**Software Architecture**

**Makepassword.js**

* Inputs: Plaintext password.txt file
* Outputs: Encrypted password.enc.txt file

**Makepassword.js** (transformData() function)

* Inputs: line of login information formatted as “email:password” in plaintext
* Outputs: Hashed line of login informated formatted as “email:hashedPassword”

**Password.js**

* Inputs: 3 Arguments (filename, email, password)
* Outputs: Either True or False, depending on whether the specified email and password exists in the file

**Utility.js** (readFile() function)

* Inputs: 1.)Name of file to read
* Outputs: Array of each line inside the file being read

**Utility.js** (writeFile() function)

* Inputs: 1.)Array of lines to add to a file. 2.)Name of file to write to
* Outputs: File containing lines designated in the array in addition to what was previously there. If the file never existed, it is created.

**Utility.js** (hash() function)

* Inputs: accepts string as main input
* Outputs: Hashed string using sha256

Below is the software architecture diagram. The utility module is utilized by makepassword and password to both create encrypted password files, then use them to validate email/password information. Password.js is tested via an acceptance test *acceptance.bat*. And makepassword.js is tested through makepassword.test.js