4.1

For this part, we specify the email\_scope to be “email” so that we have access to the client’s email information and also the principle of least privilege is satisfied.

4.2

Before calling getPeopleInfo(), a new, similar function validateToken() is used. An XMLHttpRequest is created to get the token info for a particular access token. On a good request (status 200), the request’s responseText is parsed into JSON and the value for the audience key is compared with the client ID. If it matches, the given access token was actually for that client and getPeopleInfo() can be called to display profile information. Otherwise, no profile info is shown and the validity check is shown to have failed.

4.3

For this part, we implemented drive() to handle the authentication process. We first check if the client already has an authentication code, if not, we redirect them to the Google Login API which will prompt the client to log into their account and grant access based on the SCOPE we provide. Once the client is successfully authenticated, we retrieve the client’s credentials by providing their authentication code and store the credentials in a file locally.

4.4

Extending drive() after 4.3, the access token from credentials is applied to an Http object. A Google+ API service object is created to get the person info of the authenticated user (‘me’) and write it to profile.out. Then, a Google Drive service object is created to upload (insert) a file with the body containing specified metadata and the media\_body containing profile.out.