Handover

Final Customer Delivery

During a meeting with the customer on 20180511 it was decided that final delivery of the RedRiver chat product will take place during week 22. This final delivery will include:

- A copy of the source code
- A downloaded version of the our documentation wiki

The customer also has, and will continue to have, access to the GitHub repo where both the source code and wiki are maintained.

The most current version of our product can be found here (https://clientredriver.azurewebsites.net/)
(Test users are available: username: sTestUser{x} password:sTestUser{x} where x runs from 0-9)

We have expressed to the customer that we are willing to answer a certain amount of support questions concerning functionality, implementation etc. should they wish to further develop the product.

Achieved Functionality

In our opinion the project has gone well, without any major setbacks or significant technical problems. Any technical problems which did arise, for example CORS problems within the hosting environment or differing SignalR versions, were dealt with in a timely manner and did not cause undue delay to the project. However, the scale of the project was sizeable from the outset, and we have been candid with the customer as to what we believed was possible within the project's timeframe. In this regard, our predictions were accurate and we have implemented only a part of the total required functionality.

We deliver then a working client-server-database architecture where the user can:

- register (including email verification), login/logout, update details (including avatar), change password
- add/delete friends
- create, join and leave chat groups
- send private (HTTPS) chat messages with chat groups, where no chat messages are stored on the client device.

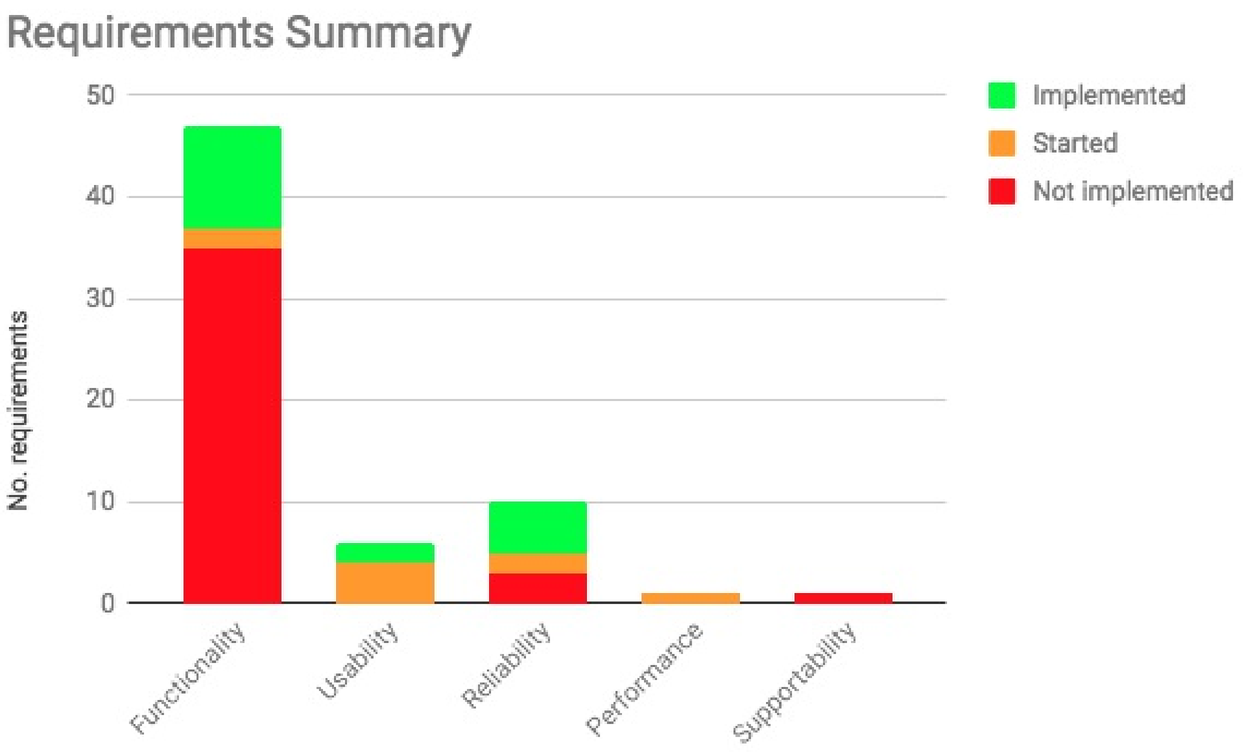
We also deliver limited video chat functionality.

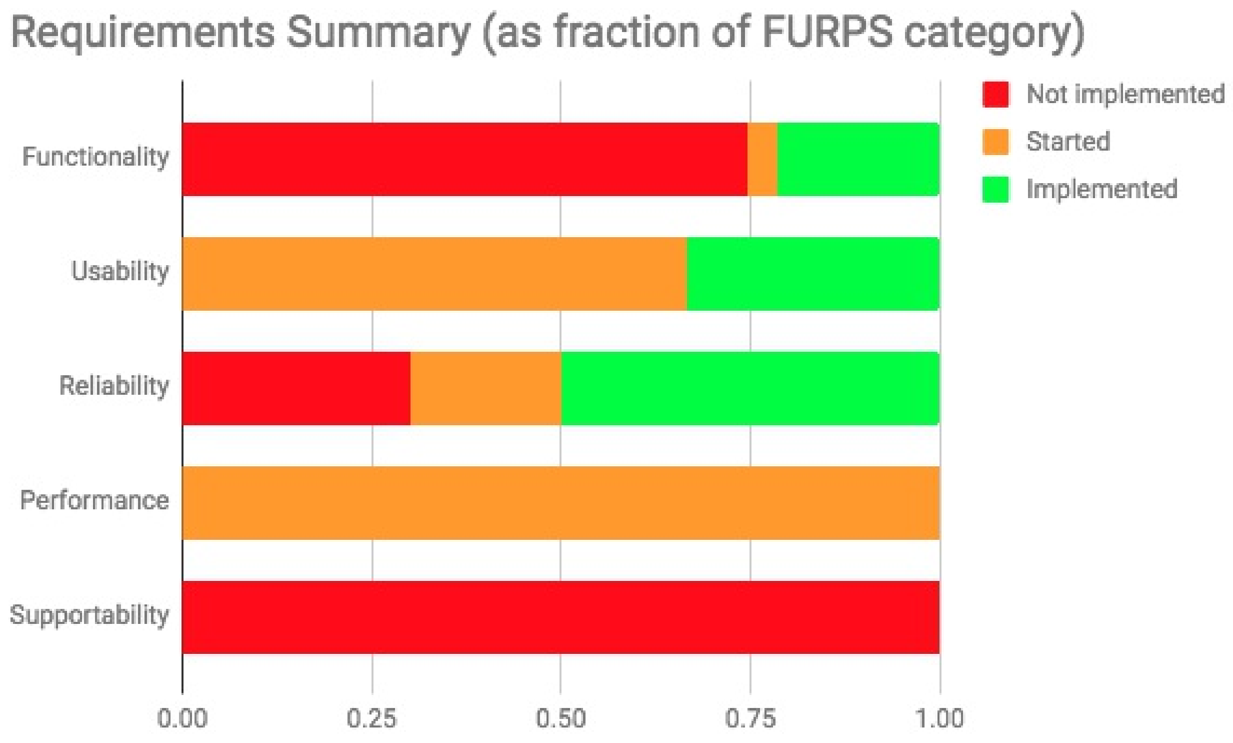
Our entire product backlog can be found here (https://docs.google.com/spreadsheets/d/1fQBQyhCi1xqPi2szUyqQn_dRg4coN8g6ybaqaGmtYyc/e

Implemented requirements are summarized in the following table:

Status	Functionality	Usability	Reliability	Perform/e	Support/y	Total
Not implemented	35	1	3	1	1	41
Started	2	5	5	0	0	12
Implemented	10	0	2	0	0	12

Sum	**47**	**6**	**10**	**1**	**1**	**65**





Further development

We believe that both our code and documentation set a solid ground for further development, both in terms of the available source code and documentation. Going forward, we recommend the following:

- SignalR at present uses an in-memory dictionary to store current connection details. This is fast and suitable for lower volumes of users. Should the number of users increase greatly then this might not to be the best solution; instead the architecture must be scaled out. Documentation concerning this is available here (https://docs.microsoft.com/en-us/aspnet/signalr/overview/performance/scaleout-in-signalr).
- In our project a Microsoft SQL database was used for all purposes. The pros and cons of this should be evaluated against document databases such as MongoDB. Should the decision be made to continue with a relational database, the tables used should be normalised. For example, a user can enter into a friendship with another user; at present this is stored as two entries in a friendship table (one friendship in each direction). Alternatively this might be seen as a single database entry between two users, thereby offering substantial space savings in a system with many users.
- Encryption: Our project featured communication over HTTPS and built in database encryption, and no data is stored on the client device. However, database admins can still see chat logs. End-to-end encryption would solve this problem. Our initial researched that this would be difficult to implement in a browser based system, but some form of partially encrypted logs is fully possible.
- Video chat is available, although it is not fully tested and may not work on all devices. Going forward, this should be further investigated and well-tested so as not to lead to an adverse user experience.
- SendGrid is used for sending verification emails to a newly registered user. For it to work properly after the handover it needs to be set up. How to do that is described here (https://github.com/jimmybengtsson/grupp03-redriver/wiki/9.8-SendGrid).