PROJECT REPORT
Wholesale Management System

Course: Software Engineering IT091IU

Luu Minh Long, Nguyen Nhat Minh, Vu Minh Huy
ITITIU18079, ITITIU18086, ITITIU17007
International University - VNUHCMC

Contents

1 Project Overview **4**
- 1.1 Group members . . . 4
- 1.2 Course expected outcomes . . . 4
- 1.3 Topic . . . 4
- 1.4 Timeline . . . 5
- 1.5 Requirements . . . 5
- 1.6 Project breakdown . . . 5
- 1.7 Resources . . . 6

2 Project Analysis **7**
- 2.1 Requirements Analysis . . . 7
  - 2.1.1 Requirements Analysis . . . 7
  - 2.1.2 Software and Hardware decision . . . 8
  - 2.1.3 Risk Management . . . 8
- 2.2 Project Schedule . . . 9
- 2.3 Use cases . . . 10
- 2.4 Entity-Relationship diagram . . . 14
- 2.5 Sequence Diagram . . . 15
- 2.6 Test cases . . . 21
- 2.7 Dummy data . . . 23

3 A Web app **24**
- 3.1 Background . . . 24
- 3.2 Analysis . . . 24
- 3.3 Constraints . . . 24
- 3.4 Libraries . . . 25

4 Installation and Run **26**
- 4.1 Installation . . . 26
  - 4.1.1 Run on localhost . . . 26
  - 4.1.2 Run on a web hosting service . . . 26
- 4.2 How to run . . . 26

5 Conclusion **28**
- 5.1 Lessons . . . 28
- 5.2 Limitations . . . 28

List of Figures

2.1 Gantt chart of the project 9
2.2 Project use case diagram 10
2.3 Entity-Relationship Diagram 14
2.4 Sequence Diagram of use case 1 15
2.5 Sequence Diagram of use case 2 15
2.6 Sequence Diagram of use case 3 16
2.7 Sequence Diagram of use case 4 17
2.8 Sequence Diagram of use case 5 18
2.9 Sequence Diagram of use case 6 19
2.10 Sequence Diagram of use case 7 20

4.1 Main page 27

List of Tables

2.1 Original requirements analysis 7
2.2 Requirements to be implemented 8
2.3 Risk management 9
2.4 Use case 1 10
2.5 Use case 2 11
2.6 Use case 3 11
2.7 Use case 4 11
2.8 Use case 5 12
2.9 Use case 6 13
2.10 Use case 7 13

Chapter 1

Project Overview

This report contains the information about the project report of Course "Software Engineering" IT076IU at International University, Vietnam National University – Ho Chi Minh City.

1.1 Group members

Our group consists of three members:

- Luu Minh Long (leader) - ITITIU18079
- Nguyen Nhat Minh - ITITIU18086
- Vu Minh Huy - ITITIU17007

1.2 Course expected outcomes

The learning outcomes of the course are as follows:

- Apply the principles and methods of software engineering in practice
- Apply critical and analytic thinking to the planning of the software development process
- Apply critical and analytic thinking to the execution and evaluation of the software development process

We therefore are required to do a project based on the provided topics.

1.3 Topic

Topic: "WHOLESALE MANAGEMENT SYSTEM".

Background: In the booming age of digital commercials, having a software to keep track of the sales is mandatory. Every company needs to make a software that helps companies and independent sellers to maintain the details of important information and activities, such as customer's details, buyer's details, sales and transactions, etc.

Seeing that this topic may help us understand online shopping and digital commercials, after discussion, we decided to choose this to be our topic.

1.4 Timeline

The project started on October 5th, 2020 and is expected to end on December 13th, 2020.

1.5 Requirements

The original requirements of the projects are as follows:

> Design a database to maintain information about stock, buyers and customers satisfying following properties:
>
> 1. Maintain the details of stock like their id, name, quantity
> 2. Maintain the details of buyers from which manager has to buy stock like buyer id, name, address, stock id to be bought
> 3. Details of customers i.e. name, address, id
> 4. Default list of customers who has not paid their pending amount
> 5. List of payment paid or pending
> 6. Stock that is to be buy if quantity goes less than a particular amount.
> 7. Profit calculation for a month
> 8. Quantity cannot be sold to a customer if required amount is not present in stock and date of delivery should be maintained up to which stock can be provided.

To avoid confusion, we decide to change **buyer** to **shop**, which indicates where the company imports the good from.

1.6 Project breakdown

In this report, we break it into two parts.

In the first part, we do requirement analysis, project schedule, Entity-Relationship diagram (ERD), Object-oriented structures, etc.

In the second part, we explain our decision to switch it to Web app, upgrade the project with advanced libraries, and host it online 24/7 rather than local host.

We then conclude the project: what we learn, what we hope to learn more, what goes as expected and what does not.

1.7 Resources

We are a group of 3 people: two third year students and one fourth year student. We have our personal laptops/PCs.

We have limited experience and knowledge in making a real and scalable big applications. Although we do this project as best as we can, errors and limitations are unavoidable.

Chapter 2

Project Analysis

Phase 1 is our process from the beginning (October 5th, 2020) to October 23rd, 2020, where we decide to make big changes.

2.1 Requirements Analysis

2.1.1 Requirements Analysis

We analyze and conclude that the functional and non-functional requirements are as follows:

ID	Name	Description	Type
1	Authentication	The system requires a password to be used	Functional
2	Store data	Store data about customer, shop, transaction, etc.	Functional
3	Search function	Search for primary key and name attribute of entities	Functional
4	Paid/Pending search	Search for customers and transactions that are paid or pending	Functional
5	Stock amount warning	Warn if a stock's amount goes below a certain amount	Functional
6	Profit calculation	Calculate the profit in a week/month/year	Functional
7	Making new transaction	Also send a notification whenever a transaction is not possible	Functional
9	Maintain date of delivery	If transaction is not possible, keep delivery date the same as import date	Functional
10	Adding new entities	Add a new item/shop/etc.	Functional
11	Usability	User-friendly GUI	Non-Functional
12	Security	Password must be encrypted	Non-Functional
13	Interactive plot	Plot should be interactive and customizable	Non-Functional

Table 2.1: Original requirements analysis

We decide to exclude and modify some of the requirements:

1. ID 5: Stock amount warning. This function requires synchronization between many functions, which is beyond our ability.

2. ID 7: Making new transaction. The description "Also send a notification whenever a transaction is not possible": due to limited resources, we are not able to do this warning function.

3. ID 9: Maintain date of delivery. We think this one is not necessary. In online shopping websites, if a stock is depleted, they simply warn "Out of stock, please check back later".

Because of these reasons, we have to limit the requirements down:

ID	Name	Description	Type
1	Authentication	The system requires a password to be used	Functional
2	Store data	Store data about customer, shop, transaction, etc.	Functional
3	Search function	Search for primary key and name attribute of entities	Functional
4	Paid/Pending search	Search for customers and transactions that are paid or pending	Functional
5	Profit calculation	Calculate the profit in a week/month/year	Functional
6	Adding new entities	Add a new item/buyer/etc.	Functional
7	Usability	User-friendly GUI	Non-Functional
8	Security	Password must be encrypted	Non-Functional
9	Interactive plot	Plot should be interactive and customizable	Non-Functional

Table 2.2: Requirements to be implemented

The software will come with only **ONE** default account, and the number of accounts will not be increased under any circumstances. Nonetheless, the software can still be used by many users, since it will be implemented with session state, with one user per session.

2.1.2 Software and Hardware decision

After analyzing the requirements, we decide to choose our software and hardware to implement the project:

- It will be a software app.
- Language: Python 3.8. Python is a general purpose language: it has a lot of libraries and a big community.
- Version control: Git, GitHub. Git is a very popular Version control system, and it syncs with GitHub.
- Database: SQLite3. By default, SQLite3 comes bundled with Python standard library, and it is very light comparing to other database management system.

To avoid conflicts when implementing the app, we decide to make a coding style. The style can be view online using the link below:
https://github.com/minhlong94/SWE_IT076IU/blob/master/coding_style.md

Further details about it will be mentioned in phase 2.

2.1.3 Risk Management

We analyze the potential risk, and come up with the table below, with possible solutions.

ID	Category	Title	Affect	Probability	Impact	Response plan
1	HR	Members lack communication	Slow down the progress of the project	Medium	Medium	-Each member independently grades his own and others' contribution. -The project will not be divided into parts but milestones for all members to work with at the same time.
2	HR	Members lack experience	Slow down the project, produce unreliable result	Medium	Medium	-Quickly learn what is necessary for the project Always improve knowledge about that part
3	Scope	Project may require additional requirements	Slow down progress, deadline	Medium	Medium	Hold a meeting to update schedule and plan
4	Hardware	Members lack hardware resources	Limit the scale of the project May not run some libraries Out-of-memory error can occur	High	High	Find workaround

Table 2.3: Risk management

2.2 Project Schedule

We break down the project schedule, with the corresponding Gantt chart:

- First week (4/10/2020 - 11/10/2020): complete Object-oriented classes, Entity-Relationship Diagram and Project structure
- Second week (11/10/2020 - 18/10/2020): implement work from the first week
- Third week to Seventh week: (18/10/2020 - 22/11/2020): Graphics User Interface and API
- Eighth week to Ninth week: (22/11/2020 to 6/12/2020): testing and debugging
- Last (one/two) week (till 13/12/2020): Testing, debugging and writing report

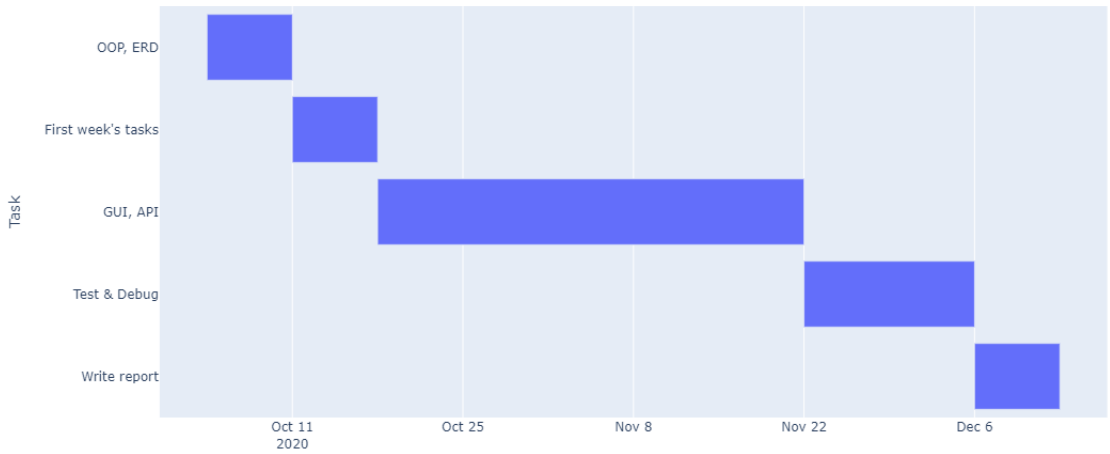

Figure 2.1: Gantt chart of the project

2.3 Use cases

Careful analysis and with real life experience, we identify these use cases:

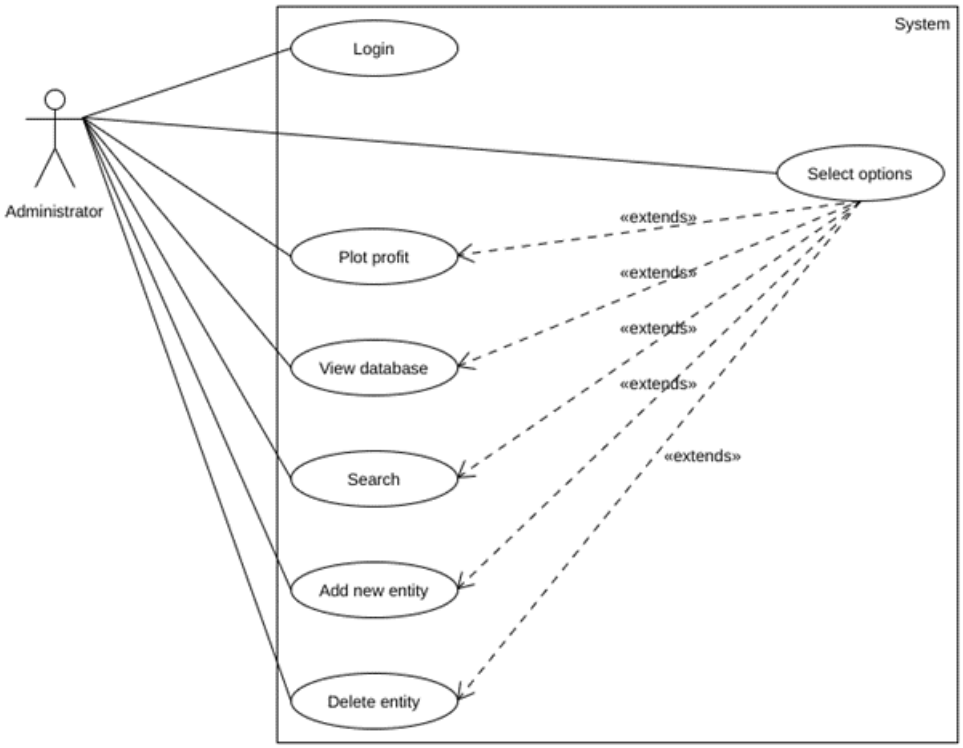

Figure 2.2: Project use case diagram

Use case 1: Login to the system
Identifier: UC1
Input: admin's password
Output:

1. If successful, redirect to the home page
2. If fail, shows a warning message

Actor	System
1. Open the login page	1.1. Display the login page
2. Enter password	
3. Submit password	3.1. Check password 3.2. If match, show success message and direct to UC2 3.3. If failed, return the login page and ask the user to enter again.

Table 2.4: Use case 1

Precondition: Logged in
Postcondition: None

Use case 2: Select menu's options

Identifier: UC2
Input: None
Output: Direct to the corresponding option page (tab) from the select box.

Actor	System
	1. Display selectbox
2. User select	2.1 Direct user to the corresponding option

Table 2.5: Use case 2

Select box includes these options: View database (as Table) (UC3), View profit plot (UC4), Add new entity (UC5), Delete entity (UC6), Search for entity (UC7)
Precondition: Logged in
Postcondition: None

Use case 3: View database (as table)
Identifier: UC3
Input: None
Output: Display the selected database table

Actor	System
	1. System displays list of available database tables
2. User selects a specific table	2.1. Display the selected database table

Table 2.6: Use case 3

Precondition: Logged in
Postcondition: None

Use case 4: Plot profit of a shop during a time period
Identifier: UC4
Input:

1. Start date
2. End date
3. Shop ID (can be multiple)

Output: Line chart of the shop's profit during the time period

Actor	System
	1. Display inputs (Shop ID, Start date and End date)
2. User inputs Shop ID, Start date, End date	
3. Submit	3.1 Check input condition (End date Start date, Shop ID exists) 3.2 If successful, show a line chart of the shop's profit during the time period. 3.3 If nothing returned, show warning

Table 2.7: Use case 4

Precondition:

- Logged in
- End date $\geqslant$ Start date
- Shop ID exists in the database

Postcondition:
If the time period is within two months, plot profit by weeks.
If the time period is within a year, plot profit by months.

Use case 5: Add new entity
Identifier: UC5
Input:

1. Select a database table
2. The corresponding object's attributes

Output:

1. If successful, shows a success message.
2. If input consists of restricted NaN values, shows a warning message, terminates input process, returns to input screen.

Actor	System
1. Open entity adding page	1.1. Display entity adding page
2. Select a database table	2.1. Display table's columns
3. Input table's columns	
4. Submit	4.1. Check input 4.2. If success, show success message 4.3. If error (input contains all NaNs), show a warning message and terminate process

Table 2.8: Use case 5

Precondition: Logged in
Postcondition: None

Use case 6: Delete entity
Identifier: UC6 Input:

1. Select a table
2. Input entity's key (ID or name)

Output:

1. If successful, shows success message.
2. If failed (key not exists), shows a warning message.

Actor	System
1. Open entity deleting page	1.1. Display entity deleting page
2. Select a database table	2.1. Display entity input attributes (ID or name)
3. Input ID or name	
4. Submit	4.1. Check input 4.2. If success, show success message 4.3. If input error (ID or name does not exist), show a warning message and terminate process

Table 2.9: Use case 6

Precondition:

- Logged in
- ID or name exists in the database

Postcondition: None

Use case 7: Search for an entity from a table
Identifier: UC7
Input:

1. Select a database table
2. Entity ID or name

Output: the corresponding table but displays only the data of the input's ID or name only

Actor	System
1. Open entity searching page	1.1. Display entity searching page
2. Select a database table from the selectbox	
3. Input ID or name	
4. Submit	4.1 Check id/name 4.2 If success, return table that has the ID or name only 4.3 If error (ID or name does not exist), show a warning message and terminate process

Table 2.10: Use case 7

Precondition:

- Logged in
- ID or name exists

Postcondition: None

2.4 Entity-Relationship diagram

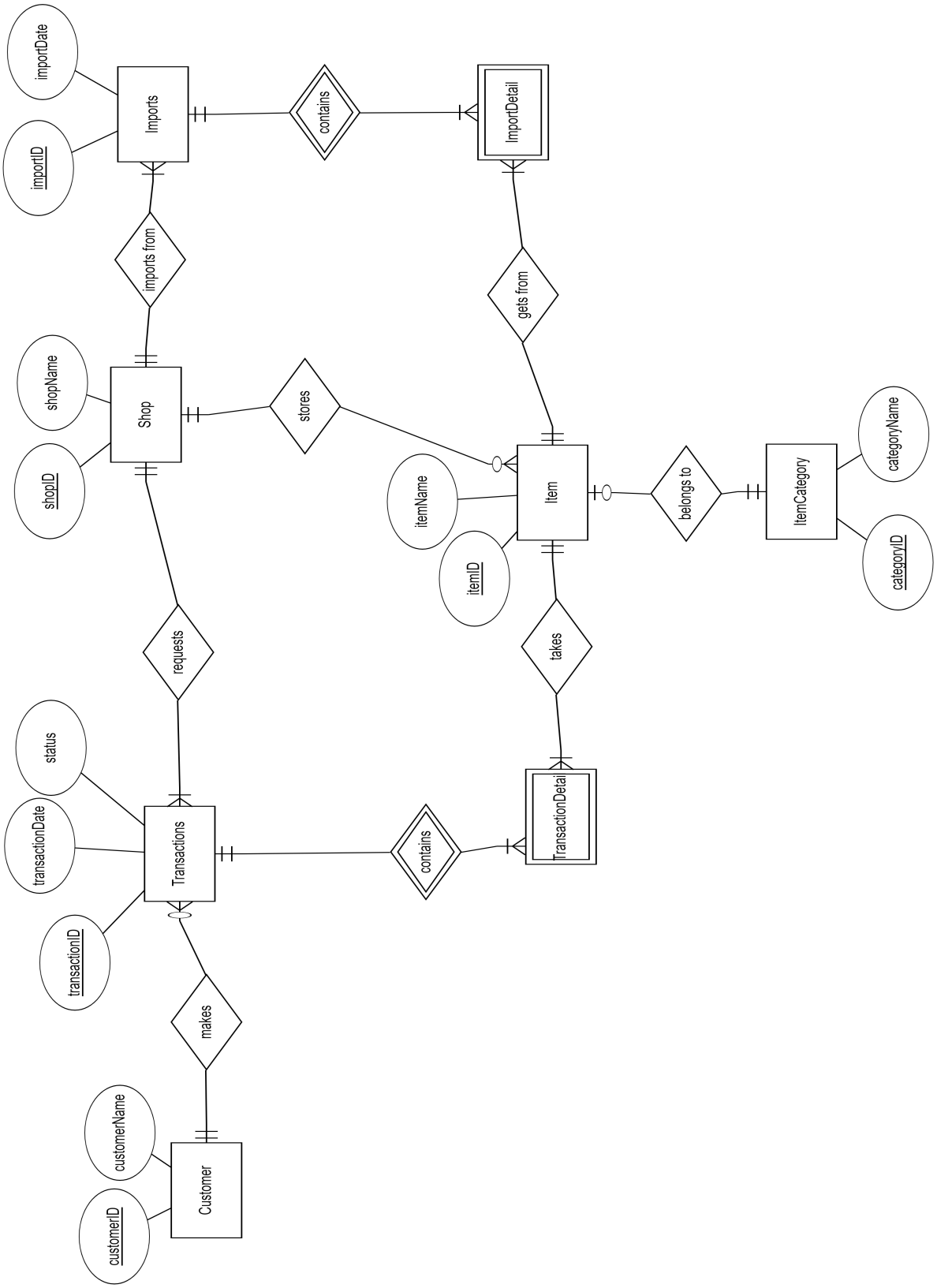

Figure 2.3: Entity-Relationship Diagram

2.5 Sequence Diagram

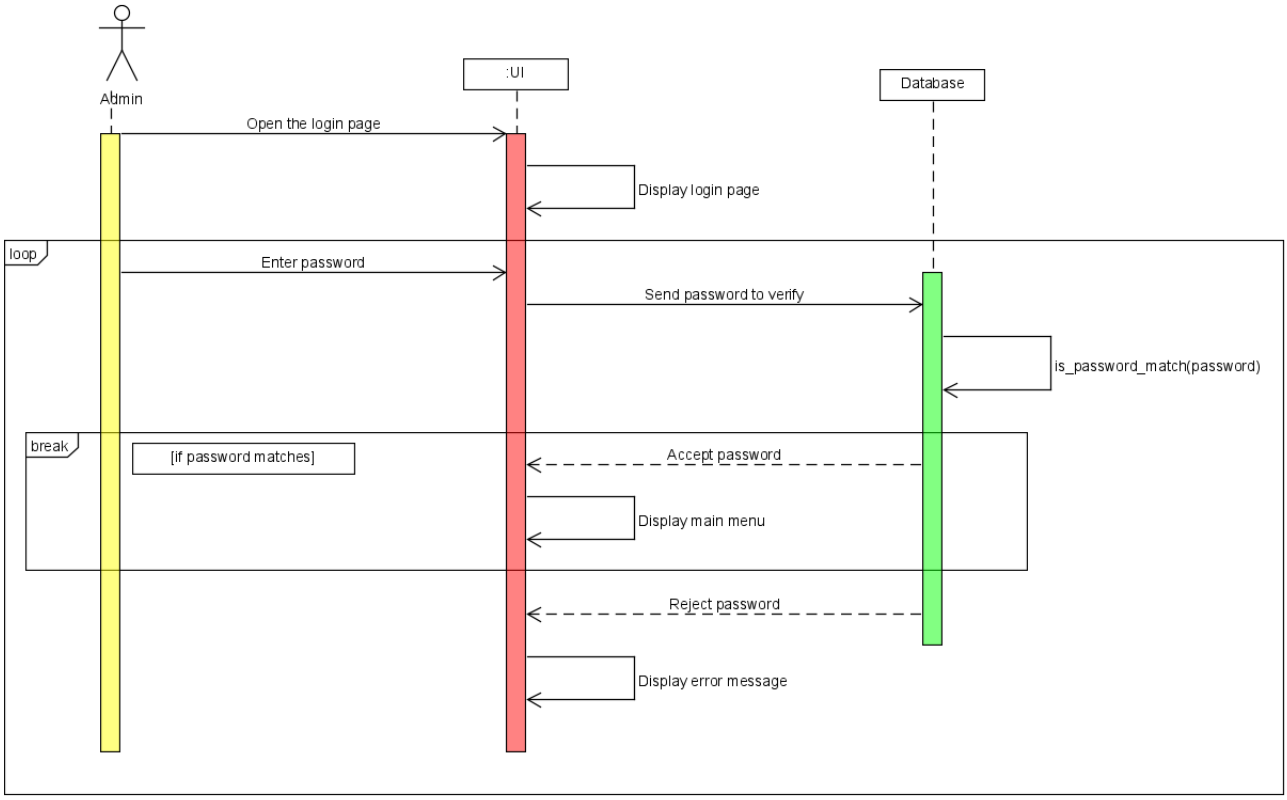

Figure 2.4: Sequence Diagram of use case 1

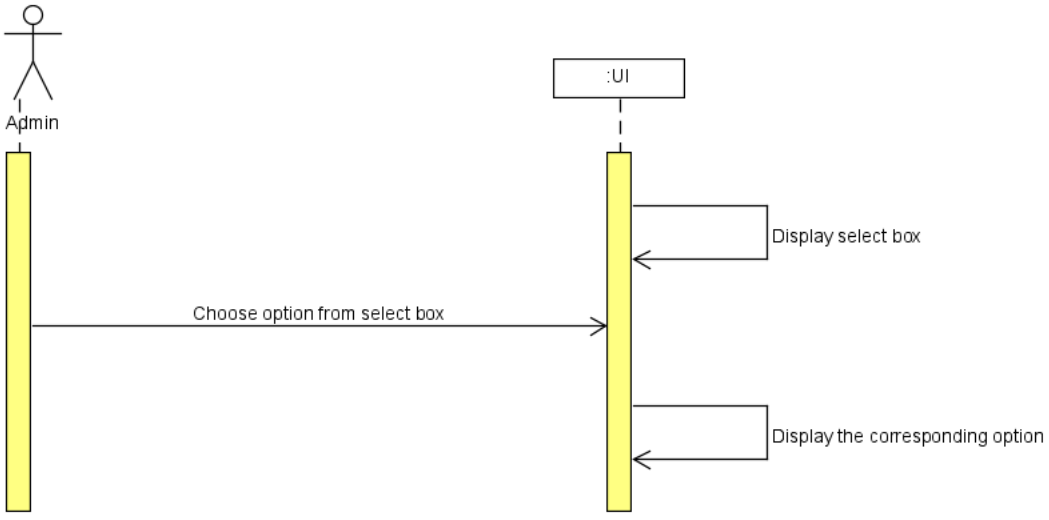

Figure 2.5: Sequence Diagram of use case 2

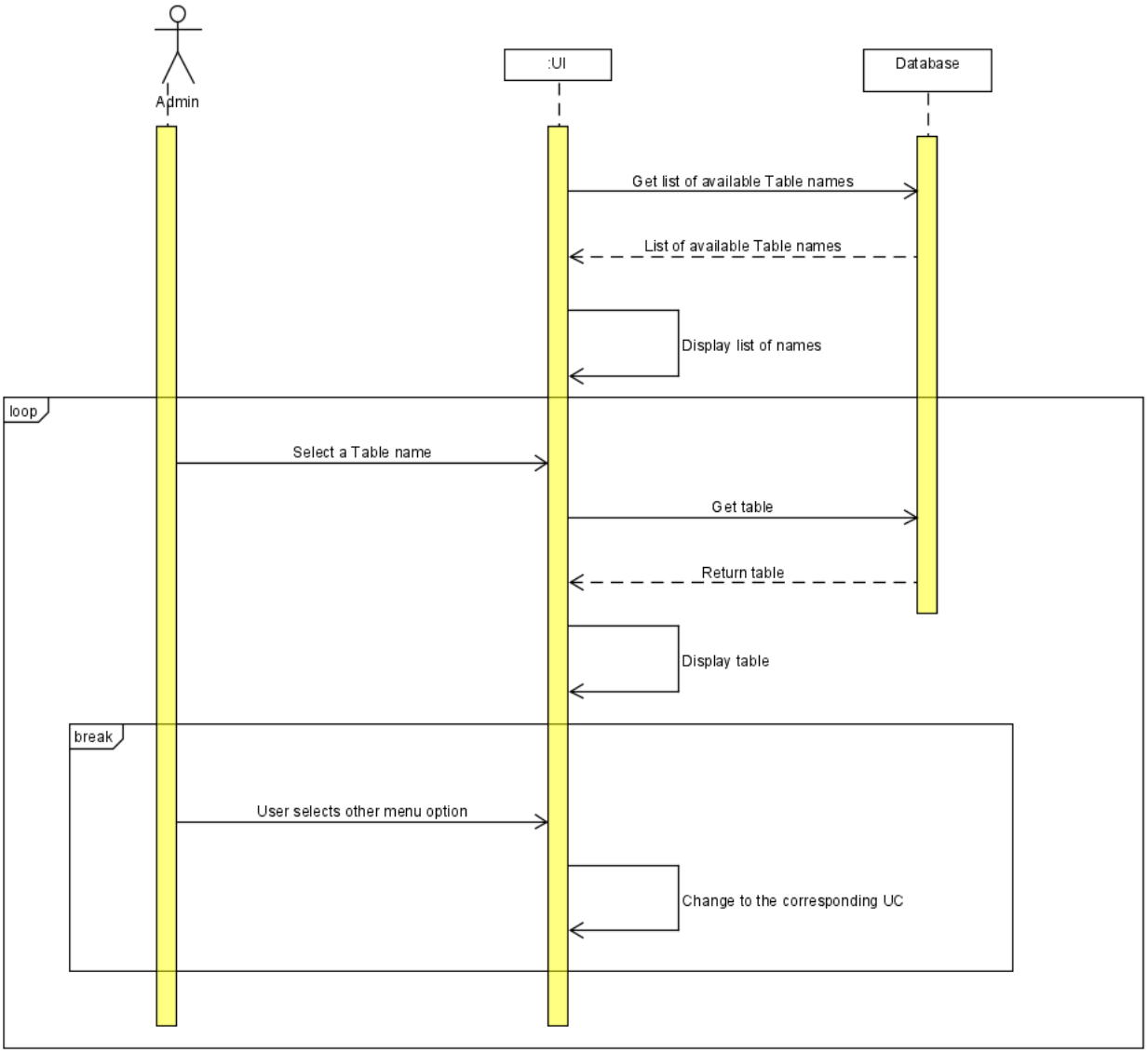

Figure 2.6: Sequence Diagram of use case 3

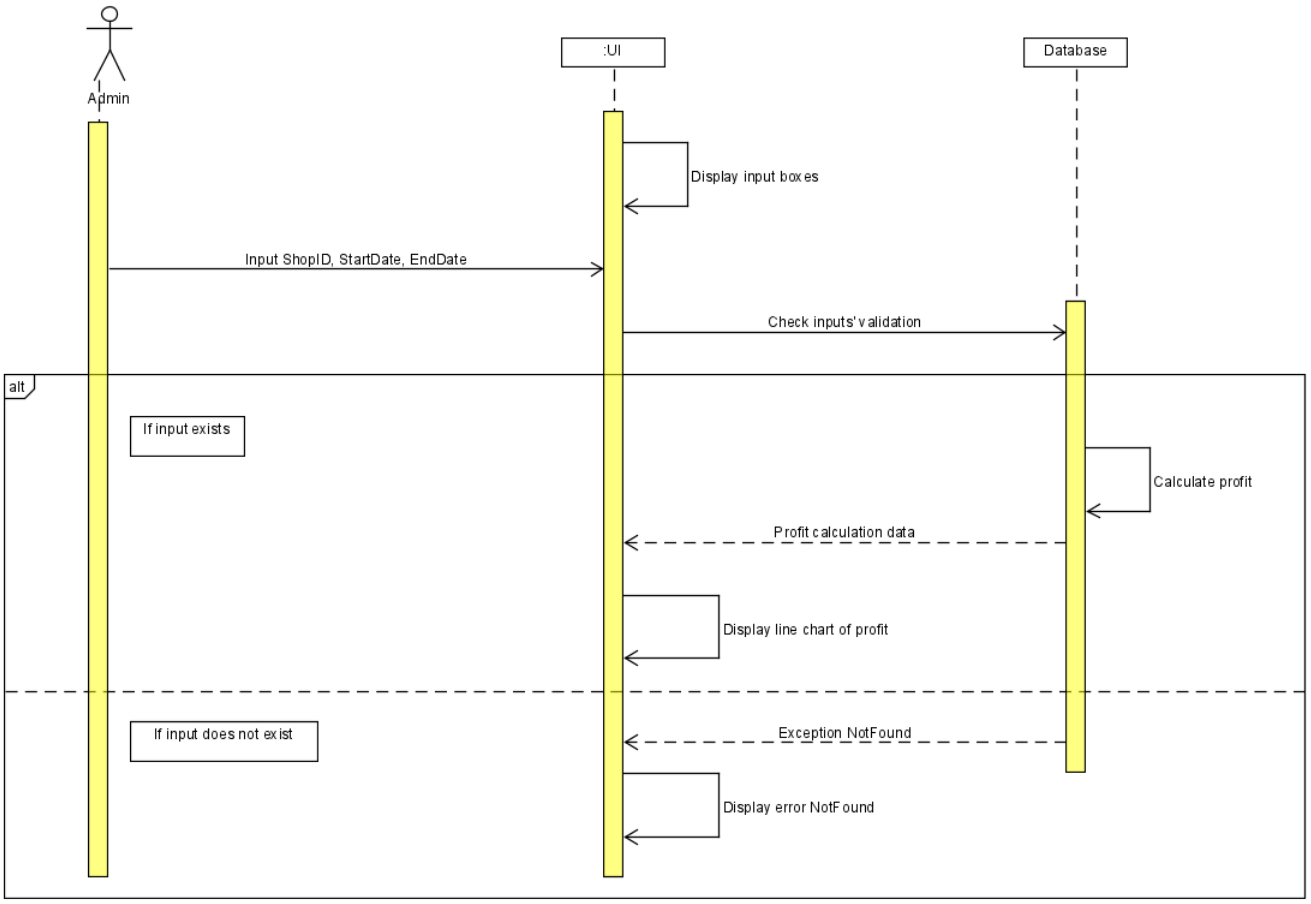

Figure 2.7: Sequence Diagram of use case 4

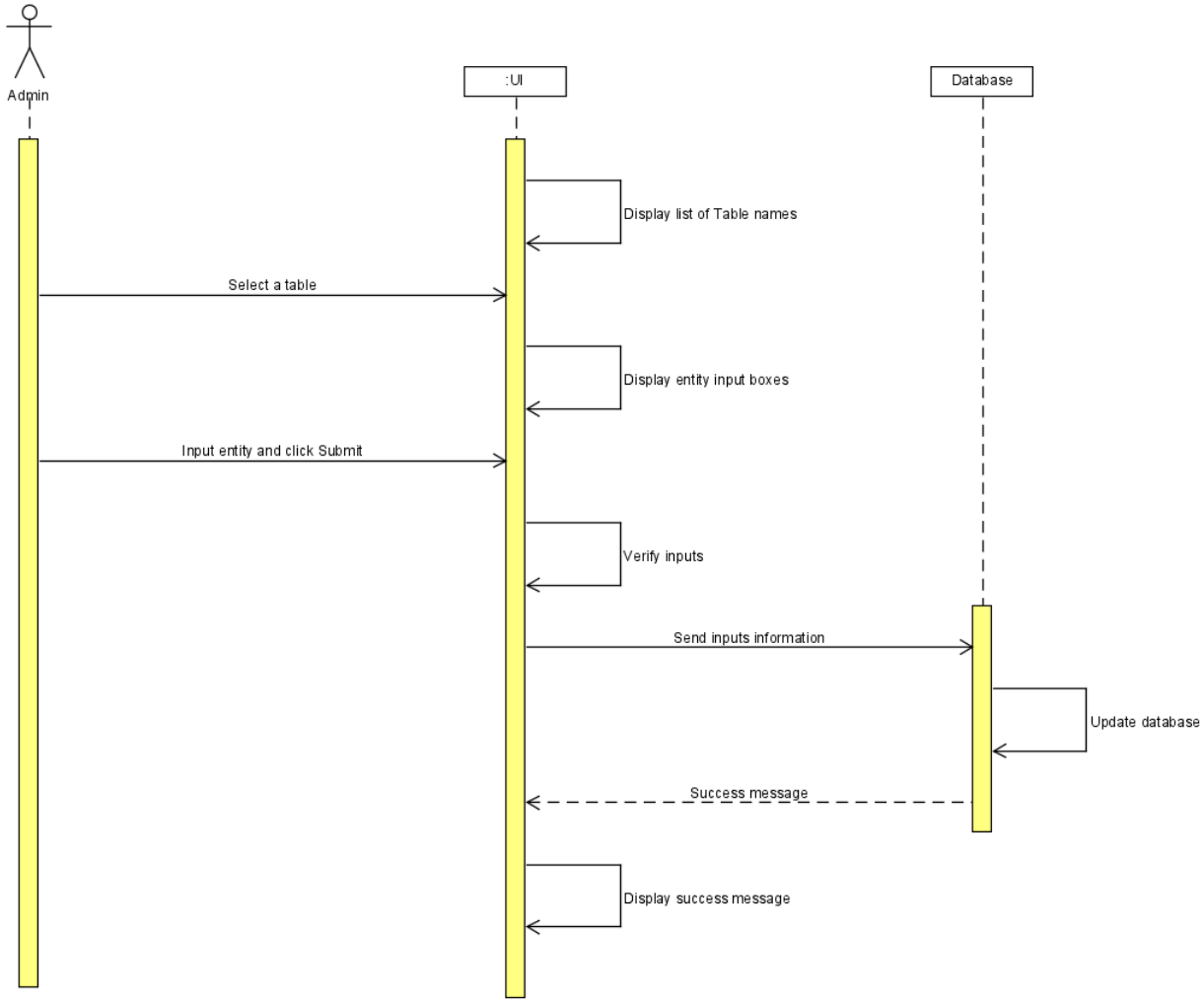

Figure 2.8: Sequence Diagram of use case 5

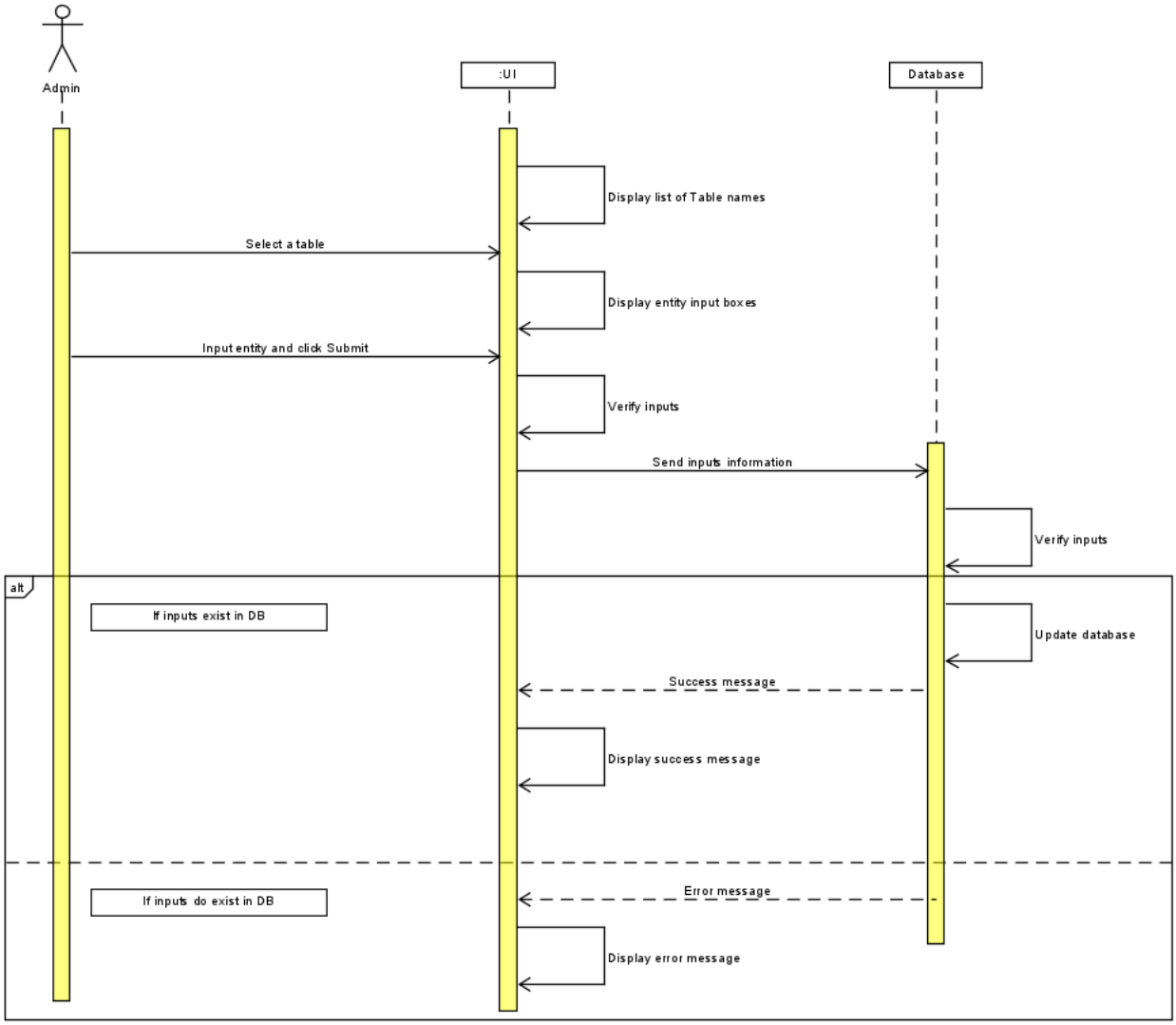

Figure 2.9: Sequence Diagram of use case 6

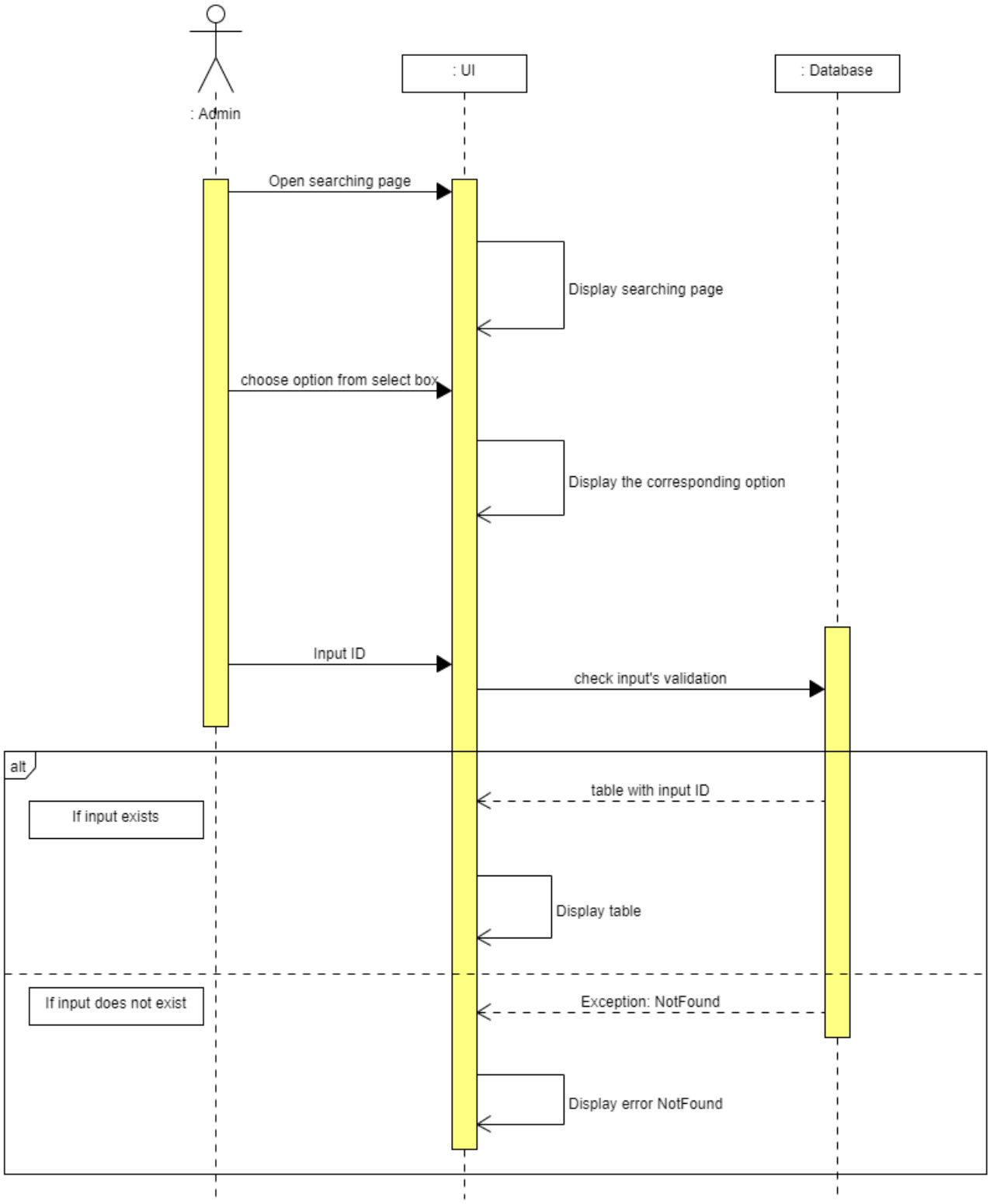

Figure 2.10: Sequence Diagram of use case 7

2.6 Test cases

ID	Description	Test step	Test data	Expected result	Actual result	Status
TC01	Check user login with valid password	1) Input password 2) Press Enter	Password: python	Login should be successful	As expected	Pass
TC02	Check user login with invalid password	1) Input password 2) Press Enter	Password: java	Login should be unsuccessful	As expected	Pass
TC03	Add a new customer	1) Go to Add page 2) Select Customer 3) Input customer's name 4) Click Add customer	Customer's name: Nguyễn Nhật Minh	Customer should be added to the database	As expected	Pass
TC04	Search an existing customer by name	1) Go to Search page 2) Select Customer 3) Input customer's name 4) Press Enter	Customer's name: Nguyễn Nhật Minh	Show the dataframe with the input customer's name	As expected	Pass
TC05	Add a new inventory	1) Go to Add page 2) Select Inventory 3) Input inventory id 4) Input inventory's name 5) Click Add inventory	Inventory id: INV001 Inventory's name: Inventory 1	Inventory should be added to the database	As expected	Pass
TC06	Search an existing inventory by name	1) Go to Search page 2) Select Inventory 3) Input inventory's name 4) Press Enter	Inventory's name: Inventory 1	Show the dataframe with the input inventory's name	As expected	Pass
TC07	Add a new category	1) Go to Add page 2) Select Category 3) Input category id 4) Input category's name 5) Click Add category	Category id: CAT001 Category's name: Laptop	Category should be added to the database	As expected	Pass

ID	Description	Test step	Test data	Expected result	Actual result	Status
TC08	Search an existing category by name	1) Go to Search page 2) Select Category 3) Input category's name 4) Press Enter	Category's name: Laptop	Show the dataframe with the input category's name	As expected	Pass
TC09	Add a new item	1) Go to Add page 2) Select Item 3) Input item's name 4) Input quantity 5) Input category id 6) Input inventory id 7) Click Add item	Item's name: Macbook Pro 2020 Quantity: 20 Category id: CAT001 Inventory id: INV001	Item should be added to the database	As expected	Pass
TC10	Search an existing item by name	1) Go to Search page 2) Select Item 3) Input item's name 4) Press Enter	Item's name: Macbook Pro 2020	Show the dataframe with the input item's name	As expected	Pass
TC11	Search an existing item by inventory id	1) Go to Search page 2) Select Item 3) Input inventory's name 4) Press Enter	Inventory id: INV001	Show the dataframe with the input inventory id	As expected	Pass
TC12	Search an existing item by category id	1) Go to Search page 2) Select Item 3) Input category's name 4) Press Enter	Category id: CAT001	Show the dataframe with the input category id	As expected	Pass
TC13	Add a new transaction	1) Go to Add page 2) Input transaction id 3) input transaction date 4) input transaction status 5) Input customer id 6) Input inventory id 7) Click Add transaction	Transaction id: TST001 Transaction date: 2020-12-07 Transaction status: PENDING Customer id: CUS001 Inventory id: INV001	Transaction should be added to the database	As expected	Pass

ID	Description	Test step	Test data	Expected result	Actual result	Status
TC14	Search an existing transaction by transaction id	1) Go to Search page 2) Input transaction id 3) Press Enter	Transaction id: TST001	Show the dataframe with the input transaction id	As expected	Pass
TC15	Search an existing transaction by transaction date	1) Go to Search page 2) Input transaction date 3) Press Enter	Transaction date: 2020-12-07	Show the dataframe with the input transaction date	As expected	Pass
TC16	Search an existing transaction by customer id	1) Go to Search page 2) Input customer id 3) Press Enter	Customer id: CUS001	Show the dataframe with the input customer id	As expected	Pass
TC17	Add a new import	1) Go to Add page 2) Input import id 3) input import date 4) Input buyer id 5) Input inventory id 6) Click Add import	Import id: IPT001 Import date: 2020-12-06 Buyer id: BUY001 Inventory id: INV001	Import should be added to the database	As expected	Pass
TC18	Search an existing import by import id	1) Go to Search page 2) Input import id 3) Press Enter	Transaction id: IPT001	Show the dataframe with the input import id	As expected	Pass
TC19	Search an existing import by import date	1) Go to Search page 2) Input import date 3) Press Enter	Transaction date: 2020-12-06	Show the dataframe with the input import date	As expected	Pass
TC20	Show profit plot	1) Go to Plot page 2) Input start date 3) Input end date 4) Select inventory id (can select multiple id)	Start date: 2020-11-07 End date: 2020-12-07 Inventory id: INV001	Show the plot of profit in a specified amount of time of (an) inventori(es)	As expected	Pass

2.7 Dummy data

We do not have a real world data, so we use the data provided by 1C company. The link to the data can be found here. [3]. We changed the data to fit our schema, using the Jupyter notebook here.

Chapter 3

A Web app

3.1 Background

Days before 23rd October, 2020, we encountered several problems:

- The GUI library we intended to use, PyQt, did not meet our expectation.
- We tried and concluded a lot of work would be required to do both GUI, database and functionality with PyQt
- We had no experience in using PyQt, thus making it to meet our expectation was not possible.

Therefore, after these careful consideration, on 23rd October we decided to switch it to a **Web app**:

- The library we intend to use, Streamlit, is a Python library built on top of JavaScript.
- Streamlit web app can be deployed free of charge, which makes it easier to demo.
- Streamlit supports very beautiful data visualization, which is our main goals: viewing and plotting profit
- Streamlit supports both desktop and mobile view automatically.

3.2 Analysis

We decide to keep all the previous analysis work. Changing from a desktop app to a web app with only one user does not change much of the analysis.

3.3 Constraints

When switching to web app, we have some constraints that we think worth mentioning:

- Streamlit hosts the web app, but limits it hardware to: 1 CPU, 800MB RAM, 800MB disk size. However, this is enough to do a live demo.

- GitHub limits 100MB of upload file size, which is much smaller than our database file. We have to zip it, and import it on the first run, which makes the first run takes time.
- Each of Streamlit's component is limited to be 50MB in size, thus we cannot view the whole database. We limit it to be the first 100 000 rows.

However, when running on localhost, the only constraint is the specification of the PC that runs it.

3.4 Libraries

These are all the libraries that we use. Note that some are not listed, because they are included in the listed library already:

- Python 3.8
- streamlit, at least 0.70.0
- pandas, at least 1.1.3
- bcrypt, at least 3.2.0
- plotly, at least 4.13.0
- hiplot, at least 0.1.20
- pandas profiling, at least 2.9.0
- numpy, version 1.19.3

All the requirements are in the `requirements.txt` file.

Chapter 4

Installation and Run

The project is publicly available as a GitHub repository:
https://github.com/minhlong94/SWE_IT076IU

To run the project, one must have pip3 and Python 3.8 installed.

4.1 Installation

4.1.1 Run on localhost

1. Download the project
2. Use Terminal to navigate to the project path `path/to/SWE_IT076IU/`
3. Enter the command: `pip install -r requirements.txt`
4. When all requirements are installed, enter the command: `streamlit run main.py`

4.1.2 Run on a web hosting service

Streamlit comes with a wonderful feature: Streamlit Sharing. It is a free web hosting service provided by Streamlit itself. For your convenience, we have deployed our app on its server.
Use this hyperlink: https://share.streamlit.io/minhlong94/swe_it076iu/main.py to use the hosted web app. No installation required.

4.2 How to run

When one runs the app from terminal, or browses it using the web app link, this is the first page:

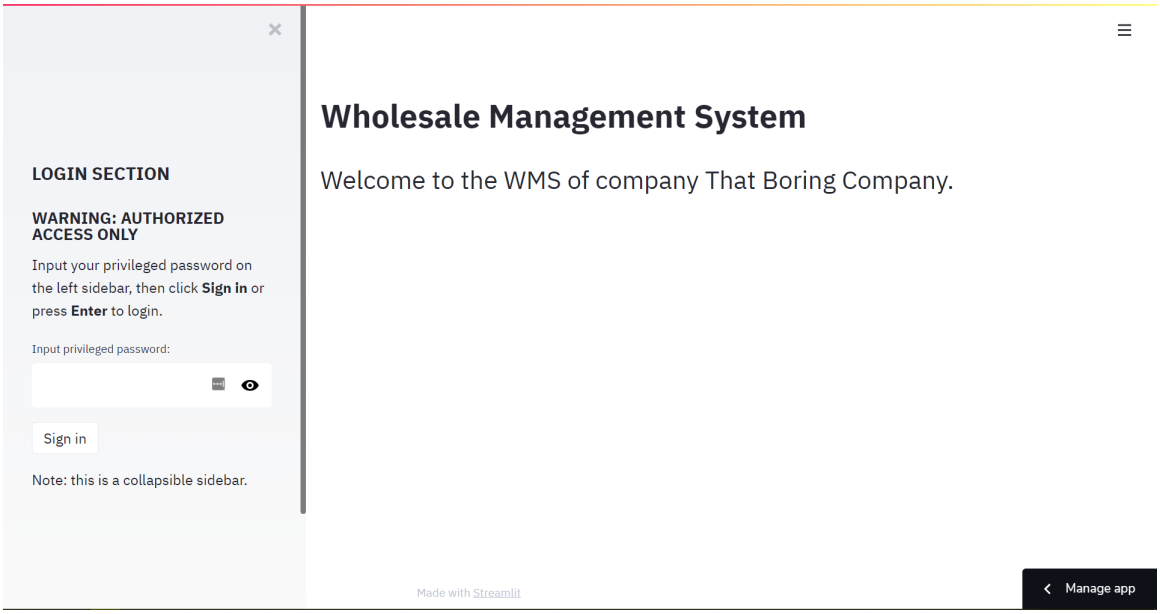

Figure 4.1: Main page

User can disable the default Wide view of the app by clicking the icon at the top right corner, => Setting => uncheck "Show app in Wide mode".

On the left sidebar, enter the password `python`, then click `Sign in`. First run will take a longer time to create database and import data from csv files in a zipped folder.

Further how to use the web app will be demonstrated on the demo.

Chapter 5

Conclusion

5.1 Lessons

Finishing this project teaches us valuable lessons that we can apply in the future:

- Software development process: from analysis to prototype implementation, testing, use cases, etc
- Use many tools for various purposes: drawing Entity-Relationship Diagram, Relational Schema, Class Diagram, Sequence Diagram
- Know how to use Python for simple web app using Streamlit, together with interactive plots
- CRUD and REST for Web app

5.2 Limitations

We identity some limitations that we hope to learn and improve in the future:

- Scalable project. This project is small, so most design patterns to scale the project are not importantly considered.
- More test cases. We may not have exhausted all possible combination of actions yet.
- Scalable database. The dataset we use is quite small. We wonder what will happen if the dataset is big.
- More complicated requirements. We only analyze a small number of requirements, and they are simple.
- Data integrity and app security. We simply encrypt the password using bcrypt, and the data is not encrypted. We wonder how big companies encrypt the data and guarantee the data integrity and the app's security.

Bibliography

[1] *bcrypt*. https://github.com/pyca/bcrypt/. 2020.

[2] Simon Brugman. *pandas-profiling: Exploratory Data Analysis for Python*. https://github.com/pandas-profiling/pandas-profiling. Version: 2.X, Accessed: INSERT$_{D}ATE_{H}ERE$. 2019.

[3] 1C Company. *Predict Future Sales data*. URL: https://www.kaggle.com/c/competitive-data-science-predict-future-sales/data (visited on 12/12/2020).

[4] D. Haziza, J. Rapin, and G. Synnaeve. *Hiplot, interactive high-dimensionality plots*. https://github.com/facebookresearch/hiplot. 2020.

[5] Plotly Technologies Inc. *Collaborative data science*. 2015. URL: https://plot.ly.

[6] *Streamlit*. https://github.com/streamlit/streamlit. 2020.

[7] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.