



UTM
UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF ENGINEERING
SCHOOL OF COMPUTING
SCSJ3303 INTERNET PROGRAMMING
GROUP PROJECT - SECTION 02

Project Final Report

Project Title : Flying Wheel

Lecturer : Dr Nor Azizah Binti Sa'adon

Group Name : Bluezone (Group 7)

Group Members :

No.	Name	Matric Number
1	Bellarina Chew Li Yen	A17CS0026
2	Khaalid Subaan Buraaleh	A17CS4037
3	Muhammad Mohsin Shaikh	A17CS4007
4	Nurul Nadhifah As Syahira Binti Ahmad Fakarudin	A17CS0185

SUBMISSION DATE : 31/12/2019

TABLE OF CONTENT	
1.0 Introduction 1.1 Project Title 1.2 Project overview 1.3 Problem Background 1.4 Proposed Solution 1.5 System Module	3
2.0 System Design 2.1 Use case diagram 2.2 Activity Diagram	6
3.0 Detailed Design 3.1 Book a bicycle 3.2 Order item 3.3 Financial Report	9
4.0 Reflection	

1.0 Introduction

1.1 Project Title

Flying Wheel Bicycle Management System

1.2 Overview

The product to be produced is Flying Wheel, a web application for Bicycle Management System that provides a facility for booking bicycles and buying its equipment. This product helps bicycle shop owners to manage bicycle shop systematically.

1.3 Problem Background

UTM Bicycle shop has been growing over the years and the number of customers kept increasing. They have been using the manual method to record shop activities since then such as bicycle rental. Due to the increasing number of customers, they are facing difficulties in :

- Keeping track on unreturned bicycles
- Keeping track of late returned bicycles
- Keeping track on bicycles and bicycle equipment stocks

On the other hand, the customers are facing the inconvenience of pre-booking a bicycle. If they wish to rent a bicycle, they will have to come at the exact date and time they wish to use the bicycle and hope they will find one that is to their liking. If they ran out of bicycles, it will be a waste of time and energy to go all the way and unable to rent a bicycle.

It is also inconvenient to buy bicycle equipment. Customers have to manually be there and it would be such a waste of time to do so if the equipment is not available. Manually, customer can request for shop manager to restock. However, customer will have to go there time by time for update.

1.4 Proposed Solution

To develop a web application known as Flying Wheel to automate all the process done in the Bicycle Shop. This helps the bicycle manager in making sure that the shop management system is eased and well-managed. Other than that, all records can also be kept in an organized manner for future use.

To book a bicycle, customers can fill up a form in the system with information including name, date, pickup and return time. Afterward, the bicycle manager can view the booking to prepare the bicycle. The customer shall pick up a reserved bicycle at the bicycle shop and payment shall be done at the bicycle shop.

Flying Wheel also acts as an online shopping platform to help Bicycle Manager to keep track of bicycle equipment stocks. Items information could be uploaded to the web application for sale. Sold items will be recorded for sales and financial report generation.

This application allows Customer to pay for their ordered item with E-wallet. This ease the process of shopping online as Customer doesn't have to pay manually at the shop. It also benefits the shop from sudden cancellation made by Customer when they order items manually.

This application also allows bicycle manager to record every booking made in the bicycle shop rather than recording it in a book manually. As a result, this helps the Bicycle Manager to keep track of unreturned bicycles and late returned bicycles. There are three modules of the System: (1) **Administrator** (2) **Customer** (3) **Bicycle Manager**.

1.5 System Module

Administrator and Bicycle Manager should log-in to use the system using their username and password. Incorrect login will be rejected. Customers, on the other hand, can use the system with logging in with their registered account.

Administrator	<ul style="list-style-type: none">● Administrator can manage Bicycle Managers● Administrator can view all Customers.● Administrator can view all bicycles.● Administrator can view all items.● Administrator can view sales and financial reports.
Customer	<ul style="list-style-type: none">● Customer can register in the system● Customer can book a bicycle● Customer can view booking● Customer can order item● Customer can edit own profile● Customer can top up E-wallet
Bicycle Manager	<ul style="list-style-type: none">● Bicycle manager can view all customer rental booking● Bicycle manager can manage ordered items● Bicycle manager can manage items for sale● Bicycle manager can manage bicycle for rent● Bicycle manager can top up amount in Customers' account.

1.6 Tools and technology usage

- Servlet/JSP (Backend/Frontend)
- glassfish(local server)
- MySQL (database)
- Bootswatch (front end design)

2.0 System Design

2.1 Use Case Diagram

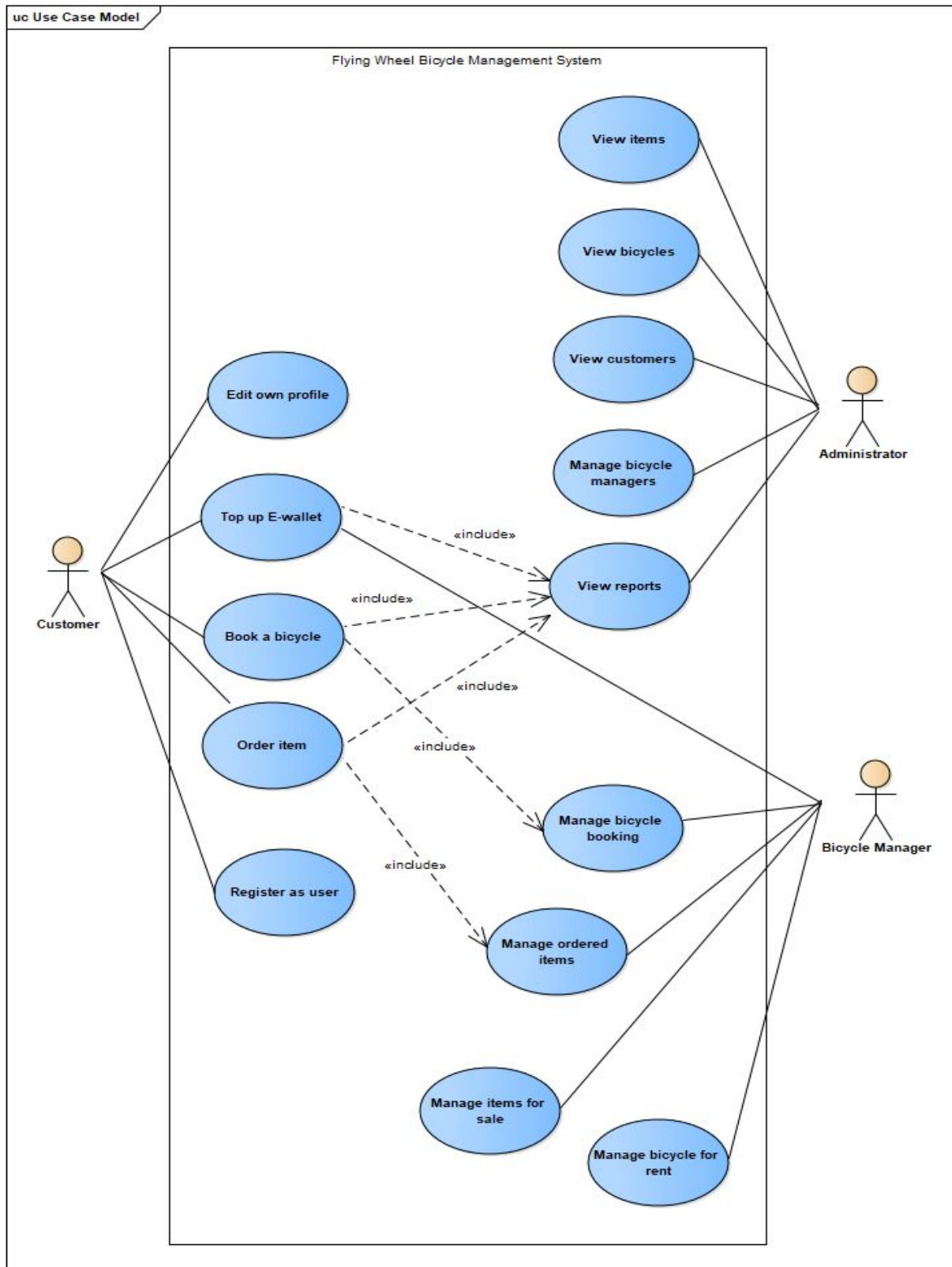


Diagram 2.1.0: Use case diagram for Flying Wheel Bicycle Management System

2.2 Activity Diagram

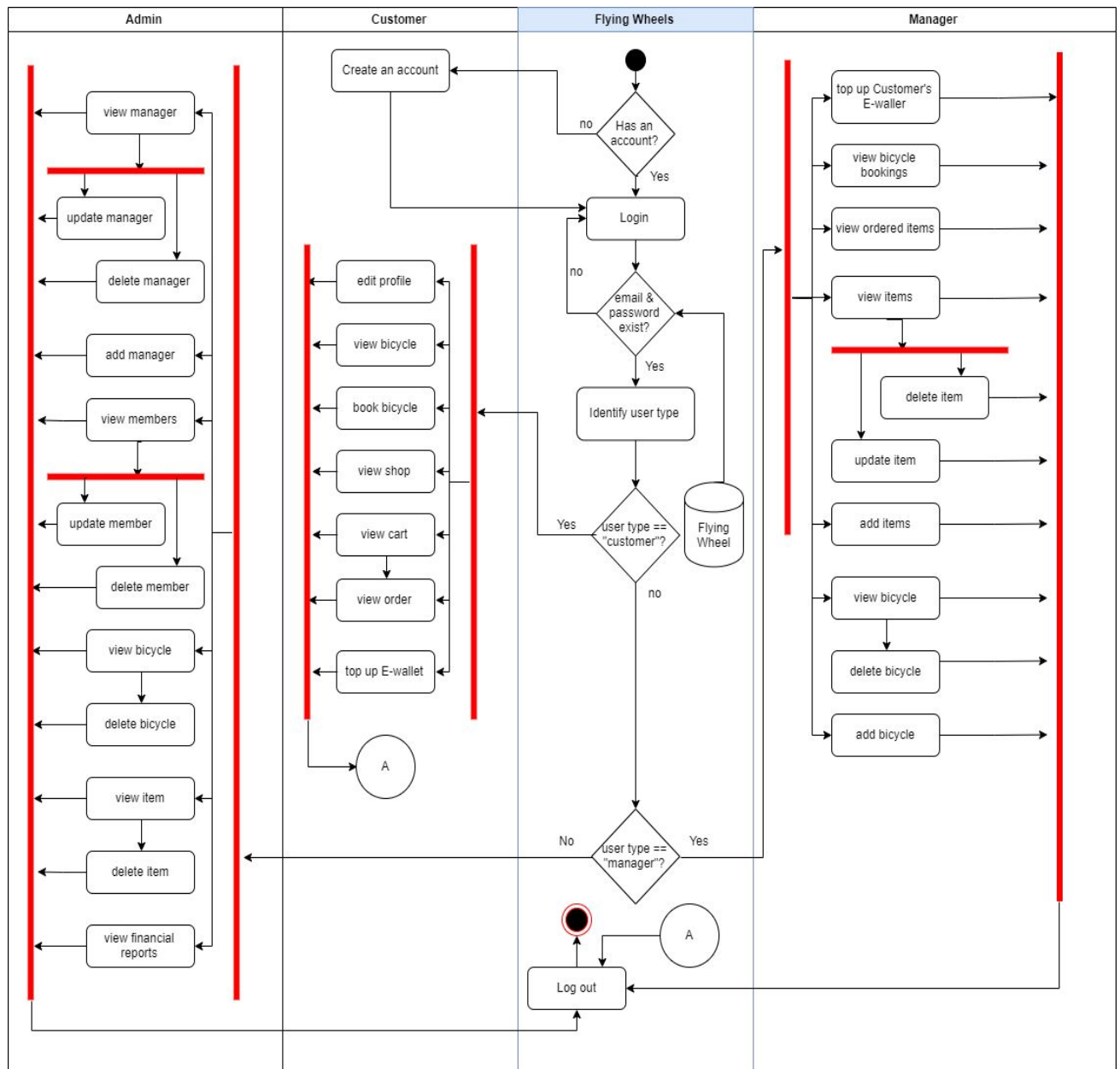


Diagram 2.1.1: Activity diagram for Flying Wheel Bicycle Management System

3.0 Detailed Description of Components

3.1 Use Case Book a bicycle

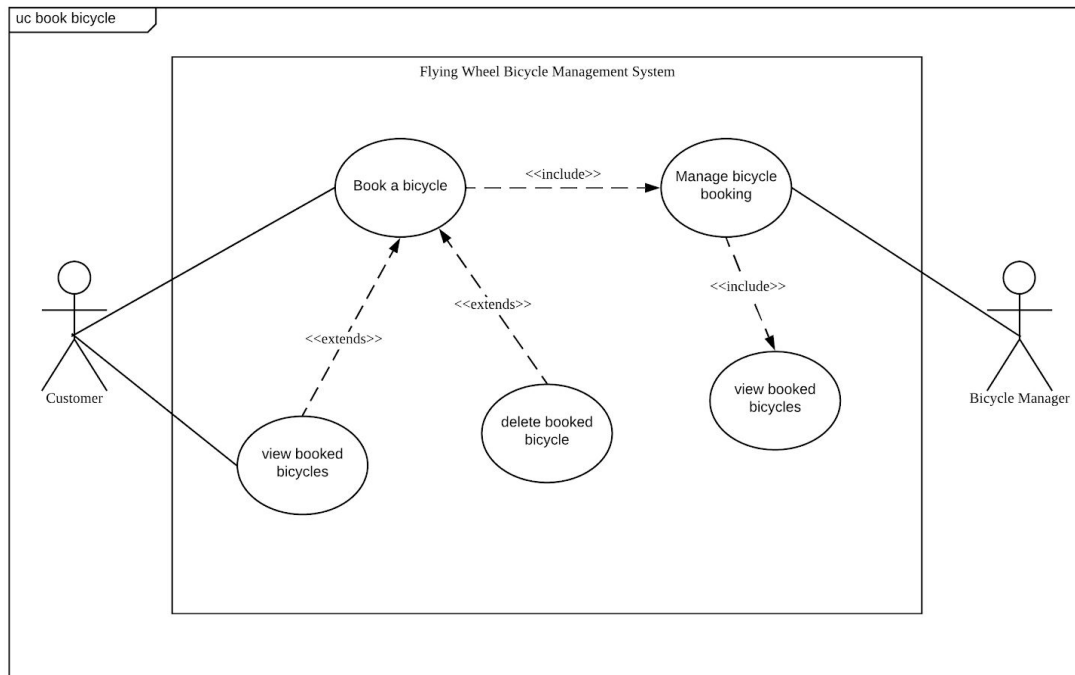


Diagram 3.1.0: Use case diagram for use case Book a Bicycle

Diagram 3.1.0 use case shows that customers can book a bicycle and that booked bicycles can be viewed. Customers can also cancel the booked bicycles. Managers can view all booked bicycles from the customers only.

3.1.1 Activity Diagram - Book a bicycle

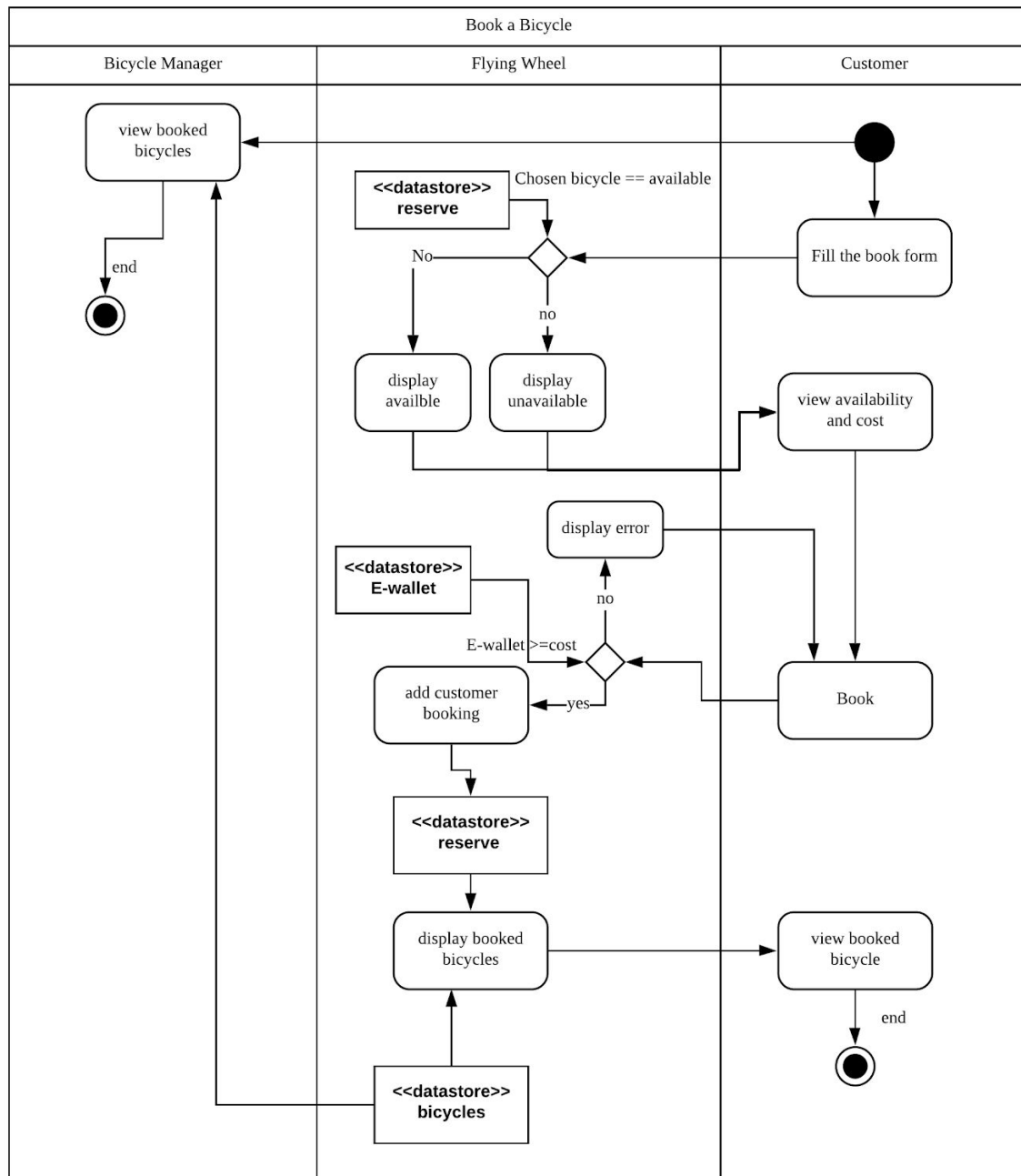


Diagram 3.1.1: Activity diagram for use case Book a Bicycle

3.1.2 Interface Design & Code Snippets - Book a bicycle

➡ Booking

Date-from	Date-to
<input type="text" value="mm/dd/yyyy"/>	<input type="text" value="mm/dd/yyyy"/>
Time-from	Time-to
<input type="text" value="--:-- --"/>	<input type="text" value="--:-- --"/>
Number of Bicycles	type of bicycle
<input type="text" value="e.g 4"/>	<input type="text" value="Electrical"/>
<input type="button" value="Check Availability"/>	

```
private void registerPage(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
    RequestDispatcher dispatcher;  
    request.setAttribute("success", "email/password are/is incorrect ");  
    dispatcher = request.getRequestDispatcher("/customer/customer-register.jsp");  
    dispatcher.forward(request, response);  
}
```

Figure 3.1.2.1: UI for booking Form

➡ Booking

Success! The number of bicycles You entered are available.

Total Cost : 40 RM

Book Bicycles

```
private void checkBooking(HttpServletRequest request, HttpServletResponse response) throws Exception {
    String date_start = request.getParameter("date-start");
    String date_end = request.getParameter("date-end");
    String time_start = request.getParameter("time-start");
    String time_end = request.getParameter("time-end");

    if (Date.valueOf(date_start).getTime() > Date.valueOf(date_end).getTime()) {
        request.setAttribute("fail", "Wrong date format! date from cannot ever be greater than date-to");
        addBookingPage(request, response);
    }
    String number = request.getParameter("number");
    String type = request.getParameter("type");
    List<Integer> value
        = userDao.checkBooking(date_start, date_end, time_start, time_end, number, type);
    // for (int i = 0; i < value.size(); i++) {
    System.out.println("check value " + value.toString());
    int size = value.size() - 1;

    System.out.println("check available " + value.get(size));
    //if bicycles not available :
    if (value.get(size) == 0) {
        request.setAttribute("fail", "The amount of bicycles you asked for are not available at that exact date");
        addBookingPage(request, response);
    } else {
        HttpSession session = request.getSession(true);

        Booking book = new Booking();
        book.setDate_start(Date.valueOf(date_start));
        book.setDate_end(Date.valueOf(date_end));
        book.setTime_start(time_start);
        book.setTime_end(time_end);

        request.setAttribute("available", "available now");

        value.remove(size);

        session.setAttribute("bicycleList", value);

        BicycleDao cycle = new BicycleDao();
        int cost
            = cycle.getCost(value);

        request.setAttribute("cost", cost);
        session.setAttribute("cost", cost);
        book.setCost(cost);
        session.setAttribute("booking", book);
        addBookingPage(request, response);
    }
}
```

Figure 3.1.2.2: UI & logic for checking the Booking

Book

ID	Date from	Date to	Time from	Time to	Cost	Action
8	2019-12-05	2019-12-04	07:30:00	19:30:00	80	Cancel
9	2019-12-12	2019-12-19	19:30:00	19:30:00	40	Cancel
10	2019-12-26	2019-12-27	19:30:00	07:30:00	80	Cancel
11	2019-12-24	2019-12-25	19:30:00	19:30:00	40	Cancel

```
private void viewBookingPage(HttpServletRequest request, HttpServletResponse response) throws Exception {
    BookingDao bookDao = new BookingDao();
    HttpSession session = request.getSession(true);
    int id = (int) session.getAttribute("customerid");
    List<Booking> bookings = bookDao.getUserBooking(id);

    request.setAttribute("BOOKING_LIST", bookings);

    RequestDispatcher dispatcher;
    dispatcher = request.getRequestDispatcher("/customer/viewBooking.jsp");
    dispatcher.forward(request, response);
}
```

Figure 3.1.2.3: UI for viewing Customer Bookings(Customer)

Customer Bookings

ID	Date from	Date to	Time from	Time to	Cost
8	2019-12-05	2019-12-04	07:30:00	19:30:00	80
9	2019-12-12	2019-12-19	19:30:00	19:30:00	40
10	2019-12-26	2019-12-27	19:30:00	07:30:00	80
11	2019-12-24	2019-12-25	19:30:00	19:30:00	40
12	2222-02-01	1111-11-11	18:41:00	08:41:00	40
13	2019-12-25	2019-12-26	07:29:00	07:29:00	80

```
private void viewBooking(HttpServletRequest request, HttpServletResponse response) throws Exception {
    BookingDao bookDao = new BookingDao();
    List<Booking> bookings = bookDao.getAllBooking();

    request.setAttribute("BOOKING_LIST", bookings);

    RequestDispatcher dispatcher;
    dispatcher = request.getRequestDispatcher("/manager/viewBooking.jsp");
    dispatcher.forward(request, response);
}
```

Figure 3.1.2.4: UI for all Customer Bookings(Manager)

3.2 Use Case Order item

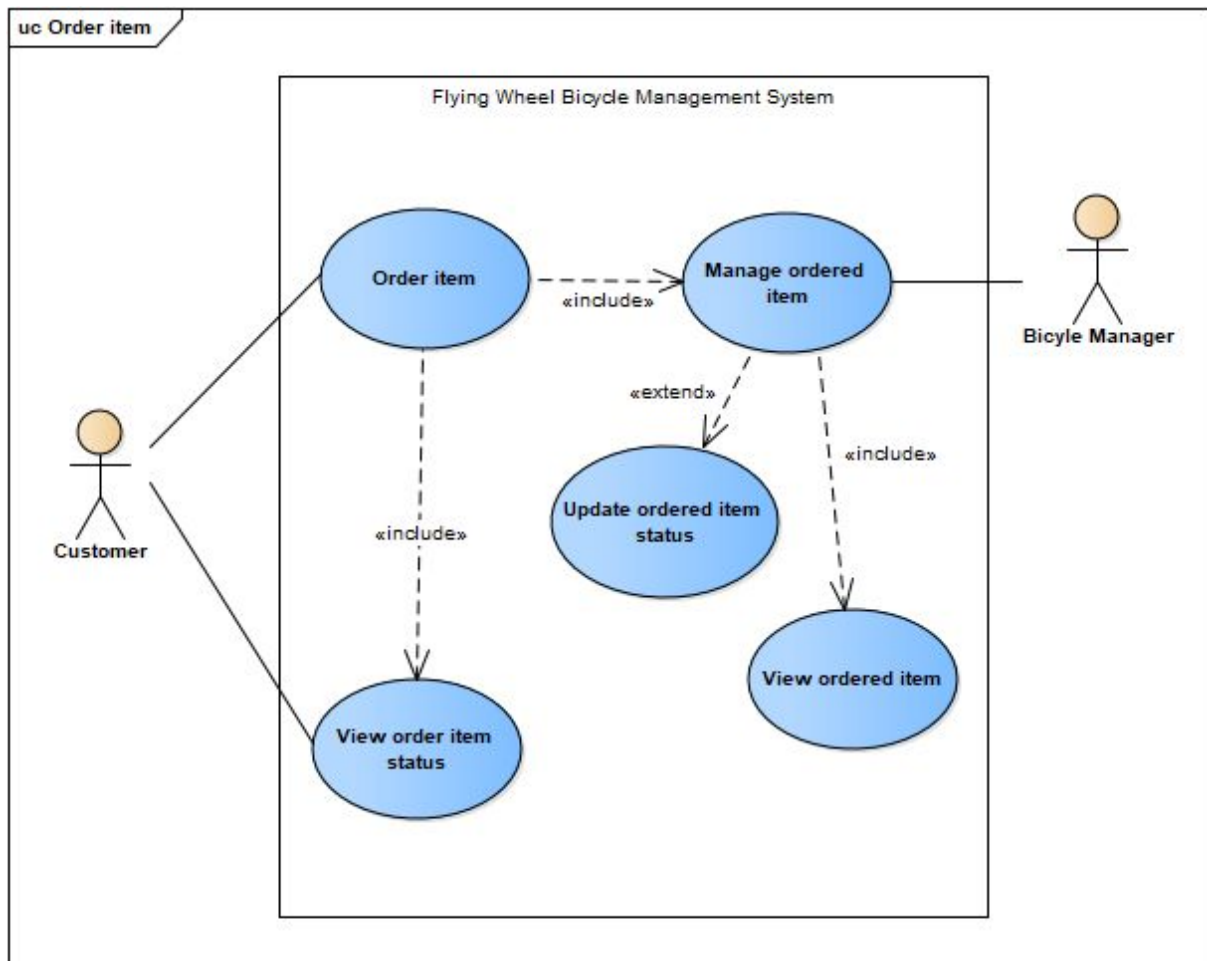


Diagram 3.2.0: Use case diagram for use case Order item

According to *Diagram 3.3.0*, Customer can view order item status for ordered item and order item from Flying Wheel web application. After successful check out, bicycle manager will manage ordered item. In this process, bicycle manager can view ordered item, view its detail as well as update ordered item status.

3.2.1 Activity Diagram - Order item

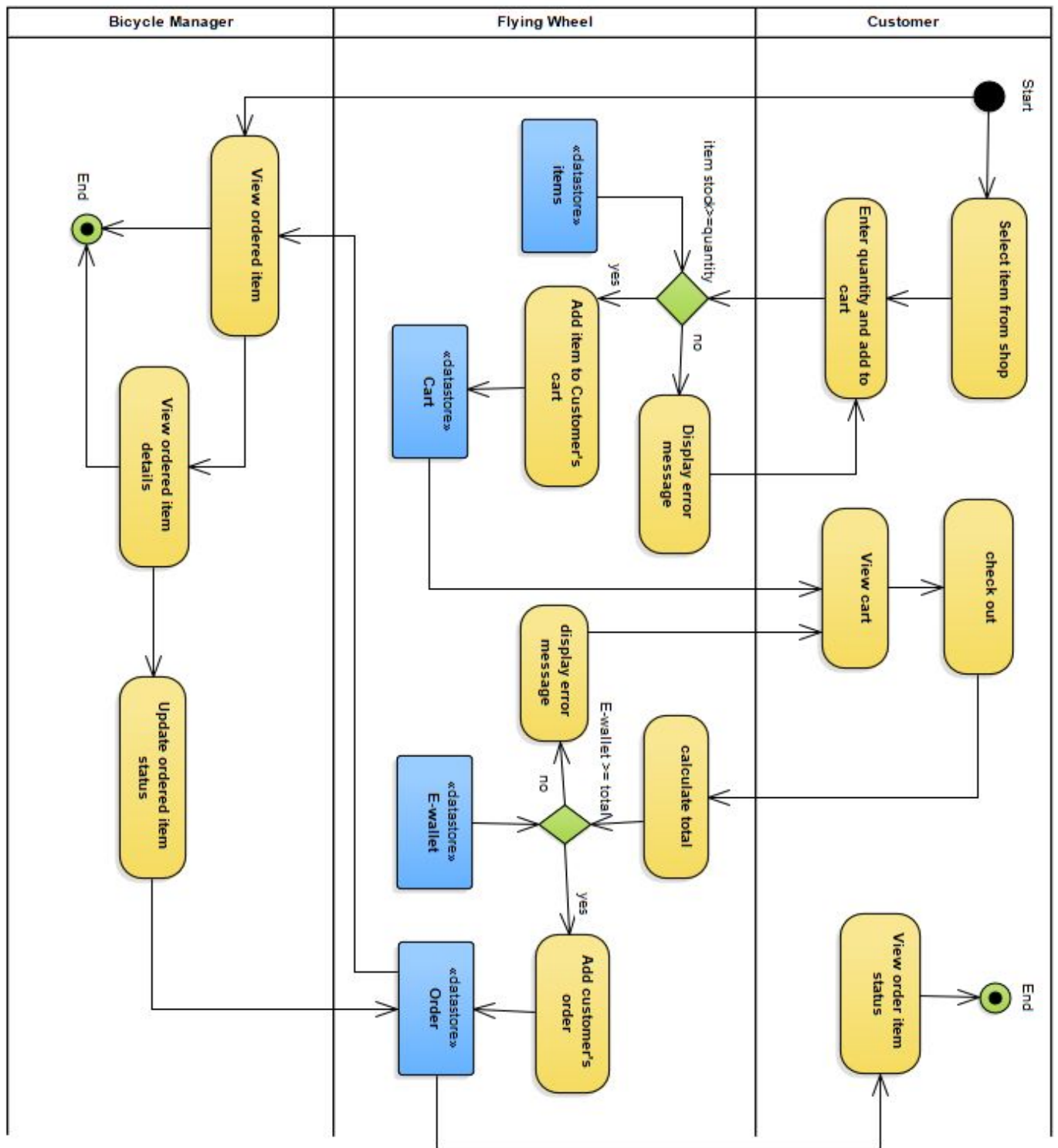



Diagram 3.2.1: Activity diagram for use case Order item

3.2.2 Interface Design & Code Snippets - Order item



Figure 3.2.2.1: UI for all Items in the shop(Customer)

 Add To Cart

Item's name: white helmet

brand: rdx

Cost: RM 20.0

Quantity

Add To Cart

Figure 3.2.2.2: UI for adding the item into the cart(Customer)


Product	Brand	Quantity	Cost	Stock	Action
 white helmet	rdx	<input type="text" value="1"/>	20.0	15	Delete

Check out:

Figure 3.2.2.3: UI for Customer cart(Customer)

Collection Address

UTM Bicycle Shop Student Mall, Jalan Meranti, UTM Skudai

Product	Brand	Cost	Quantity	Item Subtotal
 white helmet	rdx	RM 20.0	1	RM 20.0
Order Total (1 items)				RM 20.0

Payment Method:

E-wallet

Place Order

Figure 3.2.2.4: UI for Customer Checkout(Customer)

Completed Order				
Your item is successfully ordered! ✕				
Product	Quantity	Total paid	Status	Action
 white helmet	1	20.0	Order Placed	View Details

Figure 3.2.2.5: UI for Customer ordered items(Customer)


Prepared Order Ready Order Completed Order				
Product	Quantity	Total Paid	Status	Action
ahmad				
 white helmet	1	RM 20.0	Order Placed	View

Figure 3.2.2.6: UI for all orders(Manager)

Order Placed

Order ID : 24

ahmad

User ID : 37

Email : ahmad@gmail.com

Contact : 01125601863

Order Information

Item Name : white helmet

Item ID : 9

Quantity : X 1

Status :

Order Total : RM 20.0

Figure 3.2.2.7: UI for updating the order status(Manager)

3.3 Use Case View reports

4.0 Reflection

When we first started off, it was actually the leadership in Khaalid who put up a basic MVC skeleton of the project to make everyone's task easier and organized. This has eventually made our group to be high performance as we started to build this application early to be ahead of schedule and planned out the project well. By using Trello, all of us are well-disciplined to get our given task done at the given time and be able to track our progress and know exactly what to do. In addition, we will sit down together to update our progress every Sunday night and discuss about our modules.

This group project was a success due to the teamwork and the good relationship we have as a team. For instance, the leadership within Mohsin and Khaalid have lead the group a lot to solve problems. Although many would think that the task component would be more important than the relationship component, I think both are as equally important and complement each other. What makes our team eventually develops into a high-performance team is good relationship we establish along the way. As we grow closer together, we actually developed a sense of belonging to the team. We want everyone to do well because we are a team. We will solve each other's problem as a team and help each other out.

Our group is a high-performance group as we are eager to produce a quality product. When we ensured that everyone had quality content, we were actually letting go of our ego and listened to each other's opinion. We compromised for each other given the time limit. We ensured that each of us stick to the time limit and this would not be possible if our team did not develop a high relationship throughout the process. It was when we care for each other, we treated each member as our friends, and not just a group mate. And sometimes, some people would need to do more than the others.

All in all, after doing this group project together, it is important to realise that high relationship and leadership are very important components in a team. It may be the factor that spurs the team to achieve a high-task performance as well as a group with good teamwork, discipline, organized and well-planned.