國立台北科技大學

110-2 物件導向分析與設計

Meme Image Creator using Swift

資工四 范凱翔 107590051

資工四 陳小蘭 107590058

資工四 劉學逸 107590450

Contents

1. Requirement Document

1.1 Change History

Version	Description	Date
0.1	project proposal	2022/03/01
0.2	system features, use cases	2022/03/16
0.3	domain class diagram, use case refinement	2022/03/29
0.4	revise *figure 1.5.1, 2.1.1, 2.1.2, 2.3.1*	2022/04/19
0.5	Design (3.1 ~ 3.3)	2022/04/20
0.6	revise *figure 3.1.1 Logical Architecture*	2022/05/03
0.7	Programming, Unit Testing	2022/05/04
0.8	Phase II: refine use cases, system features, domain model	2022/05/24
1.0	revise 1.9, CH5, CH6	2022/06/15

1.2 Problem Statement

在巴哈姆特、批踢踢等網路論壇上，常常見到各種回覆都僅是貼一張哽圖就完事了，足以見哽圖對現代人溝通交流的重要性。但雖現市場上並不缺乏修圖軟體，專注面向改圖、製作哽圖等更為客製化的程式卻較為少見，除此之外，縱使在 Android 與網頁平台上或能找到些許資源，在 iOS 上卻又更寥寥無幾。據此，本組期望經由 Swift 在 Xcode 開發，製作一原生 iOS app，供使用者在任何想發圖的時候能夠便捷的改圖、產圖，並強調原生自然的使用體驗。

1.3 System Context Diagram (new)

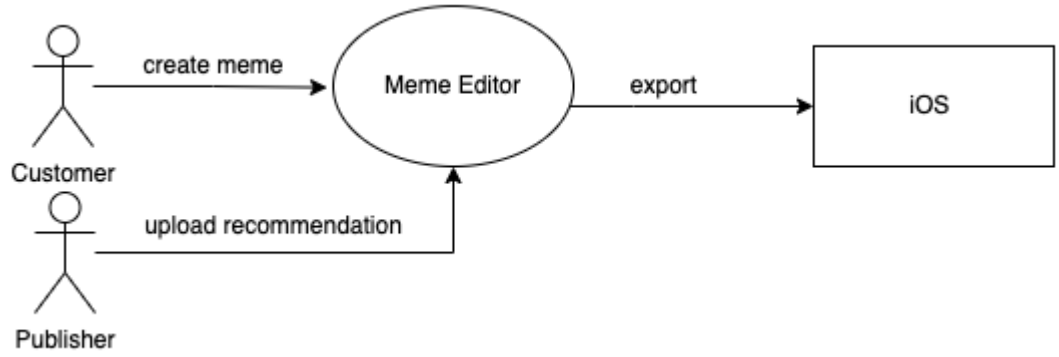


figure 2.1.1 System Context Diagram

1.4 Summary of System Features

Feature ID	Feature Description
FEA-01	Users manage their own meme creations.
FEA-02	Users import their own images to the app.
FEA-03	Users search web images to be imported to the app.
FEA-04	Image editting.
FEA-05	Editing progress saving.
FEA-06	Export meme creation as standard image formats.
FEA-07	Set image as template.
FEA-08	Account system.
FEA-09	Cloud synchronization.
FEA-10	Provide weekly recommendation memes.

1.5 Use Case Diagram (new)

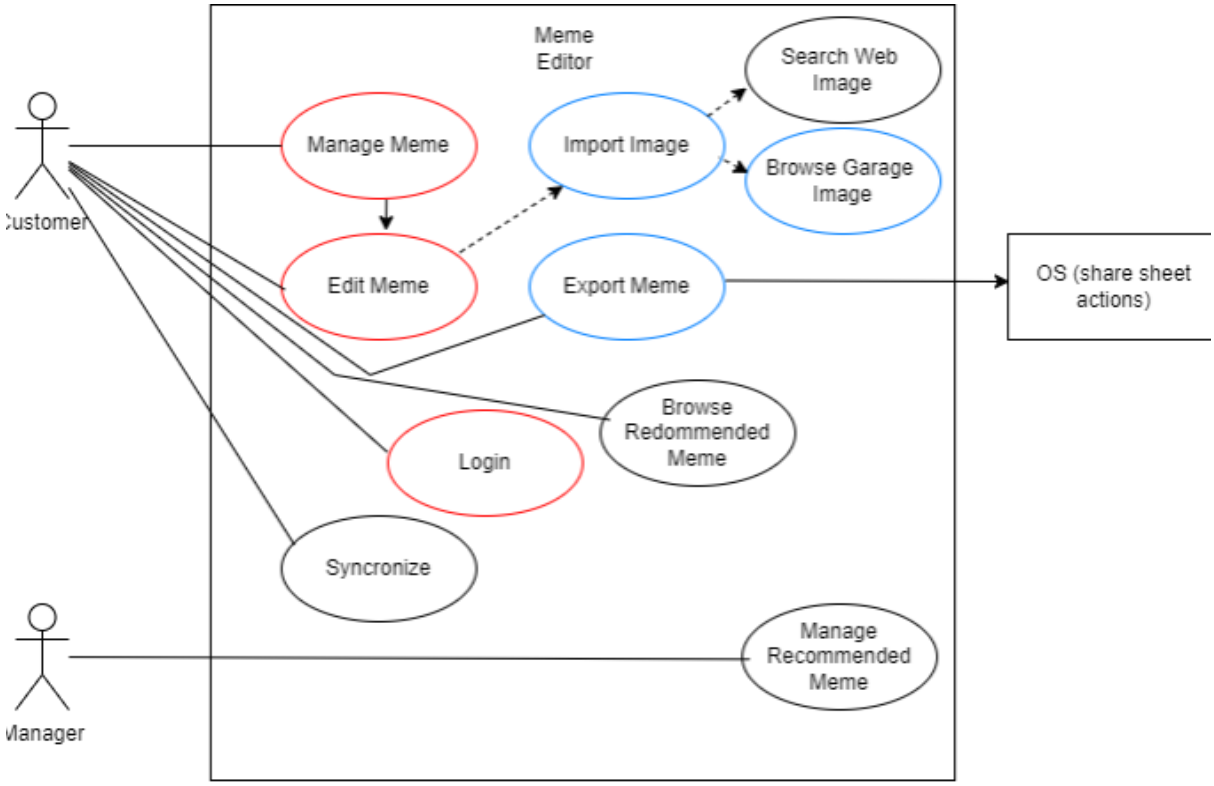


figure 1.5.1 Use Case Diagram

1.6 Use Cases

1.6.1 Manage Meme **(new)**

Use Case Section	**Comment**
Use Case Name	Manage Meme
Scope	Meme Editor application
Level	user goal
Primary Actor	customer
Stakeholders and Interests	User: Willing to manage a meme creation.
Preconditions	The device should be on iOS 15 or later.
Success Guarantee	The meme image is shared via the share sheet. The editable creation is saved in the app sandbox.
Main Success Scenario	1. User creates a new meme creation project. 2. User Import Image. 3. User Edit Meme creation.

	4. User finishes editing. 5. System stores the editable meme creation. 6. User chooses to export the meme(1.6.6). 7. System shows selection of all projects. 8. User wants to continue editing the previous creation project. 9. User chooses the wanted creation. 10. System responds and shows the latest save state of creation. 11. Go back to step 3. 12. User wants to delete the previous creation project. 13. Repeat step 9-10. 14. User chooses to delete the creation project. 15. System delete the creation project. 16. Go to 12. 17. System goes back to home screen
Extensions	*.a. At any time the app crashes or is terminated by the user or operating system: 1. The creation should be saved and the edit can be resumed. b.When loading a base image and the data is corrupted: 1. Inform the user and delete (automatic or after user confirmation) the corrupt image and creation. 1.a. When the system detects that the project name is duplicated, prompt a warning to inform the user to choose a different name. 2.a. Provide an option to cancel operation when importing the image, and go back to step 1. 8.a. If the user doesn't choose to edit previous creations, go to step 12. 12.a If the user doesn't choose to delete previous creations, go to 17.
Special Requirements	None
Technology And Data Variations List	a. The image format needs to be supported on the device's iOS version. PNG or JPEG is recommended for better performance.
Frequency of Occurrence	Often
Miscellaneous	1. How should the system handle formats like portrait photos, HDR, HIFF, HEIF… etc. ? 2. When the app suspends and the ui is on the editing screen, should the app go to the same screen when launched next time?

1.6.2 Import Image

Use Case Section	**Comment**
Use Case Name	Import Image
Scope	Meme Editor application
Level	subfunction
Primary Actor	customer
Stakeholders and Interests	User: Wants to quickly load existing images.
Preconditions	The device should be on iOS 15 or later.
Success Guarantee	The system loads the image successfully to the app.
Main Success Scenario	1. User choose the source of the image 2. System shows selection between Search Web Image(1.6.3) and Browse Local Image(1.6.4) 3. System load image successfully
Extensions	*.a. When loading fails, show a warning prompt and go to step 2
Special Requirements	None
Technology And Data Variations List	a. The image format needs to be supported on the device's iOS version. PNG or JPEG is recommended for better performance.
Frequency of Occurrence	Often
Miscellaneous	1. Should we add an option to import from the iOS file system?

1.6.3 Search Web Images

Use Case Section	**Comment**
Use Case Name	Search Image
Scope	Meme Editor application
Level	subfunction
Primary Actor	customer
Stakeholders and Interests	User: Wants to quickly use images on the Internet.
Preconditions	The device should be on iOS 15 or later.

Success Guarantee	The system shows search results.
Main Success Scenario	1. User enters the title of the meme. 2. System displays the result of images on the Internet. 3. User selects the desired image. 4. System loads the image. 5. System imports to image
Extensions	*.a. When the connection fails, show a warning prompt. 4.a. If the image took over 20 seconds to load, show "timeout" and go to step 2 2.a. Provide an option to cancel this operation
Special Requirements	None
Technology And Data Variations List	a. The image format needs to be supported on the device's iOS version. PNG or JPEG is recommended for better performance.
Frequency of Occurrence	Seldom
Miscellaneous	1. Should we add support for GIFs?

1.6.4 Browse Garage Images

Use Case Section	**Comment**
Use Case Name	Browse Local Image
Scope	Meme Editor application
Level	subfunction
Primary Actor	customer
Stakeholders and Interests	User: Wants to quickly use images on the local meme garage.
Preconditions	The device should be on iOS 15 or later.
Success Guarantee	The system shows the local gallery
Main Success Scenario	1. System accesses the garage. 2. System loads all images in the garage. 3. System displays the images in the garage. 4. User selects the desired image. 5. System loads the image. 6. System displays the image and shows options.
Extensions	2.a. When an image file can't be loaded, show a warning symbol as replacement. b.When there's no image to display, show some text to inform the user there's no image in the garage.

Use Case Section	**Comment**
Use Case Name	Browse Local Image
Scope	Meme Editor application
Level	subfunction
Primary Actor	customer
Stakeholders and Interests	User: Wants to quickly use images on the local meme garage.
Preconditions	The device should be on iOS 15 or later.
Special Requirements	None
Technology And Data Variations List	a. The image format needs to be supported on the device's iOS version. PNG or JPEG is recommended for better performance.
Frequency of Occurrence	Often
Miscellaneous	None

1.6.5 Edit Meme

Use Case Section	**Comment**
Use Case Name	Edit Meme
Scope	Meme Editor application
Level	user goal
Primary Actor	customer
Stakeholders and Interests	User: Wants a simple and easy-to-use feature for editing images.
Preconditions	The device should be on iOS 15 or later.
Success Guarantee	The system should implement and save the changes that the user made to the image.
Main Success Scenario	1. User has an image to edit. 2. User chooses an edit option and the system shows the corresponding interface for each editing option. 3. User interacts with the interface and finishes the editing action. 4. The system changes to creation according to the edit the user made. 5. Go to (2.) if the user wants to make more edits. 6. Editing complete and the user can continue with saving or "export meme(1.6.6)".
Extensions	*.a. At any time the app crashes or is terminated by the user or operating system: 1. The creation should be saved and the edit can be resumed. b. When loading creations and data is corrupted: 1. Inform the user and delete (automatic or after user confirmation) the corrupt creations. 2.a. Edit by Adding Text: 1. User chooses to add texts to the image. 2. System asks the user for the text. 3. User provides the system the text to input. 4. System adds the text to the image. 5. User adjusts the text position and style. 6. System implements the customization the user made. b. Edit by Adding Effect: 1. User chooses to add filter effects to the image. 2. System provides effects for the user to select. 3. User chooses the filter to be applied. 4. System implements the customization the user made. c. Edit by Crop:

	1. User chooses to crop the image. 2. User adjusts the area to be cropped. 3. System implements the customization the user made. d. Edit by Rotate Image: 1. User chooses to rotate the image 2. User adjust the degree of the image 3. System implements the customization the user made. e. Edit by Adding Drawing: 1. User chooses to add drawing to the image. 2. System provides drawing tools. 3. User is able to draw on the image. 4. System implements the customization the user made. f. Edit by Adding Image: 1. User chooses to add an existing image to the current image. 2. System calls Import Image(1.6.2) and continues with the import procedure. 3. System finishes loading the imported image. 4. User adjusts the position of the imported image. 5. System implements the customization the user made.
Special Requirements	None
Technology And Data Variations List	a. The image format needs to be supported on the device's iOS version. PNG or JPEG is recommended for better performance.
Frequency of Occurrence	Often
Miscellaneous	1. How should the system handle formats like portrait photos, HDR, HIFF, HEIF… etc. ? 2. When the app suspends and the ui is on the editing screen, should the app go to the same screen when launched next time? 3. How should we handle different edits made to the creation? (z-axis position… etc.)

1.6.6 Export Meme

Use Case Section	Comment
Use Case Name	Export Meme
Scope	Meme Editor application
Level	subfunction
Primary Actor	customer
Stakeholders and Interests	User: Wants to share memes on the Internet.
Preconditions	a. The device should be on iOS 15 or later. b. There's at least one creation in the gallery.
Success Guarantee	The meme image is shared via the share sheet. The system returns to the gallery or previous interface.
Main Success Scenario	1. User launches app. 2. User creates a meme(1.6.1) creation. 3. User chooses the share sheet. 4. System displays the share sheet. 5. User selects a share sheet action. 6. System returns to the gallery after the sharing action is finished.
Extensions	*.a. At any time the app crashes or is terminated by the user or operating system: 1. The creation should be saved and the edit can be resumed. b. When loading creations and data is corrupted: 1. Inform the user and delete (automatic or after user confirmation) the corrupt creations. 2.a. When user chooses an existing creation: 1. System shows the available creation for user to choose 2. User choose the creation
Special Requirements	None
Technology And Data Variations List	a. The image format needs to be supported on the device's iOS version. PNG or JPEG is recommended for better performance.
Frequency of Occurrence	Often
Miscellaneous	1. Should we provide options like quality and image format?

1.6.7 Browse Recommended Meme

Use Case Section	**Comment**
Use Case Name	Browse Recommended Meme
Scope	Meme Editor application
Level	user goal
Primary Actor	customer
Stakeholders and Interests	User: Wants to browse meme images recommended by the manager.
Preconditions	The device should be on iOS 15 or later.
Success Guarantee	System loads and shows images.
Main Success Scenario	1. User goes to the “Recommended Meme” section. 2. System loads data from the database. 3. System shows images and options.
Extensions	*.a. When the connection fails, show a warning prompt. b. When there's a non supported format in the database, skip and ignore the file.
Special Requirements	None
Technology And Data Variations List	a. Format of files in the database should be supported by the device’s iOS version.
Frequency of Occurrence	Often
Miscellaneous	1. Should we discard old recommendations when new recommendations are added to declutter?

1.6.8 Login

Use Case Section	**Comment**
Use Case Name	Login
Scope	Meme Editor application
Level	user goal
Primary Actor	customer
Stakeholders and Interests	User: Wants to Login to their account.
Preconditions	The device should be on iOS 15 or later.

Success Guarantee	System able to authenticate the User
Main Success Scenario	1. User Identifies their identity. 2. System authenticates identity. 3. System successfully authenticate User and returns to the user gallery. 4. System automatically starts the Cloud Synchronization(1.6.9).
Extensions	a. At (2.), When the system fails to authenticate the user due to incorrect data, prompt a warning to inform the user to re-authenticate. a. At (2.), the app crashes or is terminated by the user or operating system: i. The system should not authenticate the user.
Special Requirements	None
Technology And Data Variations List	None
Frequency of Occurrence	Often
Miscellaneous	1. Should we support more than one login option?

1.6.8 Manage Recommended Meme

Use Case Section	**Comment**
Use Case Name	Manage Recommended Meme
Scope	Database
Level	user goal
Primary Actor	Manager
Stakeholders and Interests	Manager: Want to manage recommended memes for the user.
Preconditions	The device should be on iOS 15 or later.
Success Guarantee	1. Manager choses “manage recommendation” option. 2. System asks for authentication. 3. Manager choses option to create or delete memes. 4. System changes data in the database accordingly. 5. Users can browse the uploaded memes (1.6.6).
Main Success Scenario	1. Manager uploads meme images and its data to the database successfully. 2. System reads the database successfully.
Extensions	*.a. When the connection fails when downloading from the

	database, discard the file and prompt the manager to try again.
Special Requirements	None
Technology And Data Variations List	a. Format of files in the database should be supported by the system.
Frequency of Occurrence	1 / week
Miscellaneous	1. How to implement the interface to hide this option from regular users(customers)?

1.6.9 Cloud Synchronize

Use Case Section	**Comment**
Use Case Name	Cloud Synchronize
Scope	Meme Editor application
Level	user goal
Primary Actor	customer
Stakeholders and Interests	User: Synchronize the editing progress and user data.
Preconditions	a. The device should be on iOS 15 or later. b. The user should be logged in.
Success Guarantee	1. The system successfully uploads new changes to the cloud. 2. The system successfully downloads new data from the cloud to the gallery.
Main Success Scenario	1. User is logged in. 2. User chooses to sync. 3. System syncs between local gallery and the cloud.
Extensions	*.a. When the user isn't logged in, prompt the user to login. b. When the data is corrupt, abort the changes and inform the user an error has occurred.
Special Requirements	None
Technology And Data Variations List	None
Frequency of Occurrence	Often
Miscellaneous	1. Should we use third party cloud storage as alternative solutions?

1.7 Non-functional Requirements and Constraints

NRF ID	Category	Description
NRF-01	Performance	When opening a creation it should take less than 1 second.
NRF-02	Reliability	The program should not crash when the imported image format is not expected.
NRF-03	Usability	Handle inactive, background, suspend state correctly.

1.8 Glossary

Term	Definition or Description
meme	An image, video, piece of text, etc., typically humorous in nature, that is copied and spread rapidly by internet users, often with slight variations.[1]
creation	An entity of editable image and text layers.
gallery	A place where users can select meme creations.
garage	A place where users can select meme images they made.
artifact	Exported image

[1] source: https://www.lexico.com/en/definition/meme

1.9 Software Environments

Platform: iOS
Development Language: Swift
UI Framework: SwiftUI

2. Domain model

2.1 Domain class diagram showing only concepts

Identified noun phrases from use cases:

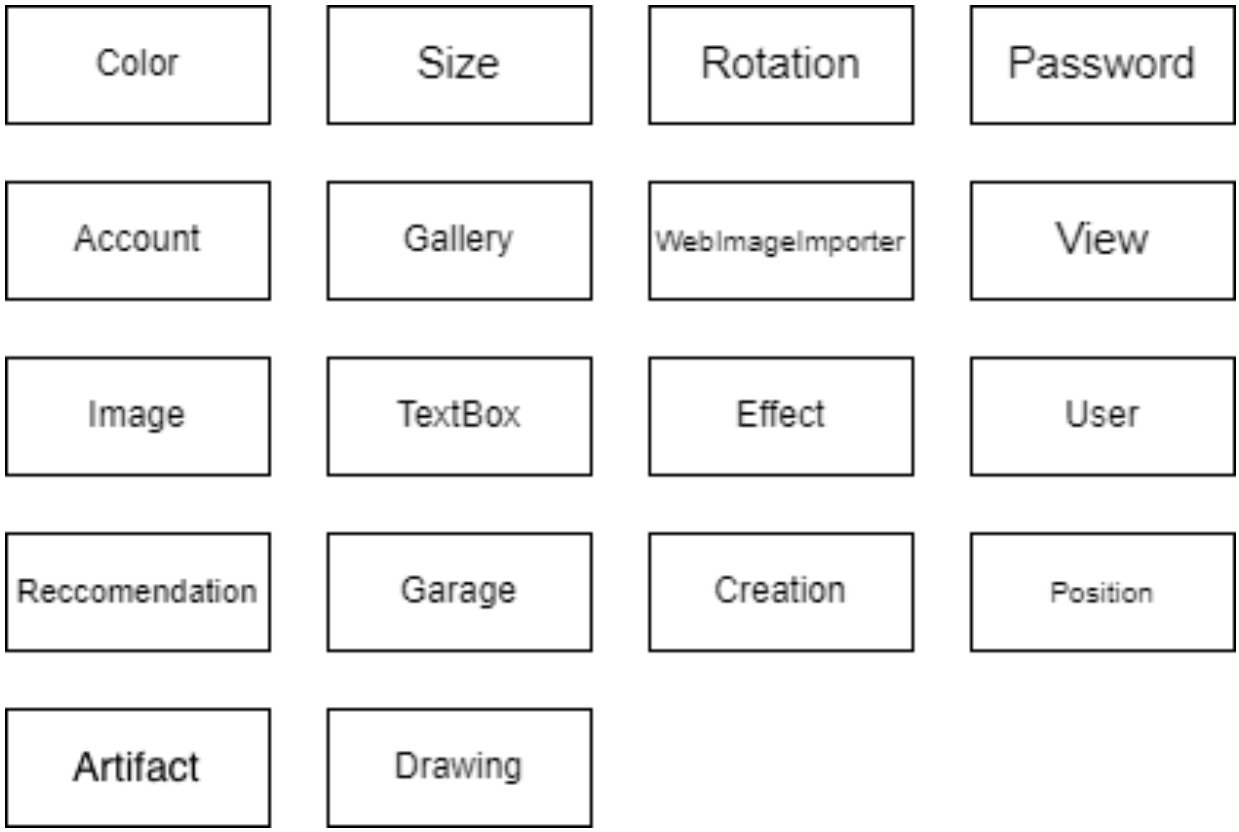


figure 2.1.1 Domain Model Nouns

Bad Classes:

Attributes:

Name, Color, Size, Rotation, password

Role:

User

Redundant:

View

Classes Identified (new) :

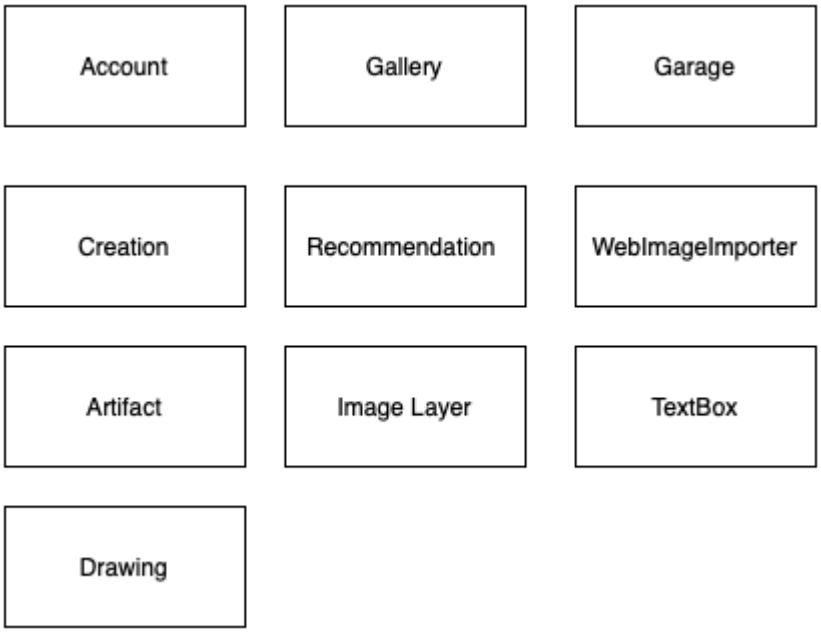


figure 2.1.2 Domain Model Classes

2.2 Add Associations (new)

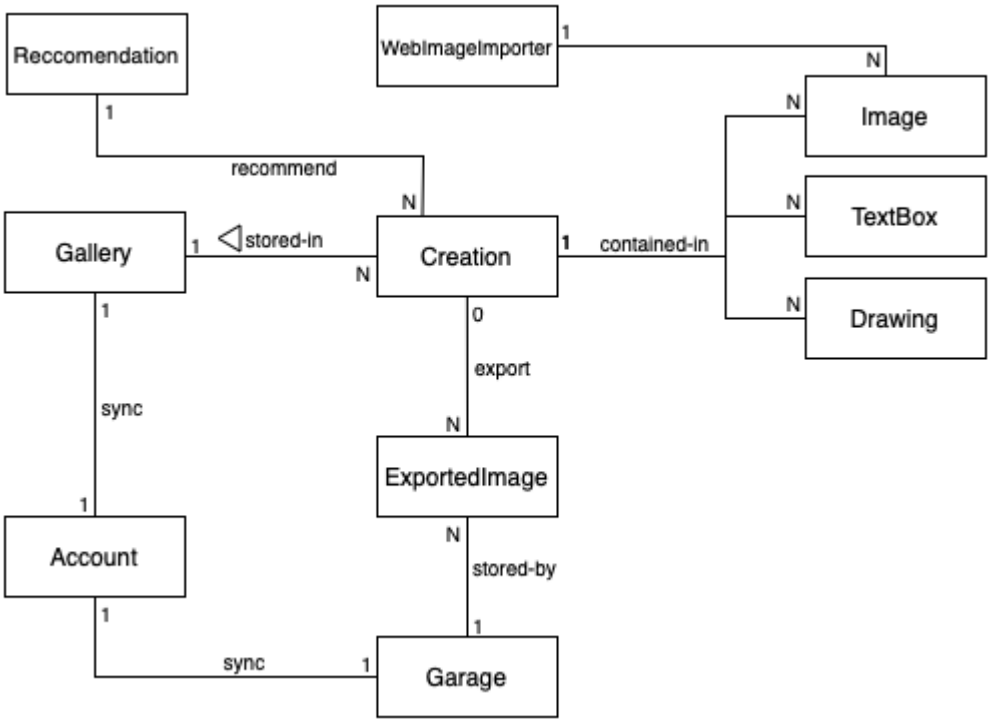


figure 2.2.1 Domain Model Associations

2.3 Add Attributes (new)

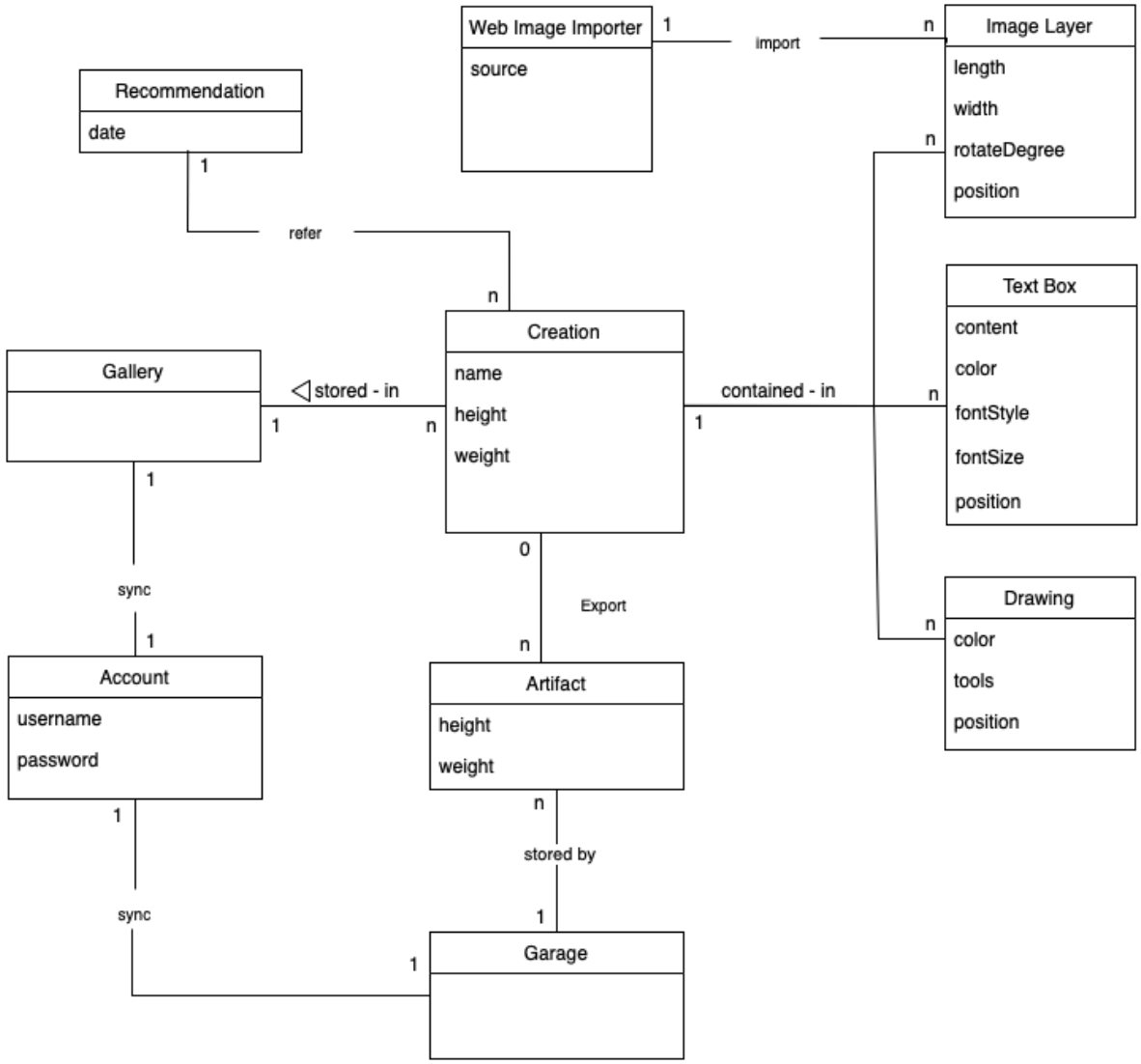


figure 2.3.1 Domain Model Attributes

說明：

Image Layer 是指 app 專案內的圖層，非一般圖片檔。

Galllery 和 Garage 透過 Account 與資料庫進行同步（sync）。

3. Design

3.1 Logical Architecture

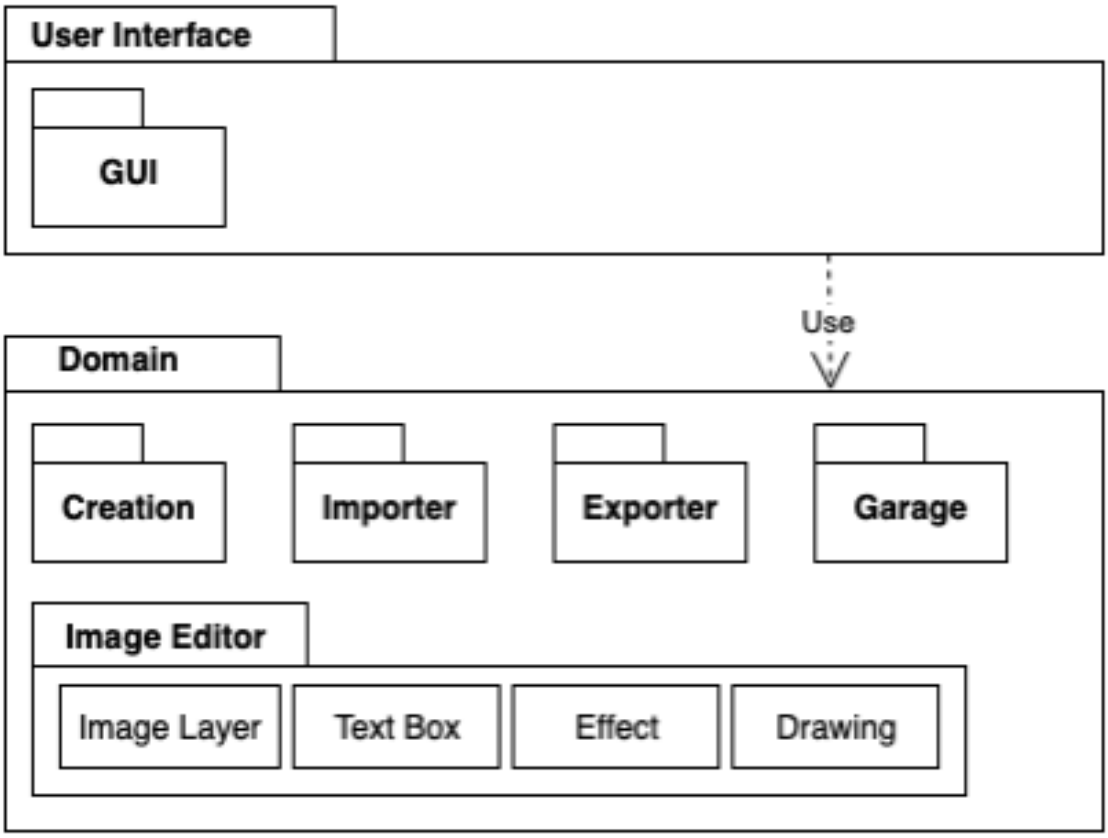


figure 3.1.1 Logical Architecture

3.2 Use-Case Realizations with GRASP Patterns

3.2.1 System Sequence Diagram

(new)

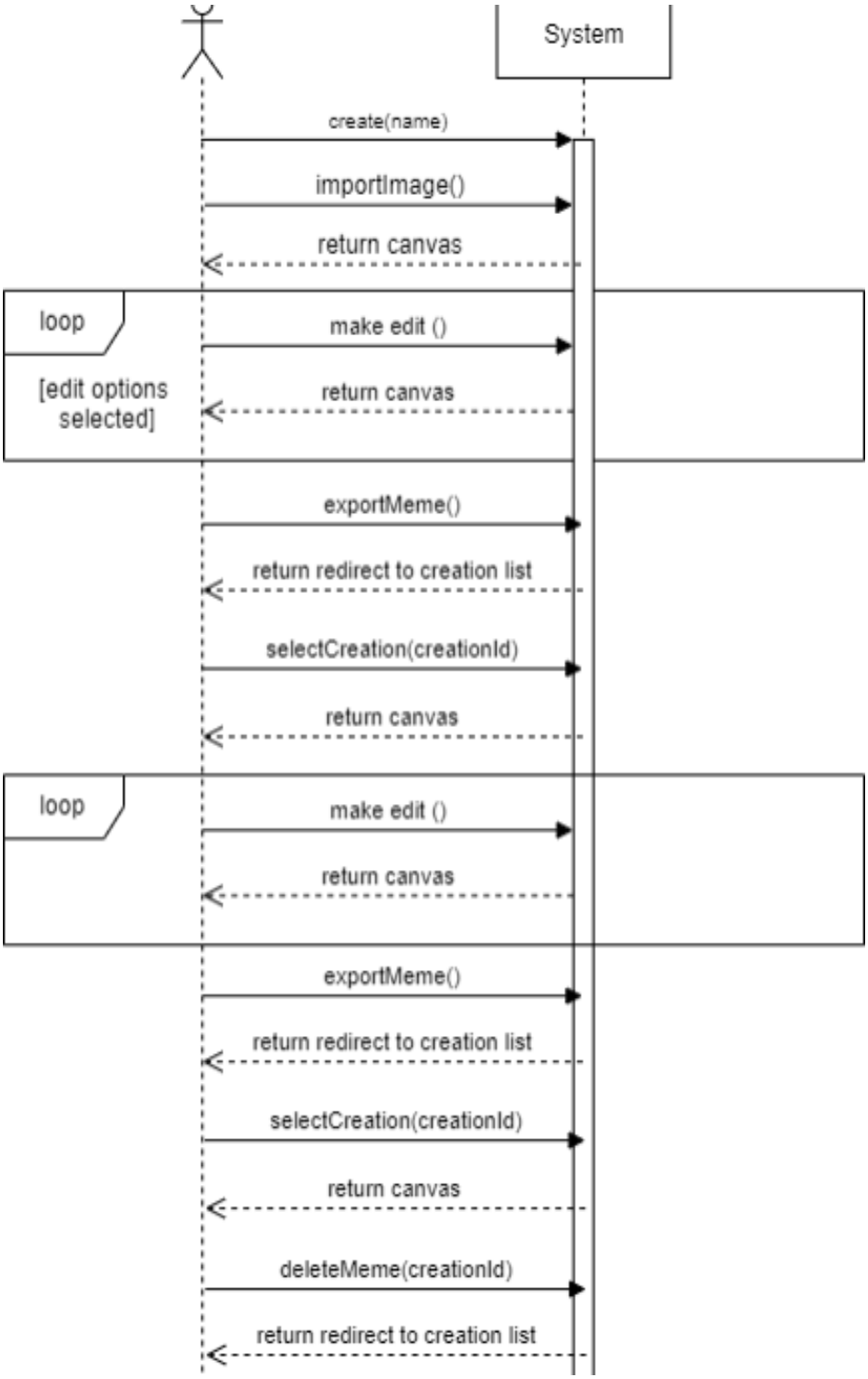


Figure 3.2.1.1 System Sequence Diagram of use case: Manage Meme

(new)

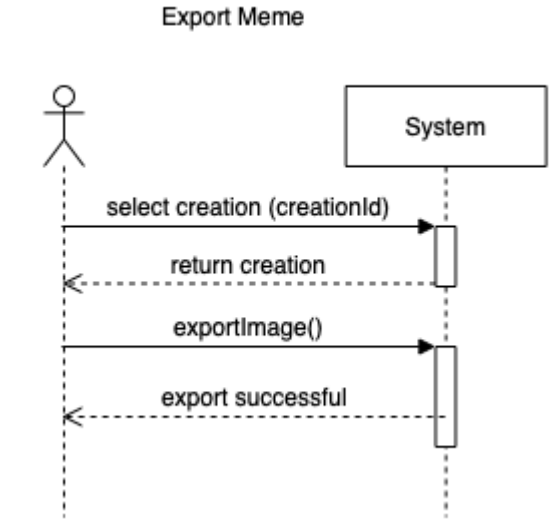


Figure 3.2.1.2 System Sequence Diagram of use case: Export Meme

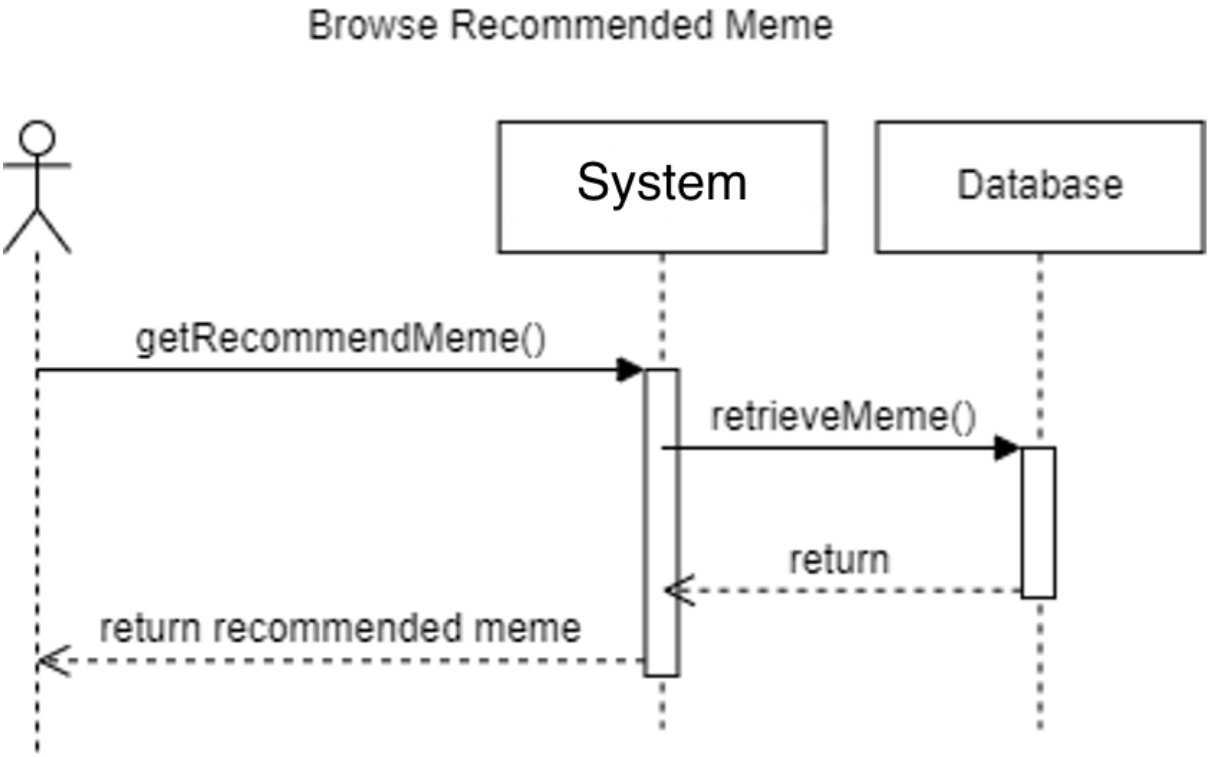


Figure 3.2.1.3 System Sequence Diagram of use case: Browse Recommended Meme

(new)

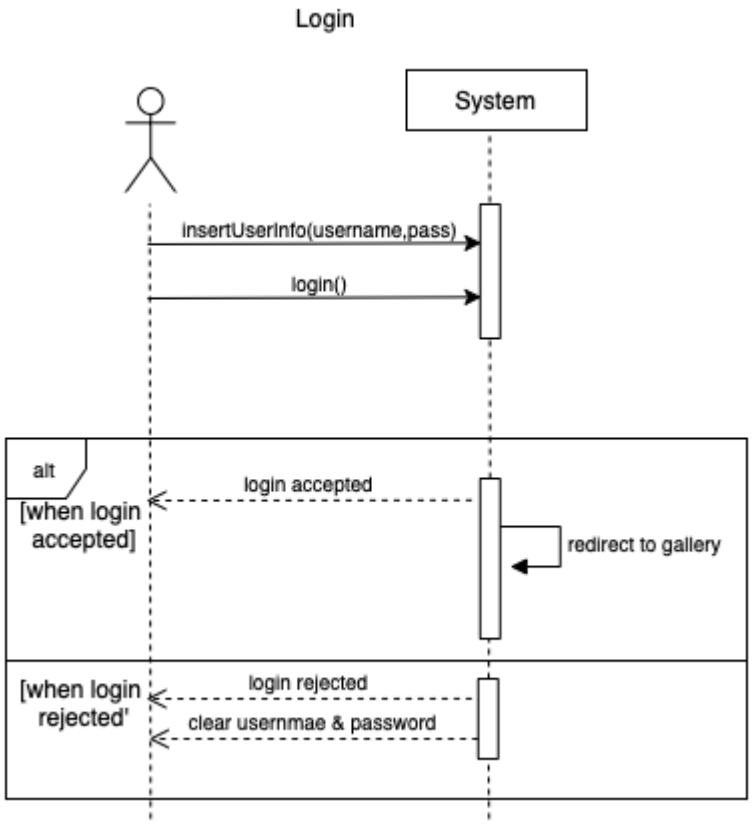


Figure 3.2.1.4 System Sequence Diagram of use case: Login

3.2.2.Operation Contract

3.2.2.1 create

Operation	createMeme() : String
Cross References	Use Case: Manage Meme
Preconditon	A Gallery instance has created
Postconditions	- A Creation instance has created (instance creation) - Creation name became name(attribute modification) - A CreatonList instance has created (instance creation)

3.2.2.2 importImage

Operation	importImage()
Cross References	Use Case: Import Image
Preconditon	- A Gallery instance has created - An ImporterHandler instance has created

Postconditions	- An ImageSearcher / ImagePicker instance has created (instance creation) - An GraphicList instance has created

3.2.2.3 findImage

Operation	findImage(title : String)
Cross References	Use Case: Search Web Image
Preconditon	- User has select to import the image from web - An ImporterHandler instance has created - An ImporterSearcher instance has created
Postconditions	An ImageList instance has created (instance creation)

3.2.2.4 selectImage

Operation	selectImage(imageId : int)
Cross References	Use Case: Search Web Image
Preconditon	- An ImporterHandler instance has created - An ImporterSearcher instance has created - An ImageList instance has created
Postconditions	An ImageLayer instance has created (instance creation)

3.2.2.5 selectFromPhotoLibrary

Operation	selectFromPhotoLibrary()
Cross References	Use Case: Browse Garage Image
Preconditon	Image Picker instance has created
Postconditions	ImageLayer instance has created (instance creation)

3.2.2.6 makeEdit

Operation	makeEdit()
Cross References	Use Case: Edit Image
Preconditon	- A Gallery instance has created - An GraphicList instance has created
Postconditions	- An ToolSelector instance has created - An TextBox instance has created - An Effect instance has created

	- An Drawing instance has created - An ImageLayer instance has created

3.2.2.7 getReccomendedMeme

Operation	getReccomendedMeme()
Cross References	Use Case: Browse Recommended Meme
Preconditon	- A Meme Recommendation instance has created - A Database instance has created
Postconditions	- An ImageList instance has created (instance creation)

3.2.2.8 selectCreation

Operation	selectCreation(creationId : int)
Cross References	Use Case: Export Meme
Preconditon	- A Gallery instance has created - A CreationList instance has created
Postconditions	

3.2.2.9 exportImage

Operation	exportImage()
Cross References	Use Case: Export Meme
Preconditon	- Am ExportHandler instance has created - A Creation instance has created
Postconditions	- Creation associated with ExportHandler (associated formed) - System save exported image to Photo Library or trigger the OS share menu.

3.2.2.10 insertUserInfo

Operation	insertUserInfo(username : string, password : string)
Cross References	Use Case: Login
Preconditon	An Authentication instance has created
Postconditions	- An User instance has created (instance creation) - Account username became username (attribute modification) - Account password become password (attribute

	modification)

3.2.2.11 login

Operation	login()
Cross References	Use Case: Login
Preconditon	- An Authentication instance has created - A FirebaseSevice instance has created
Postconditions	

3.2.2.12 sync

Operation	sync()
Cross References	Use Case: Login
Preconditon	- A Authentication instance has created - An DatabaseHandler instance has created - An Gallery instance has created
Postconditions	Gallery associated with DatabaseHandler (associated formed)

3.2.2.13 deleteMeme (new)

Operation	deketeMeme() : int
Cross References	Use Case: Manage Meme
Preconditon	- A Gallery instance has created - An DatabaseHandler instance has created - A Creation instance has created
Postconditions	- Creation is removed(attribute modification)

3.2.3 Operation Sequence Diagram

(new)

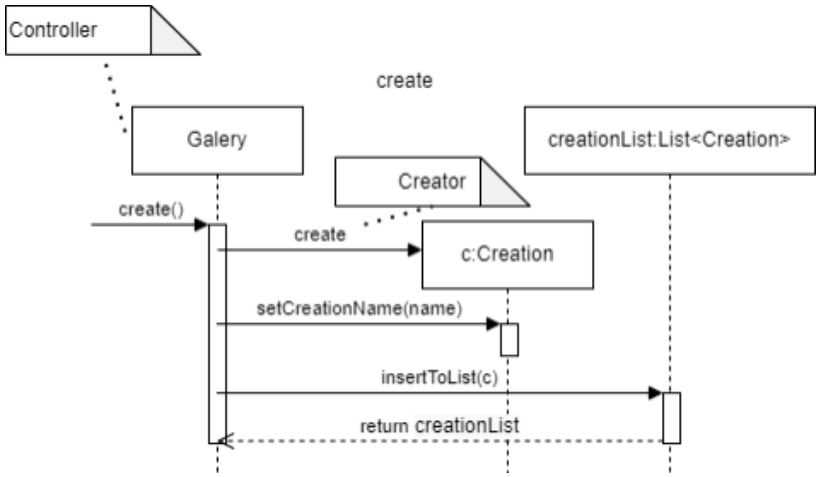


Figure 3.2.3.1 System Sequence Diagram of Operation create

(new)

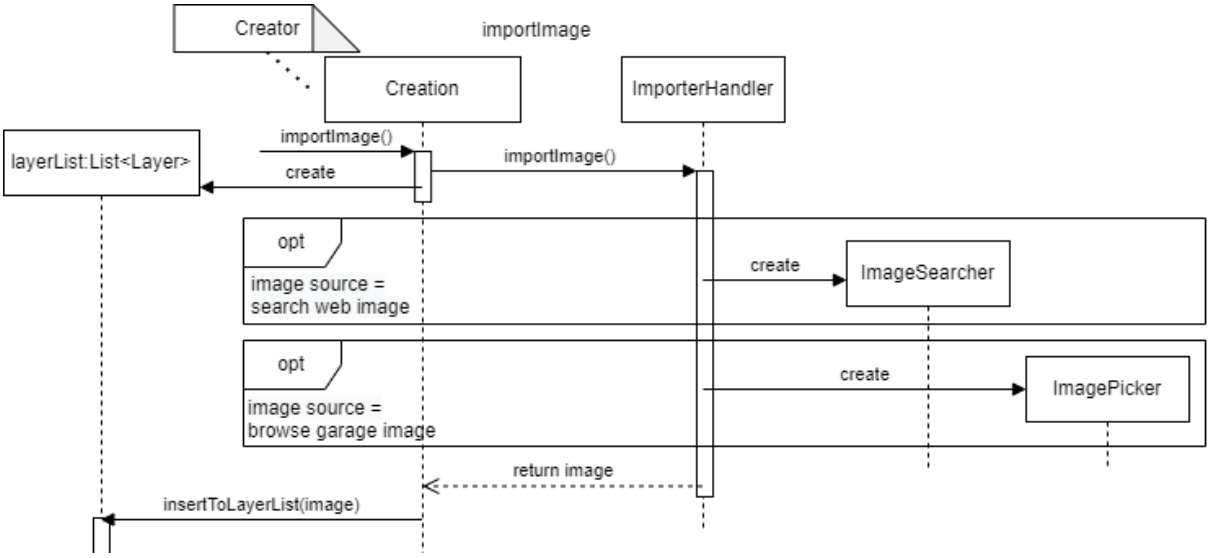


Figure 3.2.3.2 System Sequence Diagram of Operation importImage

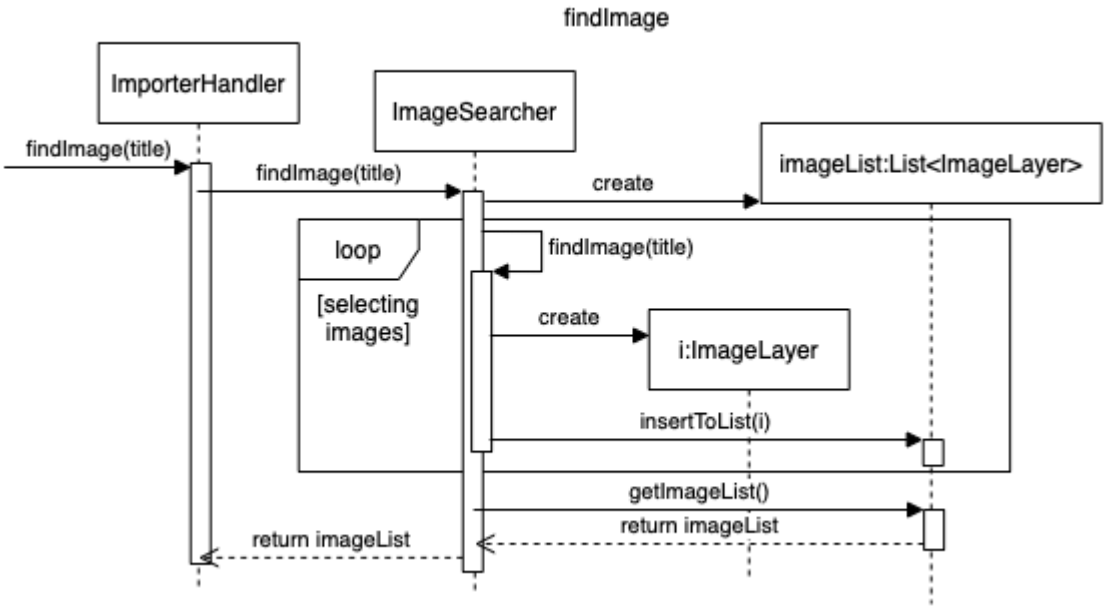


Figure 3.2.3.3 System Sequence Diagram of Operation findImage

(new)

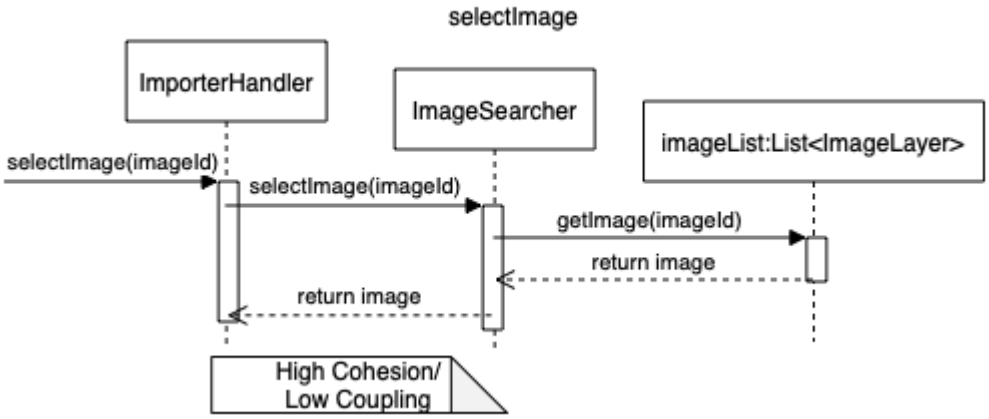


Figure 3.2.3.4 System Sequence Diagram of Operation selectImage

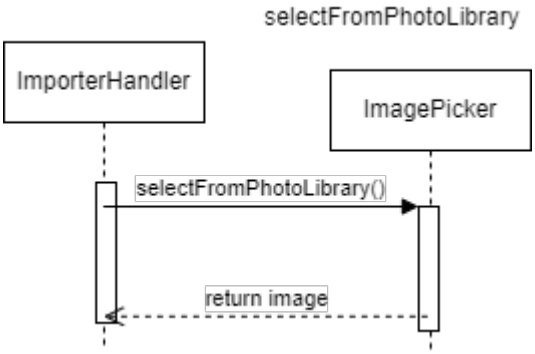


Figure 3.2.3.5 System Sequence Diagram of Operation selectFromLibrary

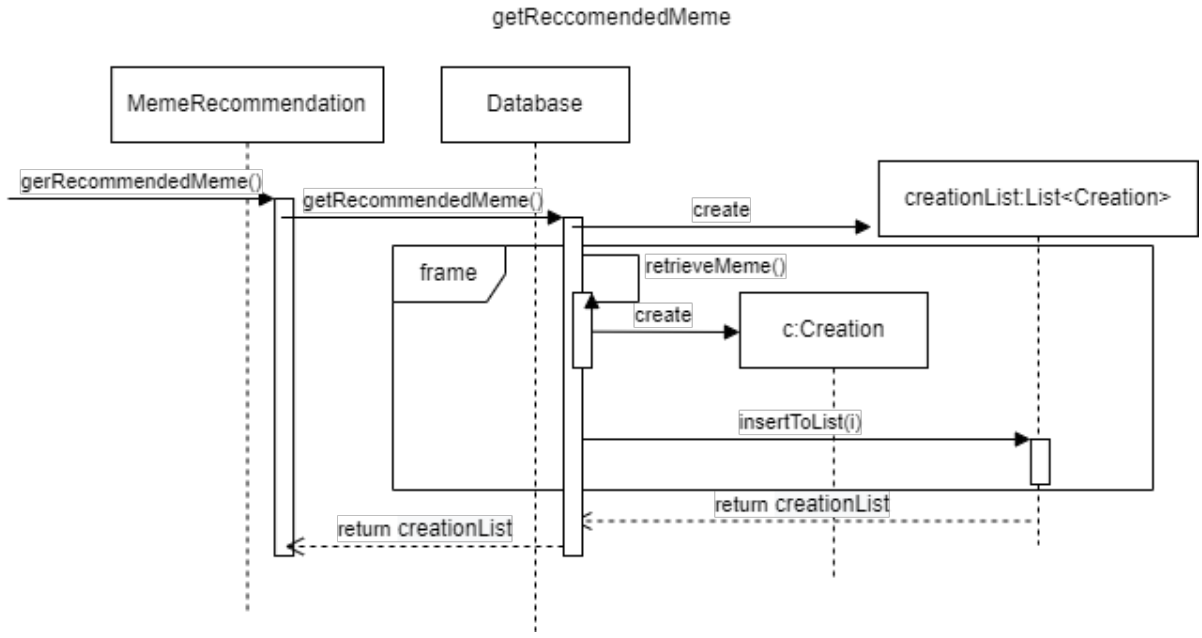


Figure 3.2.3.6 System Sequence Diagram of Operation getReccomendedMeme

(new)

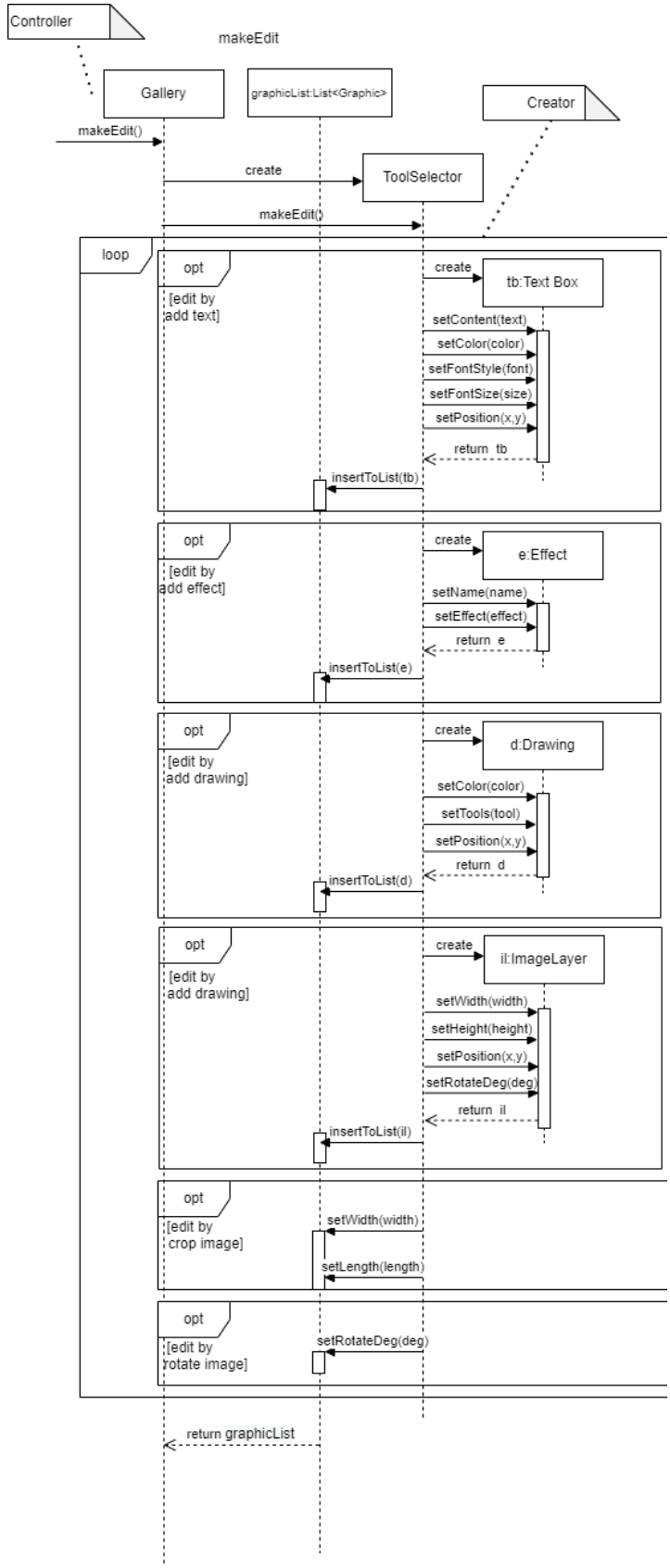


Figure 3.2.3.7 System Sequence Diagram of Operation makeEdit

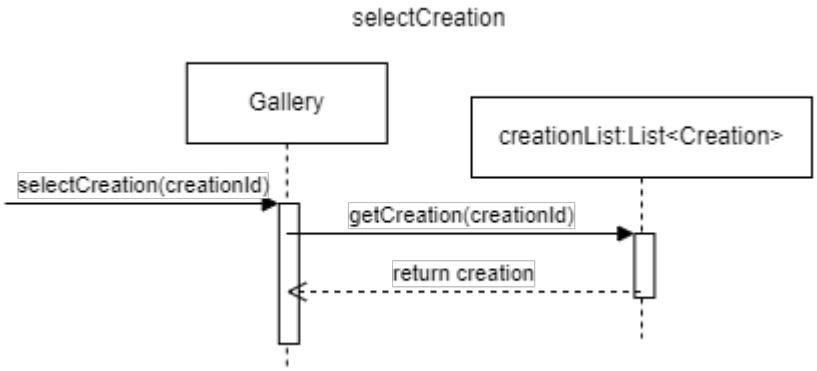


Figure 3.2.3.8 System Sequence Diagram of Operation selectCreation
(new)

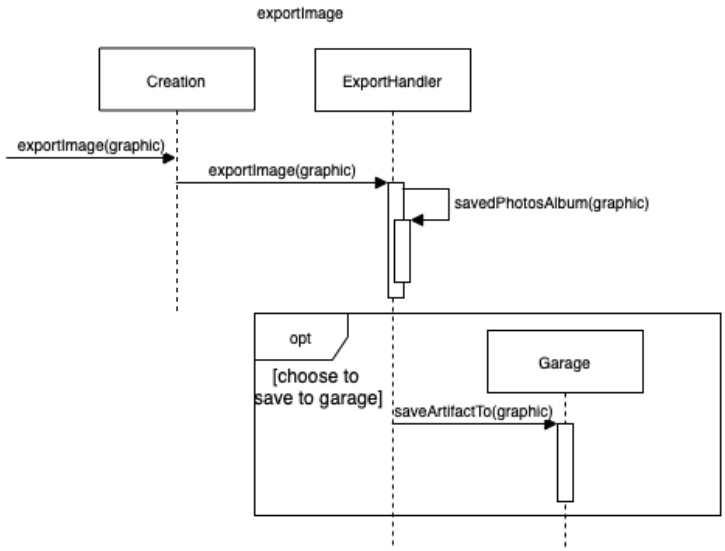


Figure 3.2.3.9 System Sequence Diagram of Operation exportImage

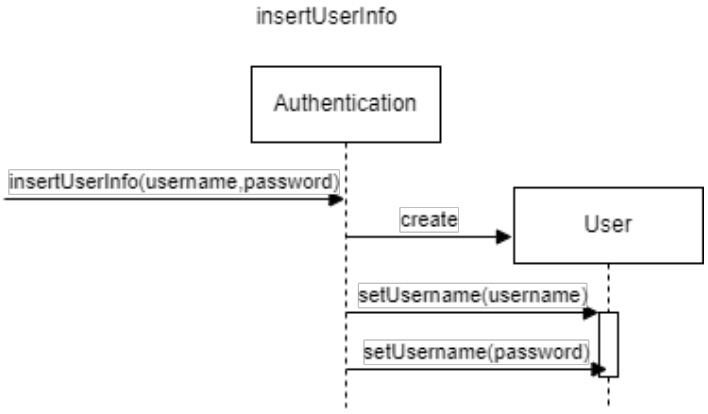


Figure 3.2.3.10 System Sequence Diagram of Operation insertUserInfo

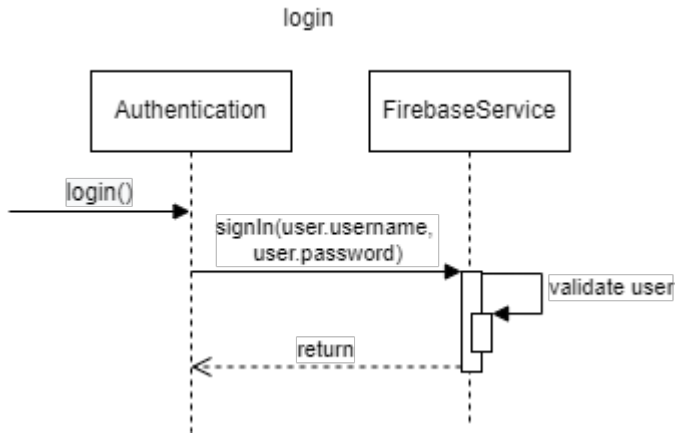


Figure 3.2.3.11 System Sequence Diagram of Operation login

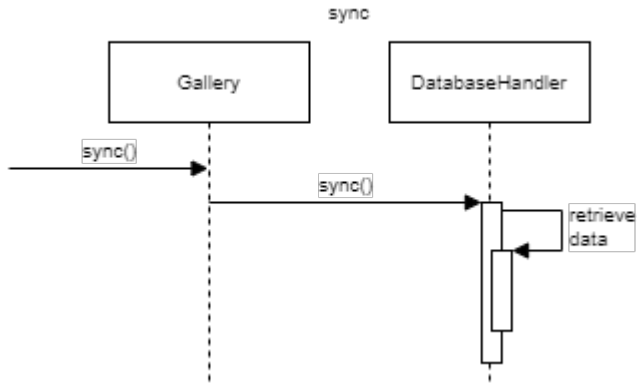


Figure 3.2.3.12 System Sequence Diagram of Operation sync

(new)

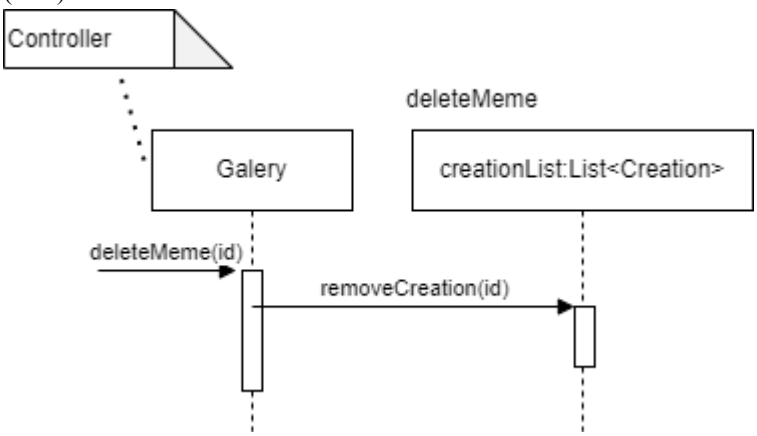
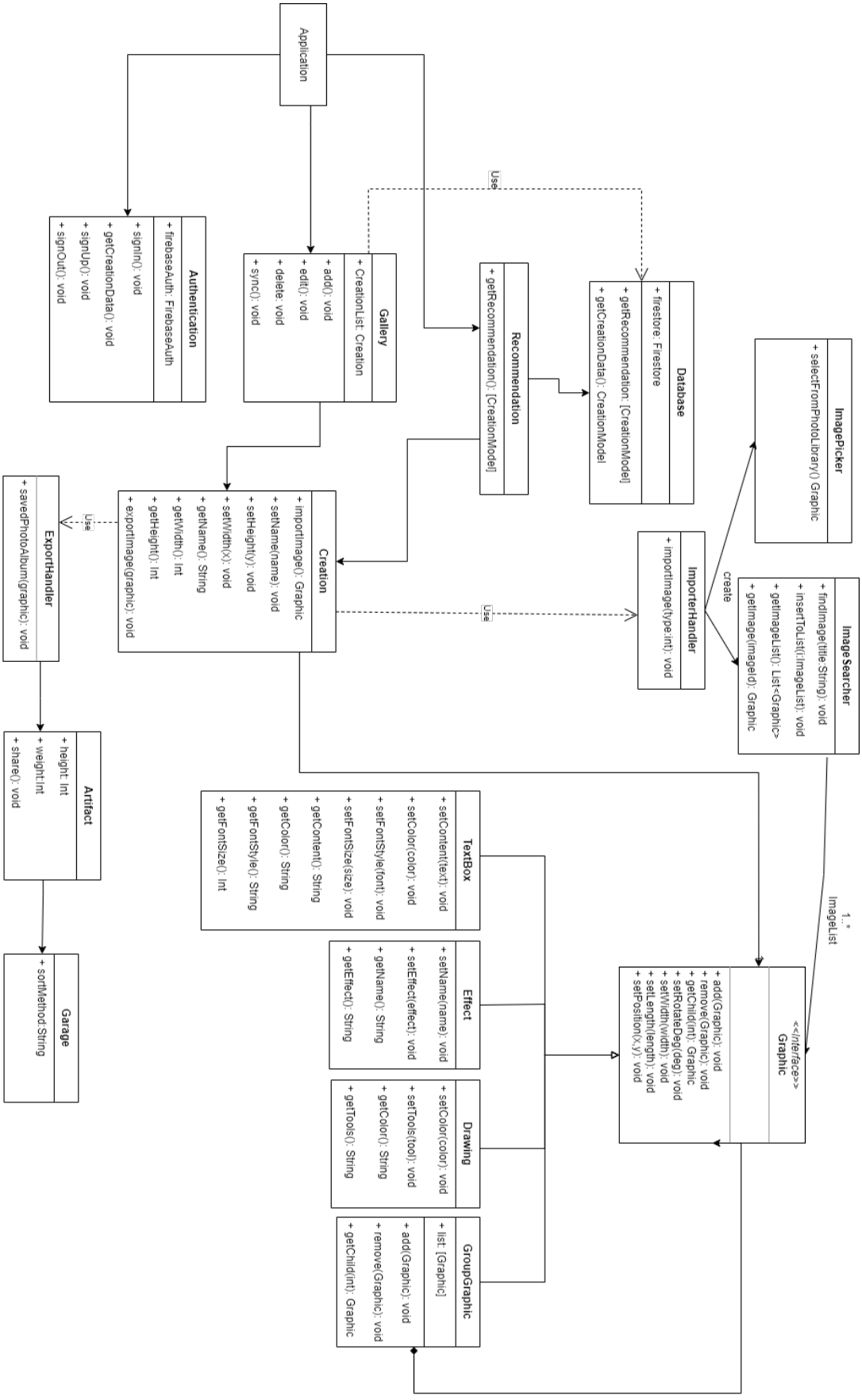


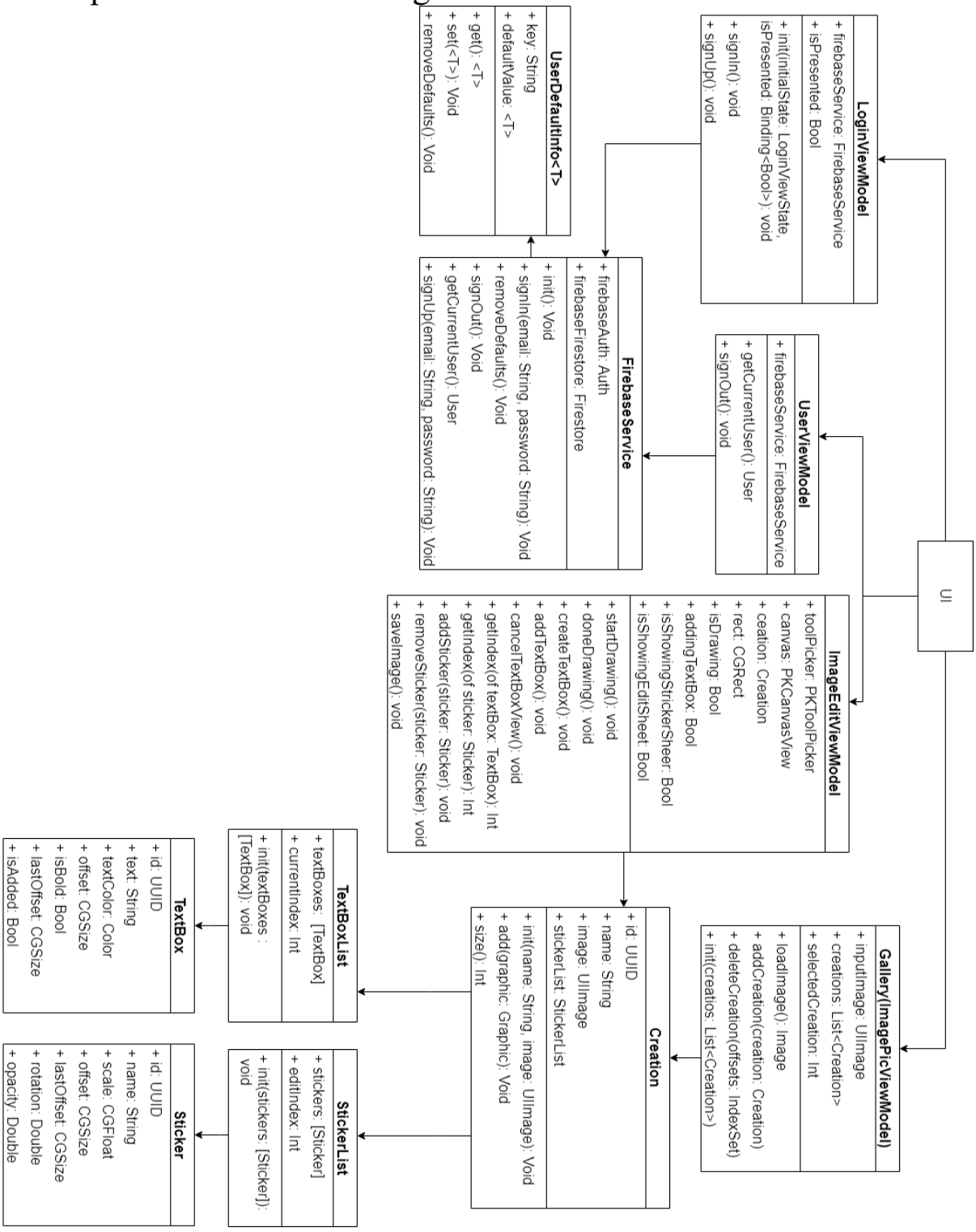
Figure 3.2.3.13 System Sequence Diagram of Operation deleteMeme

3.3 Design Class Model



Implementation Class Model

4.1 Implementation Class Diagram



4.2 Difference Between Implementation Class Model and Design Class Model **(new)**

Table 4.2.1 Comparison with design and implementation class

Class	**Method**	**Design**	**Imp.**
Canvas	makeUIView	No	Yes
	updateUIView	No	Yes
	makeCoordinator	No	Yes
Creation	addTextBoxes	No	Yes
	textBoxSize	No	Yes
	addStickers	No	Yes
	stickerSize	No	Yes
StickerList	init	No	Yes
TextBoxList	init	No	Yes
Coordinator (Canvas)	init	No	Yes
	deinit	No	Yes
ImageEditViewModel	showingFeaturesToolBar	No	Yes
	startDrawing	No	Yes
	doneDrawing	No	Yes
	createTextBox	Yes	Yes
	cancelTextBoxView	No	Yes
	saveImage	Yes	Yes
ImagePickViewModel	loadImage	Yes	Yes

ImagePicker	makeUIViewController	No	Yes
	makeCoordinator	No	Yes
Coordinator (ImagePicker)	init	No	Yes
	picker	No	Yes
FirebaseService	signIn	Yes	Yes
	signUp	Yes	Yes
	SignOut	Yes	Yes
	resetPassword	Yes	No
LoginViewModel	signIn	No	Yes
	signUp	No	Yes
	resetPassword	No	Yes
UserViewModel	getCurrentUser	No	Yes
	signOut	No	Yes

Table 4.2.2 Summary of implementation class/method changed

	Number of added	**Number of removed**	**Number of modified**
Class	2	9	1
Method	5	4	2

4.3 Calculate Line of Code

Table 4.3.1 Line of Code

No	Class Name	Number of methods	Line of Code in Class (without comment)
1	Canvas	3	51

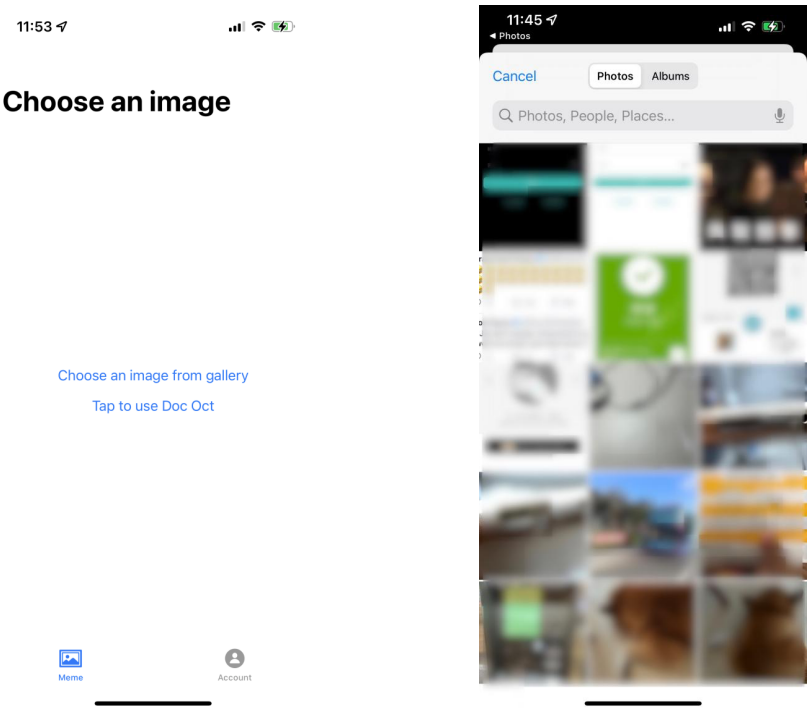
2	Coordinator(Canvas)	2	12
3	ImageEditViewModel	12	138
4	FireBaseService	3	52
5	ImagePickViewModel	6	38
6	LoginViewModel	3	32
7	UserViewModel	2	28
8	ImagePicker	2	19
9	Coordinator(ImagePicker)	2	22
10	Creation	5	36
11	TextBoxList	1	9
12	StickerList	1	9

Programming

5.1 Snapshots of system execution

Figure 5.1.1 Login Page

Figure 5.1.2 Image Picker Page



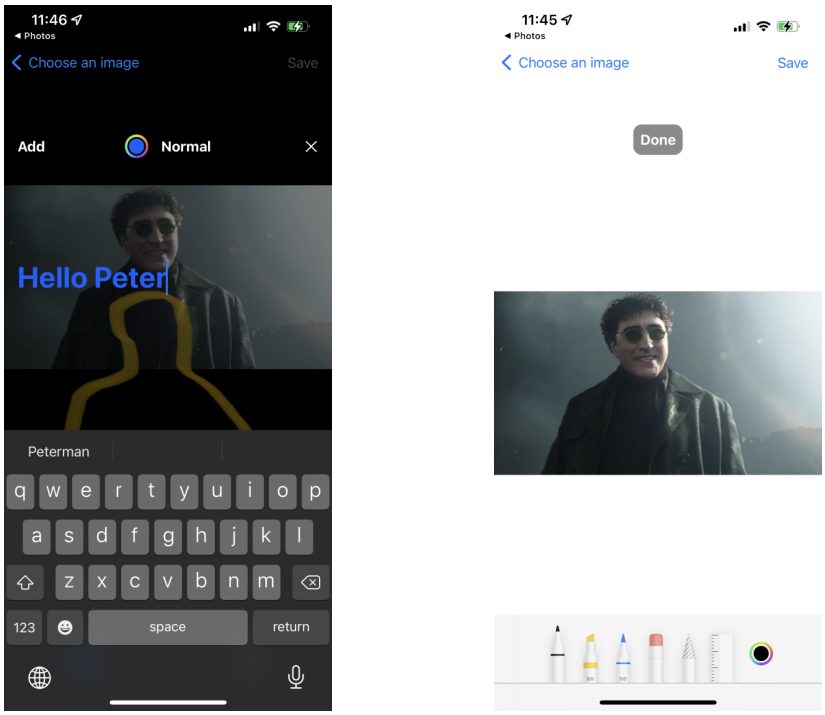


Figure 5.1.3 Editing Page

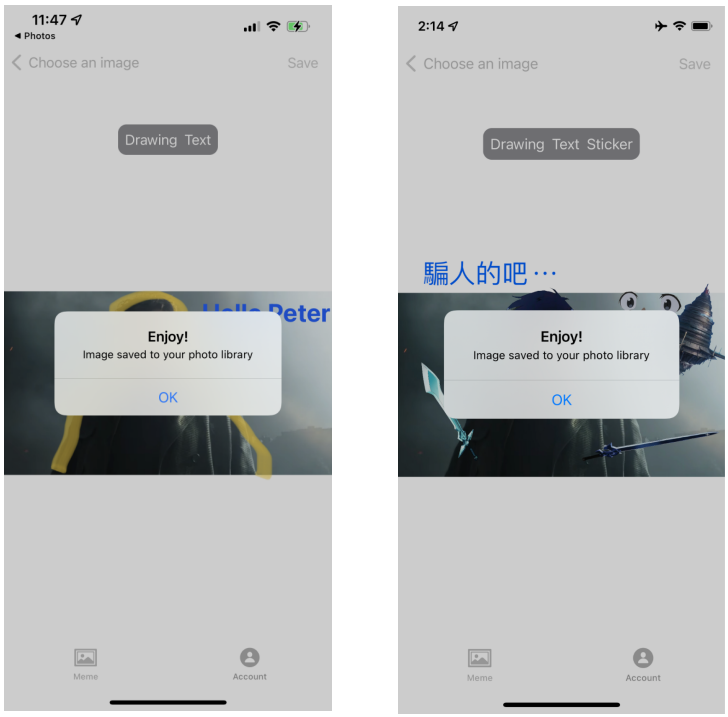


Figure 5.1.4 Export Image

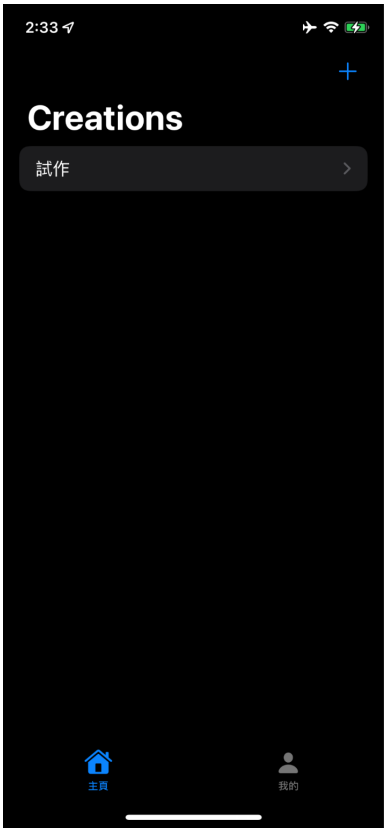


Figure 5.1.5 Creation List (new)

Figure 5.1.6 Sticker Selecter

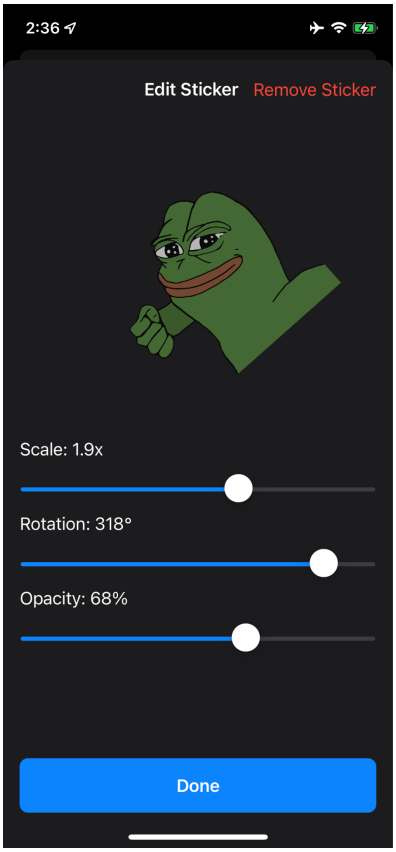


Figure 5.1.7 Sticker Editor

5.2 Source Code Listing

TextBox.Swift

```
import SwiftUI
import PencilKit

class TextBox: Graphic, Identifiable {
    var id = UUID()
    var text = ""
    var isBold = true
    var offset: CGSize = .zero
    var lastOffset: CGSize = .zero
    var textColor: Color = .white
    var isAdded = false
}
```

TextBoxList.swift

```
import Foundation

class TextBoxList {
    var textBoxes: [TextBox]
    var currentIndex: Int = 0

    init(textBoxes: [TextBox]) {
        self.textBoxes = textBoxes
    }
}
```

TextBoxView.swift

```
import SwiftUI

struct TextBoxView: View {
    @EnvironmentObject var viewModel: ImageEditViewModel
    let textBox: TextBox

    var body: some View {
        // check array empty state first to prevent  "Fatal error: Index out of range"
        Text(!viewModel.creation.textBoxList.textBoxes.isEmpty &&
viewModel.creation.textBoxList.textBoxes[viewModel.creation.textBoxList.currentIndex].id == textBox.id &&
viewModel.addingTextBox ? "" : textBox.text)
            .font(.system(size: 30))
            .fontWeight(textBox.isBold ? .bold : .regular)
            .foregroundColor(textBox.textColor)
            .offset(textBox.offset)
            .gesture(DragGesture()
                .onChanged{ value in
                    let current = value.translation
                    let last = textBox.lastOffset
                    let newTranslation = CGSize(width: last.width + current.width, height: last.height + current.height)
                    viewModel.creation.textBoxList.textBoxes[viewModel.getIndex(of: textBox)].offset = newTranslation
                }
                .onEnded{ value in
                    viewModel.creation.textBoxList.textBoxes[viewModel.getIndex(of: textBox)].lastOffset =
value.translation
                }
            )
            .gesture(LongPressGesture()
                .onEnded{ _ in
                    // Edit text box
                    viewModel.toolPicker.setVisible(false, forFirstResponder: viewModel.canvas)
                    viewModel.canvas.resignFirstResponder()
                    // 之後改abstract可以不用index
                    viewModel.creation.textBoxList.currentIndex = viewModel.getIndex(of: textBox)
                    withAnimation {
                        viewModel.addingTextBox = true
                    }
                }
            )
    }
}
```

Sticker.swift

```
import SwiftUI

struct Sticker: Identifiable {
    var id = UUID()
    var name: String

    var offset: CGSize = .zero
    var lastOffset: CGSize = .zero
    var scale: CGFloat = 1.0

    var rotation: Double = 0.0
```

```
    var opacity: Double = 1.0
}
```

SticerList.swift

```
import Foundation

class StickerList {
    var stickers: [Sticker]
    var editIndex: Int = 0

    init(stickers: [Sticker]) {
        self.stickers = stickers
    }
}
```

StickerView.swift

```
import SwiftUI

struct StickerView: View {
    @EnvironmentObject var viewModel: ImageEditViewModel
    let sticker: Sticker

    var body: some View {
        Image(sticker.name)
            .resizable()
            .scaledToFit()
            .frame(width: 100 * sticker.scale)
            .opacity(sticker.opacity)
            .rotationEffect(.degrees(sticker.rotation))
            .offset(sticker.offset)
            .gesture(DragGesture()
                .onChanged { value in
                    let current = value.translation
                    let last = sticker.lastOffset
                    let newTranslation = CGSize(width: last.width + current.width, height: last.height + current.height)
                    viewModel.creation.stickerList.stickers[viewModel.getIndex(of: sticker)].offset = newTranslation
                }.onEnded { value in
                    viewModel.creation.stickerList.stickers[viewModel.getIndex(of: sticker)].lastOffset = value.translation
                }
            )
            .gesture(LongPressGesture()
                .onEnded{ _ in
                    viewModel.creation.stickerList.editIndex = viewModel.getIndex(of: sticker)
                    viewModel.isShowingEditSheet = true
                }
            )
    }
}

struct StickerView_Previews: PreviewProvider {
    static var previews: some View {
        StickerView(sticker: Sticker(name: "sticker1"))
    }
}
```

Graphic.swift

```swift
import Foundation
import UIKit
import SwiftUI

protocol Graphic {
    var offset: CGSize { get set }
    var lastOffset: CGSize { get set }
}

class TextBox2: Graphic, Identifiable {
    var id: UUID = UUID()
    var text = ""
    var isBold = true
    var offset: CGSize = .zero
    var lastOffset: CGSize = .zero
    var textColor: Color = .white
    var isAdded = false
}
```

Creation.swift

```swift
import Foundation
import UIKit

class Creation: Identifiable {
    var id: UUID
    var name: String
    var image: UIImage?
    var graphics: [Graphic] = []
    var stickerList: StickerList = StickerList(stickers: [])
    var textBoxList: TextBoxList = TextBoxList(textBoxes: [])

    init(id: UUID = UUID(), name: String, image: UIImage?) {
        self.id = id
        self.name = name
        self.image = image
    }

    func addTextBoxes(textBox: TextBox) {
        self.textBoxList.textBoxes.append(textBox)
    }

    func addStickers(sticker: Sticker) {
        self.stickerList.stickers.append(sticker)
    }

    func textBoxesSize() -> Int {
        return textBoxList.textBoxes.count
    }

    func stickersSize() -> Int {
        return stickerList.stickers.count
    }
}
```

LoginViewState.swift

```
import Foundation
@_implementationOnly import FirebaseAuth

enum LoginState {
    case signIn
    case signUp
    case resetPassword
}

struct LoginViewState: Equatable {
    var viewState: LoginState = .signIn
    var username: String = ""
    var email: String = ""
    var password: String = ""
    var confirmPassword: String = ""
    var loginError: AuthErrorCode?
    var passwordError: String?
}
```

CanvasView.swift

```
import SwiftUI
import PencilKit

struct CanvasView: UIViewRepresentable {

    @Binding var canvas: PKCanvasView
    @Binding var toolPicker: PKToolPicker
    @Binding var image: UIImage?

    var rect: CGSize

    func makeUIView(context: Context) -> some UIView {
        canvas.isOpaque = false
        canvas.backgroundColor = .clear
        canvas.drawingPolicy = .anyInput

        if let image = image {

            let imageView = UIImageView(image: image)
            imageView.frame = CGRect(x: 0 , y: 0, width: rect.width, height: rect.height)
            imageView.contentMode = .scaleAspectFit
            imageView.clipsToBounds = false

            // Make the base image the bottom layer of the canvas
            let subView = canvas.subviews[0]
            subView.addSubview(imageView)
            subView.sendSubviewToBack(imageView)

            // Show tool picker and make canvas aware of tool changes
            toolPicker.setVisible(false, forFirstResponder: canvas)
            toolPicker.addObserver(canvas)
            canvas.isUserInteractionEnabled = false
```

```
        }
        return canvas
    }

    func updateUIView(_ uiView: UIViewType, context: Context) {
        var count = 0
        // Update the frame incase GeometryReader changes (e.g. NavigationBar added to the view)
        for subviews in canvas.subviews {
            for subview in subviews.subviews {
                if let imageView = subview as? UIImageView {
                    count += 1
                    imageView.frame = CGRect(x: 0 , y: 0, width: rect.width, height: rect.height)
                }
            }
        }
    }

    func makeCoordinator() -> Coordinator {
        Coordinator(canvasView: $canvas, toolPicker: toolPicker)
    }

    class Coordinator: NSObject, PKCanvasViewDelegate {
        var canvasView: Binding<PKCanvasView>
        private let toolPicker: PKToolPicker
        init(canvasView: Binding<PKCanvasView>, toolPicker: PKToolPicker) {
            self.canvasView = canvasView
            self.toolPicker = toolPicker
        }
        deinit {
            toolPicker.setVisible(false, forFirstResponder: canvasView.wrappedValue)
            toolPicker.removeObserver(canvasView.wrappedValue)
        }
    }
}
```

UserDefaultInfo.swift

```
import Foundation

struct UserDefaultInfo<Value> {
    var key: String
    var defaultValue: Value
}

extension UserDefaultInfo {
    func get() -> Value {
        // 拿不到的話就拿預設的值
        guard let valueUntyped = UserDefaults.standard.object(forKey: self.key) else {
            return self.defaultValue
        }
        guard let value = valueUntyped as? Value else {
            return self.defaultValue
        }
        return value
    }

    func set(_ value: Value) {
        UserDefaults.standard.set(value, forKey: self.key)
    }
```

```
}

extension UserDefaults {
    // 移除裡面自己設的key
    public func removeDefaults() {
        let defaults = UserDefaults.standard
        defaults.dictionaryRepresentation().keys.forEach { key in
            defaults.removeObject(forKey: key)
        }
    }
}
```

FireBaseService.swift

```
import Foundation
@_implementationOnly import FirebaseAuth
@_implementationOnly import FirebaseFirestore

class FirebaseService {
    let firebaseAuth = FirebaseAuth.Auth.auth()
    let firebaseFirestore = FirebaseFirestore.Firestore.firestore()

    init() {

    }

    func getCurrentUser() -> User? {
        return firebaseAuth.currentUser
    }

    func signIn(email: String, password: String, result: @escaping (_ error: AuthErrorCode?, _ data: String?) ->
Void) {
        firebaseAuth.signIn(withEmail: email, password: password) { authResult, error in
            if let error = error {
                if let errCode = AuthErrorCode(rawValue: error._code) {
                    result(errCode, nil)
                    return
                }
            }
            if let authResult = authResult {
                if !authResult.user.isEmailVerified {
                    result(nil, "Email is not verified.")
                    return
                }
            }
            result(nil, "login success")
        }
    }

    func signUp(email: String, password: String) {
        firebaseAuth.createUser(withEmail: email, password: password) { authResult, error in
            if error != nil {
                print(error!)
```

```
                return
            }

            authResult?.user.sendEmailVerification(completion: { error in
                if error != nil {
                    print(error!)
                }
            })

        }
    }

    func signOut() {
        do {
            try firebaseAuth.signOut()
        }
        catch let signOutError as NSError {
            print("Error signing out: %@", signOutError)
        }
    }
}
```

LoginViewModel.swift

```
import Foundation
import SwiftUI
@_implementationOnly import FirebaseAuth

final class LoginViewModel: StateBindingViewModel<LoginViewState> {
    let firebaseService: FirebaseService = FirebaseService()
    @Binding var isPresented: Bool

    init(initialState: LoginViewState, isPresented: Binding<Bool>) {
        self._isPresented = isPresented
        super.init(initialState: initialState)
    }

    func signIn() {
        firebaseService.signIn(email: state.email, password: state.password) { errorCode, data in
            if let errorCode = errorCode {
                self.state.loginError = errorCode
                return
            }
            print(data ?? "nil")
            self.isPresented = false
        }
    }

    func signUp() {
        firebaseService.signUp(email: state.email, password: state.password)
    }

    func resetPassword() {
        print(state.email)
    }
```

```
}
```

ImageEditViewModel.swift

```
import SwiftUI
import PencilKit

class ImageEditViewModel: ObservableObject {
    @Published var canvas = PKCanvasView()
    @Published var toolPicker = PKToolPicker()
    @Published var isDrawing = false

    @Published var addingTextBox = false

    // For saving image
    @Published var rect: CGRect = .zero

    @Published var showingAlert = false
    @Published var alertTitle = ""
    @Published var alertMessage = ""

    @Published var creation: Creation

    init(creation: Creation) {
        self.creation = creation
    }


    @Published var isShowingStickerSheet = false
//    @Published var currentStickerName = ""

    @Published var isShowingEditSheet = false

    var showingFeaturesToolBar: Bool {
        return !addingTextBox
    }

    func startDrawing() {
        // show tool picker
        toolPicker.setVisible(true, forFirstResponder: canvas)
        canvas.becomeFirstResponder()
        canvas.isUserInteractionEnabled = true
        isDrawing = true
    }

    func doneDrawing() {
        // Close the tool picker
        toolPicker.setVisible(false, forFirstResponder: canvas)
        canvas.resignFirstResponder()
        canvas.isUserInteractionEnabled = false
        isDrawing = false
    }

    func createTextBox() {
        guard !addingTextBox else { return }
```

```
    // Create a new text box
    creation.textBoxList.textBoxes.append(TextBox())
    creation.textBoxList.currentIndex = creation.textBoxList.textBoxes.count - 1
    withAnimation {
        addingTextBox.toggle()
    }
}

func addTextBox() {
    creation.textBoxList.textBoxes[creation.textBoxList.currentIndex].isAdded = true
    addingTextBox = false
}

func cancelTextBoxView() {
    withAnimation {
        addingTextBox = false
    }
    if !creation.textBoxList.textBoxes[creation.textBoxList.currentIndex].isAdded {
        creation.textBoxList.textBoxes.removeLast()
    } else {
        creation.textBoxList.textBoxes.remove(at: creation.textBoxList.currentIndex)
    }
    creation.textBoxList.currentIndex = 0
}

func getIndex(of textBox: TextBox) -> Int {
    creation.textBoxList.textBoxes.firstIndex { box -> Bool in
        textBox.id == box.id
    } ?? 0
}

func getIndex(of sticker: Sticker) -> Int {
    creation.stickerList.stickers.firstIndex { s -> Bool in
        sticker.id == s.id
    } ?? 0
}

func addSticker(_ sticker: Sticker) {
    creation.stickerList.stickers.append(sticker)
}

// used in sticker edit view
func removeSticker(_ sticker: Sticker) {
    isShowingEditSheet = false
    creation.stickerList.stickers.remove(at: creation.stickerList.editIndex)
    creation.stickerList.editIndex = 0
}

func saveImage() {
    UIGraphicsBeginImageContextWithOptions(rect.size, false, 0)

    //canvas
    canvas.drawHierarchy(in: CGRect(origin: .zero, size: rect.size), afterScreenUpdates: true)

    //SwiftUI
    let swiftUIView = ZStack {
```

```
            ForEach(creation.textBoxList.textBoxes) { textBox in
                Text(textBox.text)
                    .font(.system(size: 30))
                    .fontWeight(textBox.isBold ? .bold : .regular)
                    .foregroundColor(textBox.textColor)
                    .offset(textBox.offset)
            }
            ForEach(creation.stickerList.stickers) { sticker in
                Image(sticker.name)
                    .resizable()
                    .scaledToFit()
                    .frame(width: 100 * sticker.scale)
                    .opacity(sticker.opacity)
                    .rotationEffect(.degrees(sticker.rotation))
                    .offset(sticker.offset)
            }
        }

        let controller = UIHostingController(rootView: swiftUIView).view!
        controller.frame = rect


        controller.backgroundColor = .clear
        canvas.backgroundColor = .clear


        controller.drawHierarchy(in: CGRect(origin: .zero, size: rect.size), afterScreenUpdates: true)

        // getting image
        let generatedImage = UIGraphicsGetImageFromCurrentImageContext()

        UIGraphicsEndImageContext()

        guard let image = generatedImage?.pngData() else { print("Error: Can't generate image"); return }
        UIImageWriteToSavedPhotosAlbum(UIImage(data: image)!, nil, nil, nil)
        alertTitle = "Enjoy!"
        alertMessage = "Image saved to your photo library"
        showingAlert = true
    }
}
```

UserViewModel

```
import Foundation
@_implementationOnly import FirebaseAuth

enum LoginSelect {
    case signIn
    case signUp
}

final class UserViewModel: ObservableObject {
    @Published var showLoginView = false
    @Published var currentUser: User?
    let firebaseService = FirebaseService()
    var signInOrSignUp: LoginSelect = .signIn
```

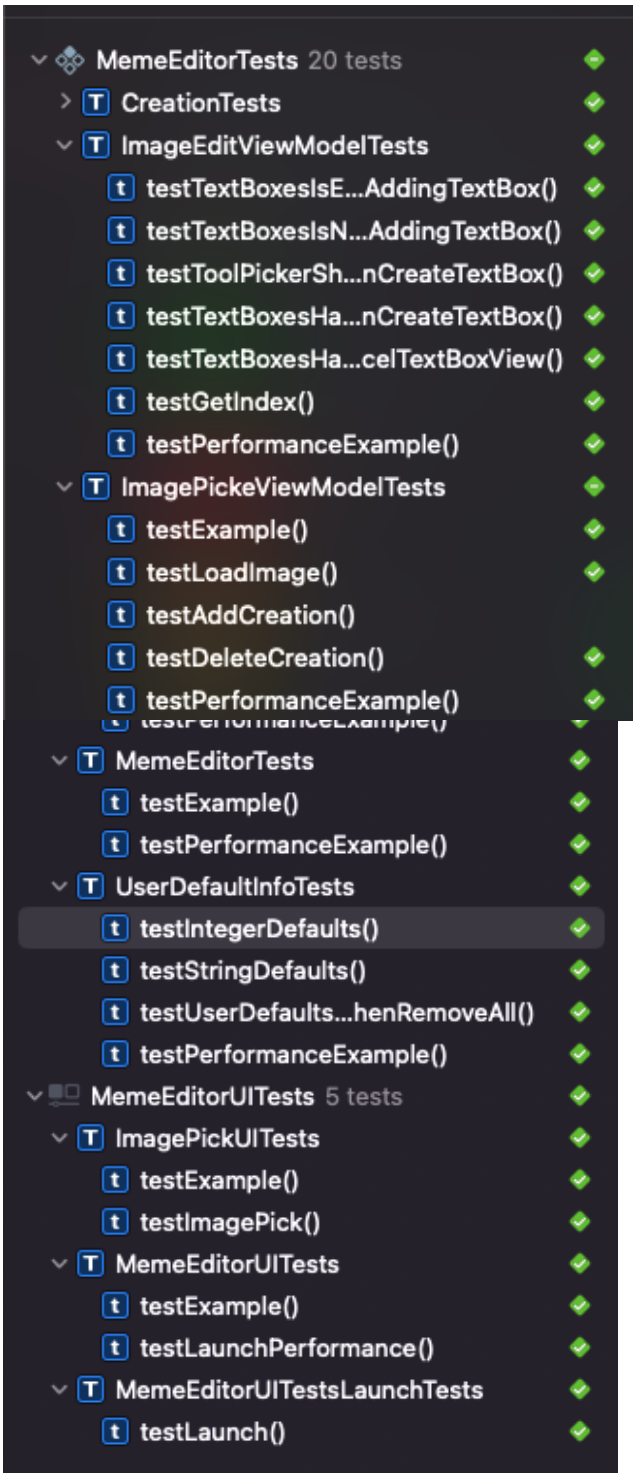
```
    init() {
        firebaseService.firebaseAuth.addStateDidChangeListener { auth, user in
            self.currentUser = user
        }
    }

    func getCurrentUser() -> User? {
        return firebaseService.getCurrentUser()
    }

    func signOut() {
        firebaseService.signOut()
    }
}
```

Unit Testing

6.1 Snapshots of testing result



6.2 Unit Test Code Listing

```swift
class ImagePickeViewModelTests: XCTestCase {
    private var viewModel: ImagePickViewModel!

    override func setUpWithError() throws {
        // Put setup code here. This method is called before the invocation of each test method in the class.
        viewModel = ImagePickViewModel()
    }

    func testLoadImage() {
        viewModel.inputImage = UIImage(named: "hello-peter")
    }

    func testAddCreation() {
        viewModel.addCreation(creation: Creation(name: "test", image: UIImage(named: "hello-peter")))
        viewModel.addCreation(creation: Creation(name: "test", image: UIImage(named: "hello-peter")))
        XCTAssertEqual(viewModel.creations.count, 2)
    }

    func testDeleteCreation() {
        viewModel.addCreation(creation: Creation(name: "test", image: UIImage(named: "hello-peter")))
        viewModel.addCreation(creation: Creation(name: "test", image: UIImage(named: "hello-peter")))
        viewModel.deleteCreation(at: .init(integer: 1))
        XCTAssertEqual(viewModel.creations.count, 1)
    }


}
```


```swift
class ImagePickeViewModelTests: XCTestCase {
    private var viewModel: ImagePickViewModel!

    override func setUpWithError() throws {
        // Put setup code here. This method is called before the invocation of each test method in the class.
        viewModel = ImagePickViewModel()
    }

    func testLoadImage() {
        viewModel.inputImage = UIImage(named: "hello-peter")
    }

    func testAddCreation() {
        viewModel.addCreation(creation: Creation(name: "test", image: UIImage(named: "hello-peter")))
        viewModel.addCreation(creation: Creation(name: "test", image: UIImage(named: "hello-peter")))
        XCTAssertEqual(viewModel.creations.count, 2)
    }

    func testDeleteCreation() {
        viewModel.addCreation(creation: Creation(name: "test", image: UIImage(named: "hello-peter")))
        viewModel.addCreation(creation: Creation(name: "test", image: UIImage(named: "hello-peter")))
        viewModel.deleteCreation(at: .init(integer: 1))
        XCTAssertEqual(viewModel.creations.count, 1)
    }


}
```

```swift
class UserDefaultInfoTests: XCTestCase {

    override func setUpWithError() throws {
        UserDefaults.standard.removeDefaults()
    }

    override func tearDownWithError() throws {
        UserDefaults.standard.removeDefaults()
    }

    func testIntegerDefaults() {
        let integerInfo = UserDefaultInfo(key: "integerInfo", defaultValue: 12345)
        XCTAssertEqual(integerInfo.get(), 12345)
        integerInfo.set(54321)
        XCTAssertEqual(integerInfo.get(), 54321)
        integerInfo.set(9876)
        XCTAssertNotEqual(integerInfo.get(), 0)
    }

    func testStringDefaults() {
        let integerInfo = UserDefaultInfo(key: "integerInfo", defaultValue: "12345")
        XCTAssertEqual(integerInfo.get(), "12345")
        integerInfo.set("54321")
        XCTAssertEqual(integerInfo.get(), "54321")
        integerInfo.set("9876")
        XCTAssertNotEqual(integerInfo.get(), "0")
    }

    func testUserDefaultsShouldBeEmptyWhenRemoveAll() {
        let integerInfo = UserDefaultInfo(key: "integerInfo", defaultValue: 12345)
        let stringInfo = UserDefaultInfo(key: "stringInfo", defaultValue: "12345")
        var integer: Int {
            get { return integerInfo.get() }
            set { return integerInfo.set(newValue) }
        }
        var string: String {
            get { return stringInfo.get() }
            set { return stringInfo.set(newValue) }
        }
        integer = 123
        string = "123"
        UserDefaults.standard.removeDefaults()
        // should be default value after remove
        XCTAssertEqual(integer, 12345)
        XCTAssertEqual(string, "12345")
    }


}

```

```swift
//
//  ImageEditViewModelTests.swift
//  MemeEditorTests
//
//  Created by 范桶 on 2022/5/4.
//

import XCTest
@testable import MemeEditor


class ImageEditViewModelTests: XCTestCase {
    private var viewModel: ImageEditViewModel!
    private var canvasView: CanvasView!


    override func setUpWithError() throws {
        // Put setup code here. This method is called before the invocation of each test method in the class.
        viewModel = ImageEditViewModel(creation: Creation.sampleData[0])
        canvasView = CanvasView(canvas: .constant(viewModel.canvas), toolPicker: .constant(viewModel.toolPicker), image: .constant(UIImage(named: "hello-peter")), rect: CGSize())
    }

    override func tearDownWithError() throws {
        // Put teardown code here. This method is called after the invocation of each test method in the class.
    }

    func testTextBoxesIsEmptyWhenCreateTextBoxIsAddingTextBox() {
        viewModel.addingTextBox = true;
        viewModel.createTextBox()
        XCTAssertTrue(viewModel.creation.textBoxList.textBoxes.isEmpty)
    }

    func testTextBoxesIsNotEmptyWhenCreateTextBoxNotAddingTextBox() {
        viewModel.createTextBox()
        XCTAssertFalse(viewModel.creation.textBoxList.textBoxes.isEmpty)
    }

    func testToolPickerShouldBeInVisibleWhenCreateTextBox() {
        viewModel.createTextBox()
        XCTAssertFalse(viewModel.toolPicker.isVisible)
    }

    func testTextBoxesHaveCorrectLengthWhenCreateTextBox() {
        viewModel.createTextBox()
        viewModel.addTextBox()
        viewModel.createTextBox()
        viewModel.addTextBox()
        viewModel.createTextBox()
        viewModel.addTextBox()
        viewModel.createTextBox()
        viewModel.addTextBox()
        XCTAssertEqual(viewModel.creation.textBoxList.textBoxes.count, 4)
    }

    func testTextBoxesHaveCorrectLengthWhenCreateTextBoxAndCancelTextBoxView() {
        viewModel.createTextBox()
        viewModel.addTextBox()
        viewModel.createTextBox()
        viewModel.addTextBox()
        viewModel.createTextBox()
        viewModel.addTextBox()
        viewModel.createTextBox()
        viewModel.cancelTextBoxView()
        XCTAssertEqual(viewModel.creation.textBoxList.textBoxes.count, 3)
    }

    func testGetIndex() {
        let textBox = TextBox()
        viewModel.creation.textBoxList.textBoxes.append(TextBox())
        viewModel.creation.textBoxList.textBoxes.append(textBox)
        viewModel.creation.textBoxList.textBoxes.append(TextBox())
        viewModel.creation.textBoxList.textBoxes.append(TextBox())
        XCTAssertEqual(viewModel.getIndex(of: textBox), 1)
    }

}

```

Measurement

107590051 范凱翔	107590058 陳小蘭	107590450 劉學逸	107A50039 黃發泉（已撤選）
HW#1			
22/03/01: 21:00~23:00	22/03/01: 21:00~23:00	22/03/01: 21:00~23:00	22/03/01: 21:00~23:00
total: 2hrs	total: 2hrs	total: 2hrs	total: 2hrs
HW#2			
22/03/07 16:00~20:00	22/03/14 14:00~15:00	22/03/11 14:00~16:00	22/03/13 22:00~23:00
22/03/13 22:00~23:00	22/03/14 19:00~23:00	22/03/13 12:20~17:30	22/03/14 19:00~21:00
22/03/14 19:00~23:00	22/03/15 11:00~12:00	22/03/13 19:30~23:00	22/03/14 22:00~23:00
		22/03/16 18:40~20:20	22/03/16 19:30~20:30
total: 9 hrs	total: 6 hrs	total: 12 hrs	total: 5 hrs
HW#3			
22/03/19 20:00~24:00		22/03/18 16:30~18:30	
22/03/20 0:00~3:00		22/03/18 20:00~23:59	
22/03/20 14:00~20:00		22/03/19 12:20~15:40	
22/03/21 14:00~18:00		22/03/19 19:00~02:50	
22/03/29 20:30~23:30	22/03/29 20:30~23:30	22/03/29 20:30~23:30	22/03/29 20:30~23:30
total: 20 hrs	total: 3 hrs	total: 20 hrs	total: 3 hrs

HW#4			
22/03/31 7:30~8:00 (implement project)	22/04/19 12:30~14:30 (write report diagrams)	22/04/10 18:00~18:30 (refactor and discuss)	
22/04/11 2:00~3:00 (implement project)	22/04/19 16:00~18:30 (write report diagrams)	22/04/11 16:00~19:30 (refactor and bug fix)	
22/04/11 13:00~15:00 (implement project)	22/04/19 21:15~23:15 (discuss and write report diagrams)	22/04/19 20:00~23:15 (discuss and write report diagrams)	
22/04/14 12:30~13:00 (implement project)	22/04/20 13:30~16:00 (write report diagrams)	22/04/20 20:00~25:00 (discuss and write report diagrams)	
22/04/14 19:00~24:00 (implement project)	22/04/20 16:30~18:00 (write report diagrams)		
22/04/17 17:00~17:30 (implement project)	22/04/20 21:30~25:00 (discuss and write report diagrams)		
22/04/17 21:00~24:30 (implement project)			
22/04/19 21:00~23:00 (discuss and write report diagrams)			
22/04/20 21:30~25:00 (discuss and write report diagrams)			
total: 19.5 hrs	total: 14 hrs	total: 11 hrs	撤選
HW#5			
22/04/23 15:00~18:00	22/04/29 23:00~27:00	22/04/28 18:00~20:30	
22/04/23 18:30~20:00	22/05/01 23:00~25:00	22/05/01 14:00~16:00	
22/04/23 22:00~23:00	22/05/03 22:00~25:30	22/05/02 22:00~26:00	
22/04/24 0:00~3:00	22/05/04 20:00~25:00	22/05/03 22:30~25:00	
22/04/24 14:00~16:00		22/05/04 20:00~25:00	

22/04/24 19:00~23:00			
22/04/25 15:00~18:00			
22/04/25 19:00~23:00			
22/04/30 19:00~23:00			
22/05/01 19:00~23:00			
22/05/03 06:30~8:30			
22/05/04 06:30~8:30			
22/05/04 20:00~25:00			
total: 38.5 hrs	total: 14.5 hrs	total: 17 hrs	
HW#6			
22/05/25 6:00~8:00	22/05/17 15:00~19:00	22/05/21 15:00~18:00	
	22/05/24 13:00~15:00	22/05/22 21:00~23:30	
total: 2 hrs	total: 6 hrs	total: 5.5 hrs	
HW#7			
22/06/04 11:00~12:30	22/06/07 13:00~16:00	22/06/07 21:00~22:40	
22/06/07 21:00~22:40	22/06/07 21:00~23:00	22/06/10 14:00~18:00	
22/06/11 15:00~18:00	22/06/15 18:30~27:00	22/06/14 20:00~23:00	
22/06/12 15:00~18:00		22/06/15 18:30~27:00	

22/06/13 7:30~20:00			
22/06/14 20:00~24:30			
22/06/15 19:00~27:00			
total: 31 hrs	total: 13.5 hrs	total: 18 hrs	