Object-Oriented Analysis and Design

Multifunctional Memo with Calendar

Homework #7

團隊成員

資工碩一	**鍾亮節**	**112598014**
資工碩一	**張鈺亭**	**112598023**
資工四	**侯育名**	**109590006**

目錄

第一章 Requirement Document

1.1 Change History

Revision	Description	Date
1.1	Problem statement	2024/02/28
1.2	Change History Summary of System Features Use Case Diagram Use Cases Non-functional Requirements and Constraints Glossary	2024/03/13
1.3	Domain class diagram showing only concepts Add associations Add attributes	2024/03/18 2024/03/22
1.4	Implementation Class Model Programming	2024/05/01
1.5	Modify Use cases Modify Domain Class Model Modify Implementation Class Model	2024/05/22
1.6	Modify Implementation Class Model Use Case Diagram Implementation class diagram Calculate Line of Code Snapshots of system execution Source Code Listing Snapshots of testing result	2024/06/11

1.2 Problem statement

這個 web application 是一個簡單使用的備忘錄結合行事曆的軟體。

每個人的日常生活中都有許多事項需要處理，有個方便列出待辦事項並且清楚表示未來行程規劃的工具能夠使規劃簡單許多，例如進行小組作業的學生們可以使用這個軟體進行代辦事項的共享，並一同規劃未來行程。

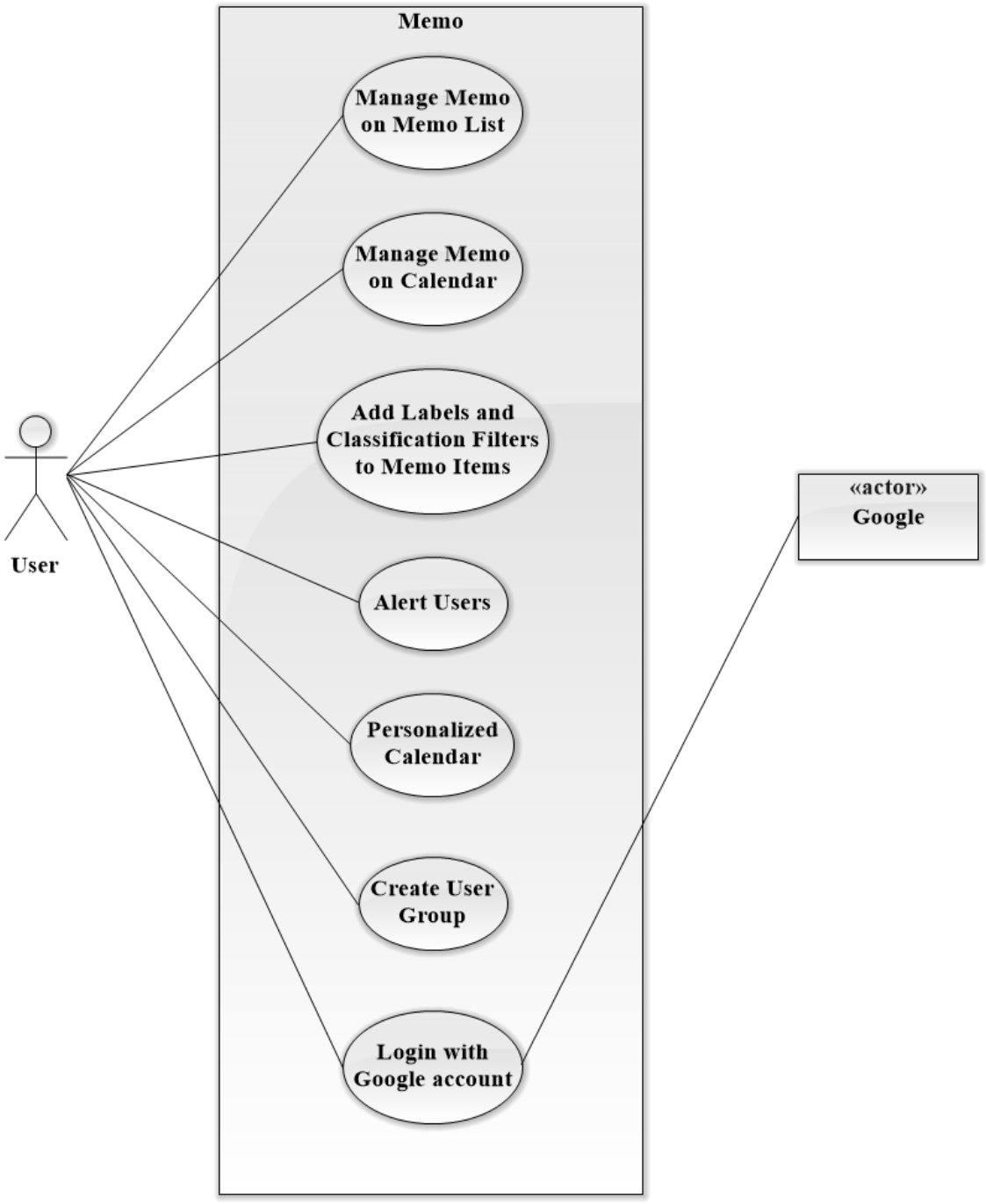
這個軟體將藉由網頁，使用 HTML 搭配 CSS 語法做介面設計，後端功能邏輯運作使用 Java 進行設計，採前後端分離，由後端做運算及事件處理再交由前端呈現。

使用者可以新增待辦項目並描述該事項的細節，加入日期便可在行事曆上做出標記，方便看出何時有行程需要處理，並且可以加入事項的重要性或分類標籤，在待辦事項列表中可以使用這兩項因素來做排序與整理。

1.3 Summary of System Features

Feature ID	Description
FEA-01	編輯管理備忘項目
FEA-02	群組的建立管理
FEA-03	以行事曆管理備忘項目
FEA-04	通知使用者完成備忘項目

1.4 Use Case Diagram



1.5 Use Cases

Use case ID	UC-01
Use case Name	Manage Memo on Memo List(從備忘錄列表建立備忘

	項目)
Scope	Memo Web Application
Level	User goal
Primary Actor	User(使用者)
Stakeholders and interests	User: 使用者想要建立自己的備忘項目於備忘錄中
Preconditions	使用者已連接網路並開啟網站
Success Guarantee	使用者成功建立備忘項目
Main Success Scenario	1. 使用者點選新增備忘項目於備忘錄中 2. 使用者輸入備忘項目 3. 點選建立 4. 網頁顯示剛才建立的備忘項目
Extensions	*a. 使用者未連接網路，則跳出提示框顯示"未連接網路"。 3a. 未輸入或輸入空白，則無法建立備忘錄，網頁顯示建立失敗，回到 2。
Special Requirements	使用者連接網路並開啟網頁

Technology and Data Variations List	無
Frequency of Occurrence	經常發生
Miscellaneous	無

Use case ID	UC-02
Use case Name	Create User Group(建立使用者群組)
Scope	Memo web application
Level	User goal
Primary Actor	User
Stakeholders and interests	User: 使用者想要建立群組來共用備忘項目
Preconditions	使用者已登入或註冊
Success Guarantee	使用者成功建立群組
Main Success Scenario	1. 使用者點選新增群組 2. 輸入欲加入的使用者帳號或 ID 3. 選擇使用者帳號加入 4. 點選建立

	5. 網頁顯示群組人員與共用的備忘項目
Extensions	*a. 使用者未連接網路，則跳出提示框顯示"未連接網路"。 2a. 輸入的使用者不存在，回到 2.
Special Requirements	使用者連接網路並開啟網頁
Technology and Data Variations List	無
Frequency of Occurrence	偶爾發生
Miscellaneous	無

Use case ID	UC-03
Use case Name	Manage Memo on Calendar(從行事曆建立備忘項目)
Scope	Memo web application
Level	User goal
Primary Actor	User
Stakeholders and interests	User: 使用者想要建立自己的備忘項目於行事曆中
Preconditions	使用者已連接網路並開啟網站

Success Guarantee	1. 使用者點選行事曆中的新增備忘項目 2. 使用者輸入自己的備忘項目 3. 點選建立 4. 網頁將顯示行事曆中剛建立的備忘項目
Main Success Scenario	使用者成功於行事曆中新增備忘項目
Extensions	*a. 使用者未連接網路，則跳出提示框顯示"未連接網路"。 3a. 未輸入或輸入空白，則無法建立備忘錄，網頁顯示建立失敗，回到 2
Special Requirements	使用者連接網路並開啟網頁
Technology and Data Variations List	無
Frequency of Occurrence	經常發生
Miscellaneous	無

Use case ID	UC-04

Use case Name	Set Classification(制定分類)
Scope	Memo web application
Level	user-goal
Primary Actor	User
Stakeholders and interests	User: 使用該系統，想要得到被加上分類的備忘項目
Preconditions	使用者已新增一個或以上的備忘項目
Success Guarantee	使用者對備忘項目加上的標籤被正確的儲存。
Main Success Scenario	1. 使用者選取一個備忘項目 2. 使用者新增標籤 3. 使用者填寫合法的標籤文字並儲存 4. 系統將使用者填寫的資訊儲存
Extensions	*a. 使用者未連接網路，則跳出提示框顯示"未連接網路"。4a. 若無正確儲存，返回錯誤訊息，並跳出提示框顯示"儲存失敗，請再次嘗試"，回到3
Special Requirements	1. 操作過程網路狀態皆正常連接。

	2. 標籤欄位需輸入合法字串。
Technology and Data Variations List	無
Frequency of Occurrence	經常發生
Miscellaneous	無

Use case ID	UC-05
Use case Name	Alert User(發出通知)
Scope	Memo web application
Level	User-goal
Primary Actor	Memo web application
Stakeholders and interests	User: 想要接收到通知 Memo: 負責發出通知。
Preconditions	使用者已新增一個以上有設定通知時間的備忘項目
Success Guarantee	Memo 於正確的時間發出通知
Main Success Scenario	1. 使用者選取一個備忘項目 2. 使用者新增通知時間設定

	3. 使用者設定時間並儲存 4. Memo 將設定儲存 5. 到設定的時間時 Memo 發出通知
Extensions	*a. 使用者未連接網路，則跳出提示框顯示"未連接網路"。 1a. 若無備忘項目存在，則無對象物件可以執行。 3a. 若使用者將時間設定在過去，則跳出提示框提醒使用者是否設置正確。 4a. 若無正確儲存，返回錯誤訊息，並跳出提示框顯示"儲存失敗，請再次嘗試"，回到 3。
Special Requirements	1. 使用者須於設定的時間到時也將頁面開著並連接網路才能收取通知。
Technology and Data Variations List	無
Frequency of Occurrence	經常發生
Miscellaneous	由於網頁動作大部分與網路以及系統是否運行中有關，可以考慮是否將通知功能加入使用者的系統中等

	方法試著與網頁運作脫離，使得不須一直開啟頁面也能夠照常收到通知。

Use case ID	UC-06
Use case Name	Personalized calendar (個人化行事曆)
Scope	Memo web application
Level	User-goal
Primary Actor	User
Stakeholders and interests	User: 想要自行設定並取得符合使用習慣的行事曆。
Preconditions	使用者連接網路並開啟網頁
Success Guarantee	使用者操作自訂頁面過程沒有發生錯誤，並且行事曆外觀符合使用者預期。
Main Success Scenario	1. 使用者按下自訂行事曆的選項。 2. 使用者操作自訂介面並按下儲存。 3. 系統儲存使用者設置的設定。 4. 行事曆呈現使用者所設定的樣式。
Extensions	*a. 使用者未連接網路，則跳出提示框顯示"未連接網

	路"。 3a. 若無正確儲存，返回錯誤訊息，並跳出提示框顯示"儲存失敗，請再次嘗試"，回到 2。
Special Requirements	1. 操作過程網路狀態皆正常連接。
Technology and Data Variations List	無
Frequency of Occurrence	偶爾發生
Miscellaneous	需考量到能夠客製化多少元件，並對應確認是否有可能發生錯誤的情境需要處理。

Use case ID	UC-07
Use case Name	Login with Google account (使用 Google 帳號登入系統)
Scope	Memo web application
Level	User-goal
Primary Actor	User
Stakeholders and interests	User: 想要使用已有的 Google 帳號登入本系統

Preconditions	使用者連接網路
Success Guarantee	使用者成功登入系統
Main Success Scenario	1. 使用者開啟網頁 2. 使用者使用 Google 帳號通過認證並登入 3. 使用者成功登入並進入系統主頁面
Extensions	*a. 使用者未連接網路，則跳出提示框顯示"未連接網路"。 2a. 若未使用正確的 Google 帳號登入，應提示錯誤並回到 2。
Special Requirements	1. 操作過程網路狀態皆正常連接。
Technology and Data Variations List	無
Frequency of Occurrence	偶爾發生
Miscellaneous	無

1.6 Non-functional Requirements and Constraints

NFR ID	Category	Description
NFR-01	Testability	Code Coverage 至少 85% 以上

NFR-02	Supportability	支援多個瀏覽器 Ex： edge、chrome、safari
NFR-03	Usability	提供方便使用的使用者介面
NFR-04	Performance	系統反應速度不超過 1 秒
NFR-05	Performance	系統能夠存取最少 10000 個使用者帳戶

1.7 Glossary

Item	Definition or Description
備忘項目 **(Memo item)**	由使用者建立，用於記住和提醒使用者該完成此事項。
備忘表 **(Memo list)**	用以儲存使用者建立的多個備忘項目。
群組 **(Group)**	多個使用者共同使用與管理同一組備忘表和行事曆。
使用者 **(User)**	會在網頁上建立正式帳戶以使用所有網頁功能。
標籤 **(Label)**	用於分類備忘項目的類別，可在篩選備忘項目時發揮作用。
通知 **(Alert)**	當來到備忘項目的通知時間時，透過訊息或聲音來提醒使用者該來完成備忘項目。

1.8 Software Environments

前端介面採用 HTML 及 CSS 設計 後端採用 Java 語法設計。

第二章 Domain Class Model

2.1 Domain class diagram showing only concepts

2.1.1 Classes Identified

User	自訂行事曆的選項 (Custom calendar options)	Memo list	Memo item	網頁 (Webpage)
Group	使用者帳號 (User account)	ID	Calendar	System
Label	備忘項目描述 (Memo item description)	Alert	標籤字串 (Label text)	通知時間 (Alert time)
自訂介面 (Custom interface)	**認證**(Authorization)			

以上為使用 Identify noun phrases 在 Use Cases 所找出的單詞

2.1.2 Bad Classes

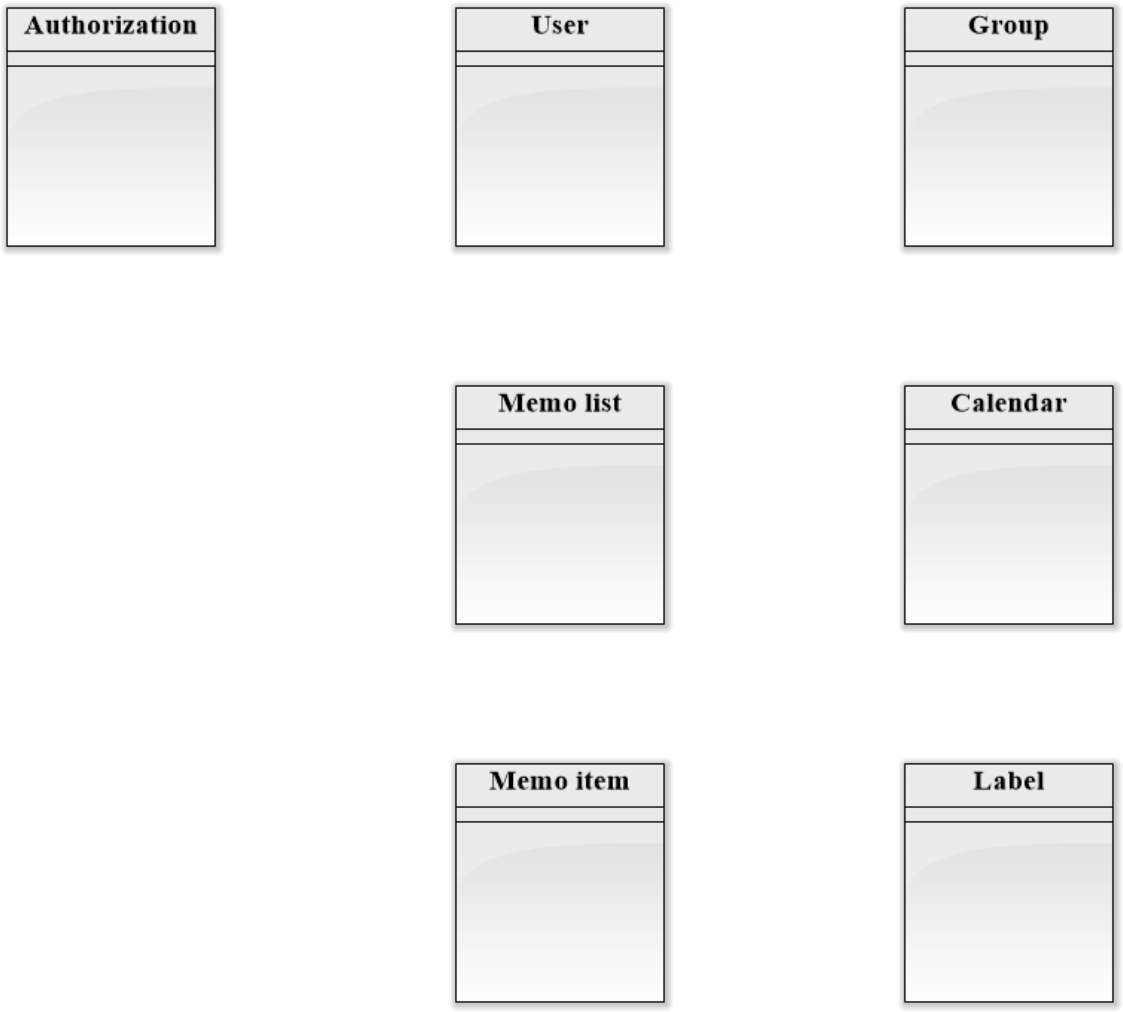
Attributes	Vague	Operation	Roles	Implementation Construction
使用者帳號 (User account)	自訂介面 (Custom interface)	Alert	網頁 (Webpage)	
ID	自訂行事曆的選項 (Custom calendar options)		System	
備忘項目描述 (Memo item description)				
標籤字串 (Label text)				
通知時間 (Alert time)				

以上將不好的 Classes 歸類成五大類

- Attributes：列為屬性
- Vague：定義模糊不清
- Operation：操作步驟中的一部份
- Roles：角色介面
- Implementation Construction：實作產生的結構

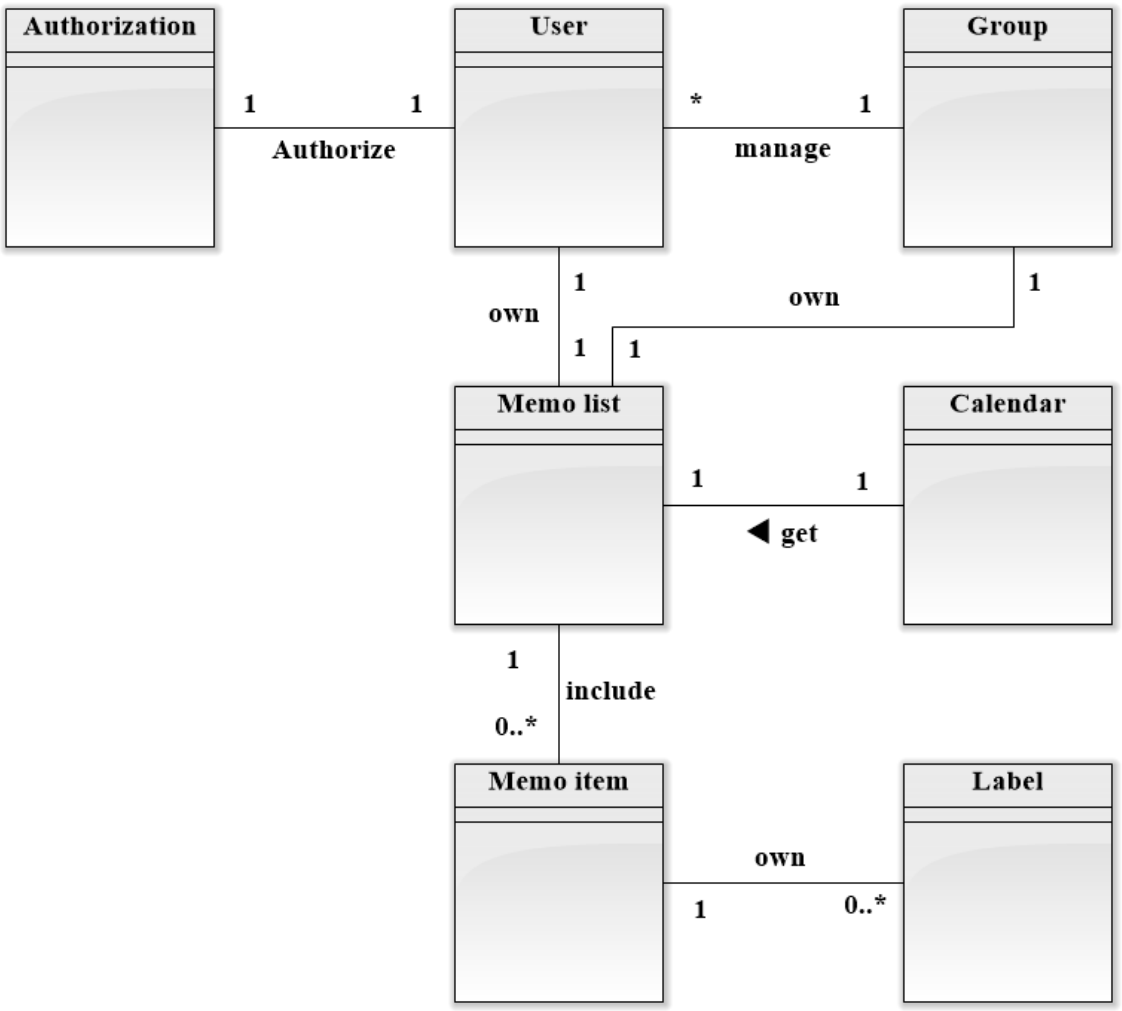
2.1.3 Good Classes

User	Label	Memo list	Memo item	Group
Calendar	Authorization			

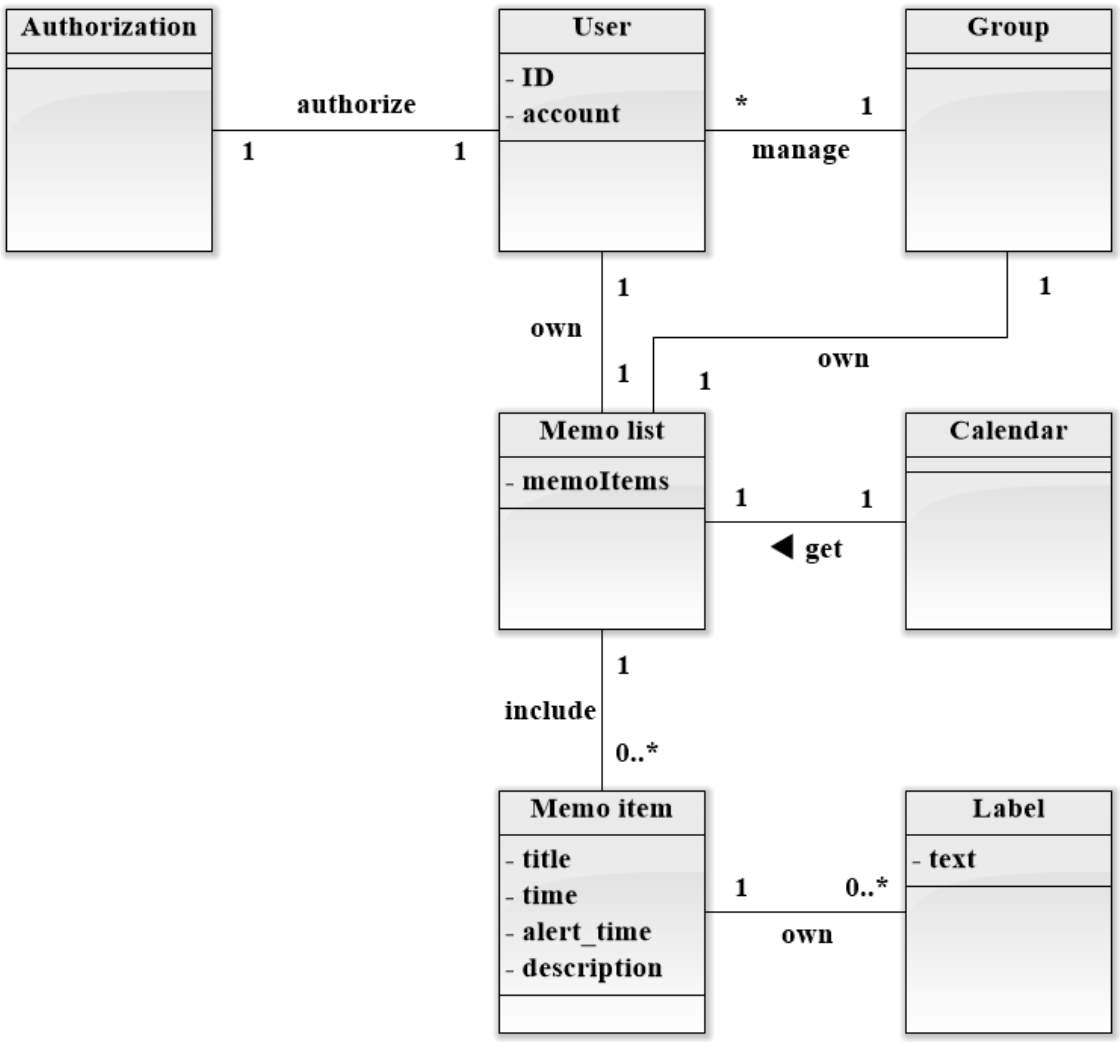


2.2 Add associations

- One User manages Memo items of Memo list
- One User owns one Memo list
- Several Users manage one Group
- One Memo list includes several Memo items
- One Group owns one Memo list
- One Memo item owns several Labels
- Calendar gets Memo items included in Memo list
- One Authorization Authorize One User

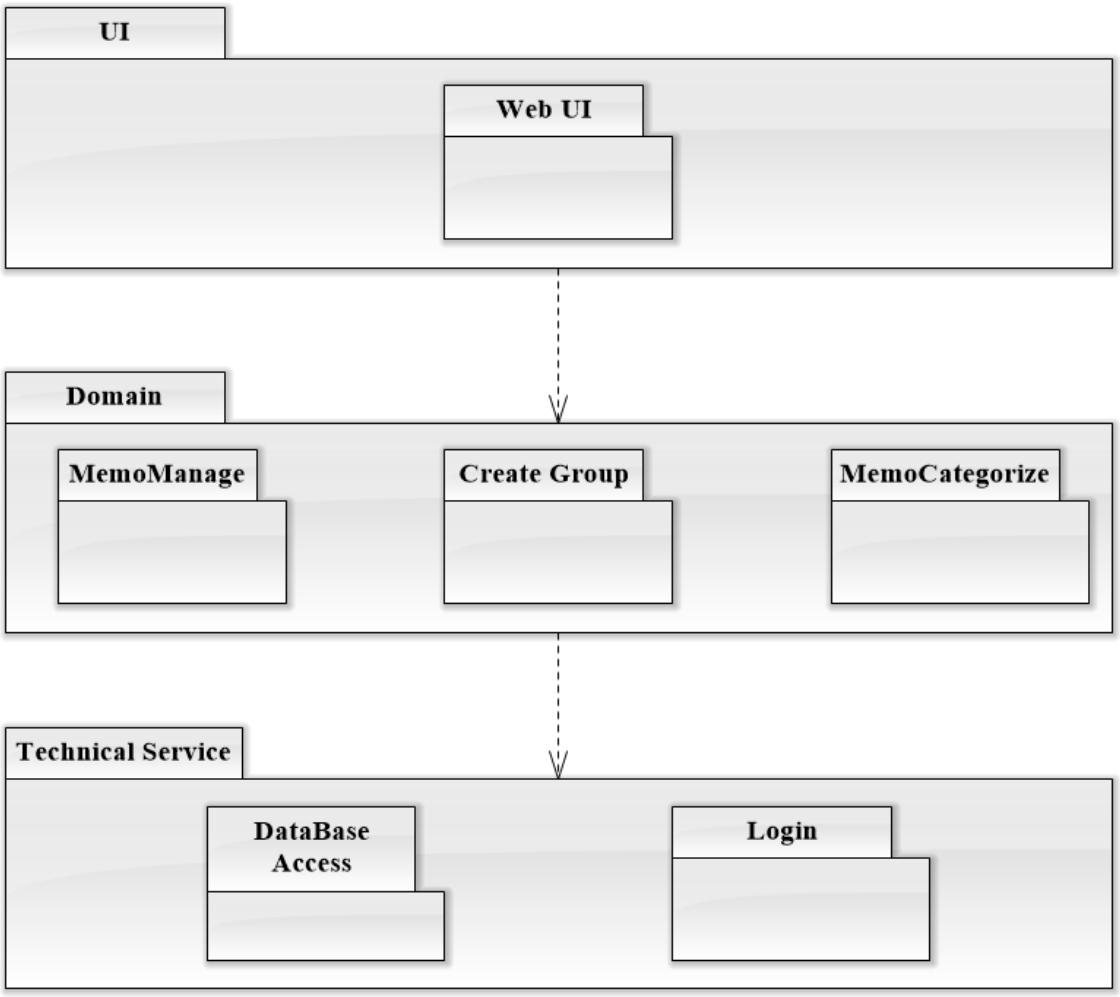


2.3 Add attributes


Authorization
User
- ID
- account
Group
authorize
1
1
*
1
manage
1
own
1
1
1
own
Memo list
- memoItems
Calendar
1
1
get
1
include
0..*
Memo item
- title
- time
- alert_time
- description
Label
- text
1
0..*
own

第三章 Design

3.1 Logical Architecture



3.2 Use-Case Realizations with GRASP Patterns

3.2.1 SSD(System Sequence Diagram)

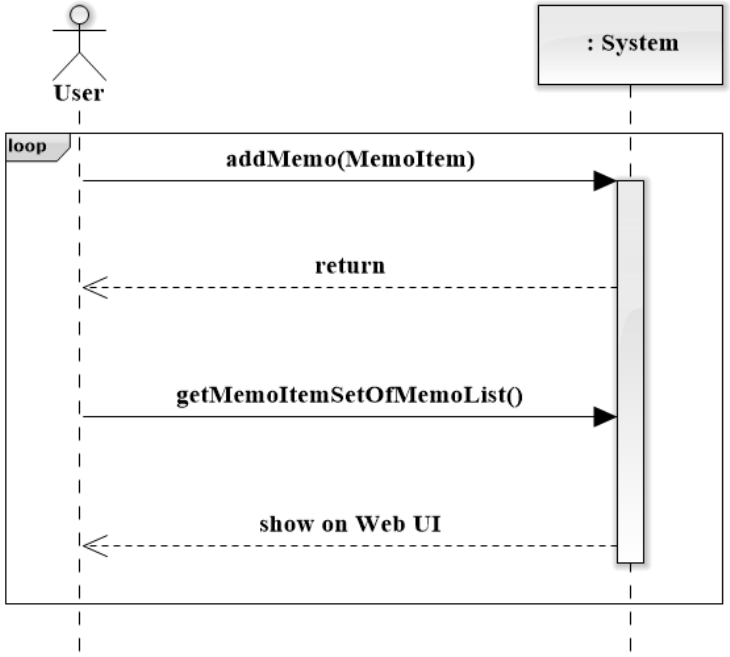


Figure 3-1 System Sequence Diagram of use case: Manage Memo on Memo List

3.2.2 Sequence Diagram

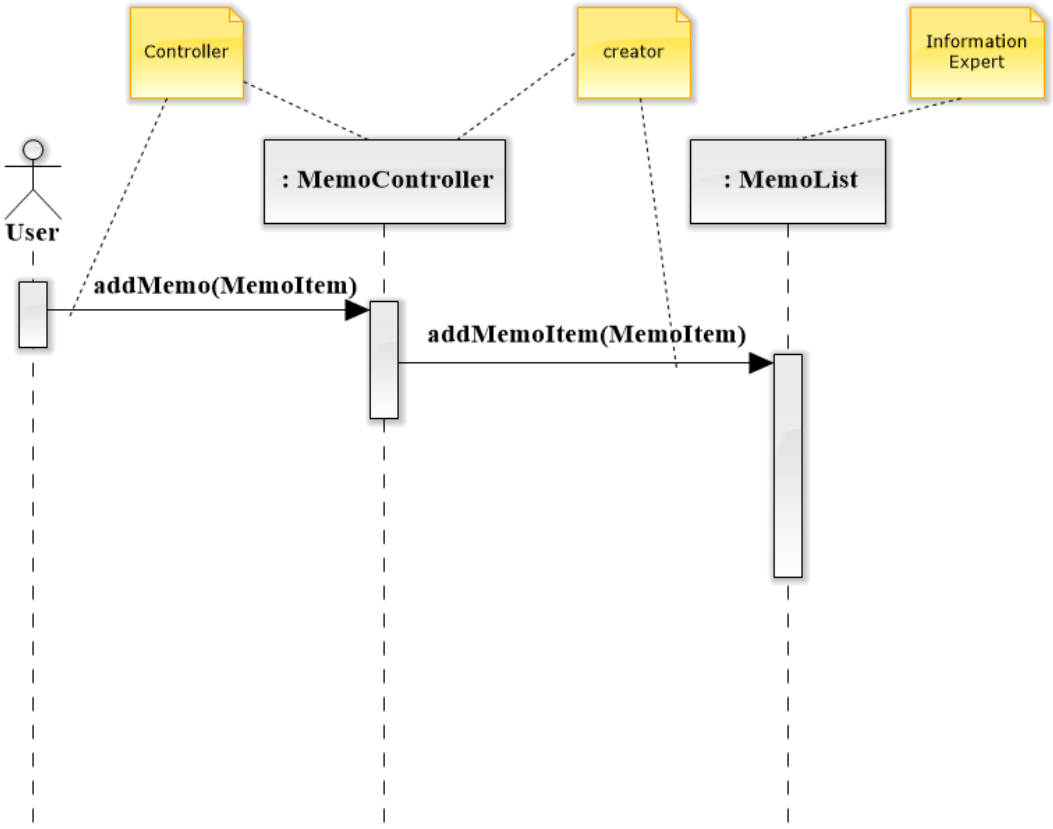


Figure 3-2 Sequence Diagram: addMemo

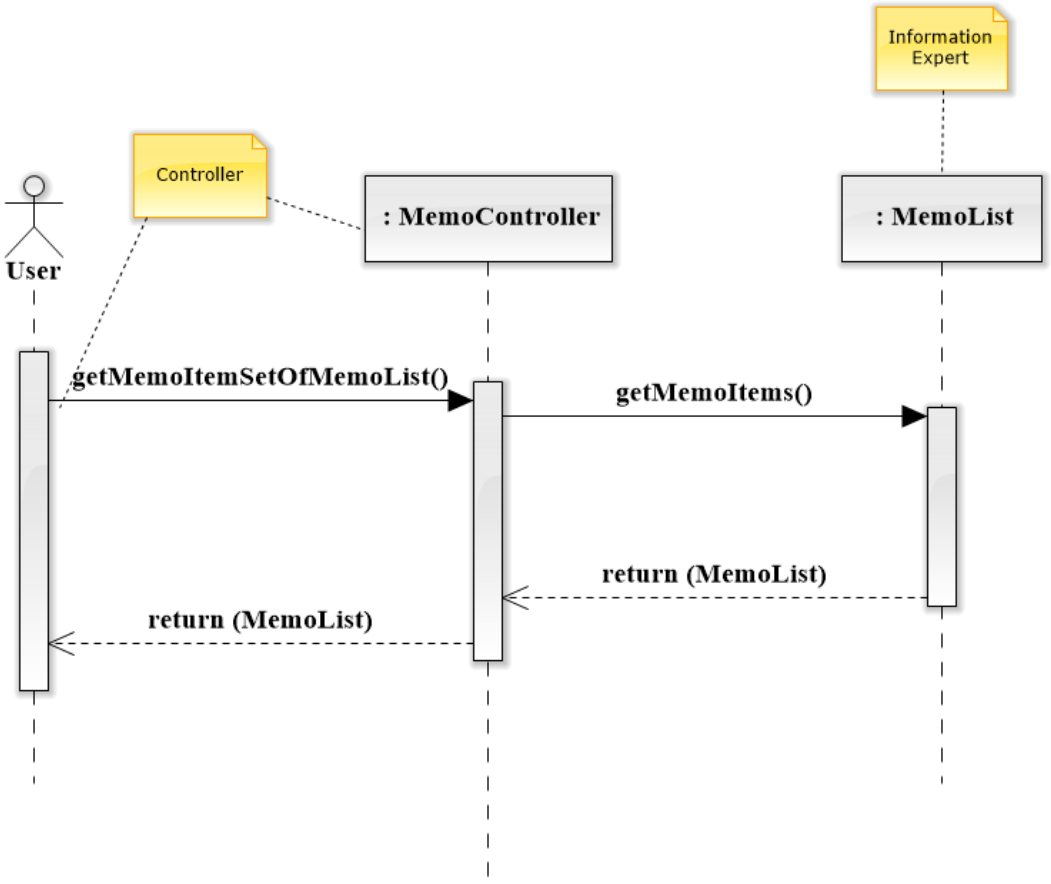
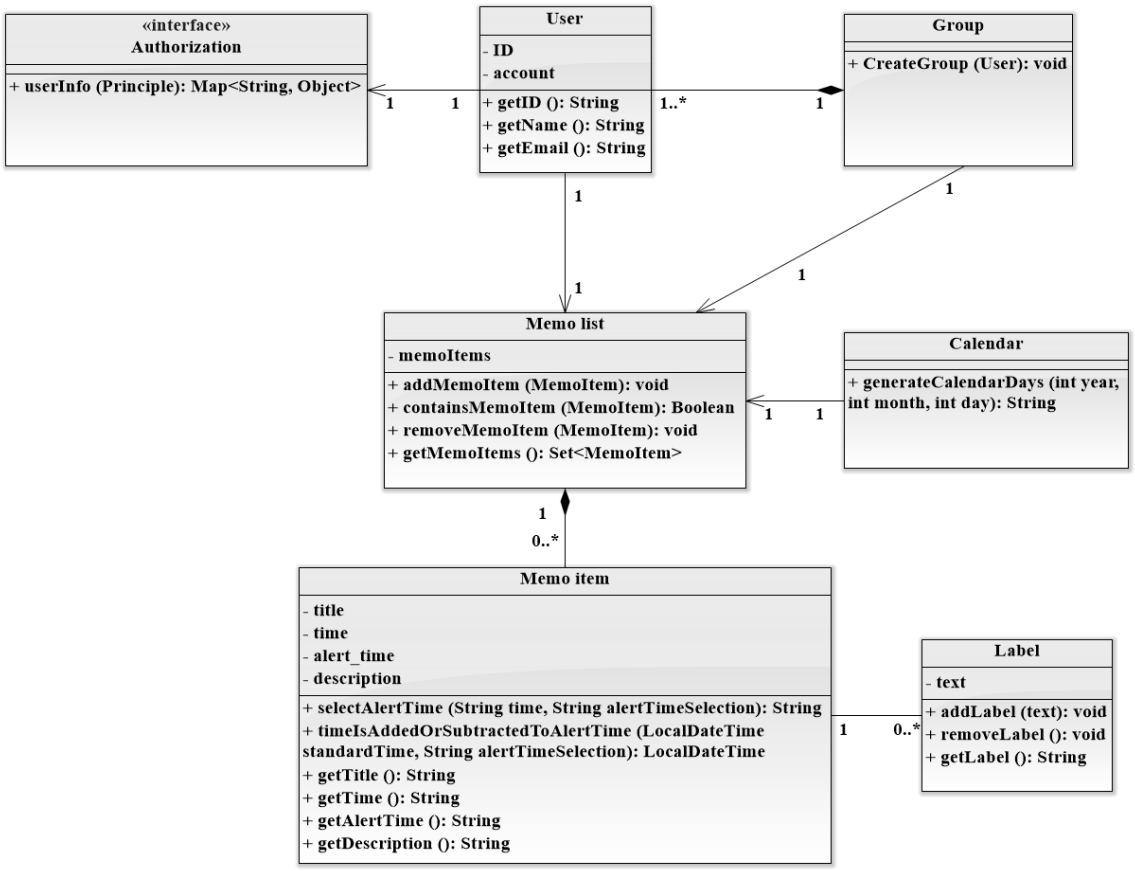


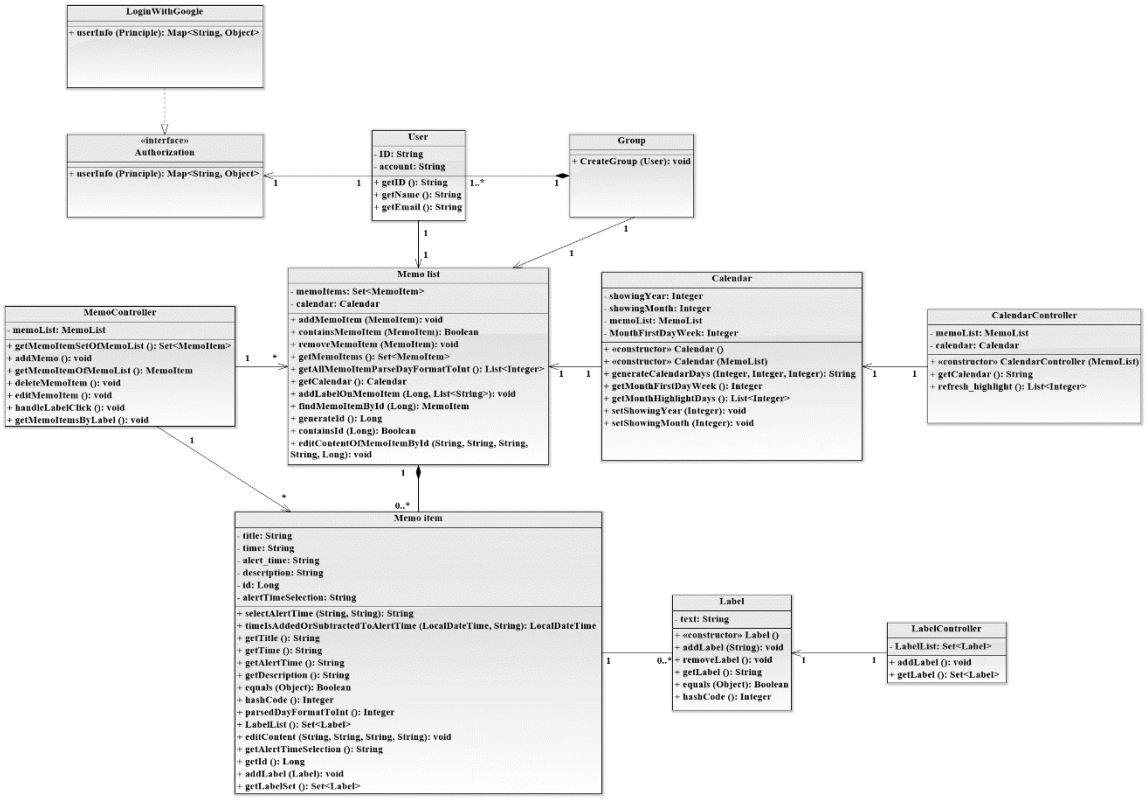
Figure 3-3 Sequence Diagram: getMemoItemSetOfMemoList

3.3 Design Class Model (domain class model after class design)



第四章 Implementation Class Model

4.1 Implementation class diagram



4.2 Show the difference between implementation class model and design class model

Construct the following two tables:

Table 4.2.1: Comparison with design and implementation class

Class	Method	Design	Imp.
Calendar	generateCalendarDays	Yes.	Yes.
	getMonthFirstDayWeek	No.	Yes.
CalendarController	getCalendar	No.	Yes.
Authorization	userInfo	Yes.	Yes.
LoginWithGoogle	userInfo	No.	Yes.
MemoController	getMemoItemSetOfMemoList	Yes.	Yes.
	addMemo	Yes.	Yes.
MemoItem	SelectAlertTime	Yes.	Yes.
	timeIsAddedOrSubtractedToAlertTime	Yes.	Yes.
	getTitle	Yes.	Yes.
	getTime	Yes.	Yes.
	getAlertTime	Yes.	Yes.
	getDescription	Yes.	Yes.
	equals	No	Yes
	hashCode	No	Yes
MemoList	addMemoItem	Yes.	Yes.

	containsMemoItem	Yes.	Yes.
	removeMemoItem	Yes.	Yes.
	getMemoItems	Yes.	Yes.
User	getId	Yes.	Yes.
	getName	Yes.	Yes.
	getEmail	Yes.	Yes.
label	addLabel	Yes.	Yes.
	removeLabel	Yes.	Yes.
	getLabel	Yes.	Yes.
Group	CreateGroup	Yes.	Yes.

Table 4.2.2: Summary of implementation class/method changed

	Number of added	Number of removed	Number of modified
Class	2	0	0
Method	4	0	0

4.3 Calculate Line of Code

Table 4.3.1: Line of Code of classes

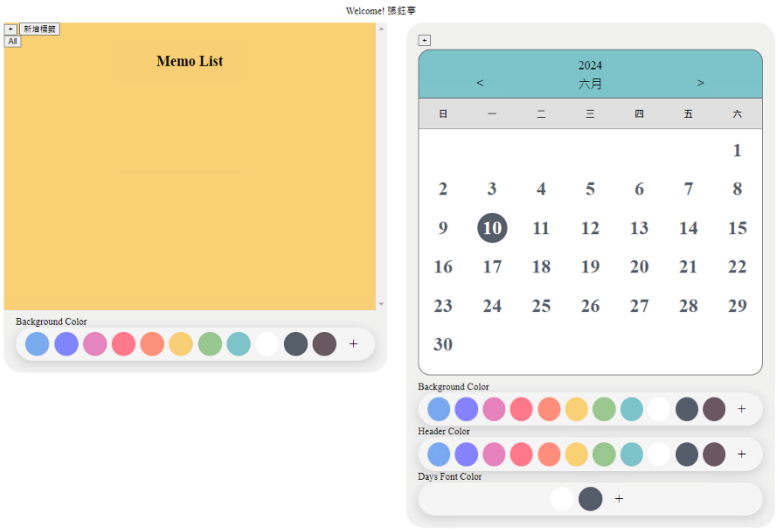
No.	Class Name	Number of methods	Line of Code in Class (without comment)
1	Calendar	3	90
2	CalendarController	2	28
3	Authorization	2	18
4	MemoController	3	32
5	MemoBackendApplication	1	9
6	MemoItem	10	125
7	MemoList	9	79
8	MvcConfig	1	11
9	SecurityConfig	1	23
10	ServletInitializer	1	9
11	User	3	20
12	LoginWithGoogle	2	28
13	Label	5	21
14	LabelController	2	19
total		45	512

第五章 Programming

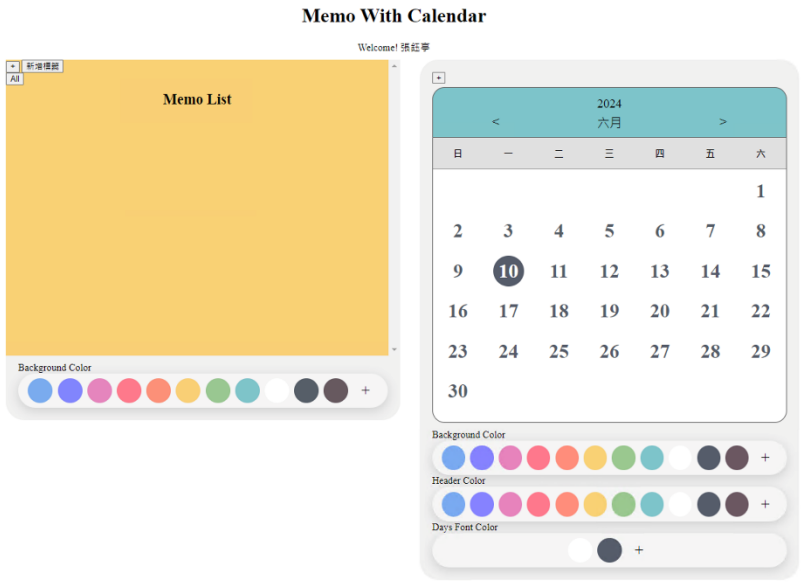
5.1 Snapshots of system execution

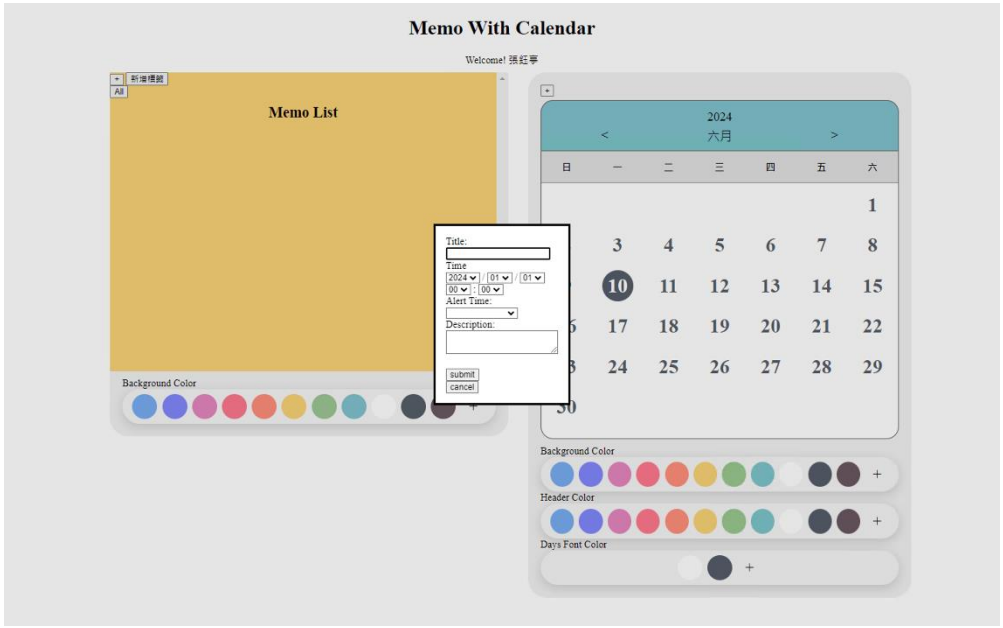
Login

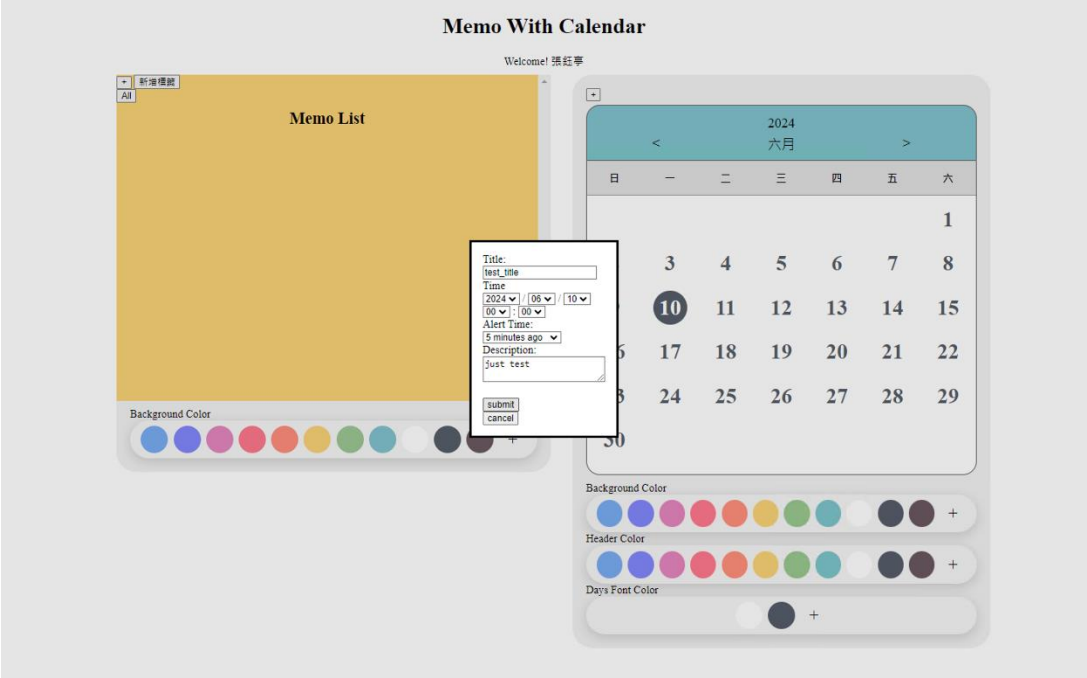
Memo With Calendar

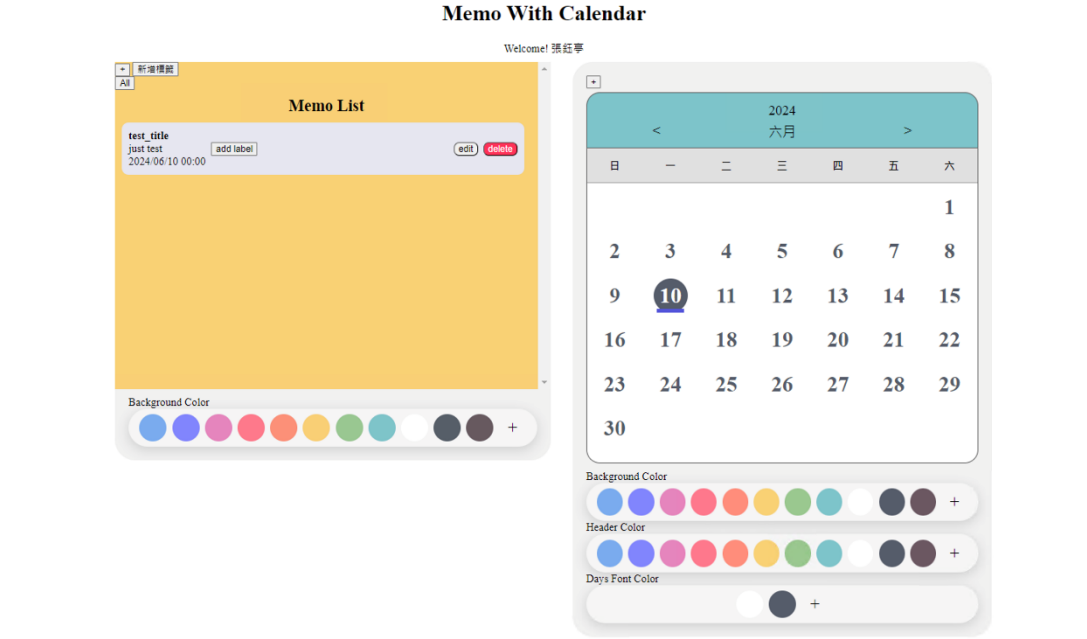


Add Memo Item









5.2 Source Code Listing

Calendar.java

```
package com.example.memobackend;
import java.time.LocalDate;
import java.time.YearMonth;
import java.util.ArrayList;
import java.util.List;


public class Calendar {
    private int MonthFirstDayWeek;
    private int showingYear, showingMonth;
```

```
    private MemoList memoList;

    public Calendar() {
    }

    public Calendar( MemoList memoList) {
        this.memoList = memoList;
    }

    public String generateCalendarDays(int year, int month, int day) {
        try {
            if(year < 0 || month < 1 || month > 12 || day < 1 || day > 31) {
                throw new Exception();
            }
            if(month==2 && day>29) {
                throw new Exception();
            }
            if((month==4 || month==6 || month==9 || month==11) && day>30) {
                throw new Exception();
            }
        } catch (Exception e) {
            return "Illegal date input!";
        }
        // 儲存正在顯示的年月以做為後續更新日曆判別是否加上 event highlight 的參數
        setShowingYear(year);
        setShowingMonth(month);
        System.out.println("update showingYear: " + showingYear + ", showingMonth: " +
showingMonth);
        YearMonth yearMonth = YearMonth.of(year, month);
        LocalDate firstOfMonth = yearMonth.atDay(1);
        int daysInMonth = yearMonth.lengthOfMonth();
        MonthFirstDayWeek = firstOfMonth.getDayOfWeek().getValue() % 7; // 0 = Sunday,
1 = Monday, ..., 6 = Saturday

        // 取得今天的日期來決定 selected_day 的 class 加在哪一天
        int todayMonth = LocalDate.now().getMonthValue();
        int todayYear = LocalDate.now().getYear();

        StringBuilder calendarHtml = new StringBuilder();

        // 暫時在事前處理時僅先將當天的日期加上 selected_day 的 class
        int initDay = 1;
        for (int i = 0; i < 6; i++) { // for each week
            String htmlCode_event = "";
            for (int j = 0; j < 7; j++) { // for each day
                if (i == 0 && j < MonthFirstDayWeek || initDay > daysInMonth) {
                    calendarHtml.append("<div></div>");
                    htmlCode_event += "<div class='no_event'></div>";
                } else {
                    String htmlCode;
                    if(initDay == day && todayMonth == month && todayYear == year) {
                        htmlCode = "<div class='selected_day' id='d_" + initDay + "'
onclick='selectDay()'>";
                    } else {
                        htmlCode = "<div class='unselected_day' id='d_" + initDay + "'
onclick='selectDay()'>";
                    }
                    calendarHtml.append(htmlCode).append(initDay).append("</div>");

                    htmlCode_event += "<div class='no_event' id='e_" + initDay +
"'></div>";

                    initDay++;
                }
            }
            calendarHtml.append(htmlCode_event);

            if (initDay > daysInMonth) {
                break;
            }
```

```
        }

        return calendarHtml.toString();
    }

    int getMonthFirstDayWeek() {
        return MonthFirstDayWeek;
    }

    // 取得當年月的所有有 event 的日期
    public List<Integer> getMonthHighlightDays() {
        List<Integer> monthHighlightDays = new ArrayList<>();
        List<Integer> allMemoItemDay = memoList.getAllMemoItemParseDayFormatToInt();
        System.out.println("memoList.getAllMemoItemDay().size(): " +
allMemoItemDay.size());
        for (Integer i : allMemoItemDay) {
            int year = i / 10000;
            if (year == showingYear) {
                int month = (i % 10000) / 100;
                if (month == showingMonth) {
                    monthHighlightDays.add(i % 100);
                }
            }
        }
        System.out.println("showingYear: " + showingYear + ", showingMonth: " +
showingMonth);
        return monthHighlightDays;
    }

    public void setShowingYear(int showingYear) {
        this.showingYear = showingYear;
    }

    public void setShowingMonth(int showingMonth) {
        this.showingMonth = showingMonth;
    }
}
```

CalendarController.java

```
package com.example.memobackend;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
public class CalendarController {
    @Autowired
    private MemoList memoList;
    private Calendar calendar;

    public CalendarController(MemoList memoList) {
        this.memoList = memoList;
        this.calendar = memoList.getCalendar();
    }

    @GetMapping("/calendar")
    public String getCalendar(@RequestParam int year, @RequestParam int month,
@RequestParam int day) {
        return calendar.generateCalendarDays(year, month, day);
    }

    @GetMapping("refresh_highlight")
    public List<Integer> refresh_highlight() {
        List<Integer> eventDays = calendar.getMonthHighlightDays();
        System.out.println("eventDays: l=" + eventDays.size() + " ");
        for (Integer eventDay : eventDays) {
            System.out.print(eventDay+" ");
        }
```

```
        System.out.println();
        return eventDays;
    }
}
```

Label.java

```
package com.example.memobackend;

public class Label {
    private String name;

    public Label(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Label label = (Label) o;
        return name.equals(label.name);
    }

    @Override
    public int hashCode() {
        return name.hashCode();
    }
}
```

LabelController.java

```
package com.example.memobackend;

import org.springframework.web.bind.annotation.*;

import java.util.HashSet;
import java.util.Set;

@RestController
public class LabelController {

    Set<Label> labelList = new HashSet<>();

    @PostMapping(value = "/add_label")
    public void addLabel(@RequestParam("title") String name) {
        Label newLabel = new Label(name);
        if (!labelList.contains(newLabel)) {
            labelList.add(newLabel);
        }
    }

    @GetMapping(value = "/get_label")
    public Set<Label> getLabel() {
        return labelList;
    }


}
```

LoginWithGoogle.java

```
package com.example.memobackend;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.exc.InvalidDefinitionException;
```

```java
import
org.springframework.security.oauth2.client.authentication.OAuth2AuthenticationToken;
import org.springframework.security.oauth2.core.user.OAuth2User;
import org.springframework.web.bind.annotation.RestController;

import java.security.Principal;
import java.util.Map;

@RestController
public class LoginWithGoogle {

    // 取得登入的使用者資訊
    // 進入 http://localhost:8080/user 可以看到回傳的使用者資訊
    public Principal user(Principal principal) throws JsonProcessingException,
InvalidDefinitionException {
        System.out.println("principal: ");
        System.out.println(principal.toString());
        return principal;
    }

//    要取得"想像的"使用者名稱要使用 OAuth2AuthenticationToken
//    也就是下面那個 function
//    @GetMapping("/username")
//    public String getUsername(Principal principal) {
//        return principal.getName();
//    }

    // 取得登入的使用者 Google 資訊
    public Map<String, Object> userInfo(Principal principal) {
        OAuth2AuthenticationToken token = (OAuth2AuthenticationToken) principal;
        OAuth2User userInfo = token.getPrincipal();

        // set user info
        User user = new User(userInfo.getAttribute("sub"),
userInfo.getAttribute("given_name"), userInfo.getAttribute("email"));
        System.out.print("id: ");
        System.out.println(user.getId());
        System.out.print("name: ");
        System.out.println(user.getName());
        System.out.print("email: ");
        System.out.println(user.getEmail());

        return userInfo.getAttributes();
    }
}
```

MemoController.java

```java
package com.example.memobackend;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.*;

@RestController
public class MemoController {
    @Autowired
    private MemoList memoList = new MemoList();

    // 從 MemoList 中獲取 MemoItem
    // 之後會看要不要新增 ID 搜尋，透過 ID 尋找指定的 MemoList
    @RequestMapping(value = "/get_memo_items")
    public Set<MemoItem> getMemoItemSetOfMemoList() {
        return memoList.getMemoItems();
    }

    @RequestMapping(value = "/get_memo_item/{id}")
```

```
    public MemoItem getMemoItemOfMemoList(@PathVariable Long id) {
        return memoList.getMemoItemById(id);
    }

    @PostMapping(value = "/add_memo_item")
    public void addMemo(@RequestParam("title") String title,
                        @RequestParam("year") String year,
                        @RequestParam("month") String month,
                        @RequestParam("day") String day,
                        @RequestParam("hour") String hour,
                        @RequestParam("minute") String minute,
                        @RequestParam("alertTimeSelection") String alertTimeSelection,
                        @RequestParam("description") String description) {
        String time = year + '/' + month + '/' + day + ' ' + hour + ':' + minute;
        Long id = memoList.generateId();
        memoList.addMemoItem(new MemoItem(title, time,
                alertTimeSelection, description, id));
    }

    // 删除指定的 MemoItem
    @DeleteMapping(value = "/delete_memo_item/{id}")
    public void deleteMemoItem(@PathVariable String id) {
        Long memoItemId = Long.parseLong(id);
        memoList.removeMemoItemById(memoItemId);
    }

    @PutMapping(value = "/edit_memo_item/{id}")
    public void editMemoItem(@PathVariable Long id,
                             @RequestParam("title") String title,
                             @RequestParam("year") String year,
                             @RequestParam("month") String month,
                             @RequestParam("day") String day,
                             @RequestParam("hour") String hour,
                             @RequestParam("minute") String minute,
                             @RequestParam("alertTimeSelection") String
alertTimeSelection,
                             @RequestParam("description") String description) {
        String time = year + '/' + month + '/' + day + ' ' + hour + ':' + minute;
        memoList.editContentOfMemoItemById(title, time, alertTimeSelection,
description, id);
    }

    @PostMapping("/click_label")
    public void handleLabelClick(@RequestBody Map<String, Object> payload) {
        long GetId = ((Number) payload.get("id")).longValue();
        List<String> labels = (List<String>) payload.get("labels");

        System.out.println("Received ID: " + GetId);
        System.out.println("Received labels: " + labels);

        memoList.addLabelOnMemoItem(GetId, labels);

        //return new ResponseEntity<>("Labels received", HttpStatus.OK);
    }

    @GetMapping("/get_memo_items_by_label")
    public ResponseEntity<Set<MemoItem>> getMemoItemsByLabel(@RequestParam String
label) {
        Set<MemoItem> memoItems = new HashSet<>();
        Set<MemoItem> oldMemoItems = memoList.getMemoItems();
        for (MemoItem memoItem : oldMemoItems) {
            for (Label memoLabel : memoItem.getLabelSet()) {
                if (Objects.equals(memoLabel.getName(), label)) {
                    memoItems.add(memoItem);
                }
            }
        }
        return new ResponseEntity<>(memoItems, HttpStatus.OK);
    }
}
```

MemoItem.java

```
package com.example.memobackend;

import java.util.HashSet;
import java.util.Objects;
import java.util.Set;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class MemoItem {
    String title;

    String time; // time 和 alert_time 的格式為 yyyy/MM/dd hh:mm

    String alertTimeSelection;

    String alert time;

    String description;

    final Long id;

    Set<Label> labelList = new HashSet<>();

    public MemoItem(String title, String time, String alertTimeSelection, String
description, Long id) {
        this.title = title;
        this.time = time;
        this.alertTimeSelection = alertTimeSelection;
        this.alert_time = selectAlertTime(time, alertTimeSelection);
        this.description = description;
        this.id = id;
    }

    public String selectAlertTime(String time, String alertTimeSelection) {
        if (!Objects.equals(alertTimeSelection, "")) {
            // 將 time 的內容分離出年、月、日、時、分 5 個元素
            String regex = "(\\d{4})/(\\d{2})/(\\d{2}) (\\d{2}):(\\d{2})";
            Pattern pattern = Pattern.compile(regex);
            Matcher matcher = pattern.matcher(time);
            int year = 0, month = 0, day = 0, hour = 0, minute = 0;
            if (matcher.matches()) { // 如果沒有 matcher.matches()，將無法取得對應元素
                year = Integer.parseInt(matcher.group(1));
                month = Integer.parseInt(matcher.group(2));
                day = Integer.parseInt(matcher.group(3));
                hour = Integer.parseInt(matcher.group(4));
                minute = Integer.parseInt(matcher.group(5));
            } else {
                System.out.println("未找到匹配的日期時間格式！");
                return "";
            }
            // 將分離出的 5 個元素放進 LocalDateTime 裡，方便做時間的加減
            LocalDateTime standardTime = LocalDateTime.of(year, month, day, hour,
minute);
            // 透過選擇出的 alertTimeSelection，對 time 做加減成 alert_time，接著將 alert_time
轉 String 後回傳
            DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy/MM/dd
HH:mm");
            return timeIsAddedOrSubtractedToAlertTime(standardTime,
alertTimeSelection).format(formatter);
        } else {
            return "";
        }
    }

    public LocalDateTime timeIsAddedOrSubtractedToAlertTime(LocalDateTime standardTime,
```

```
String alertTimeSelection) {
        return switch (alertTimeSelection) {
            case "5 minutes ago" -> standardTime.minusMinutes(5);
            case "10 minutes ago" -> standardTime.minusMinutes(10);
            case "15 minutes ago" -> standardTime.minusMinutes(15);
            case "30 minutes ago" -> standardTime.minusMinutes(30);
            case "1 hour ago" -> standardTime.minusHours(1);
            case "2 hours ago" -> standardTime.minusHours(2);
//            case "1 day ago" -> standardTime.minusDays(1);
//            case "1 days ago" -> standardTime.minusDays(2);
//            case "1 week ago" -> standardTime.minusDays(7);
            default -> standardTime;
        };
    }

    public void editContent(String title, String time, String alertTimeSelection,
String description) {
        this.title = title;
        this.time = time;
        this.alertTimeSelection = alertTimeSelection;
        this.alert_time = selectAlertTime(time, alertTimeSelection);
        this.description = description;
    }

    public String getTitle() {
        return this.title;
    }

    public String getTime() {
        return this.time;
    }

    public String getAlertTimeSelection() {
        return this.alertTimeSelection;
    }

    public String getAlertTime() {
        return this.alert_time;
    }

    public String getDescription() {
        return this.description;
    }

    public Long getId() {
        return this.id;
    }

    public void addLabel(Label label) {
        labelList.add(label);
    }
    public Set<Label> getLabelSet() {
        return this.labelList;
    }

    @Override
    public boolean equals(Object obj) { // 由於 MemoItem 內容會有被改變的可能，因此 MemoItem 間
的比較以 id 為主
        if (this == obj) {
            return true;
        }
        if (obj == null || getClass() != obj.getClass()) {
            return false;
        }
        MemoItem other = (MemoItem) obj;
        return id.equals(other.getId());
    }

    @Override
    public int hashCode() {
        return Objects.hash(id);
```

```
    }

    // return format is yyyyMMdd
    public int parseDayFormatToInt(){
        String time = this.time;
        String regex = "(\\d{4})/(\\d{2})/(\\d{2}) (\\d{2}):(\\d{2})";
        Pattern pattern = Pattern.compile(regex);
        Matcher matcher = pattern.matcher(time);
        int year = 0, month = 0, day = 0;
        try {
            if (matcher.matches()) {
                year = Integer.parseInt(matcher.group(1));
                month = Integer.parseInt(matcher.group(2));
                day = Integer.parseInt(matcher.group(3));
            } else {
                throw new IllegalArgumentException("未找到匹配的日期時間格式！");
            }
        } catch (IllegalArgumentException e) {
            e.printStackTrace();
        }
        day = year* 10000 + month * 100 + day;
        return day;
    }


}
```

MemoList.java

```
package com.example.memobackend;

import org.springframework.stereotype.Component;

import java.util.*;

@Component
public class MemoList {
    private Set<MemoItem> memoItems;
    private Calendar calendar;

    public MemoList() {
        memoItems = new HashSet<>();
        calendar = new Calendar(this);
    }

    // 添加 memoItem 到集合
    public void addMemoItem(MemoItem memoItem) {
        memoItems.add(memoItem);
    }

    // 查找集合中是否包含某個 memoItem
    public boolean containsMemoItem(MemoItem memoItem) {
        return memoItems.contains(memoItem);
    }

    // 刪除集合中的某個 memoItem
    public void removeMemoItemById(Long id) {
        MemoItem memoItem = getMemoItemById(id);
        memoItems.remove(memoItem);
    }

    public void addLabelOnMemoItem(long id, List<String> labels) {
        MemoItem memoItem = findMemoItemById(id);
        for(String label : labels) {
            memoItem.addLabel(new Label(label));
        }
    }

    public MemoItem findMemoItemById(Long id) {
        for (MemoItem memoItem : memoItems) {
```

```
            if (memoItem.getId().equals(id)) {
                return memoItem;
            }
        }
        return null;
    }

    public Long generateId() {
        Long id;
        // 生成隨機數字
        Random random = new Random();
        do {
            id = random.nextLong() % 1000000; // 產生隨機數
        } while (id <= 0 || containsId(id)); // 檢查是否為 0 或負數，以及是否已經存在

        return id;
    }

    // 檢查集合中是否存在具有特定 id 的 MemoItem
    public boolean containsId(Long id) {
        for (MemoItem memoItem : memoItems) {
            if (memoItem.getId().equals(id)) {
                return true;
            }
        }
        return false;
    }

    public void editContentOfMemoItemById(String title, String time, String
alertTimeSelection,
                                          String description, Long id) {
        MemoItem memoItem = getMemoItemById(id);
        memoItem.editContent(title, time, alertTimeSelection, description);
    }

    public Set<MemoItem> getMemoItems() {
        return this.memoItems;
    }

    // 在集合中尋找具有特定 id 的 MemoItem
    public MemoItem getMemoItemById(Long id) {
        for (MemoItem memoItem : memoItems) {
            if (memoItem.getId().equals(id)) {
                return memoItem;
            }
        }
        return null;
    }

    // 取得所有的日期
    public List<Integer> getAllMemoItemParseDayFormatToInt(){
        List<Integer> days = new ArrayList<>();
        System.out.println("memoItems.size(): " + memoItems.size());
        for (MemoItem memoItem : memoItems) {
            days.add(memoItem.parseDayFormatToInt());
        }
        return days;
    }

    public Calendar getCalendar() {
        return calendar;
    }
}
```

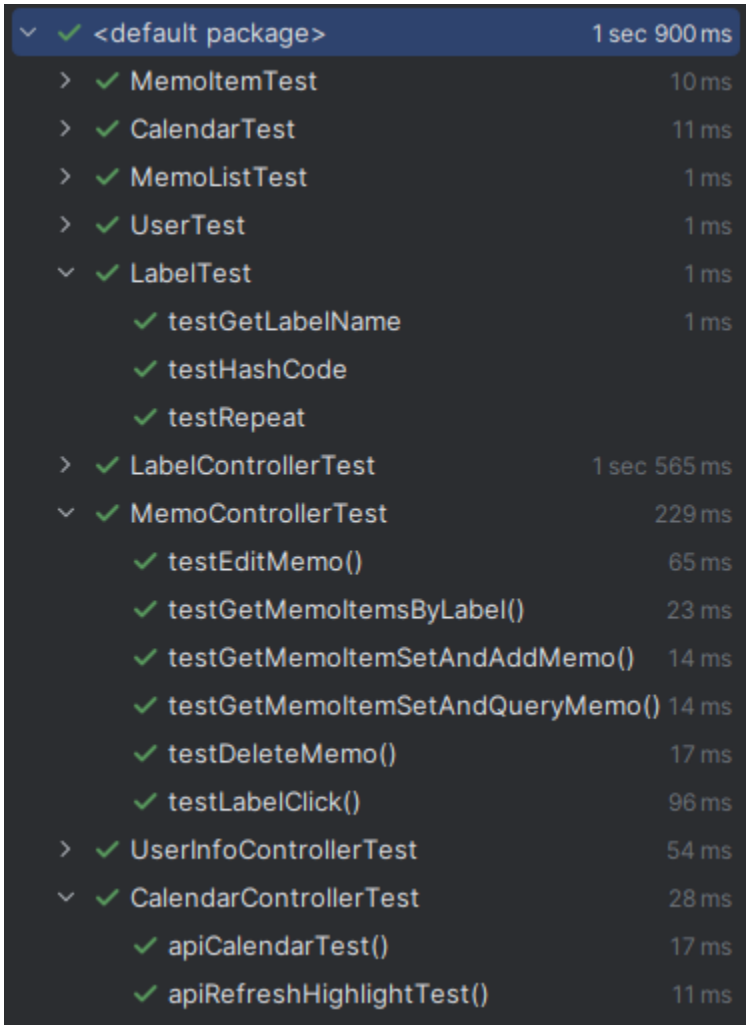
User.java

```
package com.example.memobackend;

public class User {
    final private String id;
```

```
    final private String account;
    final private String name;

    public User(String id, String name, String account) {
        this.id = id;
        this.name = name;
        this.account = account;
    }

    public String getId() {
        return id;
    }
    public String getName() {
        return name;
    }

    public String getEmail() {
        return account;
    }
}
```

第六章 Unit Testing

6.1 Snapshots of testing result



6.2 Unit Testing Code Listing

CalendarControllerTest.java

```
package com.example.memobackend;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.security.test.context.support.WithMockUser;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;
import org.springframework.test.web.servlet.result.MockMvcResultMatchers;

@SpringBootTest
@AutoConfigureMockMvc
class CalendarControllerTest {
    @Autowired
    private MockMvc mockMvc;

    @Autowired
    private MemoList memoList;
```

```
    @Test
    @WithMockUser
    public void apiCalendarTest() throws Exception {
        int year = 2024;
        int month = 4;
        int day = 25;
        String api = "/calendar?" + "year=" + year + "&month=" + month + "&day=" + day;
        mockMvc.perform(MockMvcRequestBuilders.get(api))
                .andExpect(MockMvcResultMatchers.status().isOk());
    }

    @Test
    @WithMockUser
    public void apiRefreshHighlightTest() throws Exception {
        memoList.addMemoItem(new MemoItem("title", "2000/10/05 12:00", "5 minutes ago",
"description", 1L));
        memoList.addMemoItem(new MemoItem("title", "2000/10/10 12:00", "5 minutes ago",
"description", 2L));
        memoList.addMemoItem(new MemoItem("title", "2000/10/15 12:00", "5 minutes ago",
"description", 3L));
        memoList.addMemoItem(new MemoItem("title", "2000/07/15 12:00", "5 minutes ago",
"description", 4L));
        memoList.addMemoItem(new MemoItem("title", "2000/05/15 12:00", "5 minutes ago",
"description", 5L));

        Calendar calendar = memoList.getCalendar();
        calendar.setShowingYear(2000);
        calendar.setShowingMonth(10);

        mockMvc.perform(MockMvcRequestBuilders.get("/refresh_highlight"))
                .andExpect(MockMvcResultMatchers.status().isOk())
                .andExpect(MockMvcResultMatchers.content().json("[5, 10, 15]"));
    }
}
```

CalendarTest.java

```
package com.example.memobackend;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import java.util.List;

import static org.junit.Assert.*;

public class CalendarTest {
    @Before
    public void setUp() throws Exception {
    }

    @After
    public void tearDown() throws Exception {
    }

    @Test
    public void generateCalendarDays() {
        Calendar calendar = new Calendar();
        String result = calendar.generateCalendarDays(2024, 4, 25);
        assertNotNull(result);
        assertEquals(1, calendar.getMonthFirstDayWeek());

        result = calendar.generateCalendarDays(2024, 5, 25);
        assertNotNull(result);
        assertEquals(3, calendar.getMonthFirstDayWeek());

        result = calendar.generateCalendarDays(2024, 3, 25);
        assertNotNull(result);
        assertEquals(5, calendar.getMonthFirstDayWeek());
```

```
    }

    @Test
    public void generateCalendarDaysIllegalDate() {
        Calendar calendar = new Calendar();
        String result = calendar.generateCalendarDays(2024, 4, 31);
        assertEquals("Illegal date input!", result);

        result = calendar.generateCalendarDays(2024, 2, 30);
        assertEquals("Illegal date input!", result);

        result = calendar.generateCalendarDays(2024, -1, 31);
        assertEquals("Illegal date input!", result);
    }

    @Test
    public void getMonthHighlightDays(){
        MemoList memoList = new MemoList();
        memoList.addMemoItem(new MemoItem("title", "2024/06/05 12:00", "5 minutes ago",
"description", 1L));
        memoList.addMemoItem(new MemoItem("title", "2024/06/10 12:00", "5 minutes ago",
"description", 2L));
        memoList.addMemoItem(new MemoItem("title", "2024/06/15 12:00", "5 minutes ago",
"description", 3L));
        memoList.addMemoItem(new MemoItem("title", "2024/07/15 12:00", "5 minutes ago",
"description", 4L));
        memoList.addMemoItem(new MemoItem("title", "2024/05/15 12:00", "5 minutes ago",
"description", 5L));

        Calendar calendar = memoList.getCalendar();
        // 設定正在顯示的月份
//        calendar.generateCalendarDays(2024, 6, 1);
        calendar.setShowingYear(2024);
        calendar.setShowingMonth(6);
        List<Integer> result = calendar.getMonthHighlightDays();
        assertEquals(3, result.size());
        assertEquals(5, result.get(0).intValue());
        assertEquals(10, result.get(1).intValue());
        assertEquals(15, result.get(2).intValue());
    }

}
```

LabelControllerTest.java

```
package com.example.memobackend;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;
import org.springframework.context.annotation.Import;
import org.springframework.security.test.context.support.WithMockUser;
import org.springframework.test.context.junit4.SpringRunner;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;
import org.springframework.test.web.servlet.result.MockMvcResultMatchers;

@RunWith(SpringRunner.class)
@WebMvcTest(controllers = LabelController.class)
@Import(SecurityConfig.class)
public class LabelControllerTest {
    @Autowired
    private MockMvc mockMvc;

    @Test
    @WithMockUser
    public void testGetLabelSetAndAddLabel() throws Exception {
        mockMvc.perform(MockMvcRequestBuilders.post("/add_label")
                        .param("title", "aaa"))
```

```
                .andExpect(MockMvcResultMatchers.status().isOk());
        mockMvc.perform(MockMvcRequestBuilders.get("/get_label"))
                .andExpect(MockMvcResultMatchers.status().isOk())
                .andExpect(MockMvcResultMatchers.content().json("[{\"name\":\"aaa\"}]")
);
    }
}
```

LabelTest.java

```
package com.example.memobackend;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.*;

public class LabelTest {
    @Before
    public void setUp() throws Exception {
    }

    @After
    public void tearDown() throws Exception {
    }

    @Test
    public void testGetLabelName() throws Exception {
        String LabelName = "aaa";
        Label label = new Label(LabelName);
        assertEquals(LabelName, label.getName());
    }

    @Test
    public void testRepeat() throws Exception {
        String LabelName = "aaa";
        String anotherLabelName = "bbb";
        String LabelNameRepeat = "aaa";
        Label label = new Label(LabelName);
        Label anotherLabel = new Label(anotherLabelName);
        Label labelRepeat = new Label(LabelNameRepeat);
        assertFalse(label.equals(anotherLabel));
        assertTrue(label.equals(labelRepeat));
    }

    @Test
    public void testHashCode() throws Exception {
        String LabelName = "aaa";
        Label label = new Label(LabelName);
        assertEquals(96321, label.hashCode());
    }
}
```

LoginTest.java

```
package com.example.memobackend;

import org.junit.jupiter.api.Test;
import org.mockito.Mock;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.security.core.context.SecurityContext;
import org.springframework.security.core.context.SecurityContextHolder;
import
org.springframework.security.oauth2.client.authentication.OAuth2AuthenticationToken;
import org.springframework.security.oauth2.core.user.DefaultOAuth2User;
import org.springframework.security.test.context.support.WithMockUser;
```

```java
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;
import org.springframework.test.web.servlet.result.MockMvcResultMatchers;

import java.security.Principal;
import java.util.HashMap;
import java.util.Map;

import static org.mockito.Mockito.when;

@SpringBootTest
@AutoConfigureMockMvc
class UserInfoControllerTest {
    @Autowired
    private MockMvc mockMvc;

    @MockBean
    private OAuth2AuthenticationToken token;

    @Mock
    private Principal principal;

    @Mock
    private DefaultOAuth2User userInfo;

    @Test
    @WithMockUser
    public void apiUserInfoTest() throws Exception {
        Map<String, Object> attributes = new HashMap<>();
        attributes.put("sub", "test_id");
        attributes.put("given_name", "Test User");
        attributes.put("email", "test@example.com");

        DefaultOAuth2User userInfo = new DefaultOAuth2User(null, attributes, "sub");

        when(token.getPrincipal()).thenReturn(userInfo);

        SecurityContext securityContext = SecurityContextHolder.getContext();
        securityContext.setAuthentication(token);

        //先當作錯誤的登入方式會產生錯誤...?   * isForbidden=403
        mockMvc.perform(MockMvcRequestBuilders.get("/user_info")
                        .principal(principal))
                .andExpect(MockMvcResultMatchers.status().isForbidden());
    }

}
```

MemoItemTest.java

```java
package com.example.memobackend;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import java.util.Optional;

import static org.junit.Assert.*;

public class MemoItemTest {

    @Before
    public void setUp() throws Exception {
    }

    @After
    public void tearDown() throws Exception {
    }
```

```
    @Test
    public void testSelectAlertTimeNormal() throws Exception {
        String title = "aaa";
        String time = "2024/04/29 15:00";
        String alertTimeSelection1 = "Time of Memo";
        String alertTimeSelection2 = "5 minutes ago";
        String alertTimeSelection3 = "10 minutes ago";
        String alertTimeSelection4 = "15 minutes ago";
        String alertTimeSelection5 = "30 minutes ago";
        String alertTimeSelection6 = "1 hour ago";
        String alertTimeSelection7 = "2 hours ago";
        MemoItem memoItem1 = new MemoItem(title, time, alertTimeSelection1, "", 0L);
        MemoItem memoItem2 = new MemoItem(title, time, alertTimeSelection2, "", 0L);
        MemoItem memoItem3 = new MemoItem(title, time, alertTimeSelection3, "", 0L);
        MemoItem memoItem4 = new MemoItem(title, time, alertTimeSelection4, "", 0L);
        MemoItem memoItem5 = new MemoItem(title, time, alertTimeSelection5, "", 0L);
        MemoItem memoItem6 = new MemoItem(title, time, alertTimeSelection6, "", 0L);
        MemoItem memoItem7 = new MemoItem(title, time, alertTimeSelection7, "", 0L);
        assertEquals("2024/04/29 15:00", memoItem1.getAlertTime());
        assertEquals("2024/04/29 14:55", memoItem2.getAlertTime());
        assertEquals("2024/04/29 14:50", memoItem3.getAlertTime());
        assertEquals("2024/04/29 14:45", memoItem4.getAlertTime());
        assertEquals("2024/04/29 14:30", memoItem5.getAlertTime());
        assertEquals("2024/04/29 14:00", memoItem6.getAlertTime());
        assertEquals("2024/04/29 13:00", memoItem7.getAlertTime());
    }

    @Test
    public void testSelectAlertTimeTimeFormatIsWrong() throws Exception {
        String title = "aaa";
        String time = "2024/4/29 15:0";
        String alertTimeSelection = "Time of Memo";
        MemoItem memoItem = new MemoItem(title, time, alertTimeSelection, "", 0L);
        assertEquals("", memoItem.getAlertTime());
    }

    @Test
    public void testSelectAlertTimeAlertTimeSelectionIsEmpty() throws Exception {
        String title = "aaa";
        String time = "2024/04/29 15:00";
        String alertTimeSelection = "";
        String description = "123456";
        MemoItem memoItem = new MemoItem(title, time, alertTimeSelection, description,
0L);
        assertEquals("aaa", memoItem.getTitle());
        assertEquals("2024/04/29 15:00", memoItem.getTime());
        assertEquals("", memoItem.getAlertTimeSelection());
        assertEquals("", memoItem.getAlertTime());
        assertEquals("123456", memoItem.getDescription());
        assertEquals(Optional.of(0L), Optional.of(memoItem.getId()));
    }

    @Test
    public void parseDayFormatToInt() throws Exception {
        String title = "aaa";
        String time = "2024/06/05 15:00";
        String alertTimeSelection = "Time of Memo";
        String description = "123456";
        MemoItem memoItem = new MemoItem(title, time, alertTimeSelection, description,
0L);
        assertEquals(20240605, memoItem.parseDayFormatToInt());

    }

    @Test
    public void checkMemoItemNotRepeat() throws Exception {
        String title1 = "aaa";
        String time1 = "2024/04/29 15:00";
        String alertTimeSelection1 = "";
        String description1 = "123456";
        MemoItem memoItem1 = new MemoItem(title1, time1, alertTimeSelection1,
```

```
description1, 0L);
        assertTrue(memoItem1.equals(memoItem1));
        String title2 = "aaa";
        String time2 = "2024/04/29 15:00";
        String alertTimeSelection2 = "";
        String description2 = "123456";
        MemoItem memoItem2 = new MemoItem(title2, time2, alertTimeSelection2,
description2, 0L);
        assertTrue(memoItem1.equals(memoItem2));
        MemoItem memoItem3 = null;
        assertFalse(memoItem1.equals(memoItem3));
    }
}
```

MemoListTest.java

```
package com.example.memobackend;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
import java.util.Set;

import static org.junit.Assert.*;

public class MemoListTest {
    String title = "aaa";
    String time = "2024/04/29 15:00";
    String alertTimeSelection = "Time of Memo";
    String description = "123456";
    Long id = 0L;
    MemoItem memoItem = new MemoItem(title, time, alertTimeSelection, description, id);
    MemoList memoList = new MemoList();

    @Before
    public void setUp() throws Exception {
    }

    @After
    public void tearDown() throws Exception {
    }

    @Test
    public void testAddMemoItem() throws Exception {
        memoList.addMemoItem(memoItem);
        assertTrue(memoList.containsMemoItem(memoItem));
    }

    @Test
    public void testRemoveMemoItem() throws Exception {
        memoList.addMemoItem(memoItem);
        memoList.removeMemoItemById(memoItem.getId());
        assertFalse(memoList.containsMemoItem(memoItem));
    }

    @Test
    public void testMemoItemNotFound() throws Exception {
        memoList.addMemoItem(memoItem);
        assertNull(memoList.getMemoItemById(111111L));
    }

    @Test
    public void testGetMemoItems() throws Exception {
        memoList.addMemoItem(memoItem);
        String title = "bbb";
        String time = "2024/04/30 12:00";
        String alertTimeSelection = "1 hour ago";
        String description = "654321";
```

```
        Long id = 1L;
        MemoItem memoItem2 = new MemoItem(title, time, alertTimeSelection, description,
id);
        memoList.addMemoItem(memoItem2);
        Set<MemoItem> resultMemoItemSet = memoList.getMemoItems();

        for (MemoItem expectedMemoItem : Arrays.asList(memoItem, memoItem2)) {
            boolean isMatched = false;
            for (MemoItem resultMemoItem : resultMemoItemSet) {
                if (expectedMemoItem.getTitle().equals(resultMemoItem.getTitle())
                        && expectedMemoItem.getTime().equals(resultMemoItem.getTime())
                        &&
expectedMemoItem.getAlertTimeSelection().equals(resultMemoItem.getAlertTimeSelection())
                        &&
expectedMemoItem.getDescription().equals(resultMemoItem.getDescription())) {
                    isMatched = true;
                    break;
                }
            }
            assertTrue(isMatched);
        }
    }

    @Test
    public void getAllMemoItemParseDayFormatToInt(){
        memoList.addMemoItem(memoItem);

        String title = "bbb";
        String time = "2024/06/05 12:00";
        String alertTimeSelection = "1 hour ago";
        String description = "654321";
        Long id = 1L;
        MemoItem memoItem2 = new MemoItem(title, time, alertTimeSelection, description,
id);
        memoList.addMemoItem(memoItem2);

        List<Integer> result = memoList.getAllMemoItemParseDayFormatToInt();
        assertEquals(20240605, result.get(0).intValue());
        assertEquals(20240429, result.get(1).intValue());
        assertEquals(2, result.size());
    }
}
```

MemoControllerTest.java

```
package com.example.memobackend;

import org.json.JSONArray;
import org.json.JSONObject;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.http.MediaType;
import org.springframework.security.test.context.support.WithMockUser;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.MvcResult;
import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;
import org.springframework.test.web.servlet.result.MockMvcResultMatchers;

import java.util.List;
import java.util.Set;

import static org.junit.Assert.*;

@SpringBootTest
@AutoConfigureMockMvc
class MemoControllerTest {
    @Autowired
    private MockMvc mockMvc;
```

```
    @Autowired
    private MemoList memoList;

    @BeforeEach
    public void setup() {
        memoList.getMemoItems().clear(); // 在每个测试前清空 MemoList
    }

    @Test
    @WithMockUser
    public void testGetMemoItemSetAndAddMemo() throws Exception {
        mockMvc.perform(MockMvcRequestBuilders.post("/add_memo_item")
                        .param("title", "aaa")
                        .param("year", "2024")
                        .param("month", "04")
                        .param("day", "29")
                        .param("hour", "15")
                        .param("minute", "00")
                        .param("alertTimeSelection", "Time of Memo")
                        .param("description", "654321"))
                .andExpect(MockMvcResultMatchers.status().isOk());
        mockMvc.perform(MockMvcRequestBuilders.post("/add_memo_item")
                        .param("title", "bbb")
                        .param("year", "2024")
                        .param("month", "06")
                        .param("day", "06")
                        .param("hour", "12")
                        .param("minute", "00")
                        .param("alertTimeSelection", "1 hour ago")
                        .param("description", "123456"))
                .andExpect(MockMvcResultMatchers.status().isOk());
        mockMvc.perform(MockMvcRequestBuilders.get("/get_memo_items"))
                .andExpect(MockMvcResultMatchers.status().isOk())
                .andExpect(MockMvcResultMatchers.content().json("[{\"title\":\"aaa\",\"
time\":\"2024/04/29 15:00\",\"alertTimeSelection\":\"Time of
Memo\",\"alertTime\":\"2024/04/29 15:00\",\"description\":\"654321\"}," +
                        "{\"title\":\"bbb\",\"time\":\"2024/06/06
12:00\",\"alertTimeSelection\":\"1 hour ago\",\"alertTime\":\"2024/06/06
11:00\",\"description\":\"123456\"}]"));
    }

    @Test
    @WithMockUser
    public void testEditMemo() throws Exception {
        mockMvc.perform(MockMvcRequestBuilders.post("/add_memo_item")
                        .param("title", "bbb")
                        .param("year", "2024")
                        .param("month", "06")
                        .param("day", "06")
                        .param("hour", "12")
                        .param("minute", "00")
                        .param("alertTimeSelection", "1 hour ago")
                        .param("description", "123456"))
                .andExpect(MockMvcResultMatchers.status().isOk());
        MvcResult getResult =
mockMvc.perform(MockMvcRequestBuilders.get("/get_memo_items"))
                .andExpect(MockMvcResultMatchers.status().isOk())
                .andReturn();

        String jsonResponse = getResult.getResponse().getContentAsString();
        JSONArray memoItemsArray = new JSONArray(jsonResponse);
        JSONObject memoItemObject = memoItemsArray.getJSONObject(0);
        Long id = memoItemObject.getLong("id");

        mockMvc.perform(MockMvcRequestBuilders.put("/edit_memo_item/" + id)
                        .param("title", "qqq")
                        .param("year", "2024")
                        .param("month", "04")
                        .param("day", "29")
                        .param("hour", "15")
```

```
                        .param("minute", "00")
                        .param("alertTimeSelection", "Time of Memo")
                        .param("description", "654321"))
                .andExpect(MockMvcResultMatchers.status().isOk());
        mockMvc.perform(MockMvcRequestBuilders.get("/get_memo_items"))
                .andExpect(MockMvcResultMatchers.status().isOk())
                .andExpect(MockMvcResultMatchers.content().json("[{\"title\":\"qqq\",\"
time\":\"2024/04/29 15:00\",\"alertTimeSelection\":\"Time of
Memo\",\"alertTime\":\"2024/04/29 15:00\",\"description\":\"654321\"}]"));
    }

    @Test
    @WithMockUser
    public void testDeleteMemo() throws Exception {
        mockMvc.perform(MockMvcRequestBuilders.post("/add_memo_item")
                        .param("title", "bbb")
                        .param("year", "2024")
                        .param("month", "06")
                        .param("day", "06")
                        .param("hour", "12")
                        .param("minute", "00")
                        .param("alertTimeSelection", "1 hour ago")
                        .param("description", "123456"))
                .andExpect(MockMvcResultMatchers.status().isOk());
        MvcResult getResult =
mockMvc.perform(MockMvcRequestBuilders.get("/get_memo_items"))
                .andExpect(MockMvcResultMatchers.status().isOk())
                .andReturn();

        String jsonResponse = getResult.getResponse().getContentAsString();
        JSONArray memoItemsArray = new JSONArray(jsonResponse);
        JSONObject memoItemObject = memoItemsArray.getJSONObject(0);
        Long id = memoItemObject.getLong("id");

        mockMvc.perform(MockMvcRequestBuilders.delete("/delete_memo_item/" + id))
                .andExpect(MockMvcResultMatchers.status().isOk());

        mockMvc.perform(MockMvcRequestBuilders.get("/get_memo_items"))
                .andExpect(MockMvcResultMatchers.status().isOk())
                .andExpect(MockMvcResultMatchers.content().json("[]"));
    }

    @Test
    @WithMockUser
    public void testAddMemoItems() throws Exception {
        mockMvc.perform(MockMvcRequestBuilders.post("/add_memo_item")
                        .param("title", "aaa")
                        .param("year", "2024")
                        .param("month", "04")
                        .param("day", "29")
                        .param("hour", "15")
                        .param("minute", "00")
                        .param("alertTimeSelection", "Time of Memo")
                        .param("description", "654321"))
                .andExpect(MockMvcResultMatchers.status().isOk());
        mockMvc.perform(MockMvcRequestBuilders.post("/add_memo_item")
                        .param("title", "bbb")
                        .param("year", "2024")
                        .param("month", "06")
                        .param("day", "06")
                        .param("hour", "12")
                        .param("minute", "00")
                        .param("alertTimeSelection", "1 hour ago")
                        .param("description", "123456"))
                .andExpect(MockMvcResultMatchers.status().isOk());
    }

    @Test
    @WithMockUser
    public void testQueryMemoItem() throws Exception {
        // 先添加一個 memo item
```

```
        mockMvc.perform(MockMvcRequestBuilders.post("/add_memo_item")
                        .param("title", "aaa")
                        .param("year", "2024")
                        .param("month", "04")
                        .param("day", "29")
                        .param("hour", "15")
                        .param("minute", "00")
                        .param("alertTimeSelection", "Time of Memo")
                        .param("description", "654321"))
                .andExpect(MockMvcResultMatchers.status().isOk());

        // 獲取添加的 memo item 的 id
        MvcResult getResult =
mockMvc.perform(MockMvcRequestBuilders.get("/get_memo_items"))
                .andExpect(MockMvcResultMatchers.status().isOk())
                .andReturn();
        String jsonResponse = getResult.getResponse().getContentAsString();
        JSONArray memoItemsArray = new JSONArray(jsonResponse);
        JSONObject memoItemObject = memoItemsArray.getJSONObject(0);
        Long id = memoItemObject.getLong("id");

        // 使用獲取的 id 來查詢 memo item
        mockMvc.perform(MockMvcRequestBuilders.get("/get_memo_item/" + id))
                .andExpect(MockMvcResultMatchers.status().isOk())
                .andExpect(MockMvcResultMatchers.content().json("{\"title\":\"aaa\",\"t
ime\":\"2024/04/29 15:00\",\"alertTimeSelection\":\"Time of
Memo\",\"alertTime\":\"2024/04/29 15:00\",\"description\":\"654321\"}"));
    }

    @Test
    @WithMockUser
    public void testLabelClick() throws Exception {
        mockMvc.perform(MockMvcRequestBuilders.post("/add_memo_item")
                        .param("title", "aaa")
                        .param("year", "2024")
                        .param("month", "04")
                        .param("day", "29")
                        .param("hour", "15")
                        .param("minute", "00")
                        .param("alertTimeSelection", "Time of Memo")
                        .param("description", "654321"))
                .andExpect(MockMvcResultMatchers.status().isOk());
        MvcResult getResult =
mockMvc.perform(MockMvcRequestBuilders.get("/get_memo_items"))
                .andExpect(MockMvcResultMatchers.status().isOk())
                .andReturn();

        String jsonResponse = getResult.getResponse().getContentAsString();
        JSONArray memoItemsArray = new JSONArray(jsonResponse);
        JSONObject memoItemObject = memoItemsArray.getJSONObject(0);
        Long id = memoItemObject.getLong("id");

        JSONObject payload = new JSONObject();
        payload.put("id", id);
        payload.put("labels", new JSONArray(List.of("label1", "label2")));


        mockMvc.perform(MockMvcRequestBuilders.post("/click label")
                        .content(payload.toString())
                        .contentType(MediaType.APPLICATION_JSON))
                .andExpect(MockMvcResultMatchers.status().isOk());
    }

    @Test
    @WithMockUser
    public void testGetMemoItemsByLabel() throws Exception {
        mockMvc.perform(MockMvcRequestBuilders.post("/add_memo_item")
                        .param("title", "aaa")
                        .param("year", "2024")
                        .param("month", "04")
                        .param("day", "29")
```

```
                        .param("hour", "15")
                        .param("minute", "00")
                        .param("alertTimeSelection", "Time of Memo")
                        .param("description", "654321"))
                .andExpect(MockMvcResultMatchers.status().isOk());
        mockMvc.perform(MockMvcRequestBuilders.post("/add_memo_item")
                        .param("title", "bbb")
                        .param("year", "2024")
                        .param("month", "06")
                        .param("day", "06")
                        .param("hour", "12")
                        .param("minute", "00")
                        .param("alertTimeSelection", "1 hour ago")
                        .param("description", "123456"))
                .andExpect(MockMvcResultMatchers.status().isOk());
        MvcResult getResult =
mockMvc.perform(MockMvcRequestBuilders.get("/get_memo_items"))
                .andExpect(MockMvcResultMatchers.status().isOk())
                .andReturn();
        String jsonResponse = getResult.getResponse().getContentAsString();
        JSONArray memoItemsArray = new JSONArray(jsonResponse);
        JSONObject memoItemObject = memoItemsArray.getJSONObject(0);
        Long id = memoItemObject.getLong("id");

        JSONObject payload = new JSONObject();
        payload.put("id", id);
        payload.put("labels", new JSONArray(List.of("label1", "label2")));

        memoItemObject = memoItemsArray.getJSONObject(0);
        id = memoItemObject.getLong("id");

        payload = new JSONObject();
        payload.put("id", id);
        payload.put("labels", new JSONArray(List.of("label2")));

        mockMvc.perform(MockMvcRequestBuilders.get("/get_memo_items_by_label")
                        .param("label", "label1")
                        .contentType(MediaType.APPLICATION_JSON))
                .andExpect(MockMvcResultMatchers.status().isOk());
    }
}
```

UserTest.java

```
package com.example.memobackend;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.*;

public class UserTest {

    @Before
    public void setUp() throws Exception {
    }

    @After
    public void tearDown() throws Exception {
    }

    @Test
    public void testUser() {
        User user = new User("1", "John", "123@gmail.com");
        assertEquals("1", user.getId());
        assertEquals("John", user.getName());
        assertEquals("123@gmail.com", user.getEmail());
    }
}
```

第七章 Measurement

2024/02/28	14:10 ~ 15:35
2024/03/08	19:30 ~ 21:30
2024/03/13	14:10 ~ 15:40
2024/03/18	19:00 ~ 21:45
2024/03/22	19:15 ~ 22:50
2024/03/29	19:00 ~ 21:30
2024/04/12	19:25 ~ 20:55
2024/04/17	20:30 ~ 23:00
2024/04/19	19:00 ~ 21:45
2024/04/26	19:20 ~ 22:00
2024/05/03	19:00 ~ 21:00
2024/05/10	19:00 ~ 19:55
2024/05/24	19:00 ~ 20:40
2024/05/31	14:15 ~ 15:45
2024/06/07	19:30 ~ 22:05
2024/06/12	20:30 ~ 21:30