Code Karin

Client: Dr. Krishna Kadiyala

Developers: Jacob Hollis, Megan Tran, Kate Brayshaw, Nithesh Bonugu, Dylan Wulfson

Overview

- Our Client & Team
- Vision
- Design Overview
- Architecture Solution / Diagrams
- Demonstration
- Experiences and Takeaways
- Questions

Our Team

Jacob Hollis

Team Leader /
Backend Developer

Ngan Hanh Tran

Frontend Developer

Kate Brayshaw

Lead Documentor /
Website Manager

Nithesh Bonugu:

Poster Design /
Documentation

Dylan Wulfson

API Gateway

Our Client

- Dr. Kadiyala is an Assistant Professor of Computer Science at TCU since 2020.
- Her expertise is in Computer Networking.
- She has been teaching the Introductory Programming class at TCU since Fall 2020.
- This project is motivated by her experiences teaching this class.

Vision

- Reason for Development:
  - Extend in-class demonstrations
  - Allow anonymity in practice and in-class contest leaderboards
  - Teacher-Driven Programming Platform
  - Motivate students to engage with programming challenges in and out of the class environment
- Comparisons:
  - Hackerrank
  - GitHub Classroom

Comparisons

Hackerrank

- Professor is unable to see in real-time on how the students are performing
- Instructor does not have individual control over student participation (the platform is managed by an external entity)
- It is a paid service

GitHub Classroom

- Professor is unable to see in real-time on how the students are performing
- There is no student anonymity.
- It is also a paid service.

Design Overview

- Prototyping/planning
- Designed and built AWS infrastructure

- Migration to AWS CDK
- Created and implemented a deployment scheme for infrastructure

- Implement login, coding contest, and practices
- Feature expansion

- Cleanup
- Bug fixes
- Testing

User Interactions

Professor

See students who attend which practice / contest	Create Practice / Contest	Delete Practice / Contest
Assign Practice / Contest	Update Practice / Contest	Create Classroom
Delete Classroom	Update Classroom	

Student

- View Practice / Contest
- Do Practice / Contest
- See Solutions
- See Ranks

Amazon Web Services

- Architecture
  - AWS ECS (Elastic Container Service)
  - AWS Lambda
  - AWS API Gateway
  - AWS CodeBuild
  - AWS Route53
- Deployment
  - AWS CDK (Cloud Development Kit)
  - AWS CloudFormation

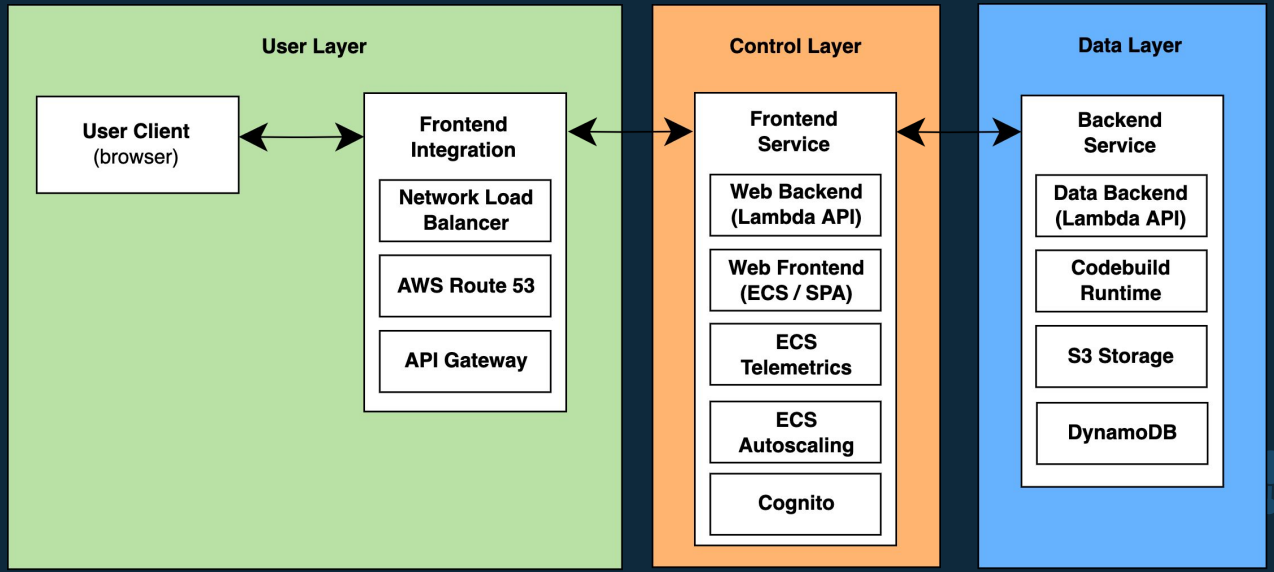
Amazon Web Services

- Availability
  - Auto-Scaling
  - Load-Balancing
  - Serverless Backend Architecture
- Security
  - AWS Cognito
  - AWS Certificate Manager
  - AWS IAM (Identity and Access Management)

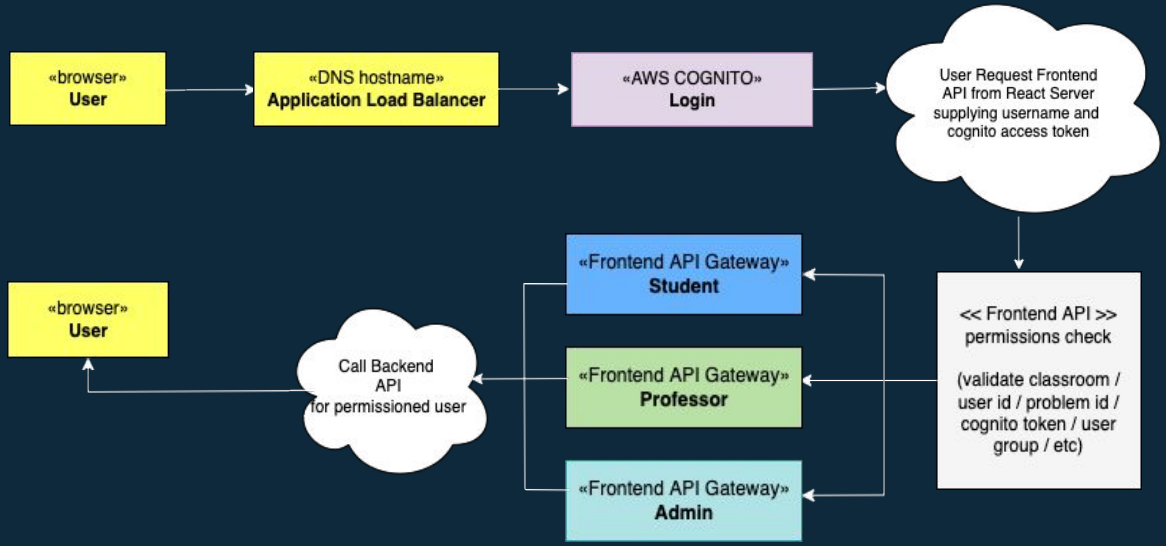
Overview

- Our Team & Client
- Vision
- Design Overview
- Architecture Solution / Diagrams
- Demonstration
- Experiences and Takeaways
- Questions

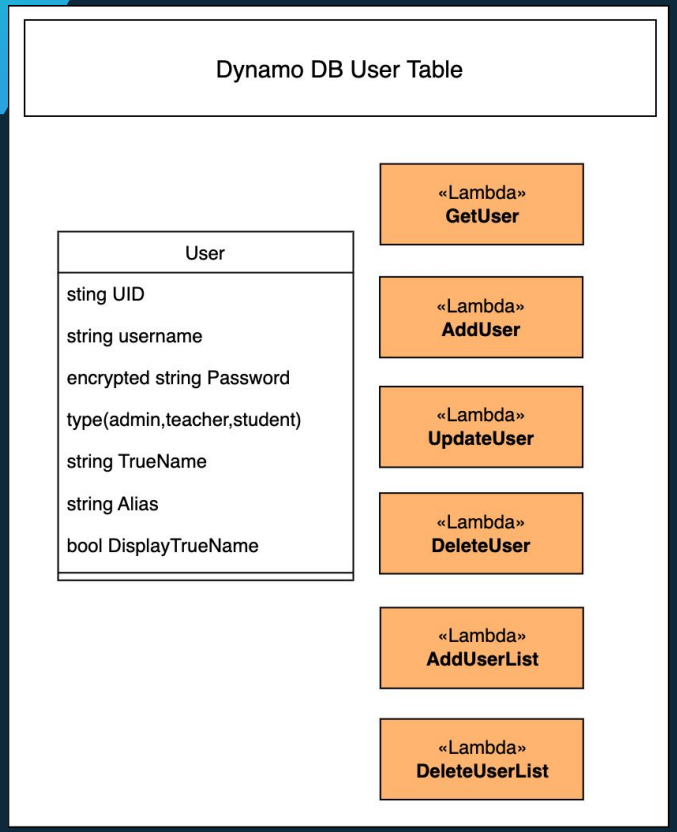
Three-Layered Architecture

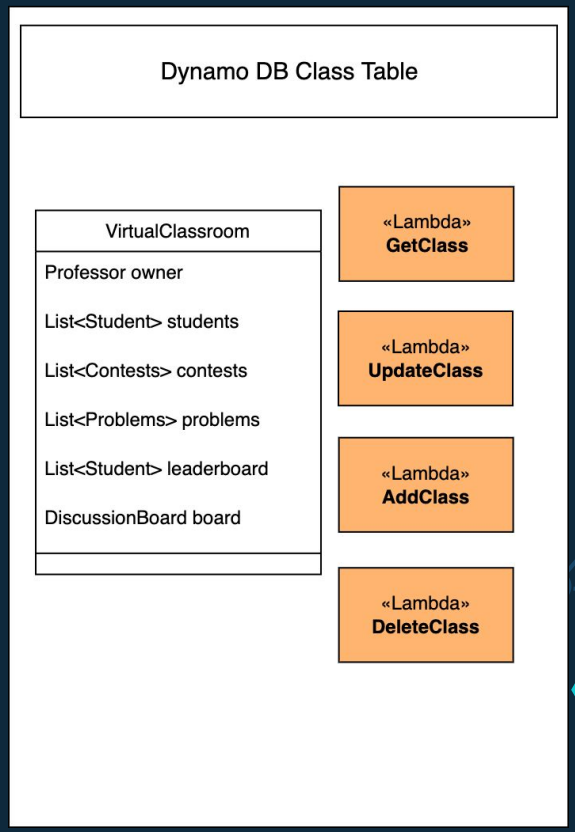


FrontEnd


«browser»
User
«DNS hostname»
Application Load Balancer
«AWS COGNITO»
Login
User Request Frontend API from React Server supplying username and cognito access token
<< Frontend API >>
permissions check
(validate classroom / user id / problem id / cognito token / user group / etc)
«Frontend API Gateway»
Student
«Frontend API Gateway»
Professor
«Frontend API Gateway»
Admin
Call Backend API for permissioned user
«browser»
User

BackEnd


Dynamo DB User Table
User
sting UID
string username
encrypted string Password
type(admin,teacher,student)
string TrueName
string Alias
bool DisplayTrueName
«Lambda» GetUser
«Lambda» AddUser
«Lambda» UpdateUser
«Lambda» DeleteUser
«Lambda» AddUserList
«Lambda» DeleteUserList


Dynamo DB Class Table
VirtualClassroom
Professor owner
List<Student> students
List<Contests> contests
List<Problems> problems
List<Student> leaderboard
DiscussionBoard board
«Lambda» GetClass
«Lambda» UpdateClass
«Lambda» AddClass
«Lambda» DeleteClass

BackEnd

Dynamo DB Problem Table

Problem (dynamo)
DateTime end_time
Solution s3_path
Problem_name string
Problem_set string
Problem_id problem-set/problem-name

Problem (s3)
FILE description.json
FILE boilerplate.java
FILE Code.java (solution)
FILE Tester.java

«Lambda» **GetProblemSet**

«Lambda» **AddProblemSet**

«Lambda» **DeleteProblemSet**

«Lambda» **GetProblemById**

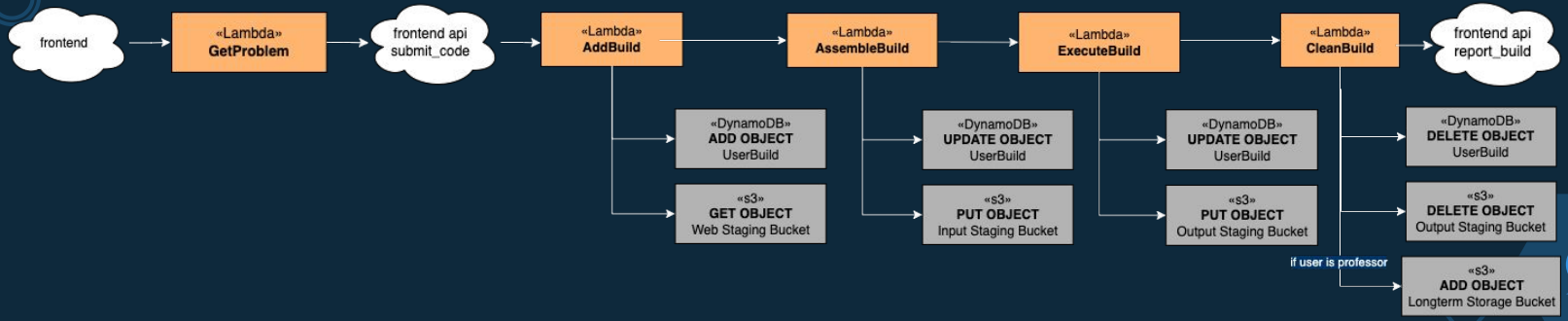
Dynamo DB Grades Table

Grade
Problem_id pid
Class_id cid
User_id uid
Tests_Passed int
Tests_Total int

«Lambda» **AddGrade**

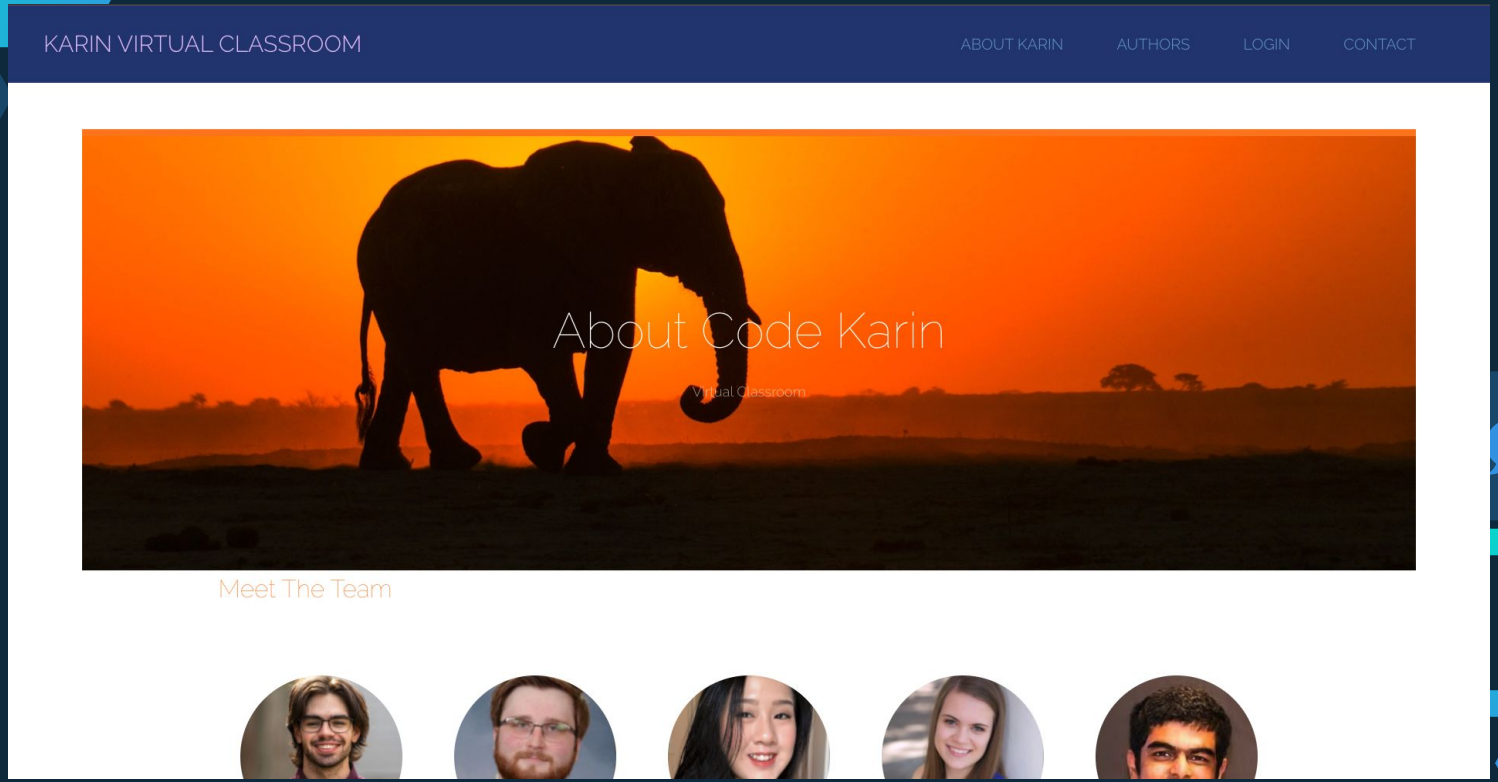
«Lambda» **DeleteGrade**

BackEnd


frontend
«Lambda»
GetProblem
frontend api
submit_code
«Lambda»
AddBuild
«Lambda»
AssembleBuild
«Lambda»
ExecuteBuild
«Lambda»
CleanBuild
frontend api
report_build
«DynamoDB»
ADD OBJECT
UserBuild
«s3»
GET OBJECT
Web Staging Bucket
«DynamoDB»
UPDATE OBJECT
UserBuild
«s3»
PUT OBJECT
Input Staging Bucket
«DynamoDB»
UPDATE OBJECT
UserBuild
«s3»
PUT OBJECT
Output Staging Bucket
«DynamoDB»
DELETE OBJECT
UserBuild
«s3»
DELETE OBJECT
Output Staging Bucket
if user is professor
«s3»
ADD OBJECT
Longterm Storage Bucket

Demonstration

https://codekarin.com https://youtu.be/syfUauYR58E



Experiences & Takeaways

- Learning new Technologies
- Conflict Management
- Setbacks and Risk Management
- Client Requirement Change
- Time Management
- Communication

Questions