國立臺灣大學電機資訊學院電機工程學研究所

碩士論文

Graduate Institute of Communication Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master’s Thesis

QUX : 問卷基底 GUI 模型之軟體架構

QUX : A Software Framework for Questionnaire-Based GUI Models

易行祐

HSING-YU YI

指導教授：王 凡 博士

Advisor: Farn Wang, Ph.D.

中華民國 113 年 7 月

July, 2024

doi:10.6342/NTU202402814

誌謝

能完成這篇論文，特別感謝我的父母，也感謝所有一路上曾經給予我支持與鼓勵的人，謝謝大家。

doi:10.6342/NTU202402814

中文摘要

在現今的社會，手機應用程式已經十分普及，人們透過這些程式來協助完成他們各式各樣的任務，像是時間管理、購物、通訊等等。但跟電腦應用程式不同的是，手機應用程式的開發通常要同時面對時間上的壓力和廣大同類型應用的競爭。此時，除了效率和創新以外，另一個能令其脫穎而出的關鍵就是 GUI 的設計。但 UI 設計是個很耗時的工程，需要不斷根據反饋和測試結果進行調整，確保最終的設計能夠滿足用戶的期望。正因如此，我們希望能找出方法，為 UI 開發過程提供有效的自動化支持。在本論文中，我們利用問卷蒐集回答作為模型訓練用的標籤，進一步結合機器學習的技術去做出一個能夠針對給定的 GUI，自動回答問卷問題的模型。我們希望能透過這個模型自動化地檢測和識別 UI 設計中的問題，並向開發者提供建議，進而提高整個 UI 開發過程的效率，使設計師和開發者能夠更好地集中精力在創造性的方面，而不是繁瑣的錯誤修復上。

關鍵字：手機應用程式、使用者介面設計、設計準則、問卷、自動回饋

doi:10.6342/NTU202402814

ABSTRACT

In today's society, mobile applications have become ubiquitous, assisting people in various tasks such as time management, shopping, and communication. However, unlike computer applications, the development of mobile applications typically involves time constraints and fierce competition among similar apps. In addition to efficiency and innovation, another key factor that distinguishes an app is the design of its Graphical User Interface (GUI). UI design is a time-consuming process that requires continuous adjustments based on feedback and testing to ensure the final design meets user expectations. Therefore, we aim to find methods to provide effective automation support for the UI development process.

In this thesis, we use questionnaires to collect responses as labels for model training. We further integrate machine learning techniques to develop a model capable of automatically answering questionnaire questions for a given GUI. Our goal is to automate the detection and identification of issues in UI design, providing suggestions to developers. This approach aims to enhance the overall efficiency of the UI development process, allowing designers and developers to focus more on creative aspects rather than tedious error fixes.

Keywords : Mobile apps, UI Design, Design Guidelines, Automated Support, Survey, Questionnaire-Based Model

doi:10.6342/NTU202402814

CONTENTS

口試委員會審定書 .... #

誌謝 i

中文摘要 ii

ABSTRACT iii

CONTENTS iv

LIST OF FIGURES vi

LIST OF TABLES viii

Chapter 1 Introduction 1

1.1 Background 1

1.2 Motivation 2

1.3 Contributions 3

Chapter 2 Related Work and Preliminaries 5

2.1 Academic Research 5

2.2 Figma 6

2.3 UI Automator 6

Chapter 3 Methodology 8

3.1 Global Picture 8

3.2 Building UI Generating Algorithm 10

3.2.1 UI Generating Algorithm 12

3.2.2 Mutation of UI 14

3.3 Survey Design and Feedback Collection 15

3.3.1 Questionnaire Questions 16

doi:10.6342/NTU202402814

3.4 Questionnaire-based Model 23

3.4.1 Model Architecture and Training 23

3.4.2 Dataset 25

Chapter 4 Implementations and Evaluations 26

4.1 Programming Language and Environment Setup 26

4.2 Model Performance on Training & Testing 26

4.3 Testing on Unlabeled Real-World Data 30

Chapter 5 Conclusion and Future Work 36

REFERENCE 38

doi:10.6342/NTU202402814

LIST OF FIGURES

Figure 1.1 Android Material Design – Design Guidelines Example 2

Figure 3.1 Material 3 Design Kit 6

Figure 3.1 Flowchart of Global Picture...8

Figure 3.2 Process of Building UI Generating Algorithm 10

Figure 3.3 Flowchart of UI Generating Algorithm 12

Figure 3.4 Example of UI Generation...13

Figure 3.5 Survey Respondents Distribution 16

Figure 3.6 Question 1..17

Figure 3.7 Question 2..17

Figure 3.8 Question 3..18

Figure 3.9 Question 4..18

Figure 3.10 Question 5..18

Figure 3.11 Question 6&7...19

Figure 3.12 Question 8&9...20

Figure 3.13 Question 10&11...20

Figure 3.14 Question 12&13...21

Figure 3.15 Question 14..21

Figure 3.16 Construction of Questionnaire-based Model..23

Figure 4.1 Choice Questions Answering Performance 30

Figure 4.2 Rating Questions Answering Performance..30

Figure 4.3 Question Related to Class Classification...32

Figure 4.4 3 Types of Navigation Bar..32

doi:10.6342/NTU202402814

Figure 4.5 Question Related to Finding Missing Items 33

Figure 4.6 Model Rating and Play Store Rating 34

doi:10.6342/NTU202402814

LIST OF TABLES

Table 3.1 List of Mutation Types 15

Table 3.2 Correlation between number of mutation and rating problem 22

Table 4.1 Training Evaluation 27

Table 4.2 Testing Evaluation 27

Table 4.3 Training Evaluation of the Simplified Models with Different Sizes 28

Table 4.4 Testing Evaluation of the Simplified Models with Different Sizes 28

Table 4.5 Model Performance with and without Data Preprocessing 29

Table 4.6 List of Mobile Applications 31

Table 4.7 Evaluation of Model Performance on Real-World APP 34

doi:10.6342/NTU202402814

Chapter 1 Introduction

1.1 Background

In the modern era, mobile applications have become the primary gateway to the Internet and essential tools for daily tasks like reading, shopping, banking, and communication. However, unlike traditional desktop applications, mobile apps face fierce competition and time-to-market pressures, with over 3.8 million Android and 2 million iPhone apps vying for users.

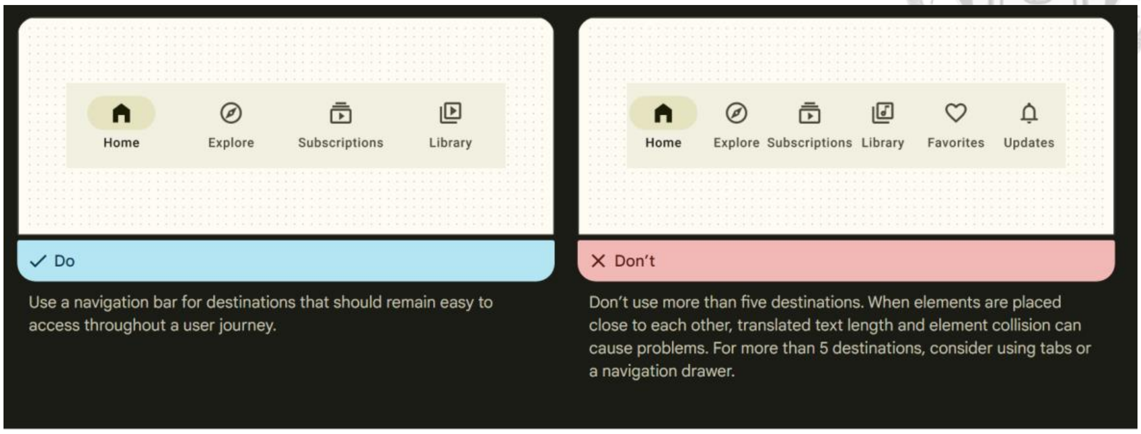
A crucial aspect of mobile app success lies in the Graphical User Interface (GUI), facilitating user-software interaction. A poorly designed GUI can lead to user frustration and app uninstallation. Successful mobile apps require an intuitive, elegant GUI aligned with effective User Interface (UI) design guidelines.

In order to build a user-friendly and aesthetic GUI, we can utilize the concept of "mental model" to understanding how individuals perceive system functionality based on prior knowledge. Several methods are employed to determine users' existing mental models, including Jakob’s Law, card sorting, surveys, and user reviews/interviews. In addition, following the UI design principles could be helpful. Mobile platforms, such as Android Material Design[1] and iOS Human Interface Guidelines, set specific GUI design standards that mobile applications running on these platforms are expected to follow.

Building a successful UI/UX for mobile apps is a complex, iterative process compounded by challenges inherent in mobile app development. These challenges include the continuous pressure for frequent releases, the necessity to promptly address user reviews to enhance app quality, frequent platform updates, API instability, and the demand for custom components. The intricate nature of mobile app UI/UX design

doi:10.6342/NTU202402814

underscores the need for efficient processes and tools to navigate these challenges successfully.



Use a navigation bar for destinations that should remain easy to access throughout a user journey.

Don't use more than five destinations. When elements are placed close to each other, translated text length and element collision can cause problems. For more than 5 destinations, consider using tabs or a navigation drawer.

Figure 1.1 Android Material Design – Design Guidelines Example[1]

1.2 Motivation

Implementing an intuitive and visually appealing Graphical User Interface (GUI) is widely recognized as a challenging task, given its complexity and high associated costs. The practical need for effective automated support becomes apparent to enhance the process of detecting and reporting design violations, providing developers with more accurate and actionable information.

Previous research has shown limited concern for detecting visual design violations, creating a gap in addressing crucial aspects of GUI design. While most GUI testing methodologies[2][3][4] dynamically explore the behaviors of an application, they primarily focus on simulating user interactions to trigger app functionalities. Their evaluation is centered on code and GUI coverage, lacking the ability to validate the visual effects of GUI designs.

[1] https://m3.material.io/components/navigation-bar/guidelines

doi:10.6342/NTU202402814

In response to these challenges, some research initiatives[6][8] have introduced design guideline-based testing techniques. These techniques offer direct feedback to developers and UI designers, providing a more straightforward approach. However, these methods deviate from the typical feedback format, such as questionnaires, commonly received by developers and UI designers. This disparity highlights the need for a different approach to address design violations and provide actionable insights in a format familiar to development teams.

1.3 Contributions

1. Proposed questionnaire-based software framework QUX:

We contribute to the field by introducing a questionnaire-based model designed to reduce the cost associated with analyzing the mental models of app users. This model leverages machine learning techniques with the goal of providing automated suggestions for GUI design improvements to app developers. We named this framework QUX.

2. Synthetic GUI Datasets:

To facilitate the training of our proposed model, we have built datasets of synthetic GUIs. This was achieved through the utilization of a self-developed UI generation algorithm, contributing a resource for training and testing the effectiveness of our approach.

3. Pioneering Use of Questionnaire-Based Model in UI Development:

As far as our knowledge extends, our work represents the first instance of employing a questionnaire-based model as an automated tool in the realm of UI development. This innovation is poised to streamline the process of gathering user insights and translating

doi:10.6342/NTU202402814

them into actionable design recommendations.

4. Construction of UI Design Guidelines Knowledge Base:

To enhance the effectiveness of our questionnaire-based model, we have constructed a knowledge base of UI design guidelines. This knowledge base serves as a crucial reference point, aiding in the formulation of questionnaire questions that align with established principles in UI design.

Overall, our contributions aim to advance the field of UI development by offering a comprehensive solution that combines user insights and machine learning to streamline the GUI design process for app developers.

doi:10.6342/NTU202402814

Chapter 2 Related Work and Preliminaries

2.1 Academic Research

B. Yang, Z. Xing, X. Xia, C. Chen, D. Ye, and S. Li propose a multi-dimensional analysis method for detecting design guideline violations[8]. Their approach involves automatically parsing UI screenshots or prototype files to extract component metadata. This metadata is then matched with guideline violation conditions, offering a comprehensive method for identifying design guideline violations.

Dehai Zhao, Zhenchang Xing, Chunyang Chen, Xiwei Xu, Liming Zhu, Guoqiang Li, and Jinshui Wang present an unsupervised deep learning method specifically for detecting GUI animation design guideline violations[6]. The method treats the validation against design-don't guidelines as a multi-class classification problem. Initially, a GUI animation feature extractor is trained to autonomously extract crucial temporal-spatial features from GUI animations in an unsupervised way. The trained feature extractor is then employed to map typical GUI animations that violate design guidelines into a dense vector space. For a given GUI animation, it is also mapped into this vector space. The violation is determined by conducting a KNN search to identify the most similar violating GUI animations.

These related works showcase innovative approaches to detecting design guideline violations, one focusing on multi-dimensional analysis through UI screenshots and prototype files[8], and the other employing unsupervised deep learning for GUI animation guideline violations[6]. Each method contributes to the diverse range of techniques aimed at enhancing the automated detection of design guideline violations in user interfaces.

doi:10.6342/NTU202402814

2.2 Figma

Figma stands as a collaborative interface design tool, commanding a significant market share of 36.17% in the collaborative design and prototyping market. Within this platform, Google has developed the Material 3 Design[1] Kit, offering components that align with official design guidelines. These components, including buttons, dialogs, navigation bars, etc., serve as the foundation for our UI Generating Algorithm.

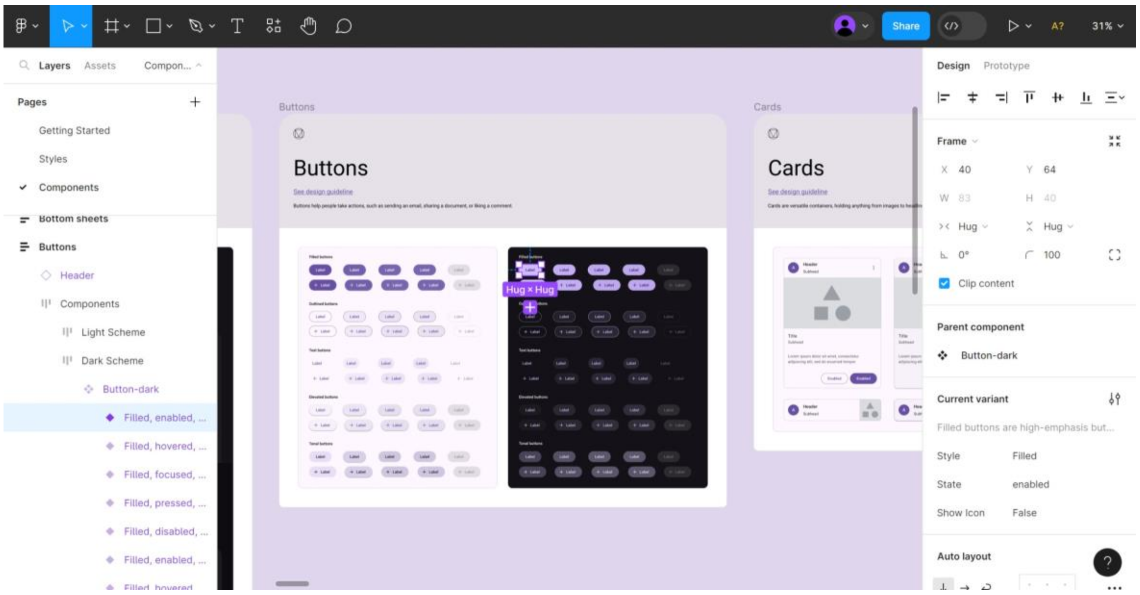


Figure 2.1 Material 3 Design Kit

2.3 UI Automator

UI Automator is a UI testing framework designed for cross-app functional UI testing across both system and installed apps. Its APIs enable interaction with visible elements on a device, irrespective of the focused activity. This flexibility allows operations such as opening the Settings menu or the app launcher in a test device.

In the context of UI Automator, it serves as a tool for retrieving UI Layout Files, essential for our model's training data. These files contain the hierarchical structure of the UI, component coordinates, text, and other pertinent information. Utilizing UI

doi:10.6342/NTU202402814

Automator simplifies the process of obtaining critical data for training our model.

doi:10.6342/NTU202402814

Chapter 3 Methodology

3.1 Global Picture of QUX

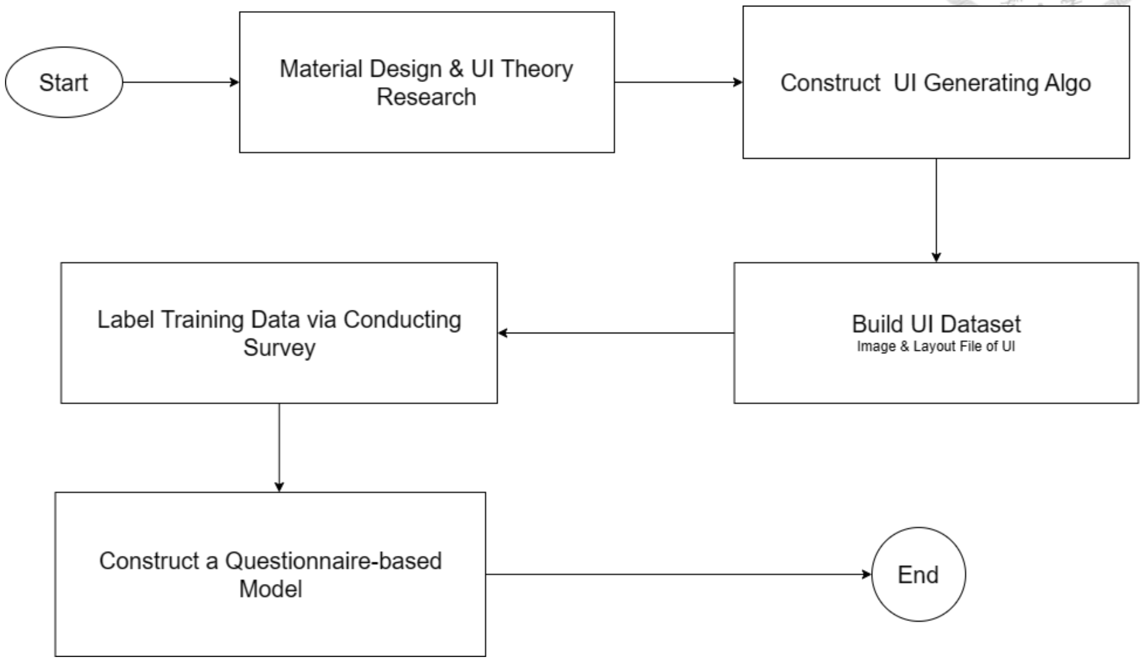

Start
Material Design & UI Theory Research
Construct UI Generating Algo
Build UI Dataset
Image & Layout File of UI
Label Training Data via Conducting Survey
Construct a Questionnaire-based Model
End

Figure 3.1 Flowchart of Global Picture

As shown in Figure 4.1, this flowchart represents the core steps outlined in the thesis and Software Framework of QUX. The process begins with an in-depth study of Material Design, Google's UI design guidelines, and relevant UI theories. Drawing inspiration from these guidelines, we formulate the methodology for designing questionnaire questions and developing the UI generation algorithm.

The first pivotal step involves the construction of the UI generation algorithm. This algorithm is designed to generate UI based on the established guidelines. Subsequently, we utilize this algorithm to create a dataset comprising UI images paired with their corresponding layout files in XML format. The inclusion of layout files in the dataset is significant, as these files serve as additional training data for the subsequent stages.

Following the dataset creation, we conduct a survey to collect user feedback on the

doi:10.6342/NTU202402814

generated UIs. The gathered feedback from the survey is then utilized as labels for the dataset.

In the final phase, armed with the dataset and corresponding labels, we employ machine learning techniques to train a model capable of automatically answering the questionnaire questions for a given GUI. This model represents a crucial component in our proposed methodology, aiming to streamline the process of UI evaluation and feedback collection.

In summary, the global methodology involves research on design guidelines, the formulation of questionnaire questions, and the development of a UI generation algorithm. The subsequent steps encompass dataset creation, user feedback collection through surveys, and the training of a machine learning model to automate the questionnaire process for GUI evaluation. This comprehensive approach is geared towards enhancing the efficiency and effectiveness of UI design and evaluation in accordance with established principles.

doi:10.6342/NTU202402814

3.2 Building UI Generating Algorithm

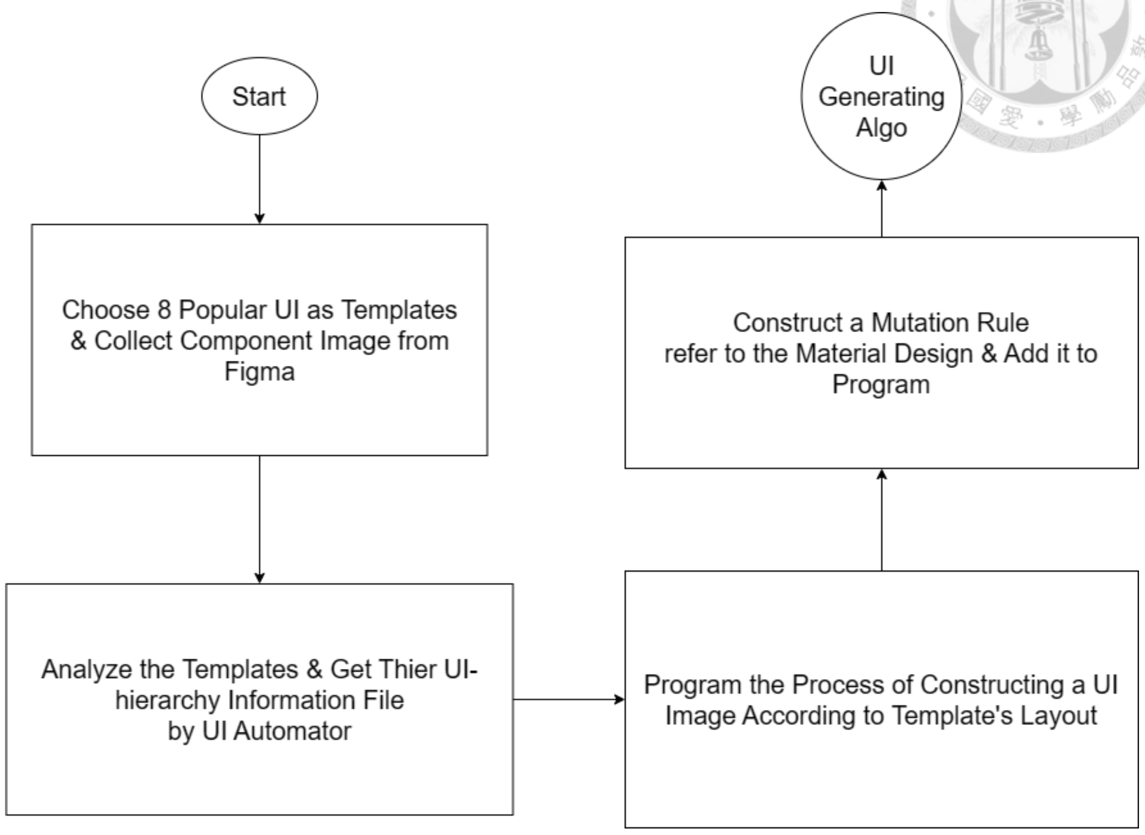

Start
Choose 8 Popular UI as Templates & Collect Component Image from Figma
Analyze the Templates & Get Thier UI-hierarchy Information File by UI Automator
Program the Process of Constructing a UI Image According to Template's Layout
Construct a Mutation Rule refer to the Material Design & Add it to Program
UI Generating Algo

Figure 3.2 Process of Building UI Generating Algorithm

The foundation of our research lies in the development of a UI generation algorithm, crucial for training our model and creating a diverse dataset. The following flowchart illustrates the conceptualization and step-by-step construction of this algorithm.

1. Selection of UI Templates:

We initiated the algorithm by carefully selecting eight UI templates from popular mobile applications. These templates serve as the algorithm's foundational building blocks. Additionally, we curated a library of various component images from Figma, which would later be utilized to assemble UIs.

doi:10.6342/NTU202402814

2. UI Analysis using UI Automator:

Employing UI Automator, we conducted a detailed analysis of the chosen templates, extracting their UI layout files. These layout files, serving as templates themselves, are integral to our algorithm's design. The algorithm generates both UI images and their corresponding layout files.

3. Automated UI Construction:

The next step involved the development of a program capable of automatically selecting components from the library based on the chosen template's layout. By referencing the arrangement and positions of components in the template, this program constructs a UI image that mirrors the layout of the chosen template.

4. Introduction of Design Guideline-Informed Mutation:

In the final stage, we incorporated design guideline-informed mutations to introduce variability. Drawing inspiration from design guidelines, we determined the types of mutations, such as changes in color or component positions. These mutations, added in a randomized manner, bring about design variations, ensuring that the generated UIs exhibit diversity and creativity.

In essence, this algorithm is rooted in a fixed template layout, augmented with random variations to create a UI generation process that produces both UI images and their corresponding layout files. The ensuing sections will delve into the specifics of each step, providing a comprehensive understanding of our approach to UI generation.

doi:10.6342/NTU202402814

3.2.1 QUX - UI Generating Algorithm

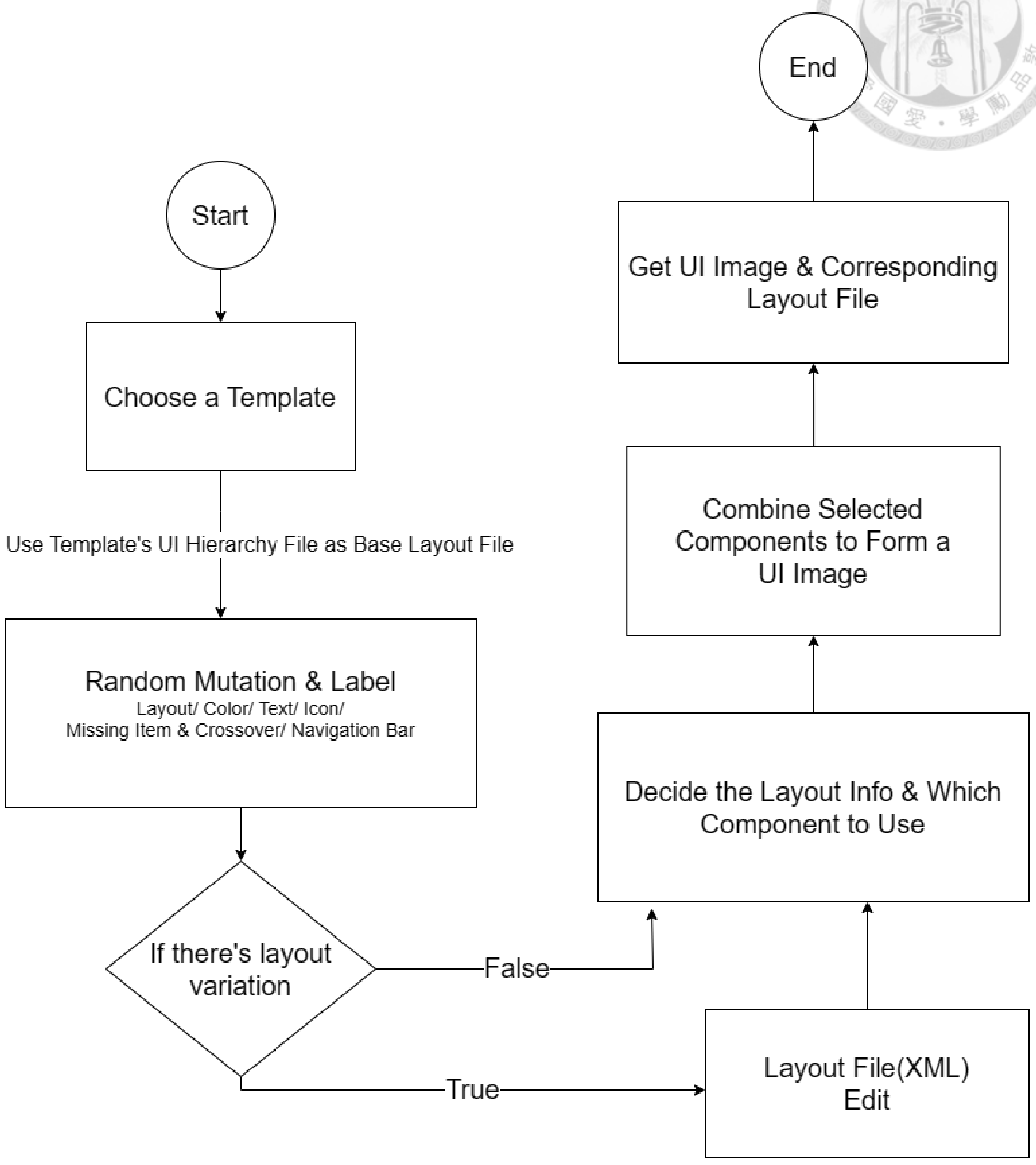

End
Start
Get UI Image & Corresponding Layout File
Choose a Template
Combine Selected Components to Form a UI Image
Use Template's UI Hierarchy File as Base Layout File
Random Mutation & Label
Layout/ Color/ Text/ Icon/
Missing Item & Crossover/ Navigation Bar
Decide the Layout Info & Which Component to Use
If there's layout variation
False
True
Layout File(XML) Edit

Figure 3.3 Flowchart of UI Generating Algorithm

The above flowchart(Figure 4.3) for the UI generation process provides a step-by-step breakdown of each iteration, resulting in the creation of a UI image and its corresponding layout file.

doi:10.6342/NTU202402814

1. Template and Mutation Selection:

Choose one template from the set of eight, along with its corresponding layout file. Randomly select a mutation type using the program, recording the mutation type as a label associated with the generated UI. During dataset organization, this label is combined with feedback from the questionnaire, resulting in dual-labeling for each UI (questionnaire response and mutation type).

2. Layout Editing for Mutations:

If the mutation involves layout changes, such as component repositioning, edit the base template's layout file to align with the altered UI. For mutations like font changes, layout file editing is unnecessary.

3. Component Assembly and Recordkeeping:

With the selected template and mutation type, assemble the UI by retrieving the necessary components from the curated library. Assemble the UI according to the layout, simultaneously recording the edited layout file as the UI's corresponding layout file.

Record the mutation type as an additional label, aiming to enable the model to learn distinctive features associated with each mutation type.

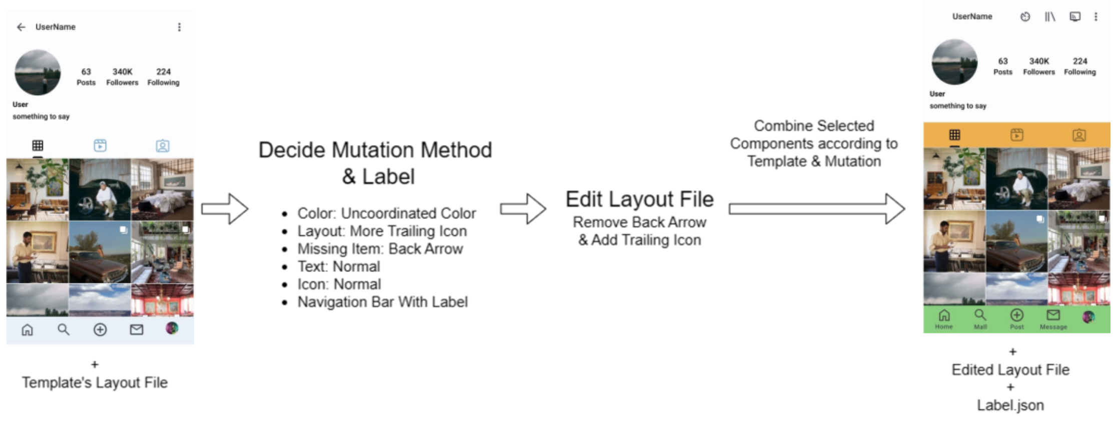

UserName
63 Posts
340K Followers
224 Following
User
something to say
+
Template's Layout File
Decide Mutation Method & Label
• Color: Uncoordinated Color
• Layout: More Trailing Icon
• Missing Item: Back Arrow
• Text: Normal
• Icon: Normal
• Navigation Bar With Label
Edit Layout File
Remove Back Arrow & Add Trailing Icon
Combine Selected Components according to Template & Mutation
Home
Mall
Post
Message
+
Edited Layout File
+
Label.json

Figure 3.4 Example of UI Generation

doi:10.6342/NTU202402814

Figure 4.4 illustrate the practical application of the UI generation process. In this instance, a template referencing Instagram is selected. A random mutation type is chosen, determining modifications such as inconsistent color schemes, excess trailing icons, and the removal of the back arrow. The layout file is edited to accommodate the mutations, reflecting changes in component placement and components are retrieved from the library and assembled into a UI. The resulting UI, post-mutation, is visualized to provide clarity and its edited layout file are saved, and the mutation type is recorded in a separate JSON file.

3.2.2 Mutation of UI

The concrete implementation of mutations encompasses six dimensions: Layout, Color, Typography, Icon, Missing Item & Crossover, and Navigation Bar. Each dimension consists of 3 to 4 mutation types, offering a diverse set of alterations. The occurrence of mutations in each dimension is probabilistic, independent, and fixed. Additionally, there is a possibility that no mutations occur, maintaining the original appearance.

These mutations are deliberately designed to deviate from design guidelines, serving as intentional challenges for the model to identify. The probability-based and independent nature of mutations ensures a diverse and unpredictable set of alterations in each dimension, contributing to the overall variability of the generated UIs. Table 4.1 shows the list of mutation type.

Dimension	**Mutation Type**
Layout	Too many Trailing Icons
	Too many icons in Navigation Bar
	Lack of icon in Navigation Bar

doi:10.6342/NTU202402814

	Follow the F mode or not?
	Additions of Login Button
	More than one Floating Action Button
Color	Change in color theme
	Inconsistent color schemes
	Alterations to color saturation
	Navigation Bar has same color as background
Typography,	Inconsistent weight
	Alterations to weight
	Alterations to spacing
	Changes in font styles
Missing Item & Crossover	Crossover: mix elements from other templates into another
	Intentional omissions of elements
Icon	Inconsistent weight
	Alterations to weight
	Changes in icon styles
	Inconsistent colored icon set
Navigation Bar	Additions or omissions of Label

Table 3.1 List of Mutation Types

3.3 QUX - Survey Design and Feedback Collection

A key component of our methodology involves the creation of a questionnaire aimed at examining the rationality of UI design and providing suggestions for improvement. Each questionnaire is meticulously crafted to assess various aspects of UI

doi:10.6342/NTU202402814

design, enabling UI developers and designers to save on the costs associated with data collection.

Following the questionnaire's design, feedback on UI designs is collected through surveys, with each questionnaire paired with a generated UI. Respondents access the questionnaire via a provided link and answer questions based on the accompanying UI image. The platform chosen for this purpose is Google Forms, offering a user-friendly and widely accessible interface for data collection.

The collected feedback serves as labeled data for model training, providing valuable insights into users' perceptions and preferences regarding UI designs. This section outlines the methodology employed for survey design and feedback collection, emphasizing the importance of leveraging user feedback to enhance the efficacy of our model in automating questionnaire responses.

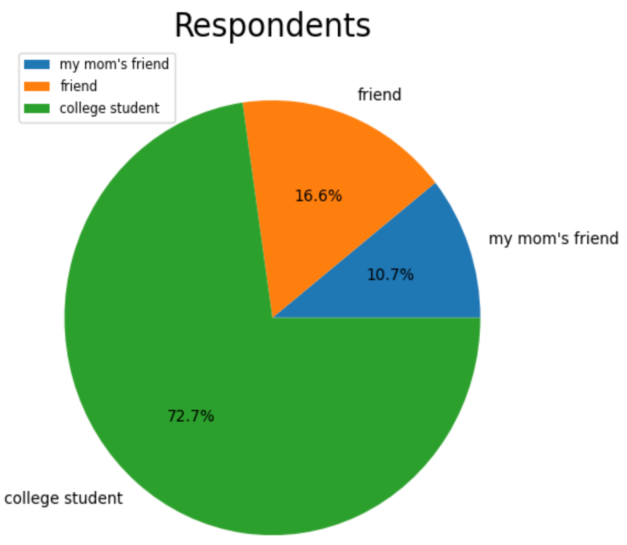


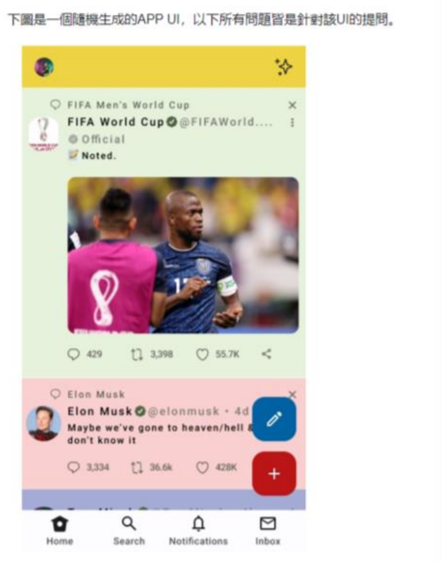
Figure 3.5 Survey Respondents Distribution

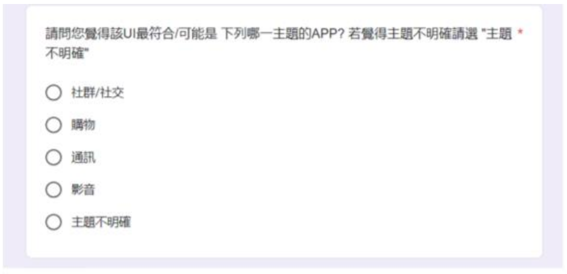
3.3.1 Questionnaire Questions

In the questionnaire, we inquire about various aspects of user experience, all

doi:10.6342/NTU202402814

centered around the UI images provided in the questionnaire, addressing elements such as color, icons, fonts, layout, and more. The questionnaire commences with the presentation of UI images generated by our system, followed by the series of questions. The question types include multiple-choice, multiple-choice but may be more than one answer, and rating questions. We reference the guidelines of questionnaire design to set questions

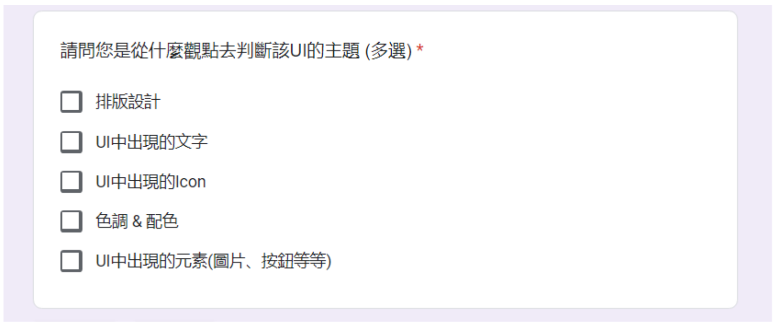

下圖是一個隨機生成的APP UI，以下所有問題皆是針對該UI的提問。


請問您覺得該UI最符合/可能是 下列哪一主題的APP? 若覺得主題不明確請選 "主題不明確" *

- ◯ 社群/社交
- ◯ 購物
- ◯ 通訊
- ◯ 影音
- ◯ 主題不明確

Figure 3.6 Question 1

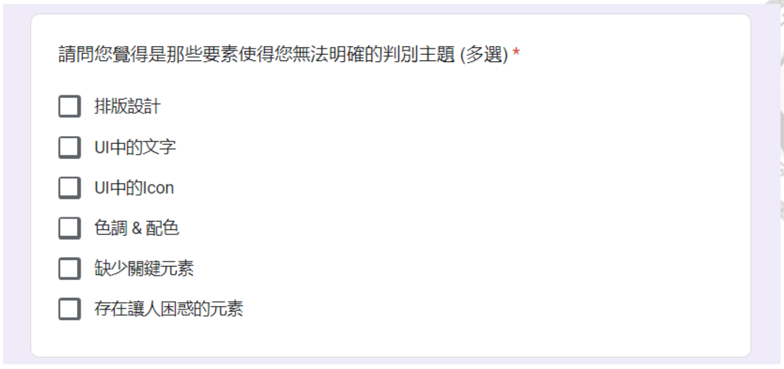
As shown in Figure 4.6, The first question (Q1) is a multiple-choice question regarding the UI theme, enabling developers to assess the clarity of their UI themes based on feedback.


請問您是從什麼觀點去判斷該UI的主題 (多選) *

- ☐ 排版設計
- ☐ UI中出現的文字
- ☐ UI中出現的Icon
- ☐ 色調 & 配色
- ☐ UI中出現的元素(圖片、按鈕等等)

Figure 3.7 Question 2

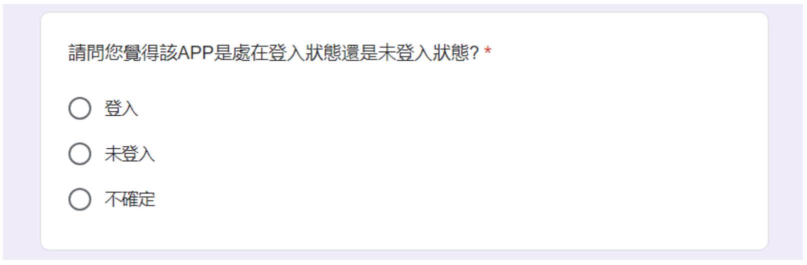
doi:10.6342/NTU202402814


請問您覺得是那些要素使得您無法明確的判別主題 (多選) *

☐ 排版設計
☐ UI中的文字
☐ UI中的Icon
☐ 色調 & 配色
☐ 缺少關鍵元素
☐ 存在讓人困惑的元素

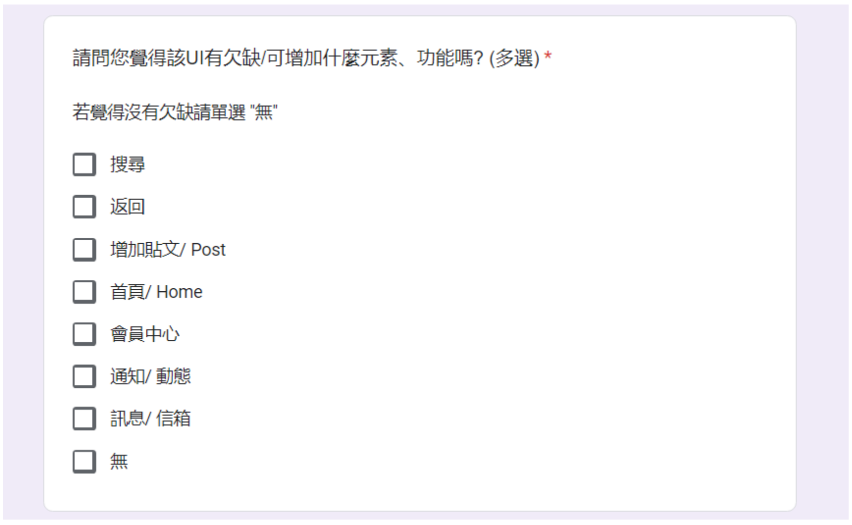
Figure 3.8 Question 3

Following the theme assessment, respondents are queried about the factors influencing their choice. If the previous response indicates an unclear theme, respondents are asked to specify why (Q2/Q3).


請問您覺得該APP是處在登入狀態還是未登入狀態? *

○ 登入
○ 未登入
○ 不確定

Figure 3.9 Question 4


請問您覺得該UI有欠缺/可增加什麼元素、功能嗎? (多選) *

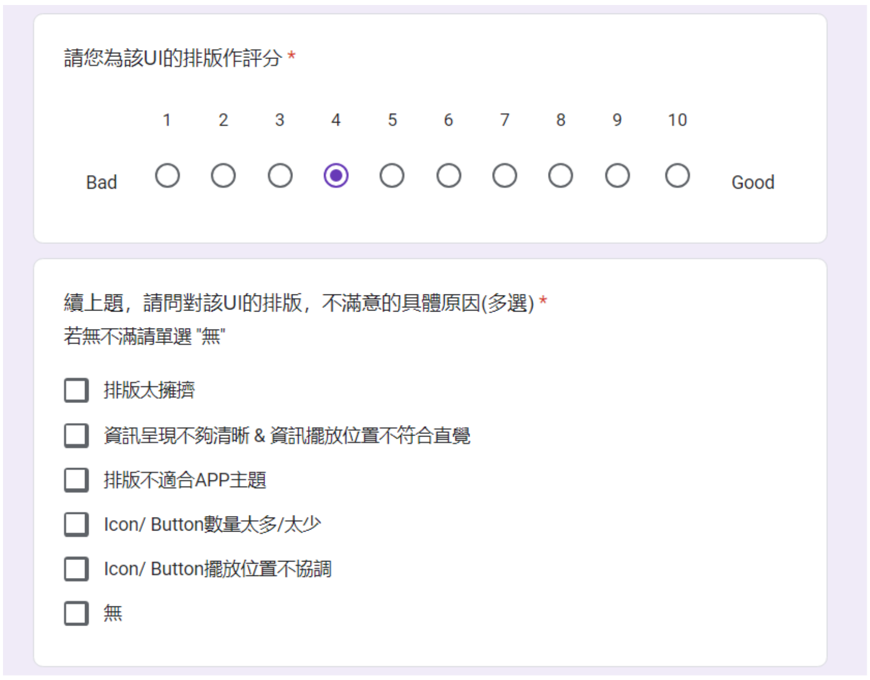
若覺得沒有欠缺請單選 "無"

☐ 搜尋
☐ 返回
☐ 增加貼文/ Post
☐ 首頁/ Home
☐ 會員中心
☐ 通知/ 動態
☐ 訊息/ 信箱
☐ 無

Figure 3.10 Question 5

doi:10.6342/NTU202402814

Question 4 focuses on the login status, providing insight into the clarity of the UI's login status. Question 5 inquires whether respondents feel any elements or functionalities are lacking, offering developers insights into potential deficiencies in their UI.



請您為該UI的排版作評分 *

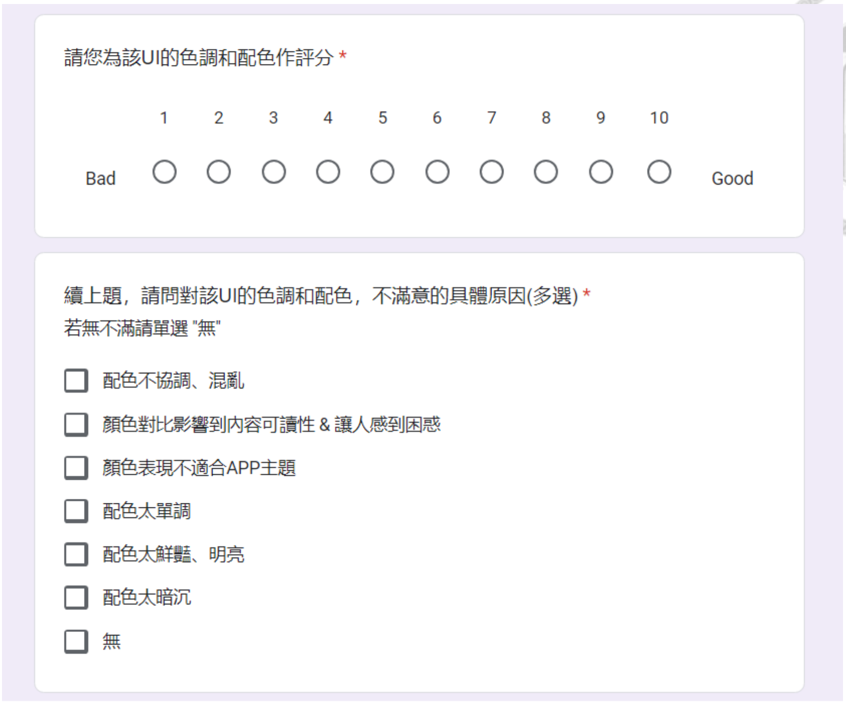
	1	2	3	4	5	6	7	8	9	10	
Bad	○	○	○	◉	○	○	○	○	○	○	Good

續上題，請問對該UI的排版，不滿意的具體原因(多選) *
若無不滿請單選 "無"

- ☐ 排版太擁擠
- ☐ 資訊呈現不夠清晰 & 資訊擺放位置不符合直覺
- ☐ 排版不適合APP主題
- ☐ Icon/ Button數量太多/太少
- ☐ Icon/ Button擺放位置不協調
- ☐ 無

Figure 3.11 Question 6&7

doi:10.6342/NTU202402814

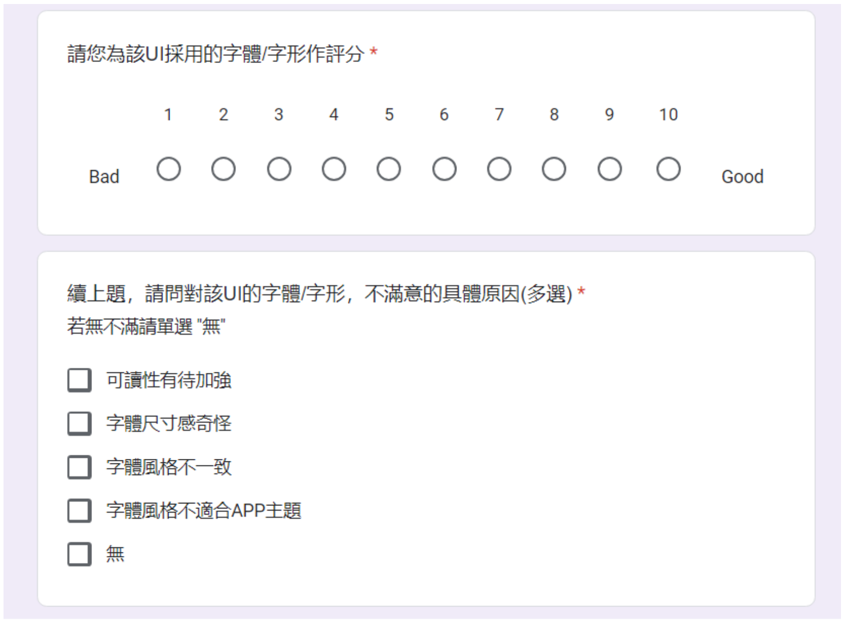

請您為該UI的色調和配色作評分 *

	1	2	3	4	5	6	7	8	9	10	
Bad	○	○	○	○	○	○	○	○	○	○	Good

續上題，請問對該UI的色調和配色，不滿意的具體原因(多選) *
若無不滿請單選 "無"

- ☐ 配色不協調、混亂
- ☐ 顏色對比影響到內容可讀性 & 讓人感到困惑
- ☐ 顏色表現不適合APP主題
- ☐ 配色太單調
- ☐ 配色太鮮豔、明亮
- ☐ 配色太暗沉
- ☐ 無

Figure 3.12　　Question 8&9


請您為該UI採用的字體/字形作評分 *

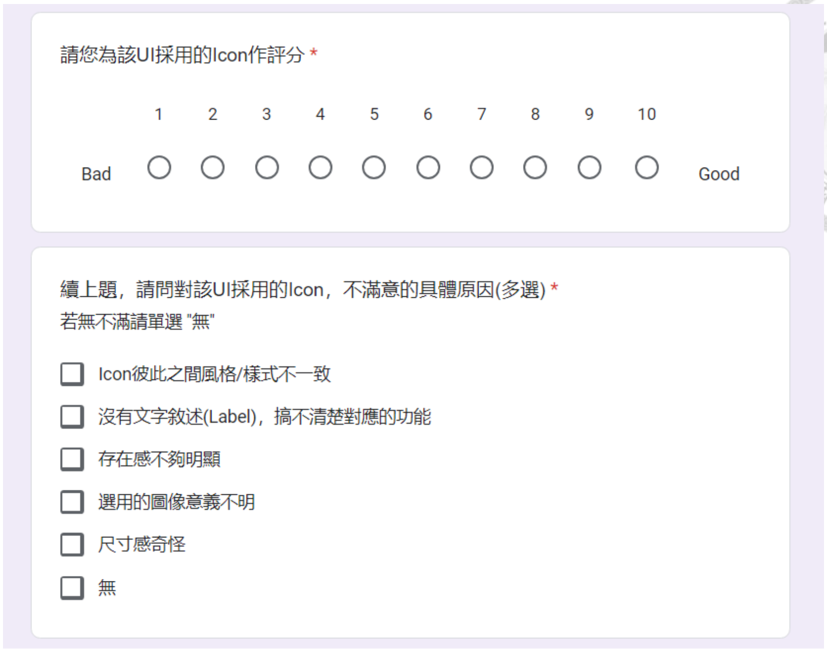
	1	2	3	4	5	6	7	8	9	10	
Bad	○	○	○	○	○	○	○	○	○	○	Good

續上題，請問對該UI的字體/字形，不滿意的具體原因(多選) *
若無不滿請單選 "無"

- ☐ 可讀性有待加強
- ☐ 字體尺寸感奇怪
- ☐ 字體風格不一致
- ☐ 字體風格不適合APP主題
- ☐ 無

Figure 3.13　　Question 10&11

doi:10.6342/NTU202402814


請您為該UI採用的Icon作評分 *

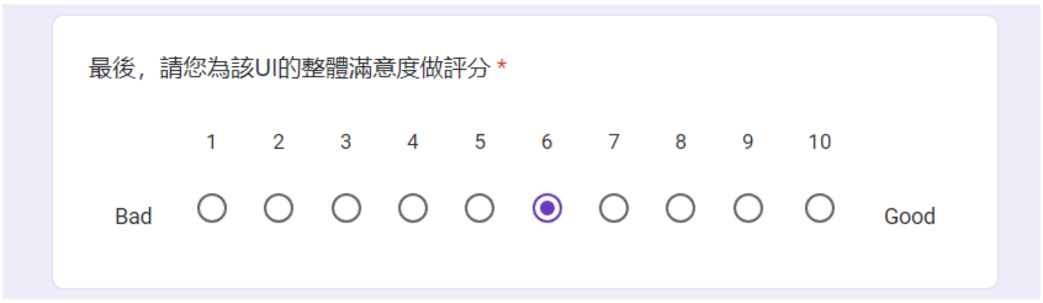
	1	2	3	4	5	6	7	8	9	10	
Bad	○	○	○	○	○	○	○	○	○	○	Good

續上題，請問對該UI採用的Icon，不滿意的具體原因(多選) *
若無不滿請單選 "無"

- ☐ Icon彼此之間風格/樣式不一致
- ☐ 沒有文字敘述(Label)，搞不清楚對應的功能
- ☐ 存在感不夠明顯
- ☐ 選用的圖像意義不明
- ☐ 尺寸感奇怪
- ☐ 無

Figure 3.14 Question 12&13

Questions 6&7 pertain to layout issues, with respondents first rating the UI layout and then specifying dissatisfaction points, providing developers with specific feedback. Similarly, Question 8&9 address UI color and tone, aiming to gather feedback on these aspects. Question 10&11&12&13 follow the same concept but focus on fonts and icons, respectively. Through these questions, developers can obtain recommendations and performance references across different aspects.


最後，請您為該UI的整體滿意度做評分 *

	1	2	3	4	5	6	7	8	9	10	
Bad	○	○	○	○	○	●	○	○	○	○	Good

Figure 3.15 Question 14

Finally, the overall rating section provides respondents with a scale of 1 to 10 for

doi:10.6342/NTU202402814

rating.

	Number of Mutation	Q6	Q8	Q10	Q12	Q14
Correlation Coefficient	1	-0.283495	-0.236261	-0.233441	-0.309036	-0.382062

Table 3.2 Correlation between number of mutation and rating problem

To understand the effect of mutations, we additionally analyzed the survey responses. We performed a correlation analysis between the number of mutations in the UI and the scores in various aspects. As shown in Table 4.2, there is a negative correlation between the number of UI mutations and the scores. In other words, the greater the number of mutations, the lower the scores, which aligns with our initial expectations.

doi:10.6342/NTU202402814

3.4 QUX - Questionnaire-based Model

3.4.1 Model Architecture and Training

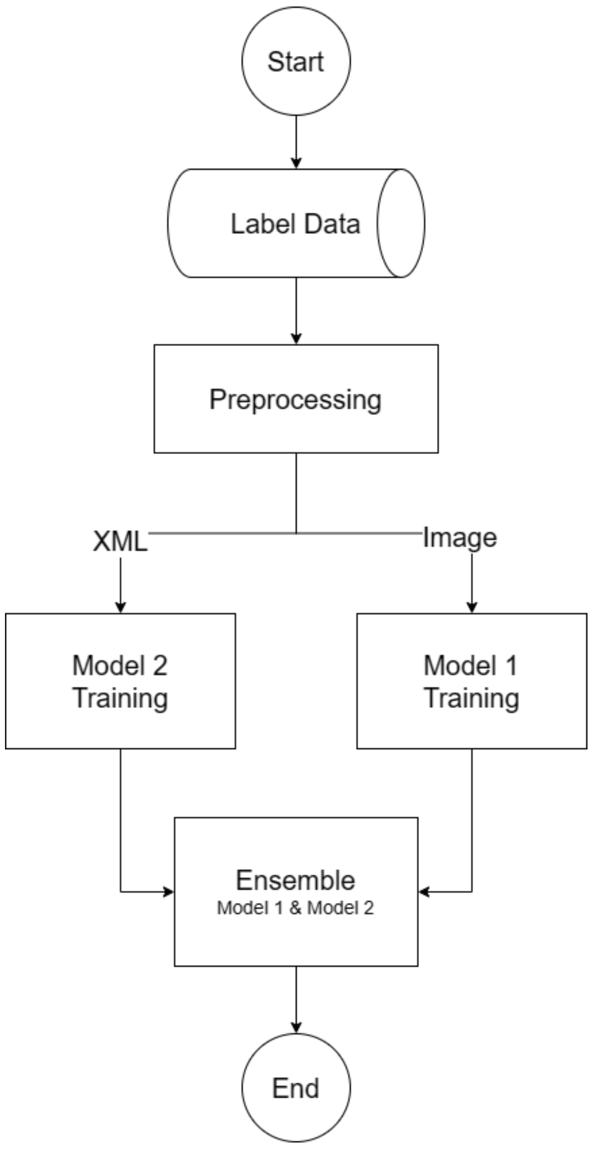


Figure 3.16 Construction of Questionnaire-based Model

In the training process, we utilize the dataset obtained from generated UI images and corresponding layout files, along with the labels collected from surveys, to train a questionnaire-based model using machine learning techniques. The training process involves two distinct models, each handling different types of input data: Model 1

doi:10.6342/NTU202402814

processes UI images, while Model 2 handles layout files. Both models are trained following a multi-task learning framework, where each question from the questionnaire, along with the prediction of mutation types, is treated as a separate task. This approach allows the models to simultaneously address multiple tasks, corresponding to the various question types in the questionnaire. The tasks we are dealing with can be categorized into three types:

1. Multi-Class Classification**:** Used for handling multiple-choice questions. In this task, each option corresponds to a class, and ultimately one of them is chosen as the answer.
2. Multi-Label Classification**:** Used for handling multiple-choice questions which may be more than one answer. Unlike multi-class classification, where only one option is selected from multiple choices, in multi-label classification, there can be multiple answers, resulting in multiple output labels.
3. Regression Problem**:** Used for handling rating questions. In this task, the answers to the questions are treated as a typical regression problem, where each question's rating corresponds to a numerical output.

Model 1, which operates on image data, employs the ResNet50 architecture, while Model 2, which processes layout files, utilizes the XLNet architecture. Pretrained models serve as the basis for both, fine-tuned to optimize performance.

In terms of other setup, the loss functions employed vary based on the task type: cross-entropy for multi-class classification tasks, binary cross-entropy for multi-label classification tasks, and smooth L1 loss for regression tasks. These loss functions are applied to each task independently, and the total loss is computed as the sum of losses across all tasks. The Adam optimizer is used to optimize the models during training.

doi:10.6342/NTU202402814

Finally, an ensemble approach is employed to combine the outputs of Model 1 and Model 2. A weighted average is applied, determined through experimentation to find the optimal performance.

3.4.2 Dataset

The labeled data is divided into two distinct sets: the training set and the testing set. The training set is utilized for training the model, while the testing set is employed to evaluate the model's performance.

The labeling methodology varies based on the type of questions present in the questionnaire. For choice questions, a One-Hot Encoding technique is applied for labeling. For instance, if there are five options and the first and second options are selected, a vector with five elements is used as the label, with the first and second elements marked as 1 and the rest as 0. For rating questions, integer values are used as labels.

doi:10.6342/NTU202402814

Chapter 4 Implementations and Evaluations

4.1 Programming Language and Environment Setup

First of all, the UI Generating Algorithm we proposed and training process of Questionnaire-based Model are implemented in Python 3.10.12 being a high-level, object-oriented, and interpreted programming language.

The training process of Questionnaire-based Model is implemented on Google Colaboratory(Colab). Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs.

For the machine learning aspects, specifically model development and training, we relied on PyTorch version 2.1.0. PyTorch is a powerful open-source deep learning framework known for its flexibility, ease of use, and strong community support. It offers a wide range of functionalities for building and training neural networks, making it an ideal choice for our research purposes.

4.2 Model Performance on Training & Testing

We conducted evaluations to assess both the performance of the models themselves and their real-world applicability when tested on actual mobile application UIs.

Firstly, we evaluate the model themselves. Table 5.1 and Table 5.2 show the performance data of the models, with the Table 5.1 depicting training data and the Table 5.2 depicting testing data. "M1" and "M2" denote Model 1 and Model 2, respectively, where Model 1 processes images and Model 2 processes Layout Files. The "S" prefix on

doi:10.6342/NTU202402814

M1 and M2 indicates simplified versions of the models, where the pretrained models used in the shared layers are replaced with standard CNN architectures. The combination of both models is represented by "M1 + M2". Additionally, we included the performance of GPT-4o on the test set at the bottom of the Table 5.2 for comparison.

For the evaluation of choice questions, we utilized metrics such as Accuracy, Precision, Recall, and F1 Score. Meanwhile, for rating questions, treated as a regression problem, we evaluated using Mean Absolute Error (MAE).

Model	**Size**	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**MAE**
M1	110.1MB	90.25	80.02	67.8	73.4	1.069
M2	1.07GB	70.54	59.61	54.8	57.1	2.403
S_M1	16.3MB	88.45	75.28	64.21	69.3	1.132
S_M2	2.8MB	83.21	65.55	45.54	53.74	1.506

Table 4.1 Training Evaluation

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**MAE**
M1	86.86	70.34	62.29	66.07	1.443
M2	73.87	42.52	33.51	37.48	6.368
M1 + M2	86.64	70.13	61.19	65.35	1.446
S_M1	85.29	66.74	58.11	62.12	1.603
S_M2	72.22	40.04	27.21	32.4	1.764
M1 + S_M2	86.85	70.95	61	65.6	1.422
S_M1 + M2	85.7	68.36	57.61	62.53	1.739
S_M1 + S_M2	85.58	68.83	55.82	61.65	1.56
GPT-4o	70.1	52.81	30.62	38.76	2.5

Table 4.2 Testing Evaluation

The performance analysis revealed that Model 2 exhibited comparatively lower performance than Model 1. This discrepancy may be attributed to Layout Files containing excessive noise and lacking visual representativeness. Layout Files often contain extensive hierarchical information based on developers' construction methods,

doi:10.6342/NTU202402814

leading to a potential overload of irrelevant information. Additionally, Layout Files lack the ability to capture visual aspects such as color and style, further diminishing their relevance to static UI analysis.

Regarding the simplified models, we experimented with different sizes to determine the optimal complexity level. By adjusting the neuron count in the fully connected layers preceding each task's output layer, we compared the performance of simplified Model 1 and Model 2 across three different sizes, ultimately selecting the simplified models with the highest testing F1 Score. Table 5.3 and Table 5.4 show the performance data of the models with different sizes

Model	**Size**	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**MAE**
S_M1	51.5MB	88.17	74.41	64.24	68.81	1.204
S_M1	27.3MB	88.54	75.03	65.53	69.95	1.193
S_M1	16.3MB	88.45	75.28	64.21	69.3	1.132
S_M2	3.3MB	84.05	66.48	50.05	57.1	1.516
S_M2	2.8MB	83.21	65.55	45.54	53.74	1.506
S_M2	2.6MB	82.5	64.76	41.43	50.53	1.478

Table 4.3 Training Evaluation of the Simplified Models with Different Sizes

Model	**Size**	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**MAE**
S_M1	51.5MB	84.78	65.72	57.41	61.28	1.643
S_M1	27.3MB	85	66.47	57.41	61.6	1.694
S_M1*	16.3MB	85.29	66.74	58.11	62.12	1.603
S_M2	3.3MB	70.8	37.78	27.35	31.72	1.781
S_M2*	2.8MB	72.22	40.04	27.21	32.4	1.764
S_M2	2.6MB	72.35	38.65	23.73	29.4	1.754

Table 4.4 Testing Evaluation of the Simplified Models with Different Sizes

We also evaluated the impact of preprocessing on model performance. Preprocessing involved removing irrelevant information from Layout Files to address issues such as excessive length and GPU memory constraints. The Layout File captured

doi:10.6342/NTU202402814

by UI Automator contains not only the hierarchy information and component coordinates of the UI but also a significant amount of information related to component interactions, such as whether they are clickable or draggable. Although this information is relevant to the user experience, our problem primarily concerns static visual analysis, where the relevance to dynamic aspects might not be as significant. Therefore, I attempted to remove this part of the information, retaining only the details relevant to the types of components and the hierarchical relationships within the UI. The results indicated a slight improvement in performance after preprocessing. Table 5.5 shows the performance data of the models built with and without data preprocessing.

S_M2	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**MAE**
Raw					
-Training	82.34	63.59	42.68	51.07	1.502
-Testing	71.63	38.58	26.27	31.25	1.769
Preprocess					
-Training	85.59	69.25	54.56	61.03	1.46
-Test	72.22	40.04	27.21	32.4	1.764

Table 4.5 Model Performance with and without Data Preprocessing

Furthermore, we conducted an analysis of individual questionnaire questions to gain insights into the models' performance on each task. The ensemble combinations displayed varying performance across choice and rating questions, with multiple-choice questions generally exhibiting superior performance compared to multiple-choice questions which may be more than one answer. Figure 5.1 shows the choice questions answering performance of all ensemble combinations and Figure 5.2 shows the rating questions answering performance

doi:10.6342/NTU202402814

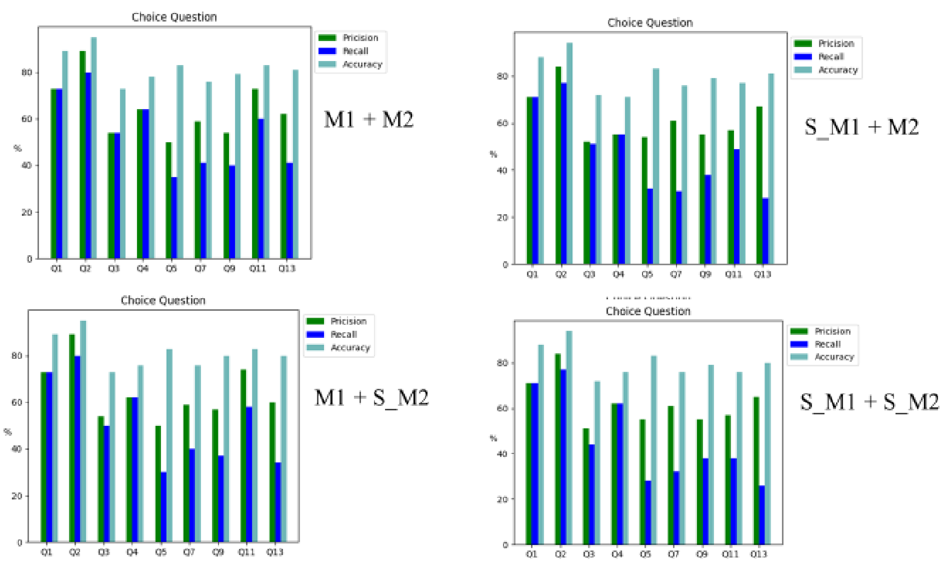


Figure 4.1 Choice Questions Answering Performance

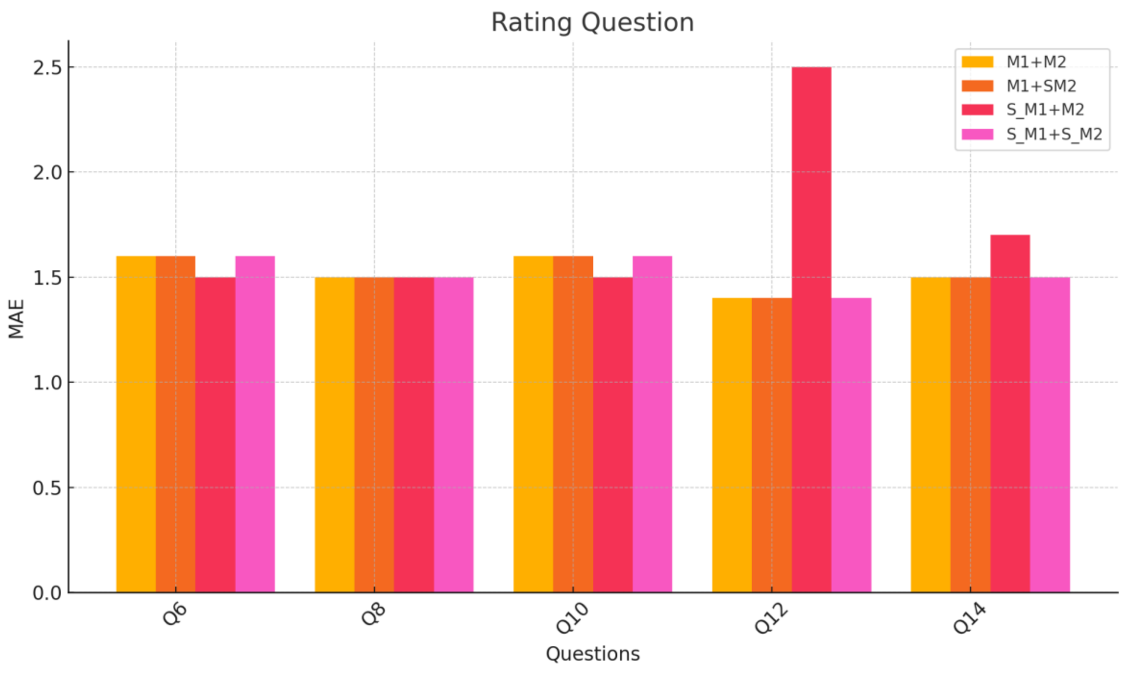


Figure 4.2 Rating Questions Answering Performance

4.3 Testing on Unlabeled Real-World Data

To evaluate the model's performance on real-world data, we conducted tests using

doi:10.6342/NTU202402814

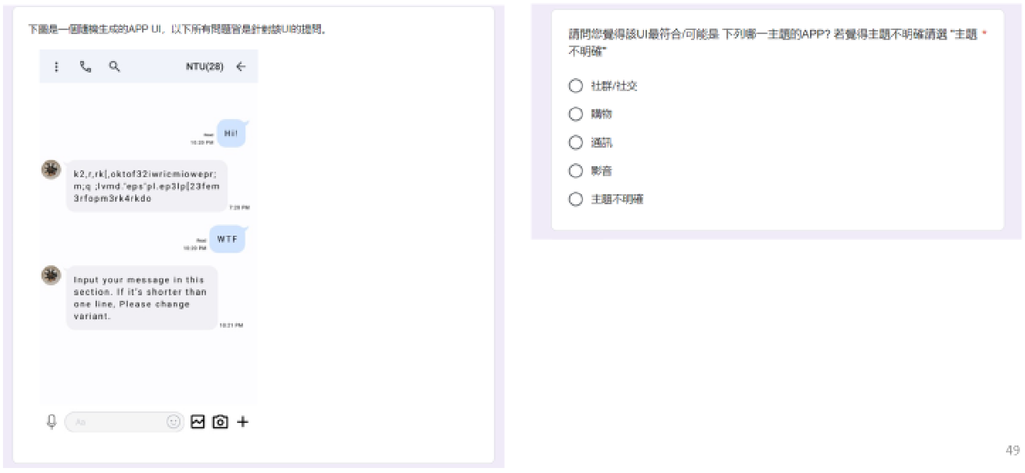
actual mobile applications as our testing objects. We fed screenshots and layout files of these apps into the model for assessment, focusing on aspects that could verify the correctness of answers and identify mutations, which are the objectives of our model. Table 5.6 shows the list of mobile applications for test.

1	Airbnb	17	KKbox	33	Slack
2	Amazon Shopping	18	Line TV	34	Snapchat
3	Apple Music	19	Linkedin	35	Spotify
4	BePtt	20	露天拍賣	36	TED
5	Bilibili	21	Medium	37	Telegram
6	Booking	22	Messenger	38	Google 翻譯
7	旋轉拍賣	23	Mixcloud	39	Twitter
8	Google Chat	24	Mixerbox	40	Wechat
9	Dcard	25	Momo 購物	41	Whatsapp
10	Ebay	26	Net a porter	42	Wall Street Journal
11	Facebook	27	NY Times	43	Yoox
12	Forest	28	PChome	44	Youtube
13	Friday 影音	29	Pinterest	45	Zara
14	Gimy	30	Reddit	46	Google meet
15	Ikea	31	Shopee		
16	Instagram	32	Skype		

Table 4.6 List of Mobile Applications

Firstly, we assessed the model's ability in classifying the main themes of apps, corresponding to the first question in our questionnaire. We calculated the accuracy of the model in determining the application's theme.

doi:10.6342/NTU202402814


下圖是一個隨機生成的APP UI，以下所有問題皆是針對該UI的提問。

NTU(28)

Hi!

k2,r,rk|,oktof32iwricmiowepr;m;q ;lvmd.'eps'pl.ep3lp[23fem3rfopm3rk4rkdo

WTF

Input your message in this section. If it's shorter than one line, Please change variant.

請問您覺得該UI最符合/可能是 下列哪一主題的APP? 若覺得主題不明確請選 "主題不明確" *

- ○ 社群/社交
- ○ 購物
- ○ 通訊
- ○ 影音
- ○ 主題不明確

Figure 4.3 Question Related to Class Classification

Next, we evaluated the model's capability in identifying the types of navigation bars, which is one of the mutations types predicted by the model. The model categorizes apps into three types: those with a navigation bar and text-attached icons (similar to the UI on the Figure 5.4’s far left), those with a navigation bar but no text-attached icons, and those without a navigation bar. We calculated the accuracy of the model's classification of navigation bar types, testing its ability to capture UI features.

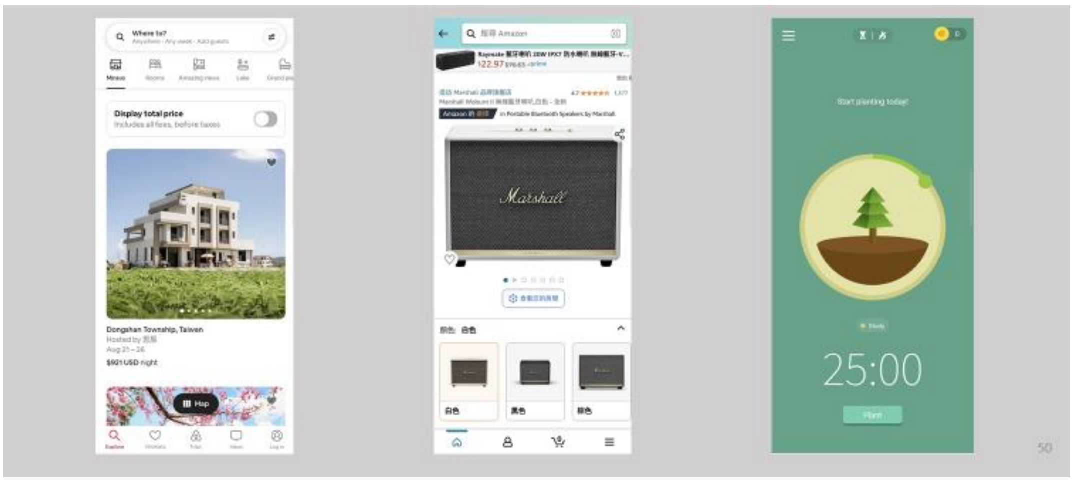
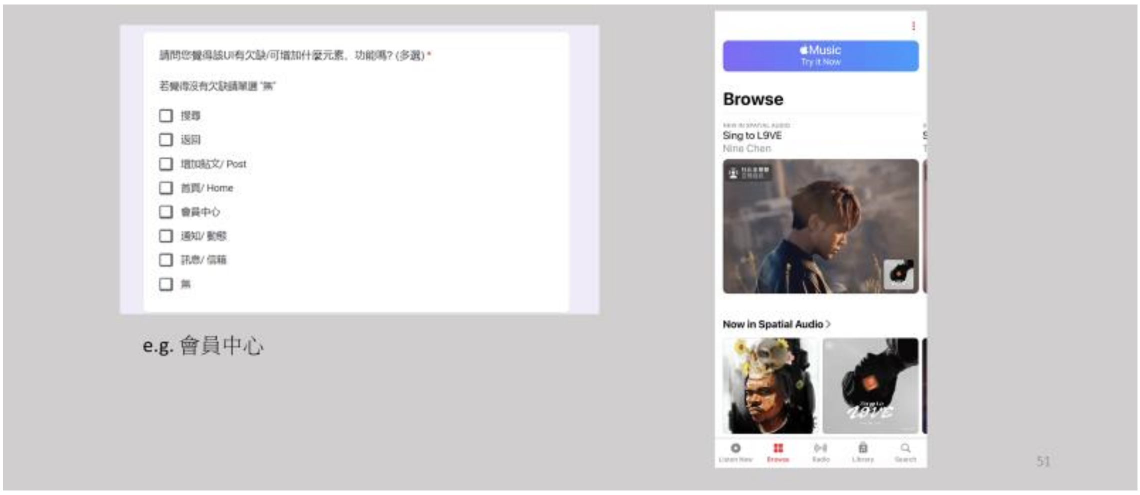


Figure 4.4 3 Types of Navigation Bar

Corresponding to question 5 in the questionnaire and the "missing item" mutation,

doi:10.6342/NTU202402814

the model provides feedback to developers regarding potential missing elements and functionalities in their UI. We assessed the accuracy of this feedback. For instance, if the model suggests the presence of a "Member Center" in the UI, we would check if there is indeed a corresponding icon or option on the tested UI page. If not, it would be considered a reasonable suggestion by the model; otherwise, it would be classified as a false prediction. We calculated accuracy using this approach.



請問您覺得該UI有欠缺/可增加什麼元素、功能嗎? (多選) *

若覺得沒有欠缺請單選 "無"

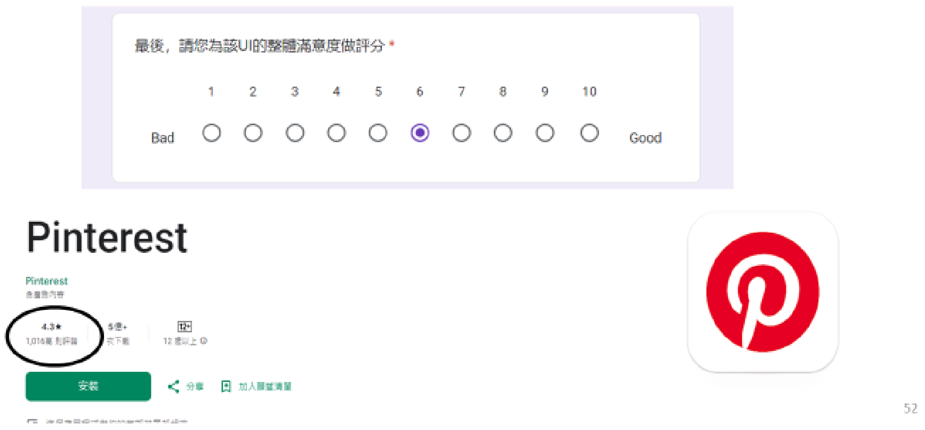
- ☐ 搜尋
- ☐ 返回
- ☐ 增加貼文/ Post
- ☐ 首頁/ Home
- ☐ 會員中心
- ☐ 通知/ 動態
- ☐ 訊息/ 信箱
- ☐ 無

e.g. 會員中心

Figure 4.5 Question Related to Finding Missing Items

Finally, we correlated the overall rating given by the model with the Google Play Store rating through regression analysis. We aimed to observe a positive correlation between them to validate the model's effectiveness, indicating that highly-rated apps also receive high scores from the model.

doi:10.6342/NTU202402814



最後，請您為該UI的整體滿意度做評分 *

Bad 1 2 3 4 5 [✓ 6] 7 8 9 10 Good

Figure 4.6 Model Rating and Play Store Rating

Table 5.7 presents the results of these tests for four ensemble combinations across the four dimensions mentioned above. The "Class" column represents the accuracy of theme classification, followed by the accuracy of navigation bar type classification, accuracy of missing item feedback, and the correlation coefficient between the overall score and the Play Store rating. And this table also includes the performance of GPT-4o for comparison.

Model	Class	Navigation Bar	Missing Item	Correlation
M1 + M2	36.95	50	70	0.11698
M1 + S_M2	32.6	54.34	63.63	0.18275
S_M1 + M2	43.47	52.17	58.33	0.13422
S_M1 + S_M2	43.47	56.5	63.63	-0.01695
M1	39.13	52.17	66.66	0.14235
M2	36.95	47.82	62.5	0.11633
S_M1	45.65	54.34	57.14	0.15856
S_M2	28.26	52.17	58.33	0.12493
GPT-4o	82.6	54.34	60	-0.12368

Table 4.7 Evaluation of Model Performance on Real-World APP

Notably, the performance of theme classification appears less satisfactory compared to the other dimensions. This might be attributed to the diverse themes

doi:10.6342/NTU202402814

present in modern mobile applications, with many examples encompassing multiple themes simultaneously. Thus, it might be more appropriate to revise theme classification as a multi-label question.

Regarding the correlation between the overall score and the Play Store rating, the correlation coefficient suggests a weak association between them. This might be because the Play Store rating is influenced by various factors such as brand image, pricing structure, service quality, etc., in addition to UI design. Hence, directly comparing the model's score with the Play Store rating might not reveal a strong correlation. A better approach could involve finding a universally recognized rating solely focused on UI. However, such a criterion is currently unavailable. If such a rating is identified in the future, further experiments could be conducted to observe its correlation with the model's scores.

doi:10.6342/NTU202402814

Chapter 5 Conclusion and Future Work

In this study, we propose a new method and framework(QUX) to automatically evaluate the UI designs. QUX include a questionnaire-based machine learning model aimed at providing automated feedback on GUI design for mobile applications and it's training method. The model generates completed questionnaires, allowing developers and designers to promptly receive application feedback without the need for manual data collection. Using QUX, we could reduce the cost of analyzing the mental models of APP users and improve the overall efficiency of the mobile APP development process.

To facilitate the training of our model, we developed a UI generation algorithm to construct synthetic GUI datasets. These datasets were designed in accordance with UI design guidelines to ensure their relevance to real-world applications.

Our dataset has the potential for further expansion in terms of diversity and quantity. Techniques such as Generative Adversarial Networks (GAN) could be employed to enhance dataset variability, thereby improving model training.

Furthermore, there are various avenues for enhancing our model. Different architectural setups beyond multitask learning could be explored. Experimentation with alternative combinations of loss functions, optimizers, and training methods, including unsupervised or semi-supervised approaches, may also yield performance improvements.

Regarding the output format, while we presented questionnaire responses as the current output, future iterations could consider alternative formats tailored specifically for UI design feedback. Additionally, incorporating more detailed questions or dynamic analysis could provide developers with more comprehensive insights into their

doi:10.6342/NTU202402814

application's UI. Developers can define new questions based on their preferences, and train their own model.

In conclusion, there is ample room for improvement and expansion in both the dataset construction process and the model architecture. These avenues for future work represent opportunities to enhance the effectiveness and applicability of our questionnaire-based model for GUI design feedback in mobile applications.

doi:10.6342/NTU202402814

REFERENCE

[1] Material Design, https://m3.material.io/

[2] C. Degott, N. P. Borges Jr, and A. Zeller, "Learning user interface element interactions," in *Proceedings of the 28th ACM SIGSOFT Inter national Symposium on Software Testing and Analysis*, 2019, pp. 296–306.

[3] Ting Su, Guozhu Meng, Yuting Chen, Ke Wu, Weiming Yang, Yao Yao, Geguang Pu, YangLiu, andZhendongSu.2017. Guided,stochastic model-based GUI testing of Android apps. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, Paderborn, Germany, September 4-8, 2017*. ACM, 245–256.

[4] Thomas D White, Gordon Fraser, and Guy J Brown. 2019. Improving random GUI testing with image-based widget detection. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 307–317.

[5] Zhen Dong, Marcel Böhme, Lucia Cojocaru, and Abhik Roychoudhury. 2020. Time-travel testing of android apps. In *Proceedings of the ACM/IEEE 42nd Inter national Conference on Software Engineering*. 481–492.

[6] Dehai Zhao, Zhenchang Xing, Chunyang Chen, Xiwei Xu, Liming Zhu, Guo qiang Li, and Jinshui Wang. 2020. Seenomaly: vision-based linting of GUI animation effects against design-don't guidelines. In *ICSE '20: 42nd Interna tional Conference on Software Engineering, Seoul, South Korea, 27 June- 19 July, 2020*, Gregg Rothermel and Doo-Hwan Bae (Eds.). ACM, 1286–1297.

[7] Kevin Moran, Boyang Li, Carlos Bernal-Cárdenas, Dan Jelf, and Denys Poshy

doi:10.6342/NTU202402814

vanyk. 2018. Automated reporting of GUI design violations for mobile apps. In *Proceedings of the 40th International Conference on Software Engineering.* 165–175.

[8] B. Yang, Z. Xing, X. Xia, C. Chen, D. Ye, and S. Li, UIS-hunter: Detecting UI design smells in Android apps, In *Proceedings of the 43rd International Conference on Software Engineering: Companion Proceedings*, pp. 89–92.

[9] Sen Chen, Lingling Fan, Chunyang Chen, Ting Su, Wenhe Li, Yang Liu, and Lihua Xu. 2019. Storydroid: Automated generation of storyboard for Android apps. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 596–607.

[10] Farnaz Behrang, Steven P Reiss, and Alessandro Orso. 2018. GUIfetch: sup porting app design and development through GUI search. In *5th International Conference on Mobile Software Engineering and Systems*. 236–246.

[11] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th Annual ACMSymposium on User Interface Software and Technology*. 845–854.

[12] Ruder, S. (2017). An Overview of Multi-Task Learning in Deep Neural Networks. *ArXiv, abs/1706.05098.*

[13] Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *Conference on Empirical Methods in Natural Language Processing*.

doi:10.6342/NTU202402814