**Використання фреймворку QT для реалізації проєкту фінансового менеджеру**

1. **Які конкретні задачі планували вирішувати за допомогою цієї бібліотеки?**

За допомогою даного фреймворку, планувалася повна реалізація проєкту фінансового менеджеру.

**Конкретні задачі, які мали реалізуватись за допомогою даного фреймворку:**

* Розробка функціонального GUI: QT було обрано для створення зручного інтерфейсу користувача, що включає в себе певний вибір секцій з різним функціоналом: основне вікно з переглядом категорій, додаванням транзакцій та певними аналітичними даними; вікно з повним списком транзакцій упорядкованим за датами, в якому можна їх додавати, змінювати, видаляти, фільтрувати і тд.
* Також додаткові віджети для функціоналу з функціональними розділами: для логінізації, фільтрації транзакцій, перегляд за категоріями.

Організація взаємодії з базою даних для можливості логінізації юзера та збереження його даних:

* Було обране використання бази даних FireBase що пропонує Realtime Database, що надає можливість синхронізувати між клієнтом і сервером дані у реальному часі для збереження користувачів, їх транзакцій, та інших даних. Робота з базою даних у Qt планувалася з використанням Rest API, оскільки FireBase надає повноцінний доступ до своїх сервісів через HTTP-запити.

1. **Чому було обрано саме цю бібліотеку, а не аналоги?**

Рішення, щодо вибору Qt базувалося на кількох аспектах, що робить даний фреймворк оптимальним, на порівняння з іншими:

* Кросплатформеність: Qt дозволяє створювати додатки які працюють на Windows, MacOs, Linux без змін у коді, що забезпечує універсальність і можливість запуску програми на різних ОС, що надає йому перевагу відносно інших фреймворків (наприклад Windows Forms)
* Розвинена система GUI: модулі Qt дозволяють розробляти як класичні інтерфейси, так і інтерактивні за допомогою QML. Фінансовий менеджер вимагає чіткої вузалізації даних, таких як транзакцій, категорій, графіки та інше, і Qt дозволяє реалізувати це ефективно.
* Зручна інтеграція з базами даних та мережевих можливостей: у поєднанні з певними адаптованими для Qt бібліотек (наприклад Qt Network) можна інтегрувати FireBase через використання RestAPI. Проєкт вимагав постійної роботи з даними(завантаження, оновлення, синхронізація). Завдяки Qt Network інтеграція з FireBase набагато спрощується.
* Висока продуктивність: Qt використовується в поєднанні з с++ (можливе використання і з іншими мовами програмування), що забезпечує високу швидкість виконання та мале споживання ресурсів.
* Qt має велику базу документації, що спрощує навчання і швидке розв'язання проблем.

1. **Наскільки просто та зрозуміло було отримати, встановити, налаштувати та почати використовувати цю бібліотеку?**

Проблем зі встановленням даного фреймворку не виникало. З офіційного сайту завантажується інсталятор. У sdk є зручний інтерфейс у якому можна обрати потрібну версію, модулі та інші компоненти. Для зручного використання є окрема IDE Qt Creator, яка додатково надає приклади різних проєктів, конструктори для GUI та зручні можливості тестування проєкту. Можуть виникнути складнощі стосовно встановлення нестандартних додаткових компонентів, яких немає в інсталяторі (до прикладу драйвер QMYSQL для використання MySql. Можна встановити з сайту та вимагає ручного налаштування та підключення до Qt).

1. **Наскільки зрозумілою та корисною була документація бібліотеки?**

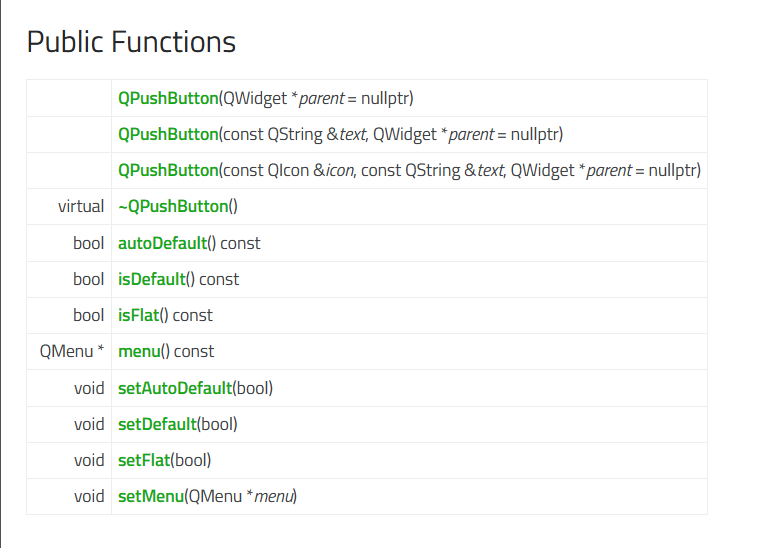
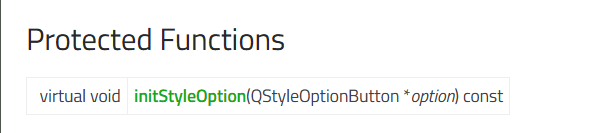
Документація даного фреймворку була доволі детальна та добре структурованою. Має поділ на розділи накшатлт роботи з GUI, мережевими запитами, мультимедіа і тд. У документації гарно розписані класи: його назначення, перелік публічних, захищених та приватних методів, опис параметрів…  
Приклад:   
Для прикладу візьмемо QPushButton, про який у документації міститься:

* Приклад створення кнопки:

QPushButton \*button = new QPushButton("Click me", parent);

* Обробка події натискання кнопки:

connect(button, &QPushButton::clicked, this, &MainWindow::onButtonClicked);

А також багато іншої корисної інформації стосовно класів

1. **Наскільки було зрозуміло, як саме використовувати бібліотеку, які класи/методи/функції використовувати для вирішення поставлених задач?**

Виходячи з пункту 4, документація є доволі детально розписаною. Фреймворк містить структуровані класи за окремим областями (GUI, бази даних, мережа…) Більшість методів класів мають зрозумілі назви (setText(), on\_button\_clicked(), hasMatch(), currentTime() та інші)

Приклад з проєкту:

Класи:

* ***QNetworkAccessManager*** (для відправки HTTP-запитів)
* ***QNetworkRequest, QNetworkReply*** (для роботи з запитами і відповідями)

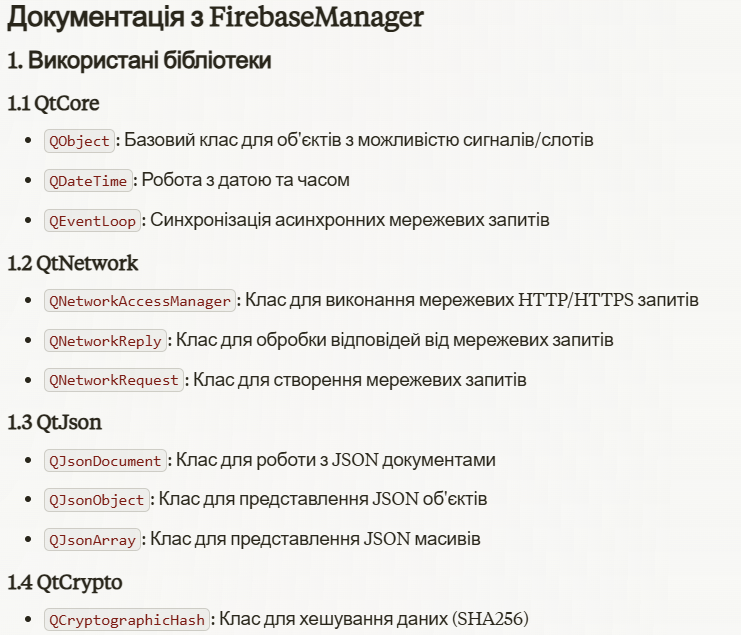
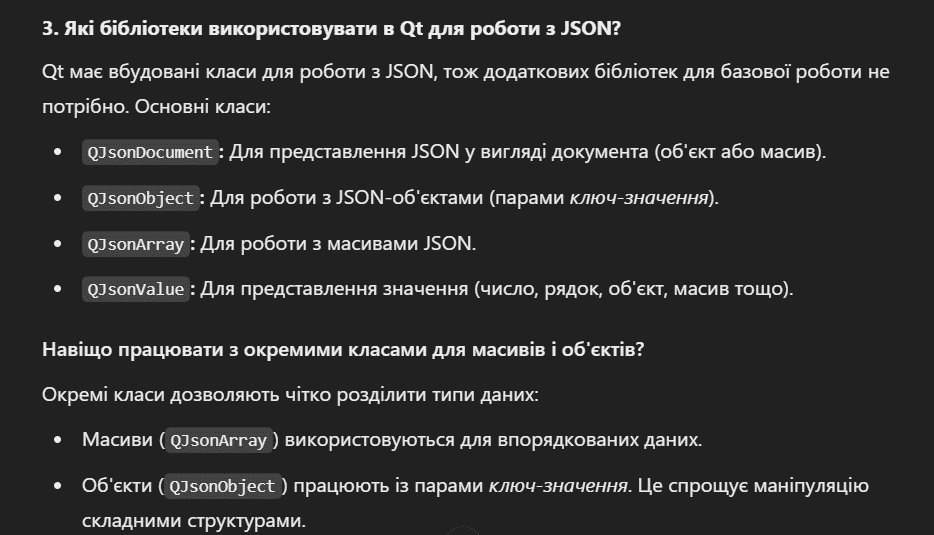
Методи:

* ***get(), post()***  (для отримання та посту інформації)
* ***readAll()*** (для зчитування всіх даних з потоку)

В проекті реалізований клас з використанням методів з вище наведених класів: ***DatabaseManager::synchronousRequest***



Деякі труднощі виникли при інтеграції FireBase (Rest API), але даний сегмент роботи був розібраний для вивчення за допомогою AI Chat-Gpt, який вказав на використання конкретних бібліотек, їх методів та як правильно функціонувати з ними.

На початкових етапах виникали деякі труднощі з коректним розумінням сигналів та слотів. Додатково не встановити драйвер для MySql, через щл був зроблений вибір під’єднати Firebase (що додатково надало можливості відійти від локального зберігання даних)

1. **Наскільки зручно було використовувати бібліотеку, чи не треба було писати багато надлишкового коду?**

Як було зазначено у попередніх питаннях, Qt має дуже великий спектр різноманітних оптимізованих бібліотек, що додатково допомагають уникнути надлишкового коду. Для прикладу можна взяти ті самі *layout,* що допомагають автоматично підлаштувати розмірність елементів, а не налагоджувати розміри вручну

Не дивлячись на чудову оптимізацію Qt, є специфічні випадки коли виникає багато зайвого коду. Загалом це при процесах кастомізації певних віджетів, у випадку проєкту це інтеграція Firebase, для якої потрібно додатково прописати код для побудови HTTP запитів, використання складних графічних, інтерактивних інтерфейсів (QGraphicsView) …

1. Наскільки зрозумілою була поведінка класів/методів/функцій з бібліотеки?

Завдяки тому, що Qt має досить чітку систему назв,розуміння більшості методів та класів не викликає великих непорозумінь і їх поведінка є відносно передбачуваною.

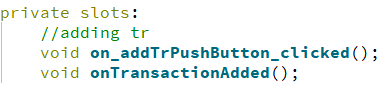
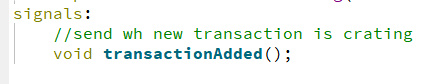


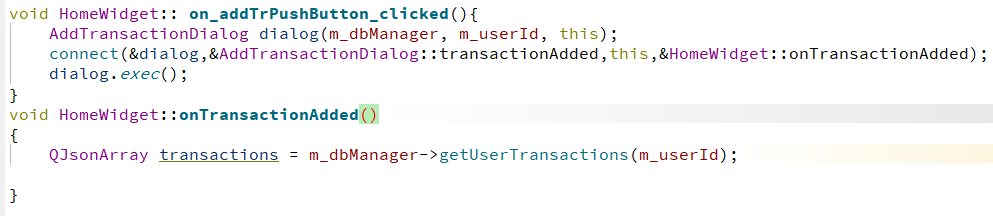
Для прикладу можна взяти метод валідації транзакції **AddTransactionDialog::validateAndAccept().** Методи перевірки полів: **…text().isEmpty(); currentText(); isChecked()** – є доволі тривіальними для розуміння. QMessageBox::warning – теж є доволі зрозумілим, про подальше отримання вікна з попередженням в разі виконання певних прописаних умов.

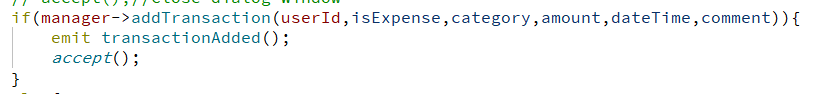
Менш простими для розуміння можуть виявитись більш специфічні методи, які потребуються для інтеграції складніших речей (хоча вони теж не викликають великих труднощів). Підсумувавши, можна сказати що класи, бібліотеки, функції даного фреймворку є доволі простими і інтуїтивно зрозумілими, але можуть виникнути деякі складнощі при створенні кастомних моделей.

1. Наскільки зрозумілою була взаємодія між різними класами/методами/функціями цієї бібліотеки, а також взаємодія між бібліотекою та власним кодом?

Як вже було сказано раніше, Qt має добре написаний функціонал, що дозволяє в більшості ситуацій без будь-яких проблем налаштовувати взаємодії між класами, методами і тд. Qt використовує зручний механізм сигналів і слотів, що дозволяє легко поєднати графічний інтерфейс з прописаним для нього функціоналом.





Можемо наглядно бачити на прикладі коду з проєкту, що при натисканні кнопки (так само певний сигнал) додавання транзкації створюється AddTransactionDialog:: dialog, після чого встановляюється з’єднання між методами. При успішному додаванні транзакції до бази даних, виконується emit який буде функціонувати для оновлення всього списку транзакцій.

Якщо розглядати зі сторони GUI елементів, можна також побачити, що завдяки ієрархічному поєднання елементів батьківські елементи керують дочірніми(позиціювання, відображення видалення).

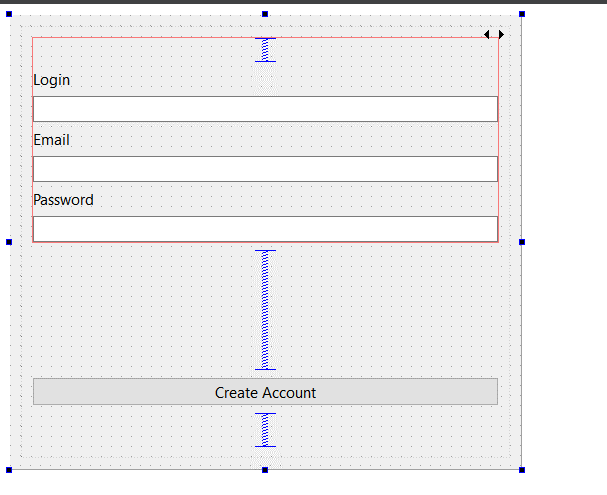
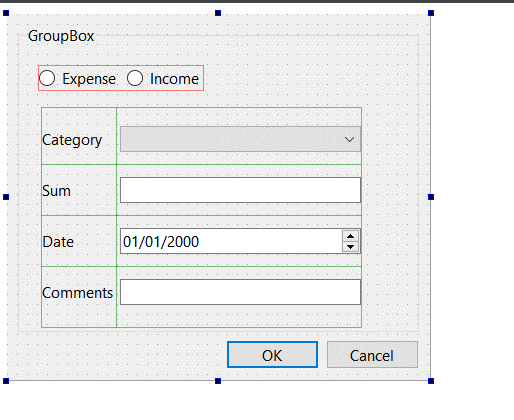
1. Чи виникали якісь проблеми з використанням бібліотеки? Чи вдалось їх вирішити, як саме?

Загальна проблема виникла при спробі встановленню драйвера для інтеграції MySql. Завантаживши потрібні компоненти та розформувавши їх по папкам, не вдалося під’єднати даний драйвер. Для вирішення підключення бази даних було вирішено інтегрувати Firebase, використовуючи додаткові Q-бібліотеки, що надало проєкту більше можливостей. Також на початку роботи з проєктом виникали певні труднощі з роботою у CMake файлі, а саме з коректним додаванням певних Qt модулів та їх підключенням до проєкту.

1. Що хорошого можна сказати про цю бібліотеку, які були позитивні аспекти використання бібліотеки?

Якщо підсумувати усі плюси, що були сказані вище- можна сказати, що Qt є зручним та гнучким інтерфейсом для створення програмних додактів, який має ряд суттєвих переваг:

* **Кросплатформеність**: після завершення роботи над проєктом, його можна буде компілювати для різних платформ, гарантуючи, що функціонал залишиться без змін
* **Документація**: як було сказано раніше уданий фреймворк має дуже добре детально та структуровану наисану документацію, що дозволяє розібратись з призначенням та використанням вбудованих класів та методів.
* **Механізм сигналів і слотів**: дозволяє без великих зусиль правильно налагодити з’єднання між функціональністю елементів на GUI та методами написаними для них.
* **Адаптовані Qt бібліотеки**: одним з найбільших плюсів є велика кількість модулів (Qt Widgets, Qt Quick, Qt SQL, Qt Network…), завдяки яким робота з різними аспектами програми стає в рази легшою
* **Робота з інтерфейсом**: наявність конструкторів допомагає швидко створити інтерфейс додаючи статичні елементи, та одразу бачити картину. Дана можливість дозволяє швидко вносити візуальні зміни, що дуже полегшує роботу. Як приклад з проєкту-це вікна логінізації та додавання транзакцій.

\ 

* **Гнучкість**: легка інтеграція бібліотек та API. ПідключенняFirebase через Rest API та бібліотеки для роботи з JSON це доводять.

1. Що поганого можна сказати про цю бібліотеку, які були негативні аспекти використання бібліотеки?

Незважаючи на потужність даного інструменту, все таки є деякі недоліки. Якщо проєкт вимагає інтеграції платіжних систем або складних графічних деталей безкоштовної версії застосунку буде недостатньо для реалізації даних речей.

Також одним з мінусів є неможливість редагування згенерованого програмою коду графічного інтерфейсу. Якщо потрібно було внести незначні зміни до створених форм, це потребувало додаткової роботи, адже редагування файлів .ui вручну може викликати конфлікти. Якщо говорити про кросплатформеність, то програми створені на Qt, відповідають системному оформленню на деяких платформах, що може ускладнити створення певного «універсального» дизайну. Qt є дуже зручним фреймворком, який не викликатиме проблем при написані маленьких проєктів, але якщо це набуватиме більш масштабного рівня (корпоративні проєкти, великі стартапи) будуть виникати додаткові бар’єри.

1. Якби довелось вирішувати аналогічну задачу, але вже враховуючи досвід використання в цій лабораторній роботі, що варто було б робити так само, а що змінити? Можливо, використати інші бібліотеки, чи використати інші можливості цієї бібліотеки, чи інакше організувати код, чи ще щось?

При вирішені аналогічних задач, я залишив би використання даного фреймворку, внісши деякі зміни. Перш за все, приділив би більше уваги рефакторингу коду для досягнення його більшої простоти та читабельності. Це дозволить легше підтримувати та масштабувати проєкт у майбутньому. Крім того, я б активніше використовував можливості QML для створення сучасного і привабливого інтерфейсу. Це не лише покращить візуальну складову програми, але й зробить її зручнішою для користувачів.