**Project Summary**

The project, or experiment, I am proposing is to create a hashing function that can hash and store hashed passwords. The goal is seemingly simple: create a hashing algorithm (semi-based on SHA) to generate a hashed password that while seemingly random, is actually deterministic, and cannot be reversed. The goal really, as this is a high level problem, is to have something that’s working.

**Methods**

I have done a lot of research into this problem and discovered methods known as salting and bit shifting. We start with 4 initial states, hexadecimal values. We put add these states to our salted password and throw them into the ‘washing machine’ (as Computerphile refers to it), using bitwise operations and bit shifting and bit rotations to make it look like we are ‘randomly’ moving things around, though this process will actually be repeatable. Once this process has been done a number of times, we readd the new values to the original state, to get a new hex value which will be the hashed password. Then at the location of the (hashed password -> base 10)mod1531, we will store the hashed password, the username, and the salt. The salt and username can be left as plain strings, because without the password they are meaningless (someone could try brute force methods but it would be tricky).

**Case Analysis**

For this project there will be one main use: to hash and store passwords into a hashtable. To test this we will have two types of users, and regular user and an admin. Users create a login and can login to their account, while

**Data Design**