111-2 網路程式設計 期末專題說明

投票系統

1102943 資工二甲 顏莉諭
1102966 資工二甲 邱翊銨

影片網址：https://youtu.be/zr6ayi5nYOs

一、 簡要說明作品

這是一個使用 Python 實作的投票系統，Server 建立投票 (投票主題、投票選項) ，Clients 加入投票。其中透過 PyQt6 實作 GUI、Socket 進行 TCP 連線、 Threading 服務各 Client。與一般投票系統不同之處在於，不限制 Clients 投票次數，直到 Server 結束投票，因此相較之下是個較自由的投票系統，適合日常使用。

二、 作品架構

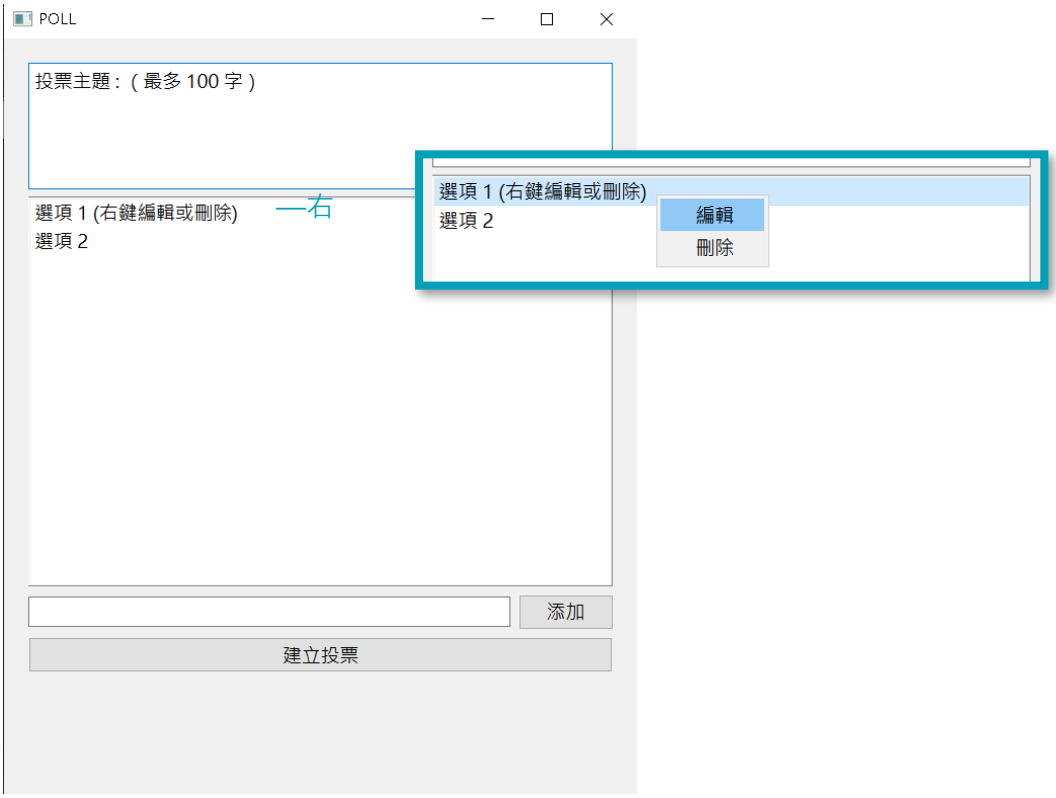
- 架構：Client / Server
- 語言：Python
- GUI：PyQt6

三、 作品功能

- 可自訂主題、選項、手動結束的投票系統
- 特色：不限制 Clients 各選項的投票次數
- 困難、挑戰：第一次實作 GUI ，花了不少時間熟悉 PyQt6
- 花最多精力：挑不明顯的小蟲：為了提高介面的可讀性，增加了很多文字來顯示各種狀況，在整合的過程中意外將文字轉換成了整數型別而發生錯誤導致連線中斷

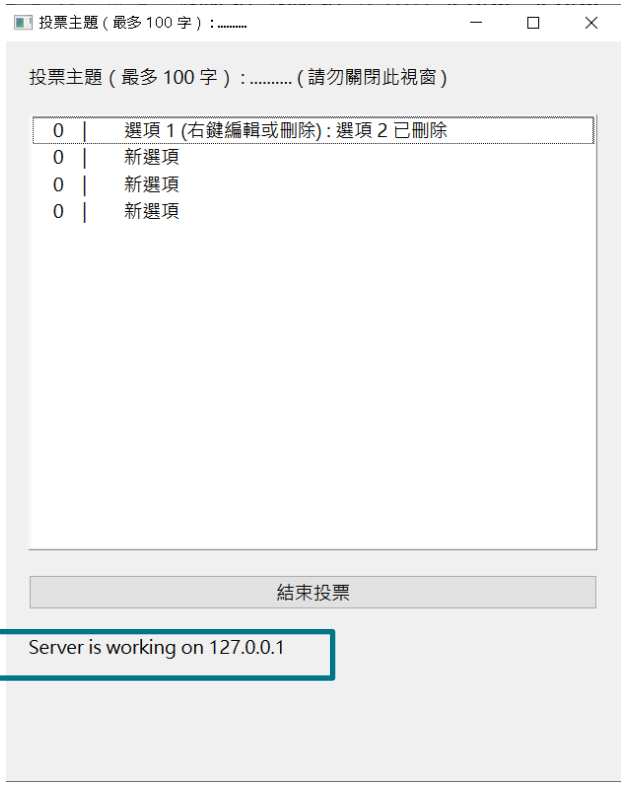
四、 作品執行過程說明

- Server 建立投票：執行 Server 的初始畫面

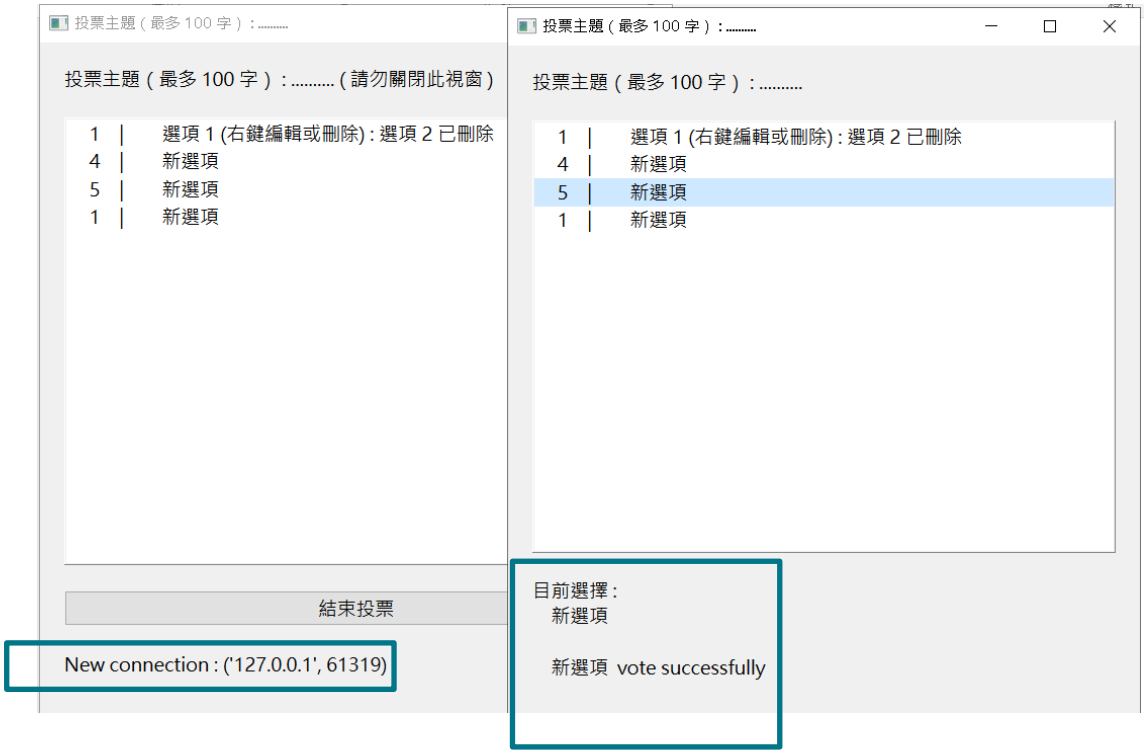


- Server 建立投票：編輯 / 刪除 / 添加選項

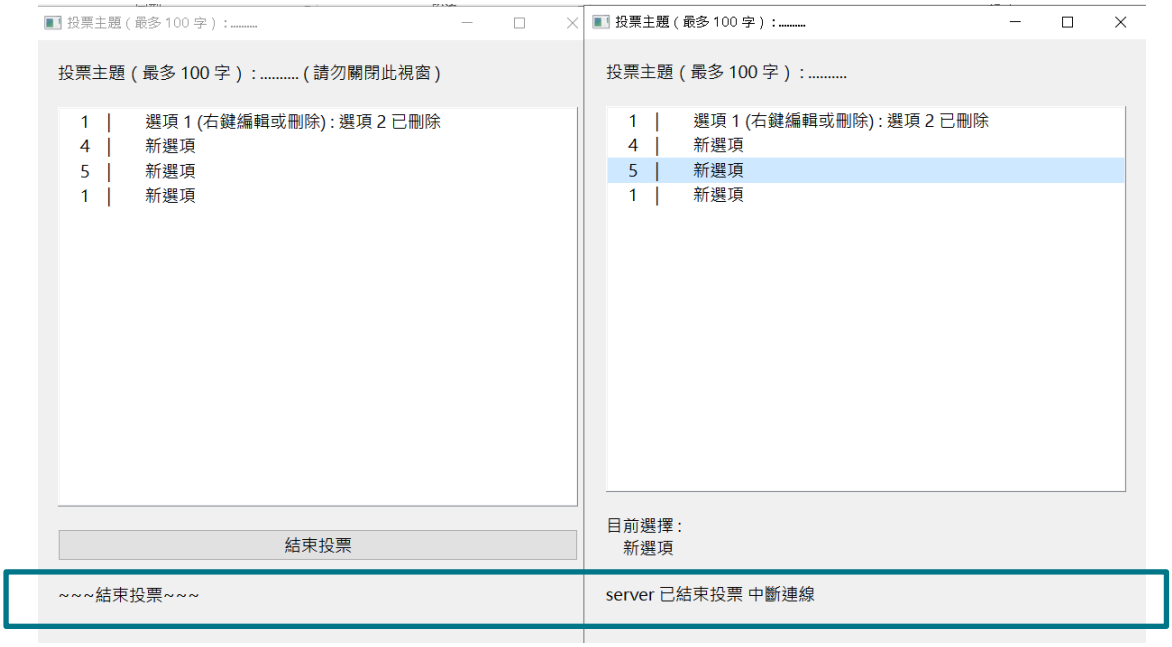
- Server 建立投票：建立投票



- 執行 Client 連線並投票 (左 Server / 右 Client)



- 結束投票 (左 Server / 右 Client)



- Server 未結束投票前斷線， Client 會收到並提示錯誤，無法投票
- Client 未結束投票前斷線不造成問題
- Server 結束投票後結束所有連線， Client 無法投票，新的 Client 無法加入連線

五、 使用技術

- 課程所學
  - SOCKET、TCP
  - Threading
- 自學
  - PyQt6（GUI）

六、 人員分工與時間分配

- 人員分工
  - 共同合作：書面報告、設計投票系統運作流程與細節、整合程式
  - 1102943 顏莉諭：設計 GUI 畫面與元件事件
  - 1102966 邱翊銨：連線與資料傳輸、影片
- 時間分配
  - GUI：47.5%
  - 連線與資料傳輸：27.5%
  - 最終調整（整合程式與細節）：10%
  - 書面報告：10%
  - 影片：5%

七、 心得

- 1102943 顏莉諭

  在這次的專題中，我主要負責實作 GUI。會選擇負責實作 GUI 主要有兩個原因，一是想用 Python 寫寫看 GUI，二是想提高系統完成度。謝謝組員已經先想好系統大致的功能與架構，因為有這個前提，才能讓我更有方向地專心探索如何使用新工具實作想要的效果。PYQT 不像之前學過的 Forms 應用程式有設計工具能預覽實作畫面、明確地提示可用的元件及其屬性、事件，只能自己依需求去找對應的指令，順利完成、妥協抑或換個方式，在過程中不斷調整、消化資訊、轉換邏輯，逐漸實現自己腦中的想像，雖然燒腦，但結合網路程式讓視窗程式真的能動起來後，真的很有成就感。

- 1102966 邱翊銨

  經過了這次期末專題，我更了解了 Server 和 Client 如何運作，也知道了 GUI 的應用與實作。一開始想架構和大概的實作方法時比較耗時間和腦力，有了大方向之後實作連線和資料傳輸、Server 要送收什麼資料 、Client 要送收什麼資料、哪邊要用多緒才不會卡住等等，寫起來就比較快，中間邊做邊改再加強一些細節部分就寫完了。最後整合的時候，為了看懂 GUI 的部分還花了蠻多時間研究 PYQT 的各種 Tag 和指令。最後整個作品自己還是蠻滿意的，雖然應該還有更多功能可以加入或是細節調整可以讓它更完整，不過畢竟只用了 3 天的時間寫出來，自己還是給不錯的評價。揮常桿蟹我的隊友，她真的好扛喔>_<

八、 程式碼 (https://hackmd.io/@yian0529/HJdMSKsU3)

Server :

```
# web

import socket
import threading
import time

# gui
from PyQt6 import QtWidgets, QtGui
from PyQt6.QtCore import QEvent, Qt
from PyQt6.QtWidgets import QApplication, QListWidget, QLineEdit,
QVBoxLayout, QWidget
import sys

# gui
class MyWidget(QtWidgets.QWidget): # 建立投票
  w_size = 500
  h_size = 600
  voteli = ['選項 1 (右鍵編輯或刪除)', '選項 2'] # 預設選項

  def __init__(self):
    super().__init__()
    self.setWindowTitle('POLL')
    self.current_item = None
    self.resize(self.w_size, self.h_size)
    self.style()
    self.ui()
    self.w_size = 500
    self.h_size = 600

  def ui(self): # 視窗程式畫面與元件事件綁定
    self.box = QtWidgets.QWidget(self)
    self.box.setGeometry(10, 10, self.w_size - 20, self.h_size - 100)
# 設定位置
    self.v_layout = QtWidgets.QVBoxLayout(self.box)
    self.h_layout = QtWidgets.QHBoxLayout()

    self.question_inp = QtWidgets.QPlainTextEdit(self)  # 投票主題輸入框
    self.question_inp.setPlainText('投票主題： （最多 100 字）')
    self.question_inp.installEventFilter(self)  # 字數限制
    self.question_inp.setFixedHeight(100)

    self.listwidget = QtWidgets.QListWidget(self)  # 建立列表選擇框元件
    self.listwidget.addItems(self.voteli)    # 加入選項

self.listwidget.setContextMenuPolicy(Qt.ContextMenuPolicy.CustomContextM
enu) # 選項右鍵事件綁定（修改、刪除）

self.listwidget.customContextMenuRequested.connect(self.on_context_menu)

    self.senddata = QtWidgets.QLineEdit(self) # 新增選項輸入框
    self.senddata.setMaxLength(30) # 字數限制
```

```
    self.add_button = QtWidgets.QPushButton("添加") # 新增選項按鈕
    self.add_button.clicked.connect(self.add_item)  # 按鈕事件綁定

    self.subm_button = QtWidgets.QPushButton("建立投票") # 建立投票按鈕
    self.subm_button.clicked.connect(self.createp) # 按鈕事件綁定

    self.v_layout.addWidget(self.question_inp) # 排版以上元件
    self.v_layout.addWidget(self.listwidget)
    self.h_layout.addWidget(self.senddata)
    self.h_layout.addWidget(self.add_button)
    self.v_layout.addLayout(self.h_layout)
    self.v_layout.addWidget(self.subm_button)

  def style(self): # QSS
    self.setStyleSheet('''
      font-size: 15px;
      QListWidget{
        color:#00f;
      }
      QListWidget::item{
        width:30px;
      }
      QListWidget::item:selected{
        color:#f00;
        background:#000;
      }

    ''')

  def eventFilter(self, obj, event): # 投票主題輸入框字數限制
    if obj == self.question_inp and event.type() ==
QEvent.Type.KeyPress: # 輸入內容
      text = self.question_inp.toPlainText()
      key = event.key()
      if key == Qt.Key.Key_Backspace or key == Qt.Key.Key_Left or key ==
Qt.Key.Key_Right or key == Qt.Key.Key_Up or key == Qt.Key.Key_Down:
        return super().eventFilter(obj, event)  # 只允許刪除和移動
      elif len(text) >= 100:
        return True  # 超過限制字數就忽略輸入內容
    return super().eventFilter(obj, event)

  def on_context_menu(self, pos): # 選項右鍵事件綁定（修改、刪除）
    item = self.listwidget.itemAt(pos) # 透過點擊位置取得欲操作的選項

    if item:
      context = QtWidgets.QMenu(self) # 建立選單

      ac_edit = QtGui.QAction("編輯", self) # 選項
      ac_dele = QtGui.QAction("刪除", self)

      ac_edit.triggered.connect(lambda: self.onContextAction(item, "編
輯")) # 綁定事件
      ac_dele.triggered.connect(lambda: self.onContextAction(item, "刪
除"))
```

```
        context.addAction(ac_edit) # 選項加入選單
        context.addAction(ac_dele)

        context.exec(self.listwidget.mapToGlobal(pos))

    def onContextAction(self, item, action): # 綁定的事件
        print(f"Selected item: {item.text()}, Action: {action}")
        match action:
            case '編輯':
                self.startEditing(item)
            case '刪除':
                row = self.listwidget.row(item) # 選擇
                self.listwidget.takeItem(row) # 刪除


    def startEditing(self, item): # 開始編輯選項內容
        if self.current_item is not None:
            self.closeEditor()

        self.current_item = item

        editor = QLineEdit(item.text()) # 欲編輯的選項上開啟編輯器
        self.listwidget.setItemWidget(item, editor)
        editor.setMaxLength(30)

        editor.editingFinished.connect(self.finishEditing) # 完成編輯後呼叫的
事件

        editor.setFocus()

    def finishEditing(self): # 關閉編輯器
        if self.current_item is not None:
            editor = self.listwidget.itemWidget(self.current_item)
            if editor:
                self.current_item.setText(editor.text()) # 套用文字
                self.listwidget.removeItemWidget(self.current_item)
                editor.deleteLater()

                self.current_item = None

    def add_item(self): # 新增項目
        try:
            text = self.senddata.text()
            if text:
                self.listwidget.addItem(text) # 加入新項目內容
                self.senddata.clear()
                self.senddata.setFocus()
        except:
            pass

    def createp(self): # 建立投票
        self.voteli = []
        strt = self.question_inp.toPlainText() # 取得投票主題
        for index in range(self.listwidget.count()): # 將所有選項加入列表
```

```
      self.voteli.append(self.listwidget.item(index).text())
    self.nw = newWindow(self.voteli, strt) # 傳送至新視窗
    self.nw.show()
    self.close()

class newWindow(QtWidgets.QWidget): # 正式投票
  w_size = 500
  h_size = 600
  voteli = []
  votecnt = []
  tit = ''

  def __init__(self, votel, subj): # 設定傳入的投票項目
    self.current_item = None
    self.voteli = votel
    super().__init__()
    self.setWindowTitle(subj)
    self.resize(self.w_size, self.h_size)
    self.ui(subj)
    self.tit = subj

    #web
    start = threading.Thread(target=self.startChat) # 收送資料
    start.start()

  def ui(self, subj): # 視窗程式畫面與元件事件綁定
    self.box = QtWidgets.QWidget(self)
    self.box.setGeometry(10, 10, self.w_size - 20, self.h_size - 100)
# 設定位置
    self.v_layout = QtWidgets.QVBoxLayout(self.box)

    self.showquestion = QtWidgets.QLabel(subj + ' ( 請勿關閉此視窗 )')
    self.showquestion.setWordWrap(True)

    self.listwidget = QtWidgets.QListWidget(self)  # 建立列表選擇框元件
    self.listwidget.setSelectionMode( # 僅供檢視
      QtWidgets.QAbstractItemView.SelectionMode.NoSelection
    )
    self.listwidget.addItems(self.voteli)    # 建立選單
    for i in range(len(self.voteli)): #  設定初始票數及顯示選項
      self.votecnt.append(0)
      self.listwidget.item(i).setText('  ' + str(self.votecnt[i]) + '  |
' + self.voteli[i])

    self.v_layout.addWidget(self.showquestion)
    self.v_layout.addWidget(self.listwidget)

    self.subm_button = QtWidgets.QPushButton("結束投票")
    self.subm_button.clicked.connect(self.finip)
    self.showvote = QtWidgets.QLabel("Client: ")

    self.v_layout.addWidget(self.subm_button)
    self.v_layout.addWidget(self.showvote)
    self.style()
    self.v_layout.setSpacing(20)
```

```
  def style(self):
    self.setStyleSheet("""
        font-size: 15px;
        QListWidget::item{
          padding-top: 10px;
        }
    """)

  def show_item(self, i): # 更新 index 為 i 的選項文字
    self.listwidget.item(i).setText('  ' + str(self.votecnt[i]) + '  |
' + self.voteli[i])

  #web
  def finip(self):
    self.showvote.setText("~~~結束投票~~~")
    while len(clients) != 0:  # 斷線
      for client in clients:
        try:
          client.send("pollover".encode())
          client.close()
        except:
          print("client 自行斷線")
          pass
        clients.remove(client)
    s.close()

  def startChat(self):
    def handle(conn, addr):
      self.showvote.setText(f"New connection : {addr}")
      print(f"New connection : {addr}")
      connected = True
      while connected:
        try:
          message = conn.recv(1024).decode() # 接收被投票的選項 index
          print(message)
          ind = int(message)
          self.votecnt[ind] += 1 # 增加票數
          broadcastMessage(message.encode()) # 傳送給所有 client
          self.show_item(ind) # 更新被投票的選項
        except:
          break
      conn.close()

    self.showvote.setText("Server is working on " + HOST)
    print("Server is working on " + HOST)

    s.listen()

    while True:
      try:
        conn, addr = s.accept()
      except:
        break
```

```
        conn.send(self.tit.encode())
        time.sleep(0.5)

        conn.send(str(len(self.voteli)).encode()) #  傳送主題

        for i in range(len(self.voteli)): # 傳送選項及其當前票數
          conn.send(self.voteli[i].encode())
          time.sleep(0.2)
          conn.send(str(self.votecnt[i]).encode())
          time.sleep(0.2)

        clients.append(conn)
        conn.send('Connection successful!'.encode(FORMAT))

        # Start the handling thread
        thread = threading.Thread(target=handle, args=(conn, addr))
        thread.start()

def broadcastMessage(message): # 通知所有 client 有新投票
  for client in clients:
    outdata = message.decode()
    print(outdata)
    try:
      client.send(outdata.encode(FORMAT))
    except:
      client.close()
      clients.remove(client)

if __name__ == '__main__':
  # web
  PORT = 7000
  HOST = "127.0.0.1"
  FORMAT = "utf-8"
  clients = []

  s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # tcp
  s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

  s.bind((HOST, PORT))
  s.listen(5)

  app = QtWidgets.QApplication(sys.argv)
  Form = MyWidget()
  Form.show()
      sys.exit(app.exec())
```

Client :

```
# -*- coding: utf-8 -*-

# web
import socket
import threading

# gui
from PyQt6 import QtWidgets, QtGui
from PyQt6.QtCore import QEvent, Qt
from PyQt6.QtWidgets import QApplication, QListWidget, QLineEdit,
QVBoxLayout, QWidget
import sys


class MyWidget(QtWidgets.QWidget):
  w_size = 500 # 視窗大小
  h_size = 600
  voteli = [] # 選項
  votecnt = [] # 選項票數
  tit = '' # 投票主題

  def __init__(self, votecnt, votel, subj): # 設定傳入的投票項目
    self.voteli = votel
    self.tit = subj
    self.votecnt = votecnt
    super().__init__()
    self.setWindowTitle(self.tit)
    self.resize(self.w_size, self.h_size)
    self.ui(subj)
    print(self.voteli)

    rcv = threading.Thread(target = self.receive) # 接收資料
    rcv.start()

  def ui(self, subj): # 視窗程式畫面與元件事件綁定
    self.box = QtWidgets.QWidget(self)
    self.box.setGeometry(10, 10, self.w_size - 20, self.h_size - 100)
# 設定位置
    self.v_layout = QtWidgets.QVBoxLayout(self.box) # 排版

    self.showquestion = QtWidgets.QLabel(subj) # 顯示主題
    self.showquestion.setWordWrap(True)

    self.listwidget = QtWidgets.QListWidget(self)  # 建立列表選擇框元件
    self.listwidget.addItems(self.voteli)    # 建立並顯示選項
    for i in range(len(self.voteli)):
      self.listwidget.item(i).setText('  ' + str(self.votecnt[i]) + '  |
' + self.voteli[i])
    self.listwidget.itemClicked.connect(self.toggle_selection) # 選擇選項
點擊事件：投票

    self.showvote = QtWidgets.QLabel("目前選擇： ") # 顯示目前選擇
    self.showvote.setAlignment(Qt.AlignmentFlag.AlignTop)
    self.showvote.setWordWrap(True) # 允許換行
```

```
    self.showc = QtWidgets.QLabel("Client: ") # 顯示提示

    self.v_layout.addWidget(self.showquestion) # 排版以上元件
    self.v_layout.addWidget(self.listwidget)
    self.v_layout.addWidget(self.showvote)
    self.v_layout.addWidget(self.showc)

    self.style() # QSS
    self.v_layout.setSpacing(20)

  def style(self): # QSS
    self.setStyleSheet("""
        font-size: 15px;
        QListWidget::item{
          padding-top: 10px;
        }
    """)

  def toggle_selection(self, item): # 點擊事件：投票
    ind = self.listwidget.row(item) # 取得點擊項目的 index
    if item.isSelected(): # 選擇項目狀態判斷
      try:
        client.send(str(ind).encode(FORMAT)) # 傳送點擊項目的 index 至
server 表示投此項目
      except:
        self.showc.setText("連線出現問題，中斷連線")
        print("server 中斷連線")
        client.close()
    self.show_vote(ind) # 顯示目前選擇

  def show_vote(self, ind): # 顯示目前選擇
    st = "目前選擇：\n  " + self.voteli[ind]
    self.showvote.setText(st) # 套用文字
    self.showvote.adjustSize()

  def show_item(self, i): # 更新並顯示選項票數
    self.listwidget.item(i).setText('  ' + str(self.votecnt[i]) + '  |
' + self.voteli[i])
    self.showc.setText('  ' +  self.voteli[i]+"  vote successfully")
  #web
  def receive(self):
    while True:
      try:
        message = client.recv(1024).decode(FORMAT) # 接收
        if(message=="pollover"):
          self.showc.setText("server 已結束投票 中斷連線")
          print("server 中斷連線")
          client.close()
          break
        print(message)
        ind = int(message) # 更新其他人投的票（用 index 辨識）
        self.votecnt[ind] += 1
        self.show_item(ind)
```

```
      except:
        # an error will be printed on the command line or console if
there's an error
        self.showc.setText("連線出現問題 中斷連線")
        print("server 中斷連線")
        client.close()
        break

if __name__ == '__main__':
  HOST = "127.0.0.1"
  PORT = 7000
  FORMAT = "utf-8"

  client = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
  try:
    client.connect((HOST, PORT))
    print("投票建立中....")
    votelist = []
    votecnt = []
    titlename = client.recv(1024).decode() # 接收主題
    num = int(client.recv(1024).decode()) # 接收選項數

    for i in range(num):
      data = client.recv(1024).decode() # 選項
      votelist.append(data)
      data = int(client.recv(1024).decode()) # 該選項當前的票數
      votecnt.append(data)
      print(data)
    print(client.recv(1024).decode(FORMAT))

    app = QtWidgets.QApplication(sys.argv)
    Form = MyWidget(votecnt, votelist, titlename) # 建立並顯示 GUI
    Form.show()
    sys.exit(app.exec())
  except:
    print("無法連線")
```