ADDIS ABABA UNIVERISITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

CENTER OF INFORMATION TECHNOLOGY AND SCIENTIFIC COMPUTING

DEPARTMENT OF SOFTWARE ENGINERRING

Management System for Gead General Trading PLC

Software Design Specification

PREPARED BY: -

1. Adiam Geberselassie
2. Afrah Awol
3. Biruk Wondirad
4. Eyuel Berga
5. Jemila Ibrahim
6. Yanet Endale

ADVISOR: Mr. Fistum Alemu

May 2018

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Definitions, Acronyms, and Abbreviations

Table 1 - Definitions

Term	**Definition**
MSGT	Management System for Gead Trading; the software product that is to be produced
MVC	MVC is a software approach that separates application logic from presentation
UML	Unified Modelling Language

1. Introduction

1.1 Purpose

The purpose of System Design document is to translate the business requirements and business processes into a technical design that will be used to develop the Management system for Gead Trading (MSGT) application.

1.2 General Overview

The implementation of MSGT will take the MVC (Model-View-Controller) system architecture. MVC is a software approach that separates application logic from presentation.

The first reason we chose this particular architecture is that it allows the separation of task. This will help us to implement different parts of the application without worrying about integration issues. Due to the separation of logic and presentation, the team will be able to independently implement the different parts of the application. This will help in making the implementation phase fast.

Maintenance will also be fairly easy, as it will not require a complete change in the whole system. Using this architecture allows us to make changes to the implementations of controllers and models without making any changes to the views. It also allows us to change the appearance to the views without any changes to the other parts of the application.

1.3 Development Methods & Contingencies

The development of the MSGT system will take on an object oriented approach together with the model-view-controller architectural pattern. We will be using the unified modeling language (UML) for constructing the design models. We will use the principle of decoupling to ensure the maintainability of the system.

We will also be using different frameworks in the implementation of the system. Codeigniter will be used as the application framework for the back-end implementation of the system. We will also be using Bootstrap for implementing the front-end part of the system.

2. System Architecture

2.1 Subsystem decomposition

2.1.1 Layer 1

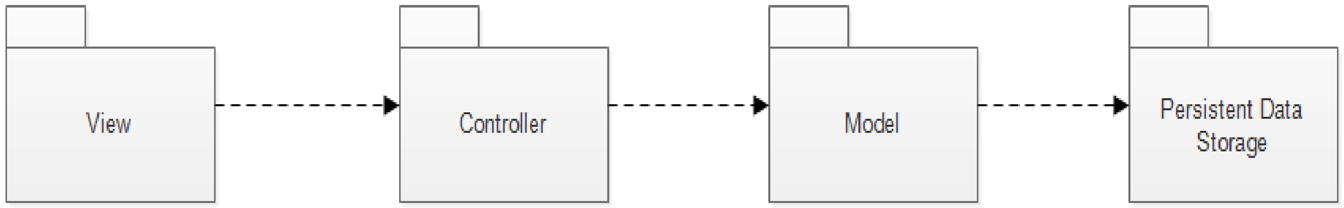

Figure 1 - Layer 1 Component Diagram

2.1.2 Layer 2

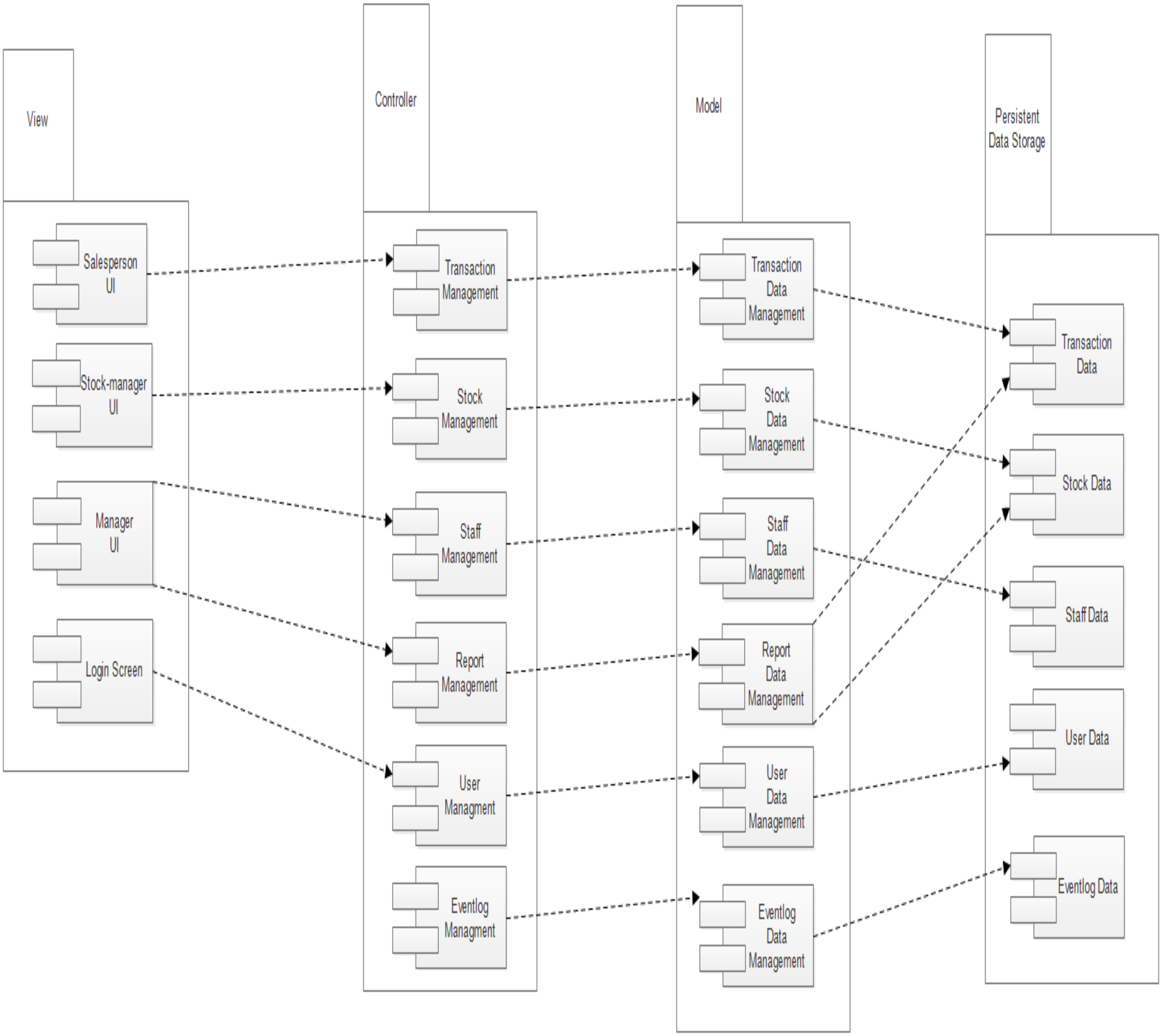

Figure 2 - Layer 2 Component Diagram

2.1.3 Layer 3

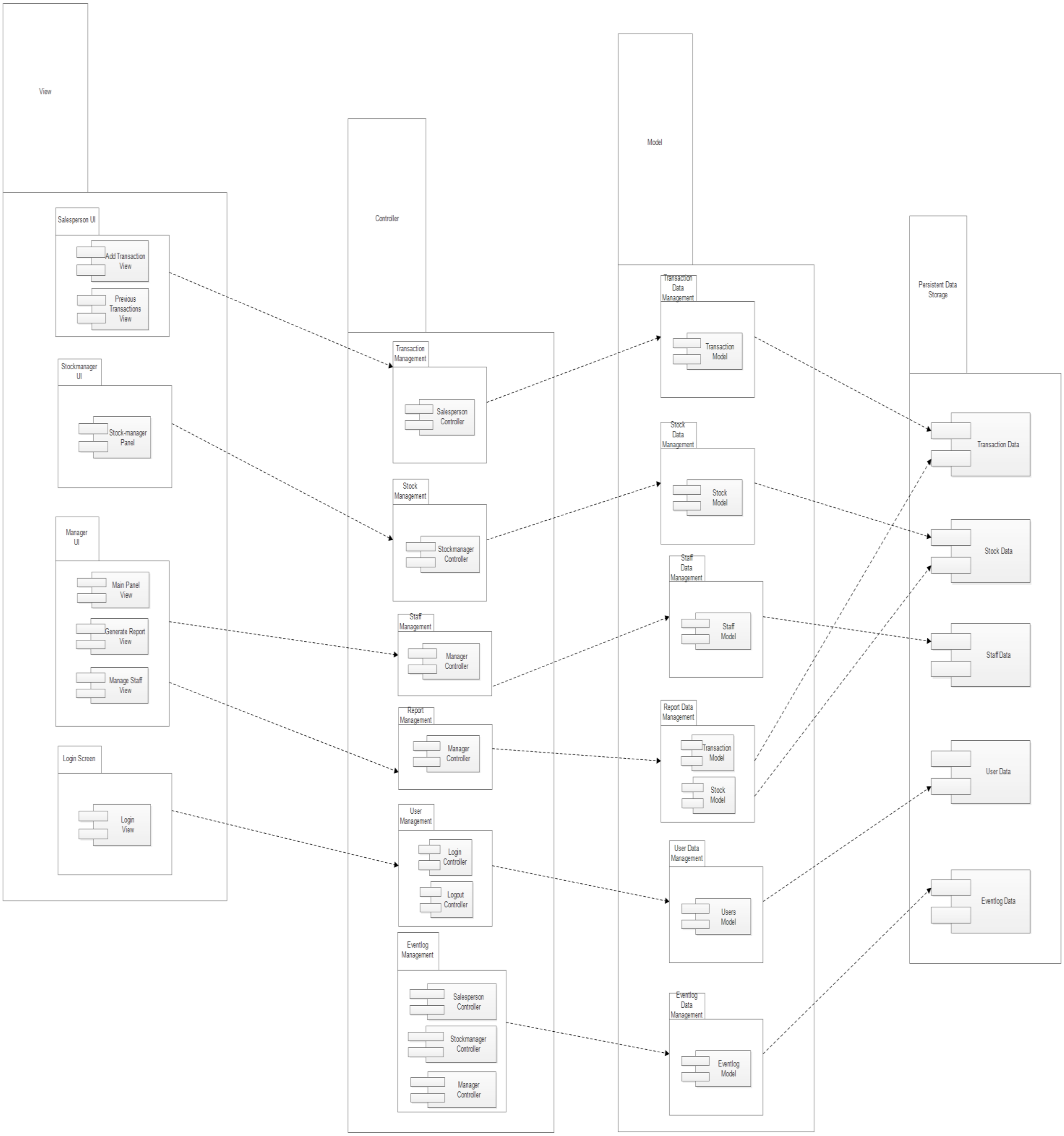

Figure 3 - Layer 3 Component Diagram

2.2 Hardware/Software mapping

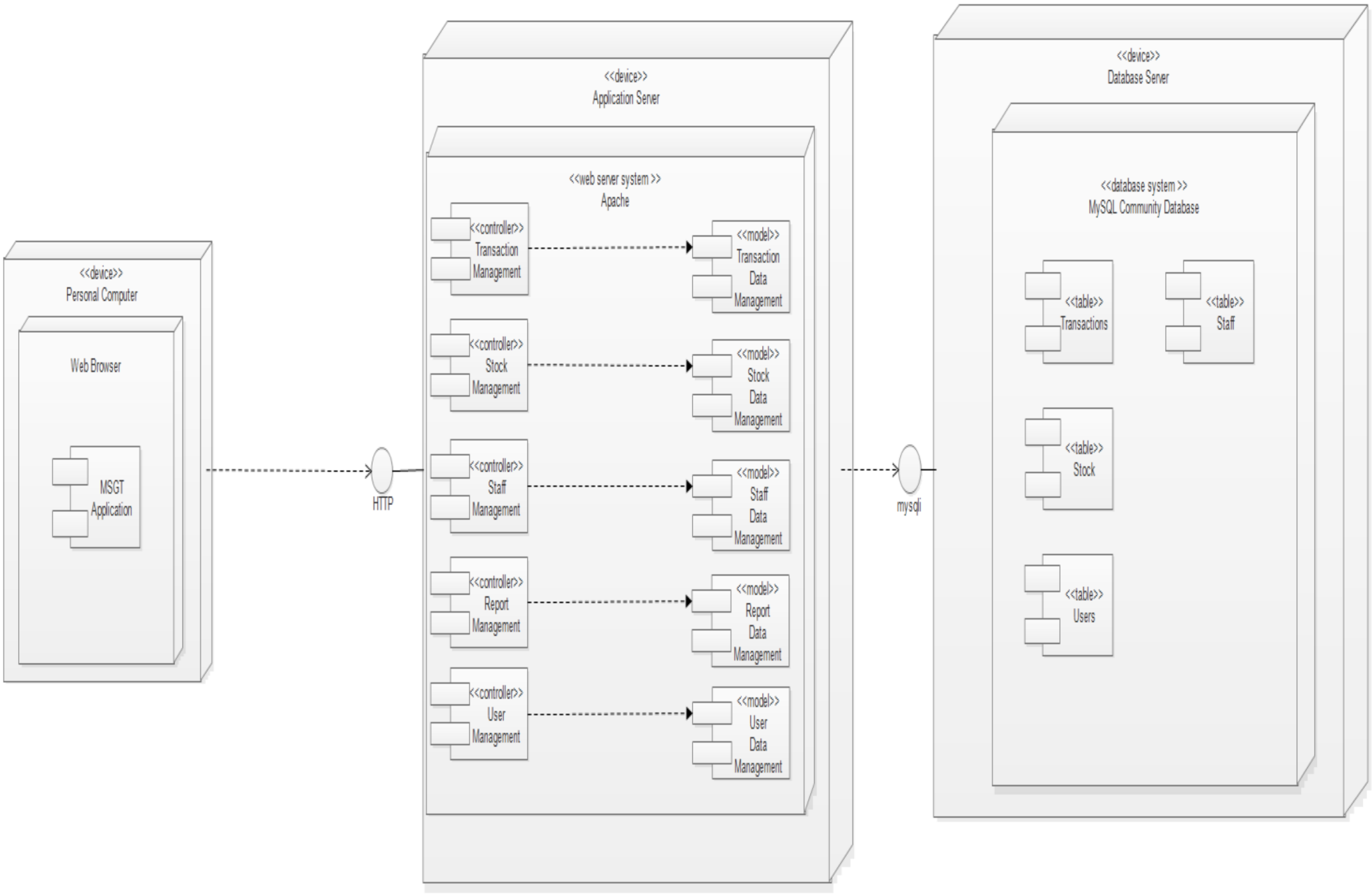
<<device>>
Personal Computer
Web Browser
MSGT Application
HTTP
<<device>>
Application Server
<<web server system >>
Apache
<<controller>> Transaction Management
<<model>> Transaction Data Management
<<controller>> Stock Management
<<model>> Stock Data Management
<<controller>> Staff Management
<<model>> Staff Data Management
<<controller>> Report Management
<<model>> Report Data Management
<<controller>> User Management
<<model>> User Data Management
mysqli
<<device>>
Database Server
<<database system >>
MySQL Community Database
<<table>> Transactions
<<table>> Staff
<<table>> Stock
<<table>> Users

Figure 4 - Deployment Diagram

3. Object Model

3.1 Class Diagram

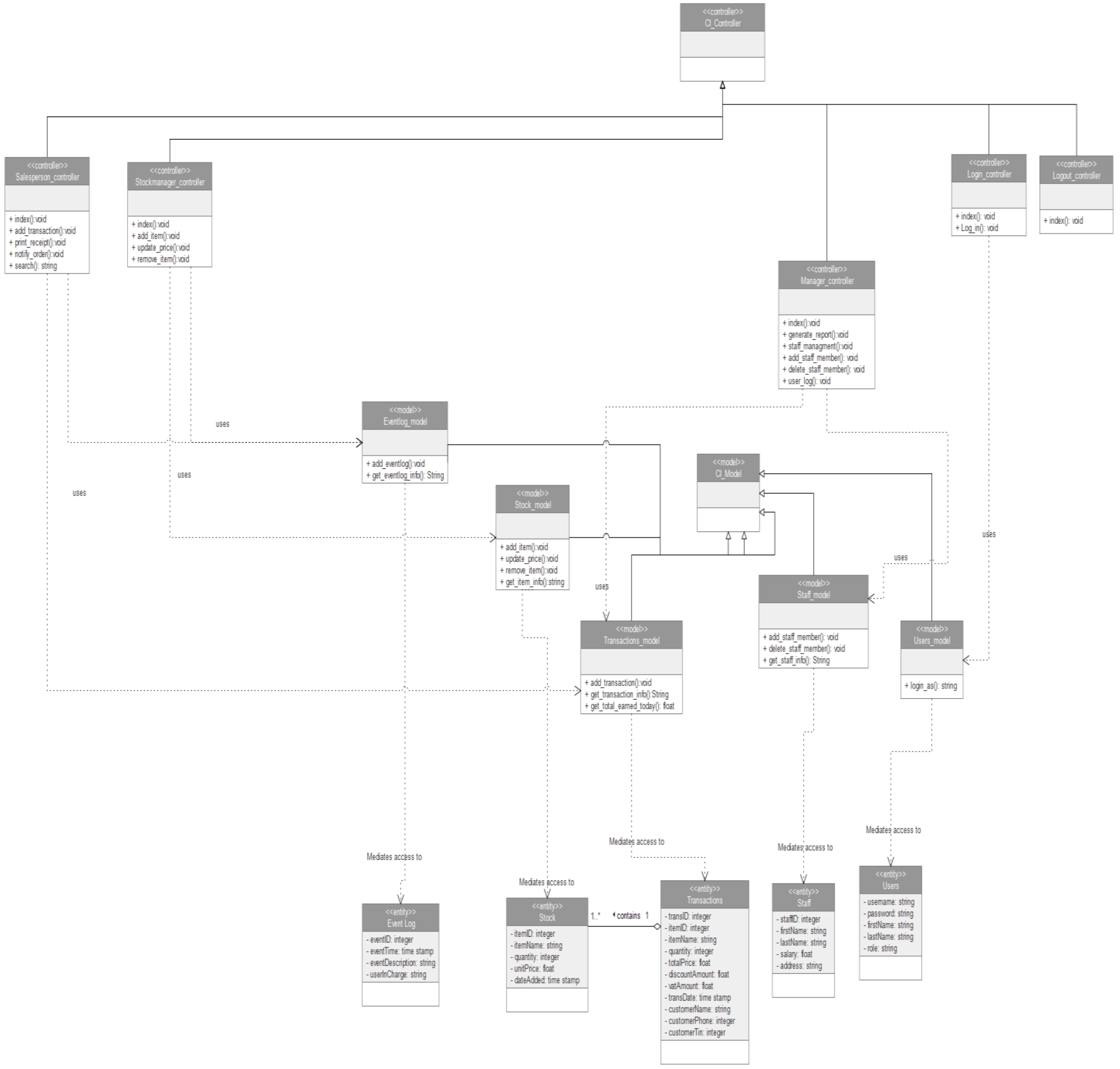

Figure 5 - Class Diagram

3.2 Sequence Diagram

3.2.1 Salesperson

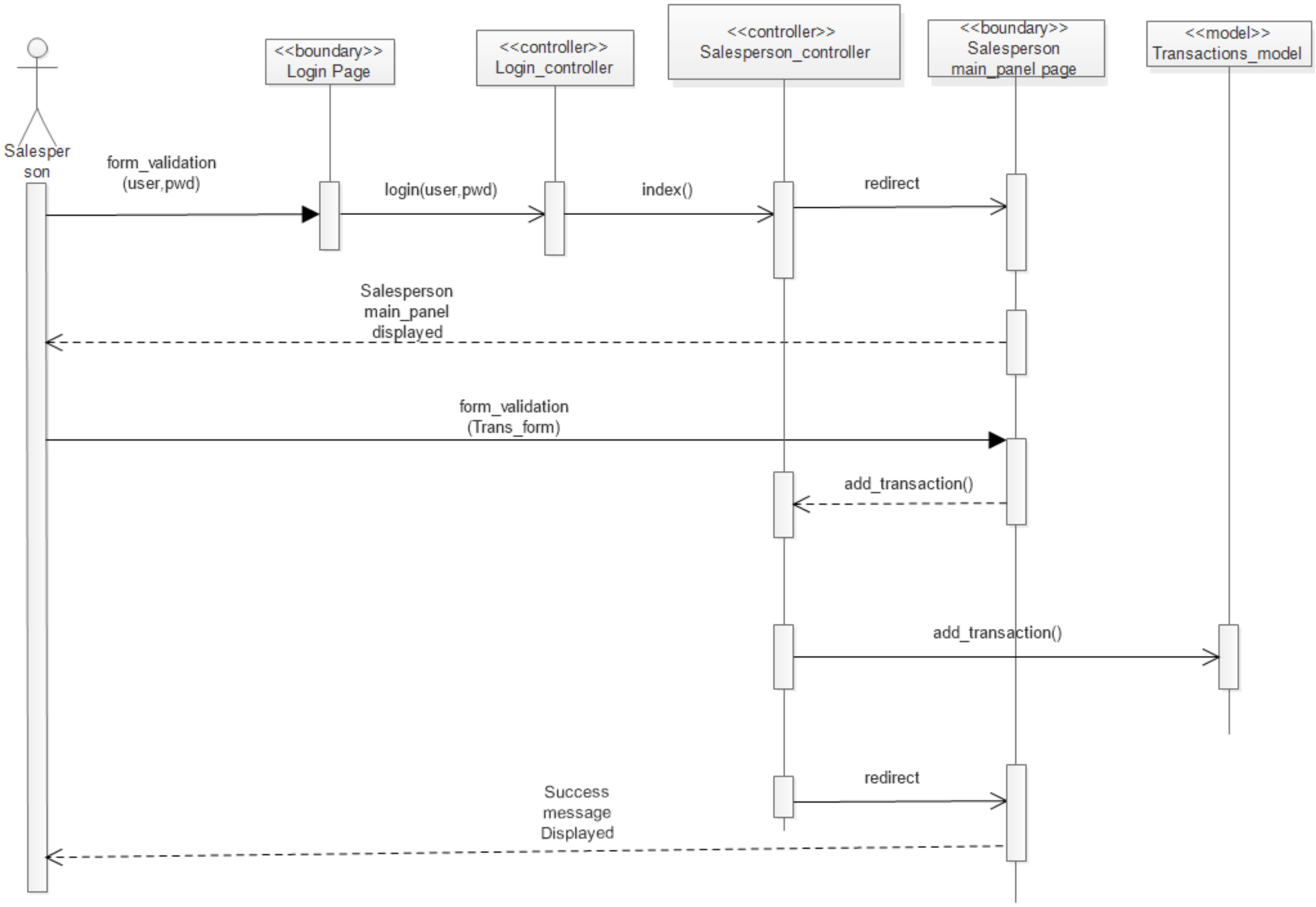

Figure 6 - Salesperson - Add Transactions Sequence Diagram

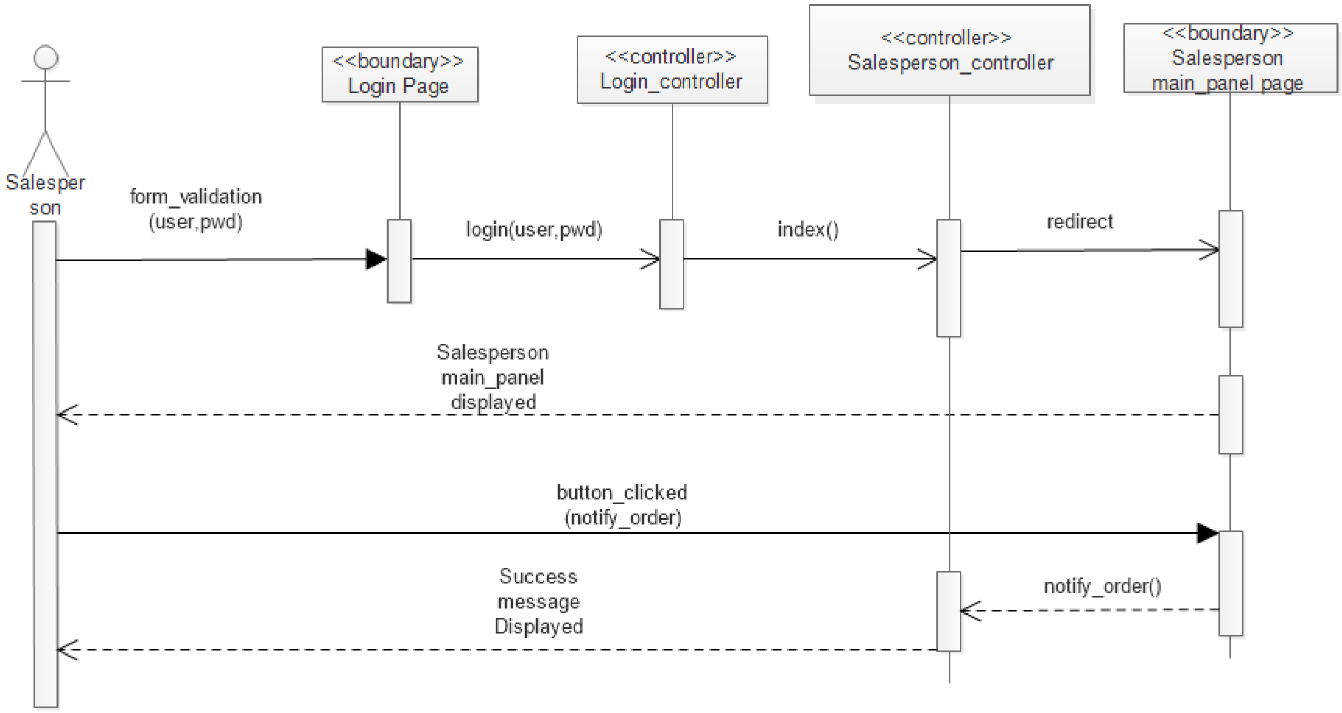

Figure 7 - Salesperson - Notify Order Sequence Diagram

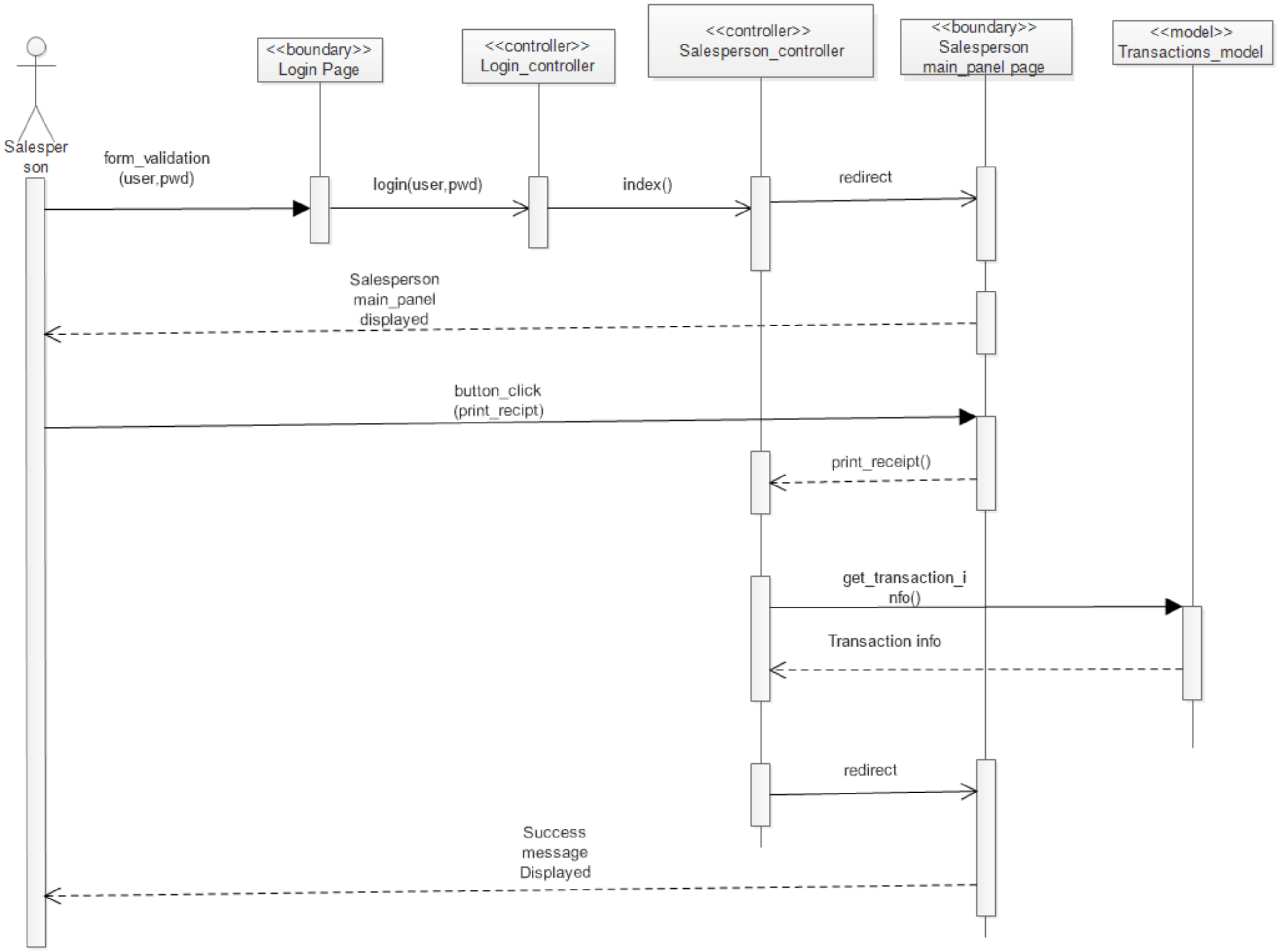

Figure 8 - Salesperson Print Receipt Sequence Diagram

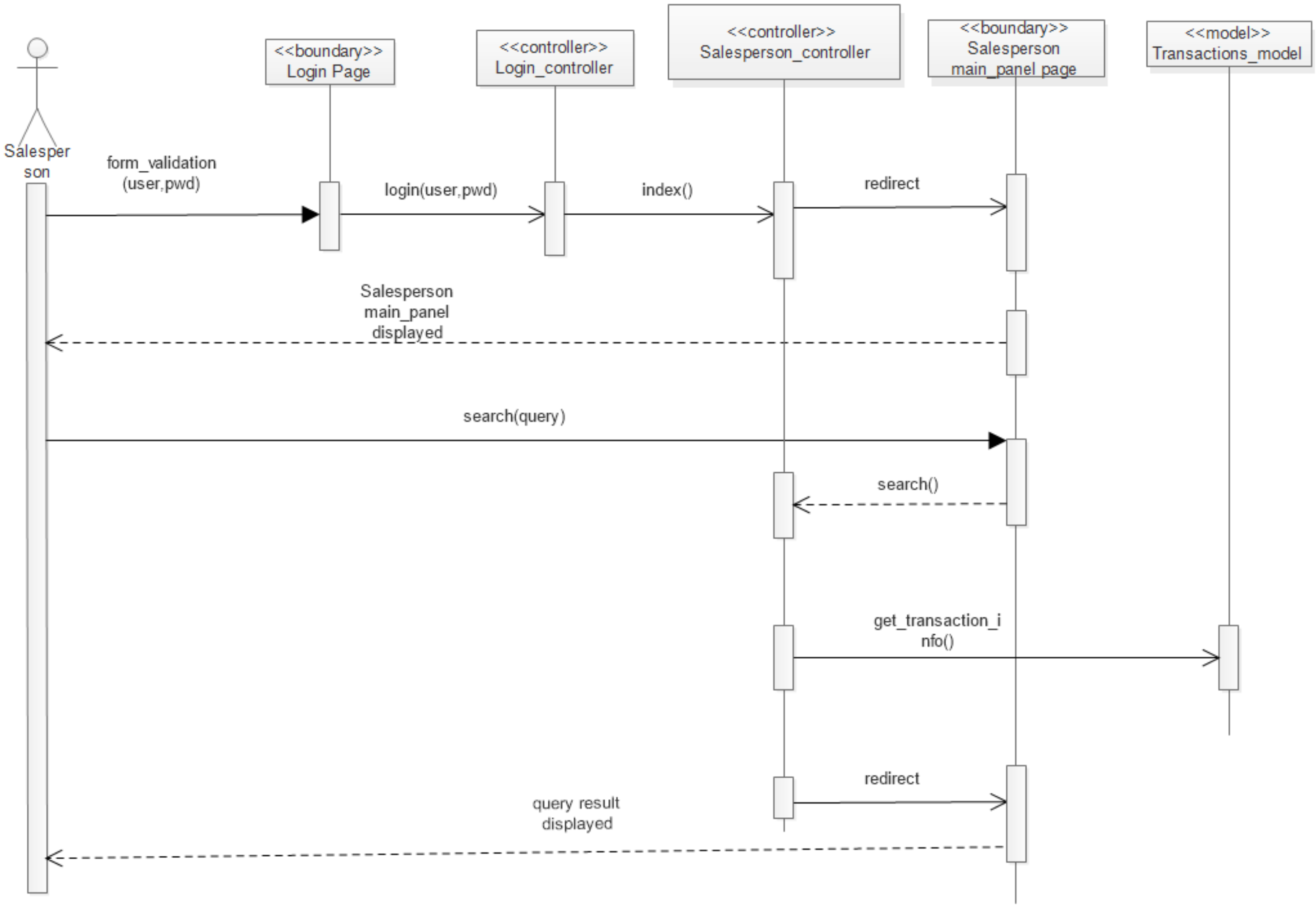
Salesperson
<<boundary>> Login Page
<<controller>> Login_controller
<<controller>> Salesperson_controller
<<boundary>> Salesperson main_panel page
<<model>> Transactions_model
form_validation (user,pwd)
login(user,pwd)
index()
redirect
Salesperson main_panel displayed
search(query)
search()
get_transaction_info()
redirect
query result displayed

Figure 9 - Salesperson Search Transactions Sequence Diagram

3.2.2 Stock-manager

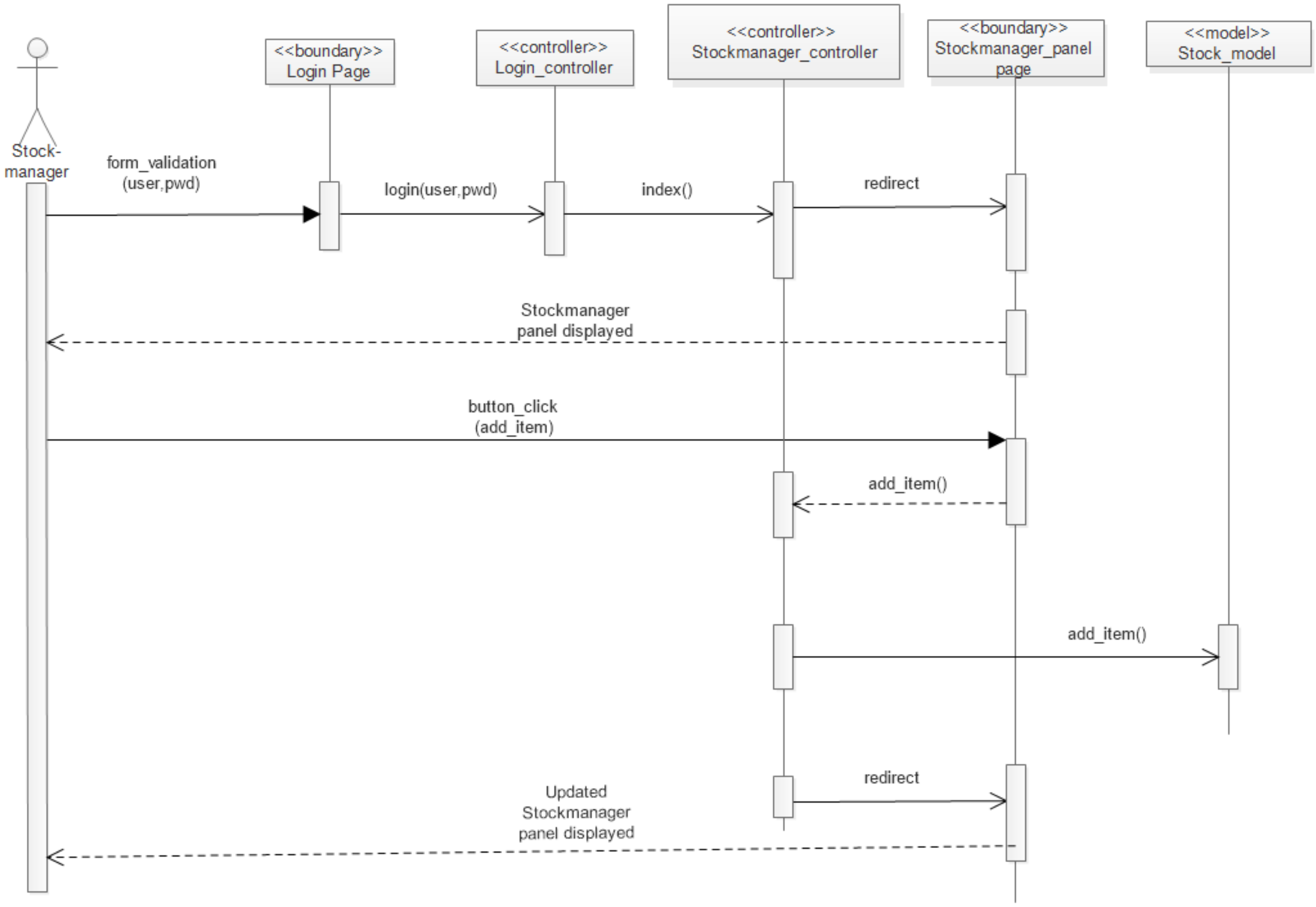

Figure 10 - Stockmanager - Add Item Sequence Diagram

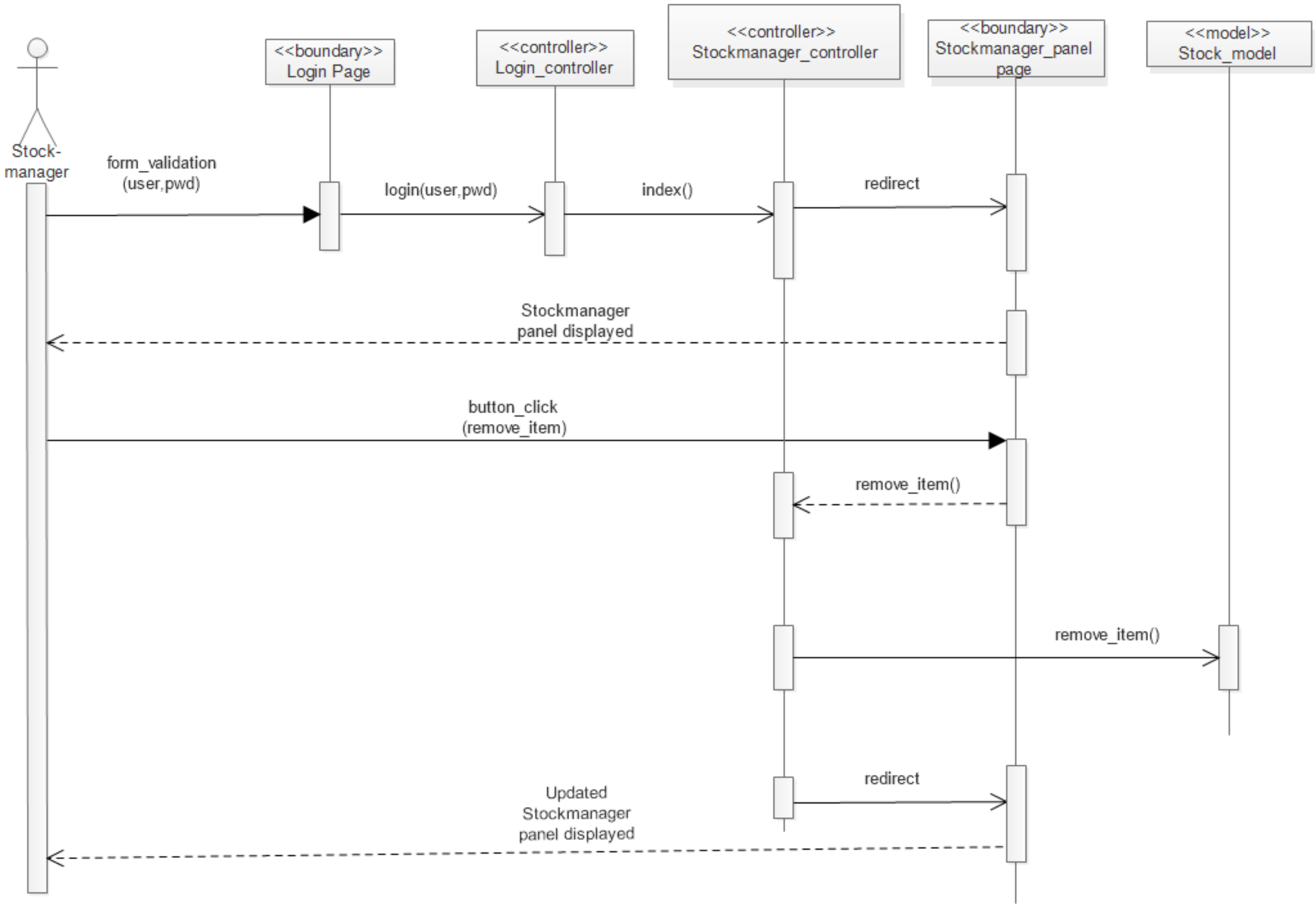

Figure 11 - Stockmanager - Remove Item Sequence Diagram

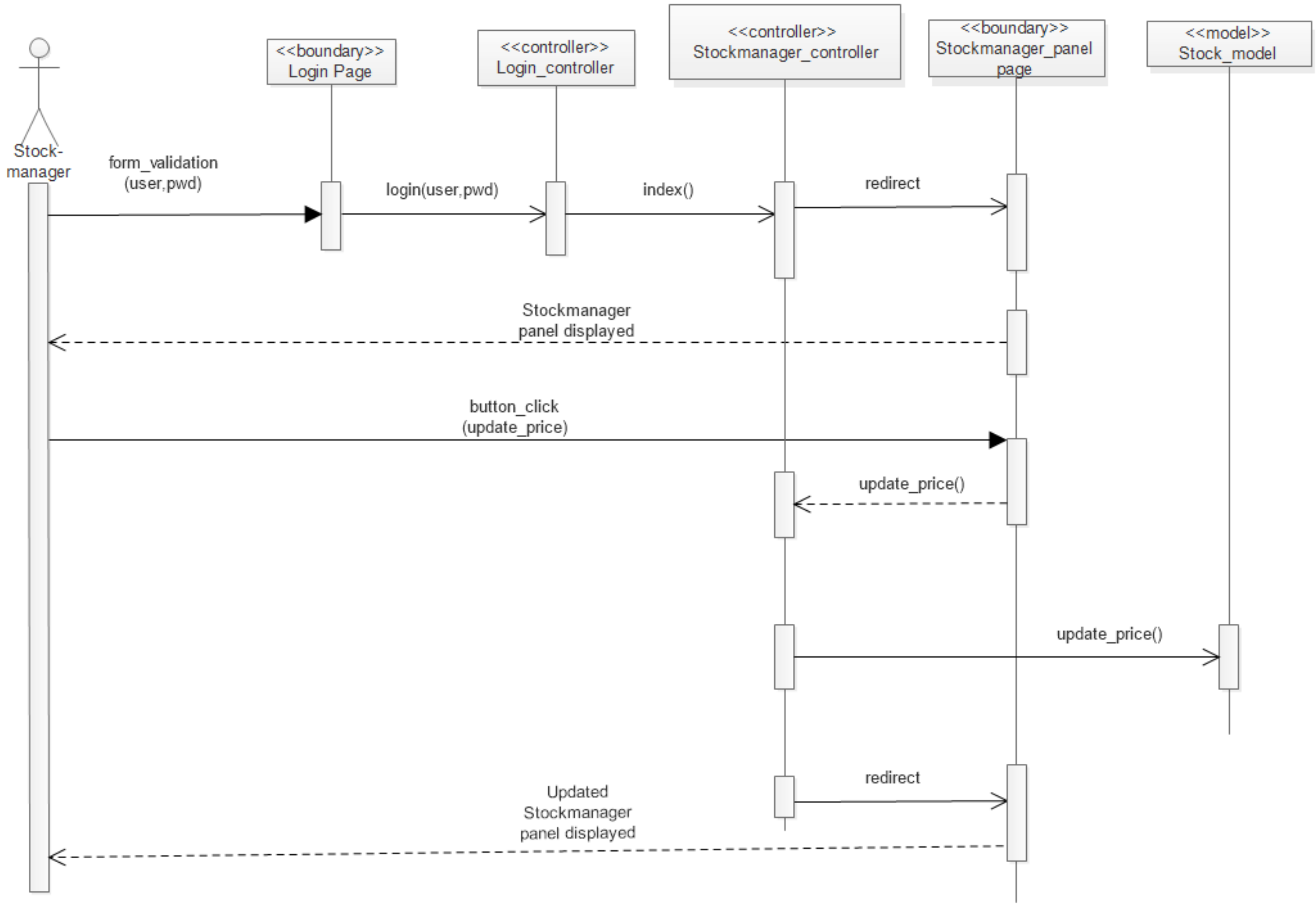

Figure 12 - Stockmanager - Update Price Sequence Diagram

3.2.3 Manager

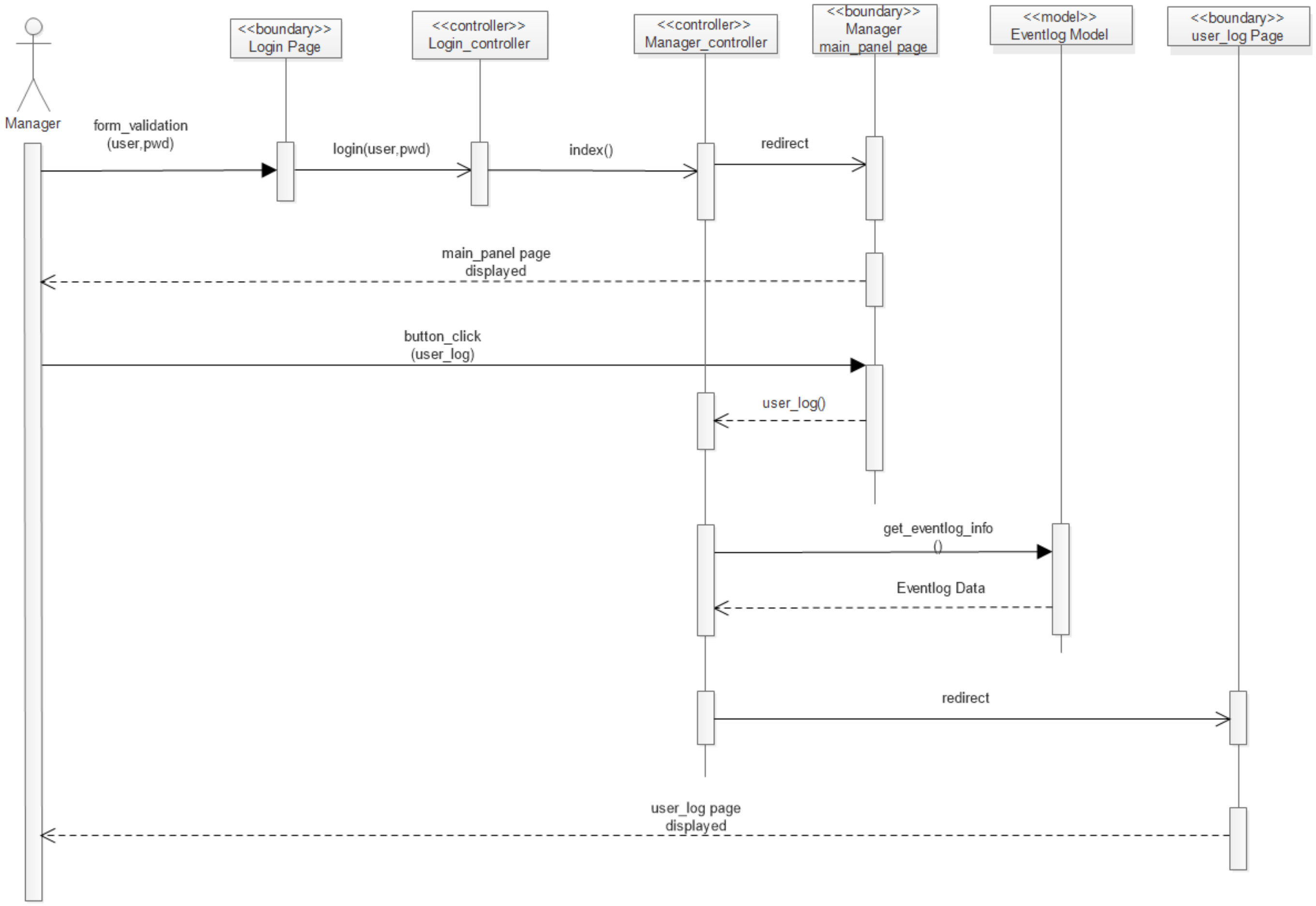

Figure 13 - Manager - View User Log Sequence Diagram

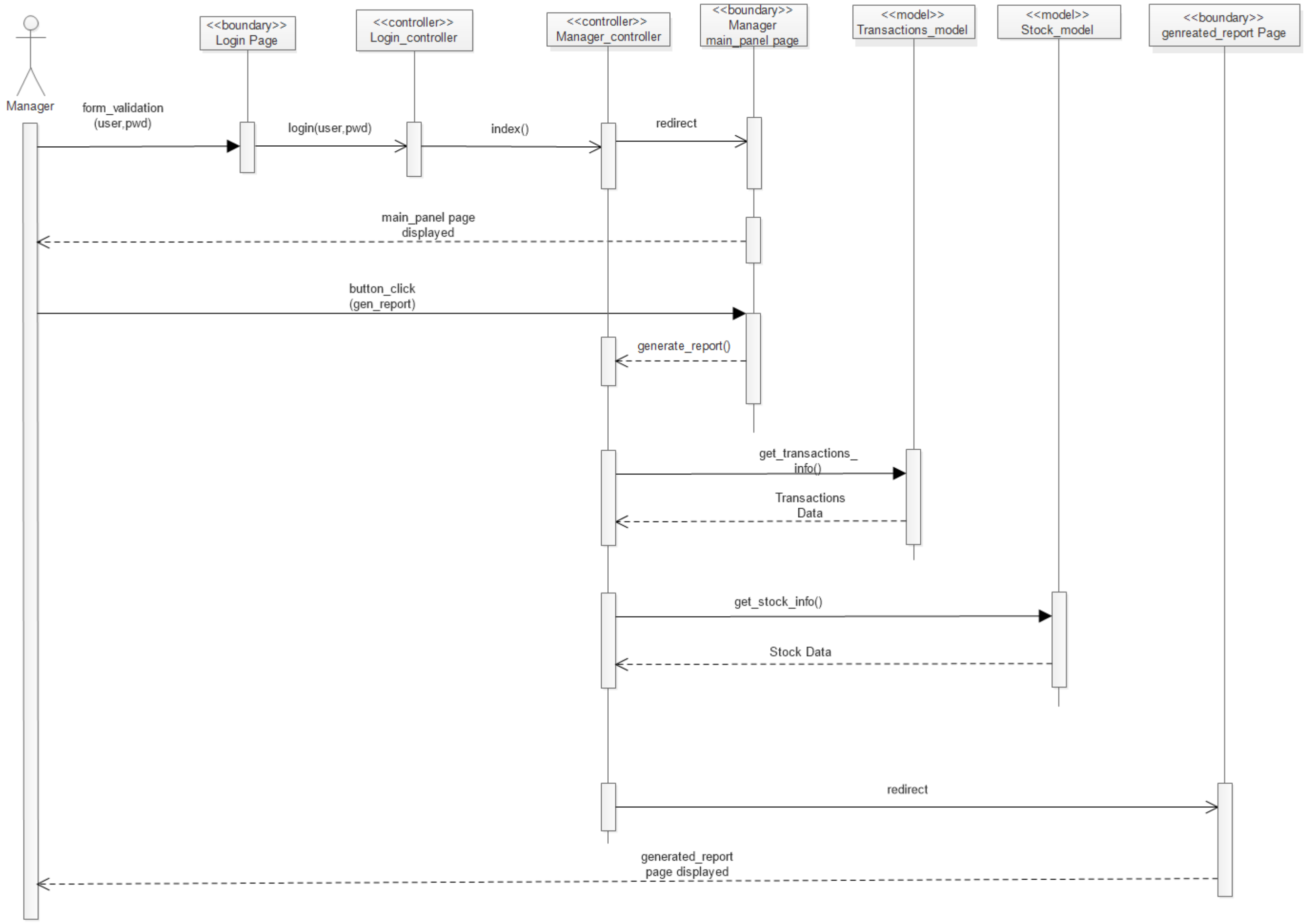
Manager
<<boundary>> Login Page
<<controller>> Login_controller
<<controller>> Manager_controller
<<boundary>> Manager main_panel page
<<model>> Transactions_model
<<model>> Stock_model
<<boundary>> genreated_report Page
form_validation (user,pwd)
login(user,pwd)
index()
redirect
main_panel page displayed
button_click (gen_report)
generate_report()
get_transactions_ info()
Transactions Data
get_stock_info()
Stock Data
redirect
generated_report page displayed

Figure 14 - Manager - Generate Report Sequence Diagram

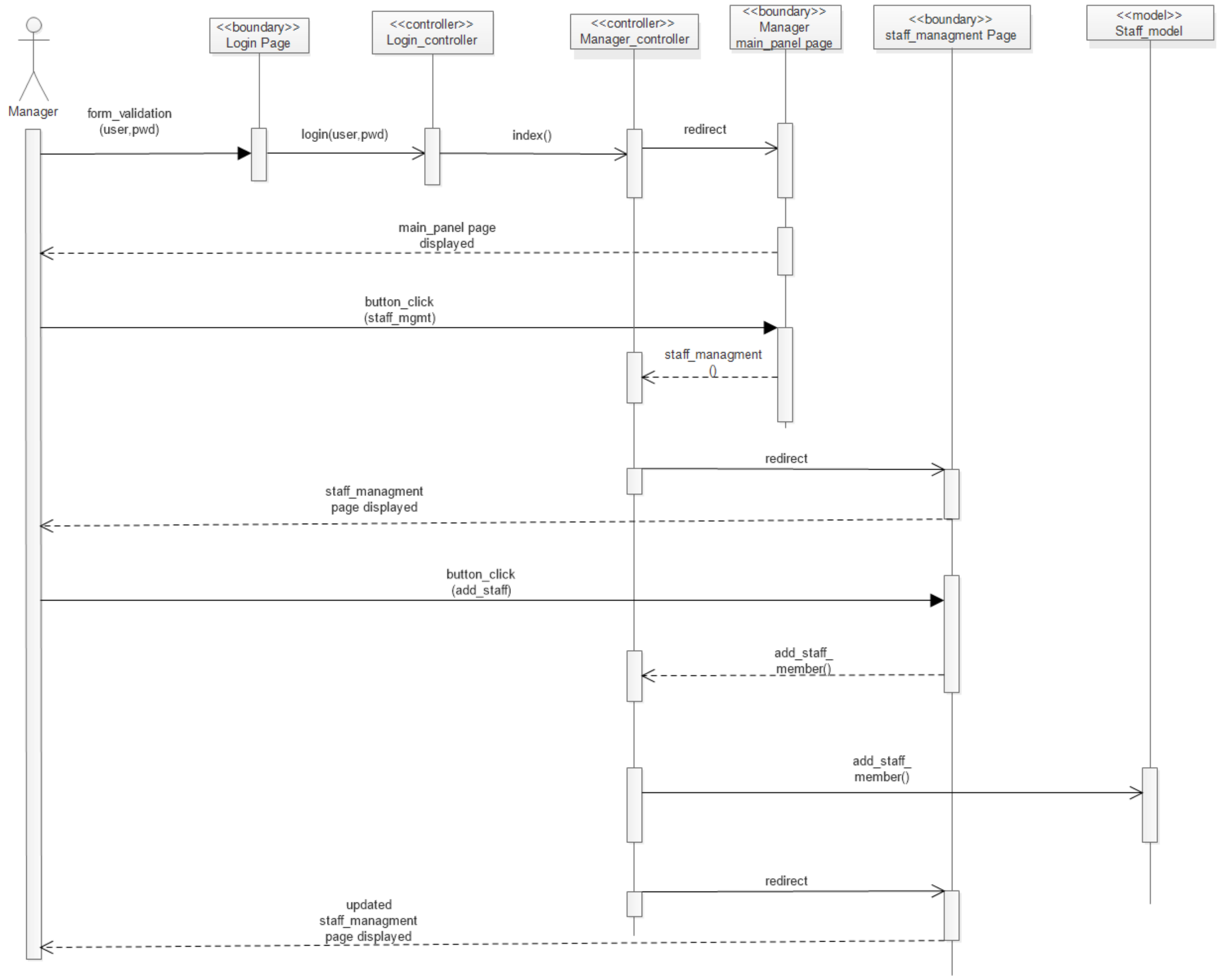

Figure 15 - Manager - Add Staff member Sequence Diagram

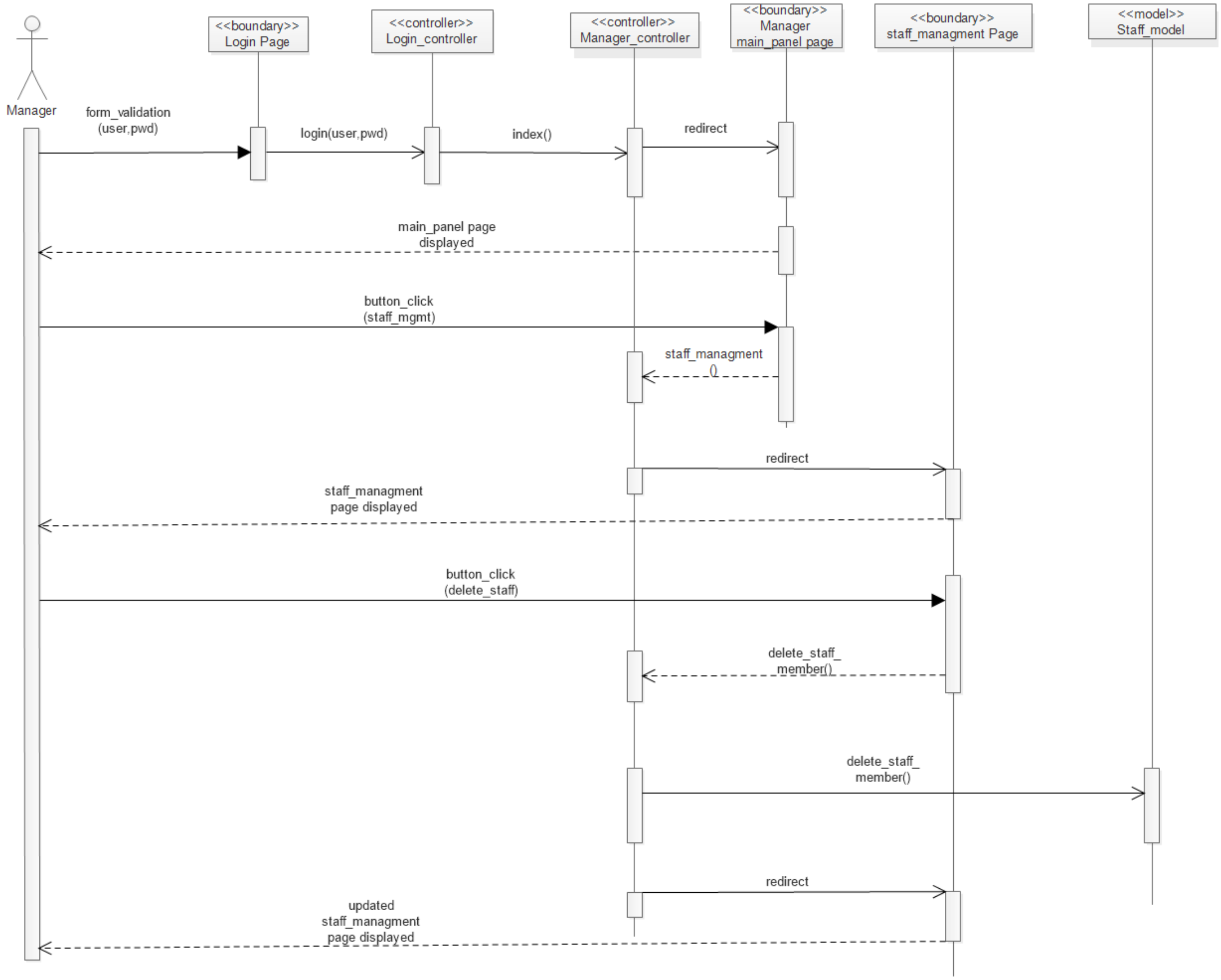

Figure 16 - Manager - Delete Staff member Sequence Diagram

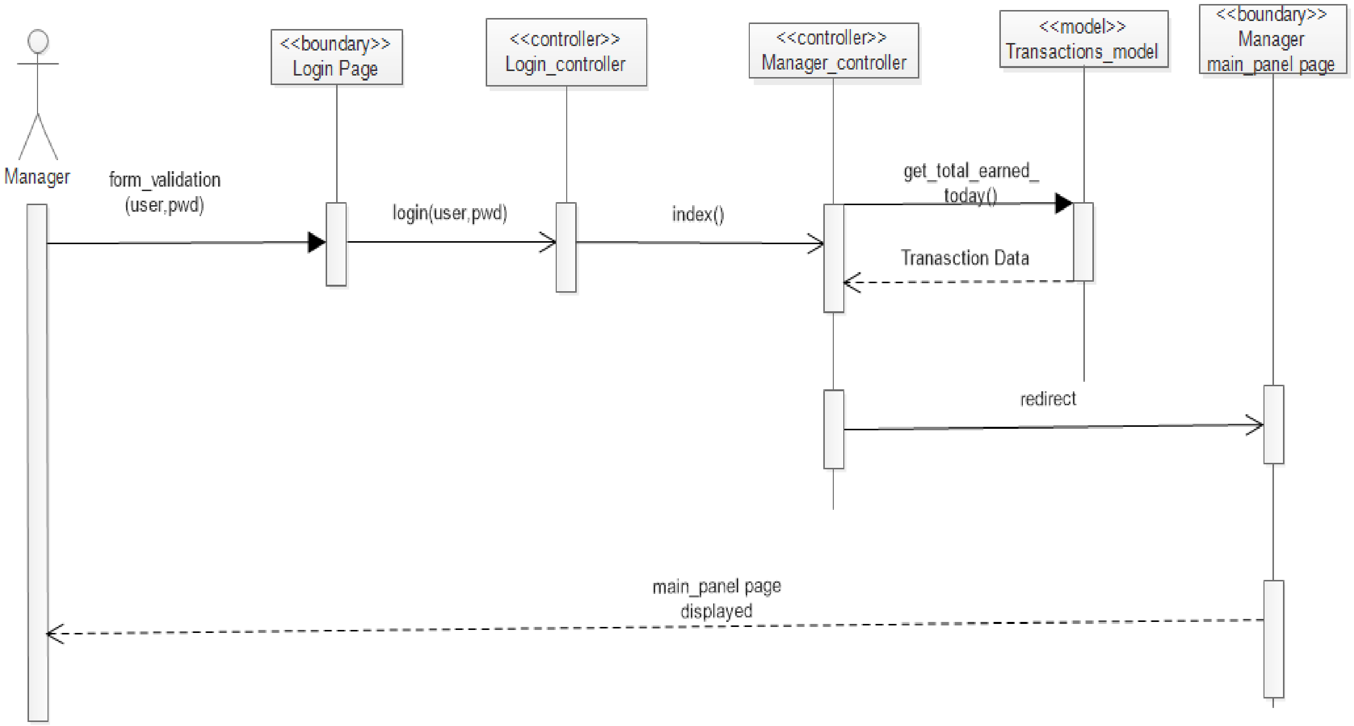

Figure 17 - Manager - View Real-time Sales Activity Sequence Diagram

4. Detailed Design

4.1 Users Class

Table 2 - General Description of Users Class

<<entity>> **Users**
- username: string - password: string - first_name: string - last_name: string - role: string

Table 3 - Attributes description for Users Class

Attribute	Type	Visibility	Invariant
username	String	Private	Username <> NULL, must contain no more than 60 characters and must be unique
password	String	Private	Password <> NULL and must contain no more than 60 characters
First_name	String	Private	First_name <> NULL and must contain no more than 25 characters
Last_name	String	Private	Last_name <> NULL and must contain no more than 25 characters
role	String	Private	Role <> NULL and must contain no more than 5 characters

4.2 Transactions Class

Table 4 - General Description of Transactions Class

<<entity>> **Transactions**
- trans_id: integer - item_id: integer - item_name: string - quantity: integer - total_price: float - discount_amount: float - vat_amount: float - trans_date: date - customer_name: string - customer_phone: integer - customer_tin: integer

Table 5 - Attributes description for Transactions Class

Attribute	Type	Visibility	Invariant
transID	integer	Private	transID <> NULL
itemID	integer	Private	itemID <> NULL
itemName	String	Private	itemName <> NULL and must contain no more than 25 characters
quantity	integer	Private	Quantity <> NULL
total_price	float	Private	total_price <> NULL
total_price	float	Private	total_price <> NULL
discount_amount	float	Private	discount_amount <> NULL
vat_amount	float	Private	vat_amount <> NULL
trans_date	Date	Private	trans_date <> NULL
customer_name	String	Private	customer_name <> NULL
customer_phone	integer	Private	customer_phone <> NULL
customer_tin	integer	Private	customer_tin <> NULL

4.3 Stock Class

Table 6 - General Description of Stock Class

<<entity>> **Stock**
- item_id: integer - item_name: string - quantity: integer - unit_price: float - date_added: time stamp

Table 7 - Attributes description for Stock Class

Attribute	Type	Visibility	Invariant
item_id	Integer	Private	item_id <> NULL
item_name	String	Private	item_name <> NULL and must contain no more than 25 characters
quantity	Integer	Private	Quantity <> NULL
unit_price	float	Private	unit_price <> NULL
date_added	Date	Private	date_added <> NULL

4.4 Staff Class

Table 8 - General Description of Staff Class

<<entity>> **Staff**
- staff_id: integer - first_name: string - last_name: string - salary: float - address: string

Table 9 - Attributes description for Staff Class

Attribute	Type	Visibility	Invariant
staff_id	integer	Private	staff_id <> NULL
first_name	String	Private	first_name <> NULL and must contain no more than 25 characters
last_name	String	Private	last_name <> NULL and must contain no more than 25 characters
Salary	float	Private	Salary <> NULL

Address	String	Private	Address <> NULL and must contain no more than 100 characters

4.5 Event log Class

Table 10 - General Description of Eventlog Class

<<entity>> **Eventlog**
- event_id: integer - event_time: date - event_description: string - user_in_charge: string

Table 11 - Attributes description for Eventlog Class

Attribute	Type	Visibility	Invariant
Event_id	integer	Private	Event_id <> NULL
Event_time	Date	Private	Event_time <> NULL
Event_description	String	Private	Event_description <> NULL and must contain no more than 100 characters
User_in_charge	String	Private	User_in_charge <> NULL and must contain no more than 60 characters

4.6 Users_model Class

Table 12 - General Description of Users_model Class

<<model>> **Users_model**
+ login_as(): string

Table 13 - Operation Description for Users_model Class

Operation	Visibility	Return type	Argument	Pre-Condition	Post-Condition
login_as()	Public	String	- Username: String - Password: String	The user must be at the login screen	User type should be returned

4.7 Transactions_model Class

Table 14 - General Description of Transactions_model Class

<<model>> **Transactions_model**
+ add_transaction():void + get_transaction_info():String + get_total_earned_today(): Float

Table 15 - Operation Description for Transactions_model Class

Operation	Visibility	Return type	Argument	Pre-Condition	Post-Condition
add_transaction()	Public	Void	- itemName - quantity - total_price - total_price - discount_amount - vat_amount - trans_date - customer_name - customer_phone - customer_tin	Transaction should not exist	Transaction should exist
get_transaction_info()	Public	String	- Transaction_id	Transaction should exist	Transaction information should be returned
get_total_earned_today()	Public	Float	- void	Transaction should exist	Total earning of the day

					should be returned

4.8 Stock_model Class

Table 16 - General Description of Stock_model Class

<<model>> **Stock_model**
+ add_item():void + update_price():void + remove_item():void + get_item_info():string

Table 17 - Operation Description for Stock_model Class

Operation	Visibility	Return type	Argument	Pre-Condition	Post-Condition
add_item()	Public	Void	- item_name - quantity - unit_price - date_added	Item should not exist	Item should exist
update_price()	Public	Void	- Price - item_id	Item should exist	Price of the item should be updated
remove_item()	Public	Void	- item_id	Item should exits	Item should not exist
get_item_info()	Public	String	- item_id	Item should exits	Item information should be returned

4.9 Staff_model Class

Table 18 - General Description of Staff_model Class

<<model>>

Staff_model
+ add_staff_member(): void + delete_staff_member(): void + get_staff_info(): String

Table 19 - Operation Description for Staff_model Class

Operation	Visibility	Return type	Argument	Pre-Condition	Post-Condition
add_staff_member()	Public	Void	- first_name - last_name - Salary - Address	Staff member should not exist	Staff member should exist
delete_staff_member()	Public	Void	- Staff_id	Staff member should exist	Staff member should not exist
get_staff_info()	Public	String	- Staff_id	Staff member should exist	Staff member information should be returned

4.10 Eventlog_model Class

Table 20 - General Description of Eventlog_model Class

<<model>> **Eventlog_model**
+ add_eventlog():void + get_eventlog_info(): String

Table 21 - Operation Description for Eventlog_model Class

Operation	Visibility	Return type	Argument	Pre-Condition	Post-Condition

add_eventlog()	Public	Void	- Event_time - Event_description - User_in_charge	Eventlog should not exist	Eventlog should exist
get_eventlog_info()	Public	String	- Event_id	Eventlog should exist	Eventlog information should be returned

4.11 Stockmanager_controller Class

Table 22 - General Description of Stockmanager_controller Class

<<controller>> **Stockmanager_controller**
+ index():void + add_item():void + update_price():void + remove_item():void

Table 23 - Operation Description for Stockmanager_controller Class

Operation	Visibility	Return type	Argument	Pre-Condition	Post-Condition
index()	Public	Void	Void	User should be logged in as a stock-manager	Default stockmanager view should be displayed
add_item()	Public	Void	Void	User should be logged in as a stock-manager	Item should exist
update_price()	Public	Void	Void	User should be logged in as a stock-manager	Price of the item should be updated
remove_item()	Public	Void	Void	User should be logged in as a stock-manager	Item should not exist

4.12 Salesperson_controller Class

Table 24 - General Description of Salesperson_controller Class

<<controller>> **Salesperson_controller**
+ index():void + add_transaction():void + print_receipt():void + notify_order():void + search():String

Table 25 - Operation Description for Salesperson_controller Class

Operation	Visibility	Return type	Argument	Pre-Condition	Post-Condition
index()	Public	Void	Void	User should be logged in as a salesperson	Default salesperson view should be displayed
add_transaction()	Public	Void	Void	User should be logged in as a salesperson	Transaction should exist
print_receipt()	Public	Void	Void	User should be logged in as a salesperson	Receipt should be printed
notify_order()	Public	Void	Void	User should be logged in as a salesperson	Order should be notified
search()	Public	Void	- query	User should be logged in as a salesperson	Search results should be displayed

4.13 Manager_controller Class

Table 26 - General Description of Manager_controller Class

<<controller>> **Manager_controller**
+ index():void + generate_report():void + staff_managment():void + add_staff_member(): void + delete_staff_member(): void + user_log(): void

Table 27 - Operation Description for Manager_controller Class

Operation	Visibility	Return type	Argument	Pre-Condition	Post-Condition
index()	Public	Void	Void	User should be logged in as a manager	Default manager view should be displayed
generate_report()	Public	Void	Void	User should be logged in as a manager	Report should be generated
staff_managment()	Public	Void	Void	User should be logged in as a manager	Staff management view should be displayed
add_staff_member()	Public	Void	Void	User should be logged in as a manager	Staff member should exist
delete_staff_member()	Public	Void	Void	User should be logged in as a manager	Staff member should not exist

user_log()	Public	Void	Void	User should be logged in as a manager	User log view should be displayed

4.14 login_controller Class

Table 28 - General Description of Login_controller Class

<<controller>> **Login_controller**
+ index(): void + Login(): void

Table 29 - Operation Description for Login_controller Class

Operation	Visibility	Return type	Argument	Pre-Condition	Post-Condition
index()	Public	Void	Void		Login view should be displayed
Login():	Public	Void	Void	User session should not exist	User session should exist

4.15 logout_controller Class

Table 30 - General Description of Logout_controller Class

<<controller>> **Logout_controller**
+ index(): void

Table 31 - Operation Description for Logout_controller Class

Operation	Visibility	Return type	Argument	Pre-Condition	Post-Condition
index()	Public	Void	Void	User session should exist	User session should not exist

REFERENCES

- TutorialsTeacher, http://www.tutorialsteacher/mvc/mvc-architecture, May 25, 2018