**Functional Requirements**

1. The user shall be able to login using a pre-existing account from a third-party vendor such as Facebook or Google. There will be no need to register a propriety account with *Chatter*.
2. A profile page shall display all of the user’s viewable public details such as birthday, date joined, country, number of friends and a picture. Information within this screen will be editable.
3. The user shall be able to view a video stream of their friend’s webcam when engaging in a chat with them.
4. The user shall be able to converse with their friend via voice communication when engaging with a chat session with them.
5. The user shall be able to converse with their friend via text when engaging in a chat session with them.
6. A chat history window will exist to store all of the previous message that were sent between the user and their friends.
7. There will be a facility to allow the user to add new contacts as well as deleting existing contacts
8. There will be a facility to search for friends so that they can be added to the contact list

**Non-Functional Requirements**

1. All functionality for chatting with contacts will appear on a single page: video stream, chat feature, history, friend list, add/remove friends
2. The application will be responsive and therefore will render appropriately for different screen sizes including tablets and smart phones
3. All client side features will be tested using an appropriate test strategy and framework
4. The application will be limited to three pages: login page, profile page and main chat page

**Technical**

1. Login – Oauth (oauth.net). This open standard will allow users to login to the application using credentials from a third party provider.
2. Communication protocol – PeerJS (peerjs.com). This will allow the use of API keys to realise a peer-to-peer connection.
3. Responsiveness - Bootstrap (getbootstrap.com). This will be used for making the web application responsive for rendering on various devices. The default bootstrap theme will be used for the application.
4. Data Storage – As this implementation is not heavily based on server side architecture, simple file based databases will be used
5. Testing – Jasmine (jasmine.github.io). Client side features will be tested using the Jasmine JavaScript framework.
6. Backend – ASP.NET. The web application will be managed using Visual Studio and will render from the default.apsx page.
7. Endpoints – Web API will be used for REST endpoints.

**Under Investigation**

1. Chat room features
2. Filesharing functionality
3. Use of AngularJS and BackboneJS