CS3343 Group Project Presentation

Hong Kong Journey Planner

Group: 38

Group member

WANG Fan

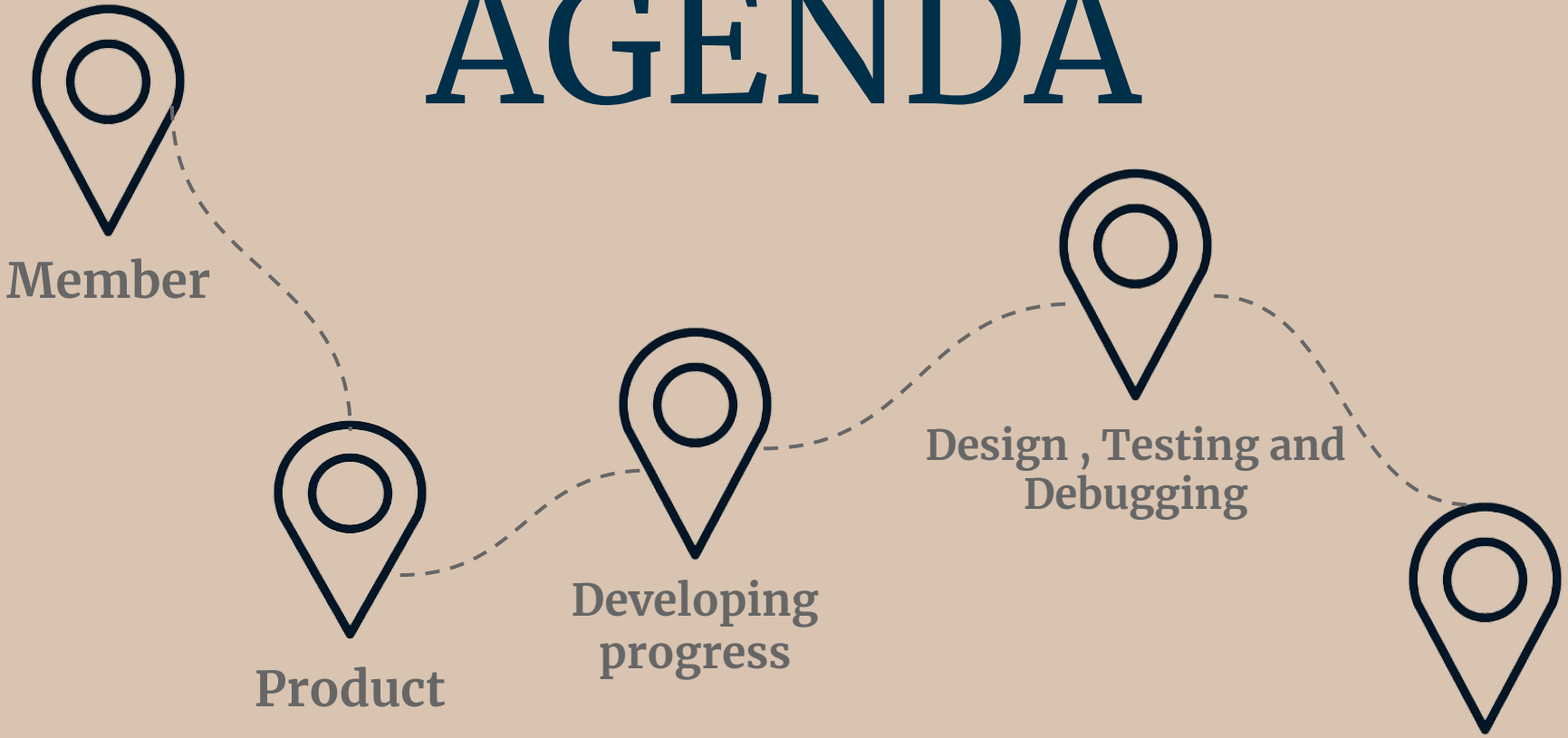
LIU Hengche

GAO Nanjie

FAN Tianrui

CHEUNG Lok Yi

AGENDA


Member
Product
Developing progress
Design , Testing and Debugging
Technology

Members

Introduction of group members

Project roles of members

Project Organization

Role	Member
Project Manager	Fan Tianrui
Coordinating Project Manager	Wang Fan
Developing Analyst	GAO Nanjie
Testing Engineer	CHEUNG Lok Yi
Analyst Programmer	LIU Hengche

TimeSheet

Course	CS3343
Term	2024-25 Semester A
Project No.	38
Project Name/Title	Hong Kong Trip Generator
No. of Team Members	5
Project Manager Nar	FAN Tianrui

Student ID	Name (Last, First)	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	Total
57854004	WANG Fan	1	3	4	1	1	1	1	2	1	3	4	5	3	30
57854329	LIU Hengche	1	3	4	1	1	1	1	2	1	3	4	5	3	30
57854956	GAO Nanjie	1	3	4	1	1	1	1	2	1	3	4	5	3	30
57853947	FAN Tianrui	1	3	4	1	1	1	1	2	1	3	4	5	3	30
57837150	CHEUNG Lok Yi	1	3	4	1	1	1	1	2	1	3	4	5	3	30
	Total Effort	5	15	20	5	5	5	5	10	5	15	20	25	15	150

Student	Relative	Contribution	e.g. Group Mark	Final Adjusted
WANG Fan	1.00	100.00%	100.00	100.00
LIU Hengche	1.00	100.00%	100.00	100.00
GAO Nanjie	1.00	100.00%	100.00	100.00
FAN Tianrui	1.00	100.00%	100.00	100.00
CHEUNG Lok Yi	1.00	100.00%	100.00	100.00
Max	**1.00**		Average	100.00
			Max	100.00
			Min	100.00

Product

Background Information

Product philosophy and objectives

Specific expected outcome

There are many interesting places in Hong Kong that visitors to Hong Kong do not know very well. Maybe they know some of these attractions, but they don't know the traffic situation in Hong Kong, and it isn't easy to connect the attractions to form a complete tour route. They may want to experience the most unique and memorable aspects of Hong Kong's culture in just a few days.

What are the problems they may come across?

Pain points for travelers

What are the most effective ways to visit all the places I want to go?

Which attractions are more popular?

Which attractions are rated highly?

Where are the places valued to visit?

How to choose a travel route within your budget

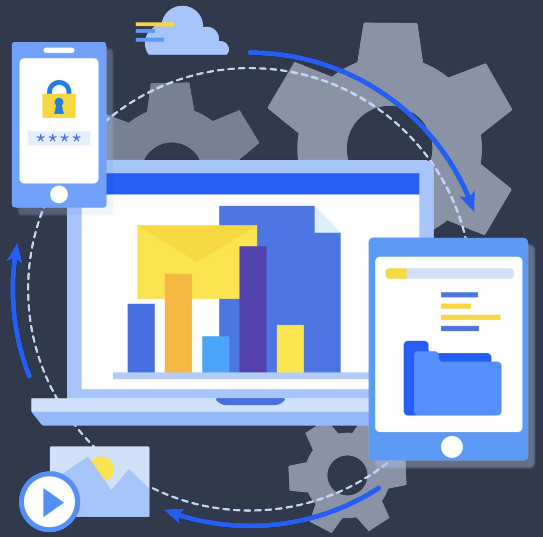
What is the shortest route to go to these places?

Objectives

Key Goals of the Application

1. Provide Customized service
2. Help users solve real problems
3. Promote Exploration of Diverse Districts

Expected outcome



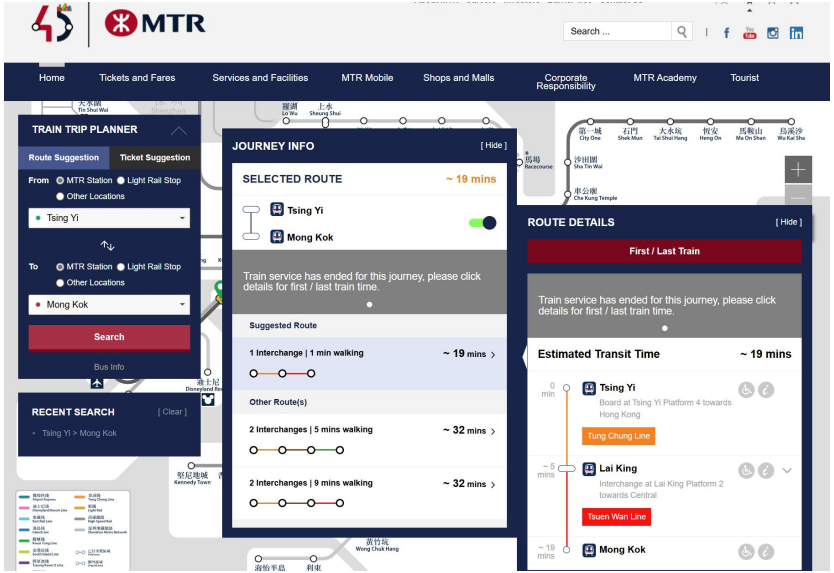
Various personalized requirements

Customized route

Adding or removing specific attractions

Edited route

Data Source


MTR
Search ...
Home
Tickets and Fares
Services and Facilities
MTR Mobile
Shops and Malls
Corporate Responsibility
MTR Academy
Tourist
TRAIN TRIP PLANNER
Route Suggestion
Ticket Suggestion
From
MTR Station
Light Rail Stop
Other Locations
Tsing Yi
To
MTR Station
Light Rail Stop
Other Locations
Mong Kok
Search
Bus Info
RECENT SEARCH
[Clear]
Tsing Yi > Mong Kok
JOURNEY INFO
[Hide]
SELECTED ROUTE
~ 19 mins
Tsing Yi
Mong Kok
Train service has ended for this journey, please click details for first / last train time.
Suggested Route
1 Interchange | 1 min walking
~ 19 mins
Other Route(s)
2 Interchanges | 5 mins walking
~ 32 mins
2 Interchanges | 9 mins walking
~ 32 mins
ROUTE DETAILS
[Hide]
First / Last Train
Train service has ended for this journey, please click details for first / last train time.
Estimated Transit Time
~ 19 mins
Tsing Yi
Board at Tsing Yi Platform 4 towards Hong Kong
Tung Chung Line
Lai King
Interchange at Lai King Platform 2 towards Central
Tsuen Wan Line
Mong Kok

Developing Progress

Develop Model

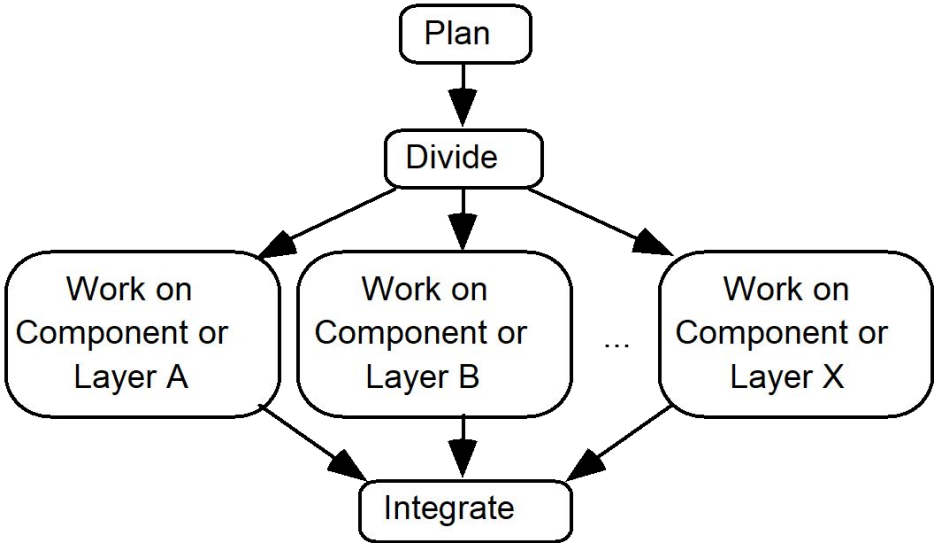
Function & Version Iteration

Time Schedule

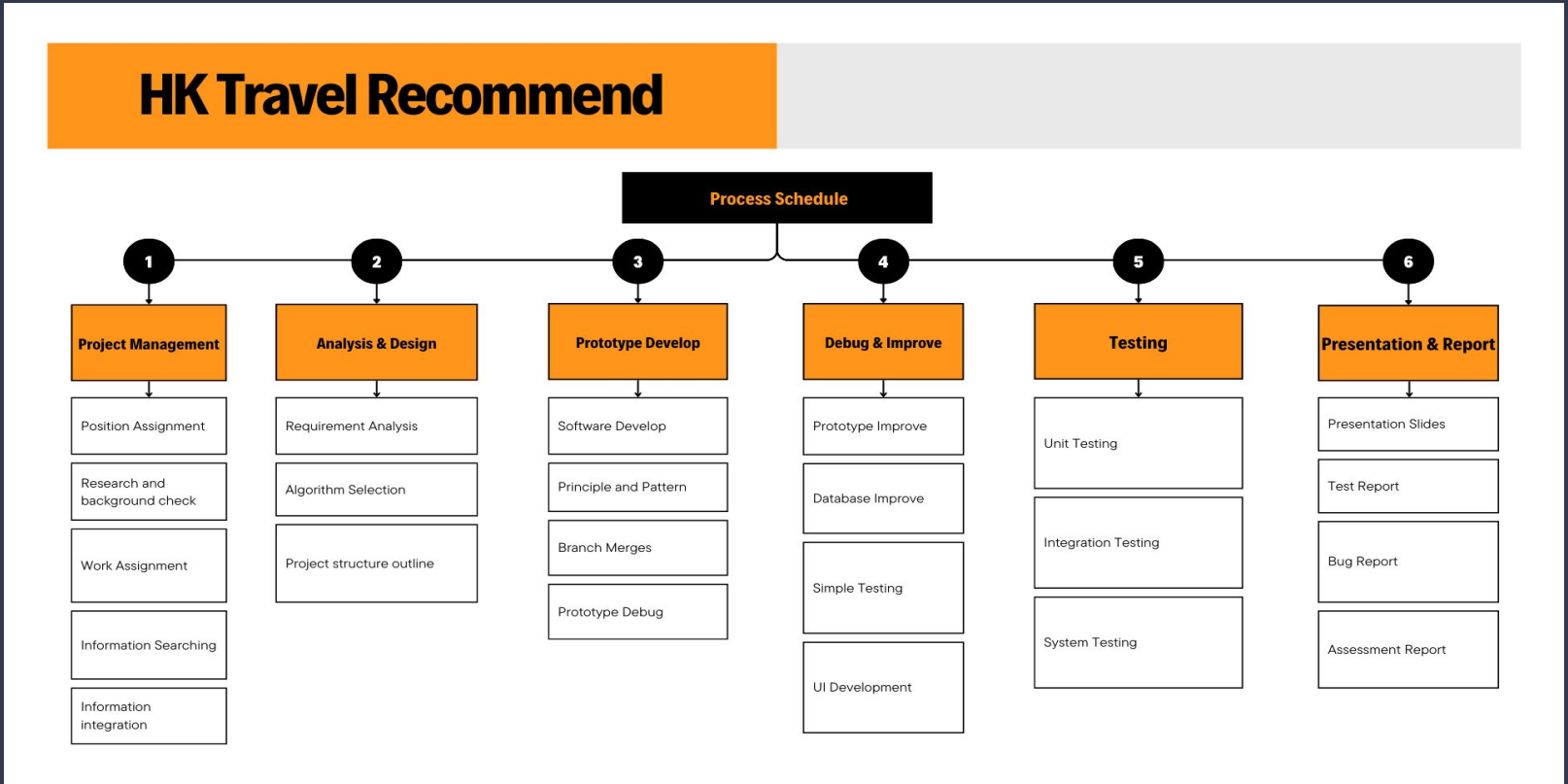
Software Development Methodology

Concurrent Engineering Model

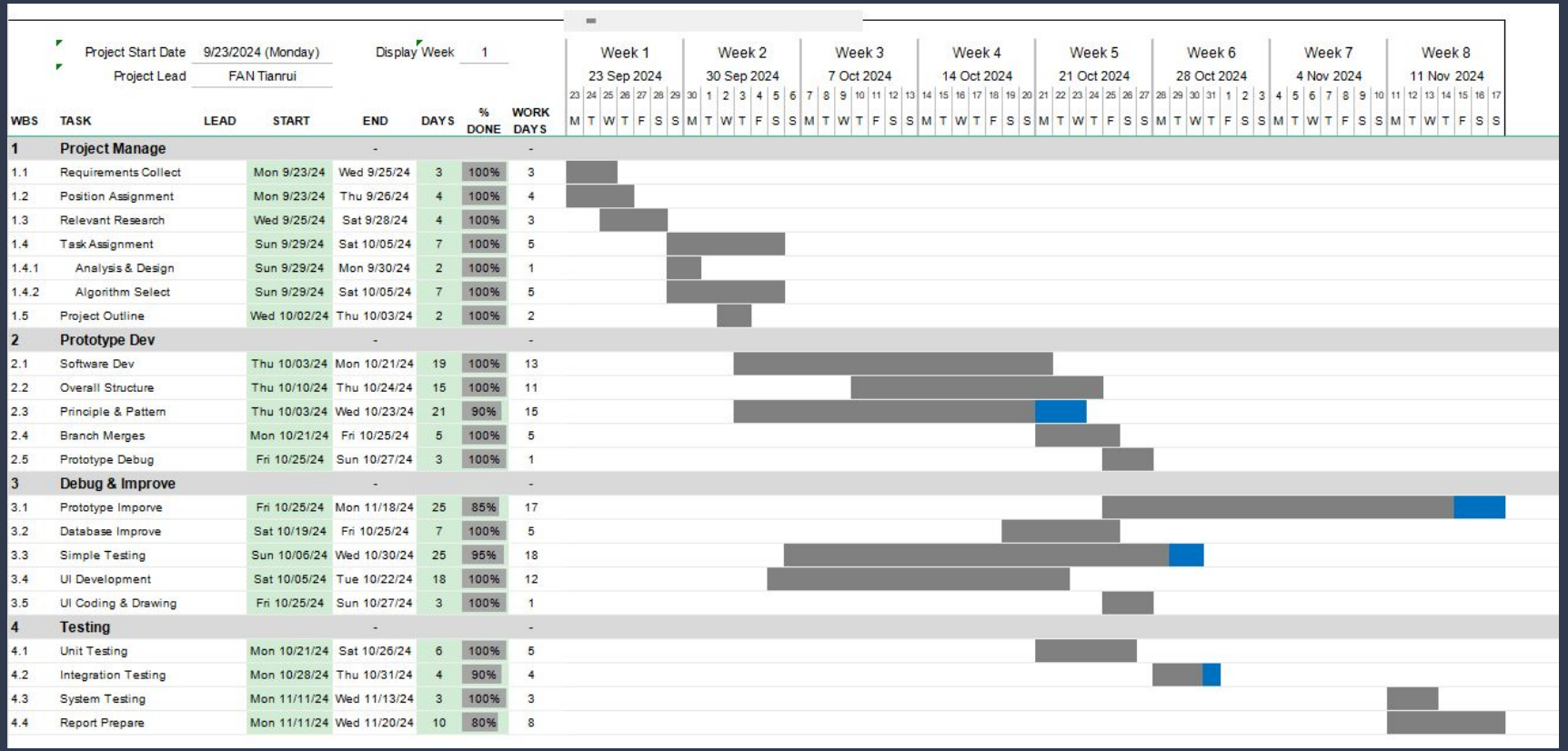
Parallel development



Work Breakdown Structure



Resource Management: GANTT CHART



Project Start Date: 9/23/2024 (Monday) | Display Week: 1 | Project Lead: FAN Tianrui

WBS	TASK	LEAD	START	END	DAYS	% DONE	WORK DAYS
1	**Project Manage**			-			-
1.1	Requirements Collect		Mon 9/23/24	Wed 9/25/24	3	100%	3
1.2	Position Assignment		Mon 9/23/24	Thu 9/26/24	4	100%	4
1.3	Relevant Research		Wed 9/25/24	Sat 9/28/24	4	100%	3
1.4	Task Assignment		Sun 9/29/24	Sat 10/05/24	7	100%	5
1.4.1	Analysis & Design		Sun 9/29/24	Mon 9/30/24	2	100%	1
1.4.2	Algorithm Select		Sun 9/29/24	Sat 10/05/24	7	100%	5
1.5	Project Outline		Wed 10/02/24	Thu 10/03/24	2	100%	2
2	**Prototype Dev**			-			-
2.1	Software Dev		Thu 10/03/24	Mon 10/21/24	19	100%	13
2.2	Overall Structure		Thu 10/10/24	Thu 10/24/24	15	100%	11
2.3	Principle & Pattern		Thu 10/03/24	Wed 10/23/24	21	90%	15
2.4	Branch Merges		Mon 10/21/24	Fri 10/25/24	5	100%	5
2.5	Prototype Debug		Fri 10/25/24	Sun 10/27/24	3	100%	1
3	**Debug & Improve**			-			-
3.1	Prototype Imporve		Fri 10/25/24	Mon 11/18/24	25	85%	17
3.2	Database Improve		Sat 10/19/24	Fri 10/25/24	7	100%	5
3.3	Simple Testing		Sun 10/06/24	Wed 10/30/24	25	95%	18
3.4	UI Development		Sat 10/05/24	Tue 10/22/24	18	100%	12
3.5	UI Coding & Drawing		Fri 10/25/24	Sun 10/27/24	3	100%	1
4	**Testing**			-			-
4.1	Unit Testing		Mon 10/21/24	Sat 10/26/24	6	100%	5
4.2	Integration Testing		Mon 10/28/24	Thu 10/31/24	4	90%	4
4.3	System Testing		Mon 11/11/24	Wed 11/13/24	3	100%	3
4.4	Report Prepare		Mon 11/11/24	Wed 11/20/24	10	80%	8

Week 1 — 23 Sep 2024 | Week 2 — 30 Sep 2024 | Week 3 — 7 Oct 2024 | Week 4 — 14 Oct 2024 | Week 5 — 21 Oct 2024 | Week 6 — 28 Oct 2024 | Week 7 — 4 Nov 2024 | Week 8 — 11 Nov 2024

Schedule

Task	Start Date	End Date
1. Project Management		
1.1 Requirement Collect	23/09/2024	25/09/2024
1.2 Position Assignment	23/09/2024	26/09/2024
1.3 Relavant research	25/09/2024	28/09/2024
1.4 Task Assignment	29/09/2024	05/10/2024
1.5 Project Outline	02/10/2024	03/10/2024
2. Prototype Development		
2.1 Software Development	03/10/2024	21/10/2024
2.2 Overall Structure	10/10/2024	24/10/2024
2.3 Principle and Pattern	03/10/2024	23/10/2024
2.4 Branch Merges	21/10/2024	25/10/2024
2.5 Prototype Debug	25/10/2024	27/10/2024
3. Debug and Improvement		
3.1 Prototype Improvement	25/10/2024	18/11/2024
3.2 Database Improvement	19/10/2024	25/10/2024
3.3 Simple Testing	06/10/2024	30/10/2024
3.4 UI Development	05/10/2024	22/10/2024
3.5 UI Coding and Drawing	25/10/2024	27/10/2024
4. Testing		
4.1 Unit Testing	21/10/2024	26/10/2024
4.2 Integration Testing	28/10/2024	31/10/2024
4.3 System Testing	11/11/2024	20/11/2024
5. Rollout		
5.1 Presentation	11/11/2024	16/11/2024
5.2 Deliverables	11/11/2024	04/12/2024

Design , Testing and Debugging

Class, Use case, Sequence diagram

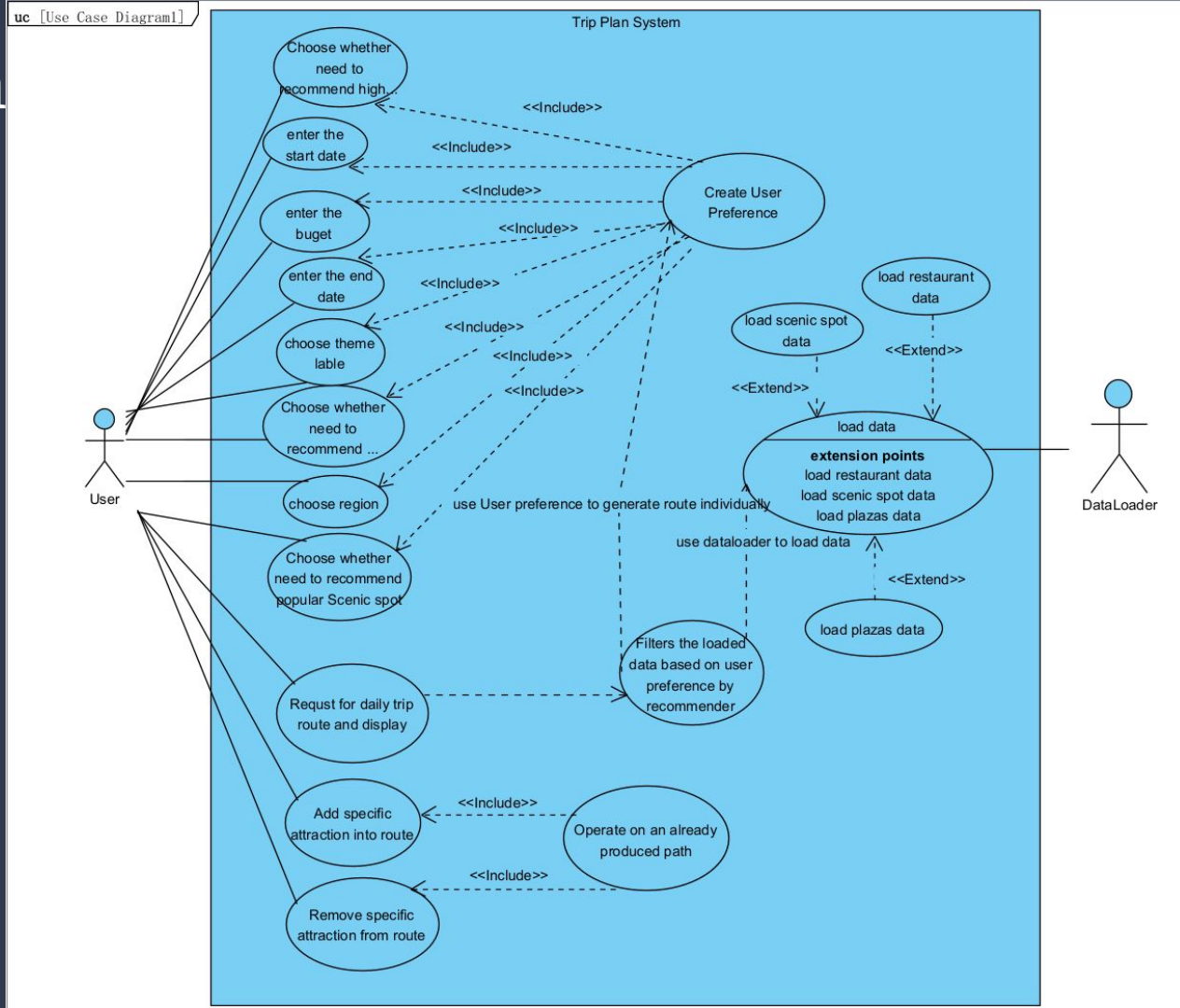
Design pattern & principle

Code Refactoring

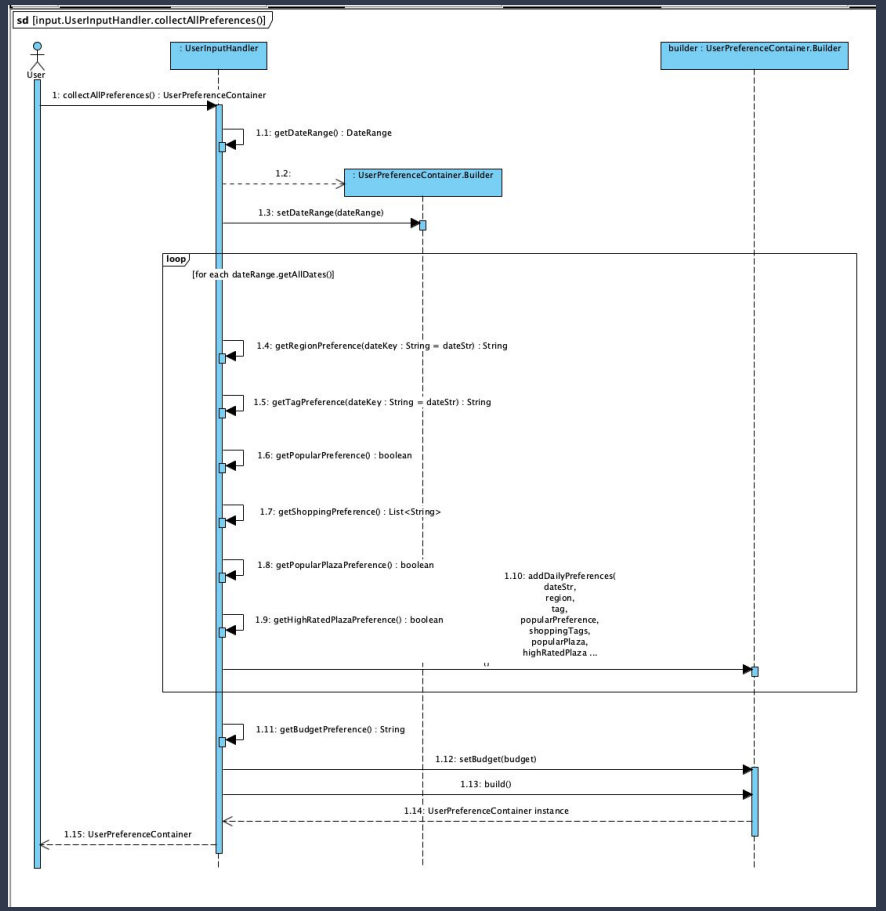
Test integration & Coverage

Bug report

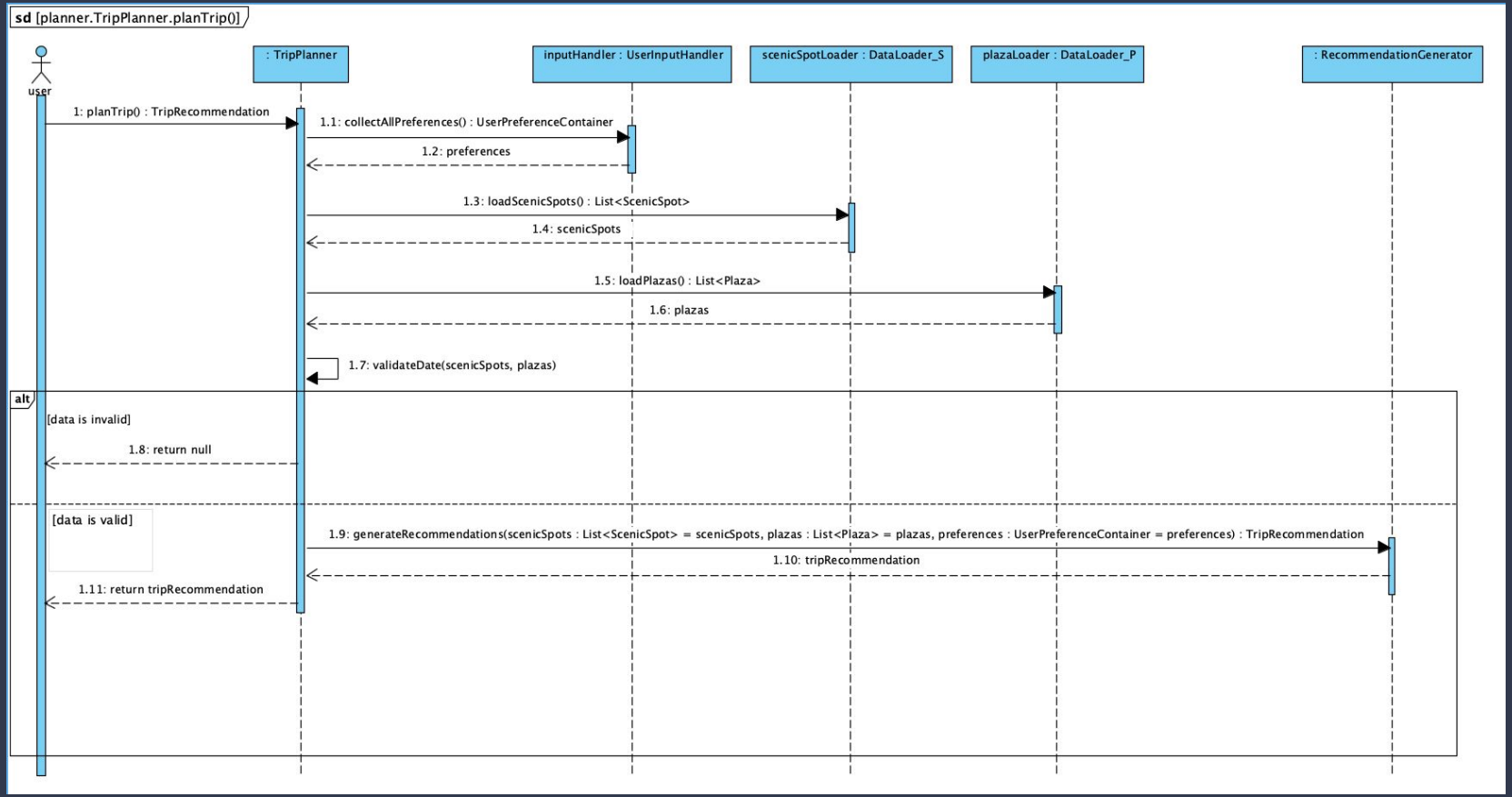
Use Case Diagram



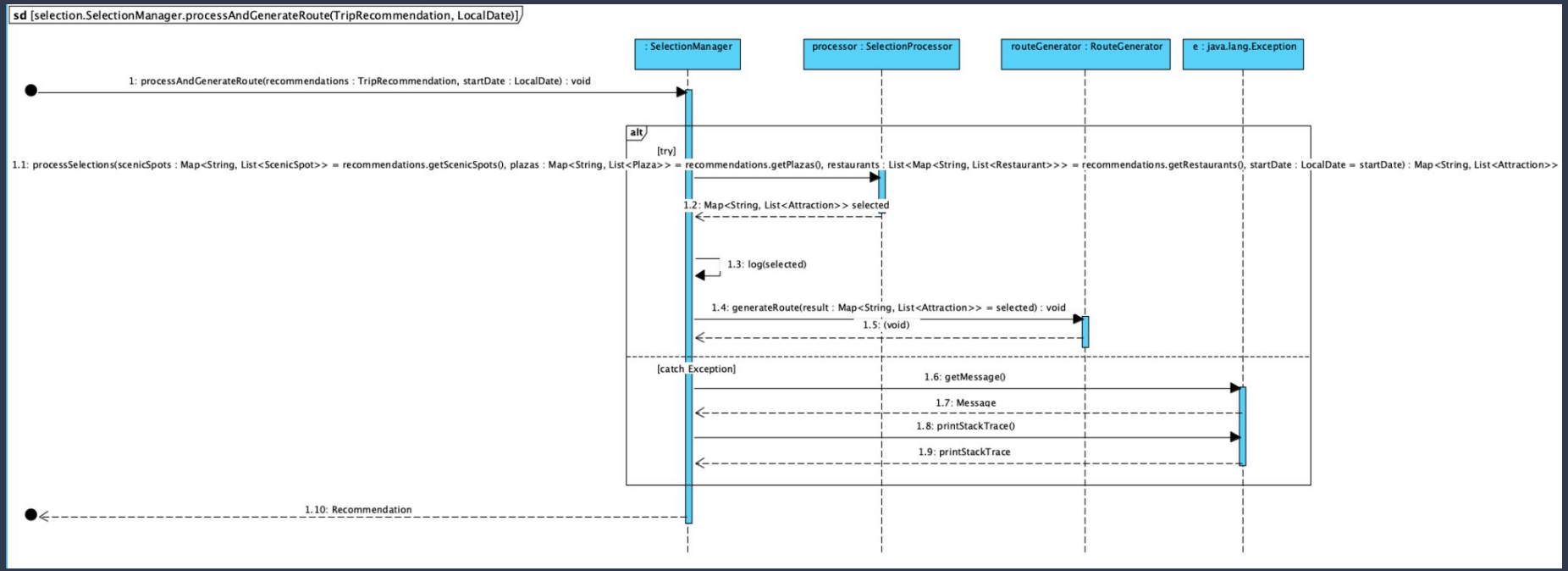
Sequence Diagram



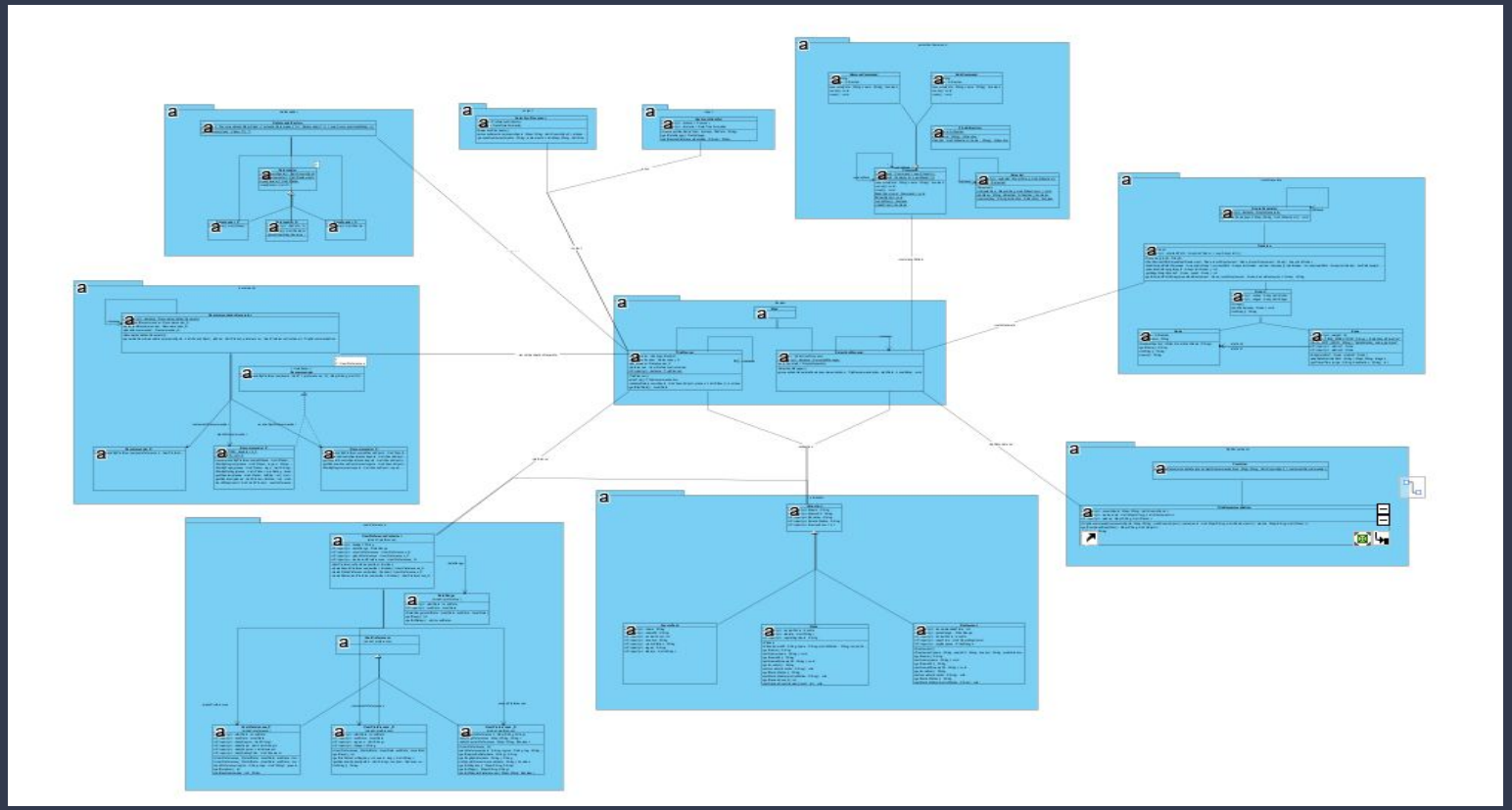
Sequence Diagram



Sequence Diagram



Class Diagram



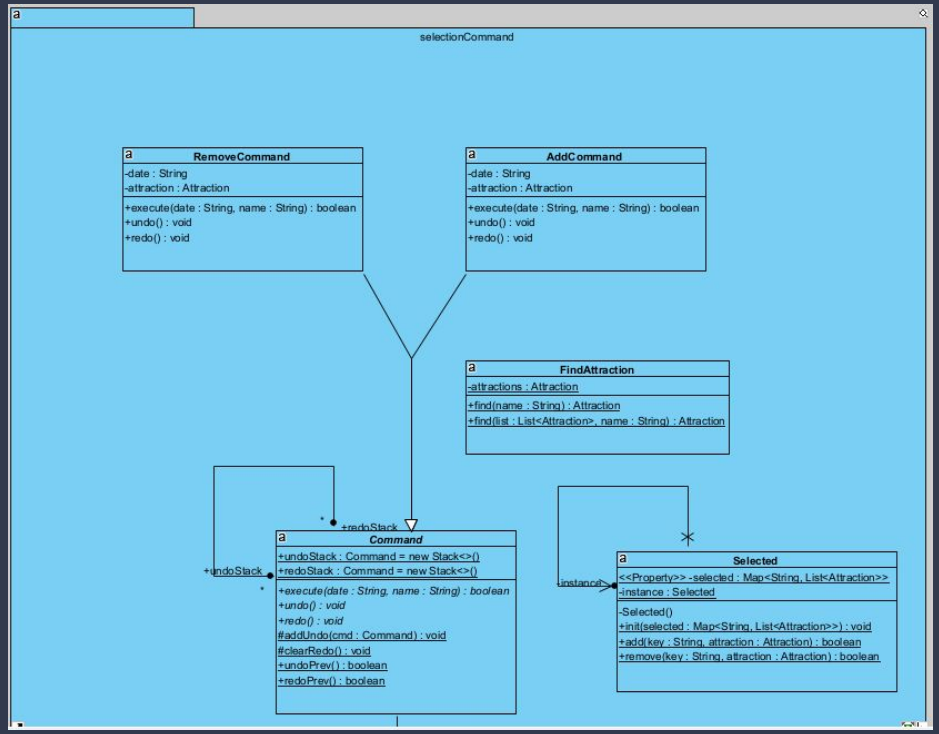
Design Principles

S	Single Responsibility Principle
O	Open-closed Principle
L	Liscov Substitution Principle
I	Interface Segragation Principle
D	Dependency Inversion Principle

Design Pattern

- Command Pattern
- Factory Pattern
- Facade Pattern
- Singleton Pattern
- Builder Pattern

selectionCommand



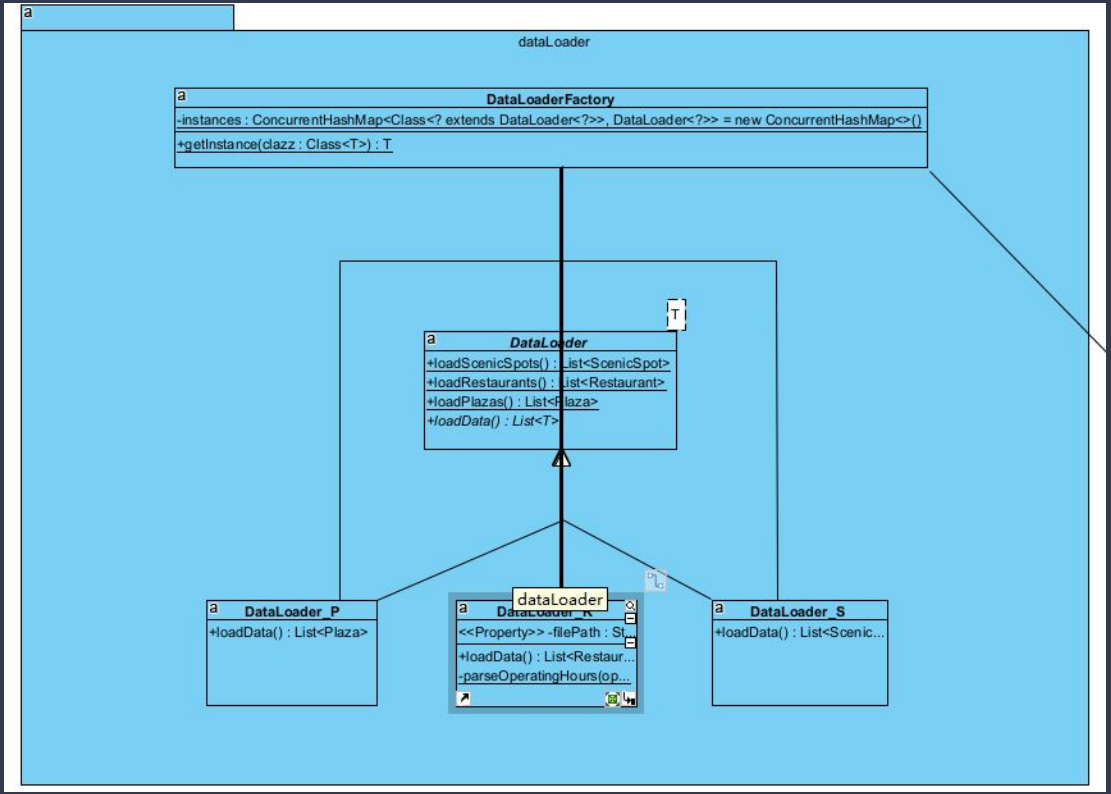
Pattern:

- Command Pattern
- Singleton Pattern

Principle:

- OCP

dataLoader



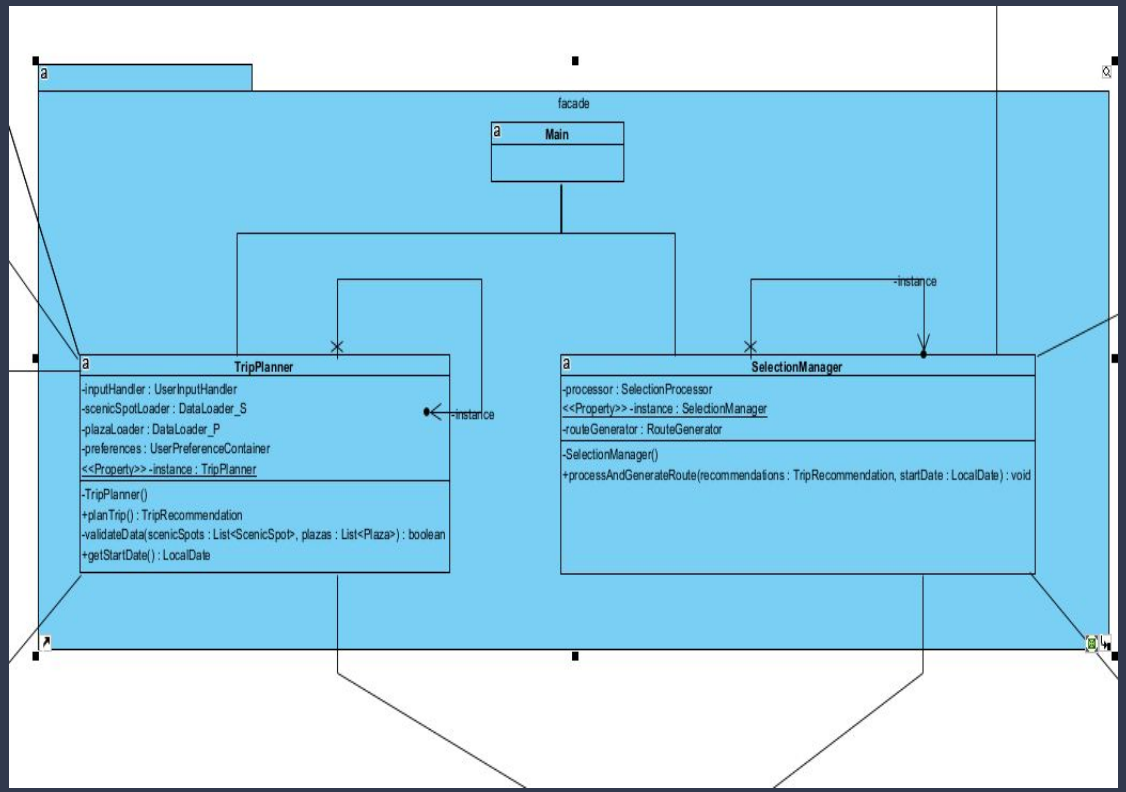
Pattern:

- Factory Pattern
- Singleton Pattern

Principle:

- LSP

facade



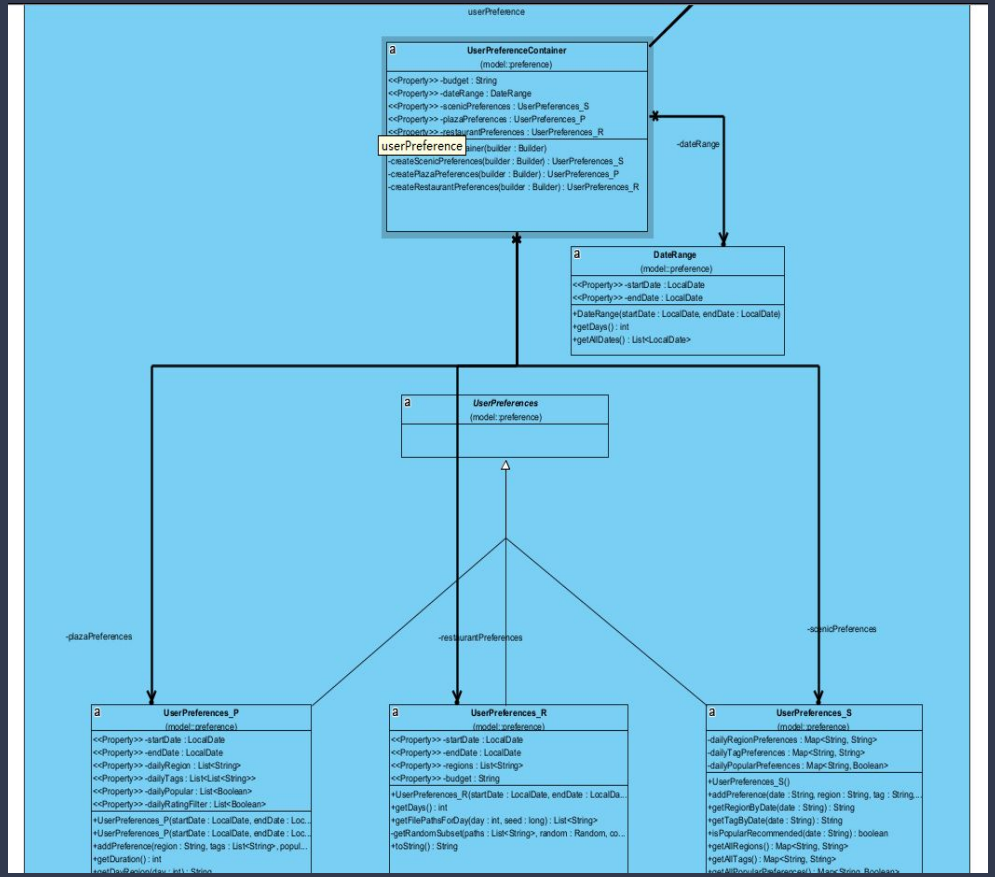
Pattern:

- Facade Pattern
- Singleton Pattern

Principle:

- SRP

userPreference



Pattern:
- Builder Pattern

Principle:
- OCP
- SRP

userPreference

```java
public class UserPreferenceContainer {
    public static class Builder {
        private DateRange dateRange;
        private Map<String, String> regionPreferences = new HashMap<>();
        private Map<String, String> tagPreferences = new HashMap<>();
        private Map<String, Boolean> popularPreferences = new HashMap<>();
        private Map<String, List<String>> shoppingTagsPreferences = new HashMap<>();
        private Map<String, Boolean> plazaPopularPreferences = new HashMap<>();
        private Map<String, Boolean> plazaRatingPreferences = new HashMap<>();
        private String budget;

        public Builder setDateRange(DateRange dateRange) {…

        public Builder addDailyPreferences(…

        public Builder setBudget(String budget) {
            this.budget = budget;
            return this;
        }

        public UserPreferenceContainer build() {
            return new UserPreferenceContainer(this);
        }
    }
```

Pattern:

- Builder Pattern

Principle:

- OCP
- SRP

Code Refactoring

Rename and Merge Duplicate Code

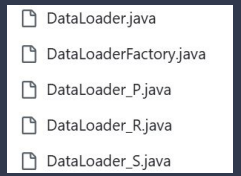
Extract Method

Replace Condition with Polymorphism

Code Refactoring

Rename and Merge Duplicate Code

Rename variables, methods, classes, etc. to make their names more consistent with their actual meaning and to enhance the self-interpretability of the code.
Extract and consolidate duplicate code segments to reduce code redundancy.




```java
import java.util.List;

public class Attraction {
    private String name;
    private String namezh;
    private int reviewCount;
    private String location;
    private String metroStation;
    private String region;
    private List<String> feature
```

```java
protected String name;
protected String nameZh;
protected String location;
protected String metroStation;
protected int reviewCount;
private double reviewScore;
private List<String> feature;
private OpeningHours openingHours;
```

```java
protected String name;
protected String nameZh;
protected String location;
protected String metroStation;
protected int reviewCount;
private int recommendedTime;
private PriceRange priceRange;
private double reviewScore;
private List<OperatingHours> openTime;
private PriceRange avgExpense;
```

Three separate classes with many duplicate attributes and code

```java
public abstract class Attraction {
    protected String name;
    protected String nameZh;
    protected String location;
    protected String metroStation;
    protected int reviewCount;
```

```java
public class Restaurant extends Attraction{

    private int recommendedTime;
    private PriceRange priceRange;
    private double reviewScore;
    private List<OperatingHours> openTime;
    private PriceRange avgExpense;
```

```java
public class Plaza extends Attraction {
    private double reviewScore;
    private List<String> feature;
//    private OperatingHours operatingHours;
    private String operatingHours;
```

```java
public class ScenicSpot extends Attraction {

    private String region;
    private List<String> feature;
```

One base class and three subclasses
Better structure to reduce duplication

Code Refactoring

Extract Method

Certain functions or logic in the code are extracted into separate methods to improve readability and reusability.

Dataloader.java

```java
public class DataLoader {
    public List<ScenicSpot> loadScenicSpots(String filePath) {
        return loadData(filePath, type:"ScenicSpot");
    }
    public List<Restaurant> loadRestaurants(String filePath) {
        return loadData(filePath, type:"Restaurant");
    }
    public List<Plaza> loadPlazas(String filePath) {
        return loadData(filePath, type:"Plaza");
    }
    private <T> List<T> loadData(String filePath, String type) {
        List<T> data = new ArrayList<>();
        ObjectMapper objectMapper = new ObjectMapper();

        try {
            JsonNode root = objectMapper.readTree(new File(filePath))
            for (JsonNode node : root) {
                try {
                    if ("ScenicSpot".equals(type)) {
                        data.add((T) parseScenicSpot(node));
                    } else if ("Restaurant".equals(type)) {
                        data.add((T) parseRestaurant(node));
                    } else if ("Plaza".equals(type)) {
                        data.add((T) parsePlaza(node));
                    }
                } catch (Exception e) {
                    System.err.println("Error processing data: " + e.
                    continue;
                }
            }
        } catch (IOException e) {
            System.err.println("Error reading file: " + e.getMessage(
        }
        return data;
    }
```

```java
private ScenicSpot parseScenicSpot(JsonNode node) {
    String name = node.get("name").asText();
    String location = node.get("location").asText();
    String description = node.get("description").asText();
    return new ScenicSpot(name, location, description);
}
private Restaurant parseRestaurant(JsonNode node) {
    String name = node.get("name").asText();
    String nameZh = node.get("nameZh").asText();
    String location = node.get("location").asText();
    String metroStation = node.get("metroStation").asText();
    int recommendedTime = node.has("recommendedTime") ? node.get("recommendedTime").asInt() : 0;
    String avgExpense = node.get("avgExpense").asText();
    int reviewCount = node.has("reviewCount") ? node.get("reviewCount").asInt() : 0;
    double reviewScore = node.has("reviewScore") ? node.get("reviewScore").asDouble() : 0.0;
    List<OperatingHours> openTime = parseOperatingHours(node.get("openTime").asText());
    return new Restaurant(name, nameZh, location, metroStation, recommendedTime, avgExpense, reviewCount, reviewScore, openTime);
}
private Plaza parsePlaza(JsonNode node) {
    String name = node.get("name").asText();
    String location = node.get("location").asText();
    return new Plaza(name, location);
}

// 解析开放时间字段
private static List<OperatingHours> parseOperatingHours(String openTime) throws DateTimeParseException {
    List<OperatingHours> operatingHoursList = new ArrayList<>();
    String[] timeRanges = openTime.split(regex:",");
    for (String timeRange : timeRanges) {
        String[] times = timeRange.trim().split(regex:"-");
        if (times.length == 2) {
            LocalTime start = LocalTime.parse(times[0].trim());
            LocalTime end = LocalTime.parse(times[1].trim());
            operatingHoursList.add(new OperatingHours(start, end));
        }
    }
    return operatingHoursList;
}
```

There is only one dataloader for the three types, and the three methods for reading data are written in one dataloader class.

Dataloader_P.java

```java
public class DataLoader_S extends DataLoader<ScenicSpot> {
    private String filePath = "data\\scenicspots.json";

    public List<ScenicSpot> loadData() {
        ObjectMapper objectMapper = new ObjectMapper();
        try {
            // 读取JSON文件并转换成List
            return objectMapper.readValue(new File(filePath),
                    objectMapper.getTypeFactory().constructCollectionType(List.class, ScenicSpot.class));
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }

    public void setFilePath(String filePath) {
        this.filePath = filePath;
    }
}
```

Dataloader_R.java

```java
public List<Restaurant> loadData() {

    List<Restaurant> attractions = new ArrayList<>();
    ObjectMapper objectMapper = new ObjectMapper();

    try {
        JsonNode root = objectMapper.readTree(new File(filePath));
        for (JsonNode node : root) {
            try {
                String name = node.get("name").asText(); // 获取景点名称
                String nameZh = node.get("nameZh").asText(); // 获取景点的中文名称
                String location = node.get("location").asText(); // 获取景点的位置
                String metroStation = node.get("metroStation").asText(); // 获取地铁站信息
                int recommendedTime = node.has("recommendedTime") ? node.get("recommendedTime").asInt() : 0; // 获取推荐游玩时间
                String avgExpense = node.get("avgExpense").asText(); // 获取获取景点的平均开销
                int reviewCount = node.has("reviewCount") ? node.get("reviewCount").asInt() : 0; // 获取评论数量
                double reviewScore = node.has("reviewScore") ? node.get("reviewScore").asDouble() : 0.0; // 获取评分
                List<OperatingHours> openTime = parseOperatingHours(node.get("openTime").asText()); // 获取开放时间
                attractions.add(new Restaurant(name, nameZh, location, metroStation, recommendedTime, avgExpense,
                                               reviewCount, reviewScore, openTime));
            } catch (Exception e) {
                // 处理处理单个景点数据出错，跳过该景点继续处理下一个
                System.err.println("Error processing attraction: " + e.getMessage());
                continue;
            }
        }
    } catch (IOException e) {
        System.err.println("Error reading file: " + e.getMessage());
    }

    return attractions;
}
```

Dataloader_S.java

```java
public class DataLoader_S extends DataLoader<ScenicSpot> {
    private String filePath = "data\\scenicspots.json";

    public List<ScenicSpot> loadData() {
        ObjectMapper objectMapper = new ObjectMapper();
        try {
            // 读取JSON文件并转换成List
            return objectMapper.readValue(new File(filePath),
                    objectMapper.getTypeFactory().constructCollectionType(List.class, ScenicSpot.class));
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
```

Encapsulate the three methods for reading different files into three subclasses.

Code Refactoring

Replace Condition with Polymorphism

Replacing complex conditional expressions with polymorphism makes the code more flexible and easily extensible.

```java
public static List<Plaza> loadPlaza(String filePath) {
    ObjectMapper objectMapper = new ObjectMapper();
    try {
        return objectMapper.readValue(new File(filePath),
                objectMapper.getTypeFactory().constructCollectionType(List.class, ScenicSpot.class));
    } catch (IOException e) {
        e.printStackTrace();
    }
    return null;
}
```

```java
public List<Restaurant> loadRestaurant(String filePath) {
    List<Restaurant> attractions = new ArrayList<>();
    try {
        int reviewCount = node.has("reviewCount") ? node.get("reviewCount").asInt() : 0; // 获取评论数量
        double reviewScore = node.has("reviewScore") ? node.get("reviewScore").asDouble() : 0.0; // 获取评分
        List<OperatingHours> openTime = parseOperatingHours(node.get("openTime").asText()); // 获取开放时间
        attractions.add(new Restaurant(name, name2h, location, metroStation, recommendedTime, avgExpense, reviewCount, reviewScore, openTime));
        attractions.add(new Restaurant(name, name2h, location, metroStation, recommendedTime, avgExpense,
                reviewCount, reviewScore, openTime));
    } catch (Exception e) {
        // 如果处理单个景点数据出错，就记录错误并继续处理下一个
        System.err.println("Error processing attraction: " + e.getMessage());
    }
    return attractions;
}
```

```java
public List<ScenicSpot> loadData(String filePath) {
    public List<ScenicSpot> loadData() {
        ObjectMapper objectMapper = new ObjectMapper();
        String filePath = "E:\\github\\CS3343_project\\data\\scenicspots.json";
        try {
            // 读取JSON文件并转换成List
            return objectMapper.readValue(new File(filePath),
    }
}
```

There are three methods with different names used to load data in the three classes

Interface

```java
public abstract class DataLoader<T> {

    // 公共方法：加载 ScenicSpot
    public static List<ScenicSpot> loadScenicSpots() {
        DataLoader<ScenicSpot> loader = DataLoaderFactory.getInstance(DataLoader_S.class);
        return loader.loadData();
    }

    // 公共方法：加载 Restaurant
    public static List<Restaurant> loadRestaurants() {
        DataLoader<Restaurant> loader = DataLoaderFactory.getInstance(DataLoader_R.class);
        return loader.loadData();
    }

    // 公共方法：加载 Plaza
    public static List<Plaza> loadPlazas() {
        DataLoader<Plaza> loader = DataLoaderFactory.getInstance(DataLoader_P.class);
        return loader.loadData();
    }

    // 抽象方法：子类实现具体加载逻辑
    public abstract List<T> loadData();
}
```

Implementation

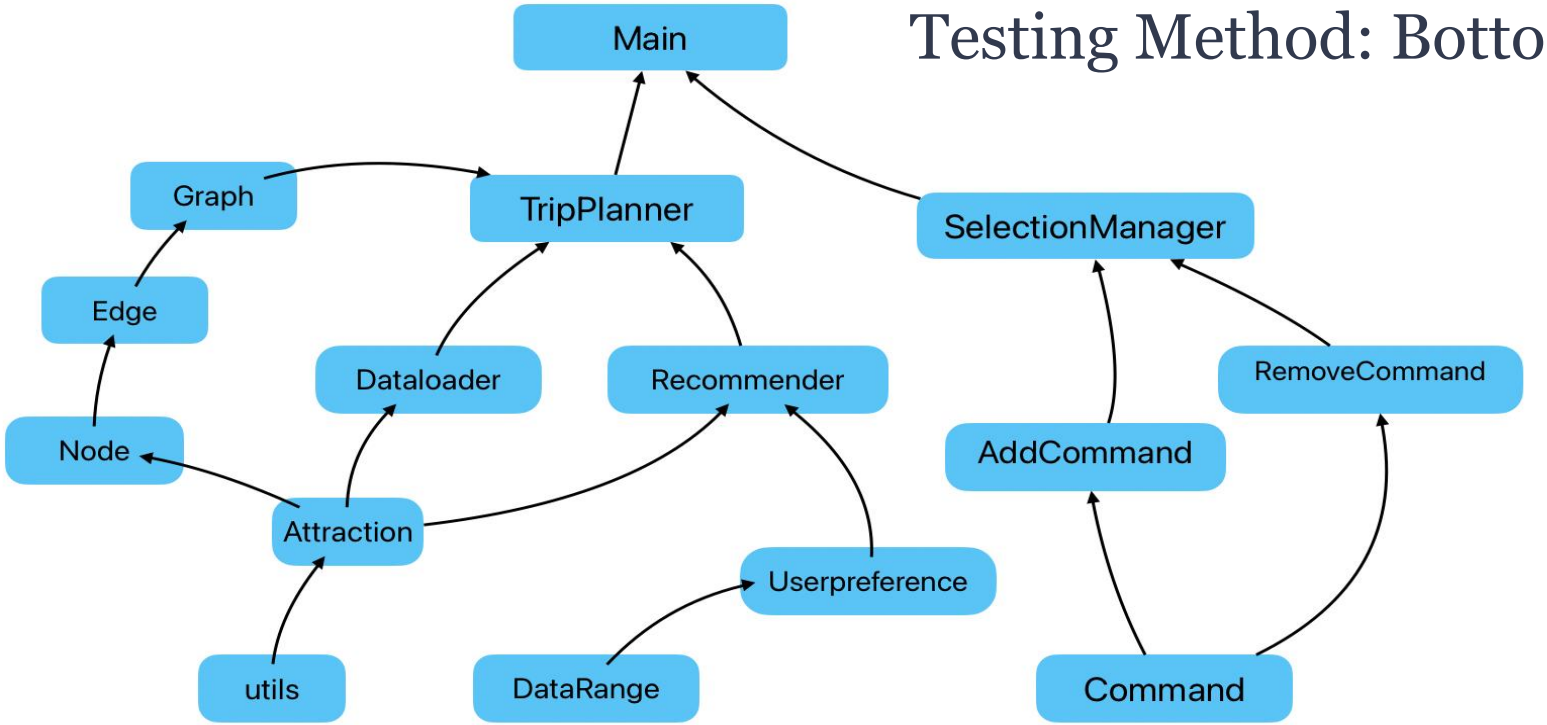
```java
public class DataLoader_P extends DataLoader<Plaza> {
    private String filePath = "data\\shopping.json";
    public List<Plaza> loadData() {
        ObjectMapper objectMapper = new ObjectMapper();
        try {
            return objectMapper.readValue(new File(filePath),
                    objectMapper.getTypeFactory().constructCollectionType(List.class, Plaza.class));
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
```

```java
public List<Restaurant> loadData() {

    List<Restaurant> attractions = new ArrayList<>();
    ObjectMapper objectMapper = new ObjectMapper();

    try {
        JsonNode root = objectMapper.readTree(new File(filePath));
        for (JsonNode node : root) {
            try {
                String name = node.get("name").asText(); // 获取景点名称
                String name2h = node.get("name2h").asText(); // 获取景点的中文名称
                String location = node.get("location").asText(); // 获取景点的位置
                String metroStation = node.get("metroStation").asText(); // 获取地铁站信息
                int recommendedTime = node.has("recommendedTime") ? node.get("recommendedTime").asInt() : 0; // 获取推荐游玩时间
                String avgExpense = node.get("avgExpense").asText(); // 获取获取游玩价格字符串
                int reviewCount = node.has("reviewCount") ? node.get("reviewCount").asInt() : 0; // 获取评论数量
                double reviewScore = node.has("reviewScore") ? node.get("reviewScore").asDouble() : 0.0; // 获取评分
                List<OperatingHours> openTime = parseOperatingHours(node.get("openTime").asText()); // 获取开放时间
                attractions.add(new Restaurant(name, name2h, location, metroStation, recommendedTime, avgExpense,
                        reviewCount, reviewScore, openTime));
            } catch (Exception e) {
                // 如果处理单个景点数据出错，就记录错误并继续处理下一个
                System.err.println("Error processing attraction: " + e.getMessage());
                continue;
            }
        }
    } catch (IOException e) {
        System.err.println("Error reading file: " + e.getMessage());
    }

    return attractions;
}
```

```java
public class DataLoader_S extends DataLoader<ScenicSpot> {
    private String filePath = "data\\scenicspots.json";

    public List<ScenicSpot> loadData() {
        ObjectMapper objectMapper = new ObjectMapper();
        try {
            // 读取JSON文件并转换成List
            return objectMapper.readValue(new File(filePath),
                    objectMapper.getTypeFactory().constructCollectionType(List.class, ScenicSpot.class));
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
```

Create an interface with the same name and different parameters, and implement the function in other classes in different ways.

Test Case


Testing Method: Bottom-up
Main
Graph
TripPlanner
SelectionManager
Edge
Dataloader
Recommender
RemoveCommand
Node
AddCommand
Attraction
Userpreference
utils
DataRange
Command

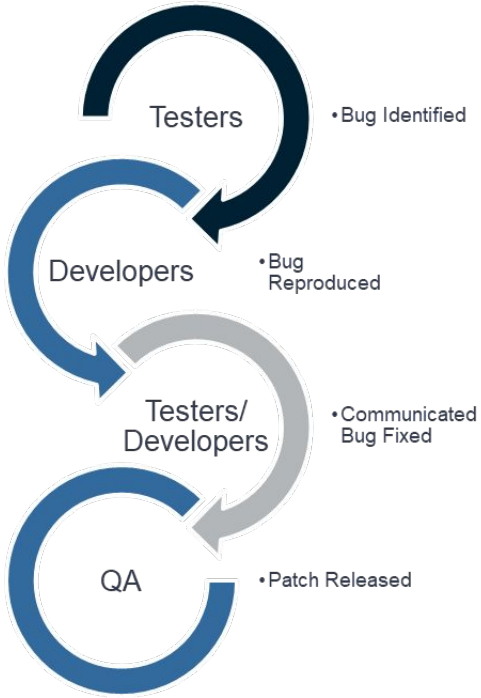
Test Coverage

Element	Coverage	d Instructions	d Instructions	al Instructions
CS3343_project	86.5 %	11,775	1,840	13,615
src	86.5 %	11,775	1,840	13,615
model.preference	100.0 %	957	0	957
model	99.6 %	512	2	514
recommendation	98.4 %	679	11	690
RecommendationGe	100.0 %	62	0	62
Recommender_P.java	100.0 %	226	0	226
Recommender_S.java	100.0 %	159	0	159
Combiner.java	96.7 %	89	3	92
Recommender_R.java	94.7 %	143	8	151
route	98.1 %	1,302	25	1,327
Graph.java	100.0 %	72	0	72
Node.java	100.0 %	23	0	23
RouteGenerator.java	100.0 %	161	0	161
Traverse_new_testing	99.3 %	273	2	275
Traverse.java	99.1 %	229	2	231
Dijkstra.java	98.5 %	261	4	265
Edge.java	94.6 %	243	14	257
RoutePrinter.java	93.0 %	40	3	43

utils	97.8 %	271	6	277
OperatingHours.java	100.0 %	72	0	72
PriceRange.java	100.0 %	129	0	129
TestParamsUtils.java	93.2 %	41	3	44
LocationUtils.java	90.6 %	29	3	32
data	96.1 %	292	12	304
DataLoader_P.java	100.0 %	33	0	33
DataLoader_S.java	100.0 %	33	0	33
DataLoader_R.java	98.4 %	181	3	184
DataLoaderFactory.ja	90.9 %	30	3	33
DataLoader.java	71.4 %	15	6	21
test	94.9 %	6,892	374	7,266
planner	94.5 %	86	5	91
TripPlanner.java	94.5 %	86	5	91
input	51.9 %	433	401	834
selection	42.3 %	339	462	801
AddCommand.java	100.0 %	36	0	36
Command.java	100.0 %	59	0	59
RemoveCommand.ja	100.0 %	40	0	40
Selected.java	100.0 %	95	0	95
FindAttraction.java	93.9 %	46	3	49
SelectionManager.ja	12.1 %	63	459	522
output	2.3 %	12	505	517
(default package)	0.0 %	0	37	37

Bug Report

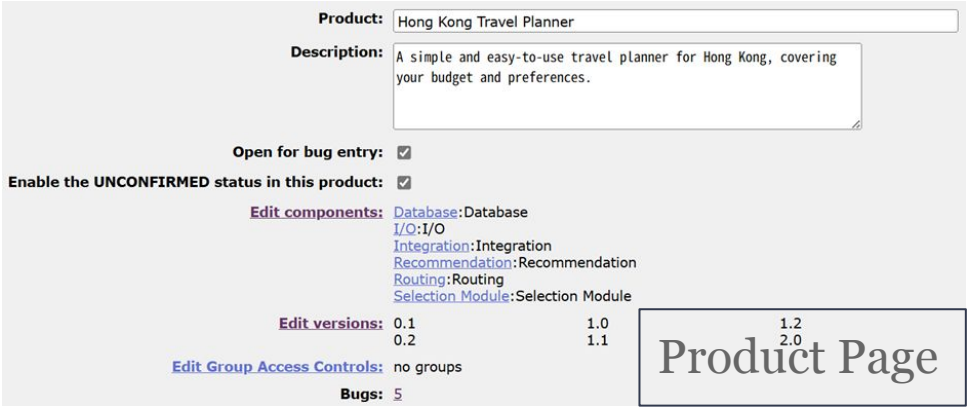
Connects Developers With Users



Bug Report: Repository

Why Bugzilla?

- Open-Source
- Widely Used in Industry
- Integration



Product: Hong Kong Travel Planner

Description: A simple and easy-to-use travel planner for Hong Kong, covering your budget and preferences.

Open for bug entry: ☑

Enable the UNCONFIRMED status in this product: ☑

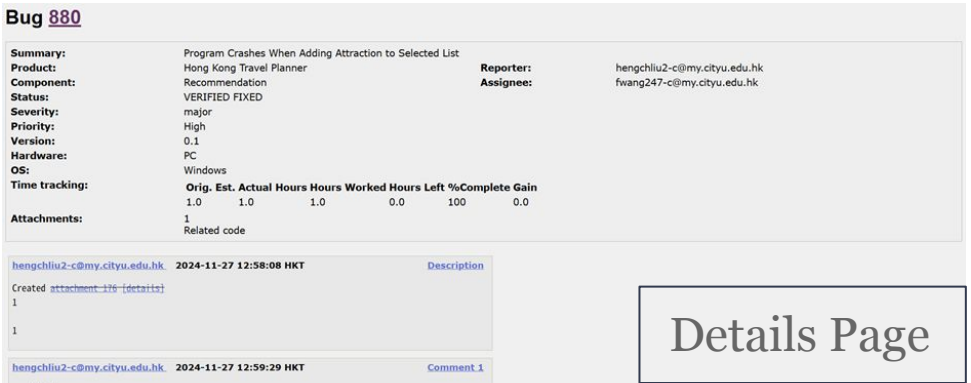
Edit components:
Database:Database
I/O:I/O
Integration:Integration
Recommendation:Recommendation
Routing:Routing
Selection Module:Selection Module

Edit versions: 0.1 0.2 1.0 1.1 1.2 2.0

Edit Group Access Controls: no groups

Bugs: 5

Product Page



Bug 880

Summary:	Program Crashes When Adding Attraction to Selected List		
Product:	Hong Kong Travel Planner	**Reporter:**	hengchliu2-c@my.cityu.edu.hk
Component:	Recommendation	**Assignee:**	fwang247-c@my.cityu.edu.hk
Status:	VERIFIED FIXED		
Severity:	major		
Priority:	High		
Version:	0.1		
Hardware:	PC		
OS:	Windows		

Time tracking:

Orig. Est.	Actual Hours	Hours Worked	Hours Left	%Complete	Gain
1.0	1.0	1.0	0.0	100	0.0

Attachments: 1
Related code

hengchliu2-c@my.cityu.edu.hk 2024-11-27 12:58:08 HKT Description

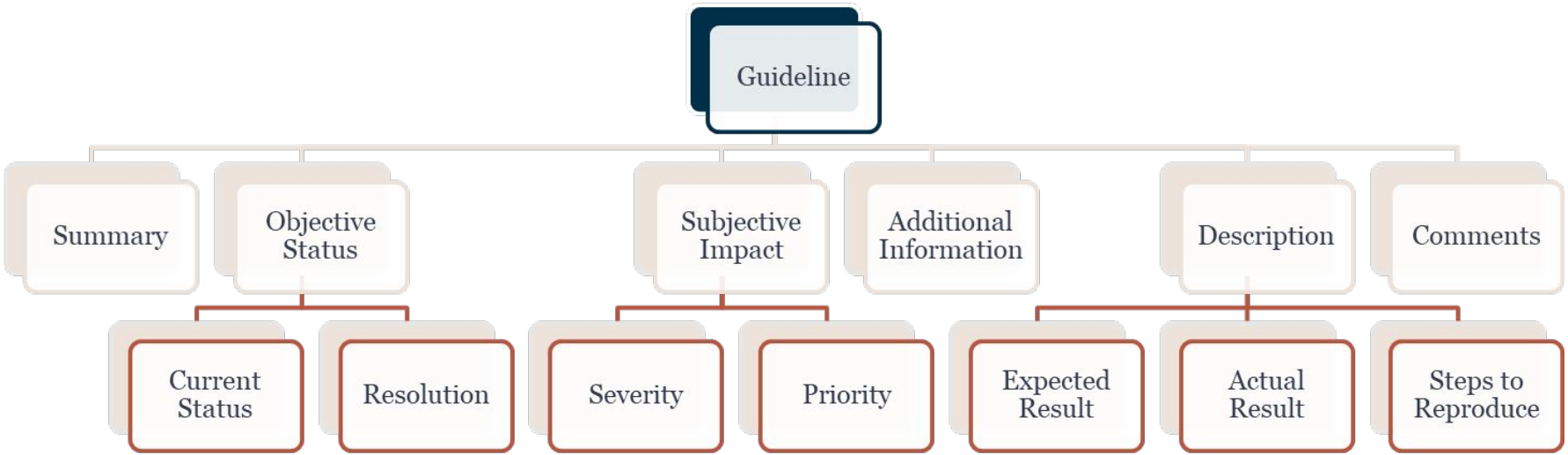
Created attachment 176 [details]
1

1

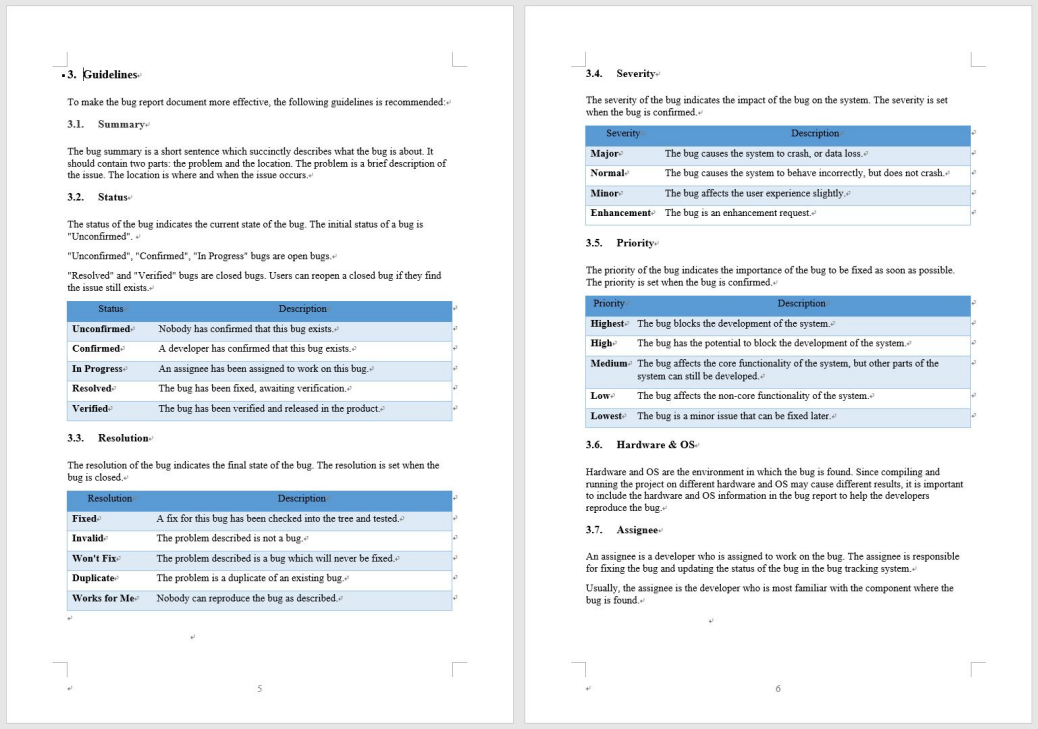
hengchliu2-c@my.cityu.edu.hk 2024-11-27 12:59:29 HKT Comment 1

Details Page

Bug Report: Guidelines



Bug Report: Guidelines & Templates



3. Guidelines

To make the bug report document more effective, the following guidelines is recommended.

3.1. Summary

The bug summary is a short sentence which succinctly describes what the bug is about. It should contain two parts: the problem and the location. The problem is a brief description of the issue. The location is where and when the issue occurs.

3.2. Status

The status of the bug indicates the current state of the bug. The initial status of a bug is "Unconfirmed".

"Unconfirmed", "Confirmed", "In Progress" bugs are open bugs.

"Resolved" and "Verified" bugs are closed bugs. Users can reopen a closed bug if they find the issue still exists.

Status	Description
Unconfirmed	Nobody has confirmed that this bug exists.
Confirmed	A developer has confirmed that this bug exists.
In Progress	An assignee has been assigned to work on this bug.
Resolved	The bug has been fixed, awaiting verification.
Verified	The bug has been verified and released in the product.

3.3. Resolution

The resolution of the bug indicates the final state of the bug. The resolution is set when the bug is closed.

Resolution	Description
Fixed	A fix for this bug has been checked into the tree and tested.
Invalid	The problem described is not a bug.
Won't Fix	The problem described is a bug which will never be fixed.
Duplicate	The problem is a duplicate of an existing bug.
Works for Me	Nobody can reproduce the bug as described.

5

3.4. Severity

The severity of the bug indicates the impact of the bug on the system. The severity is set when the bug is confirmed.

Severity	Description
Major	The bug causes the system to crash, or data loss.
Normal	The bug causes the system to behave incorrectly, but does not crash.
Minor	The bug affects the user experience slightly.
Enhancement	The bug is an enhancement request.

3.5. Priority

The priority of the bug indicates the importance of the bug to be fixed as soon as possible. The priority is set when the bug is confirmed.

Priority	Description
Highest	The bug blocks the development of the system.
High	The bug has the potential to block the development of the system.
Medium	The bug affects the core functionality of the system, but other parts of the system can still be developed.
Low	The bug affects the non-core functionality of the system.
Lowest	The bug is a minor issue that can be fixed later.

3.6. Hardware & OS

Hardware and OS are the environment in which the bug is found. Since compiling and running the project on different hardware and OS may cause different results, it is important to include the hardware and OS information in the bug report to help the developers reproduce the bug.

3.7. Assignee

An assignee is a developer who is assigned to work on the bug. The assignee is responsible for fixing the bug and updating the status of the bug in the bug tracking system.

Usually, the assignee is the developer who is most familiar with the component where the bug is found.

6

Bug Report Template

Summary:

Product:	**Hong Kong Journey Planner**
Component:	
Status:	<Unconfirmed/Confirmed/In Progress>
Severity:	<Major/Medium/Minor/Enhancement>
Priority:	<Highest/High/Normal/Low/Lowest>
Version:	
Hardware:	
OS:	
Reporter:	
Assignee:	

<Name> <Time>

Description:

A brief description of how bug happens.

Expected Result:

Expected outcome.

Actual Result:

Actual outcome. Provide logs and/or console outputs if possible.

Steps to Reproduce:

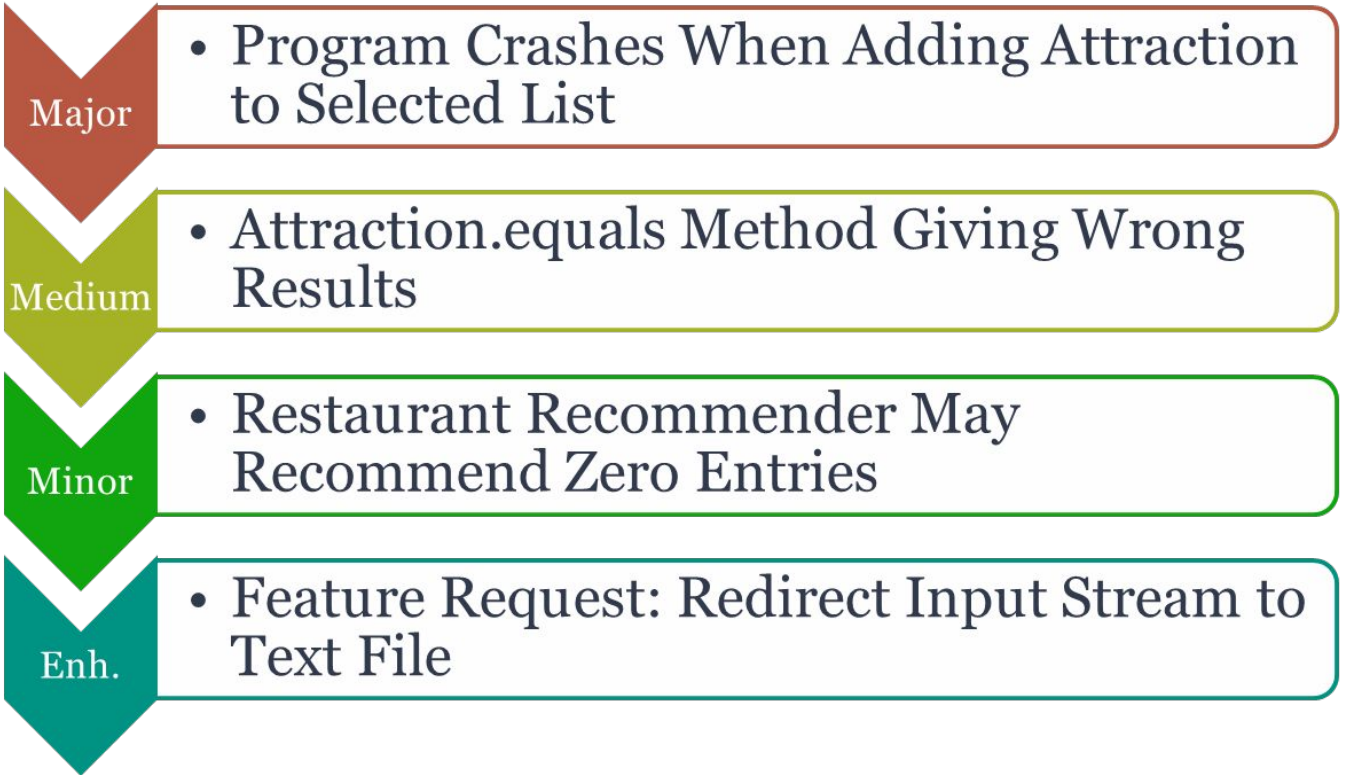
1. Run ……
2. Input ……
3. Observe ……

Related Code:

If you can locate where the problem is, provide as attachments.

Bug Report In Our Project

Bug Report Severity



Bug Report Example

4.1. Issue#1 – Restaurant Recommender Crashes with IndexOutOfBoundsException

Product:	**Hong Kong Journey Planner**
Component:	Selection Module
Status:	VERIFIED FIXED
Severity:	Major
Priority:	High
Version:	0.1
Hardware:	Machintosh
OS:	Mac OS
Reporter:	Cheung Lok Yi <lycheun24-c@my.cityu.edu.hk>
Assignee:	Liu Hengche <hengchliu2-c@my.cityu.edu.hk>

Bug Report Example

Cheung Lok Yi 2024-11-05 23:24:12 HKT

Description:

When testing the program, after entering all the user preferences and the recommender starts to work, I encountered an IndexOutOfBoundsException.

I am not sure what caused the error, but it seems to be related to the regions selected by the user. The error occurs when the program tries to get the file paths for a specific day.

Expected Result:

The program should not crash and should provide recommendations based on the user preferences.

Actual Result:

Failure Trace:

```
Exception in thread "main" java.lang.IndexOutOfBoundsException: Index 1 out of bounds for length 1
        at java.base/jdk.internal.util.Preconditions.outOfBounds(Preconditions.java:64)
        at java.base/jdk.internal.util.Preconditions.outOfBoundsCheckIndex(Preconditions.java:70)
        at java.base/jdk.internal.util.Preconditions.checkIndex(Preconditions.java:248)
        at java.base/java.util.Objects.checkIndex(Objects.java:374)
        at java.base/java.util.ArrayList.get(ArrayList.java:459)
        at UserPreferences_R.getFilePathsForDay(UserPreferences_R.java:68)
        at Recommender_R.recommendRestaurants(Recommender_R.java:16)
        at Main.main(Main.java:131)
```

Steps to Reproduce:

1. Run Main.java
2. Enter user preferences for the start date, end date, and budget. In my case, the start date is 2024-01-01, the end date is 2024-01-03, and the budget is 100-200.
3. Observe the program crash.

I observed that the program crashes immediately after entering the budget, which makes sense since Restaurant Recommender is called after the user preferences are entered.

Please investigate and fix this issue as soon as possible.

Related Code:

```java
// Main.java
    List<String> regions = new ArrayList<>();
    for (int i = 0; i < days; i++) {
        LocalDate currentDate = startDate.plusDays(i);
        String dateKey = currentDate.format(formatter);
        String selectedRegion = "";
        while (selectedRegion.isEmpty()) {
            System.out.println("Please select a region:");
            System.out.println("1. Hong Kong");
            System.out.println("2. Kowloon");
            System.out.println("3. New Territories");
            System.out.println("4. Outlying Islands");
            int regionChoice = scanner.nextInt();
            scanner.nextLine();
            switch (regionChoice) {
                // ...
            }
        }
        regions.add(selectedRegion);
    }
    dailyRegions.add(regions);
```

Dividing severe issues into manageable ones

Liu Hengche 2024-11-06 10:05:43 HKT

Received your bug report.

Liu Hengche 2024-11-06 10:17:21 HKT

Have you tried different number of days during testing?

Cheung Lok Yi 2024-11-06 11:20:12 HKT

Yes, I have tried with different number of days, including 1, 2, 3, and 4 days. The error does not occur when there is only 1 day.

Liu Hengche 2024-11-06 12:05:43 HKT

Well that probably makes sense. `dailyRegions` is of type `List<List<String>>`, so `dailyRegions.get(day)` will return a `List<String>` for the day.

In `Main.java`, regions in each day is recorded in regions, which is then put into `dailyRegions`. What we exactly want should be `regions.get(day)`, that is, a `String` for the day; instead of `dailyRegions.get(day)`, which is a `List<String>`. `regionToFilePaths.get(region)` accepts a `String` as the key, not a `List<String>`.

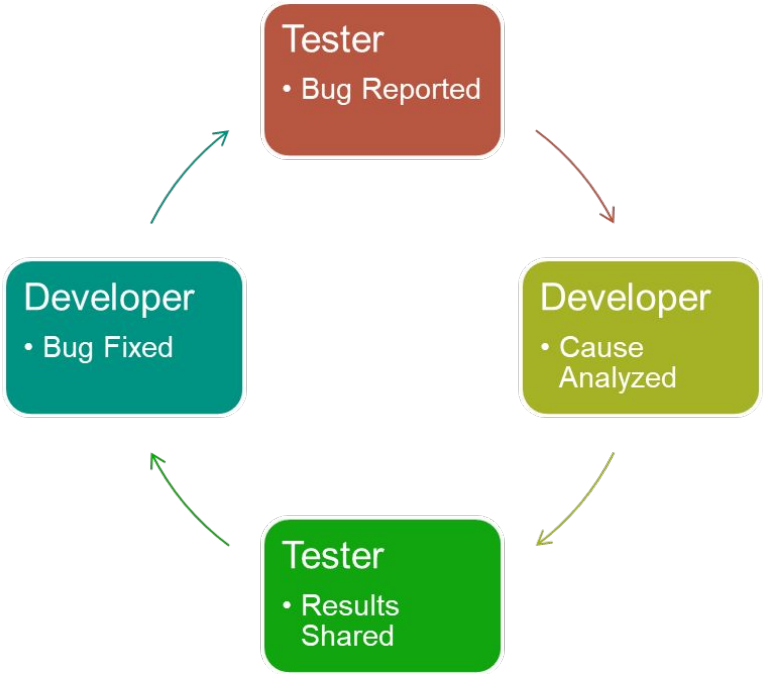
I will fix this issue and update you once it is done.

Liu Hengche 2024-11-06 12:28:41 HKT

I have fixed the issue. The program should now work as expected.

I removed `dailyRegions` and used `regions` instead.

```
// Main.java
    List<String> regions = new ArrayList<>();
    for (int i = 0; i < days; i++) {
        // ...
        regions.add(selectedRegion);
```



Technologies

Used tools

Development technologies

Use Of Technologies and Platforms

Database: Scraper Tools

Communication: Git

Development: Eclipse

Testing: Junit

Database: Web Scraping

```
import undetected_chromedriver as uc
```

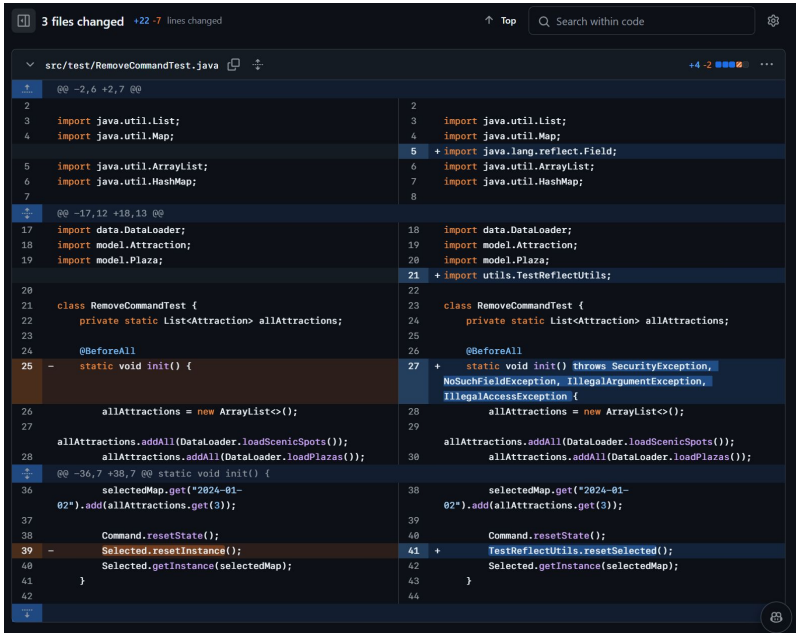
```
# Set up Chrome options (optional)
chrome_options = Options()
chrome_options.add_argument("--headless")  # Run in headless mode (no UI)
# new options
chrome_options.add_argument("--disable-gpu")  # Disable GPU acceleration
chrome_options.add_argument("--no-sandbox")  # Bypass OS security model
chrome_options.add_argument("window-size=1920x1080")  # Set window size
chrome_options.add_argument("--disable-dev-shm-usage")  # Overcome limited resource problems
chrome_options.add_argument("--blink-settings=imagesEnabled=false")  # Disable images

# Specify the path to your ChromeDriver
service = Service(r"D:\desktop\courses\3343\小组作业\chromedriver-win64\chromedriver.exe")  # Update this path

# Initialize the webdriver
driver = webdriver.Chrome(service=service, options=chrome_options)

# Visit the URL
url = "https://www.mtr.com.hk/en/customer/jp/index.php?oLabel=Kowloon&oType=HRStation&oValue=45&dLabel=Mei%20Foo&dType=HRStation&dValue=20"
driver.get(url)

# Wait for the page to load (you may need to adjust the sleep time)
time.sleep(5)

# Find the span with class "time"
time_element = driver.find_element(By.CLASS_NAME, 'time')

# Output the text content (e.g., "13 mins")
print(time_element.text)

# Close the browser
driver.quit()
```

[12]

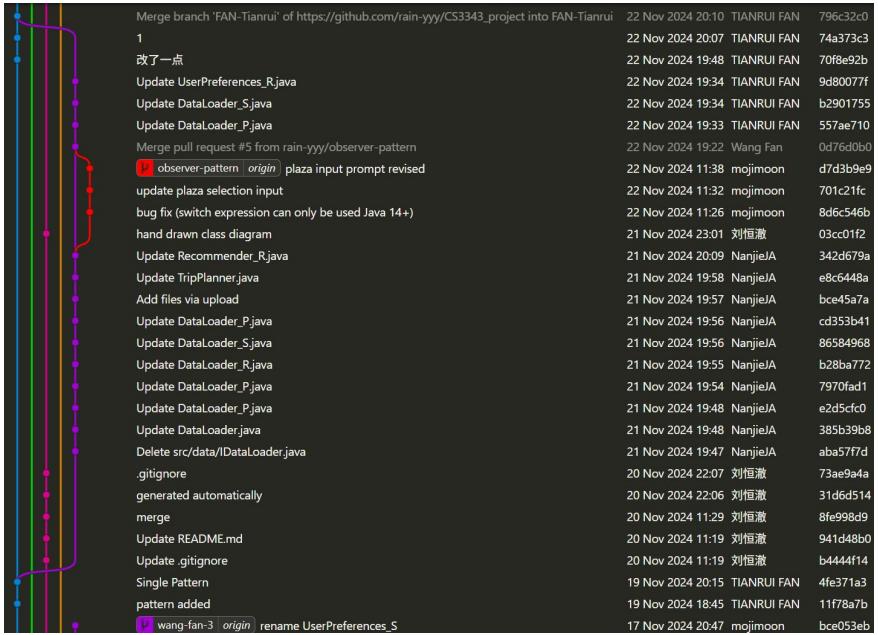
```
13 mins
```

Time between stations

Communication: Git Version Control

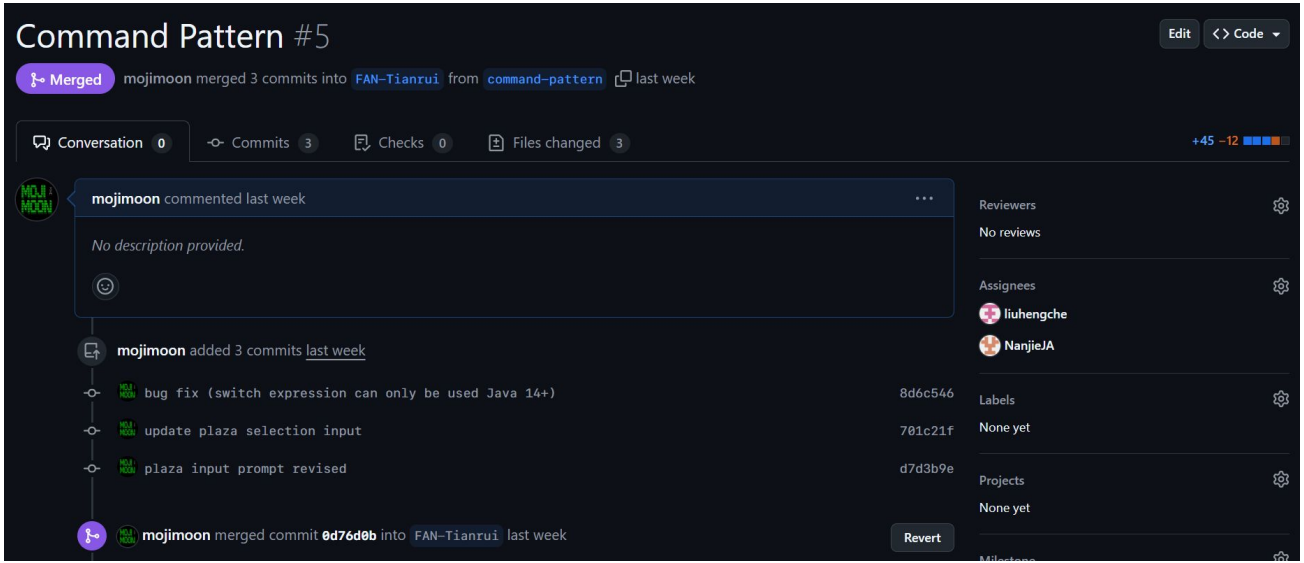


Record Changes Version-by-version



Multi-branch, Concurrent Cooperation

GitHub: Merge and Assignee Management



Development Environment

- Eclipse 2024-09
- Copilot4Eclipse
- Java 11
- JUnit 5
- Jackson (for Json)

Testing: Junit

```
    });
}

@Test
void testGetStartDate() {
    LocalDate expectedStart = LocalDate.of(2024, 3, 1);
    assertEquals(expectedStart, normalDateRange.getStartDate());
}

@Test
void testGetEndDate() {
    LocalDate expectedEnd = LocalDate.of(2024, 3, 5);
    assertEquals(expectedEnd, normalDateRange.getEndDate());
}

@Test
void testGetDaysNormalRange() {
    assertEquals(5, normalDateRange.getDays());
}

@Test
void testGetDaysSingleDay() {
    assertEquals(1, singleDayRange.getDays());
}

@Test
void testGetAllDatesNormalRange() {
    List<LocalDate> dates = normalDateRange.getAllDates();

    assertEquals(5, dates.size());

    assertEquals(LocalDate.of(2024, 3, 1), dates.get(0));
    assertEquals(LocalDate.of(2024, 3, 5), dates.get(dates.size() - 1));

    for (int i = 0; i < dates.size() - 1; i++) {
        assertEquals(1, ChronoUnit.DAYS.between(dates.get(i), dates.get(i + 1)));
    }
}
```

Junit Testing

Q&A

Please feel free to ask any questions.

Have a nice trip ~

Thank You !