Team Liam McAteer
Rachel Lee
Bonita Ryda
Afton Lim
Mentor Livia Krstic
Client Ryan Ashton

A Few Quiet Yarns Android/ IOS App

AUT

About the Project

Background:
A Few Quiet Yarns, or AFQY for short, owned by Ryan Ashton, is an initiative which runs networking events that are primarily aimed at CEOs and CTOs. The networking events aim to form better and more meaningful connections with the others that are attending by allowing the participants to “Meet the person, not the suit”. The events are different from other networking events in which there is to be no selling of products or services. This ensures that the attendees form connections over interests outside of work.

Rationale:
The events have been running for 13 years, and over this time, they have gained more attendees. This makes it more difficult to find specific people to connect with. AFQY needs a solution to help the attendees to find meaningful connections based on similar interests. Initially, AFQY used the LinkedIn event feature to promote and encourage conversation before events, however this feature changed and it is not useful to AFQY anymore.

Our Solution:
Ryan suggested that some kind of application would be useful to improve conversation before and after AFQY events. After some research and refining the project scope, we proposed that a cross-platform mobile app would be suitable. The application front end was built using Google's cross-platform mobile framework called Flutter, and the back end was hosted on Firebase which is a mobile back end as a service (MBasS). The MVP of the project included an attendee match-making system and chat. It was also proposed that the app shall support sponsored business spaces if the time allowed.

Acknowledgements:
We would like to thank our client Ryan Ashton, our mentor Livia Krstic, the BCIS R&D Project team and the AFQY community for supporting us throughout our project.

Process

Methodology:
For this project, both us and our client wanted results to be delivered iteratively. So, we decided to use the scrum framework which is part of the Agile methodology. Scrum iteratively delivers new software in short bursts called Sprints (Collab Net, 2020). During the project, our team had a total of five sprints, each lasting three weeks long. This was to ensure that the client was able to see our progress more frequently and be able to give feedback or request some changes before the final product has been made. Over this time, the product scope was shaped towards what the client wanted. At the start of each sprint, we held sprint planning meetings with the client to define the scope for the next sprint. We also held planning poker sessions to determine the difficulty of user stories as a team. At the end of the sprint we held sprint review and sprint retrospective meetings to review the product and team performance.

Planning and research:
Research was conducted before development to ensure that the product would meet the brief and the expectations of the client. Key research points include; deciding between developing a Progressive Web Application (PWA) or Native Mobile Application, choosing a framework for developing a native mobile application (Flutter or React Native), and choosing between Firebase or AWS Amplify for our backend.

Development:
Our team's codebase was hosted on GitHub and we made use of the git flow branching strategy to manage our branches. Git flow has separate branches for released code, in-development code, and features. When we were finished with a feature a pull request was created on GitHub and the team would test the feature through acceptance testing.

References:
Collab Net. (2020, May 3). What is Scrum Methodology? https://resources.collab.net/agile-101/what-is-scrum
Li, D. D., & Liu, X. Y. (2020). Research on MVP Design Pattern Modeling Based on MDA. Procedia Computer Science, 166, 51–56. https://doi.org/10.1016/j.procs.2020.02.012

Quality Assurance

Code Review:
Each feature developed by the members are pushed into different branches and must be tested by another members, on their local machine and ensure sure the feature is working before merging it to the main branch. This ensures that we have followed coding standards and that we improve code quality.

Acceptance Testing:
At the end of development of each user story, the assigned tester has performed acceptance tests to ensure the feature satisfies the acceptance criteria and to identify and resolve any errors.

Usability Testing:
Usability testing was conducted during the client meetings and the team made appropriate changes based on the client feedback. Beta version of the app was also tested in an AFQY event which has provided the team with further insight into improving the app.

Challenges

Non-Technical
Pandemic: Covid-19 caused some delays throughout the project timeline by making it more difficult to conduct mob programming sessions and to hold sprint reviews with the client. Throughout the pandemic we kept in contact via Facebook Messenger and through voice calls. We found that voice calls helped with programming, however they weren't as good as mob programming sessions. In our client meetings, we showcased the app via screen share.

Scope: The initial scope was very extensive for the timeframe of the project, so we had to hold many weekly meetings with the client to refine the scope to a more manageable size.

Technical
Mobile Development: None of the team had any experience with Mobile development, including Flutter and Firebase. To overcome the lack of skills, we used the Covid-19 lockdown as an opportunity to upskill in all areas that were lacking. As a team we decided to share helpful resources for upskilling and we helped each other when we had issues.

Artefacts and Results

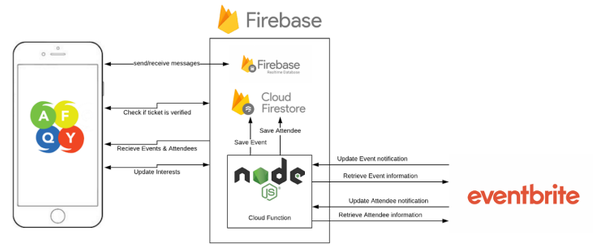

Figure 1: Technology Interaction Diagram

Artefacts
Meeting minutes, cross platform mobile application, source code and a handover document that includes technology interaction diagram, development environment, and instructions on analytics and how to manage users on firebase.

Architecture
The team followed Model View Presenter (MVP) architecture. The three components are:
- *Model* represents the data layer
- *View* represents the display of the app
- *Presenter* represents logic layer, acts as a bridge between Model and View connecting the two. Grabs user input from View, filter data with Model and updates View based on the changes. (Li & Liu, 2020)

Any communication between View and Model is performed by the presenter and they do not directly interact each other. The architecture was chosen for its code reusability and maintainability resulted by this decoupling.

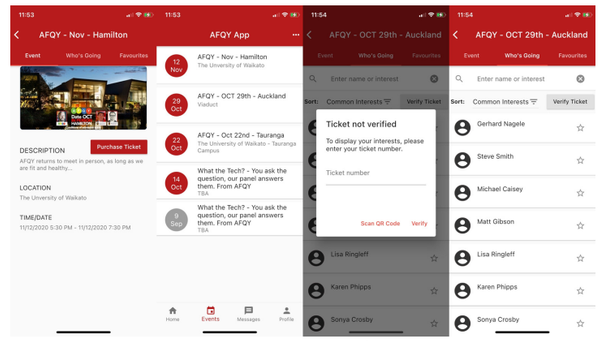

Figure 2: App Screenshotx