**Security**

**Web app & Android:**

All members need to register and upon doing so create a password. This password is hashed using jbcrypt, a key derivation function based on the blowfish cipher. During hashing, a salt value is added to the password to protect against rainbow table attacks. Bcrypt is an adaptive function in that over time the iteration count can be increased to make it slower. This increases its resistance to brute-force attacks. The Jbcrypt function offers an optional parameter that determines the computational complexity of the hashing with default at 10. Our system has this set to 12 to safe guard against the ever increasing speed of computers.

Once a member registers, only the salted and hashed password is saved in the database and the original deleted.

**Web App:**

To secure connections between client and server, the system implements TLS 1.2. This is used as soon as a member logs in and every web page while logged in to ensure security of personal data.

Once a member is signed in, a session variable is set to hold some of the member’s details. On every subsequent page visit then a filter is run to check if the member is still logged in using this session variable. This ensures that once the member actually clicks the logout button or the session expires then any of the previous pages visited cannot be accessed again unless they sign in again. This is especially useful if accessing the website from a public system.

**API:**

The API uses Basic Authentication with TLS. This means that on every web service request, depending on which web service is requested, the requester needs to be authenticated. To achieve this the request attaches a header to the https request with the members email and hashed password encoded in base64. To ensure that the members email and password cannot be accessed during transport, the connection is secured using TLS 1.2.

Things that could be done better:

* On a web service request, the web service should only allow a member to access their own data. Currently any member can access any data as long as they are a registered member. Many more checks would need to be implemented to ensure this.
* Basic authentication felt right for this application as the application won’t hold highly personal data but another possibility would be to use OAuth 1.0a, OAuth 2 or Shared Key Authentication such as Amazon’s S3 web services.

**RESTful Web Services**