Christina: {  
Hi everyone, My name is Christina. Today, I’m joined with my teammates Michael, Travis, and Sarah to talk about Amazon Cognito. Since our project is to develop a web-based code review tool integrated with Git, our site uses react javascript to build the frontend. In doing so, we require an authentication system to ensure user safety and security, particularly during user registration. These calls are defined in our javascript code through the AWS service Amplify, which is in turn paired with Amazon Cognito. Through these services, we may address these needs and easily include other related services, such as the simple email service, and our simple queue service which we will further explain later on.

Amazon Web Services, also known as AWS, offers a wide range of remote cloud services such as database storage options, computing power, content delivery, and networking among other functionalities to help organizations scale up. In short, AWS provides the building blocks for developers who want to create apps with real-world production-ready user tools. Of particular importance to us, there is a tool-chain called Amplify which provides services that can be used together, or individually, to help front-end web and mobile developers build scalable full-stack applications. This includes a wide variety of open-source libraries, drag-and-drop UI components developers can use, or, as in our case, user authentication and security.

The Amplify Framework communicates with Amazon Cognito to provide user authentication and security. Amazon Cognito is a robust user directory service that handles user registration, authentication, account recovery & other operations. Additionally, it provides simple user identity and data synchronization services that helps you securely manage and synchronize app data for your users across multiple devices. Cognito makes this process easy by allowing you to use a single user registry to authenticate users across multiple authentication types.

As you can tell, Amazon Cognito is an integral part of our site, providing us with user registration and security when paired with Amplify. And with that, I will hand it off to Michael to go further in-depth about Amplify and how it all ties together.

}

Michael: {

Thanks Christina. Let’s take a moment to talk about what Amplify actually is: Amplify is an open-source toolset which is designed to make it easier to set up backend applications through Amazon Web Services. It provides the user with a command line interface which can be used to to provision backend services, such as the Authentication service we’ll be showing you, view project status, mock testing environments and more. Keeping with the theme of accessibility and user friendliness, some components also have pre-built UI components, so you can focus on implementing other parts of your application rather than having to write login and other UI components from scratch.

Let’s take a deeper look at the command line interface and see what it offers us. First let’s look at some of the common commands, amplify init takes us through the setup process for a project, you can see it asks for information about your IDE, and the type of app which is javascript here. Other options include iOS and Android and Amplify does the setup for each accordingly. Amplify add is how you add backend services for your application, the one we’ll be showcasing in our demo is Authentication, which communicates with Cognito to put user information into a user pool, but you can see the setup process for a RESTful API service here, and there’s other categories as well. Amplify pull you can think of similarly to a git pull, where you have multiple people working on the project and one person might change what services are provisioned. Then everyone else working on that project is gonna have to pull those updates before making other changes. Amplify mock is a useful command for mocking storage and GraphQL API resources, although REST APIs aren’t supported yet.

As for categories, authentication is what we’re showing you today, but there’s also the API categories I’ve briefly mentioned, the storage category which has built-in support for Amazon S3, and half a dozen more categories.

And before I pass it off to Sarah to talk about Amazon’s Simple Email Service, Simple Notification Service and Simple Queue Service, I just want to talk about Amplify’s pre-built UI components real quick. These are designed for quick implementation, but as you’ll see they’re not necessary. There’s UI components for multiple frameworks, including React, Angular and Vue, although I’ve only worked with React. In the Auth category, here’s a few examples of the SignUp component, the SignIn component, and the ForgotPassword component. There’s some pre-built components for other categories as well but a lot of what I’ve seen has been focused in the authentication category. Hopefully that helped clarify somewhat what Amplify is, but now Sarah is going to talk more about some other services utilized in the authentication process.

Sarah: {

Thanks Michael. Hi everyone.

As previously mentioned by Michael, we’re gonna look into 3 other services from AWS.

The first one is the **Amazon Simple Email Service** known as **SES** which is a pay-per-use service that allows us to build an email functionality into an application that we are running on AWS. SES provides built-in notifications for successful and unsuccessful email deliveries and complaints. It can also send bounce-back and complaint messages by email or through Amazon Simple Notification Service that we’re going look into in a second.

But first let’s mention that SES supports different email use cases such as:

**Transactional emails** consisting of immediate emails from your application to customers, like purchase confirmations or password resets.,

Then we have **marketing emails** consisting of product promotion emails and services such as special offers and newsletters.

At last **mass email communications**: including notifications and announcements to large communities, and tracking results using configuration sets.

The second service is:

**Amazon Simple Queue Service or SQS** which is a distributed message queuing service intended to store messages in transit between computers.

So, how does SQS work?

SQS provides an API endpoint to submit messages, and another endpoint to read messages from a queue. Each message can only be retrieved once, while you can have many clients submitting messages and reading messages from a queue at the same time.

And then at last we have

The Amazon Simple Notification Service (or SNS) which is a cloud service for coordinating the delivery of push messages from software applications to subscribers. In other words, it is a network, where subscribers can subscribe to topics and will receive messages whenever a publisher publishes to that topic.

How does SNS work?

Every SNS topic has a set of subscriptions. Once a message is published to a topic, SNS handles distributing the message to all its subscribers. Those subscribers can be AWS Lambda functions and SQS queues, mobile push notifications (like iOS, macOS, Android, and Windows devices), HTTP endpoints, email addresses and mobile phone numbers capable of receiving SMS messages.

And to publish a message to an SNS topic, a message producer must use the SNS HTTP API. Once the message is published, all subscribers receive a copy of the message over the channel through which they established their subscription.

While SNS and SQS are related as messaging services, the main difference between them is that SNS implements a "push" architecture—and all subscribers receive the messages that are published to a topic, while SQS implements a "pull" architecture, in which clients pull messages from a shared queue.

Now we are going to at the demo that will be presented to us by Travis.

}

Travis: {

}