MENG PROJECT, FA20	**Slang Full Stack Development**
Name: Jonathan Gao	NetID: jg992

Prototype Scope

For my M.Eng project, I will plan on delivering a prototype of Slang by the end of the semester. While this will be a working prototype, I am aiming mostly for a proof of concept. I will define the scope of the prototype below.

For the prototype, I am mostly focusing on implementing enough of the backend to enable preliminary user testing. This includes, but is not limited to, the following services.

1. NoSQL database for scalable storage
2. API to fetch, update, and delete items from storage
3. Stateless functions that will modify the items in the database
4. Basic user authentication and authorization with access control for user information
5. A skeleton matching algorithm that will give a list of matches per user and/or group

NoSQL database

This database will serve as the storage of all information for Slang. Currently, I am planning on using DynamoDB to store all information per user. I can define my own fields for each row in DynamoDB, where I can add/modify the data according to what the app needs.

Currently I plan on having each row with these fields

1. Name
2. Phone Number
3. Email Address
4. Profile (meta information about the person)
5. List of suggested matches
6. Metadata of profile (time created, number of matches, etc.)

Each row will be indexed by a profile ID, which is unique for each user. This will need to be implemented in conjunction with user authentication. For now, I plan on just implementing this database. In the future, this can be expanded with more tables for information such as suggested events, suggested friends, and information from linked social media accounts.

API

To actually interact with the database, we will provide an API to fetch information. The API will define calls for modifying and creating data in our database without exposing implementation details. Some of the functions implemented will be as follows. These will be defined with REST API convention, but may be adapted to use AWS GraphQL.

- GET profile
- POST profile
- PUT profile
- DELETE profile

I am thinking that on each GET request, we reprocess our matches before returning the profile. Or, there is a S3 trigger that can run after new data is input, meaning that it might be possible to process matches as we get new data.

Functions

The API gateway is an interface that will invoke the Lambda Functions. For each of these API operations, we will have some Lambda Function that can process the operation and return the expected result. These are stateless functions, which will scale effectively.

Authentication and Authorization

User authentication and authorization will be handled through Amazon Cognito, which is a authentication and authorization service offered in AWS. I am pretty unfamiliar with this realm, but user authentication is fairly consistent across most web applications. I will need to figure out how to use authorization to limit access to database keys.

Matching