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

SOFTWARE ENGINEERING - CO3001

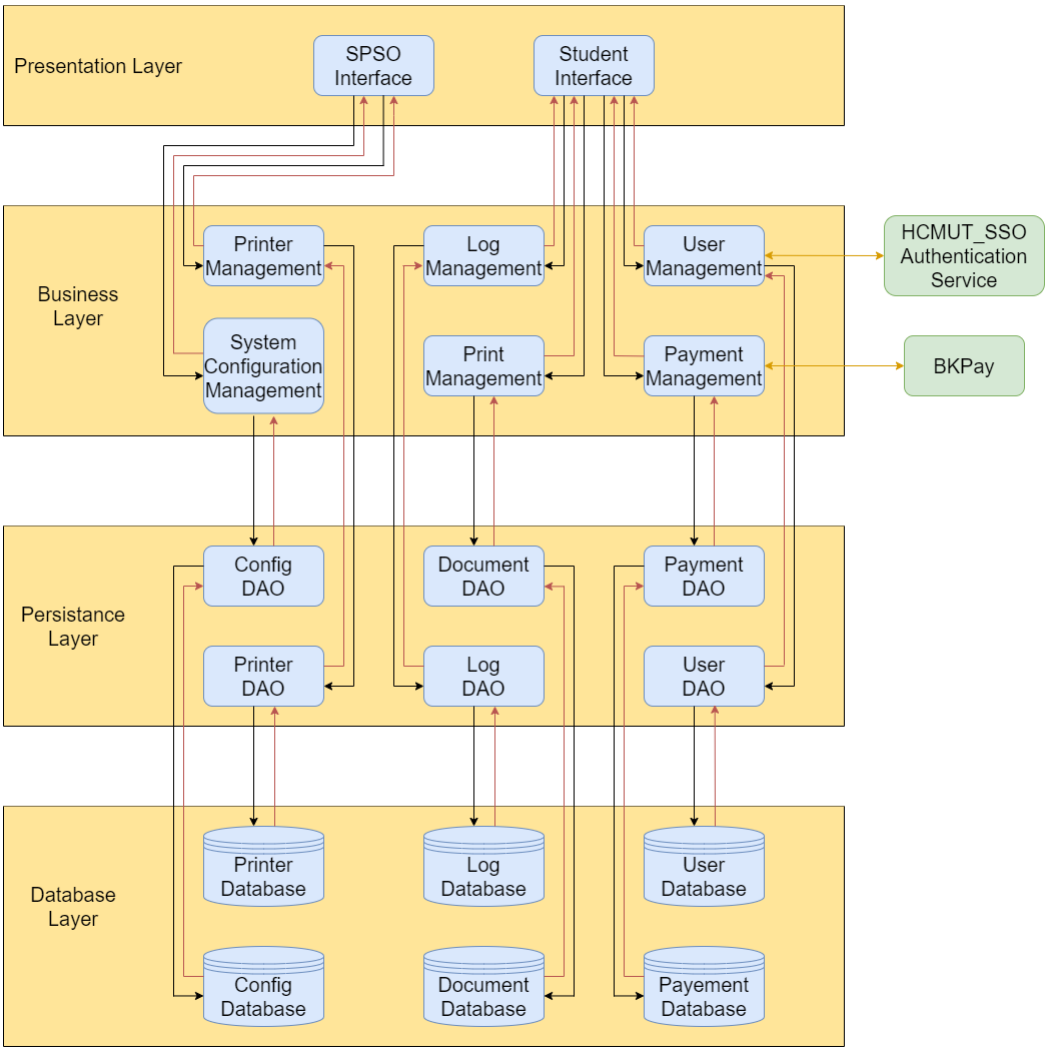
TASK 3

Group:	CC06 - 24	
Advisors:	Trương Thị Thái Minh	
Students:	Trần Gia Huy	2152600
	Lương Lê Long Vũ	2153980
	Vũ Minh Quân	2152926
	Phan Ngọc Khải	2154019
	Hoàng Đại Nam	2153594
	Nguyễn Trương Trọng Phúc	2152246
	Nguyễn Huỳnh An	2154018

HO CHI MINH CITY, October 2023

1 Task 3.1

1.1 Layer Architecture Design



1.2 Presentation Strategy

The presentation strategy for the HCMUT Student Smart Printing Service is centered around delivering a user-centric experience with two specialized interfaces tailored to the roles of students and Student Printing Service Officers (SPSOs). The Student Interface is streamlined for ease of use, enabling students to authenticate via HCMUT_SSO, upload documents, select printing options, and manage their printing jobs with minimal clicks. It also provides a straightforward pathway for purchasing additional print credits through BKPay, and a transparent view of their printing

history and remaining credits. For the SPSO, the interface is robust, providing comprehensive management tools for overseeing printer configurations, student print activities, system settings, and generating detailed reports. Both interfaces prioritize accessibility and functionality, ensuring that all interactions—from initiating a print job to configuring system-wide settings—are intuitive and efficient, fostering a productive environment for all users of the service.

1.3 Data Storage Approach

The data storage approach for the system is meticulously architected to ensure data integrity, security, and performance. At the core, the Database Layer is segmented into specialized databases for Printers, Documents, Users, Payments, and Configuration, each optimized for its specific type of data transactions and workloads. This separation allows for tailored backup strategies, security measures, and performance tuning. The Persistence Layer, comprising various DAOs, acts as an intermediary, providing a structured and standardized way of accessing and manipulating the data. These DAOs are responsible for abstracting the database complexities from the Business Layer, facilitating a decoupled architecture that enhances maintainability and scalability. With this setup, the system is well-equipped to handle concurrent transactions, maintain fast response times, and ensure that user data is consistently and reliably stored and retrieved.

1.4 API Management

API management within the HCMUT_SSPS is a critical component that facilitates secure and efficient interactions between the system's layers and external services. The APIs are designed to be restful, adhering to best practices in endpoint naming, versioning, and method use to ensure clarity and consistency for developers. Robust authentication and authorization mechanisms, powered by HCMUT_SSO, protect against unauthorized access and ensure that users can only interact with the system within their permission scopes. The API gateways serve as the traffic controllers, managing requests and responses between the Presentation and Business layers, as well as between the Business Layer and external payment gateways like BKPay. Throttling, caching, and rate-limiting are implemented to maintain system performance and prevent abuse. Furthermore, the APIs are meticulously documented and version-controlled, allowing for seamless integration with future services and facilitating easy updates and maintenance by the development team, ensuring long-term sustainability of the system's interoperability.

2 Task 3.2

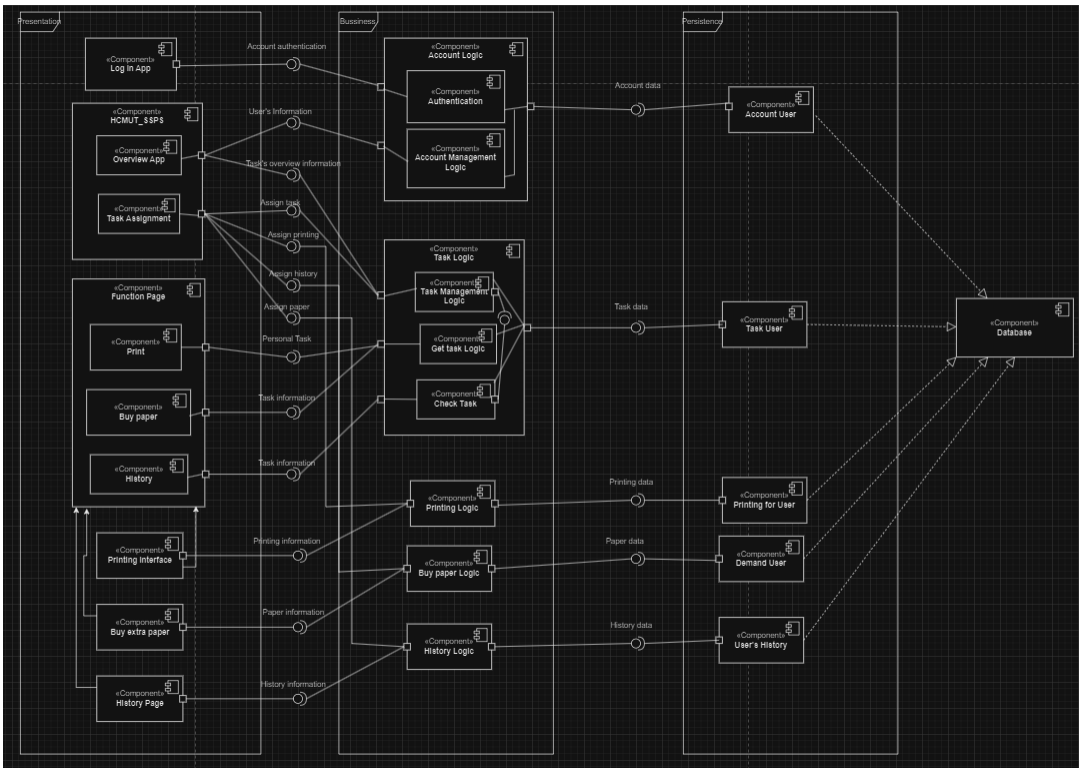
2.1 Theoretical basis: Component diagram

The physical components of object-oriented programming are modeled using UML component diagrams systems for defining, displaying, and recording component-based systems tems as well as for creating systems that can be executed using forward and reverse engineering smirking. In essence, component diagrams are class diagrams that highlight a system's elements that are frequently used to simulate a system's static implementation view. A component diagram divides the system that is being developed into different high performance standards. Each element is in charge of a single, distinct goal inside the whole system and only communicates with other necessary components when necessary. Some frequent concepts found in component diagrams are:

- **Component:** denotes a modular system component whose manifestation is interchangeable within its surroundings and embodies its contents. A component is represented in UML 2.x as a rectangle with optional compartments stacked vertically inside of it.
- **Interface:** two types
  - Required interface: shown as a socket (half-circle), this interface indicates one that the component needs.
  - Provided interface: a lollipop (full circle) that depicts an interface that the component offers.
- **Port:** a square positioned along the edge of the component rectangle, port is frequently used to assist in exposing a component's necessary and offered interfaces.
- **Relationship:** comprises linkages, dependence, association, composition, aggregation, and constraint. They have the same meaning and representation in the UML as those in the class diagram.

2.2 Component diagram of Task assignment module

2.2.1 Component diagram



2.2.2 Description and explanation

The component diagram is separated into four levels, each of which contains the following components, in accordance with the layered architecture that we selected for work 3.1:

1. **1. Presentation layer:**
    - **Log in App:** displays the log in App to Users with interface Account authentication from component Account Logic (internal component Authentication) to perform its functionality.
    - **HCMUT-SSPS:**
        - Overview App: uses interfaces to show the rear officer's overview user interface. Information about the employees was supplied by component Account Logic (an internal component of Account Management Logic), and information about the tasks themselves was supplied by component Task Logic (an internal component of Task Management Logic).
        - Task Assignment: displays the task assignment view for HCMUT-SSPS, requires interfaces Assign tasks, Assign 3 main function and route from components Task Logic (internal component Task management Logic), printing logic and buy paper Logic, history Logic respectively.
    - **Function Page:**
        - **Print:** Presents the printing interface to the user before starting to use it. This function helps output printouts according to the customer's needs through logical processing.
        - **Buy paper:** presents the user with an interface to purchase paper or buy additional paper (if needed) to facilitate easier printing through some payment logic (Using task information and check task from component Task logic)
        - **History:** Presents to users what they have printed as well as serving the need to reprint what they have used (collect information from users and through component check task)
2. **Business layer:** consists of 5 major components representing 5 modules of the Business layer we have discussed in task 3.1 which are Account Logic, Task Logic, Printing Logic, Buy paper Logic and History Logic, each of which requires the data object from its respective data mapper component to perform its business logic.
3. **Persistence layer:** consists of 5 data mappers which are Account User, Task Mapper, printing for users, Demand User, User's History, all of which establish a connection to component Database in order to retrieve and perform query on the data.
4. **Data layer:** consists of Database providing data to each user in the Persistencelayer.