AHA Ideas Team - Final Project Report

Devansh Modi[#1], Pratik Joseph Dabre[*2], Kathy Nguyen[#3]

[#] *Department of Software Engineering*
San Jose State University
1 Washington Sq, San Jose, CA 95129

[1]devansh.modi@sjsu.edu
[2]pratikjoseph.dabre@sjsu.edu
[3]kathy.nguyen09@sjsu.edu

***Abstract*— This report gives details of AHA Ideas Team final project design, architecture, and implementation details. Furthermore, the goal of this report is to give a full description of our Database Model, Cloud Computing infrastructure, SaaS multi-tenancy support, use cases and wireframes, and a reflection on our project's personas and impact.**

***Keywords*—– Ideas Management System, REST APIs, SaaS Platform, Vote and Build Ideas, Cloud Computing Infrastructure, Microservices, MySQL, Distributed Systems.**

Github Repository–
https://github.com/sjsucmpe272SP22/AHA-Ideas-Team

I. Live hosted application

Our application can be accessed from the following link:

https://main.dqzv8z394jyvm.amplifyapp.com/

Administrator Login Credentials:

Username: admin
Password: admin

II. Introduction

This is a final report presented to Professor Rakesh Ranjan for CMPE 272 by the AHA Ideas Team. Our team project focuses on building an idea sharing portal and to build a hub for growth and innovation for today's modern enterprises. Many large companies are tasked with an ever-changing consumer landscape, where new ideas are best sourced from customers, partners, and employees. We strived to build a reliable Software-as-a-Service or SaaS platform that can be easily scaled for a multi-tenant architecture in a cloud computing environment along with some of modern technology practices.

Our project timeline started in February 2022, where we spent a large amount of time on user research, interview, feature scope, and targeted personas for this project.

In today's world, companies have a lot of diversity. The greater the diversity is, the more the different point of views, different types of ideas. Our project/idea management tool aims to share ideas between the companies and to engage the community within the company.

With the help of a modern UI and an easily usable platform, users can easily access bright ideas from employees and customers. To create a scalable and a highly available platform, microservices were used. The services were kept separate from each other and each service had its own database and was deployed in an AWS instance. The endpoints of the authentication service and the ideas service were then exposed and were accessible to the react application. Users were able to successfully share their ideas, upvote their favorite ideas and give valuable comments.

III. Problem statement

Our project focused on 4 key problems faced by today's companies, from startups to Fortune 500 companies. These problems are:

1) Help businesses and companies access bright ideas from employees and customers; and, help increase customer satisfaction.
2) Allow companies and people to crowdsource feedback, engage the community, and analyze trends.
3) Make it easy for any company to host a modern UI and platform where their users can painlessly share ideas with one another.
4) Build a robust and scalable load balancing architecture by using modern cloud computing practices to build a scalable and highly available platform.

IV. Personas

Personas are vital to the success of any product because they drive the design decisions by looking at the common user requirements and bringing them to the initial view in planning before the design process starts. Personas provided our team with a shared understanding of the users in terms of goals and capabilities. As a result, this project is aimed for supporting:

1) *Product managers* - These users are the primary target users of our platform. These users want a summarized dashboard and visualization of ideas on a platform. We have built an admin panel with powerful visualization for these users.
2) *Engineers* - these users look for new ideas to improve services or products. They look for an easy way to share ideas and start developing on them. Our platform allows users to post ideas easily and integrate with Github.
3) *Product Development Teams* - These teams often scout for ideas using surveys, but by using our platform they will have access to private, exclusive, and a broad spectrum of ideas. This will help companies innovate, manage portfolios of ideas, and help marketing initiatives.
4) *Corporate Leadership* - leaders often have many ideas, but they are risk averse. By using the voting feature of our platform, these leaders will be able to tell which idea is powerful.
5) *Customers* - many customers have complaints, but many also have good ideas for a company they support. This user is interested in an easy way to post a new idea for a company. Our platform will allow that.
6) *Other Employees* - all employees of a company can have meaningful ideas to share. It's often the case that many good ideas go unheard because one's role does not have the opportunity to table an idea. By using our platform, this user will meet his/her requirement of commenting on an idea and voting on an idea to show support.

V. Project impact

Any product out there should have a powerful mission statement and impact. Our mission is customer first. Our impact is directly on a company's ability to render powerful customer service. By using our platform, a company will be able to stay ahead of the competition, innovate continuously, and offer quality products and services.

VI. Feature scope

We maintained our prime focus on allowing users to easily create new ideas, vote, and engage via commenting.

A. *Create New Ideas*

Users can create a new idea with a unique title, description, category, and a repository link. This feature is available for logged in users only.

B. *Vote an Idea*

Viewers of an idea such as customers or company employees can vote or unvote an idea to show their support for an idea.

C. *Comment on an Idea*

Any new idea needs a supportive community and a forum for engagement. Our platform allows a user to post comments and share their thoughts.

D. *Github Integration*

Our platform also differentiates from other competitors by allowing new ideas to be seamlessly integrated and developed by automatically creating a new Github repository in the company's Github organization. This Github repository is created when a new idea is created by a user.

Example Repository:
`https://github.com/sjsudev/idea-repo-1skq`

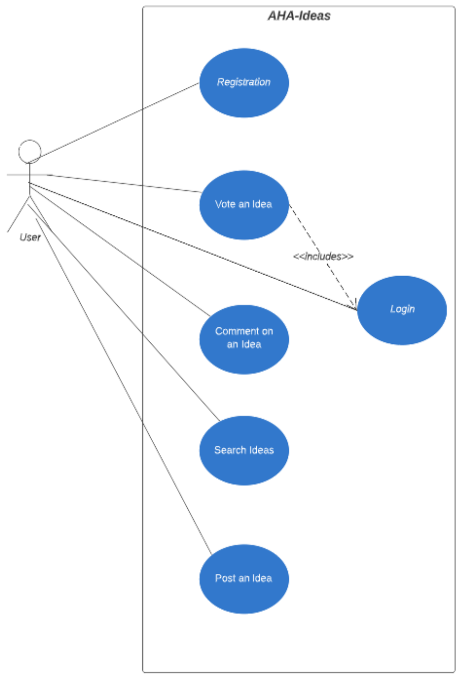
E. *Search Ideas*

With a large database of ideas, users should be able to search and find their favorite ideas effortlessly.

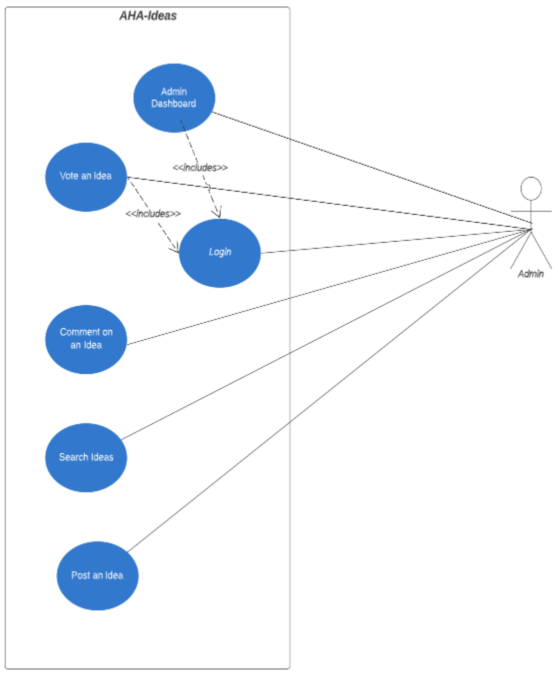
F. *Account Management*

We offer user profile management such as Account Information screen, secure login and registration screens. We also allow the user to select a username in the registration process.

VII. Use cases



Use Case Diagram - User



Use Case Diagram - Admin

Work-flow of the use cases:

1. Post an Idea

Use Case	Post an Idea
Actor	User
Basic Flow	The user navigates to the landing page. The user clicks on the Add a new Idea button. The user enters the information about the idea that includes the one-line summary of the idea, description of the idea, category that the idea falls under and when this idea is needed by. After entering the details, the user clicks on the submit button and the idea gets posted.

2. Vote an Idea

Use Case	Vote an Idea
Actor	User
Basic Flow	The user navigates to the landing page. The user is able to see all the ideas. The user clicks on the vote button next to the idea to vote for the idea. The user is only able to vote once for each idea.The vote count is updated on the idea.
Precondition	The user needs to be logged in.

3. Comment on an Idea

Use Case	Comment on an Idea
Actor	User
Basic Flow	The user navigates to the ideas dashboard. The user selects the idea that he wants to comment on. The user is then taken to that idea page and he can enter the comments in the text box given. On submitting, the user should successfully be able to see his comments.

4. Search Ideas

Use Case	Search Ideas
Actor	User
Basic Flow	The user navigates to the ideas dashboard. The user can then search any ideas that he wants through the search icon present in the navbar.

VIII. Approach

The approach was to not have a single monolithic system but to divide and deploy the application using microservices architecture. Within a microservices architecture, every different task/service is being handled by different microservice components which makes it easier to scale and deploy the application.

We used agile methodology to iterate our development week over week until our product was ready. We also demonstrated the project to friends and family who fit the project's personas.

Our approach was continuous development and integration using Docker, Github, and AWS.

The microservices architecture consisted of three parts:

1) React application for frontend development
2) User authentication service (Developed using Flask)
3) Ideas Service API (Developed using JAVA Spring Boot)

IX. UI design wireframes

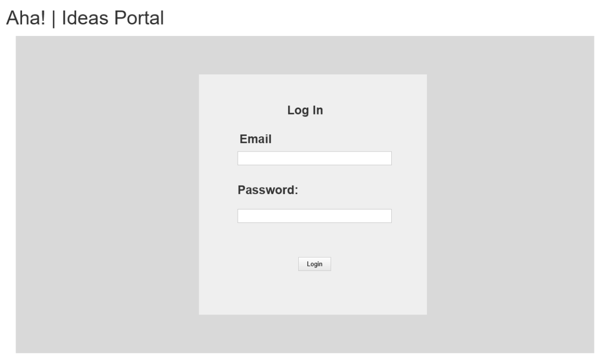


Fig. 1 Login screen wireframe

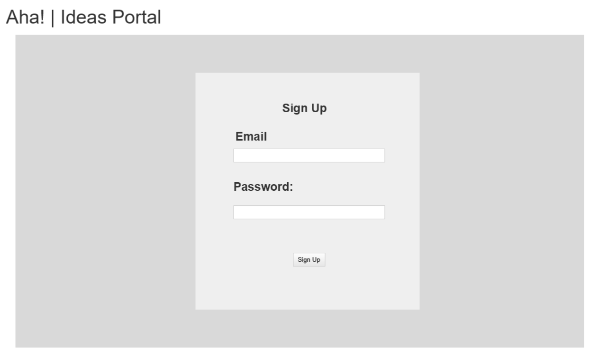


Fig. 2 Register screen wireframe

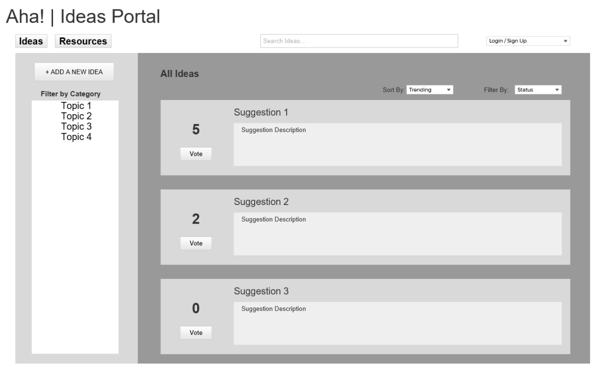


Fig. 3 Portal screen wireframe

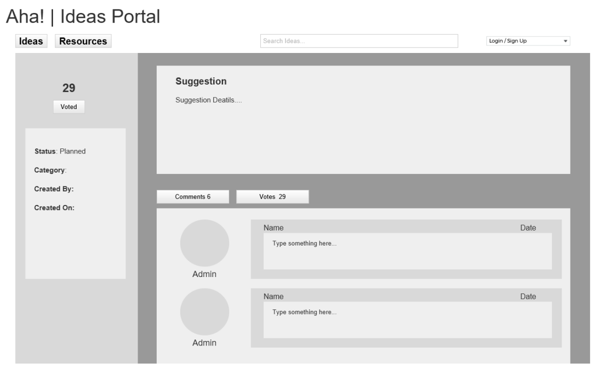


Fig. 4 Idea screen wireframe

X. System GUI Implementation

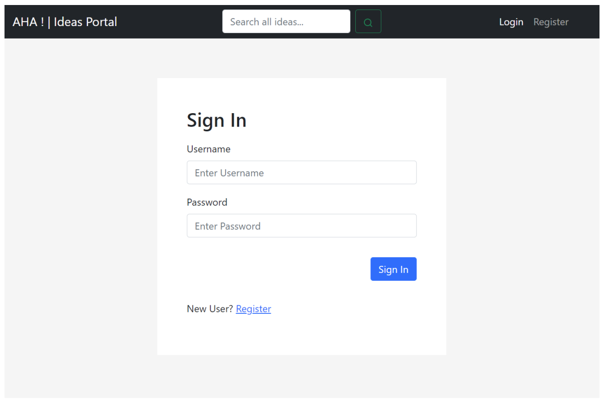


Fig. 5 Login screen final prototype

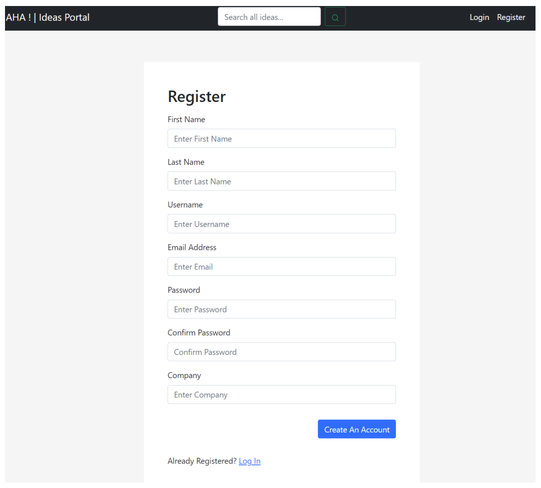
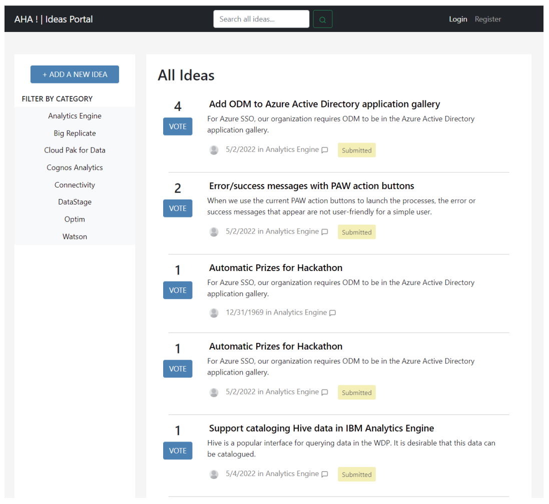


Fig. 6 Register screen final prototype



All Ideas

4 VOTE
Add ODM to Azure Active Directory application gallery
For Azure SSO, our organization requires ODM to be in the Azure Active Directory application gallery.
5/2/2022 in Analytics Engine Submitted

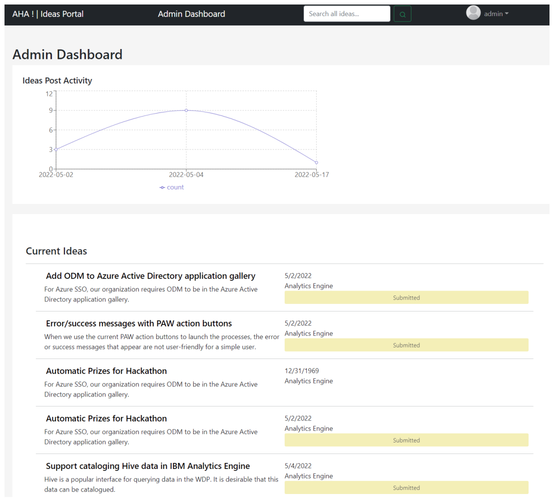
2 VOTE
Error/success messages with PAW action buttons
When we use the current PAW action buttons to launch the processes, the error or success messages that appear are not user-friendly for a simple user.
5/2/2022 in Analytics Engine Submitted

1 VOTE
Automatic Prizes for Hackathon
For Azure SSO, our organization requires ODM to be in the Azure Active Directory application gallery.
12/31/1969 in Analytics Engine

1 VOTE
Automatic Prizes for Hackathon
For Azure SSO, our organization requires ODM to be in the Azure Active Directory application gallery.
5/2/2022 in Analytics Engine Submitted

1 VOTE
Support cataloging Hive data in IBM Analytics Engine
Hive is a popular interface for querying data in the WDP. It is desirable that this data can be catalogued.
5/4/2022 in Analytics Engine Submitted

Fig. 7 Portal screen final prototype

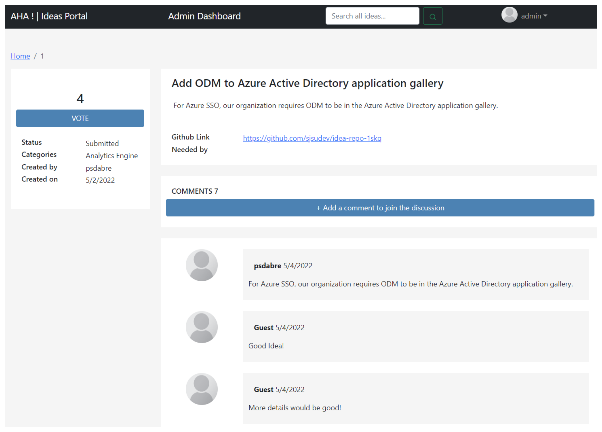


Admin Dashboard

Current Ideas

Add ODM to Azure Active Directory application gallery For Azure SSO, our organization requires ODM to be in the Azure Active Directory application gallery.	5/2/2022 Analytics Engine	Submitted
Error/success messages with PAW action buttons When we use the current PAW action buttons to launch the processes, the error or success messages that appear are not user-friendly for a simple user.	5/2/2022 Analytics Engine	Submitted
Automatic Prizes for Hackathon For Azure SSO, our organization requires ODM to be in the Azure Active Directory application gallery.	12/31/1969 Analytics Engine	
Automatic Prizes for Hackathon For Azure SSO, our organization requires ODM to be in the Azure Active Directory application gallery.	5/2/2022 Analytics Engine	Submitted
Support cataloging Hive data in IBM Analytics Engine Hive is a popular interface for querying data in the WDP. It is desirable that this data can be catalogued.	5/4/2022 Analytics Engine	Submitted

Fig. 8 Admin dashboard final prototype



4

Status	Submitted
Categories	Analytics Engine
Created by	psdabre
Created on	5/2/2022

Add ODM to Azure Active Directory application gallery

For Azure SSO, our organization requires ODM to be in the Azure Active Directory application gallery.

Github Link https://github.com/sjsudev/idea-repo-1skq
Needed by

COMMENTS 7

psdabre 5/4/2022
For Azure SSO, our organization requires ODM to be in the Azure Active Directory application gallery.

Guest 5/4/2022
Good Idea!

Guest 5/4/2022
More details would be good!

Fig. 9 Idea page

XI. Backend technologies

1) **User Authentication service:**

This service was created using flask , a micro web framework to create restful apis in python. The flask application was then containerized using docker and deployed on an amazon ec2 instance.

- *Flask:*

[1] " Flask is a framework of python and it does not require any libraries, hence it is a microframework. Flask does not have any database abstraction layer , form validation or any feature that third- party libraries provide but Flask definitely has extensions which can add these features extremely easily as if they were implemented using the framework itself".

Sample Request

```
`curl --location --request GET
'http://52.207.242.60:5000/all-users'`
```

- *API Endpoints*

1. Get users API: This endpoint is a get request and returns all the users which are using the system.

2. Login API: This endpoint is a post request and takes in the username and the password as body and returns a response. If the user credentials are correct , a session ID will be returned back as a response, if not an error message will be returned.

3. Register API: This endpoint is a post request and takes in the user credentials as body and returns a response. If the user credentials are correct , a session ID will be returned back as a response, if not an error message will be returned.

4. Get Active Sessions: This endpoint is a get request and will give all the current active sessions.

2) **Ideas API Service:**

This service uses Spring Boot Java, a microservice framework to expose a REST API and is hosted on AWS. This service can support multi-tenancy using multiple VMs and per tenant tables.

Documentation

https://documenter.getpostman.com/view/877303/UyxbppYy

Sample Request

```
`curl --location --request GET
'http://ideasapi-env.eba-mcygtrsp.us-west
-2.elasticbeanstalk.com/ideas'`
```

- API Endpoints:

1. *Get Ideas*: This endpoint is a get request and returns all the ideas.
2. *Create Idea*: This endpoint is a post request and takes the category of the idea, the username of the user creating this idea and the title and description of the idea as a body and returns back a response.
3. *Delete Idea*: This endpoint is a DELETE request for an idea.
4. *Update Idea:* This endpoint is a PUT request to update an idea.
5. *Create Comment*: This endpoint is a post request and takes the ID of the idea that the user wants to

comment on, the username of the user creating this idea and the comment as a body and returns back a response.

6. *Vote idea*: This endpoint is a post request and takes the category of the idea, the username of the user voting for this idea and the title and description of the idea as a body and returns back a response.

XII. Agile Methodology

The four core values from the Agile Manifesto [2] are:

1. **Individuals and interactions** over processes and tools
2. **Working software** over comprehensive documentation
3. **Customer collaboration** over contract negotiations
4. **Responding to change over** following a plan

We incorporated these values into our project by meeting regularly for our sprints to discuss what we were working on, what we planned to have completed next, and any issues we were facing. During sprint planning meetings we would establish features we would work on for that sprint.

We built our application in iterative steps to create small components of working software. This allowed us to focus on short term deliverables instead of getting bogged down by the scope of the project. While working on features for our application we kept in mind our customers and users as it is crucial that our application fulfills their needs. The agile principle of simplicity was especially emphasized in order to reduce wasted time and resources. From this we gained speed and reduced complexity in regards to our code, architecture, and testing. We focused on functionality over form as working software is the primary measure of progress, however our product still is cohesive and attractive. After we did our demo, we took feedback from the professor and added an admin dashboard panel to round out our application.

XIII. Database design

Users Authentication service:

The user authentication service uses mysql rds as a database and it has two tables (Users and sessions).

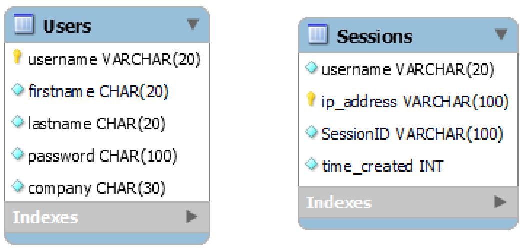
The users table has five fields namely the username, firstname,lastname,password and the company,wherein the username is a primary key.

The sessions table has four fields namely the *username*, *ip_address*, *sessionId* and the *time_created* at.

The users table is used to maintain data information about all the users registered to the system.

The sessions table is used for session management to check whether the user has logged in or not.

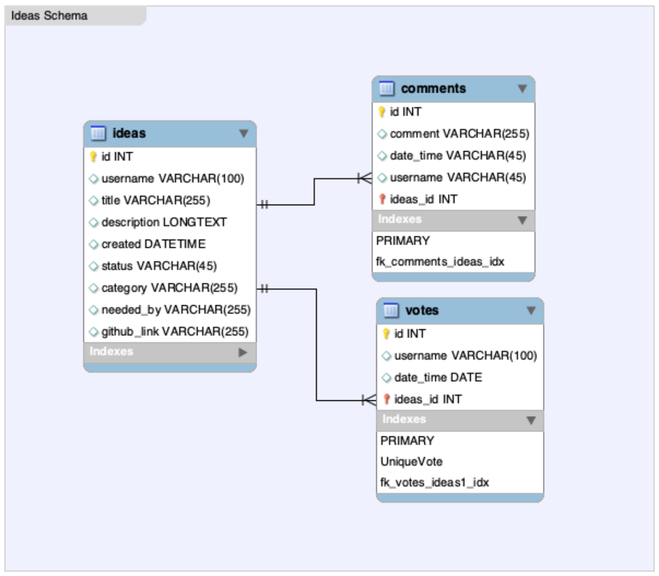
The database schema of user authentication service is given below:



Ideas API Service:

The Ideas API service uses MySQL RDS database as well, mainly due to the need of ACID properties and write consistency. The voting and commenting features both require a transactional approach, which is not easy to implement in a NoSQL data store. Ideas API database has 3 primary tables that are - ideas, comments, and votes.

The votes and comments tables have a foreign key to ideas primary key. The votes table also has a constraint to ensure unique votes by each user.

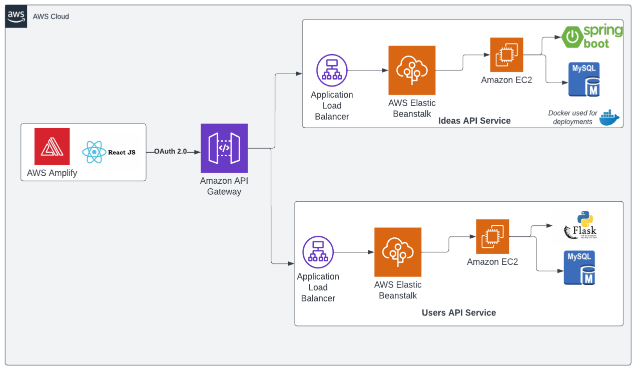


XIV. Cloud computing

A) *System Architecture and Communication:*

1) *Overview:*

Our Aha! Ideas application is hosted as a white-labeled SaaS application and hence it is completely cloud based. All the components in our application , Ideas API service, Users API service and the React (frontend) application are hosted on the cloud using AWS EC2 instances, relational databases like MySQL RDS, and AWS Amplify for orchestrating ReactJS deployments.



2) *Connectivity Design:*

We have a lightweight Python Flask server which is used for user authentication service. This service is using MySQL RDS for storing and retrieving the data. This service is dockerized and deployed on an EC2 instance using AWS ElasticBeanstalk.

We also have a Spring Boot Java service which handles the Ideas API Service. MySQL RDS, an AWS service, is used as a database in this service. This service is dockerized and deployed on an EC2 instance.

The React JS application (UI) is the only way of entering our application. the user is able to login/register to the system using the users authentication service . the user is able to post ideas, vote ideas and comment on ideas of their choice using ideas service.

As this is a microservices architecture, the two API Services also known as Ideas API and User Authentication API do not need to talk to each other and can be completely isolated from each other.

All the backend services i.e the user authentication service and the ideas service are deployed on EC2 instances and then these services are monitored using AWS Cloudwatch For log analysis, we were able to see the logs of the AWS services in case they had any issues in operation. By monitoring the instance's metrics. We were able to see the number of invocations, error and success rates, etc. By using CloudWatch we were able to effectively monitor our backend to confirm everything was working correctly

B) *Components:*

1) AWS EC2: "AWS EC2 or elastic cloud computing service is a cloud service offered by amazon web services. AWS EC2 eliminates the need to configure the use of hardware Upfront and the user can just focus on developing and deploying applications easily. EC2 also allows you to scale applications up or down depending on the traffic coming to the server." [3]

2) AWS RDS: AWS RDS or AWS Relational Database Service is a cloud computing service offered by aws which allows users to have scalable, managed databases without worrying about managing the databases themselves. It benefits users because you do not have to worry about the operation of the database and also allows pay-per-use policy i.e you only pay for the resources that you have used. [4]

3) AWS CloudWatch:

   CloudWatch is a repository which holds metrics. An AWS instance can put metrics into this repository or you can input your own metrics. You can then retrieve statistics from the metric repository. Now that you have the data(metrics and the statistics) , you can set trigger/alarm actions to specify an action when certain conditions are met. For example, you can set a trigger if which triggered will stop/start/terminate an AWS instance based on certain parameters. Users can access the CloudWatch features using the AWS Command Line Console or the AWS Management Console.

4) Amazon API Gateway:

   The Amazon API Gateway service is a fully managed service that allows developers to easily construct, publish, maintain, monitor, and secure APIs at any scale. RESTful APIs and WebSocket APIs can be created with API Gateway in order to enable real-time two-way communication between applications.API gateway handles tasks like traffic management, authorization and access control, monitoring, and API version management while processing multiple concurrent API calls. Amazon API Gateway acts like an entry point for applications to access data, business logic, or functionality from the backend service.

5) AWS Beanstalk:

AWS Beanstalk is used to quickly deploy and manage applications in aws cloud without understanding of infrastructure that runs these applications. It can manage capacity provisioning, load balancing, scaling, and application health monitoring among others for our applications. IT supports Go, Java,.NET, Node.js, PHP, Python, and Ruby applications. Elastic Beanstalk creates the selected supported platform version and allocates one or more AWS resources, such as Amazon EC2 instances, to run the application when it is deployed.

XV. Conclusions

We were able to successfully deploy a working prototype of an ideas portal that businesses could use to crowdsource ideas from people interested in their products. We found it important to clearly define personas at the very start of this process because knowing who our target customers were helped give our project direction and purpose. Our project consisted of several trending technologies right now like cloud computing and microservices which we implemented mainly through Amazon AWS.

While the concept of an ideas portal is simple, we believe that it can be worthwhile for companies to invest in popular ideas their customers want. Customer data is valuable because it is a goldmine of insights for companies so why not provide a public, direct, and centralized platform to communicate with users. This can also be a way to encourage customer engagement and improve customer satisfaction.

References

[1] "Flask (web framework)." https://en.wikipedia.org/wiki/Flask_(web_framework) (Accessed May 20, 2022).

[2] M. Beedle, A. van Bennekum, A. Cockburn, "Manifesto for Agile Software Development", 2001. [Online]. Available: https://agilemanifesto.org (Accessed May 20, 2022)

[3] *Amazon Elastic Compute Cloud: User Guide for Linux Instances*. Amazon. 2022, pp. 1-5.(Accessed May 20, 2022.) [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

[4] *Amazon Relational Database Service: User Guide.* Amazon. 2022, pp. 1-5. (Accessed May 20, 2022.) [Online]. Available: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/rds-ug.pdf