Christrina:

Thank you Sarah. So my name is Christina and I’m going to cover our design architecture and what technologies and techniques were used to build our project. GitGoing is implemented as a Client-Server pattern where the server layer will be hidden and solely used to listen to ongoing requests from the clients on the backend. A Client-server architecture of a computer network allows many clients to request and receive service from a centralized server. Client computers provide an interface to allow a computer user to request services of the server and to display the results the server returns. As shown on the screen, this is our design architecture for GitGoing.

In this case, the Client layer will be a Web browser on the frontend built through HTML, JavaScript, and React Frameworks. The output of these will essentially be what the client or user can see which is the actual login/registration system, notification system, and many other pages that we will be implementing. We will later see how these dependencies work and how they are used to build the project.

The client layer will obviously be connecting to the server layer through the internet. The Server layer will be hosted through AWS, and built using API gateways and Python/Flask frameworks. Both components works to respond to ongoing requests from the server side. Cognito, a user authentication tool within AWS will be used to enable authentication to requests. To process these requests, GitGoing uses Lambda and a MySQL database to do so. Altogether, two parties (client and server) work collectively to process, edit, store, and deliver an operable system to clients whether it be on a PC, smartphone, or laptop.

Here are a list of technologies, and techniques we used to develop our project. Each dependency is largely intertwined with the other to ensure a fully functioning system. As I mentioned before, we used react to implement most of the frontend components alongside javascript and html. React is a UI library created by Facebook that helps you create interactive web applications made up of components which is written as a plain JavaScript text. Through react, we were able to create a successful login/registration system as well as a notification component.

The login/registration system is also paired with cognito, which is an Amazon tool that provides authentication, authorization, and user management for web and mobile apps. For example, users can sign in directly with a user name and password, or through a third party such as Facebook, Amazon, Google or Apple. However, for this particular project, cognito is specifically just going to be responsible for pushing new user information into the database and ensuring authorization.

Information and data from the user will ofcrouse be stored in a MYSQL database for reference. We spent a lot of time developing a way to add, retrieve, and delete such data by creating an API using node.js and express. Node.js is an event-driven development platform for executing JavaScript code and Express is a flexible Node.js web application framework. Using Node.js and Express, we were able to create an API to communicate with with the database. The database and web application itself is all going to be hosted through AWS and we hope to continue to use these technologies to expand our project. Now, I’m going to hand it off to Michael to talk about some code that was developed through the use of these technology tools and what we’ve accomplished from them.