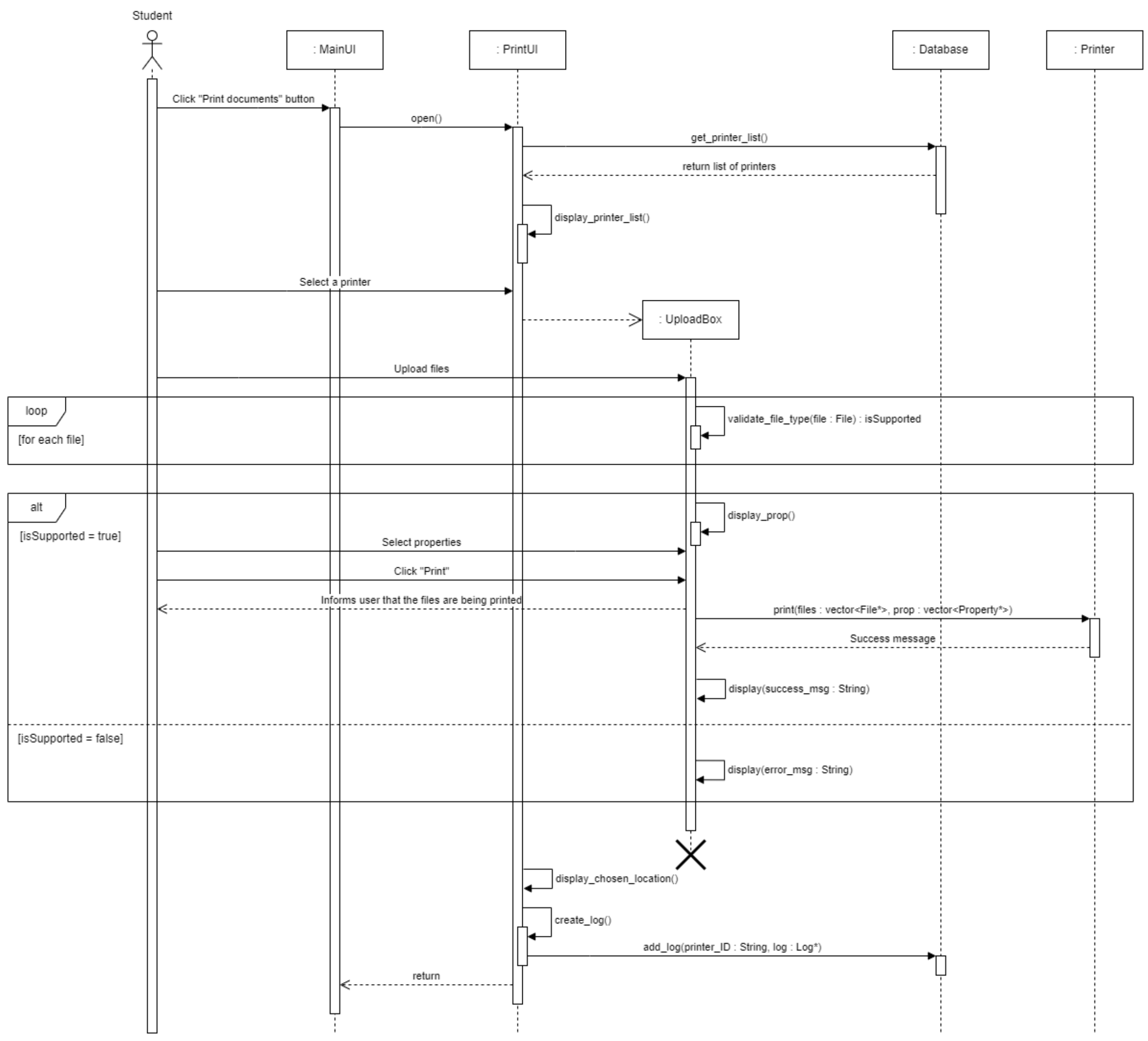
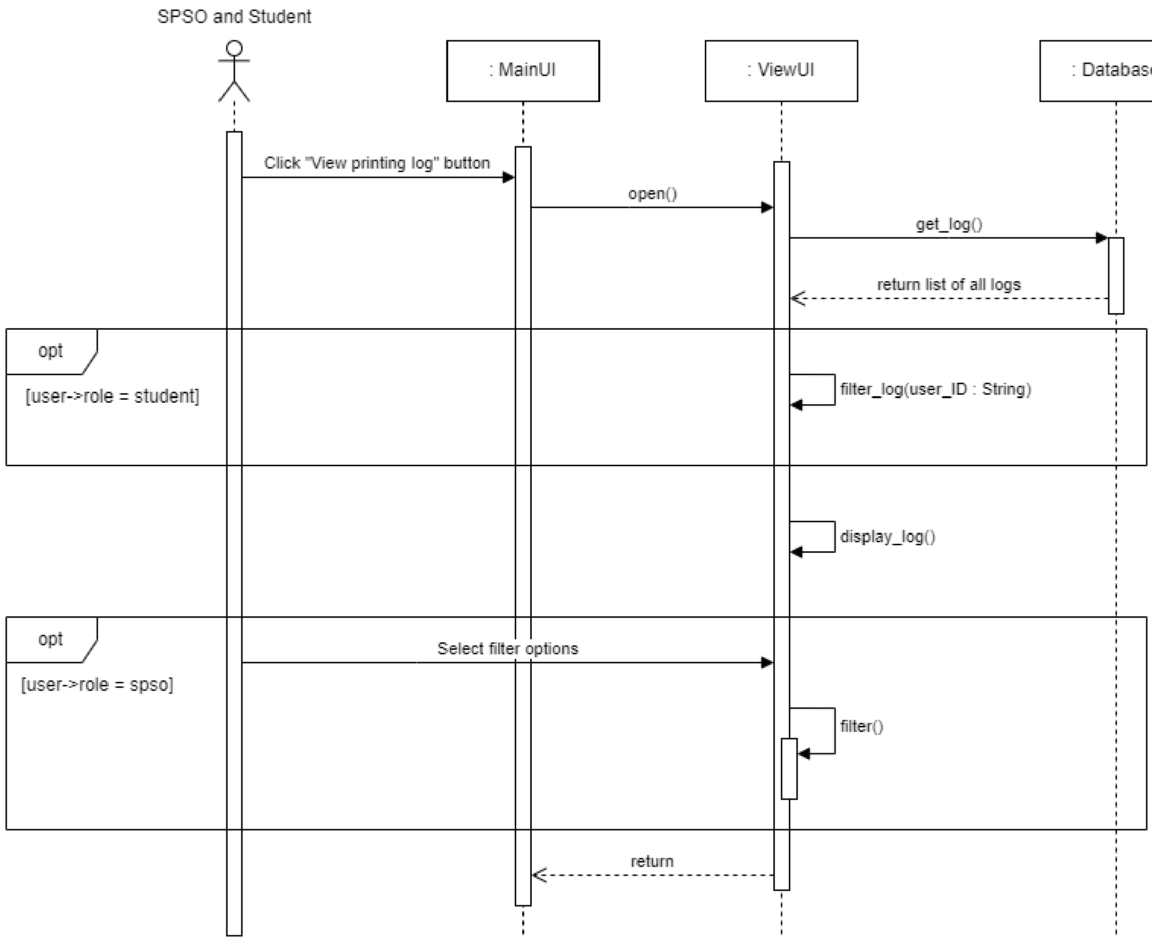
2.2. Sequence diagrams

1. Use-case: Print documents


Student
: MainUI
: PrintUI
: Database
: Printer
Click "Print documents" button
open()
get_printer_list()
return list of printers
display_printer_list()
Select a printer
: UploadBox
Upload files
loop
[for each file]
validate_file_type(file : File) : isSupported
alt
[isSupported = true]
display_prop()
Select properties
Click "Print"
Informs user that the files are being printed
print(files : vector<File*>, prop : vector<Property*>)
Success message
display(success_msg : String)
[isSupported = false]
display(error_msg : String)
display_chosen_location()
create_log()
add_log(printer_ID : String, log : Log*)
return

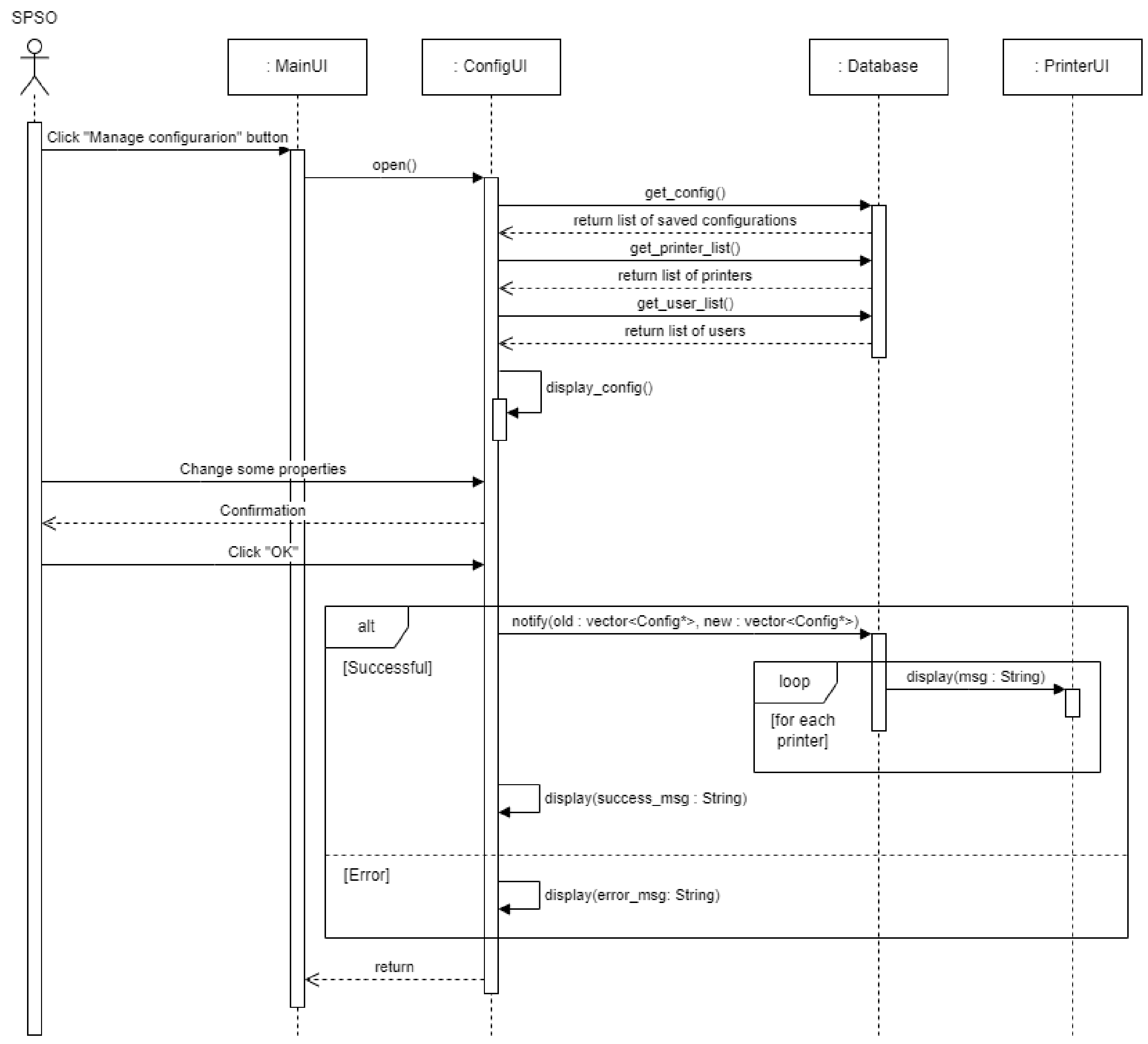
In this use-case, the actor is the student. Actor accesses the print user interface by clicking the "Print documents" button on the main interface, which directs the actor to the corresponding interface. The print interface then requests the database for the list of available printers, and displays it on screen. The actor then selects a printer, and a box for uploading (UploadBox) will be created, which allows the actor to upload some files. Iterative actions are then performed on each file. The file's type is checked, and if any file's type is unsupported, display an error message; otherwise, the print properties are displayed, and the actor is free to change some. When the actor is done and clicks the "Print" button, the interface will notify the actor, and a print signal will be sent to the chosen printer, and a return signal will be sent back, indicating successful message passing. When all files have been handled, the upload box closes, and the screen displays the locations of the chosen printers, and attempts to log the print information in the database.

2. Use-case: View printing log



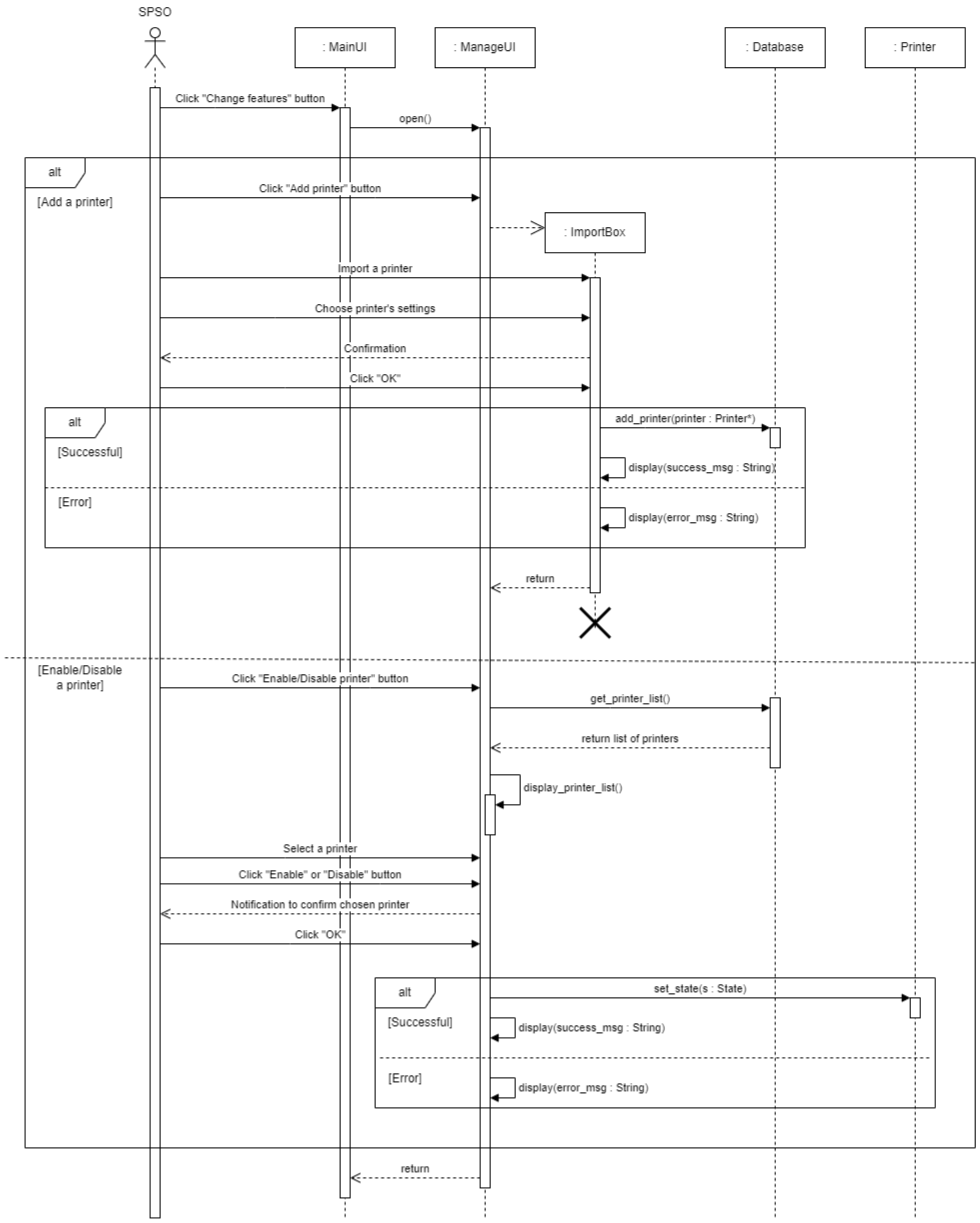
In this use-case, the actor can be the SPSO or the student. Actor accesses the view user interface by clicking the "View printing log" button on the main interface, which directs the actor to the corresponding interface. The view interface then requests the database for the list of logs. If the actor's role is a student, the retrieved list of logs will be filtered to exclude logs from other users. If the actor's a SPSO, they can optionally filter the logs according to several criteria. The final list of logs after this step will be displayed on screen.

3. Use-case: Manage configuration



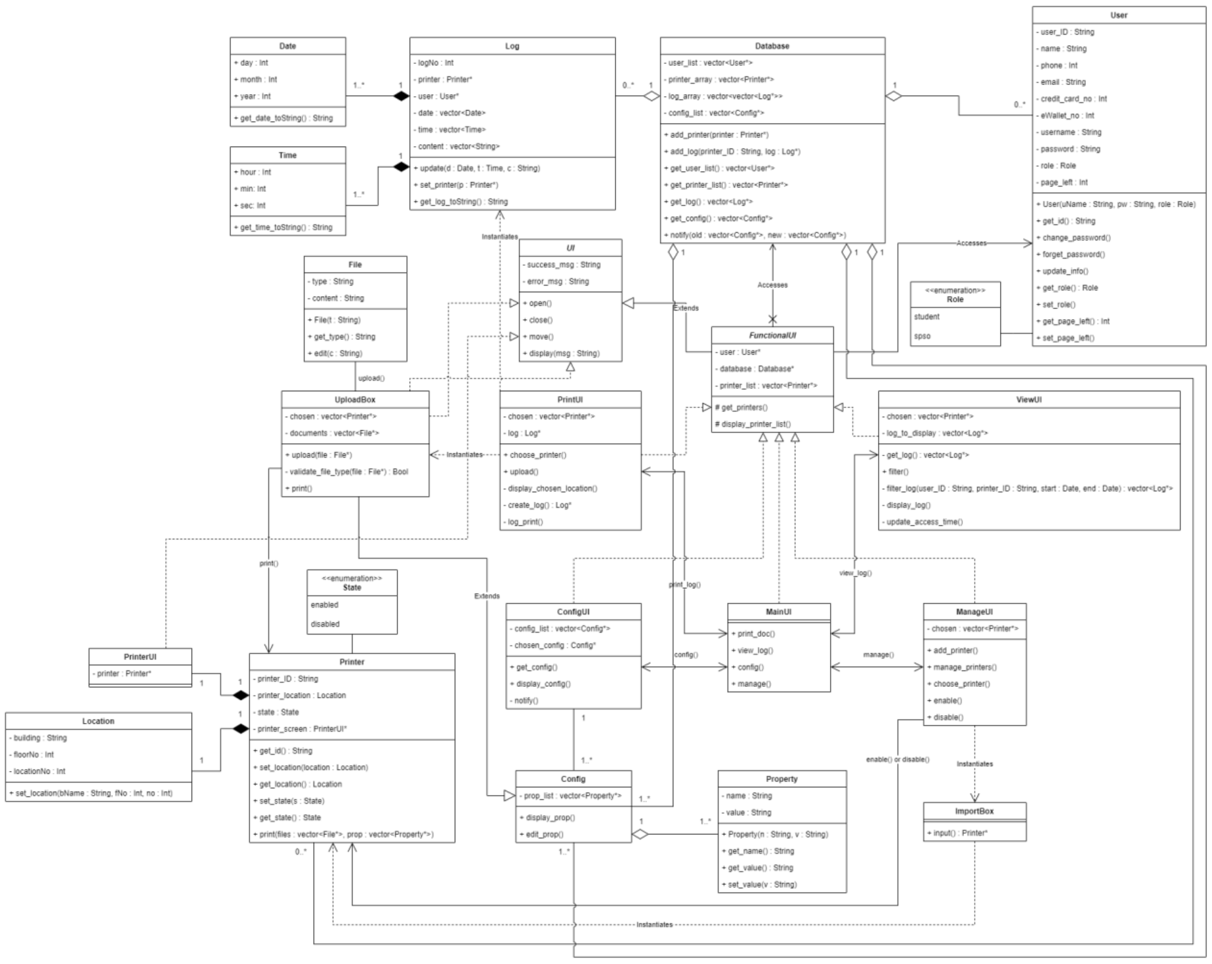
In this use-case, the actor is the SPSO. Actor accesses the configuration user interface by clicking the "Manage configuration" button on the main interface, which directs the actor to the corresponding interface. The configuration interface then requests the database for the list of saved settings the list of printers, and the list of all users, and displays the configurations on screen. Each configuration has several properties, which can be changed by the user. After that, the system will ask for save confirmation, and if confirmed and successful, it will update to the database and notify every printer to display the changes on screen, and display a success message. If unsuccessful, display an error message instead.

4. Use-case: Manage printer features


SPSO
: MainUI
: ManageUI
: Database
: Printer
Click "Change features" button
open()
alt
[Add a printer]
Click "Add printer" button
: ImportBox
Import a printer
Choose printer's settings
Confirmation
Click "OK"
alt
[Successful]
add_printer(printer : Printer*)
display(success_msg : String)
[Error]
display(error_msg : String)
return
[Enable/Disable a printer]
Click "Enable/Disable printer" button
get_printer_list()
return list of printers
display_printer_list()
Select a printer
Click "Enable" or "Disable" button
Notification to confirm chosen printer
Click "OK"
alt
set_state(s : State)
[Successful]
display(success_msg : String)
[Error]
display(error_msg : String)
return

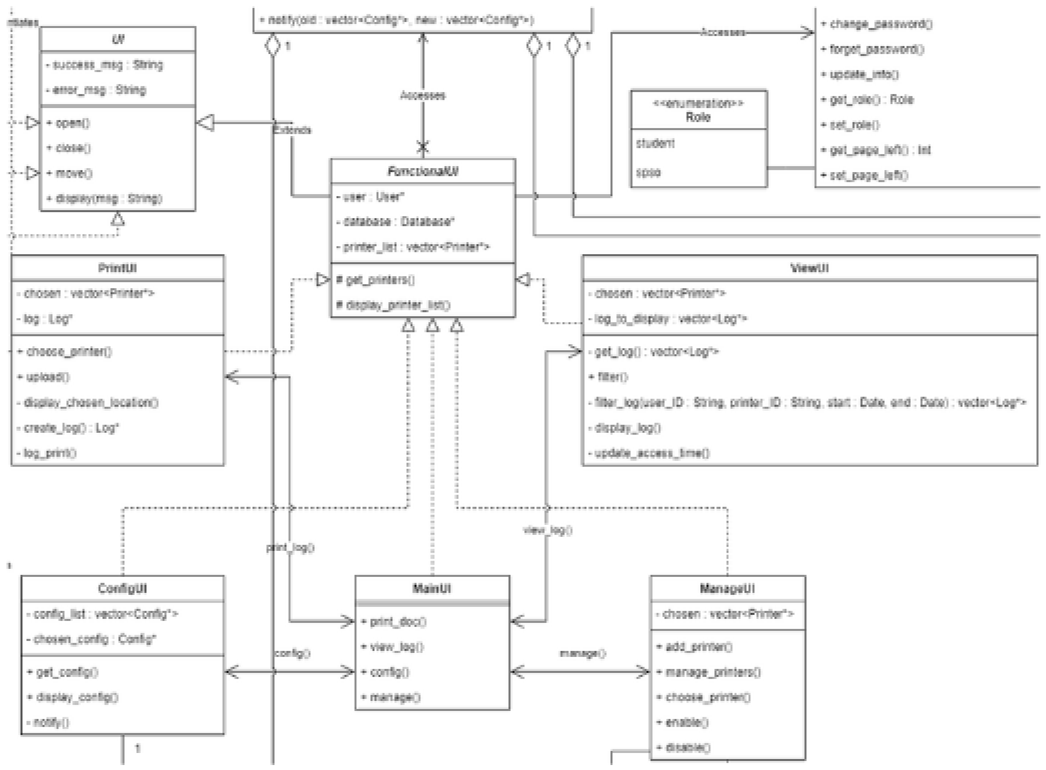
In this use-case, the actor is the SPSO. Actor accesses the manage user interface by clicking the “Change features” button on the main interface, which directs the actor to the corresponding interface. In this interface, the actor can choose between 2 functions: add a printer or enable/disable a printer. If they choose to add a printer, a box to import printer (ImportBox) will be created. In this box, the actor can set up a new printer, input their properties, change their settings. When confirmed, if successful, the newly created printer will then be added to the database, and a success message will be displayed; otherwise, an error message will be displayed. After that, the box is destroyed, and back to the manage interface. If the actor chooses to enable/disable a printer, the UI requests the database for the list of available printers, and displays it on screen. The actor can select a printer, and click the "Enable" or "Disable" button after selecting. When confirmed and successful, the UI will send a signal to the chosen printer and enable or disable them accordingly to the selected option, and display a success message. Otherwise, for some reason, if the system fails to do so, an error message will be displayed. After having done the operations, the actor can go back to the main interface.

2.3. Class diagram



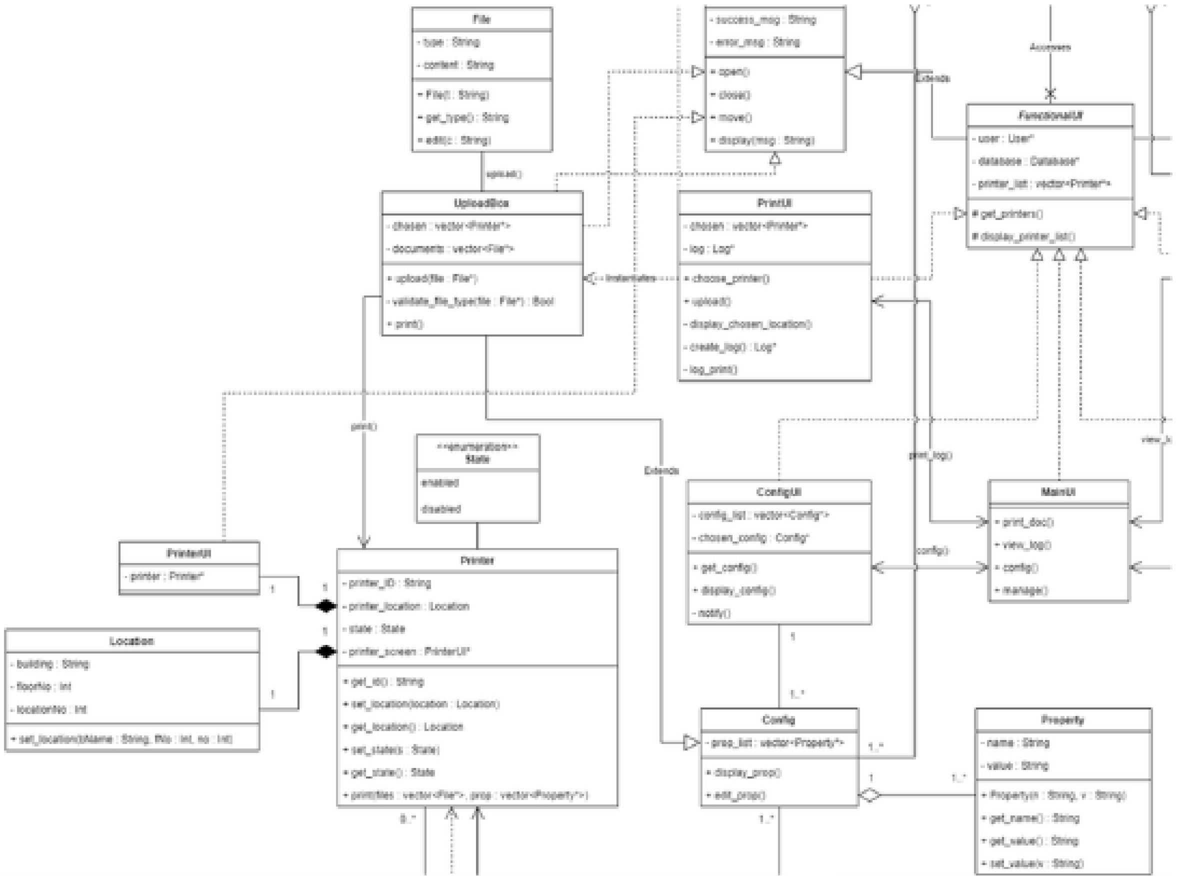
In this diagram, we have abstract classes for implementing Views: *UI* (for general-purpose UI and textboxes) and *FunctionalUI* (for querying data from Database) extending *UI*.

Besides, we also have 5 View classes that implement the abstract class *FunctionalUI*, including 4 classes for 4 different functionalities of the module (*PrintUI*, *ViewUI*, *ConfigUI*, *ManageUI*) and 1 class for the main user interface (*MainUI*).



Within each functionality, we have various view classes that represents different elements of the user interface, and entity classes that correspond to different entities in the database.

1. Print and Config classes

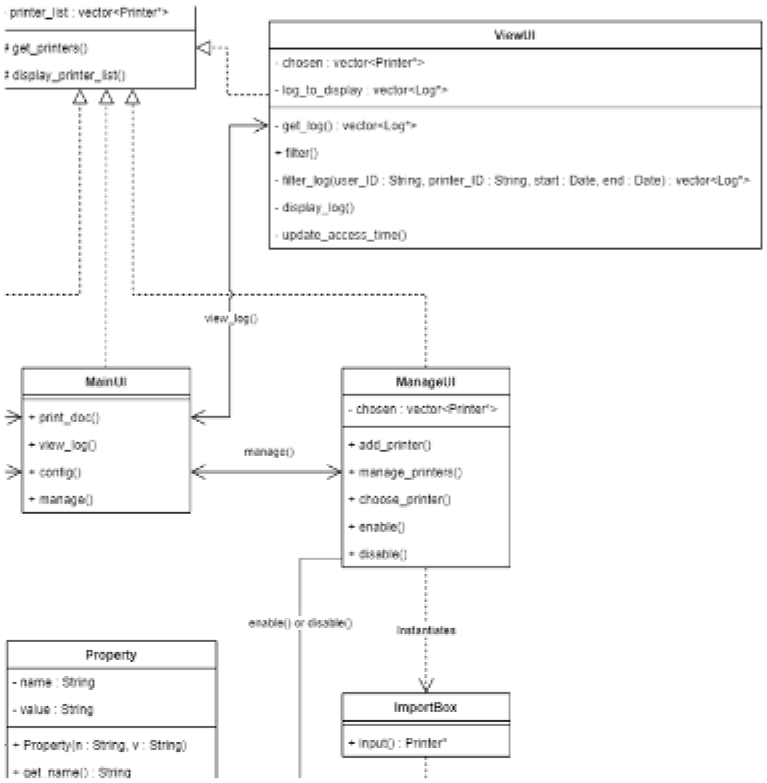


The main View classes are *PrintUI*, *ConfigUI*, which are the menu of print and manage configurations. The subsidiary View classes include *UploadBox* (the box for uploading documents) and *PrinterUI* (the logical screen of each printer). The entity classes include *Printer*, *Location*, *File*, *Config*, *Property*. *Printer* has the information of each printer and handles the print actions, while *Config* is a setting, which may include multiple fields and values, with each field is a *Property*.

When print action is performed, the *PrintUI* object will create an *UploadBox* object. A *File* object is then uploaded to *UploadBox*. *UploadBox* will display a list of Property, which user can change, and then call the print method in *Printer* and pass the *File* and *Property* to it.

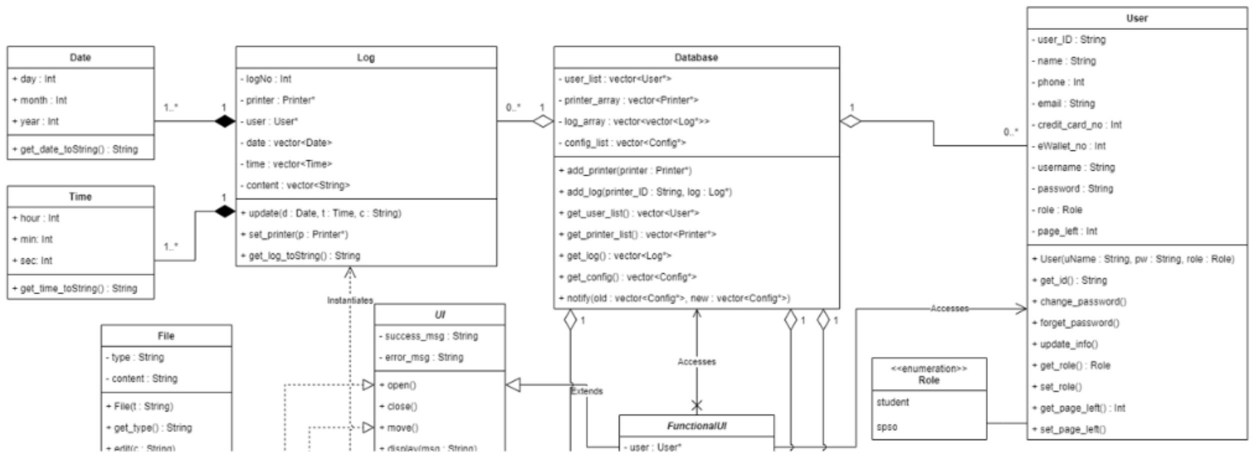
When manage configurations action is performed, the *ConfigUI* object will call *display_config* to display its list of *Config*. Each *Config* then calls its *display_prop* to display its list of *Property* and edit the *Property* values.

2. **View and Manage classes**



The main View classes are *ViewUI*, *ManageUI*, which are the menu of view logs and manage printers. The subsidiary View classes include *ImportBox* (the box for importing a printer). The *ViewUI* object stores a list of logs fetched from database to display whenever view logs action is performed. *ManageUI* has some method to add, choose, enable or disable printers (it changes the state of a printer to "enabled" or "disabled". It interacts with the entity class *Printer* to perform those actions.

3. Database classes



These are the central entity classes of the system. The central *Database* class has the list of *User*, *Printer*, *Log* and *Config*, and has some methods to add a printer, add a log, return the data, and notify the printers about the changed configurations. The *User* object stores all information of each user, for both student and SPSO, differentiated by the *Role* enumeration type. The *Log* object stores the print details, including the user, the used printer, date and time, and the printed content.