

PROJECT REPORT

Wholesale Management System

Course: Web Application Development IT093IU



Luu Minh Long, Nguyen Nhat Minh
ITITI18079, ITITI18086
International University - VNUHCMC

Contents

1	Project Overview	4
1.1	Group members	4
1.2	Topic	4
1.3	Timeline	4
1.4	Requirements	4
1.5	Report structure	5
1.6	Resources	5
2	Project Analysis	6
2.1	Requirements Analysis	6
2.1.1	Requirements Analysis	6
2.1.2	Software and Hardware decision	7
2.1.3	Risk Management	7
2.2	Use cases	8
2.3	Entity-Relationship diagram	12
2.4	Sequence Diagram	13
2.5	Dummy data	17
3	Implementation Details	18
3.1	Constraints	18
3.2	Hardware requirements	18
3.3	Libraries	18
3.4	Request and Response API	18
4	Installation and Run	20
5	Conclusion	21
5.1	Lessons	21
5.2	Limitations	21

List of Figures

2.1	Project use case diagram	8
2.2	Entity-Relationship Diagram	12
2.3	Sequence Diagram of use case 1	13
2.4	Sequence Diagram of use case 2	13
2.5	Sequence Diagram of use case 3	14
2.6	Sequence Diagram of use case 4	15
2.7	Sequence Diagram of use case 5	16
2.8	Sequence Diagram of use case 7	17
3.1	Sample APIs (1)	19
3.2	Sample APIs (2)	19

List of Tables

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

Chapter 1

Project Overview

This report contains the information about the project report of Course "Web Application Development" IT093IU at International University, Vietnam National University – Ho Chi Minh City.

1.1 Group members

Our group consists of two members:

- Luu Minh Long - ITITI18079
- Nguyen Nhat Minh - ITITI18086

1.2 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.

Moreover, it should have beautiful and interactive plots that help users to grasp the big picture of the sales. Seeing that this topic may help us understand online shopping and digital commercials, we decided to choose this to be the project's topic.

1.3 Timeline

The project started on March 25th, 2021 and is expected to end on June 19th, 2021.

1.4 Requirements

The original requirements of the projects are as follows:

Design a web 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.

1.5 Report structure

In this report, we break it into two parts.

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

In the second part, we describe the implementation details of the project.

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

1.6 Resources

We use our personal devices to develop the web app.

Chapter 2

Project Analysis

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 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
5	Profit calculation	Calculate the profit in a week/month/year	Functional
6	Adding new entities	Add new user	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 application will be delivered with **ONE** default admin account.

2.1.2 Software and Hardware decision

After analyzing the requirements, we decide to choose our programming language, libraries and tools to implement the project as follows:

- 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.
- Database: PostgreSQL, a relational database
- Container: Docker. Docker creates a virtual environment for the app to run on to avoid conflicts and requires **only** the Docker app to run the project.
- Backend: FastAPI
- Frontend: Streamlit

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/Doki064/WAD-IT093IU/blob/dev/coding_style.md

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 Use cases

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

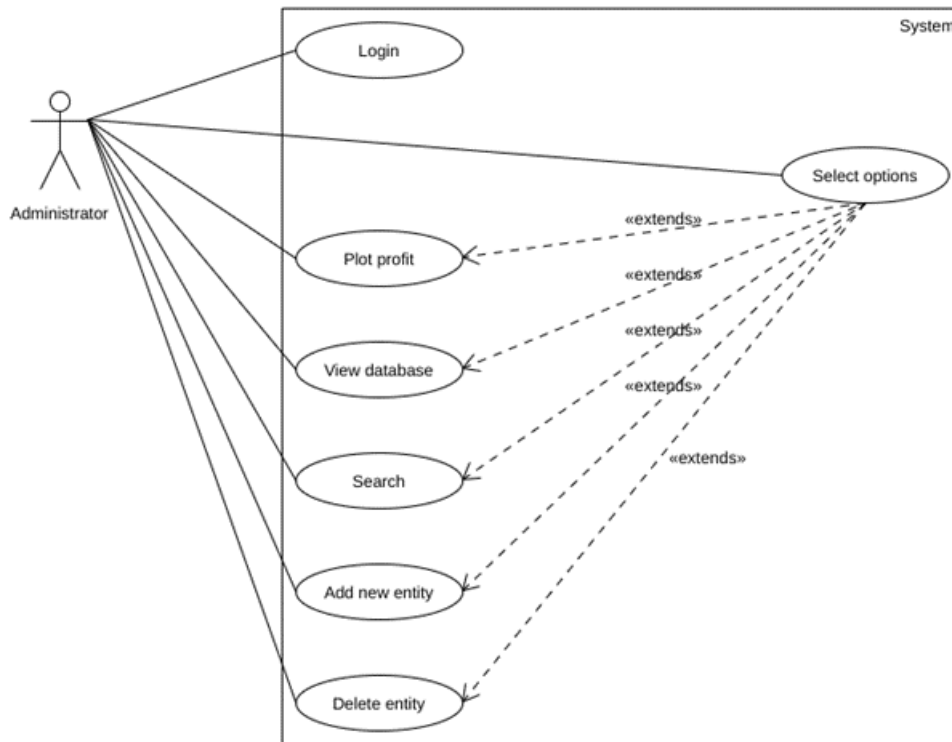


Figure 2.1: Project use case diagram

Use case 1: Login to the system

Identifier: UC1

Input: valid account

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 \geq 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 \geq 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 (account)

Identifier: UC5

Input: New user account with username and password

Output:

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

Actor	System
1. Open register page	1.1. Display input sections
2. Input username and password	2.1. Check preconditions
3. Submit	3.1. If success, show success message 3.2. If error, show a warning message and terminate process

Table 2.8: Use case 5

Precondition: username and password's characters must be greater than 8 and less than 30

Postcondition: None

Use case 6: 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 2.9: Use case 7

Precondition:

- Logged in
- ID or name exists

Postcondition: None

2.3 Entity-Relationship diagram

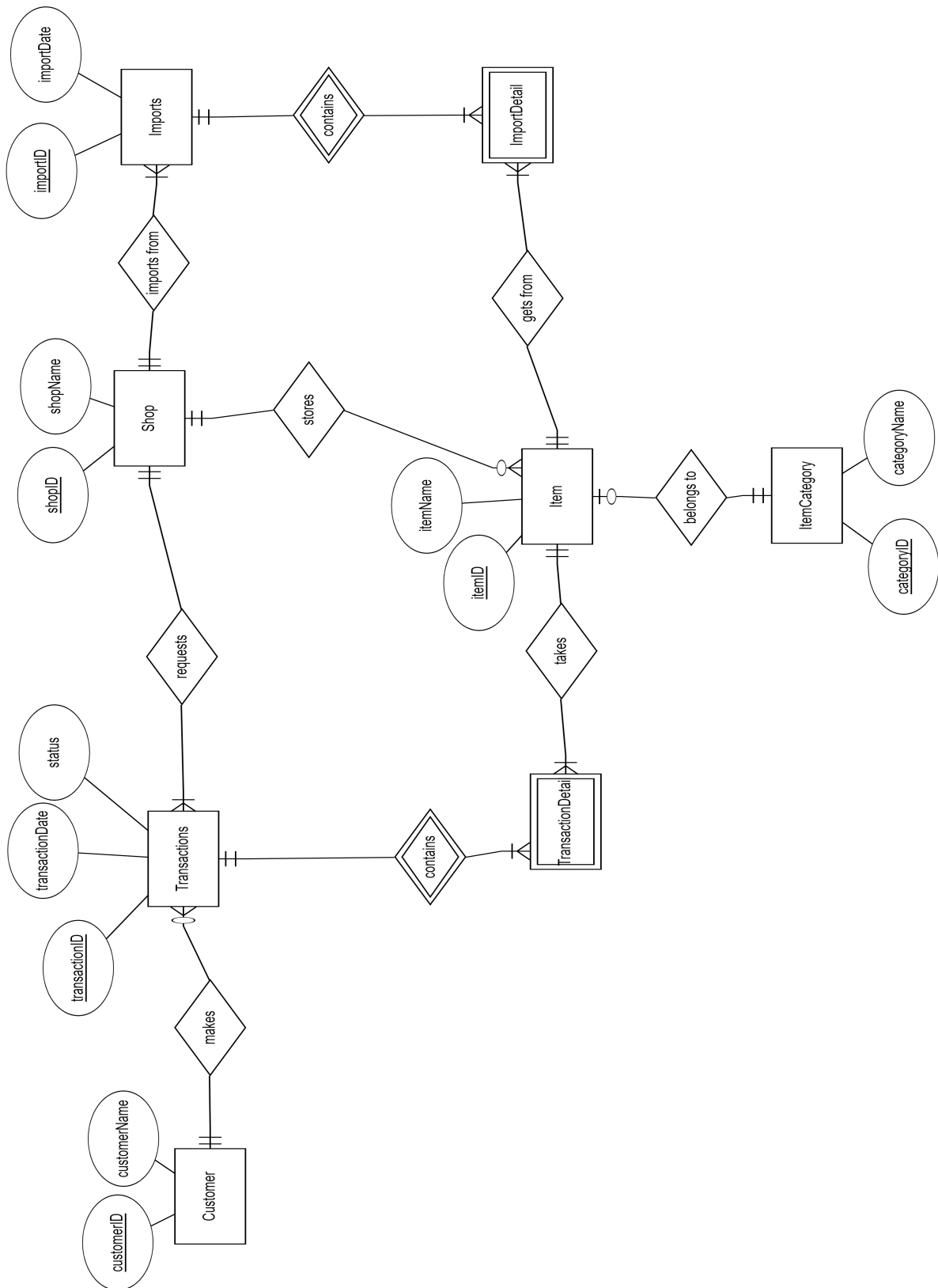


Figure 2.2: Entity-Relationship Diagram

2.4 Sequence Diagram

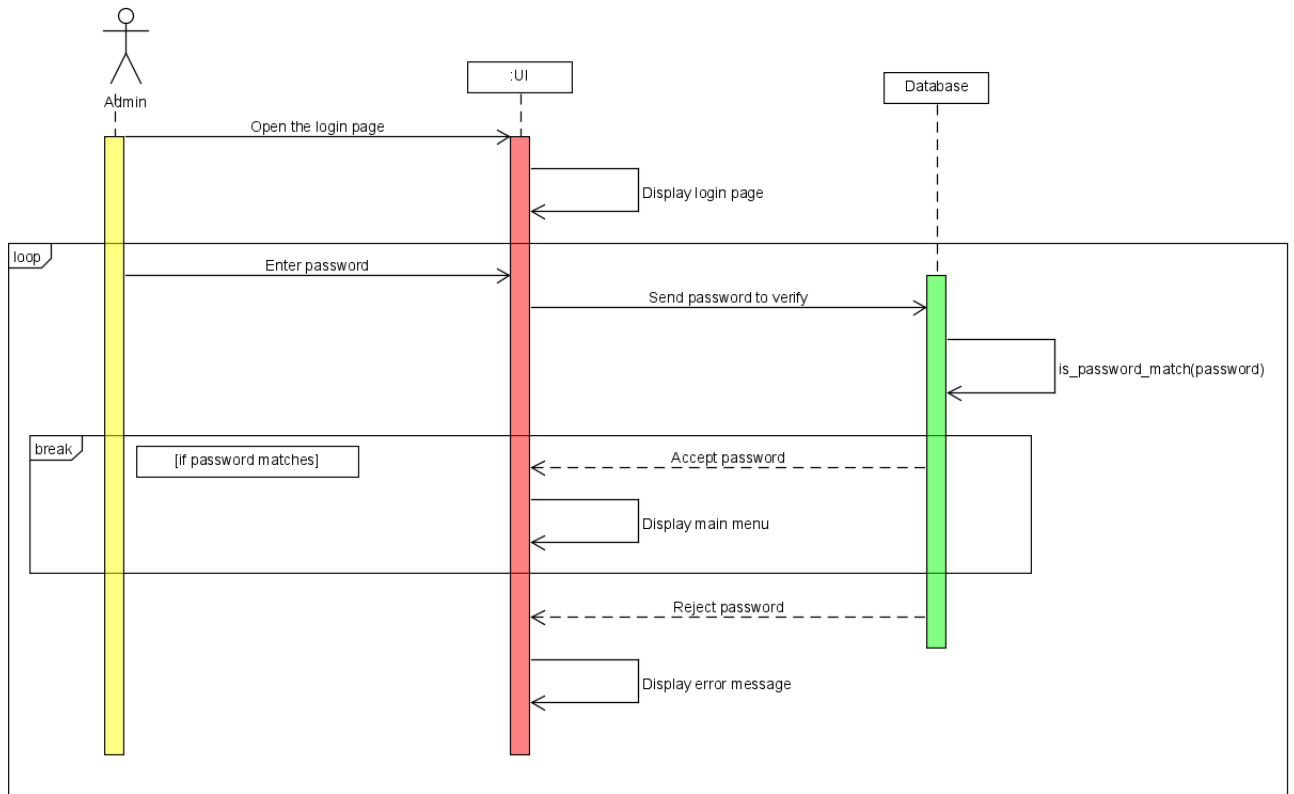


Figure 2.3: Sequence Diagram of use case 1

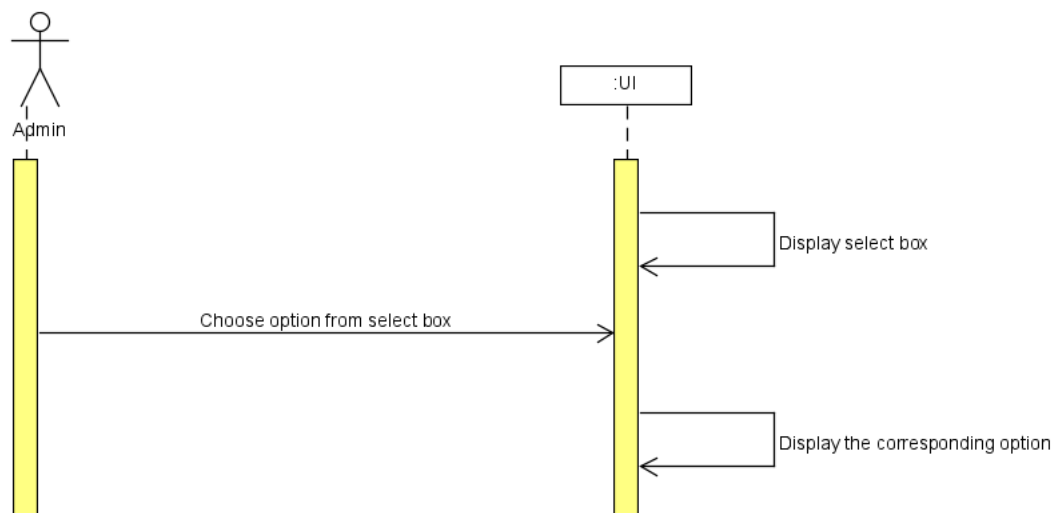


Figure 2.4: Sequence Diagram of use case 2

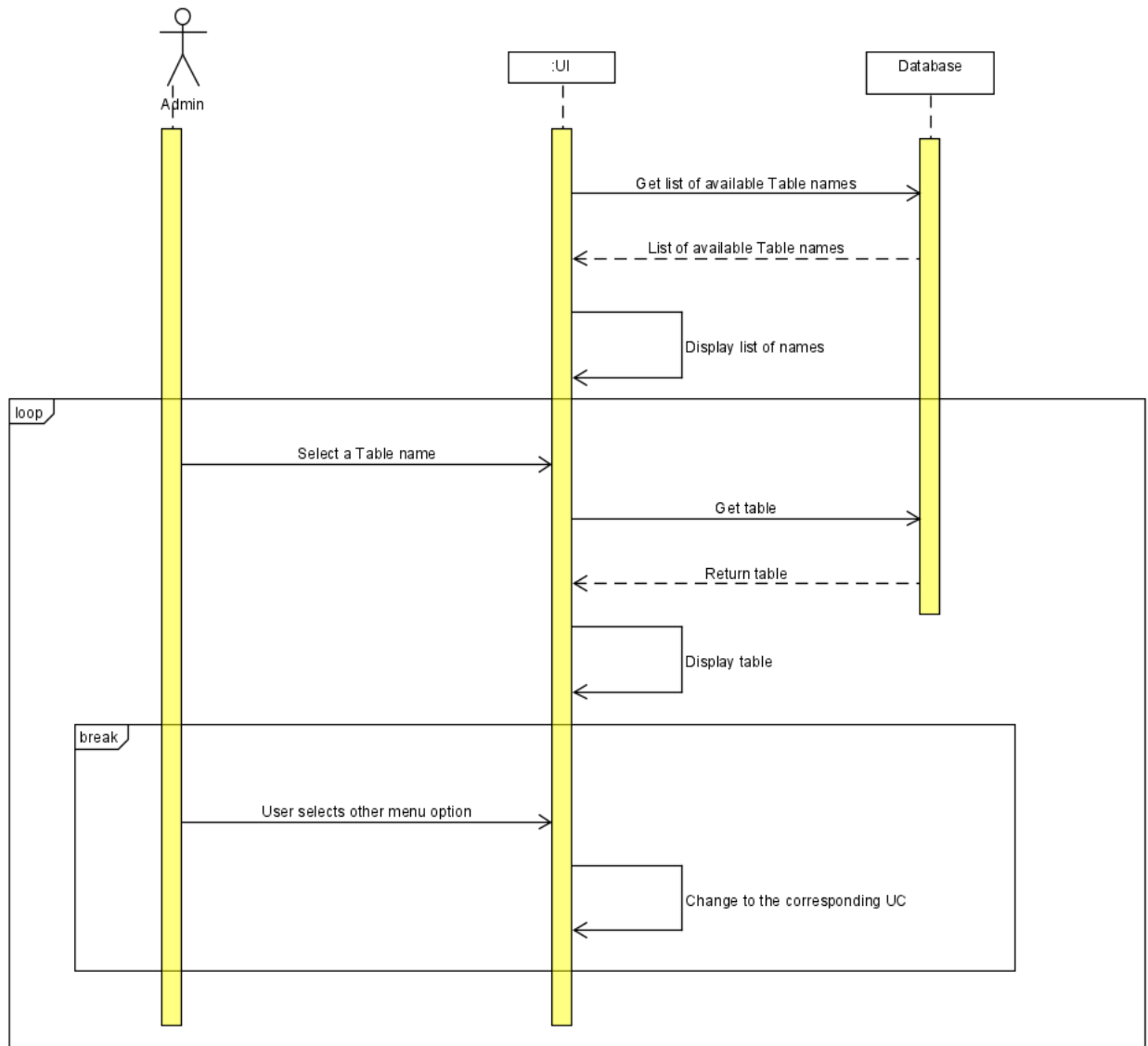


Figure 2.5: Sequence Diagram of use case 3

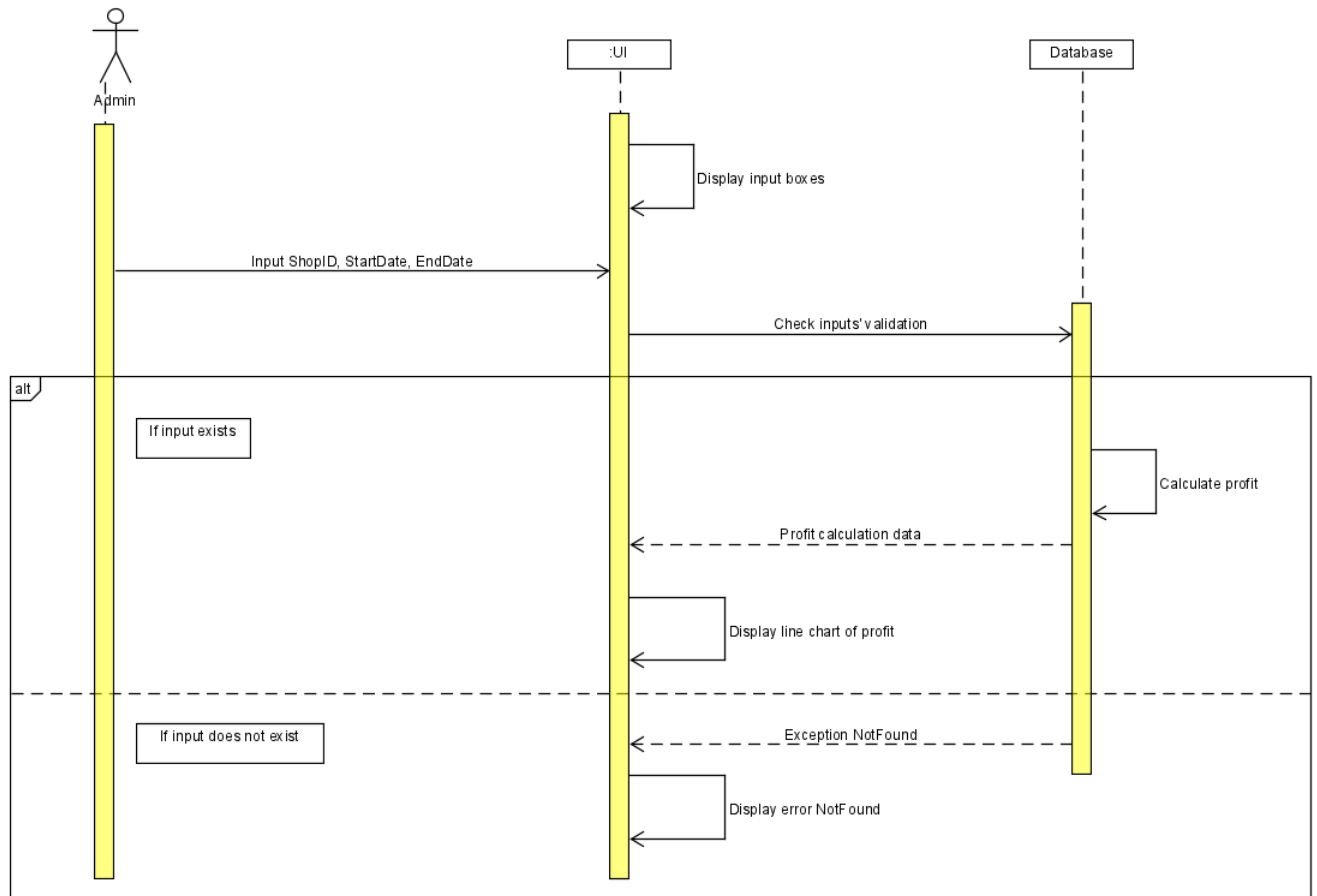


Figure 2.6: Sequence Diagram of use case 4

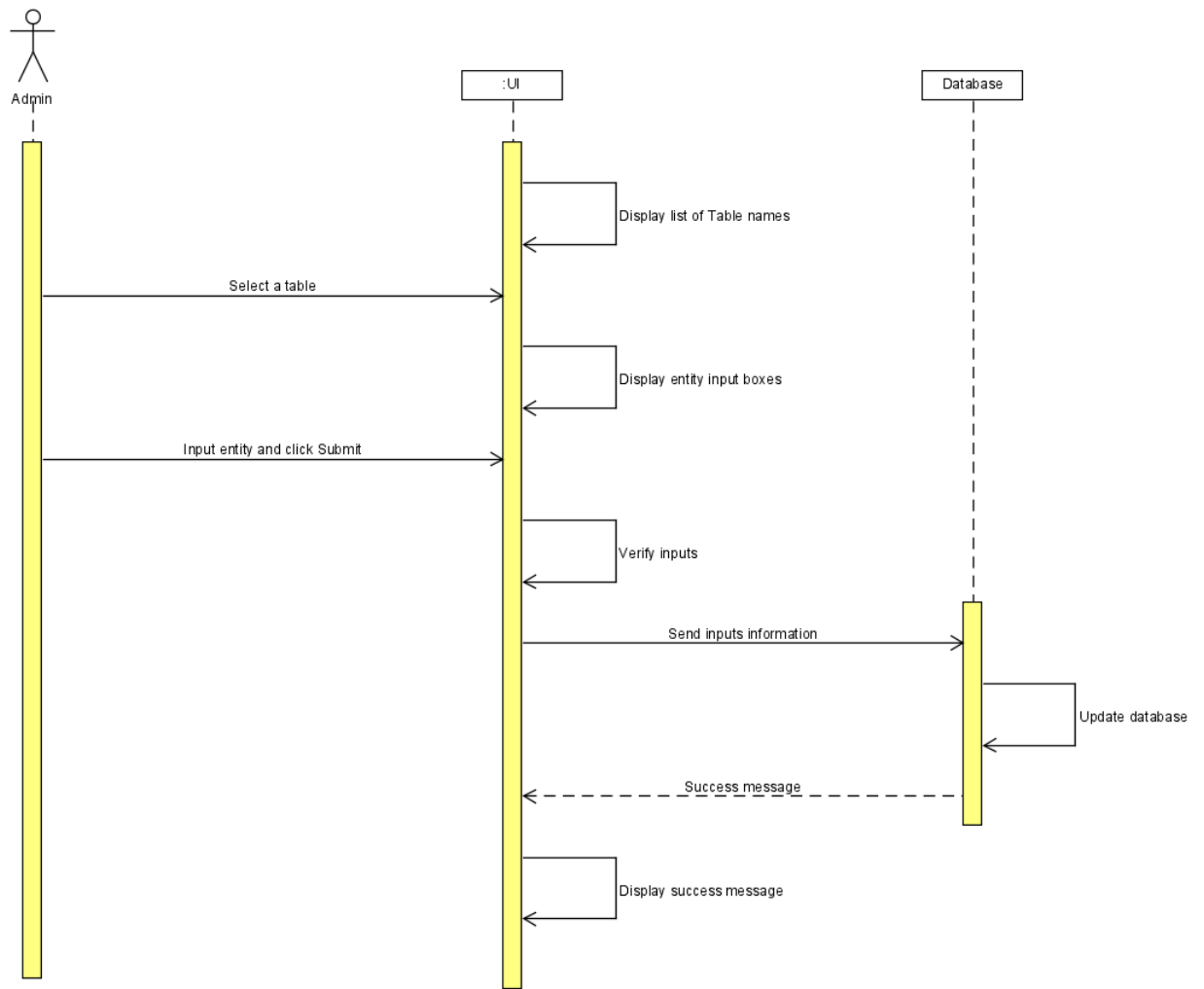


Figure 2.7: Sequence Diagram of use case 5

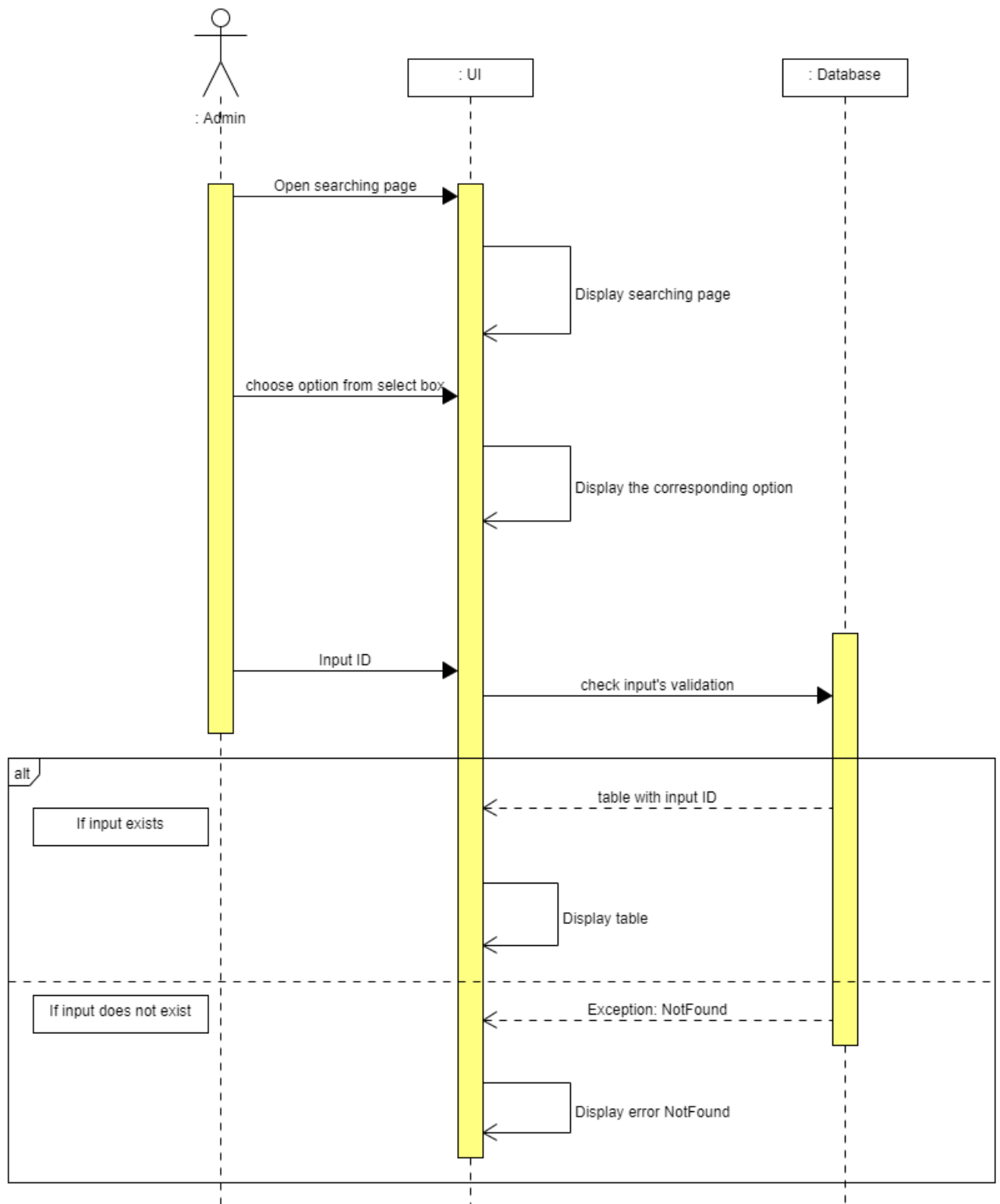


Figure 2.8: Sequence Diagram of use case 7

2.5 Dummy data

We use the sales data provided by 1C company **[data]**. We changed the data to fit our schema, of which code can be found [here](#).

Chapter 3

Implementation Details

3.1 Constraints

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. Moreover, it does not support Docker, so we cannot deploy it to its server as demo.
- GitHub limits 100MB of upload file size. We limit the data file to 98MB, though if readers want to commit bigger data to GitHub, they should give attention to this constraint.
- Each of Streamlit's component is limited to be 50MB in size, thus we cannot view the whole database as table. We limit some of the functions be the first 100,000 rows.

3.2 Hardware requirements

- 8GB RAM
- 5GB storage
- OS: Linux-based

3.3 Libraries

Dependencies can be found in the corresponding `pyproject.toml` file of each component (frontend and backend).

3.4 Request and Response API

We list some sample Request-Response APIs in the following two figures. Note that this is include, but not limited to.

Figure 3.1: Sample APIs (1)





			Authorize 
root			^
GET	/	Root	▼
users			^
POST	/api/v1/users/register	Register	▼
POST	/api/v1/users/login/auth	Login	▼
GET	/api/v1/users/me	Read User Me	▼ 
GET	/api/v1/users	Read Users	▼ 
GET	/api/v1/users/{user_id}	Read User	▼ 
customers			^
GET	/api/v1/customers	Read Customers	▼
POST	/api/v1/customers	Create Customer	▼
GET	/api/v1/customers/{customer_id}	Read Customer	▼
GET	/api/v1/customers/{customer_id}/transactions	Read Transactions Of Customer	▼
POST	/api/v1/customers/{customer_id}/transactions	Create Transaction For Customer	▼

Figure 3.2: Sample APIs (2)

shops			^
GET	/api/v1/shops	Read Shops	▼
POST	/api/v1/shops	Create Shop	▼
GET	/api/v1/shops/{shop_id}	Read Shop	▼
GET	/api/v1/shops/{shop_id}/importations	Read Importations Of Shop	▼
POST	/api/v1/shops/{shop_id}/importations	Create Importation For Shop	▼
GET	/api/v1/shops/{shop_id}/transactions	Read Transactions Of Shop	▼
GET	/api/v1/shops/{shop_id}/items	Read Items Of Shop	▼
transactions			^
GET	/api/v1/transactions/min-max-dates	Read Min Max Dates	▼
GET	/api/v1/transactions	Read Transactions	▼
GET	/api/v1/transactions/date/{date}	Read Transactions By Date	▼
GET	/api/v1/transactions/{transaction_id}	Read Transaction	▼
GET	/api/v1/transactions/{transaction_id}/details	Read Transaction Details	▼
GET	/api/v1/transaction_details	Read Transaction Details	▼

Chapter 4

Installation and Run

The project's source code is publicly available as a GitHub repository:

<https://github.com/Doki064/WAD-IT093IU>

To run the project, only Docker Compose is required. A userguide can be found in the **README.md** file.

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, FastAPI, Docker and Streamlit for a web app.
- CRUD and REST for Web app

5.2 Limitations

We identify 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 small, but we already experience slow loading time.
- 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, which raises a question: how big companies encrypt the data and guarantee the data integrity and the app's security.