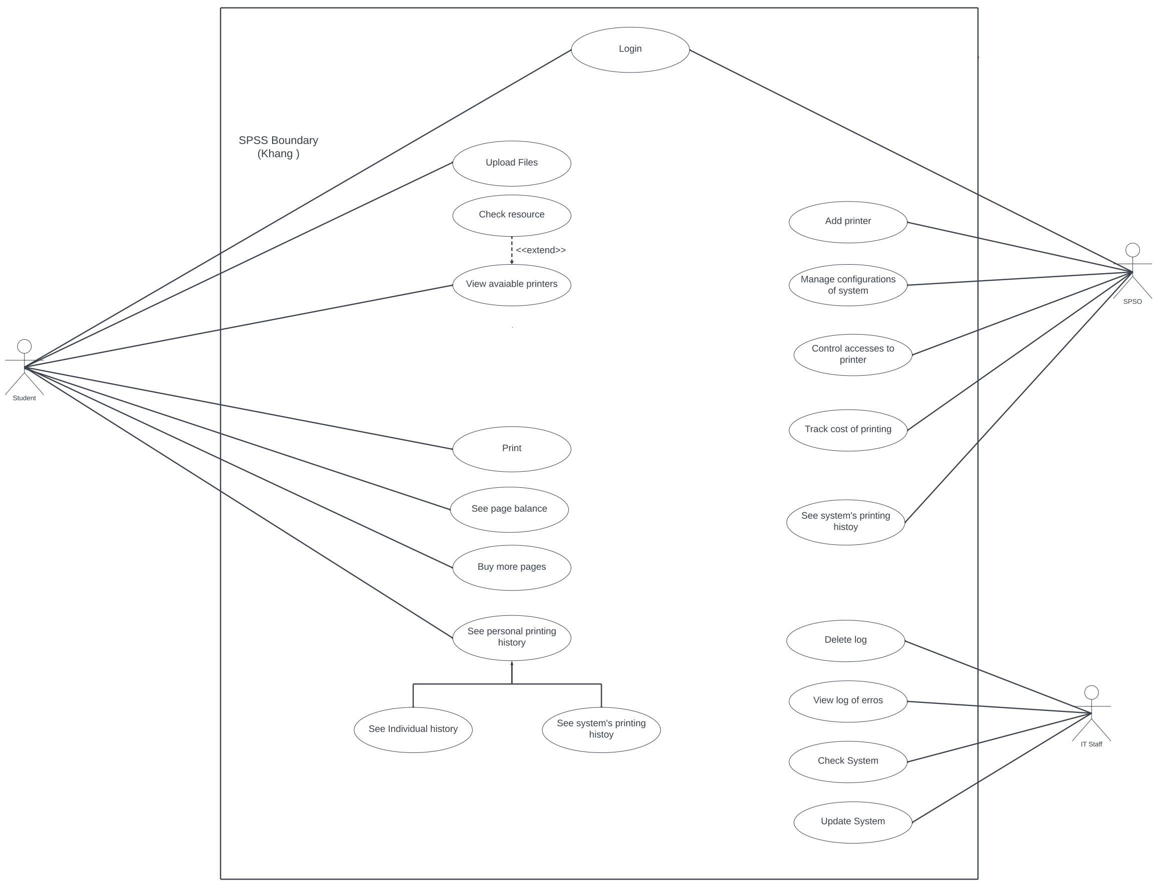
+ The system must protect private information of users and prevent unauthorized accesses or invalid actions.

- Maintainability:

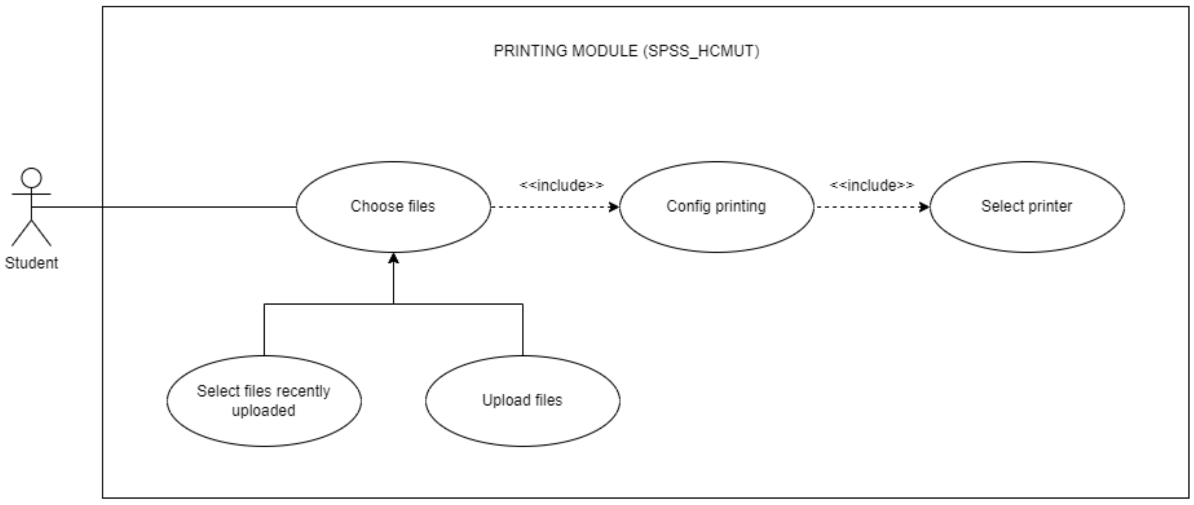
  + The system should go down monthly for maintenance, with the expected downtime being no more than 1 hour.
  + The system should be easy to update and maintain.
  + The system must not change any user data after update.
  + The system must be able to save logs of errors that occur.

1.3 Use-case diagram

1.3.1 Whole System



1.3.2 Printing Module



Use Case ID:	**UC1**
Use Case Name:	**Choose files**
Description:	Choose files to print
Actor:	Student
Preconditions:	- The user is logged in - The user's balance is not zero
Postconditions:	The user is directed to the next printing step (Config printing)
Basic Flow:	1. The user wants to use the printing system 2. The user is asked to choose files for printing 3. The user chooses the desired method to upload files or choose from recent files 4. The system directs the user to the corresponding screen
Alternative Flows:	None
Extension Points:	None
Exceptions:	- The user's balance is zero - The user cancels the printing process

Table 1.1: Choose Files

Use Case ID:	**UC1.1**
Use Case Name:	**Select files recently uploaded**
Description:	Select files recently uploaded for printing
Actor:	Student
Preconditions:	- The user is logged in. - The user's balance is not zero. - The user initiated use case UC1.
Postconditions:	The user is directed to the next printing step (Config printing)
Basic Flow:	1. The user wants to choose files from recent. 2. The system shows the list of recent files. 3. The user chooses their desired files. 4. The system saves the files.
Alternative Flows:	None
Extension Points:	None
Exceptions:	- There are no recent uploaded files. - The user cancels the printing process.

Table 1.2: Select files recently uploaded

Use Case ID:	**UC1.2**
Use Case Name:	**Upload files**
Description:	Upload files from device for printing
Actor:	Student
Preconditions:	- The user is logged in. - The user's balance is not zero. - The user initiated use case UC1.
Postconditions:	The user is directed to the next printing step (Config printing)
Basic Flow:	1. The user wants to upload files from their device. 2. The system shows the list of files from their device. 3. The user chooses their desired files. 4. The system saves the files.
Alternative Flows:	None
Extension Points:	None
Exceptions:	- There are no files on their device. - The user cancels the printing process.

Table 1.3: Upload files from device for printing

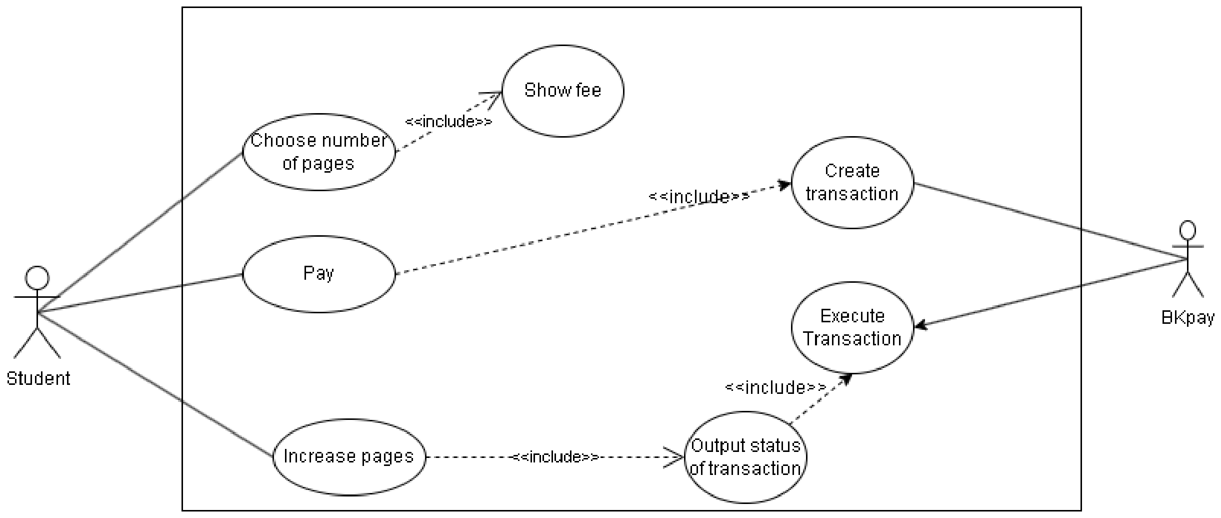
Use Case ID:	**UC1.3**
Use Case Name:	**Config printing**
Description:	Config berfore printing
Actor:	Student
Preconditions:	The user selected their files and is directed from UC1
Postconditions:	The user is directed to the next printing step (Select printer)
Basic Flow:	1. The user wants to config the print. 2. The system shows the available configuration. 3. The user configs what they need. 4. The system saves those configuration.
Alternative Flows:	The user uses the default configuration
Extension Points:	None
Exceptions:	The user cancels the printing process

Table 1.4: Config printing

Use Case ID:	**UC1.4**
Use Case Name:	**Select printer**
Description:	The user selects a printer to print their documents
Actor:	Student
Preconditions:	1. The user is directed from use case UC2. 2. There are at least one printer available.
Postconditions:	The document is printed
Basic Flow:	1. The system shows the list of available printers. 2. The user selects their desired printer. 3. The system sends the uploaded files and the configuration to the chosen printer. 4. The printer prints the documents.
Alternative Flows:	None
Extension Points:	None
Exceptions:	- There are no available printers. - The user cancels the printing process.

Table 1.5: Select printer

1.3.3 Buy Pages



Use Case ID:	**UC2.1**
Use Case Name:	**Choose number of pages**
Description:	Student choose how many pages they want to buy.
Actor:	Student
Preconditions:	Student already logged in the app.
Postconditions:	The student will be able to pay.
Basic Flow:	1.The system shows options of payment and fee for each option. 2.Student choose one of these options. 3.The student now be able to click on pay button. 4. The system route to BKpay.
Alternative Flows:	None
Extension Points:	None
Exceptions:	- Student don't want to buy anymore. - Student click on cancel button to back to the main menu.

Table 1.6: Choose number of pages

Use Case ID:	**UC2.2**
Use Case Name:	**Pay**
Description:	Student pay for the chosen option.
Actor:	Student
Preconditions:	Student already choose the option.
Postconditions:	The student will be directed to Bkpay.
Basic Flow:	1. The student click on pay button. 2. The system send a request to create a transaction and route student to Bkpay.
Alternative Flows:	None
Extension Points:	None
Exceptions:	- Student don't to buy anymore 2. Student click on cancel button to back to the app.

Table 1.7: Pay

Use Case ID:	**UC2.3**
Use Case Name:	**Create transaction**
Description:	Student pay for the chosen option.
Actor:	BKpay.
Preconditions:	UC2.2 was successed.
Postconditions:	A transaction created.
Basic Flow:	1. The BKpay system received the request from the SPSS. 2. The system create a bill and show the bill information to the student.
Alternative Flows:	None
Extension Points:	None
Exceptions: 2. No transaction created.	1. Student cannot connect to Bkpay.

Table 1.8: Create transaction

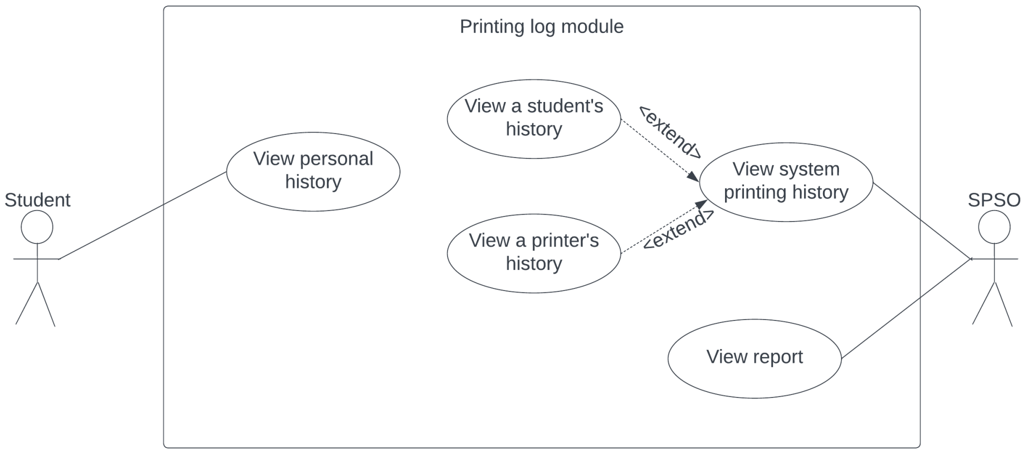
Use Case ID:	**UC2.4**
Use Case Name:	**Execute transactions**
Description:	Student execute their transaction.
Actor:	1.Student. 2.Bkpay.
Preconditions:	Student ticked on the transaction.
Postconditions:	The SPSS will get transaction status.
Basic Flow:	1. The student click on pay button of Bkpay website. 2. The system execute the transaction. 3. The system return status of the transaction to SPSS.
Alternative Flows:	None
Extension Points:	None
Exceptions:	1. Cancel before process completely finished. 2. Fail status return to SPSS. 3. No money of student would be lost.

Table 1.9: Execute transactions

Use Case ID:	**UC2.5**
Use Case Name:	**Increase pages**
Description:	SPSS update student page balance.
Actor:	1.Student. 2.SPSS.
Preconditions:	The status of UC2.4 is success
Postconditions:	The student will get their page balance increased.
Basic Flow:	1. The system receives response of BKpay. 2. The system increases pages of student. 3. The system saved the history of transactions.
Alternative Flows:	None
Extension Points:	None
Exceptions:	1. status of UC2.4 is fail. 2. Student don't get more pages

Table 1.10: Increase pages

1.3.4 View History



Use Case ID:	**UC3.1**
Use Case Name:	**View personal history**
Description:	The student wants to view their printing log
Actor:	Student.
Preconditions:	- Student has already logged in to the system. - Student has already printed something at least once using the system
Postconditions:	System displays the printing logs as well as the total number of pages.
Basic Flow:	1. Student choose the option "View history" from the "Account" menu. 2. Student choose the month that they want to see their history. 3. System gets the printing logs of that student in that month. 4. System calculates the total number of pages that student printed in that month. 5. System displays the printing logs as well as the total number of pages used in that month.
Alternative Flows:	None
Extension Points:	None
Exceptions:	None

Table 1.11: View personal history

Use Case ID:	**UC3.2**
Use Case Name:	**View system printing history**
Description:	The SPSO wants to view all the printing logs in the system.
Actor:	SPSO.
Preconditions:	The SPSO has been authenticated by the system.
Postconditions:	System displays the printing logs to the SPSO.
Basic Flow:	1. The SPSO choose the option "View system printing logs" from their main menu. 2. The SPSO choose a specific time period (date to date) in the "Time" filter. 3. System gets all the printing logs from that time period. 4. System displays the printing logs to the SPSO.
Alternative Flows:	None
Extension Points:	- View a student's history (UC3.2.1). - View a printer's history (UC3.3.2).
Exceptions:	2a The time period is invalid. - An error message shows up, back to step 2.

Table 1.12: View system printing history

Use Case ID:	**UC3.2.1**
Use Case Name:	**View a student's history**
Description:	The SPSO wants to see the printing logs of a specific student.
Actor:	SPSO.
Preconditions:	Step 2 of use case UC3.2 is completed.
Postconditions:	Systems displays the printing logs of the chosen student to the SPSO
Basic Flow:	1. The SPSO enters the Student ID (sID) into the "Student ID" filter. 2. System gets the printing logs with the identified sID in the specified time period. 3. Systems displays the printing logs of that student to the SPSO
Alternative Flows:	1a. The SPSO wants to find that student by other information. 1. The SPSO choose the option "Find Student". 2. The SPSO uses the filter for the information they have (name, email, phone number). 3. System displays the students with the corresponding information. 4. The SPSO choose the student that they want to see from that list. 5. The system identifies the sID of that student Continue step 2.
Extension Points:	None
Exceptions:	1a. The sID is not found. - An error message pops up, return to step 1.

Table 1.13: View a student's history

Use Case ID:	**UC3.2.2**
Use Case Name:	**View a printer's history**
Description:	The SPSO wants to see the printing logs of a specific printer.
Actor:	SPSO.
Preconditions:	Step 2 of use case UC3.2 is completed
Postconditions:	Systems displays the printing logs of the chosen printer to the SPSO.
Basic Flow:	1. The SPSO enters the Printer ID (pID) into the "Printer ID" filter. 2. System gets the printing logs with the identified pID in the specified time period. 3. Systems displays the printing logs of that printer to the SPSO.
Alternative Flows:	None
Extension Points:	None
Exceptions:	1a. The pID is not found. - An error message pops up, return to step 1.

Table 1.14: View a student's history

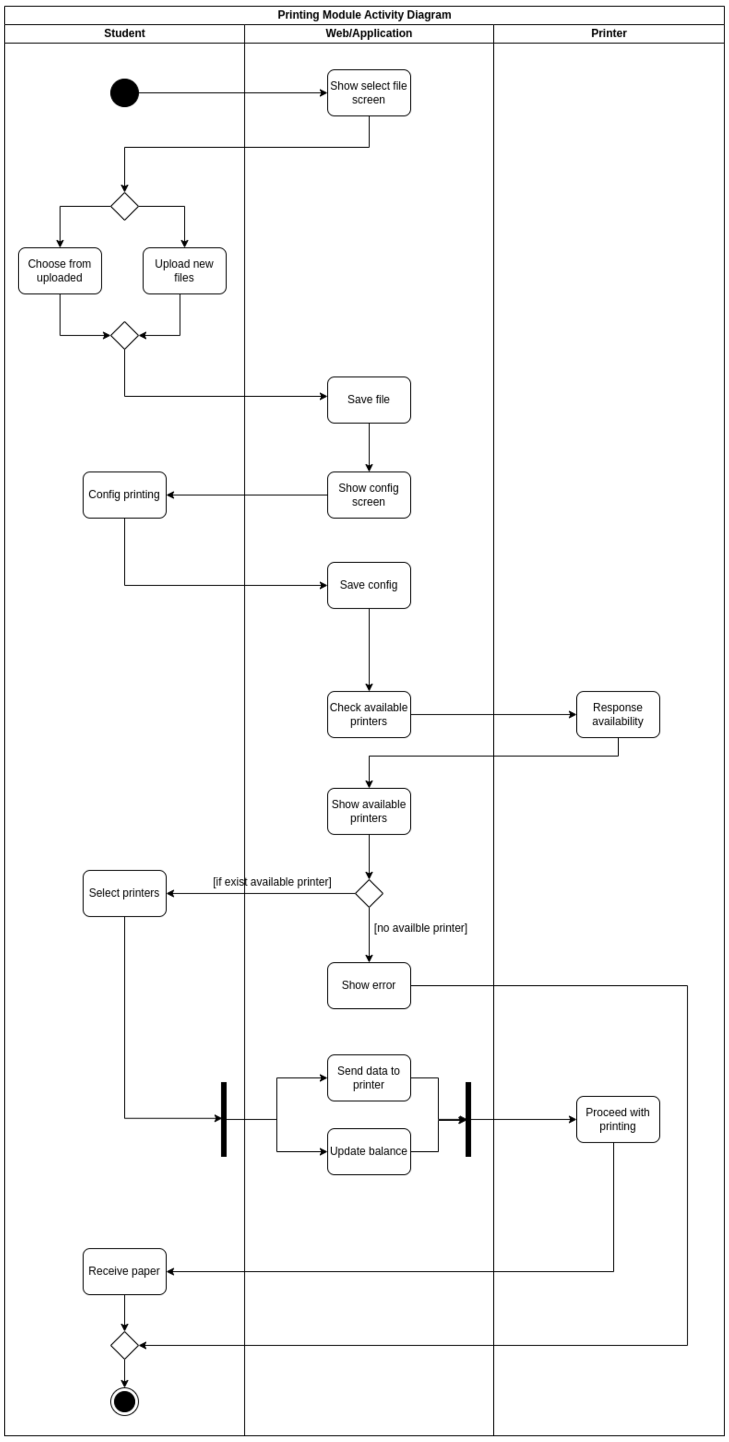
Use Case ID:	**UC3.3**
Use Case Name:	**View report**
Description:	The SPSO wants to view the monthly/annual report generated automatically by the system.
Actor:	SPSO.
Preconditions:	- The SPSO has been authorized by the system. - A report must be generated by the end of each month and each year.
Postconditions:	The system displays the generated report at the end of the specified time.
Basic Flow:	1. The SPSO chooses the "Show Report" from their main menu. 2. The SPSO chooses the month and the year that they want the report. 3. The system displays the report generated at the end of that month.
Alternative Flows:	2a. The SPSO only chooses the year - The system displays the annual report instead.
Extension Points:	None
Exceptions:	None

Table 1.15: View a student's history

2 System modelling

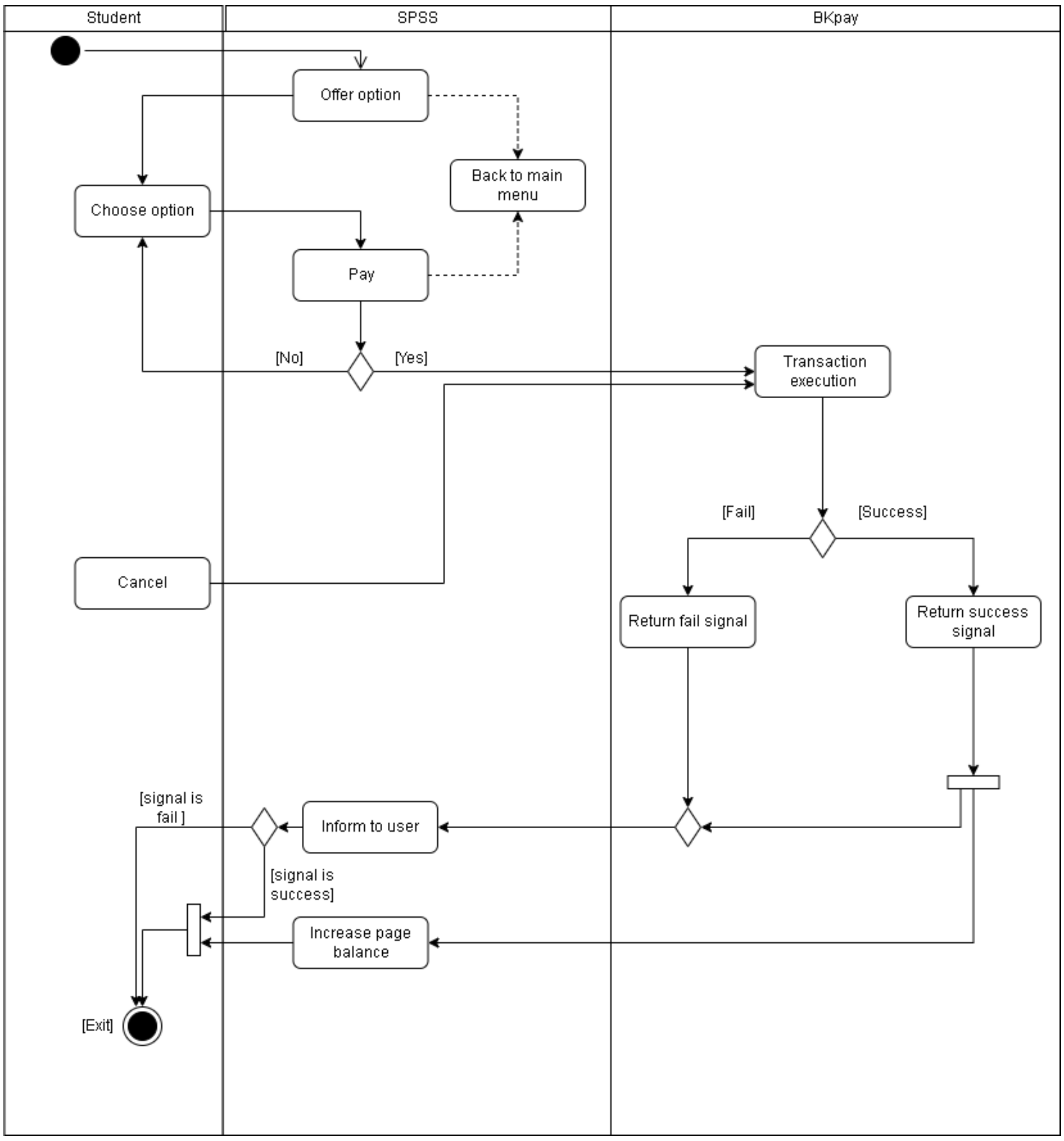
2.1 Activity Diagram

2.1.1 Printing



In the activity diagram, we have three swimlanes: Student, Application, and Printer. The flow begins with the Student stakeholders expressing their desire to use the printing module and ends with the Student obtaining the printed documents. This module encompasses three core activities in the following sequence: Upload/select files, configure, and choose a printer.

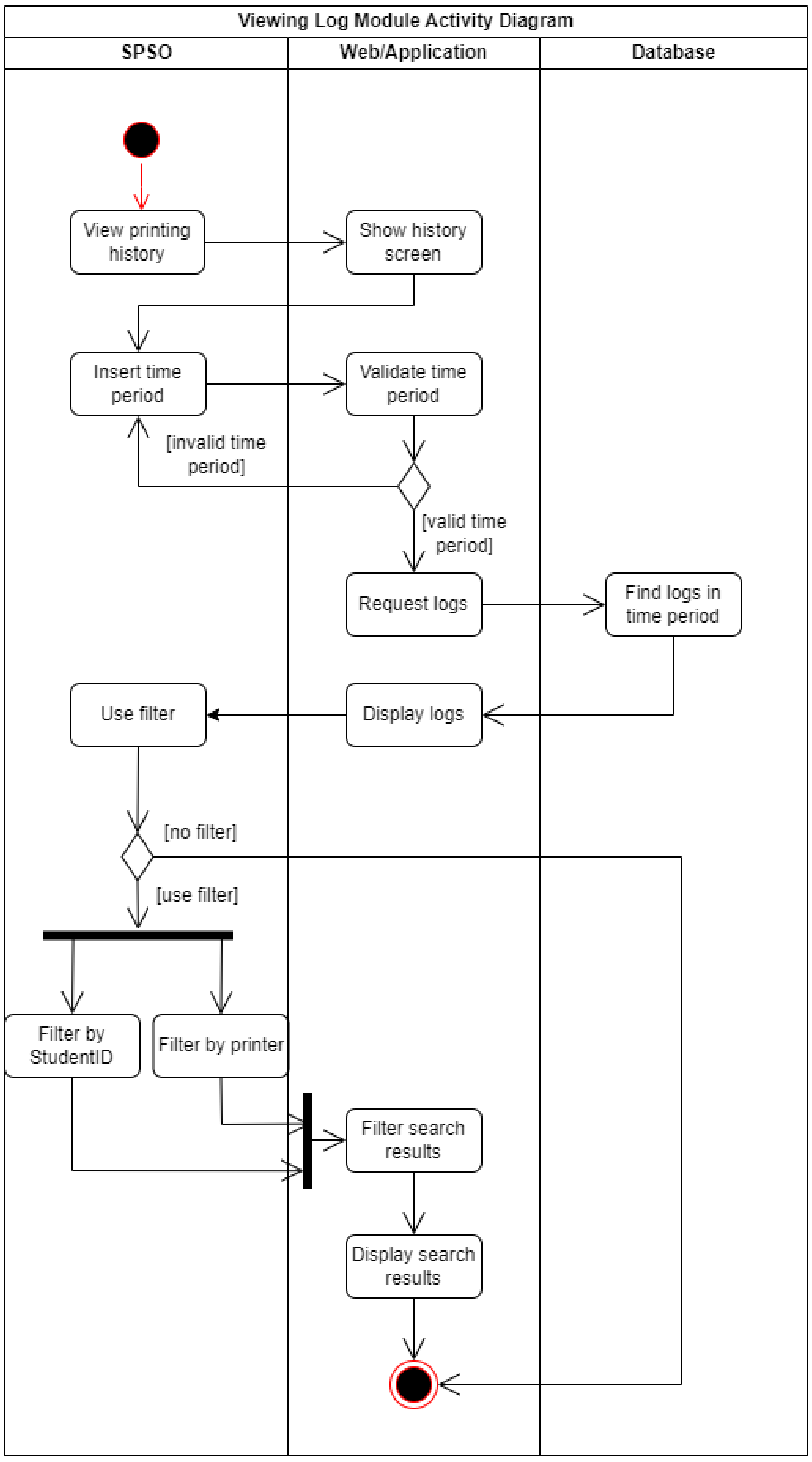
2.1.2 Buy Pages



Description:

This is a swimlane diagram that represents the process of a student making a payment using BKPay. The diagram is divided into four lanes representing the Student, SPSS, BKPay and Bank. Student Lane: The process begins with the student choosing a page and then deciding to pay. If the student decides not to pay, the process ends. If they decide to pay, the process moves to the SPSS lane. SPSS Lane: Here, the SPSS redirects student to BKPay. BKPay Lane: The transaction will be executed here. If the message return from PSP is success then it will inform to user and SPSS so SPSS can increase the page balance, otherwise, the transaction was failed.

2.1.3 View system printing logs



Description:

This is the activity diagram of the “View system printing history” use case and its affiliated use cases.

- The SPSO after opening the “View system history” screen will input the time period in which they’d like to see the logs.

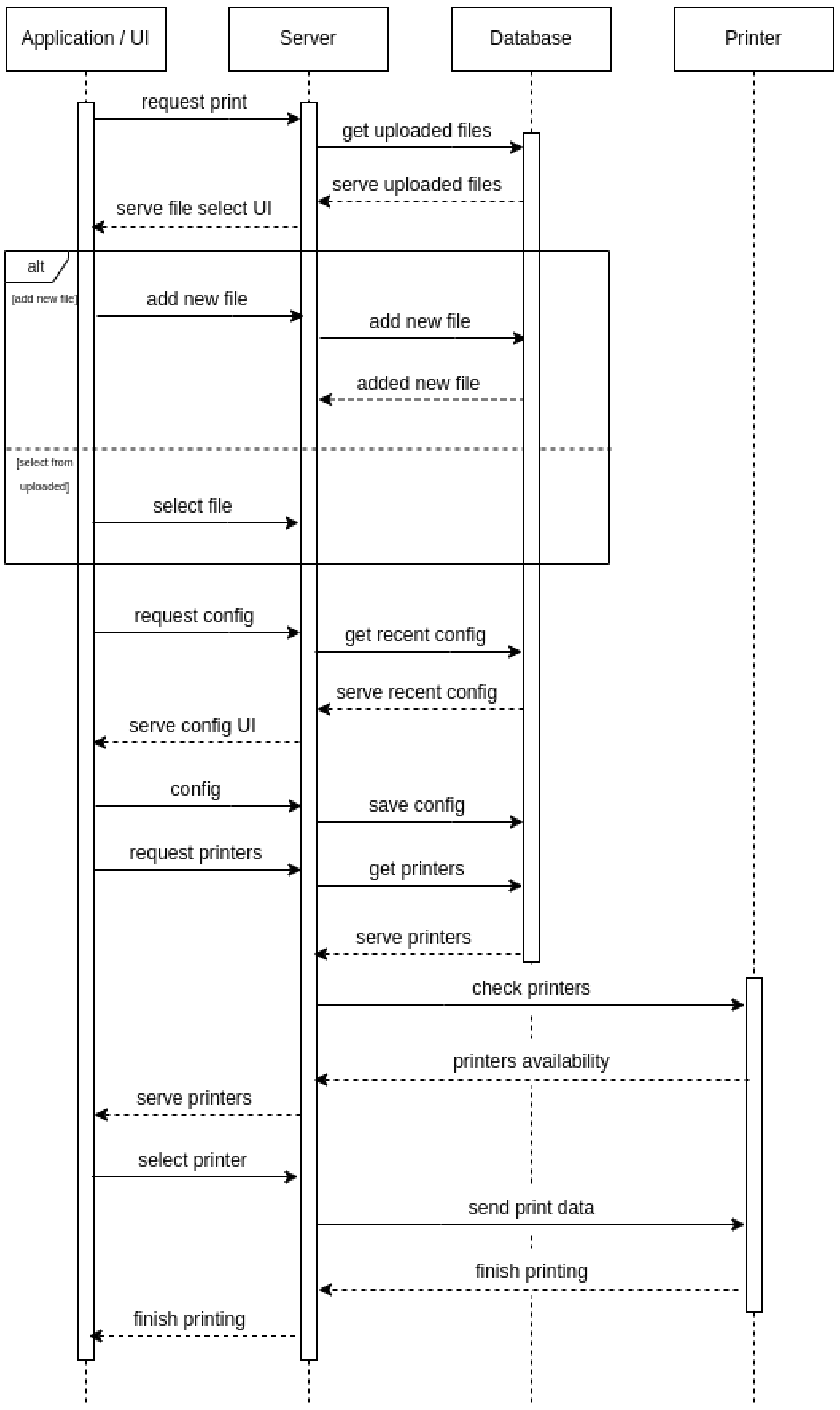
- The system then does a check to ensure that the time period is valid. If it’s invalid then the SPSO will have to try again.

- Otherwise, the system will make a request (query) to the database so that the database can then find all the logs that are in the time period. The system then displays the search results to the SPSO.

- Optionally, the SPSO can use the filter to find the logs of a specific student or printer. These can be done in parallel with each other, meaning the filter can either be used for one or even both of the student and the printer.

2.2 Sequence Diagram

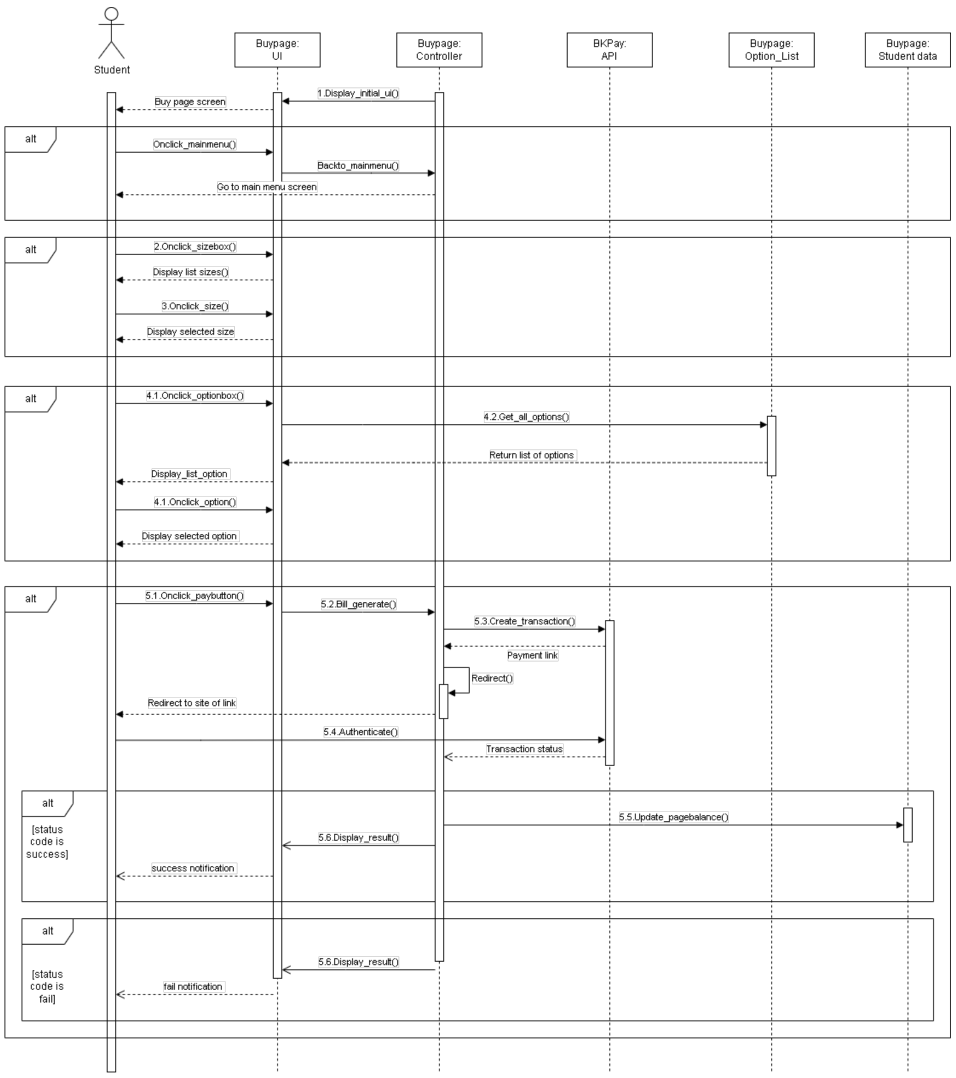
2.2.1 Printing

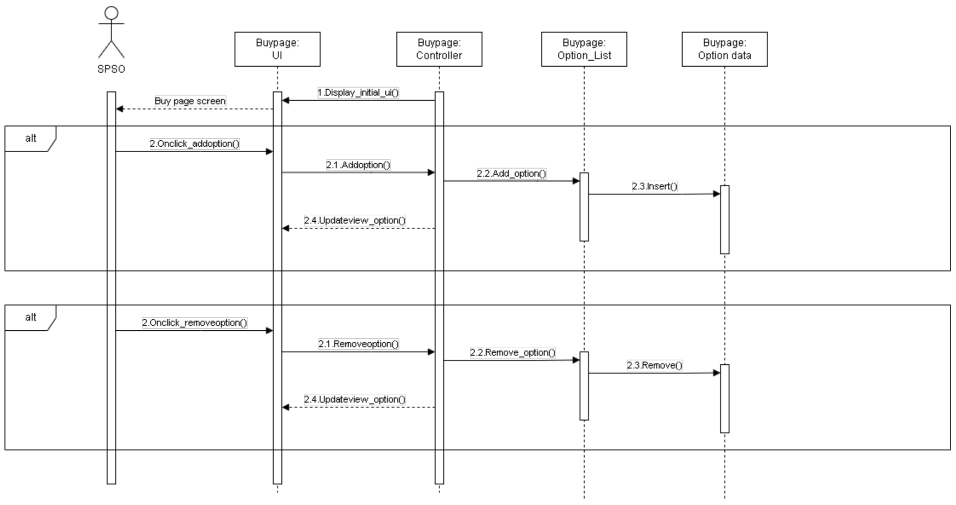


Description:

In the sequence diagram, the primary focus is on internal system communication, including interactions among the application, the server, the database, and the printers. The application and the server remain active throughout the printing process, while the database completes its actions after providing the list of printers. As expected, the printers join the process at a later stage and become inactive after the printing is completed.

2.2.2 Buy Pages



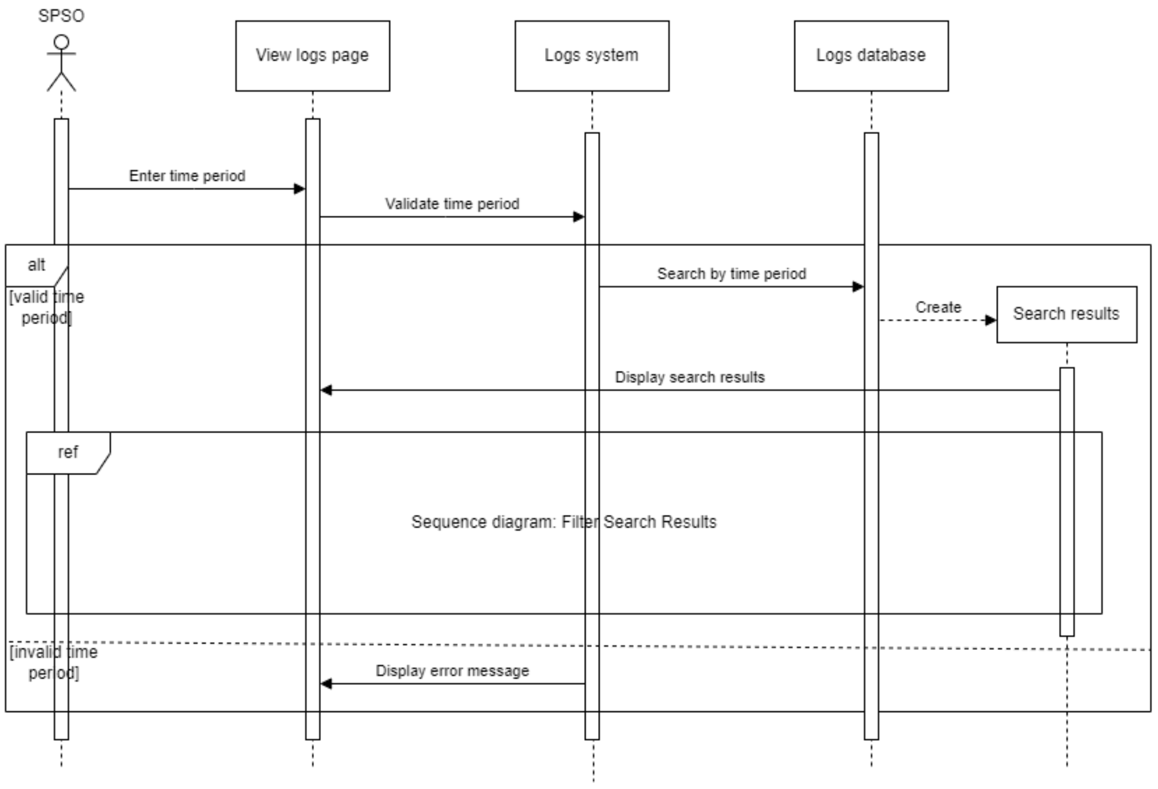


Description:

This sequence diagram describes the interaction between a Student with (UI) SPSS and a payment webpage (BKPay). Here's how it works:

1. The user starts at the UI window and chooses n.o page options.
2. If the user click on page sizes dropbox the list of sizes will be displayed.
3. If the user click on number of pages dropbox the list of option will be displayed.
4. If the user clicks on the "Pay" button. This action initiates the payment process.
5. The user is then redirected to the BKPay payment site and transaction will be executed here.
6. Finally, the BKPay webpage return message to SPSS. If it is succesful payment then student will be inform and his/her page balance will be increased, otherwise, student only receive message about unsuncessful transaction.

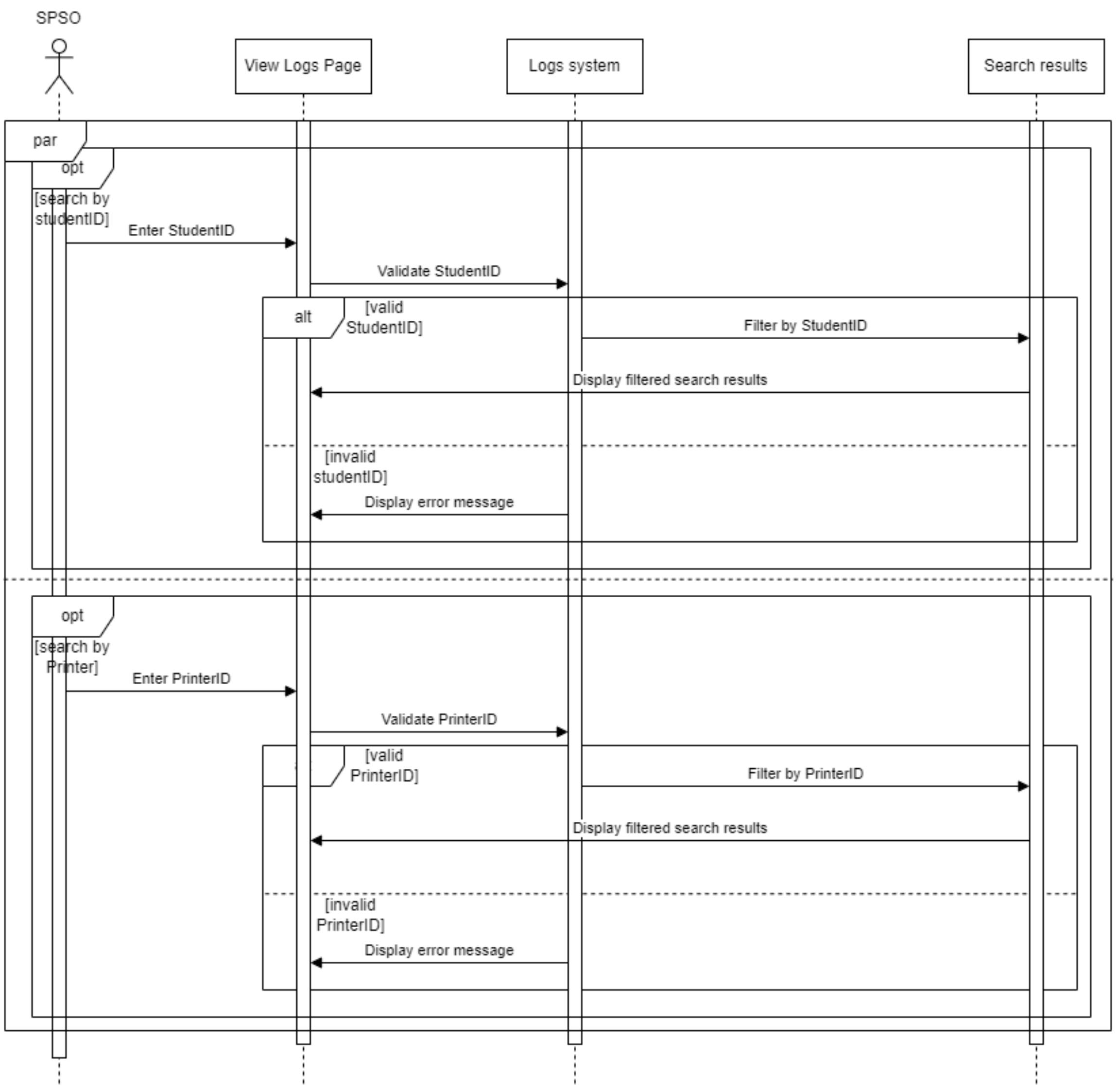
2.2.3 View system printing logs



Description:

- The diagram follows the MVC model, with the View logs page as View, System as Controller and the Database as Model.
- After the SPSO enters the time period on the screen, the View logs page passes the time period to the system to validate the time period.
- If the time period is valid, the System makes a query to the database, which will create a Search Results object. This object will display the results back to the View logs page.
- Otherwise, the system tells the page to display an error message.
- The filter options are referenced from the Filter search results diagram

Filter search results

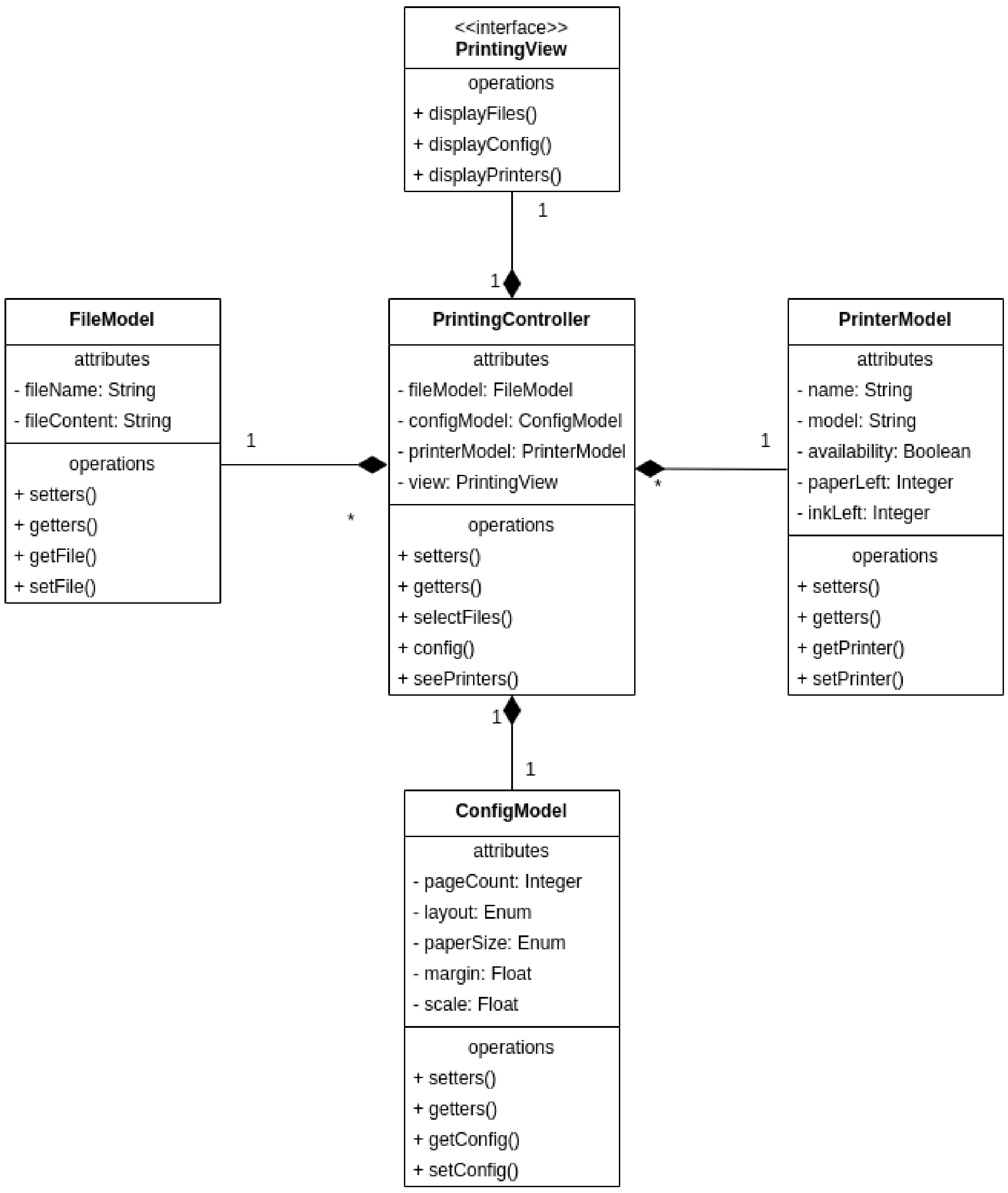


Description:

- The diagram also follows the MVC model, however only the Search Results object created by the Database object is used as Model.
- The filter for studentID and printer are both optional, and run parallel to each other. When a filter is used, the System validates the data before filtering the Search Results and returning the filtered logs to the View logs page.

2.3 Class Diagram

2.3.1 Printing

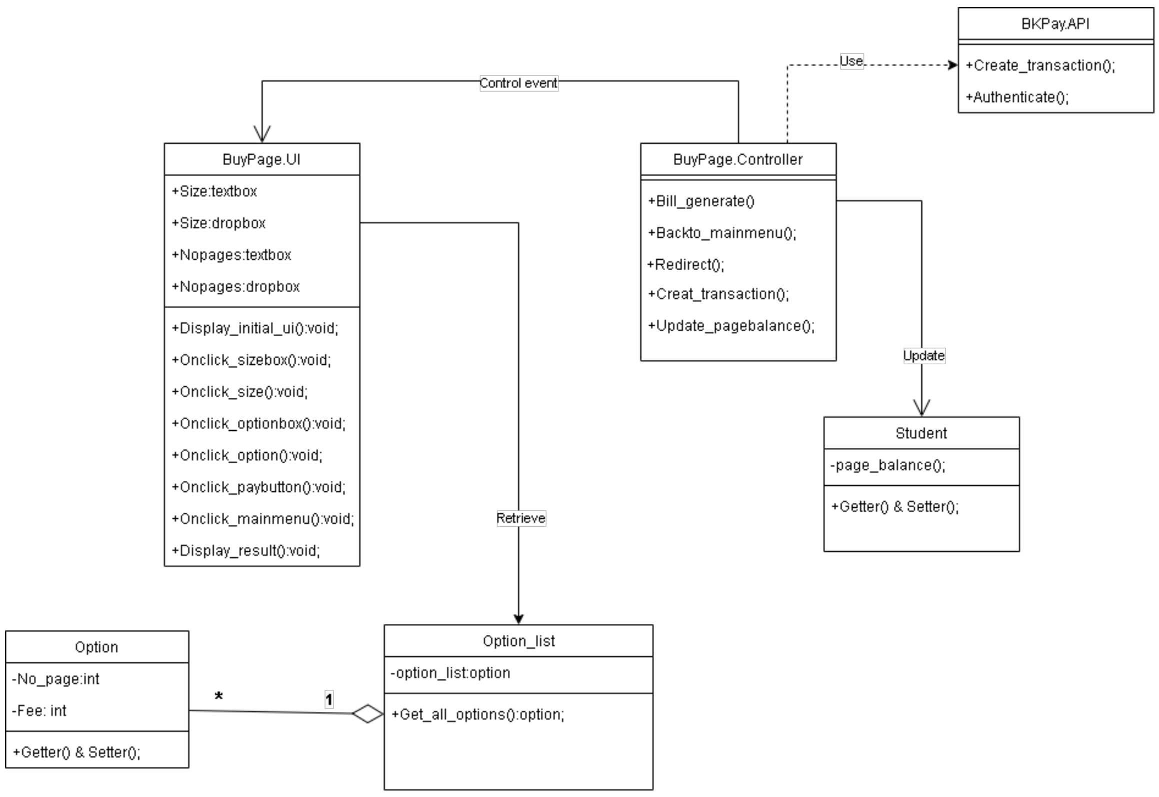


Description:

Following the MVC (Model-View-Controller) pattern, we've segmented our module into three key components: Model (comprising FileModel, ConfigModel, and Printer-

Model), View (utilizing the PrintingView interface), and Controller (implemented as PrintingController). The controller takes center stage within the module, initializing instances of both the model and view. The relationship between the controller and other classes is one of composition, as the controller holds these class instances.

2.3.2 Buy Pages



Description:

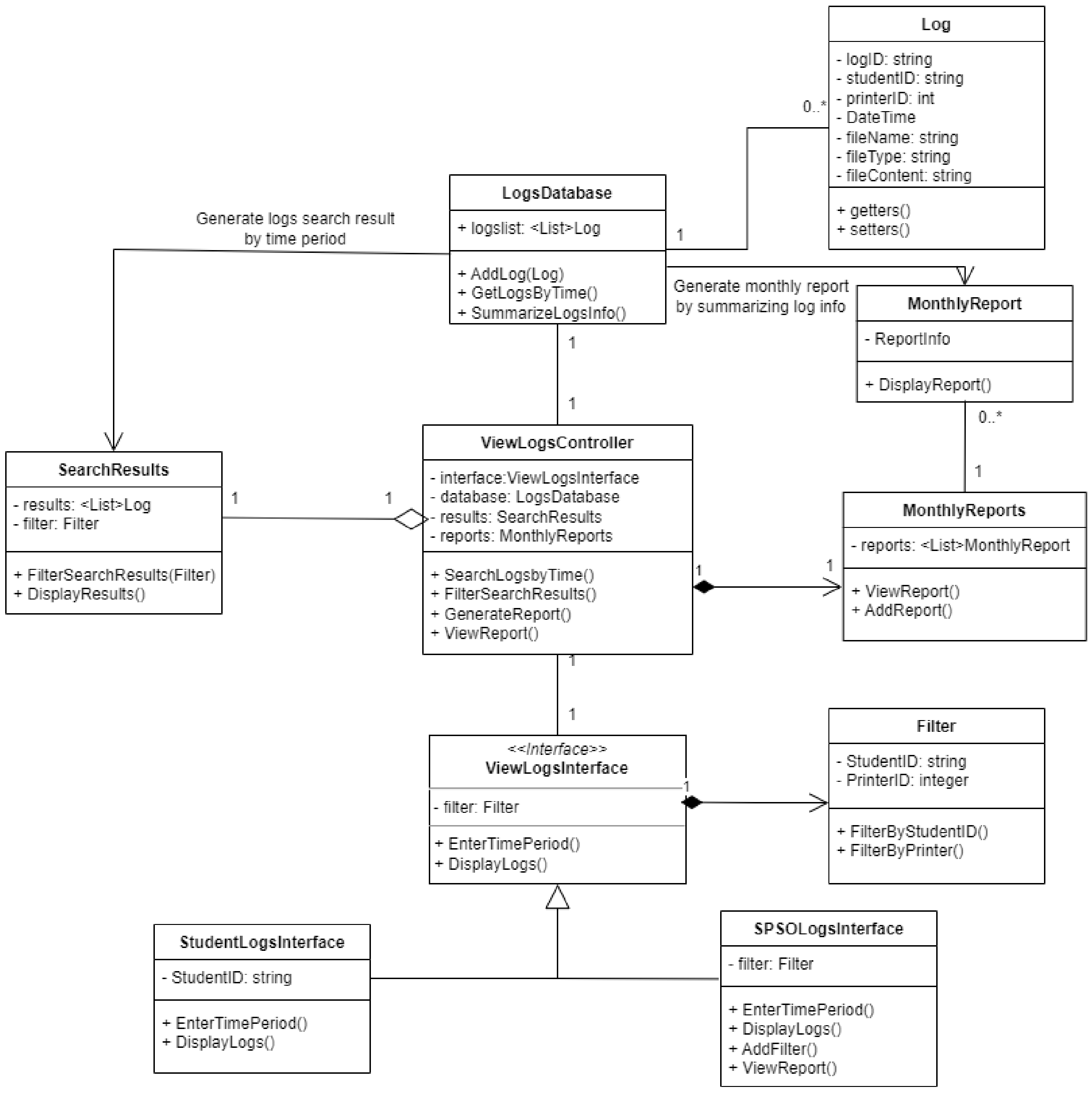
- Buypage.UI: This class is the user interface of the system. It has methods to display to screen and let user input by listening on click event via onclick methods. It can also retrieve data from the buypage model (option_list).

- SPSS Controller: This class is the controller in the system, which manages the action of user through UI. It has methods for getting and setting options and increasing page size.

- SPSS Option: This class represents an individual option in the system. It has methods for getting and setting options.

- SPSS Option List: For adding and removing option in the list or getting all option info.

2.3.3 View system printing logs

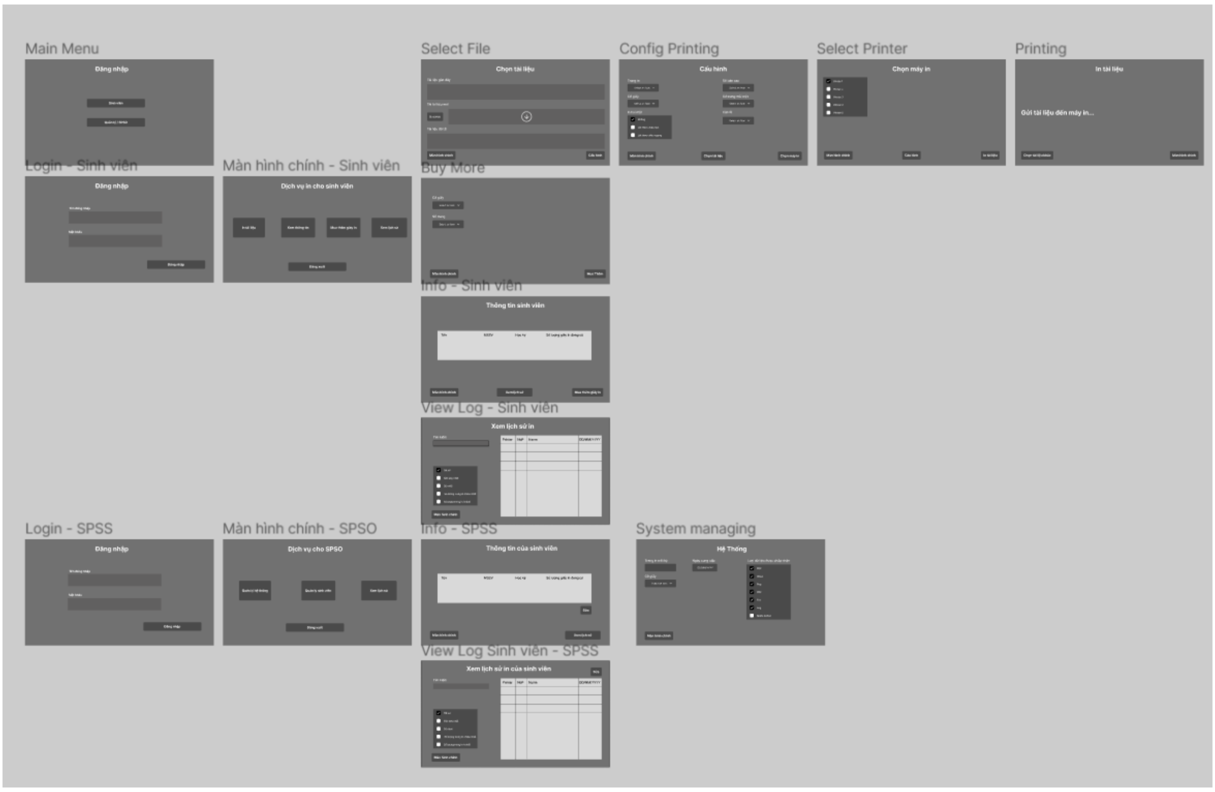


Description:

This is the class diagram for the entire View Logs module, and it follows the MVC model, with the ViewLogsInterface as the View, ViewLogsController as the Controller, and the LogsDatabase as the Model.

- The ViewLogsInterface are specialized into StudentLogsInterface and SPSOLogsInterface. The StudentLogsInterface which is used by the student has their unique StudentID and can only see their own logs, meanwhile the SPSOLogsInterface can search the entire system, create a filter and view monthly reports.
- The ViewLogsController can communicate with the LogsDatabase to create SearchResults, and also automatically generate a MonthlyReport. The list of MonthlyReports are also stored in this class, and can be accessed by the SPSOLogsInterface.
- The LogsDatabase stores a list of Logs that are automatically generated after each printing action. They can create SearchResults and summarizes information to create a MonthlyReport.

2.4 Mock Up

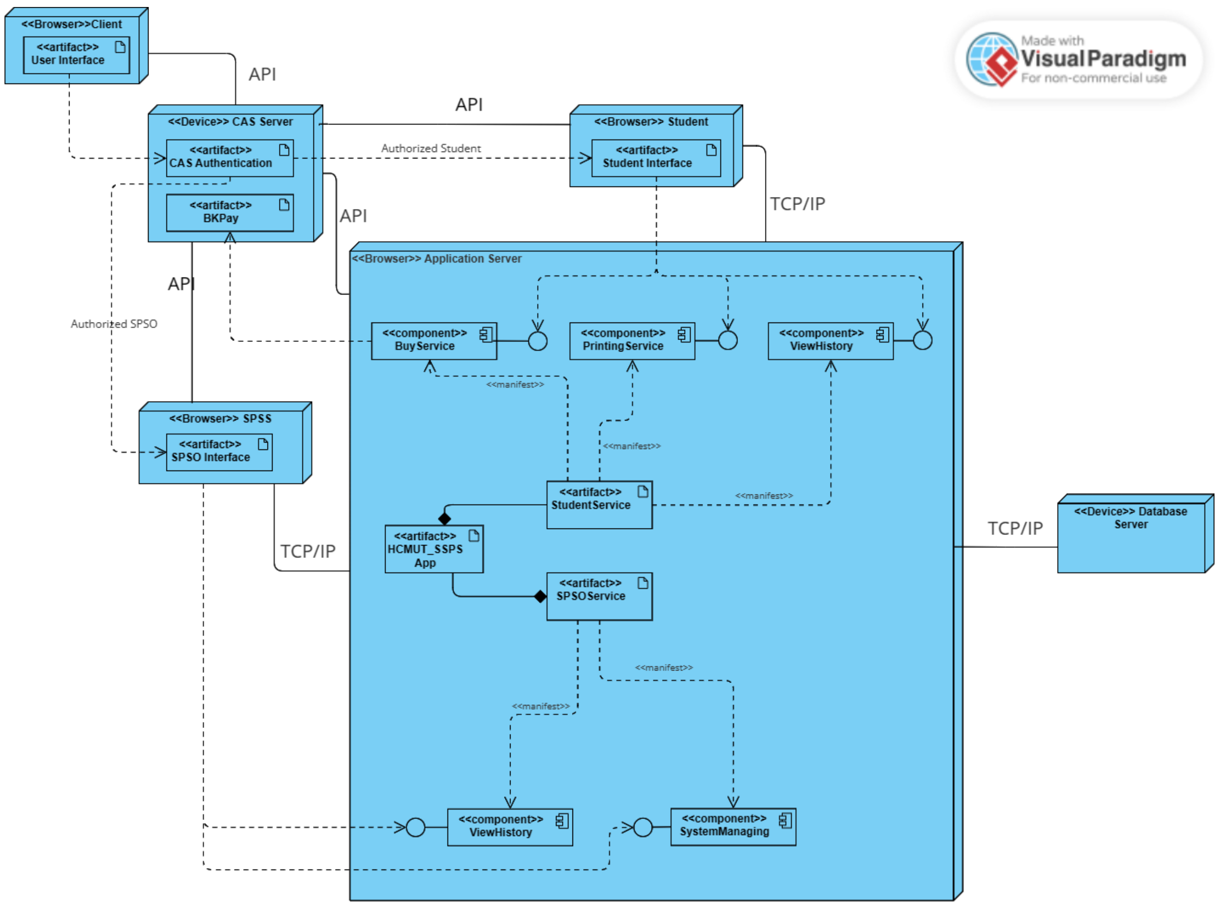


Click here to get access

3 Architecture design

3.1 Architecture design

3.1.1 Architectural Diagram:


<<Browser>>Client
<<artifact>> User Interface
API
<<Device>> CAS Server
<<artifact>> CAS Authentication
<<artifact>> BKPay
API
Authorized Student
<<Browser>> Student
<<artifact>> Student Interface
TCP/IP
API
API
Authorized SPSO
<<Browser>> Application Server
<<component>> Buy Service
<<component>> Printing Service
<<component>> ViewHistory
<<manifest>>
<<Browser>> SPSS
<<artifact>> SPSO Interface
<<artifact>> StudentService
<<artifact>> HCMUT_SSPS App
<<artifact>> SPSOService
TCP/IP
TCP/IP
<<Device>> Database Server
<<component>> ViewHistory
<<component>> SystemManaging
Made with Visual Paradigm For non-commercial use

Description:

- Students and SPSO staff log in through their respective interfaces on the browser (Client) and use APIs to communicate with the CAS Server.

- The CAS Server authenticates accounts and classifies the user based on the login information.

- After authentication, students gain access to the Student Interface and SPSO staff to the **SPSO Interface** via their browser.

- The **Student Interface** includes services: printing (Printing), viewing history (View History), and purchasing paper (BuyPaper).
- The Application Server provides services to users through components: **BuyService**, **PrintingService**, and **ViewHistory**, with which students can interact after successful login.
- APIs related to **StudentService** allow the execution of student-specific functions, while **SPSOService** serves the functions for SPSO staff.
- The Application Server queries and communicates with the Database Server using the TCP/IP protocol to retrieve necessary information, store new data, or update existing data.
- BKPay is an integrated payment service to handle transactions between students and the system and connect with API connections.

3.1.2 API connections:

- **Client to CAS Server:** A client interface communicates with the CAS (Central Authentication Service) server via API for user authentication. This likely involves sending credentials and receiving tokens or session data.
- **CAS Server to Application Server:** After successful authentication of a student by the CAS Server, the student can interact with the Application Server through an API. This indicates a secure gateway through which requests and responses are exchanged post-authentication.
- **Transaction Process with BKPay:** When the student decides to purchase pages, the Application Server interacts with the BKPay application server. The Application Server will send a request to BKPay to create a bill for the pages the student wants to buy. The billing process includes authorizing the transaction, which might involve verifying payment information and ensuring there are sufficient funds.

3.1.3 Data storage:

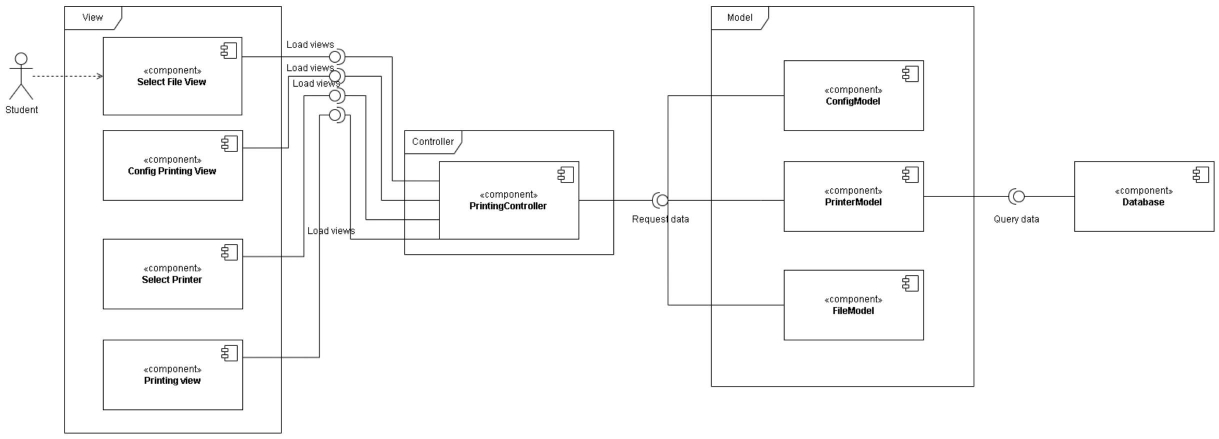
With most of the HCMUT_SSPS system's data being structured data in addition to the high frequency in reading and writing of the data, a database storage model using a relational database management system will be most fitting for storing the data of the system. The database will be stored in an SPSO's authorized server, connecting to the

Application Server via the TCP/IP protocol. The Database Server stores the following data:

- Students, SPSO Staffs, Printers are stored in 3 tables with their own attributes
- Students' Files and Printing configurations: While unstructured data such as the various file types uploaded by students can be more efficiently stored as Binary Large Object (BLOB) files in the database, storing them as a database table allows for faster queries and better security of personal files for each students.
- Printing Log: To allow for filtering by student, by printer and by time period, printing log of all printers under the system is stored in an indexed table. A database also makes it easy to read and maintain.
- Monthly Report Database: The list of automatically-generated reports are stored as a table for query access to the reports of specific months.
- Other data entities with single instance such as SPSO's configurations can be stored in the Database Server as BLOB files

3.2 Business logic

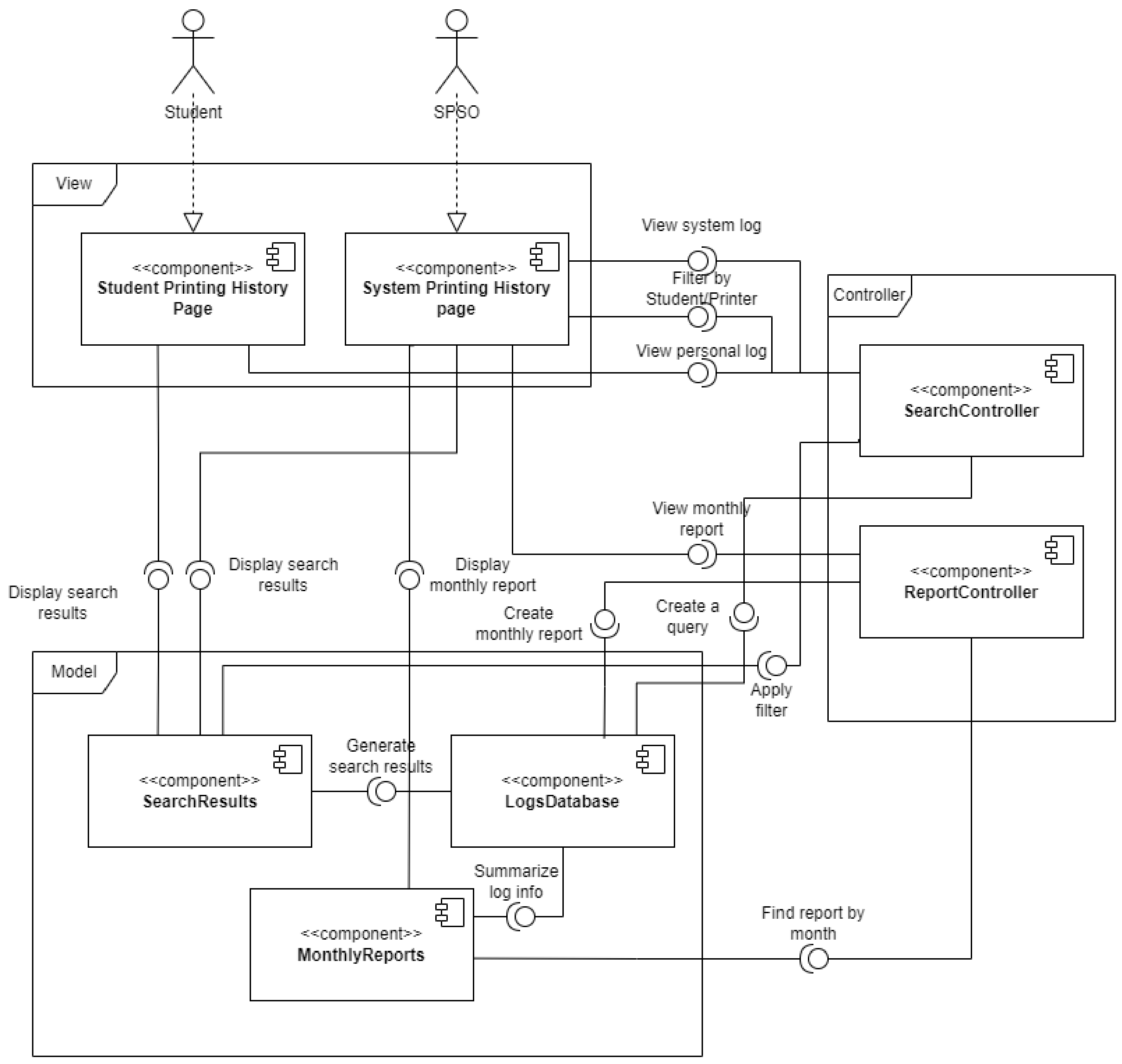
3.2.1 Printing:



Description:

Whenever a user access the printing screen, the Printing module is activated. The diagram follows MVC pattern involving 6 main components: The user will interact with the View including PrintingView component. The View component will be provided by the Controller with PrintingController component. The Controller will then request data from the Model with PrinterModel, ConfigModel and FileModel components. The model components will query the data from the database component.

3.2.2 View History:



3.2.3 Description:

- The component diagram follows the MVC model, with the Student Printing History Page and System Printing History Page components as the View element, the Controller element separated into two different components in SearchController and

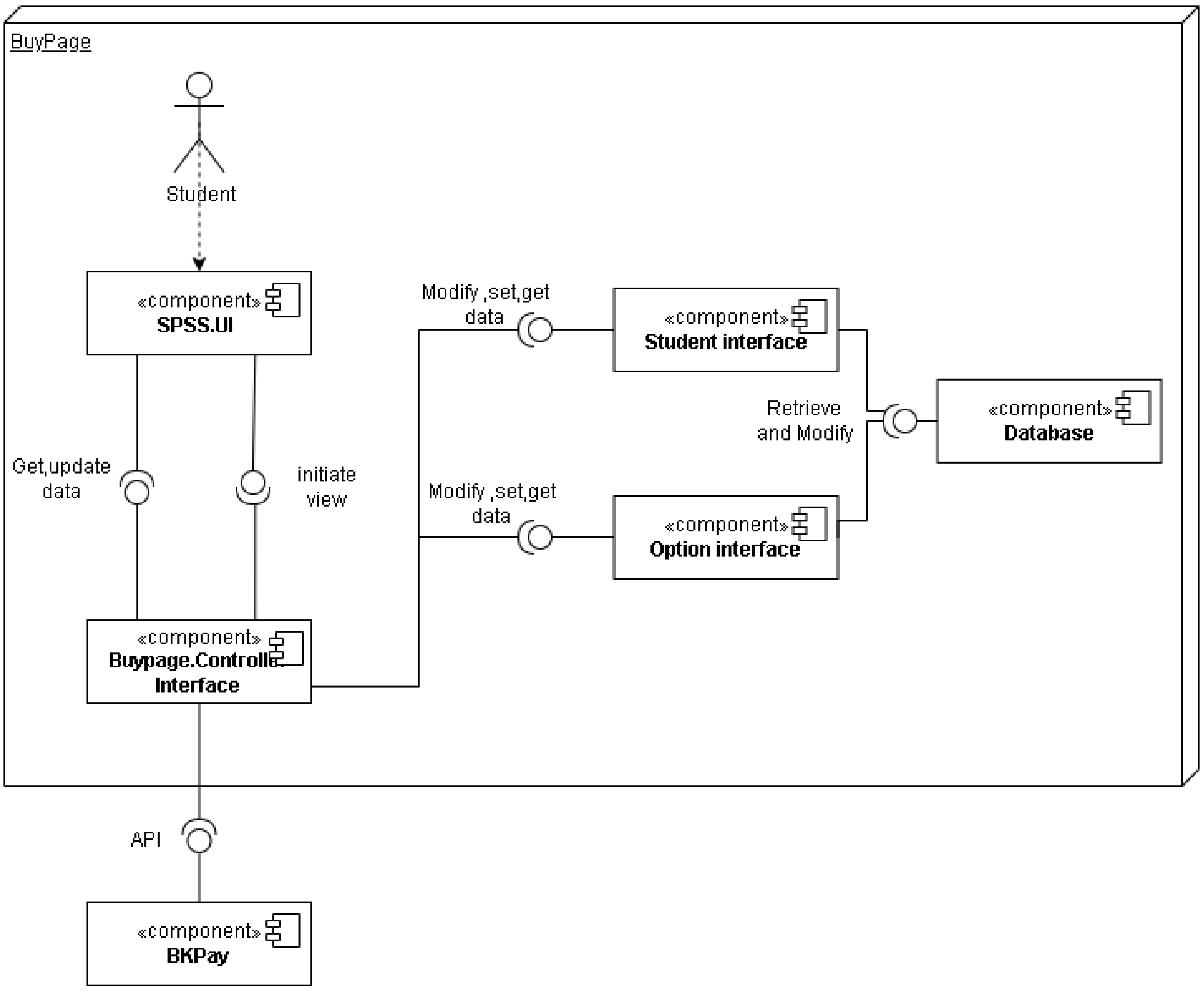
ReportController, and the LogsDatabase as the main component of the Model element.

- At the Student Printing History Page, student user can request to see their history by sending the time period. The SearchController will use that time period as well as the user's own StudentID to search for the requested logs by creating a query at the LogsDatabase. This will create a SearchResults which will be sent back to the user.

- At the System Printing History Page, the SPSO user can request to see the history of the entire system by also sending the time period. The only difference is that the SearchResults shows the whole system's logs instead of just one student. To view a specific student or printer's history in that time period, the System Printing History Page provides a filter which will be used to filter the SearchResults generated previously.

- The ReportController will automatically request the database to create a MonthlyReport at the end of each month. At the System Printing History Page, the SPSO user can request to see the report of a specific month. The ReportController will find the appropriate MonthlyReport and display it to the user.

3.2.4 Buy Pages:



Description:

- The user interact with the system via SPSS.UI Interface like hitting button, or choose option of number of pages.
- The SPSS.Controller Interface component provides SPSS.UI Interface methods to get data from Student Interface component and Option Interface component.
- The Database component provides methods to retrieve and modify data for Student interface component and Option interface component.
- The SPSS.UI Interface also let SPSS.Controller Interface initiate first view at student enter the system.

- BKPay will supply API to SPSS.UI Controller for hosting transaction between student and SPSS.Office

4 Implementation – Sprint 1

5 Implementation – Sprint 2