CS 495 Design Day

AWS Visual Configuration App

October 27th, 2020

AWS Visual Configuration App: Team Members

- Joshua Kennedy
- Wyatt Lawrence
- Nick Hammerstrom
- Benjamin Furlani
- Noah Connolly

Client Information

- Trey Gourley
- Accutech Systems, Muncie, Indiana

Business Requirements

- **BR1: Reduce the time the business spends on finding desired information about AWS services.**
- **BR2: Visualize the business's multiple AWS services in one place, getting rid of the need to switch among various AWS consoles.**

Mentor Feedback:

- Recommended we mention the visualization of connections between AWS services somewhere in the business requirements. (Mentor later saw the connections mentioned under functional requirements, and he said that's fine.)

Client Feedback:

- Recommended adding a third BR: "Enable the user to change high-level aspects of AWS services." Example: rename an instance of ECS. (We will most likely change the project based off this feedback, if not as another BR, than as another functional or nonfunctional requirement. We haven't discussed it yet.)

Use Cases

Actors

- Users
- Developers
- AWS APIs
- Server
- Admins

Use Cases

- UC1: Find desired information
- UC2: Be notified of problems
- UC3: Show and hide information
- UC4: Toggle among views

Mentor Feedback:

- Recommended added use cases for developers and admins, since they are listed as actors

Client Feedback:

- No critiques, said it looked good

Functional Requirements

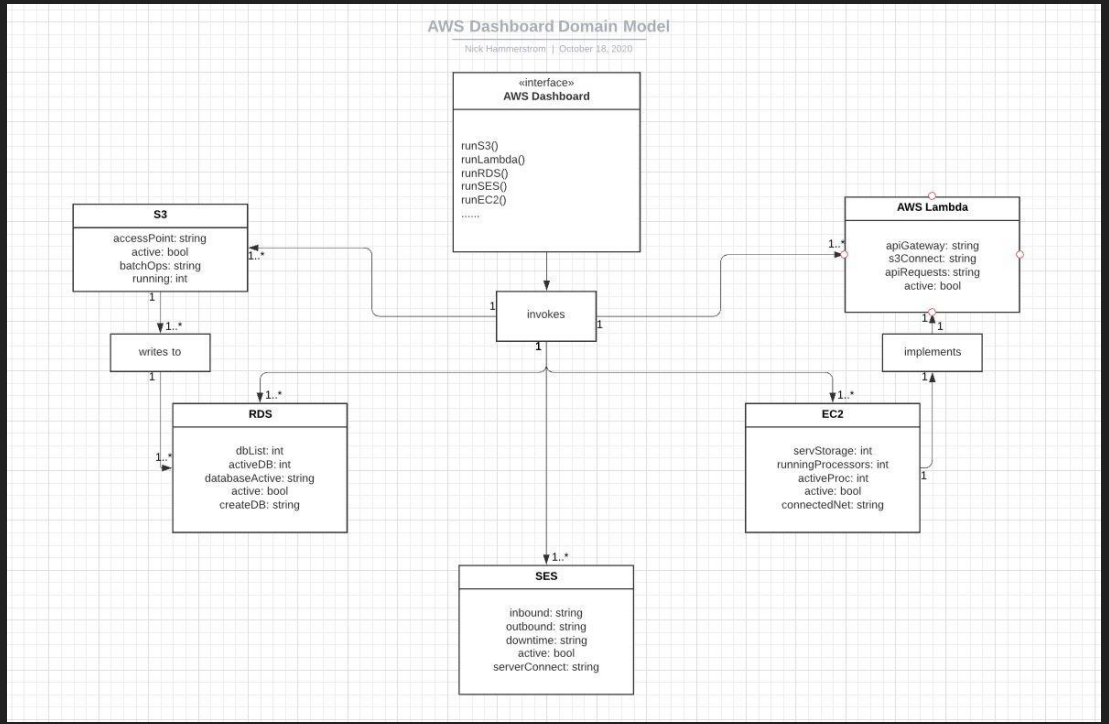
- **FR1: HIGH BR1 & BR2**
  Software must connect to and get data from AWS APIs
- **FR2: HIGH BR1 & BR2**
  Software must display information received from AWS APIs
- **FR3: MEDIUM BR2**
  Software must show how AWS services are related
- **FR4: HIGH BR2**
  Relationships between services will be annotated (so user can understand the relationships)
- **FR5: HIGH BR1**
  Errors will not compromise the software
- **FR6: MEDIUM BR2**
  Software allows user to show and hide details
- **FR7: LOW BR1**
  If there are any changes to the data/details of the services, software will display this automatically (not requiring user to hit a refresh button)
- **FR8: HIGH BR2**
  Software will use .NET CORE framework (Recommended by client, and decided by us.)
- **FR9: HIGH BR2**
  Software will use SyncFusion's WPF packages as needed (Recommended by client, and decided by us.)
- **FR10: HIGH BR2**
  Software must be a desktop application (Recommended by client, and decided by us.)

Nonfunctional Requirements

- **NFR1: LOW BR1 & BR2**
  If there are any changes to the data/details of the services, the software will display the new information within 5 seconds
- **NFR2: MEDIUM BR1**
  Software response time will be no longer than 2 seconds
- **NFR3: HIGH BR1**
  Software can be installed and uninstalled with a wizard
- **NFR4: HIGH BR1**
  Software should prevent unauthorized people from accessing the AWS information
- **NFR5: MEDIUM BR1**
  Software can be used on Windows
- **NFR6: HIGH BR1**
  Software is available as long as its dependencies are up and running.
- **NFR7: LOW BR2**
  Software will include a help menu
- **NFR8: HIGH BR1**
  Error messages will be specific, and recommend a future action for the user
- **NFR9: MEDIUM BR1**
  Software will be scalable: can accommodate more traffic, data, servers without compromising the software performance

Domain Model

- Classes: EC2, SES, RDS, AWS Lambda, and S3
- Main focus is going to be on EC2 and load balancer services.
- In theory, this domain model would be indicative how every service would interact with the dashboard
- A shared interface approach will be implementing all api information from each service



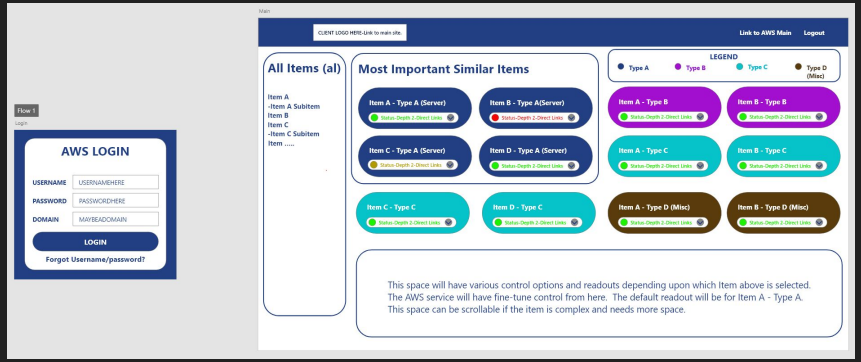
TechStack

- Microsoft .NET CORE framework
  - This utility is cross platform and had good ties with AWS.
- Programming Language: C#
  - This programming language is compatible with AWS and gives access to frameworks we can use.
- Microsoft Visual Studio
  - This is the preferred programming environment for the project. It is a fantastic editor for C#.
- AWS API
  - We will utilize AWS API as a “front door” to access important AWS data and functionality.
- Syncfusion Control Library
  - Syncfusion has good diagramming tool and has a free claimable licensure.
- Atlassian
  - This our clients team management and organizational tool.
- Github
  - This is our teams and class code and design organizational tool.
- Slack
  - This is our teams main form of communication for project and meeting planning.

Prototype

Core Concepts:

- Eliminates clicks to get where client wants to go.
- Still allows full control of all AWS items.
- Presents both a diagram and list view.
- Allows for user configuration of some items.



Mentor Feedback:

- Potentially needs more “flow.”

Client Feedback:

- Likes overall style and shape use.
- Likes List View.
- Would like to see diagram view presented as one master schema, with the ability to float between items in queried views.

First Iteration Features

1. Software must connect to and get data from AWS APIs.
    a. EC2 and Load Balancer

2. Software must display information received from AWS APIs.
    a. Show connection between EC2 and Load Balancer

Client Meeting to Discuss Design Materials