Fall 2022 ITS Swift App

Neha Lalani, Aahan Kerawala, Sakshi Deshpande, Zhen Hong Tan, Sabina Ajjan, Devang Ajmera

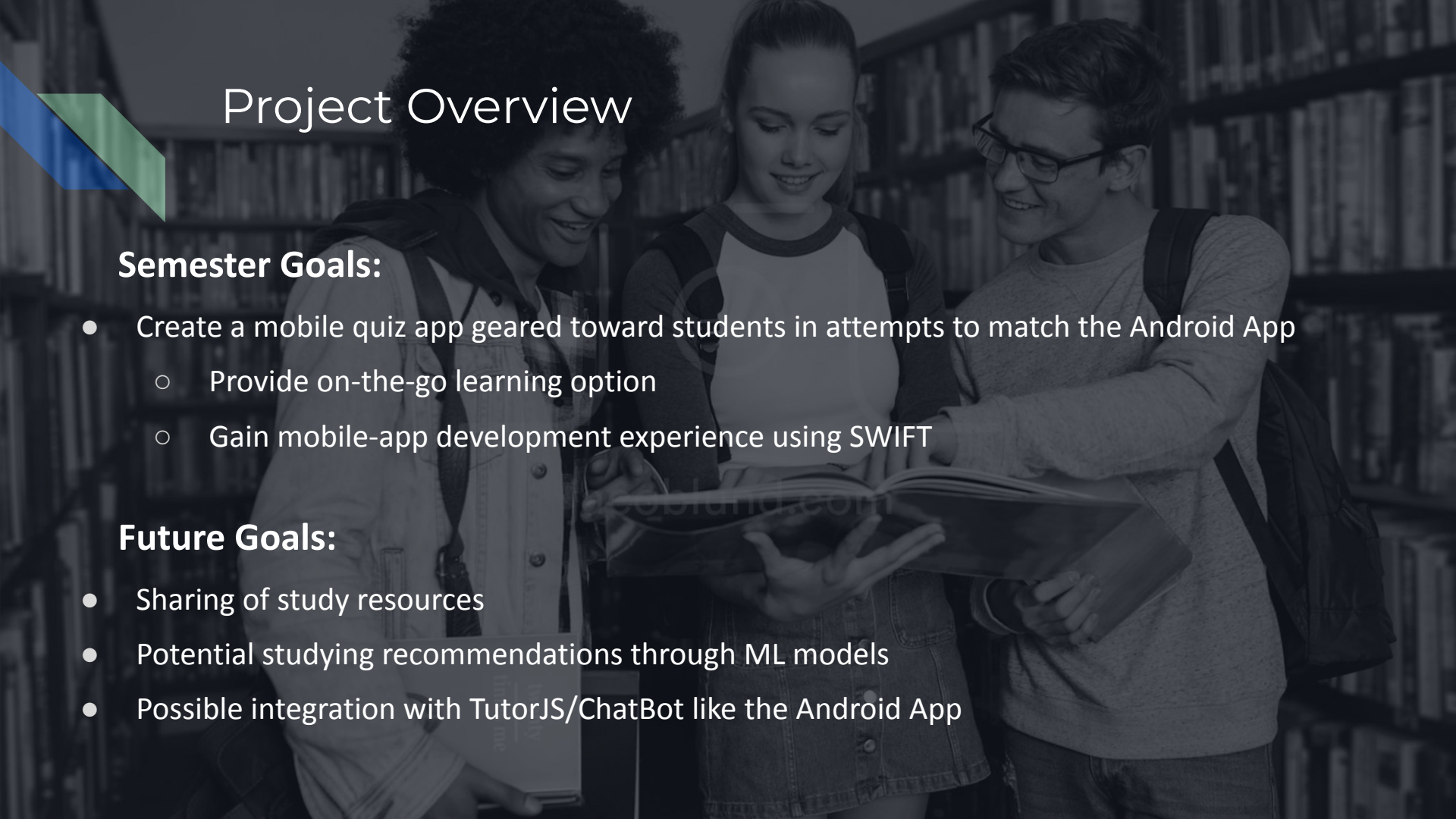
Project Overview

Initial Problem:

Existing ITS tools are not targeted for IOS devices and on-the-go studying

Existing app created to fix this was the Android Application but it was not supported by IOS devices

Project Overview



Semester Goals:

- Create a mobile quiz app geared toward students in attempts to match the Android App
  - Provide on-the-go learning option
  - Gain mobile-app development experience using SWIFT

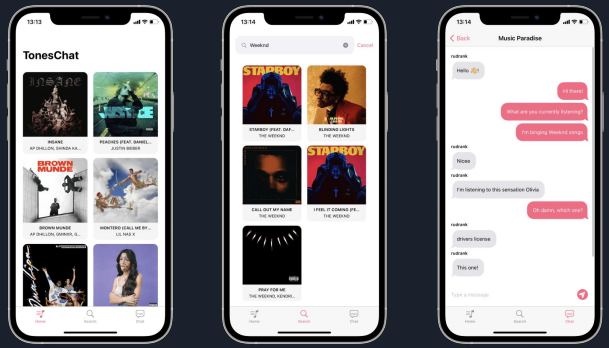
Future Goals:

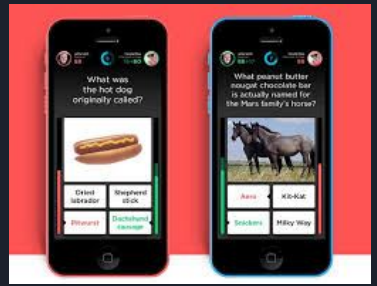
- Sharing of study resources
- Potential studying recommendations through ML models
- Possible integration with TutorJS/ChatBot like the Android App

User Research

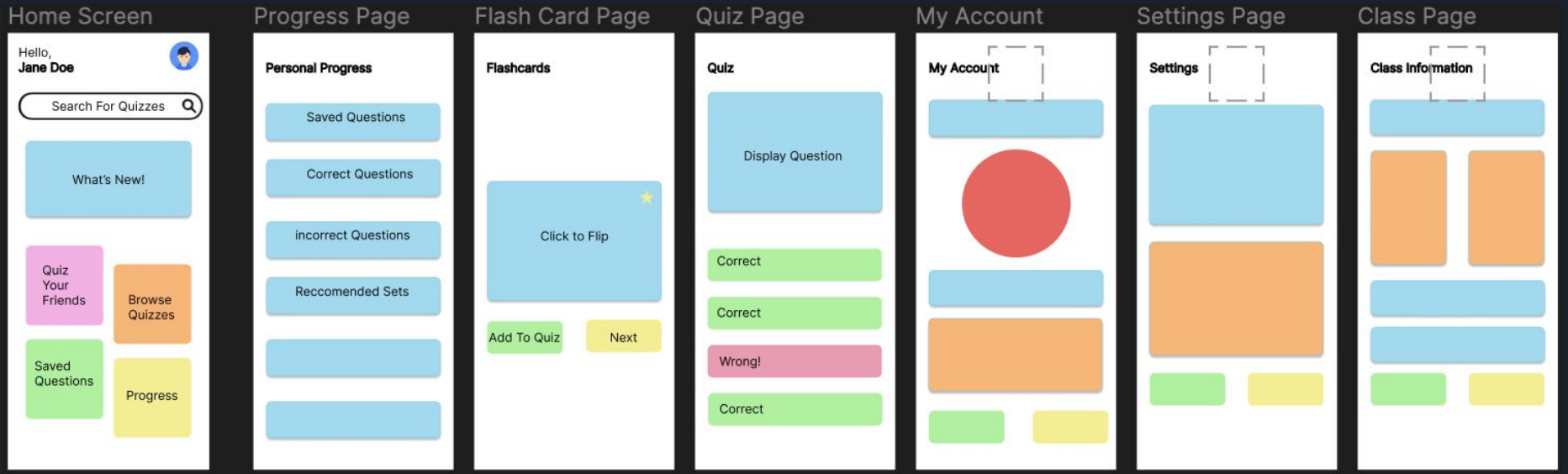
- Looked into various existing quiz applications
  - Kahoot
  - Quizizz
  - Quizlet
- Screens and Features To Include :
  - Login, Registration, Home, Browse Quizzes, Question, Results, Profile, Overall Stats
  - Navigation Bar, Display Progress Statistics
- Question Types:
  - MC
  - Short Answer
  - Flashcard

Research document:
https://docs.google.com/document/d/15algXJnZo9IXTvhdOfe71PjFgjcvsyvNWr_wMwEMzCs/edit

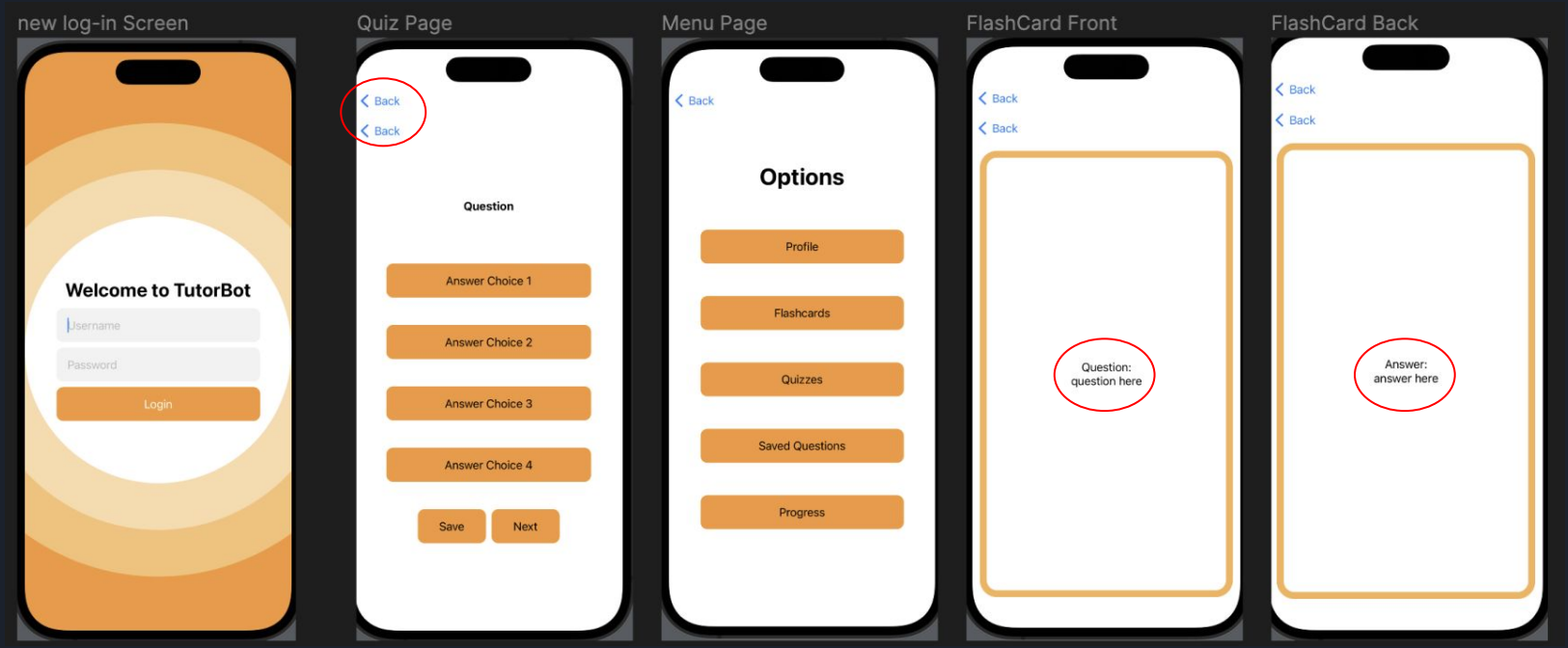




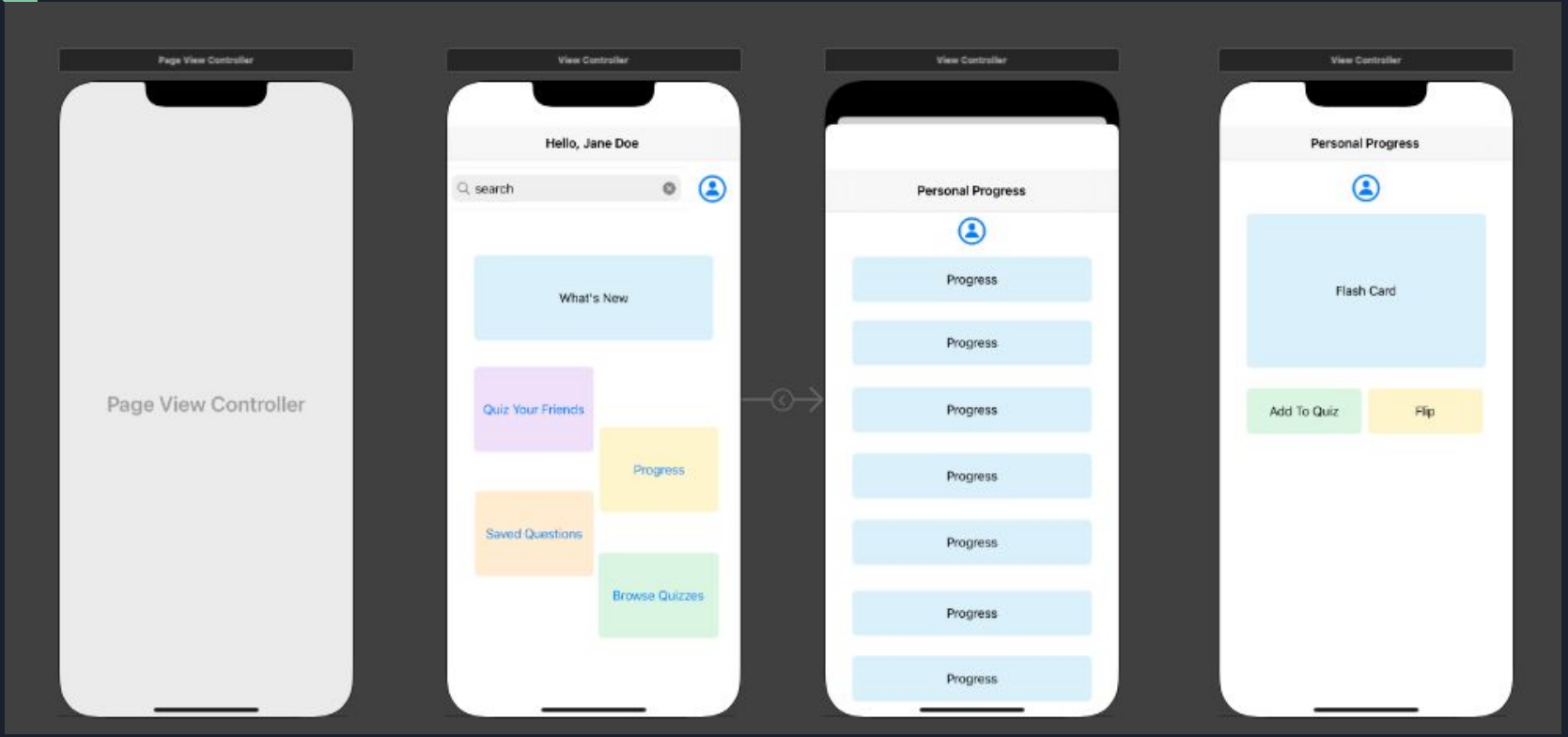
Wireframing - initial UI



Wireframe Continued



Swift StoryBoard

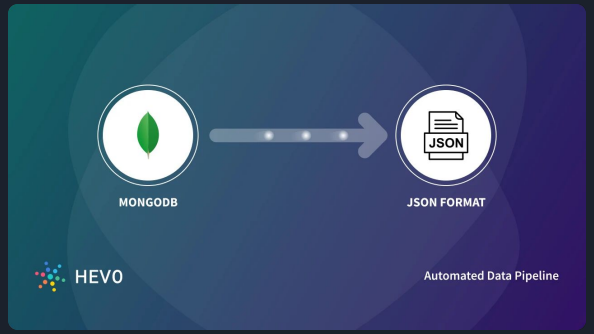


Dataset

The dataset provided is a json file consists of the chapter title, paragraph text containing the questions and the answers, context, and a boolean variable is-impossible that is set to false by default.

To set up a database with the given data, we needed a schema that consisted of a unique id, chapter title, question, answer, and an optional context of the answer as the attributes.

This was done using MongoDB as the database. The reason for choosing MongoDB was that it is classified as a NoSQL database program, and uses JSON-like documents with optional schemas.



{"version": "bookv2.0", "data": [{"title": "Chapter_-2_Section_0", "paragraphs": [{"qas": [{"question": " What is an extension of the real number system?", "id": "ioxa2ehaf7421rurx9v347ewo", "answers": [{"text": "A complex number system", "answer_start": 13, "answer_end": 36}], "is_impossible": false}, {"question": " What are complex numbers necessary to solve?", "id": "4k5craga2g3xk16p94wnj9cez", "answers": [{"text": "equations", "answer_start": 119, "answer_end": 128}], "is_impossible": false}, {"question": " How many solutions does the previous equation have?", "id": "4esyi5gv50aggmw92eezyysz0", "answers": [{"text": "two", "answer_start": 209, "answer_end": 212}], "is_impossible": false}, {"question": " What are numbers needed to solve for the two roots of a quadratic equation?", "id": "rg2yric74d9jkj0ka8w8k2yby", "answers": [{"text": "complex numbers", "answer_start": 240, "answer_end": 255}], "is_impossible": false},

MongoDB Database

- Currently using a MongoDB Atlas cluster to store data about the user and their performance on quizzes
- Motivation for cloud-based preference
- Wrote script to store the generated textbook questions into MongoDB database
    - Three attributes as of now :
        - Questions
        - Answers
        - Chapter

VIP

8 DBS 24 COLLECTIONS

FAVORITE

HOSTS
quizappdb-shard-00-00.7lt…
quizappdb-shard-00-01.7lt…
quizappdb-shard-00-02.7lt…

CLUSTER
Replica Set (atlas-105hqa-…
3 Nodes

EDITION
MongoDB 5.0.13 Enterprise

My Queries

Databases

Filter your data

- Question
  - Question
  - QuizAttempt
  - User
  - UserAnswer
- Questions
  - Data
  - Question

Documents
Questions.Data

Questions.Data

5.9k DOCUMENTS 1 INDEXES

Documents | Aggregations | Schema | Explain Plan | Indexes | Validation

FILTER { field: 'value' } OPTIONS FIND RESET

ADD DATA VIEW

Displaying documents 21 - 40 of 5944 REFRESH

```
_id: ObjectId('633e14c272d9122cce9a7f3c')
Question: "the vertical coordinate is called what?"
Answer: "imaginary part"
Chapter: "Chapter_-2_Section_1"
```

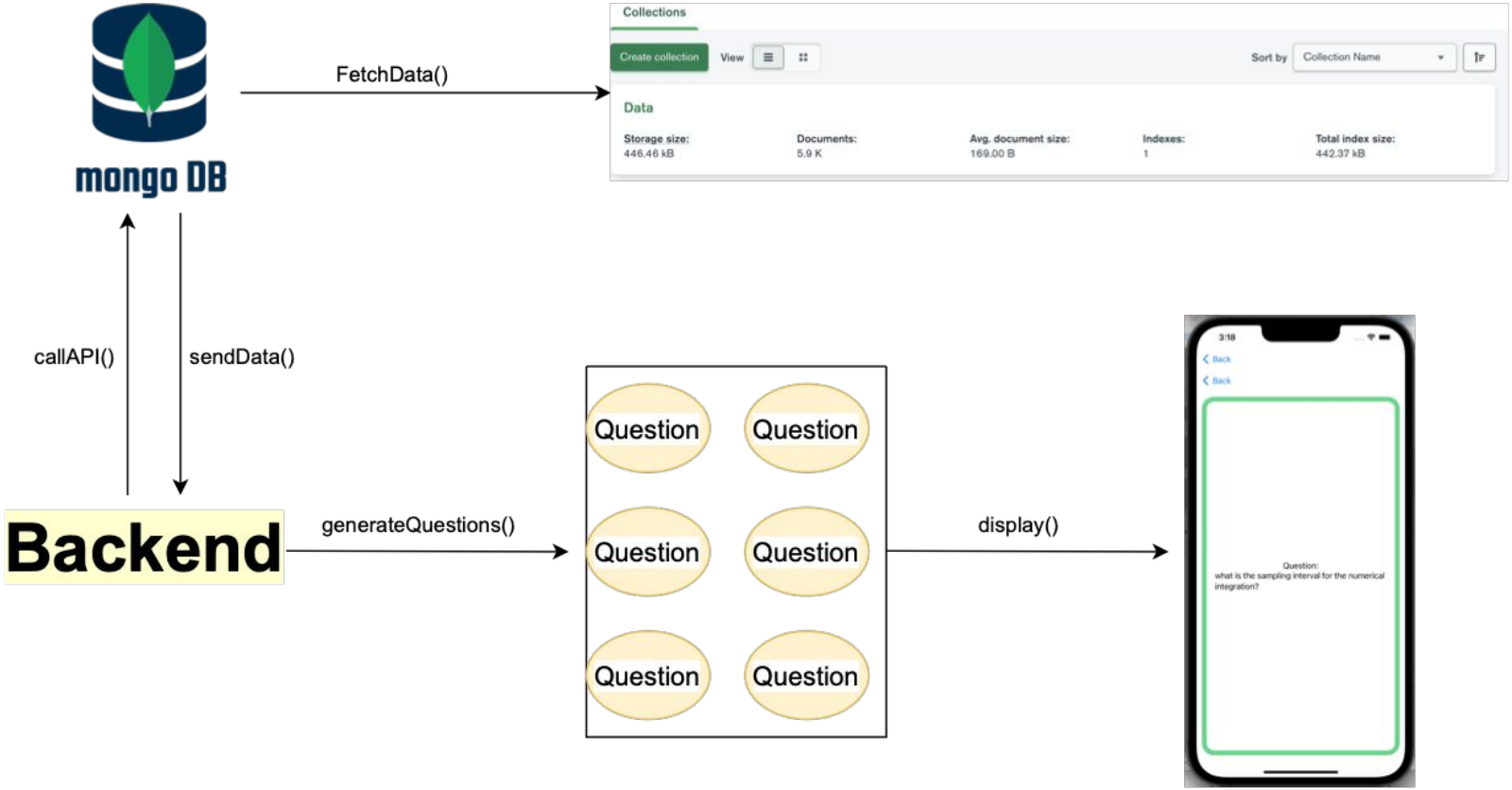
```
_id: ObjectId('633e14c272d9122cce9a7f3d')
Question: "what is called the real part?"
Answer: "The horizontal coordinate"
Chapter: "Chapter_-2_Section_1"
```

```
_id: ObjectId('633e14c272d9122cce9a7f3e')
Question: "what is the vertical coordinate called?"
Answer: "imaginary part"
Chapter: "Chapter_-2_Section_1"
```

```
_id: ObjectId('633e14c272d9122cce9a7f3f')
Question: "the operators and are provided to extract what?"
Answer: "real and imaginary parts of"
Chapter: "Chapter_-2_Section_1"
```

>_MONGOSH

Database + Backend Data Flow



Database Connection

- MongoSwiftSync Library
  - Connect to the database directly
  - Not intended to use for ios applications
- Realm.io
  - Real-time mobile to cloud data sync
  - Have to move over the database to realm

- Currently reading in a JSON file and make changes back to the JSON file

What I Have Learned This Semester - Sabina

Backend Work:

- requires a database system such as MongoDB, MySQL, etc.
- MongoDB is similar to MySQL except it does not really use tables, and the data can be exported as a JSON file
- First you must get the questions from the JSON file and upload them to MongoDB. We used python to do this.
- Once the data is uploaded to the database, you must upload the data to your project.
- Using Swift, we were able to load the data by further reading the JSON file and display the questions on the screen

```python
import json
import collections
import pymongo
from pymongo import MongoClient

# Connect to the database
cluster = MongoClient('mongodb+srv://quizAppMobileUser:xkLaBBaa1dTsZuX8@quizappdb.7ltdg.mongodb.net/test')
# Selecting Database
db = cluster['Questions']
# Selecting Collections
collection = db["Data"]

def add_data():
    with open("textbook-v1.0-1.json") as json_file:
        data = json.load(json_file)
        data = data['data']
        for v in data:
            for value in v['paragraphs']:
                for value2 in value['qas']:
                    question = value2['question'].strip().lower().replace('"', "'")
                    title = v['title'].strip()
                    answer = 'unknown'
                    if value2['is_impossible'] == False:
                        answer = value2['answers'][0]['text']
                    print(question)
                    print(answer)
                    print(title)
                    query = {
                        'Question': question,
                        'Answer': answer,
                        'Chapter': title
                    }
                    collection.insert_one(query)
```

```swift
//

import Foundation

public class DataLoader {
    @Published var userData = [UserData]()

    init() {
        load()
        sort()
    }
    func load() {
        print("Here")
        if let fileLocation = Bundle.main.url(forResource: "Data", withExtension: "json") {
            do {
                let data = try Data(contentsOf: fileLocation)
                let jsonDecoder = JSONDecoder()
                let dataFromJson = try jsonDecoder.decode([UserData].self, from: data)
                self.userData = dataFromJson
            } catch {

                print(error)
            }
        }
    }

    func sort() {
        self.userData = self.userData.sorted(by: { $0.Chapter < $1.Chapter})
    }
}
```

```
let realm = try! Realm()
var token: NotificationToken?
// Read from realm
try! realm.write {
    realm.write()
}

//  Set up the listener & observe object notifications.
token = realm.observe { change in
    switch change {
    case .change(let properties):
        for property in properties {
            print("Property '(property.name)' changed to '(property.newValue!)'");
        }
    case .error(let error):
        print("An error occurred: (error)")
    case .deleted:
        print("The object was deleted.")
    }
}
 */
    ZStack {
        var num = Int.random(in: 1..<5500)
        CardFront(degree: $frontDeg, textContext: data[num].Question)
        CardBack(degree: $backDeg, textContext:  data[num].Answer)
    }.onTapGesture {
        flipCard()
    }
```

```
struct CardFront: View {
    @Binding var degree : Double
    let textContext : String

    var body: some View {
        ZStack {
            RoundedRectangle(cornerRadius: 20).stroke(.green.opacity(0.5), lineWidth: 10).padding()

            RoundedRectangle(cornerRadius: 20).stroke(.green.opacity(0.5), lineWidth: 10).padding()

            VStack {
                Text("Question:")
//                Text("New question 1")

                Text (textContext)
                    .lineLimit(10)
//                Text("answer here")

            }
        } .rotation3DEffect(Angle(degrees: degree), axis: (x: 0.0, y: 1.0, z: 0.0))
    }
}
```

Some things I have learned about Swift:

if you were to compare it to java, structures are very similar to classes (except they are value types not reference types).They contain variables and behaviors and you can instantiate them in other classes, as is done here.

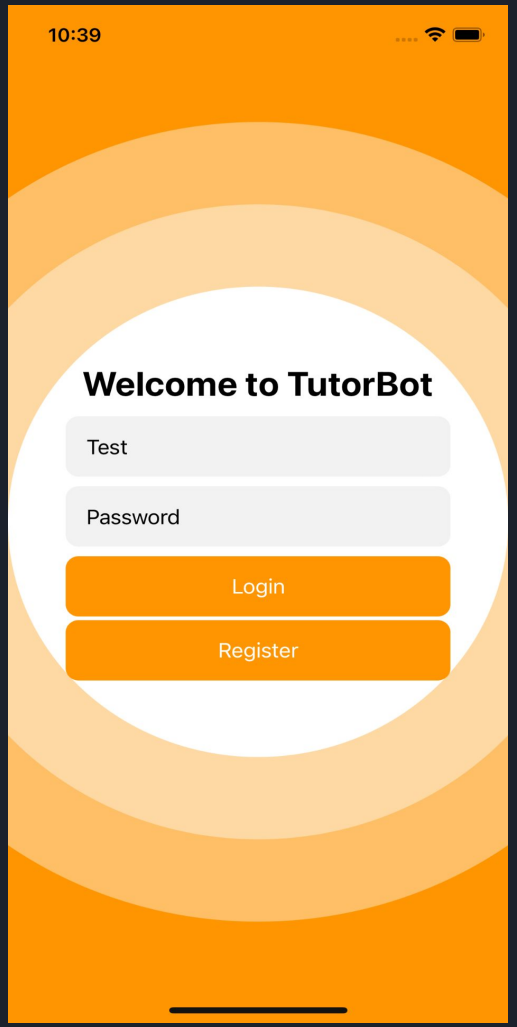
A vstack is kind of like Vbox if you were to compare it to javaFX

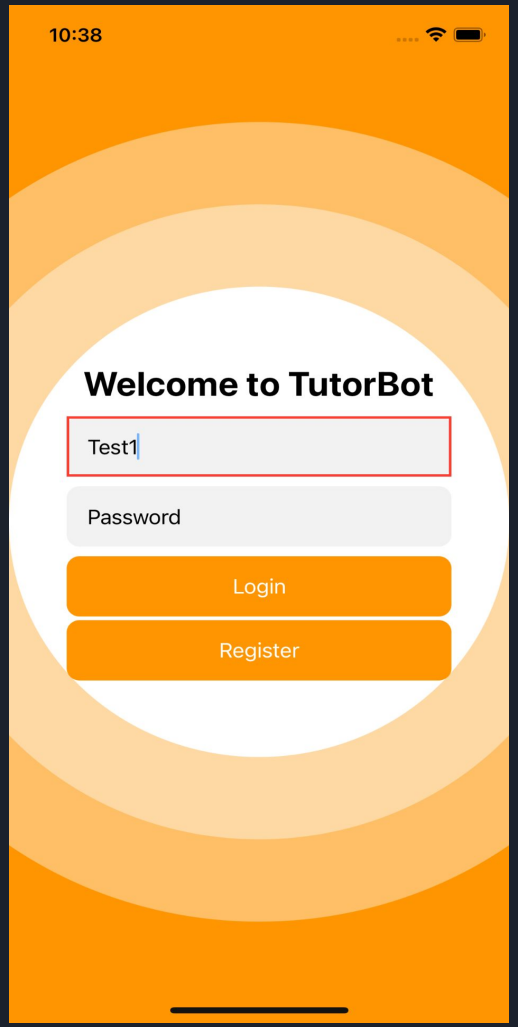
Login Screen

A User launches App

The user encounters a login screen where they can input their username and password

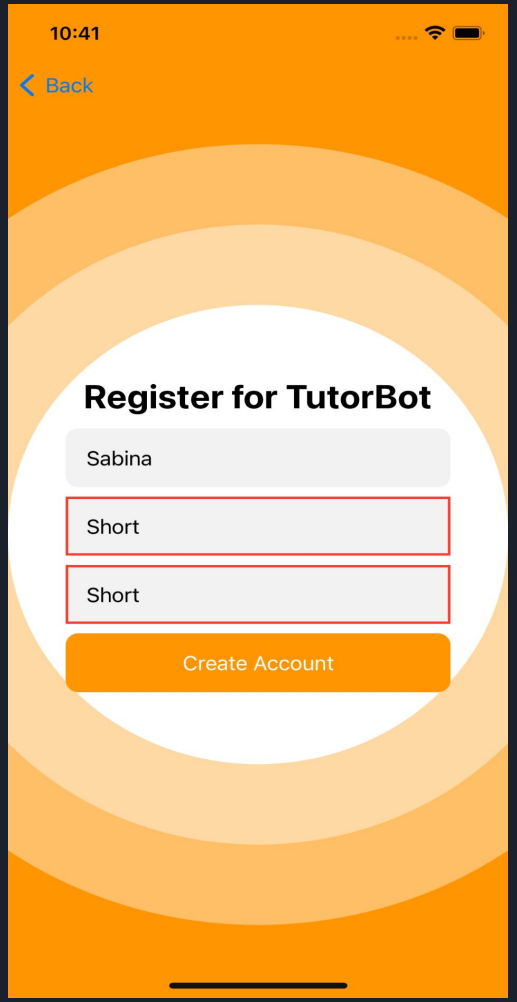
If the username or password is wrong it will highlight the wrong entry with a red box

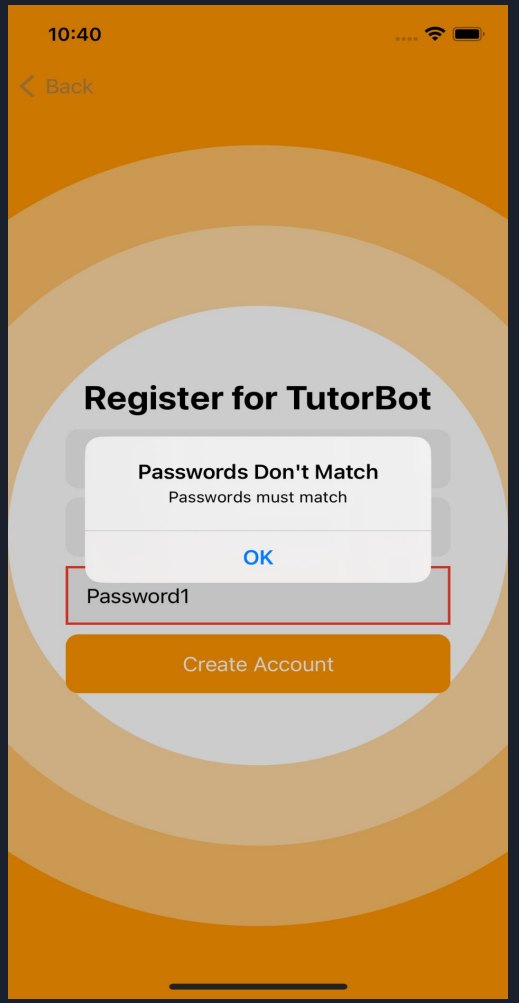




Registration Screen

- Password must be longer than 8 characters
- Username must be longer than 5 characters
- Passwords have to match
- Next semester, this registration page will be connected to the database and also check if the username is already taken



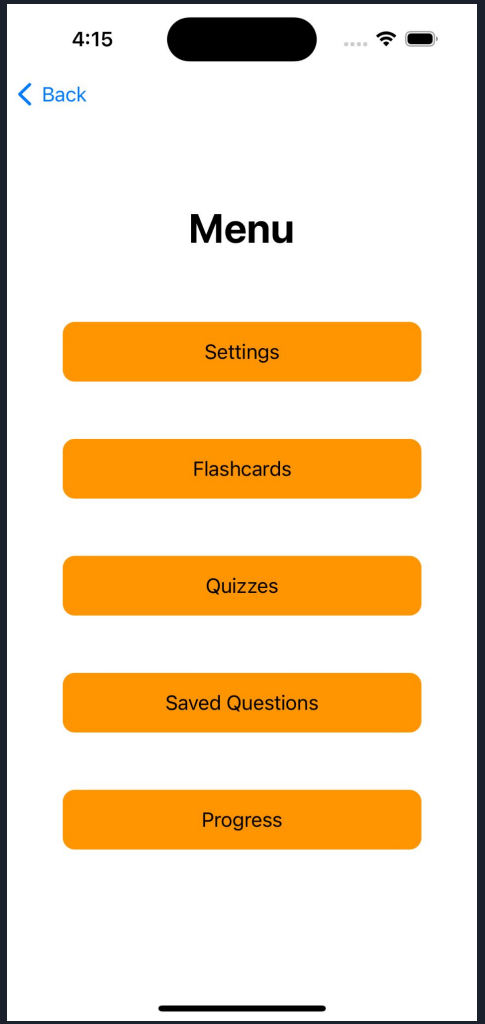


Home Screen/Menu Options

After logging in, the user is brought to the Home screen

The use can choose on multiple options to access

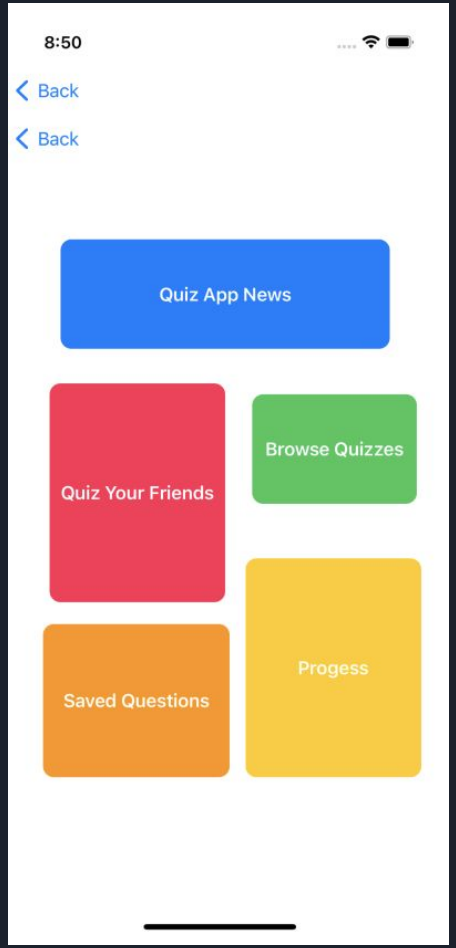
Each option is a button that takes the user to the page selected



Profile Screen

The user is able to see their profile and other elements within the profile

Overtime many elements of this screen will be connected to the backend such as saved questions.



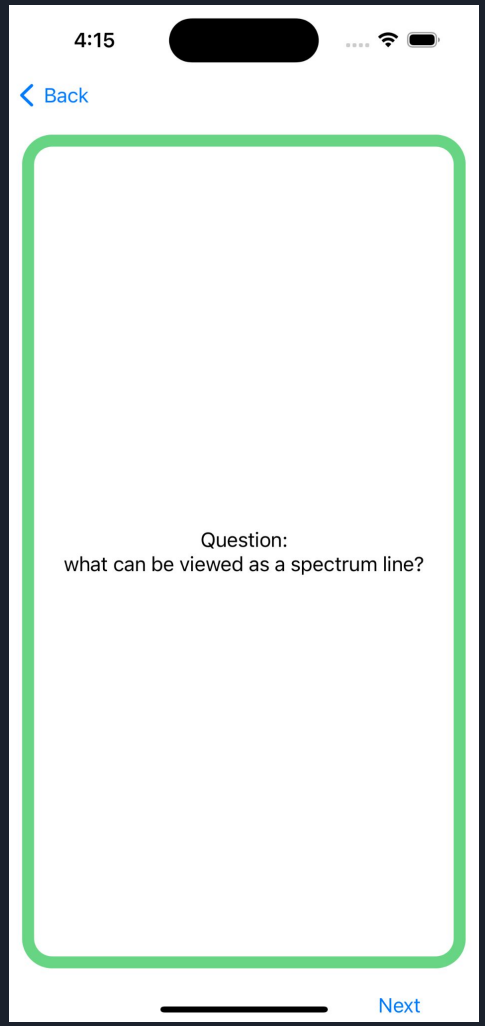
Flashcard Question Screen

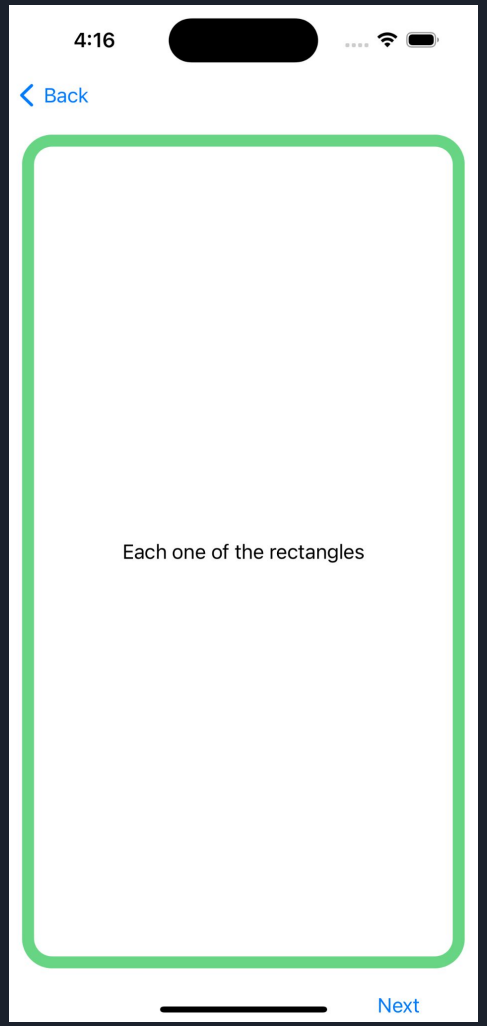
The user proceeds to the quiz screen where they can see flashcards that take in back end data

This is an example of the front of a Flashcard

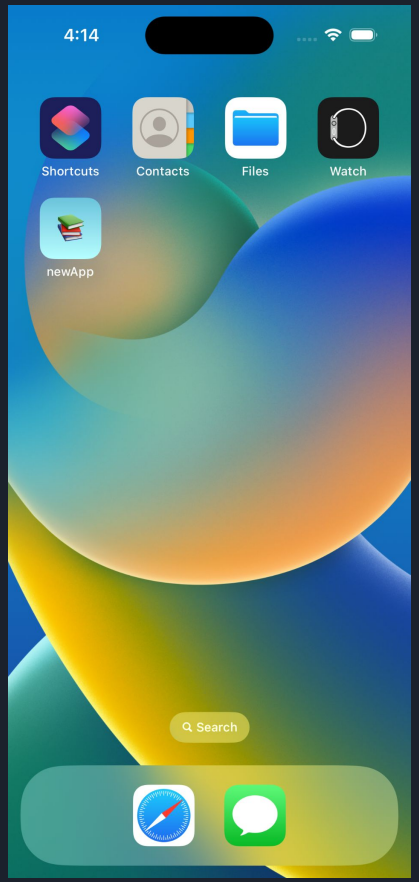
Next button created to take user to next card

Back buttons are fixed





App Icon Created



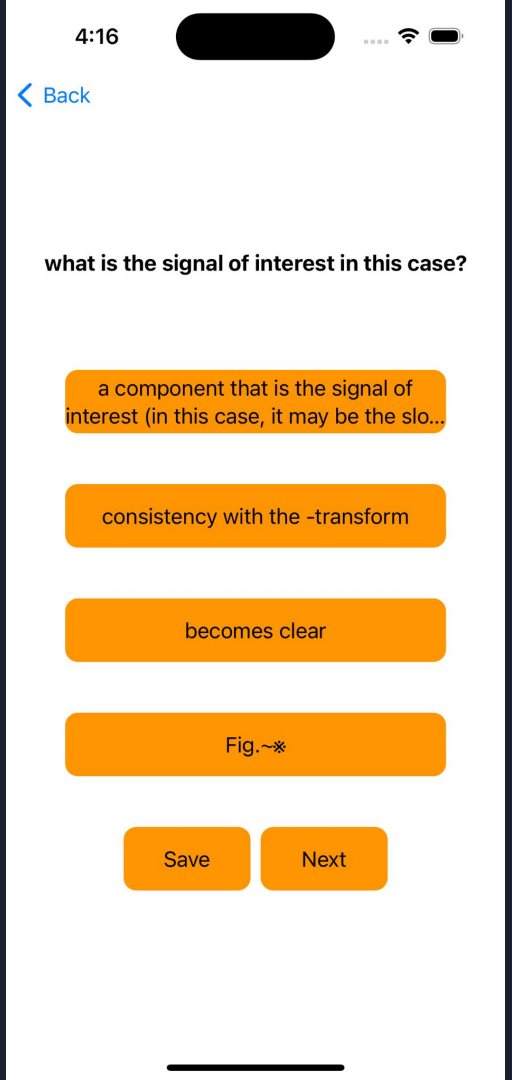
Quiz Practice Screen

The User selects Quizzes

Currently only the template for the quizzes

There will be question with 4 answer choices, only one being correct

The user quill have the ability to save the question and go to the next question when ready

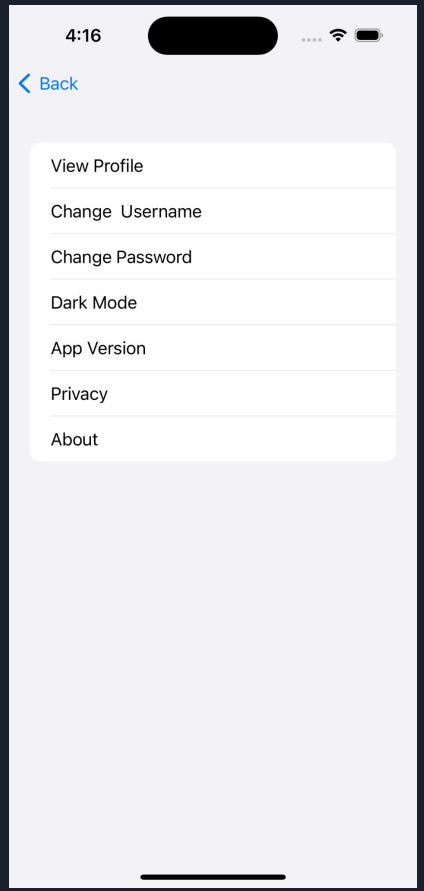


Settings Page

Currently just for visual effects

Will have integration to change their user experience and account data

Want this to be an extension of the Profile page



Live Demo

Tasks From Demo:

- Create an account option for users
- Implement next button on flash cards page functionality
- ~~Back button functionality~~
- Short answer questions
- Quiz sets page creation
- ~~Set up a backend server~~