

MICHIGAN STATE

UNIVERSITY

Project Plan Presentation

Semantic Search for Code and Architecture Assets

The Capstone Experience

Team Amazon

Zayd Abualfellat

Sampan Chaudhuri

Jerry Chen

Atharva Kirkole

Nicholas Li

Department of Computer Science and Engineering

Michigan State University

Spring 2025



*From Students...
...to Professionals*

Project Sponsor Overview

- A Big Five American IT company
- E-commerce platform
- Offers web services
 - Databases
 - LLMs
 - Hosting Platforms
 - Computing



Project Functional Specifications

- Problems
 - Codebases are scattered
 - Manual code searching is inconvenient
- Solution
 - Centralize the code database
 - Search using natural language query
 - Returns relevant code results

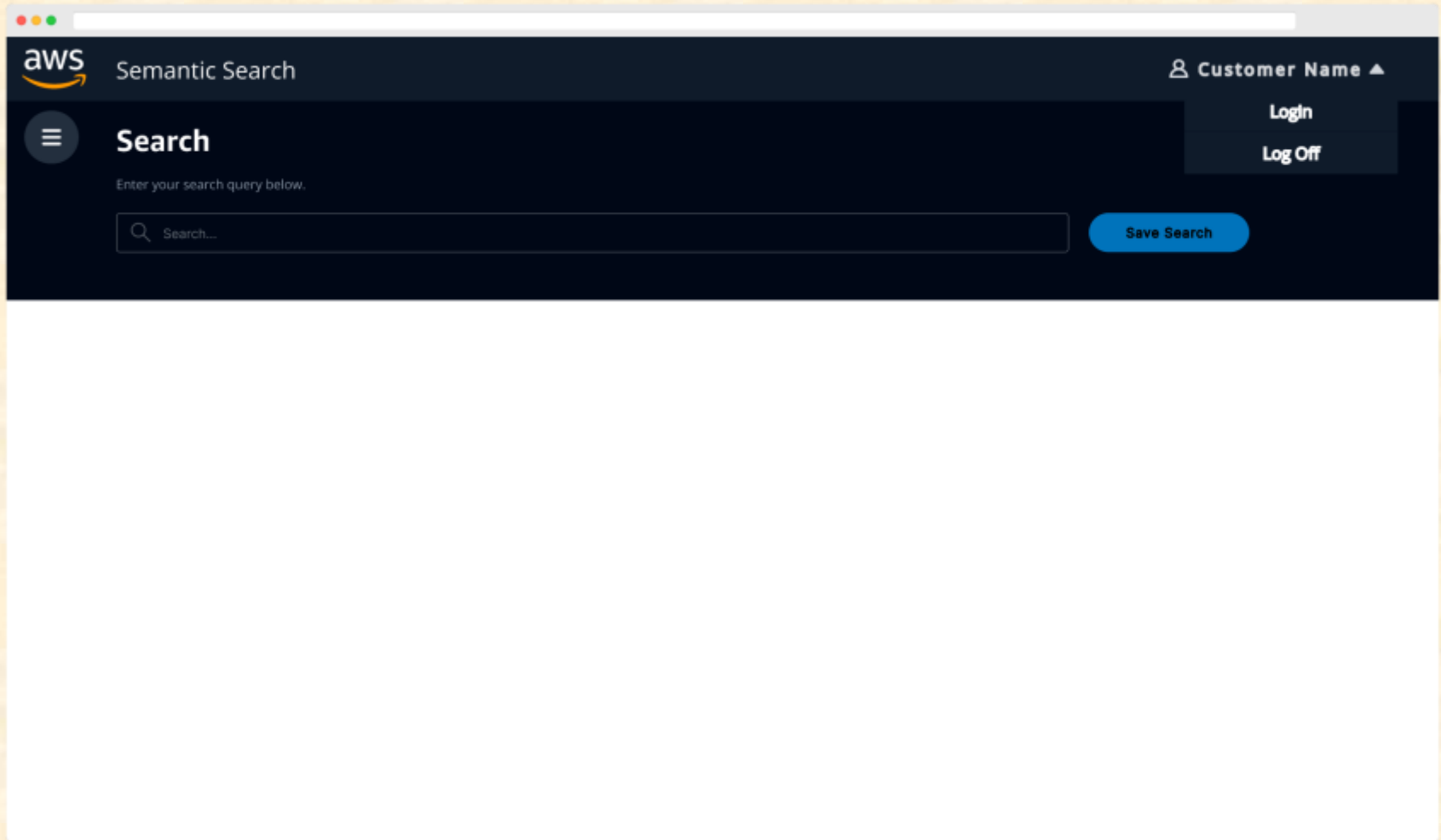


Project Design Specifications

- Intended for Amazon developers
- Enter query into search bar
- Returns top 3 relevant results
- Filter search results



Screen Mockup: Search Page



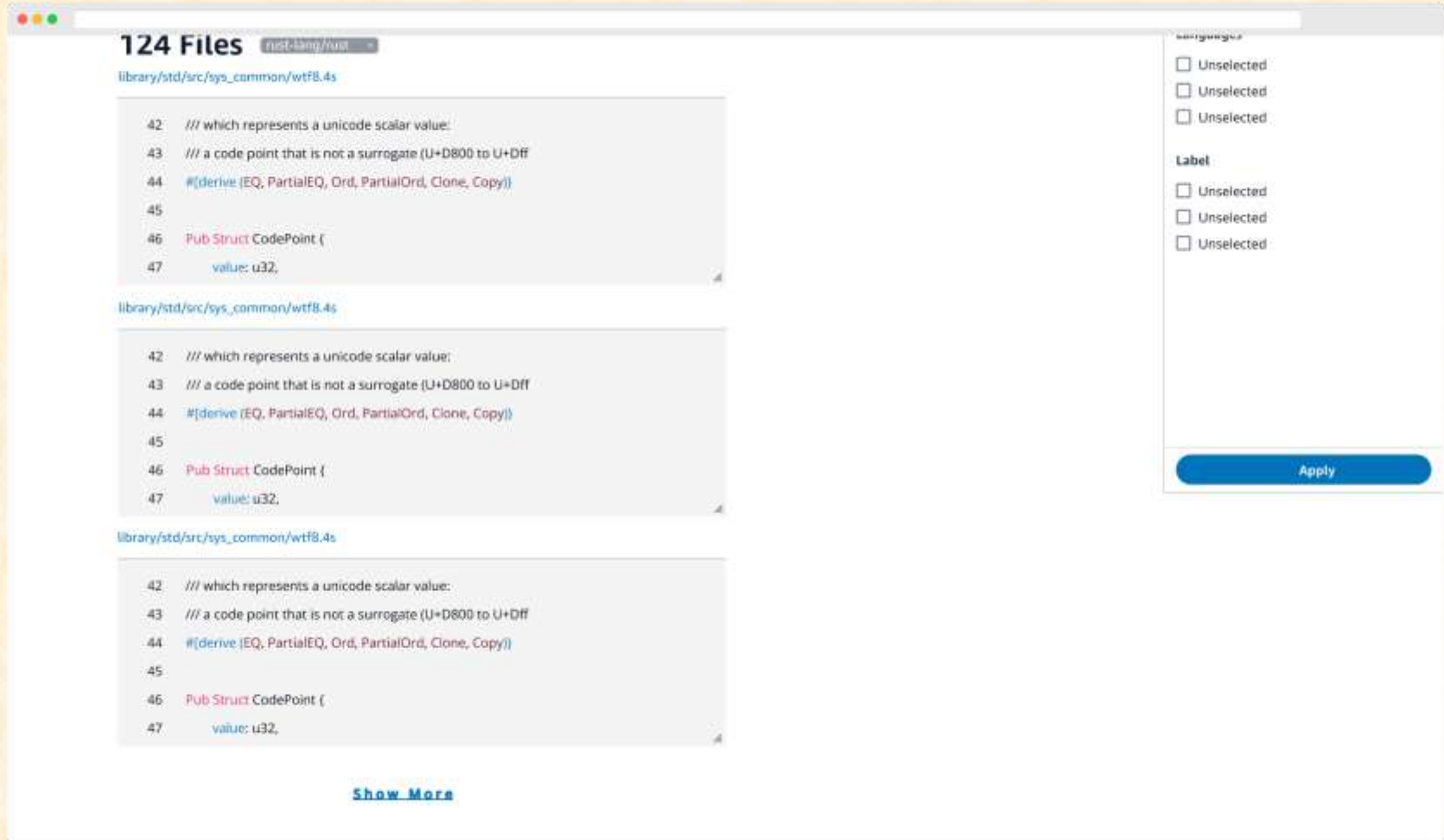
Screen Mockup: Results Displayed

The screenshot displays the AWS Semantic Search interface. At the top left is the AWS logo and the text "Semantic Search". On the top right, there is a user profile icon and the text "Customer Name" with a dropdown arrow. Below this is a "Search" section with the instruction "Enter your search query below." and a search input field containing "Repo: rust-lang/rust CodePoint". To the right of the input field is a blue "Save Search" button. Below the search bar, there are two tabs: "Code" with a count of "124" and "Repositories" with a count of "125". To the right of these tabs are two buttons: "Sort: Best Match" and "Filter". The main content area shows "124 Files" for the repository "rust-lang/rust". Below this, two code snippets are displayed, each representing a Rust struct definition for "CodePoint". The first snippet is from "library/std/src/sys_common/wtf8.rs" and the second is from "library/std/src/sys_common/wtf8.rs". Both snippets show the same code: lines 42-47, including comments and the struct definition.

```
42 /// which represents a unicode scalar value:
43 /// a code point that is not a surrogate (U+D800 to U+Dfff)
44 #[derive (EQ, PartialEQ, Ord, PartialOrd, Clone, Copy)]
45
46 Pub Struct CodePoint {
47     value: u32,
```



Screen Mockup: Show More Code



Screen Mockup: Filters Expanded

The screenshot displays a web application for searching code repositories. At the top, a search bar contains the text "Repo: rust-lang/rust CodePoint" and a "Save Search" button. Below the search bar, there are two tabs: "Code" with a count of 124 and "Repositories" with a count of 125. A "Sort: Best Match" dropdown is visible on the right. The main content area shows "124 Files" with a filter for "rust-lang/rust". Three code snippets are displayed, each with a file path "library/std/src/sys_common/wtf8.rs" and line numbers 42-47. The code snippets are identical and show comments and a struct definition for CodePoint. On the right side, a "Filter by" panel is expanded, showing "Languages" and "Label" sections, each with three "Unselected" checkboxes. An "Apply" button is located at the bottom of the filter panel.

Repo: rust-lang/rust CodePoint Save Search

Code 124 | Repositories 125 Sort: Best Match

124 Files rust-lang/rust

library/std/src/sys_common/wtf8.rs

```
42 /// which represents a unicode scalar value:
43 /// a code point that is not a surrogate (U+D800 to U+Dfff
44 #[derive (EQ, PartialEQ, Ord, PartialOrd, Clone, Copy)]
45
46 Pub Struct CodePoint {
47     value: u32,
```

library/std/src/sys_common/wtf8.rs

```
42 /// which represents a unicode scalar value:
43 /// a code point that is not a surrogate (U+D800 to U+Dfff
44 #[derive (EQ, PartialEQ, Ord, PartialOrd, Clone, Copy)]
45
46 Pub Struct CodePoint {
47     value: u32,
```

library/std/src/sys_common/wtf8.rs

```
42 /// which represents a unicode scalar value:
43 /// a code point that is not a surrogate (U+D800 to U+Dfff
44 #[derive (EQ, PartialEQ, Ord, PartialOrd, Clone, Copy)]
45
```

Filter by

Languages

- Unselected
- Unselected
- Unselected

Label

- Unselected
- Unselected
- Unselected

Apply

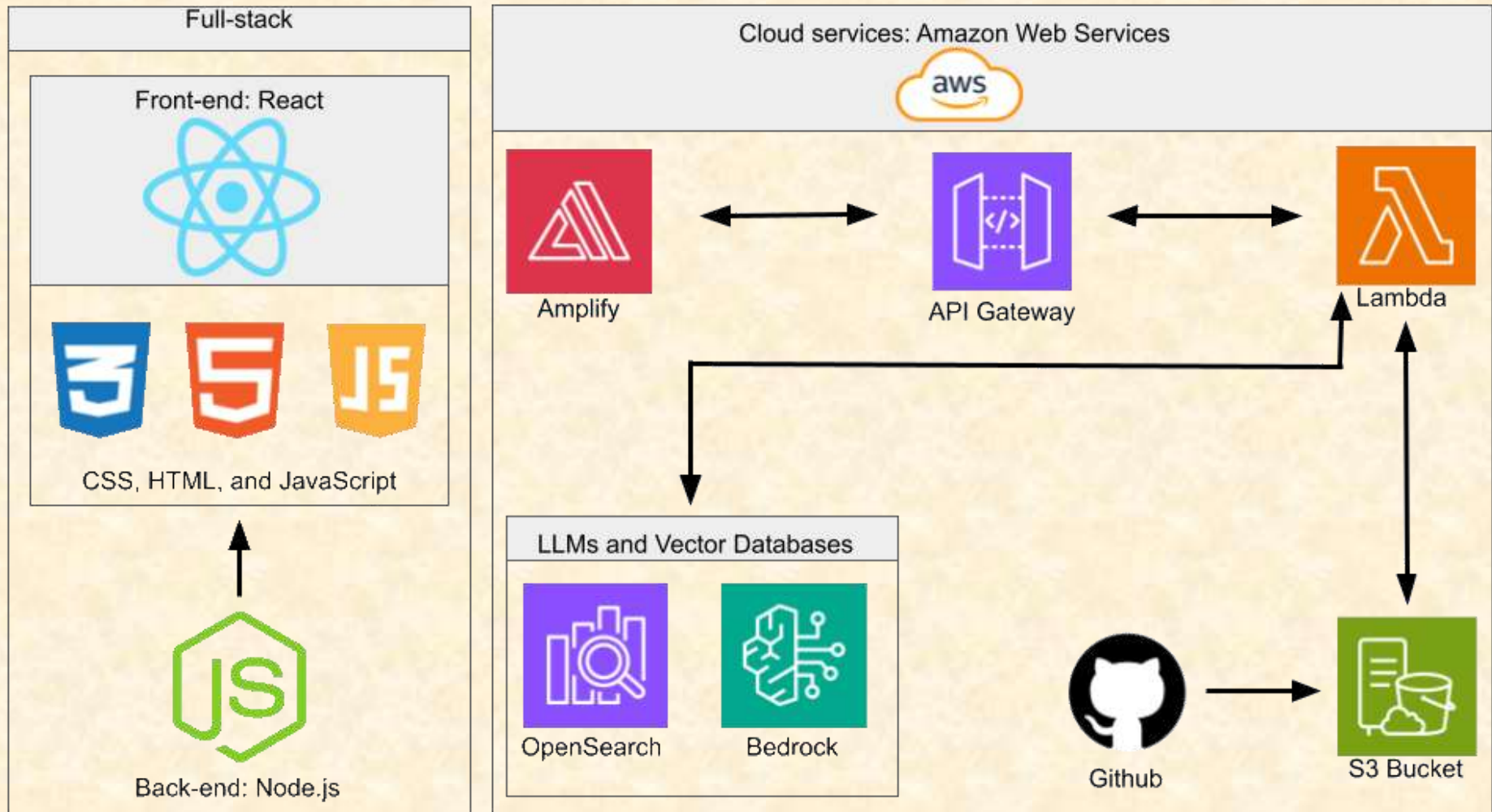


Project Technical Specifications

- Frontend: React.js
- Backend: Node.js
- Amazon Web Services
- LLMs from Amazon Bedrock
 - Titan Text Embeddings
 - Titan Text



Project System Architecture



Project System Components

- Hardware Platforms
 - None
- Software Platforms / Technologies
 - Amazon Web Services
 - AWS Lambda
 - Amazon OpenSearch
 - Amazon Bedrock
 - Amazon S3
 - Languages
 - Python
 - JavaScript



Project Risks

- Project Expenses
 - Costs for using AWS cloud resources
 - Research and provide cost estimate to client for approval
- Rate Limiting
 - Rate limits occur with quick, consecutive calls to APIs
 - Plan to segment API calls
- Frontend and Backend Connection
 - How to send LLM-generated data from back-end to front-end
 - Implement API Gateway and Amplify solutions



Questions?

?

?

?

?

?

?

?

?

?

