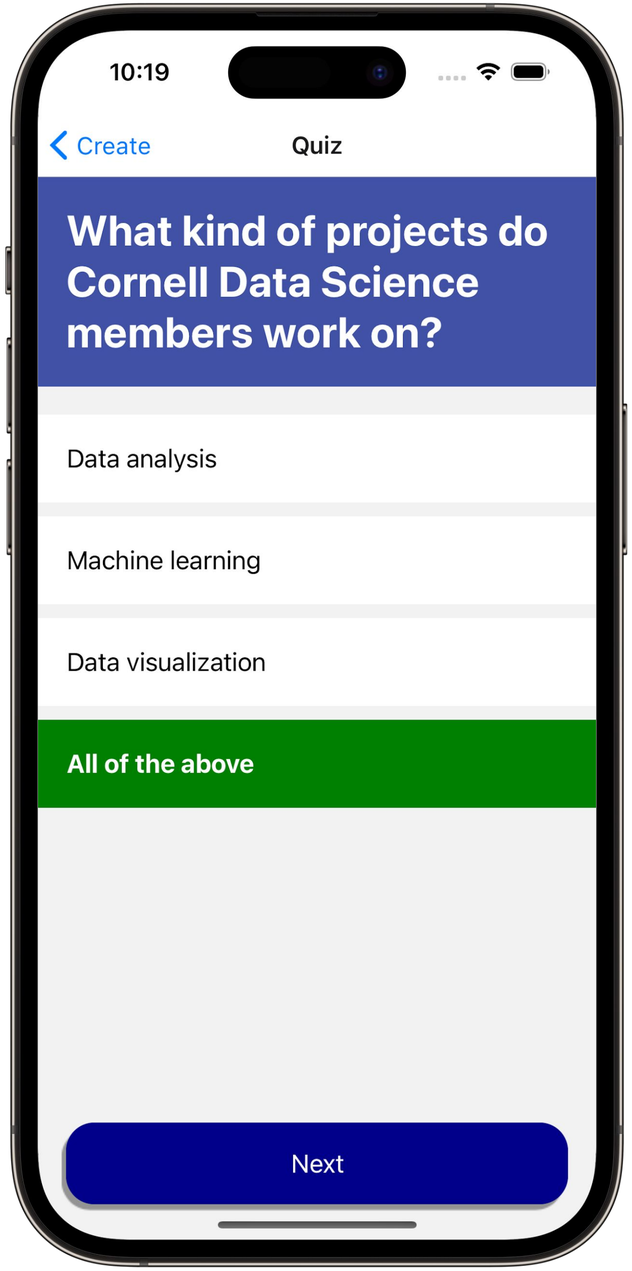

TrivAI

TrivAI

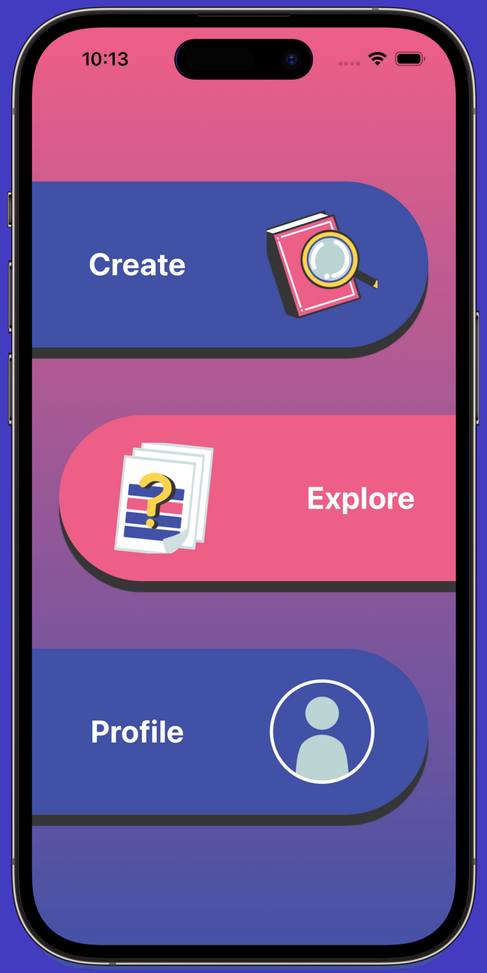
An iOS application that **generates quizzes** for users based on any topic

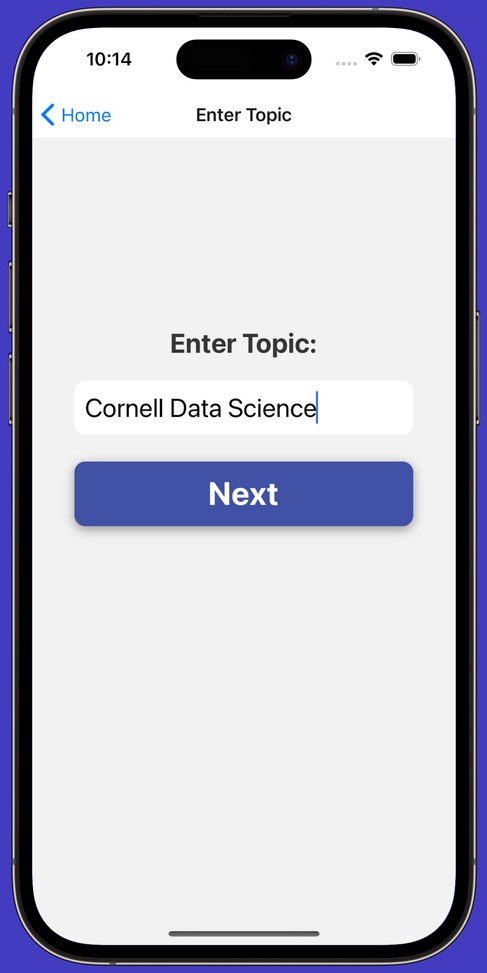
How does TrivAI generate questions?

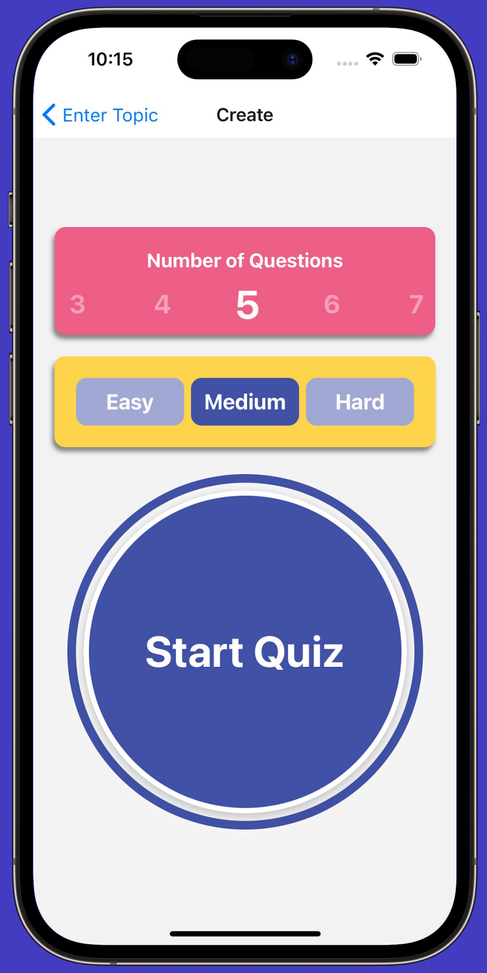
Leverages the **OpenAI GPT3.5-Turbo API** to generate questions



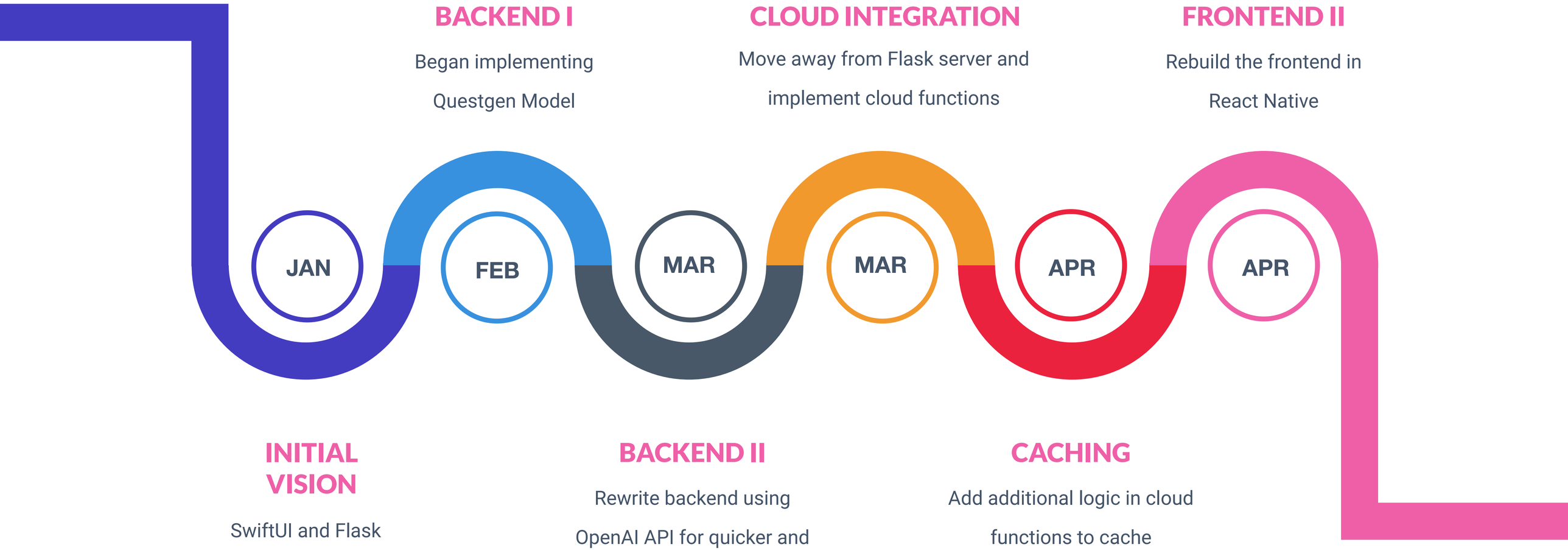
User Flow [FE]


10:13
Create
Explore
Profile


10:14
Home
Enter Topic
Enter Topic:
Cornell Data Science
Next


10:15
Enter Topic
Create
Number of Questions
3
4
5
6
7
Easy
Medium
Hard
Start Quiz

TECHNICAL TIMELINE



INITIAL VISION

SwiftUI and Flask Backend

BACKEND I

Began implementing Questgen Model

BACKEND II

Rewrite backend using OpenAI API for quicker and more reliable results

CLOUD INTEGRATION

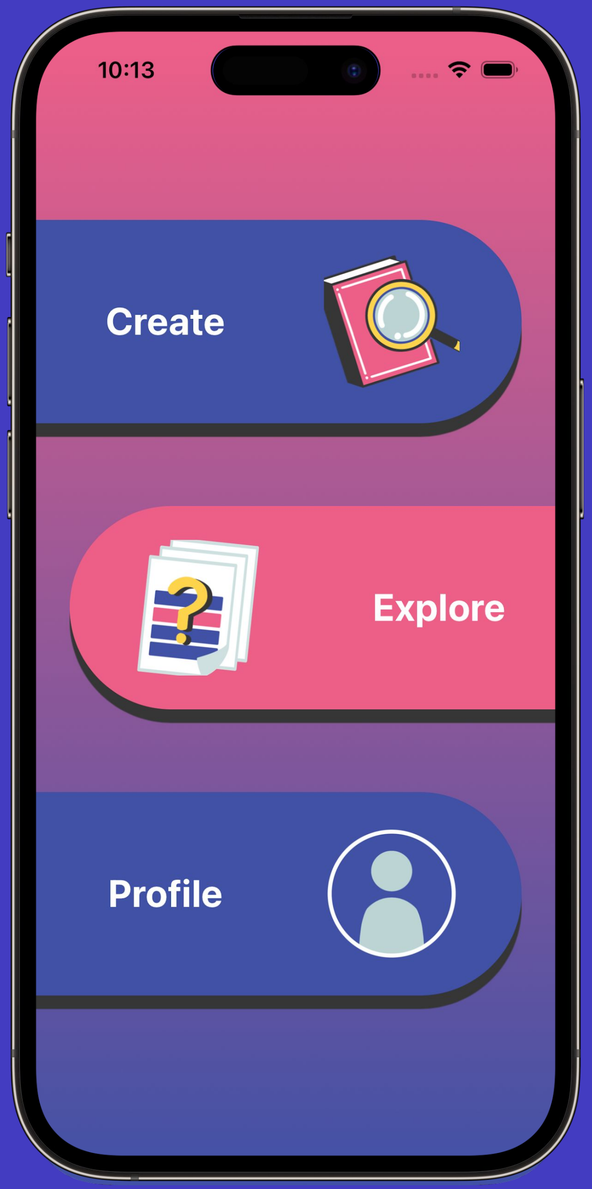
Move away from Flask server and implement cloud functions

CACHING

Add additional logic in cloud functions to cache questions into a database

FRONTEND II

Rebuild the frontend in React Native



Frontend

The frontend of TrivAI is entirely built by the team with **hand drawn graphics**, **template-free frontend**, and **color palette**.

Expo

Supports **Android and desktop**

Real-time editing with almost little to no delays

Wide range of **designs**

Error handling

SwiftUI

Better apps. Less code.

Supports **iOS only**

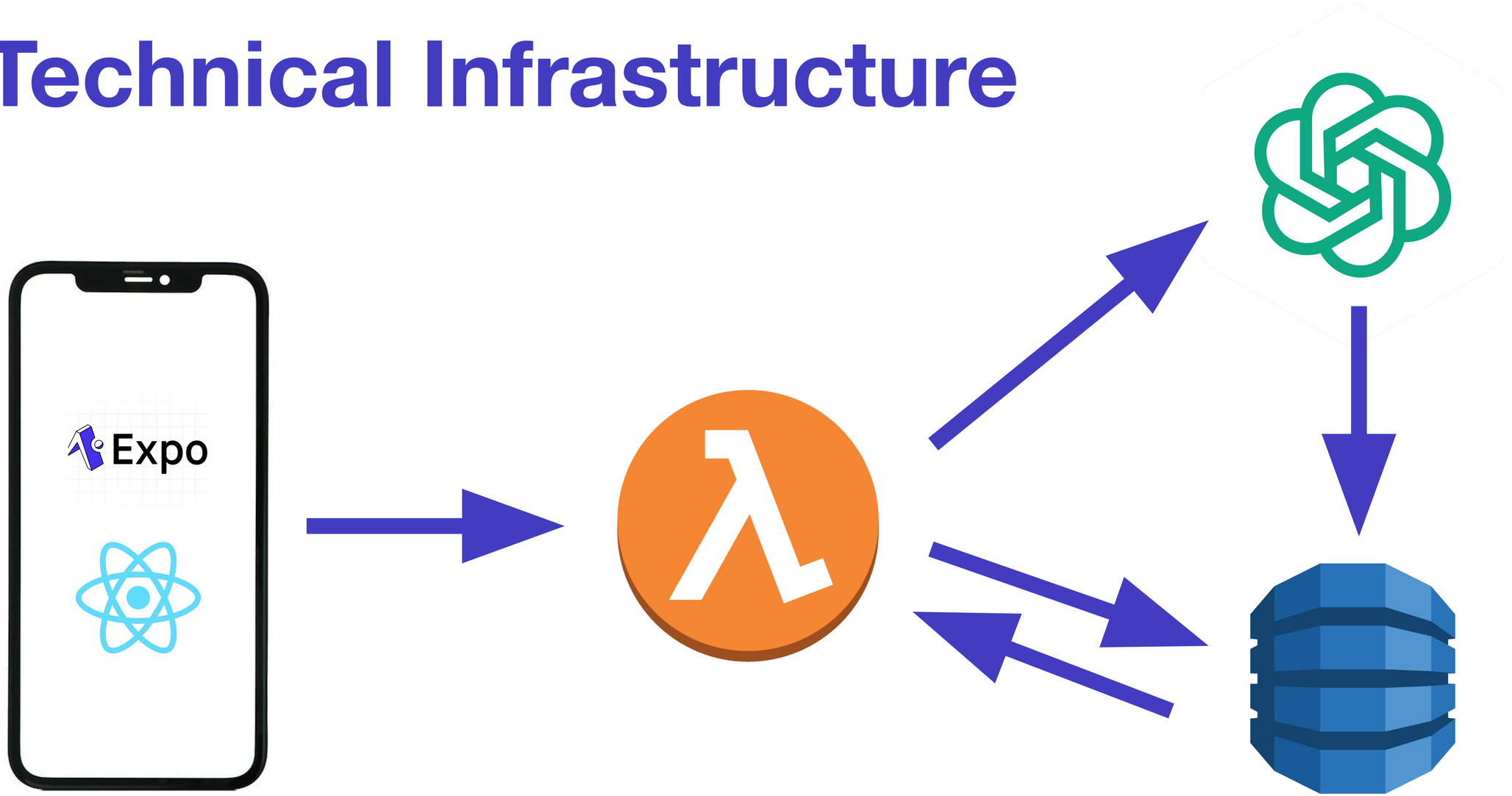
Have to use **XCode** with **slow simulators**

Limited to **10 views**

Buggy implementation

Package issues

Technical Infrastructure



Backend v1.0

Used Questgen and Wikipedia API to generate questions

Deployed as Flask app

Problem

Low quality questions, limited topics, slow computation

Prompt Engineering

```
# get MCQs based on a topic
def get_mcqs_topic(topic, num_questions):
    mcq_topic_prompt = """
    Create %d questions about %s in the following json format:
    [
        {
            "question": question,
            "options": [option 0, option 1, option 2, option 3],
            "answer_id": answer_id
        },
        {
            "question": question,
            "options": [option 0, option 1, option 2, option 3],
            "answer_id": answer_id
        }, ...
    ]
    """ % (num_questions, topic)
    return get_chatgpt(mcq_topic_prompt)
```

Different question types required different prompts

Trial and error – identified effective prompts

Key prompt descriptors: 4 answer choices per question, follow specified output format

OpenAI

GPT3.5-Turbo API

“Our most capable and cost effective model in the GPT-3.5 family”

- OpenAI HR manager

Backend v2.0

AWS Lambda

- Rewrite backend using **AWS Lambda**
- Serverless deployment
- AWS **API Gateway**
- Use **Postman** to send requests for testing

Backend v3.0

DynamoDB

- Uses DynamoDB to cache questions

- Eliminate duplicate API calls

- Good lambda function integration

- Reduce waiting time

Introducing:

Multiplayer.

Multiplayer

Leaderboard

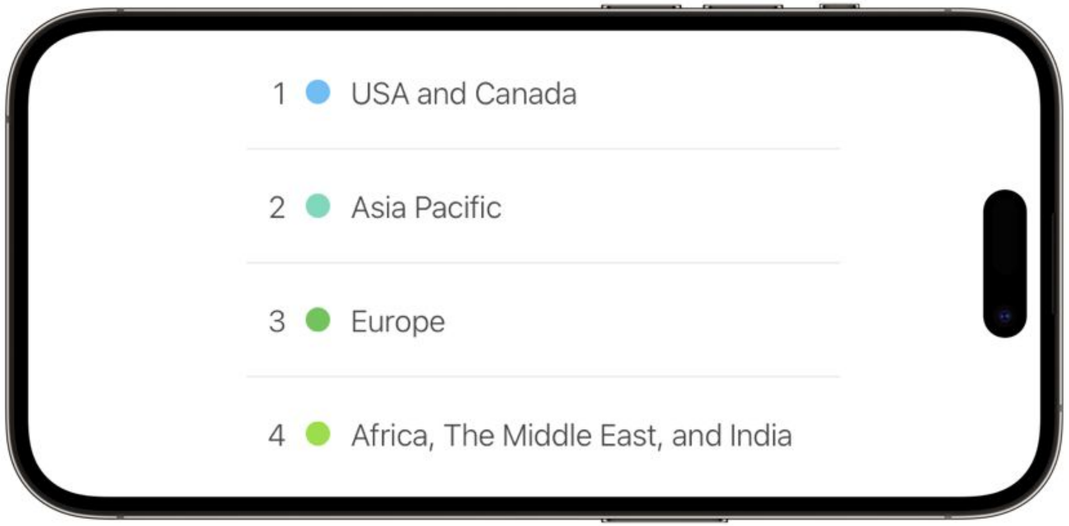
Competition

Saved Games

Play with your friends

Reach

In just one month, TrivAI has been downloaded over 50 times in over 8 countries.



The End.