



QuizApp

Frontend:

Abdulaziz Memesh

Sanjit Pingili

Sahana Krishnan

Varun Patel

Samarth Parameswar

Backend:

Rishi Nopany

Shreekrishna Bhat

Fall 2022 VIP ITS



Introduction

What is QuizApp?

An Android mobile application that uses AI based features to guide and support ECE 2026 (Intro Signal Processing) students in their learning of the course content

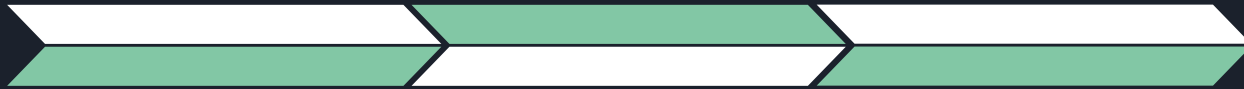
Motivation

Existing ITS tools are not targeted for mobile devices or on-the-go studying



Introduction

Timeline



Fall 2021

- QuizApp started
- Object-oriented design was created
- Major components were implemented in the UI

Spring 2022

- UI was improved
- Functionality of creating accounts was incorporated
- Backend was set up

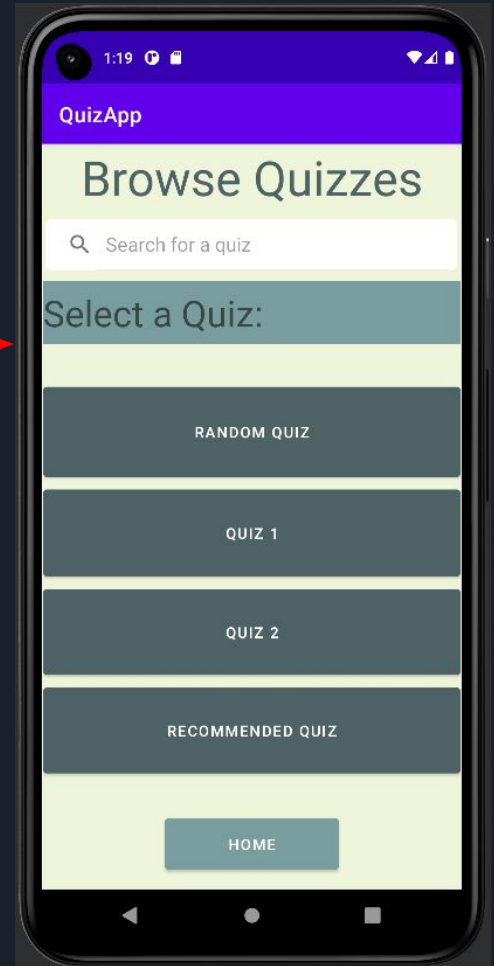
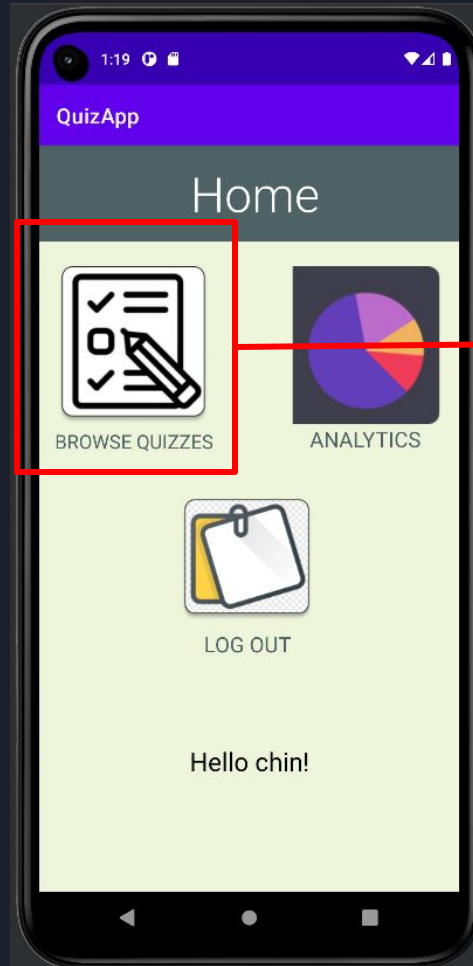
Fall 2022

- You'll see!

Last Semester

Problem 1:

Ambiguous design





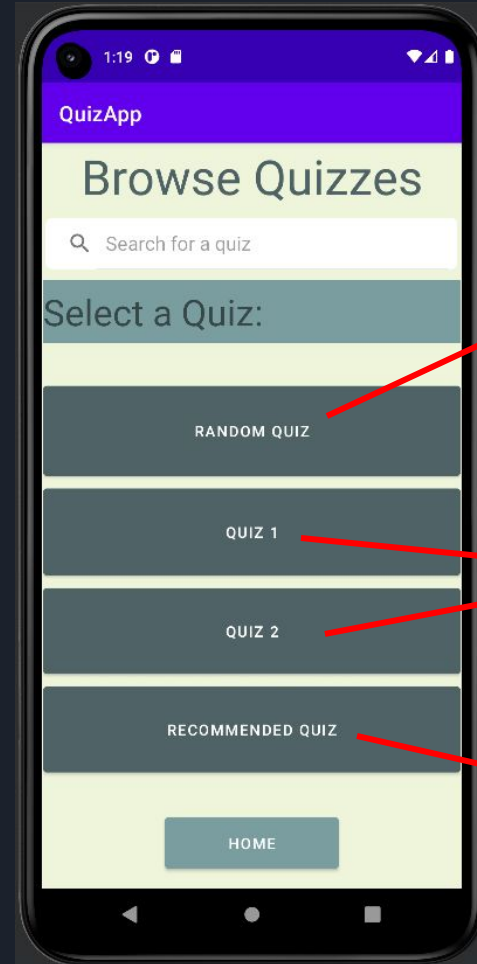
Semester Goals

- **Goal 1: Implement a clearer design.**
- 

Last Semester

Problem 2:

Non-intelligent question generation




Not what we want

Hard coded

Doesn't actually recommend a quiz



Semester Goals

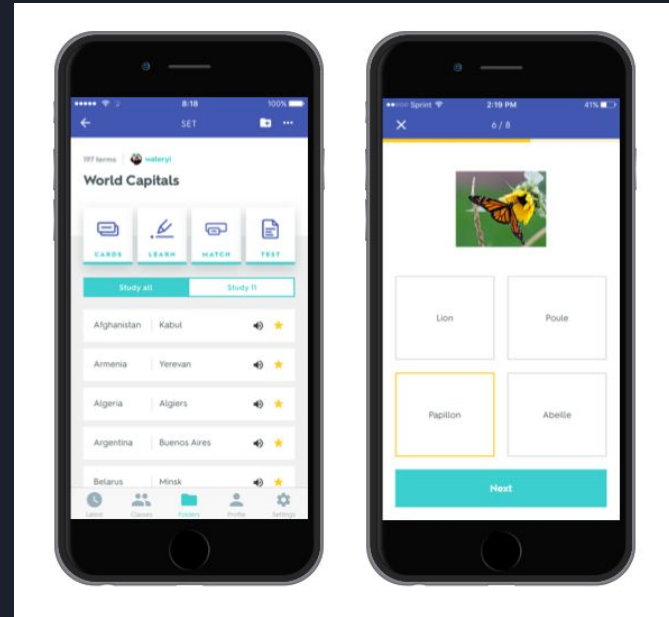
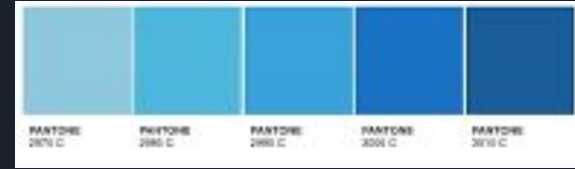
- **Goal 1:** Implement a less ambiguous design
 - **Goal 2:** Use user data for intelligent question generation
- 

Frontend



Research Findings

- Color Scheme
 - Increase in productivity
- Study mode vs quiz mode
 - Mode levels for users to thoroughly study or for quick refresher
 - Apps such as Quizlet and MedCalX
- Navigation bar
 - Efficiency purposes
 - Additional elements for stronger application interaction

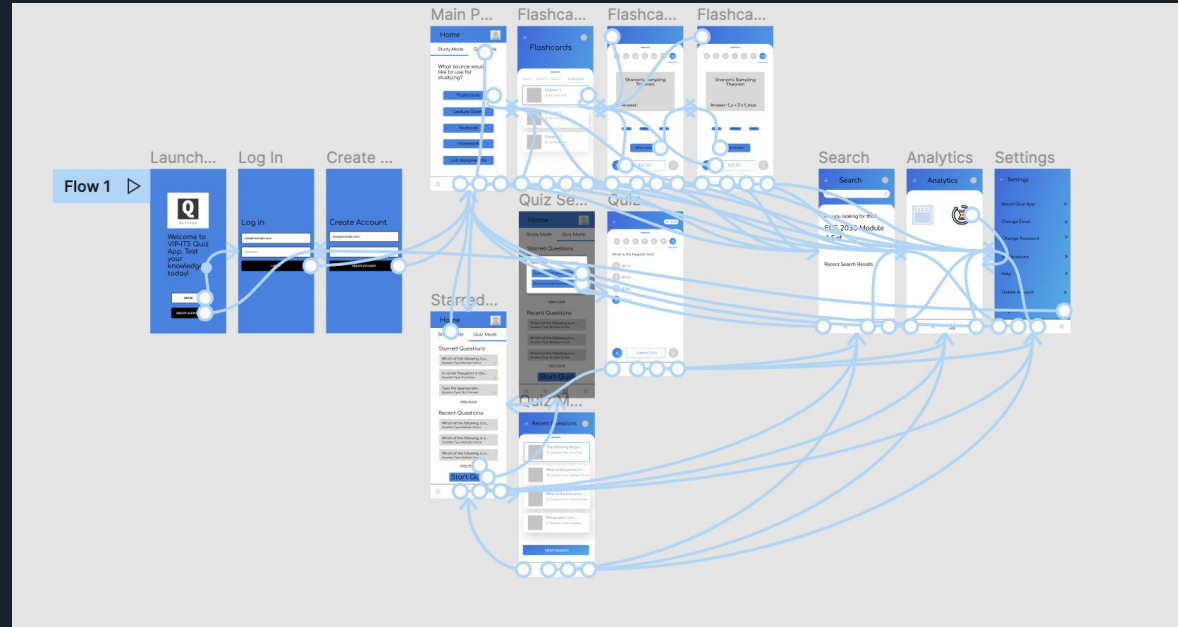




Design

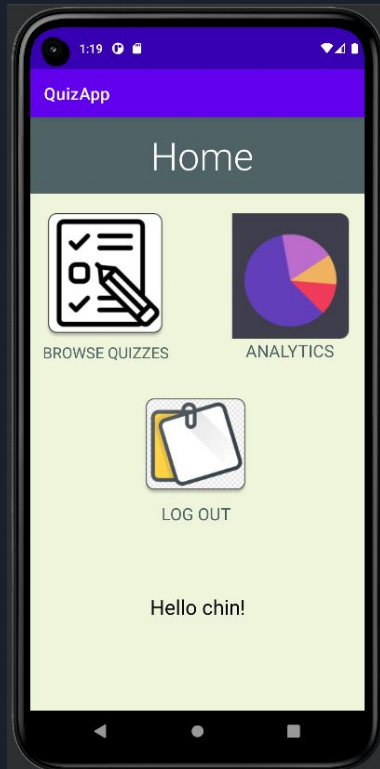
- Figma
 - We used figma to outline how we envisioned QuizApp to ideally look like
- We created how we wanted each page to look like
 - Login page
 - Register page
 - Home page
 - Flashcard page
 - Quiz page
- Wired each page to each other
- Serves as the foundation to base the frontend off and for future semesters

Design: Figma

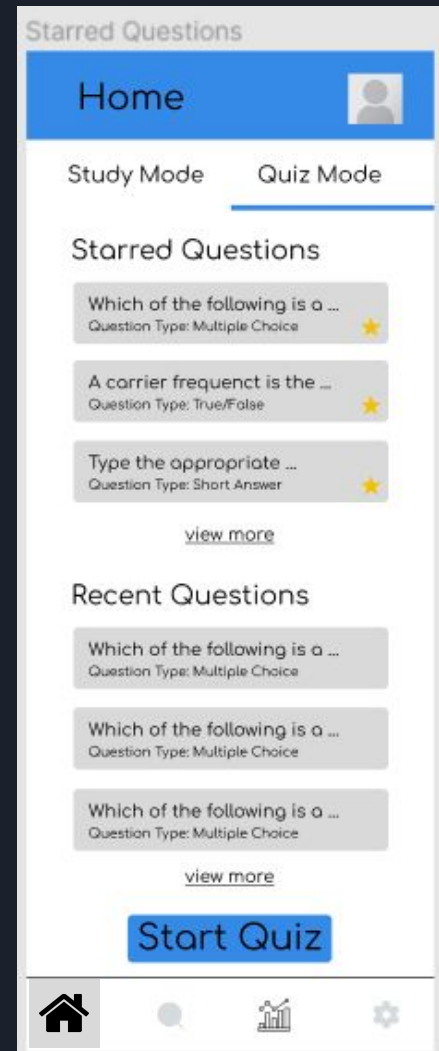
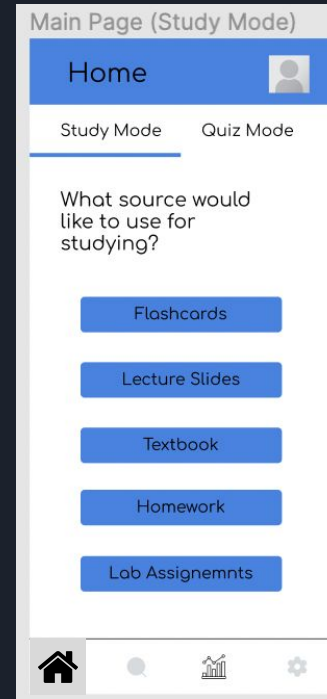


Design: Home Page

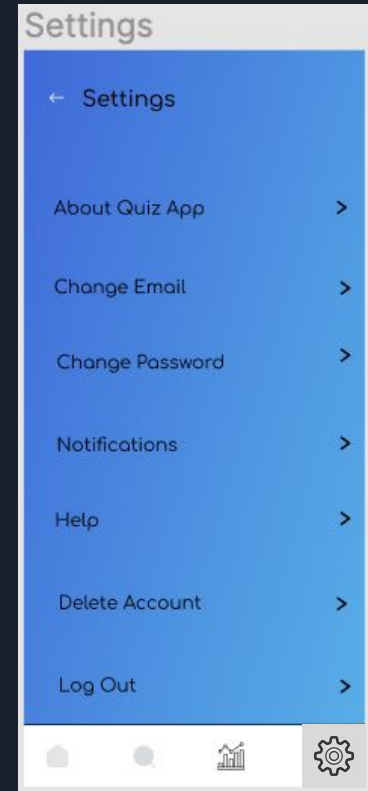
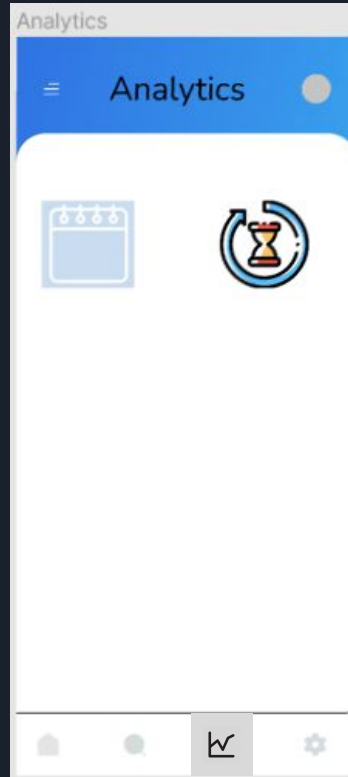
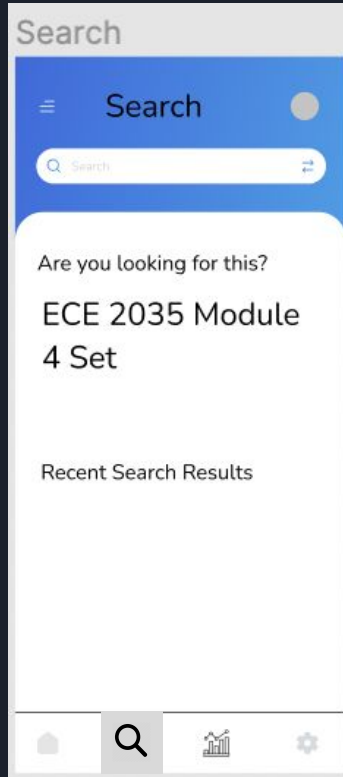
Old design



New design

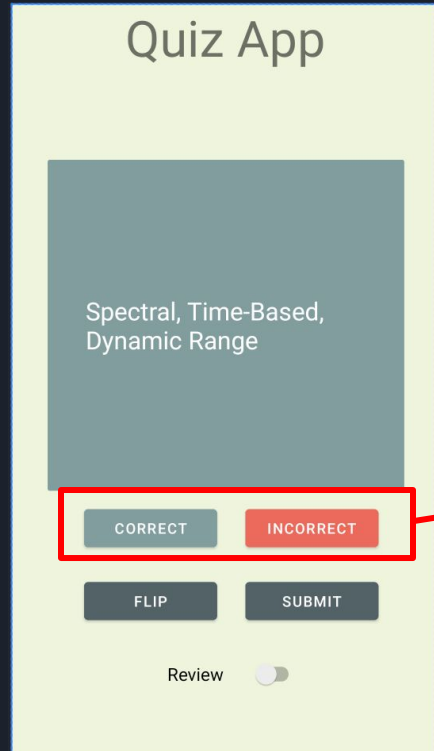


Design: Navigation Bar

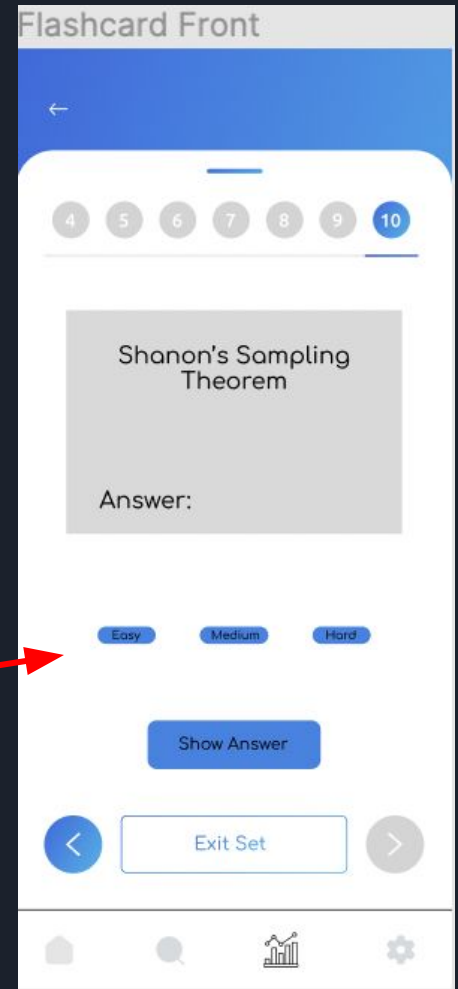


Design: Flashcard back redesign

Old design



New design



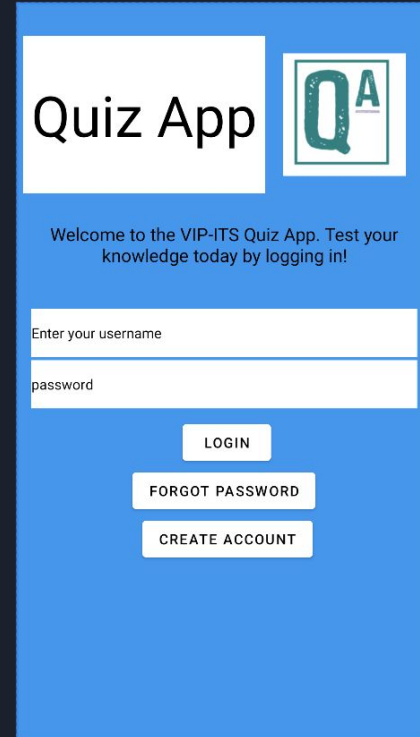
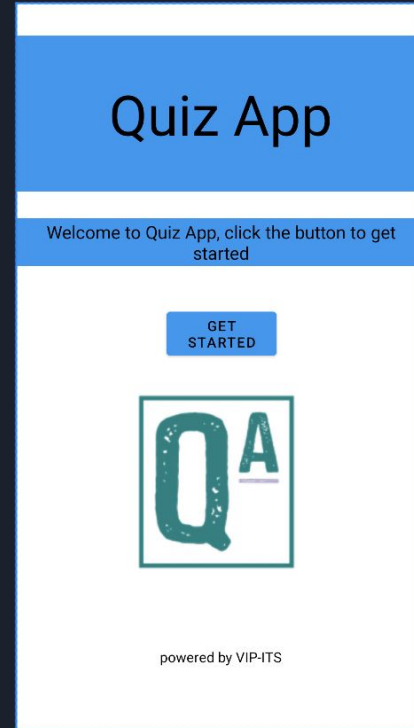
Replaced by difficulty levels

Frontend Implementation



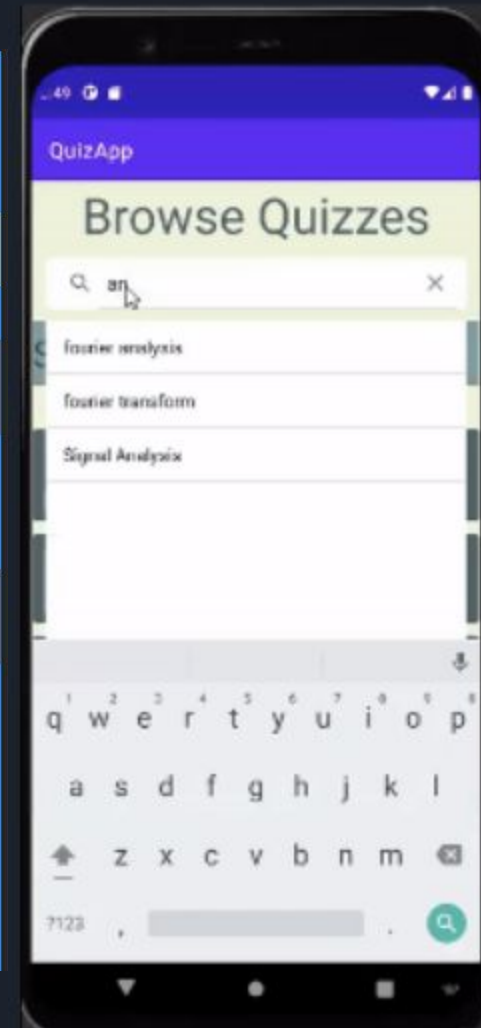
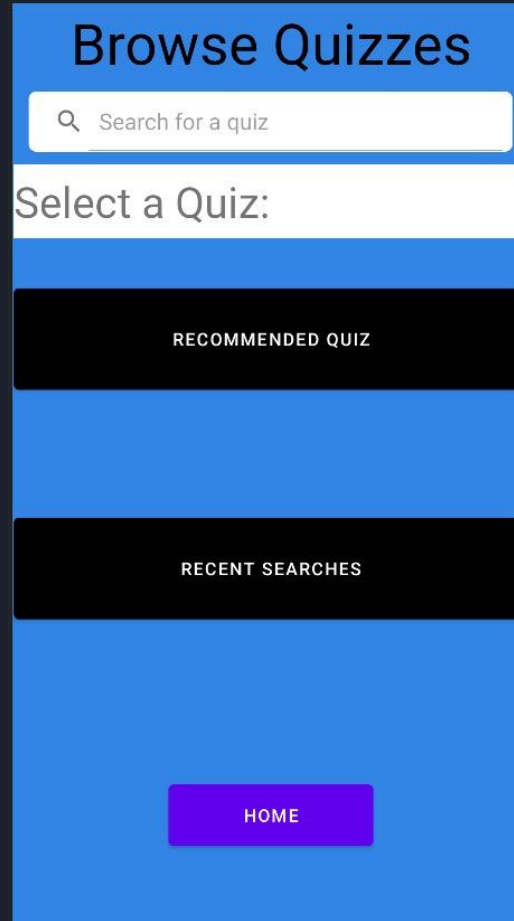
Welcome screen + Email functionality

- There was no welcome screen before
 - Init on create account
 - Fixed bug
 - Added email implementation
- Welcome Screen → initialization of app
 - Changed in Android Manifest
- Fixed bug that made Create Accounts start up
- Made login page start, not create acct



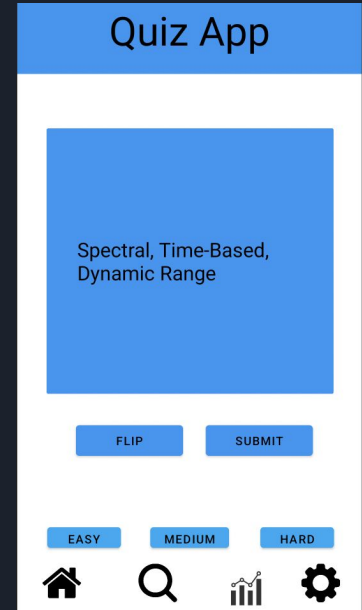
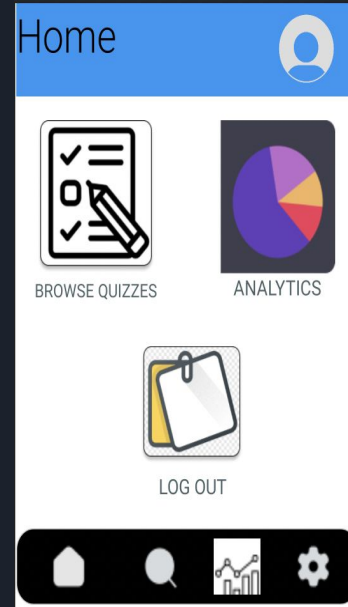
Search Page

- Made the search page more intuitive for user by adding features such as autocomplete, recommended quizzes and recent searches
- As discussed before we changed the color of the page itself
- Implemented “Recent Searches” and “Recommend Quiz” button to help users keep track of past learning and look into the future



Navigation Bar

- Implemented navigation bar
 - Carries through
- 4 icons on navigation bar
 - Home -> activity_main.xml
 - Search -> activity_browse_quizzes.xml
 - Analytics -> activity_analytics.xml
 - Settings -> activity_settings.xml



Navigation Bar - Code

```
99 <LinearLayout
100     android:layout_width="match_parent"
101     android:layout_height="74dp"
102     android:background="@color/white"
103     android:orientation="horizontal"
104     app:layout_constraintBottom_toBottomOf="parent">
105
106     <Space
107         android:layout_width="30dp"
108         android:layout_height="match_parent" />
109
110     <LinearLayout
111         android:layout_width="20dp"
112         android:layout_height="match_parent"
113         android:layout_weight="1"
114         android:orientation="horizontal">
115
116         <ImageView
117             android:id="@+id/home_icon"
118             android:layout_width="46dp"
119             android:layout_height="match_parent"
120             android:contentDescription="TODO"
121             android:onClick="launchHomeScreen"
122             app:srcCompat="@drawable/home_icon" />
123     </LinearLayout>
124
125     <Space
126         android:layout_width="63dp"
127         android:layout_height="match_parent" />
128
129     <LinearLayout
130         android:layout_width="40dp"
131         android:layout_height="match_parent"
132         android:layout_weight="1"
```

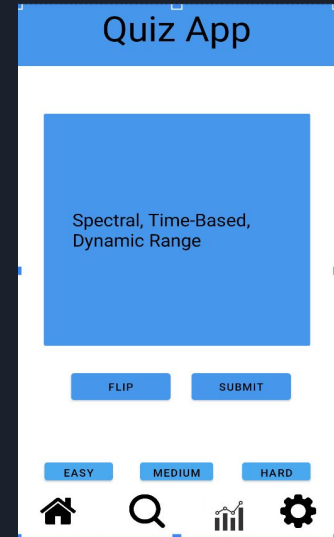
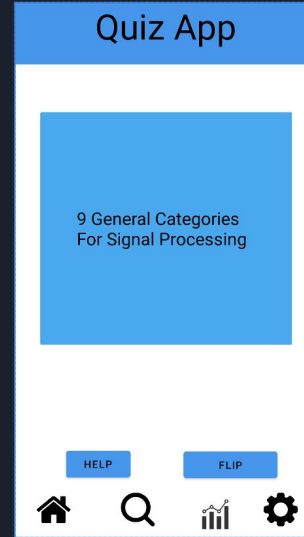
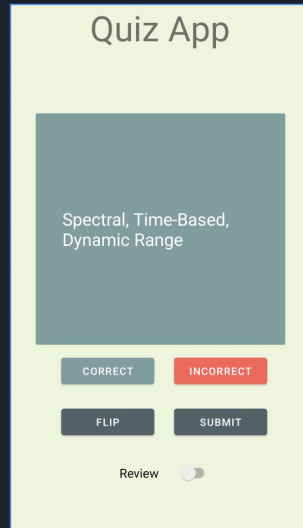
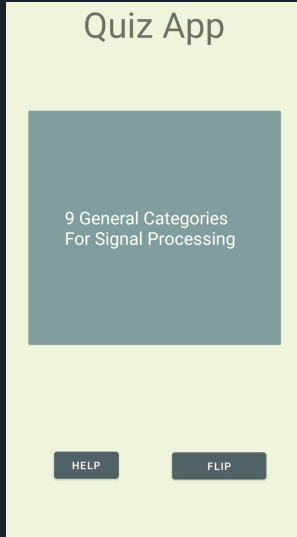
.xml File

```
54 @SuppressWarnings("SetTextI18n")
55 public void launchHomeScreen(View view) {
56     outputText.setText("You clicked Home. Taking you to Home screen.");
57     Intent intent = new Intent( packageContext: MainActivity.this, MainActivity.class);
58     startActivity(intent);
59 }
60
61 @SuppressWarnings("SetTextI18n")
62 public void launchAnalytics(View view) {
63     outputText.setText("You clicked Analytics. Taking you to analytics screen.");
64     Intent intent = new Intent( packageContext: MainActivity.this, AnalyticsActivity.class);
65     startActivity(intent);
66 }
67
68 @SuppressWarnings("SetTextI18n")
69 public void launchSearch(View view) {
70     // outputText.setText("You clicked Search. Taking you to search screen.");
71     // Intent intent = new Intent(MainActivity.this, AnalyticsActivity.class);
72     // startActivity(intent);
73     // }
74
75 // @SuppressWarnings("SetTextI18n")
76 // public void launchSettings(View view) {
77 //     outputText.setText("You clicked Settings. Taking you to settings screen.");
78 //     Intent intent = new Intent(MainActivity.this, AnalyticsActivity.class);
79 //     startActivity(intent);
80 // }
81
82 @Override
83 public void onBackPressed () { return; }
84 }
```

.java File



Flashcard redesign



- Previous flash cards had boring green design
- Added difficulty buttons
- Removed Incorrect/Correct buttons, Flag for review (kept)

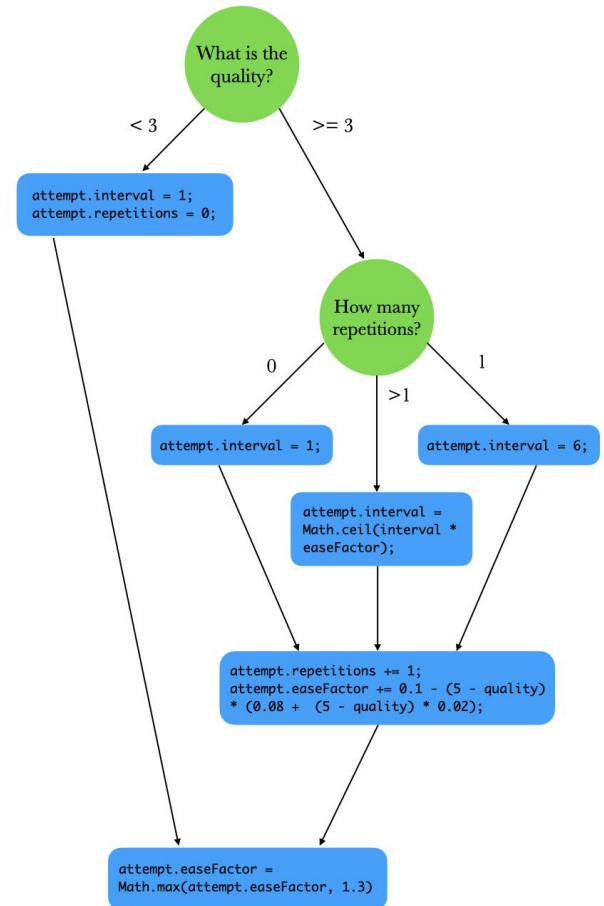
```
I/System.out: difficultyLevel: [5]
```

Backend



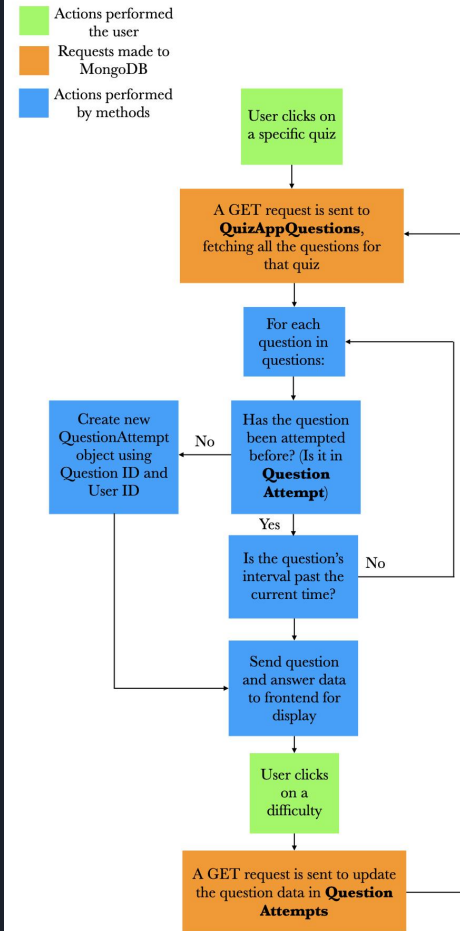
Spaced Repetition

- Spaced repetition is a powerful technique that will help you memorize information in much less time than it would take otherwise.
- **Quality (0-5):** how well was the user able to remember the answer?
- **easeFactor:** how easy was the question for the user based on past attempts.
- **Interval:** how long should the algorithm wait to show the question again?
- **Repetitions:** how many times has the user attempted this question.

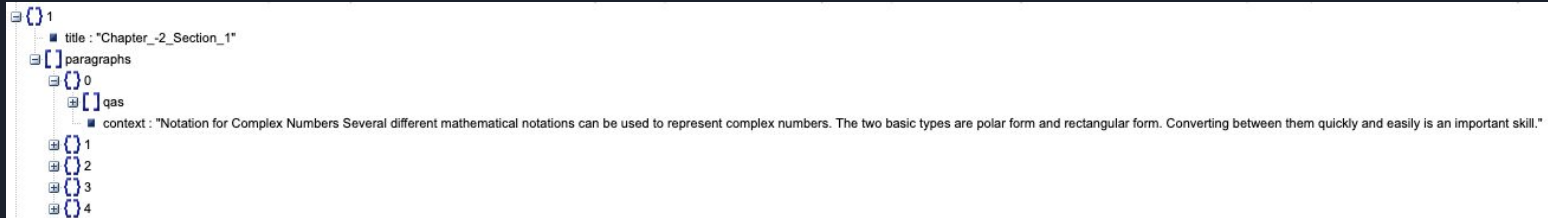


Spaced Repetition Algorithm Workflow

- We have two databases: **QuizAppQuestions** and **QuestionAttempt**.
- **QuizAppQuestions**: a database containing all the questions and answers.
- **QuestionAttempt**: a database containing all questions that any users have attempted, along with information for SM-2.
- The diagram on the right showcases the interactions between the user, the frontend, and the backend.



Data, Schema, and MongoDB Setup



We formatted the schema in such a way that it can be interpreted easily later.



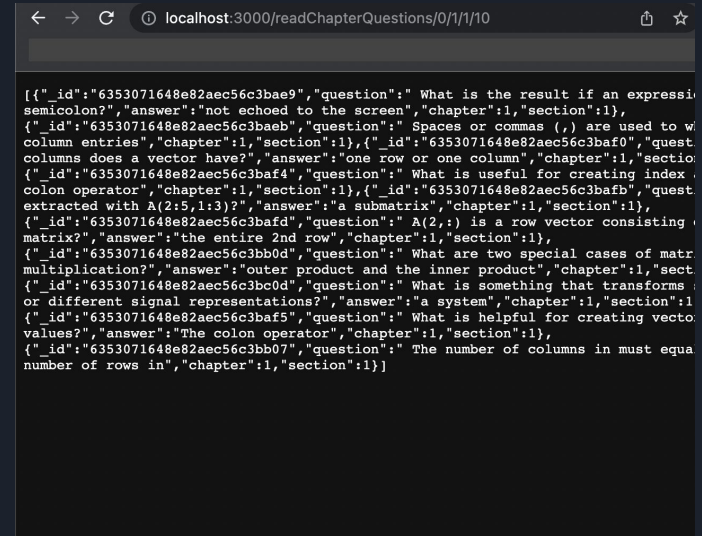
We created a python script that takes the formatted json file and uploads it to the MongoDB database

```
_id: ObjectId('6353071648e82aec56c3ba44')
question: " What are complex numbers plotted as vectors in the two-dimensional "c..."
answer: "Each is represented by a vector from the origin to the point with coor..."
chapter: 2
section: 1
```


API Endpoints

To allow interaction with mongoDB to implement spaced repetition algorithm we added 2 types of API endpoints for the front end to access:

- **Read Chapter Question:** Retrieves the specified set of questions filtered by Ch no., section and count.
- **Retrieve Question Attempt:** Retrieves question attempt matching the userID and the questionID.



```
localhost:3000/readChapterQuestions/0/1/1/10
[{"_id":"6353071648e82aec56c3bae9","question":" What is the result if an expression ends with a semicolon?","answer":"not echoed to the screen","chapter":1,"section":1}, {"_id":"6353071648e82aec56c3baeb","question":" Spaces or commas (,) are used to separate column entries","chapter":1,"section":1}, {"_id":"6353071648e82aec56c3baf0","question":" How many rows and columns does a vector have?","answer":"one row or one column","chapter":1,"section":1}, {"_id":"6353071648e82aec56c3baf4","question":" What is useful for creating index entries?","answer":"the colon operator","chapter":1,"section":1}, {"_id":"6353071648e82aec56c3bafb","question":" What is a submatrix?","answer":"A(2:5,1:3)","chapter":1,"section":1}, {"_id":"6353071648e82aec56c3bafd","question":" A(2,:) is a row vector consisting of the entire 2nd row","answer":"the entire 2nd row","chapter":1,"section":1}, {"_id":"6353071648e82aec56c3bb0d","question":" What are two special cases of matrix multiplication?","answer":"outer product and the inner product","chapter":1,"section":1}, {"_id":"6353071648e82aec56c3bc0d","question":" What is something that transforms a signal into a different signal representation?","answer":"a system","chapter":1,"section":1}, {"_id":"6353071648e82aec56c3baf5","question":" What is helpful for creating vector values?","answer":"The colon operator","chapter":1,"section":1}, {"_id":"6353071648e82aec56c3bb07","question":" The number of columns in a matrix must equal the number of rows in","answer":"the number of rows in","chapter":1,"section":1}]
```

Here the API retrieves 10 questions from Chapter 1 section 1

API Endpoints

Retrieves question attempt matching the userID and the questionID.



```
app.get(
  '/updateQuestionAttempt/:questionAttemptId/:quality',
  async (req, res, next) => {
    var quality = req.params.quality;
    const attempt = await db
      .collection('QuestionAttempt')
      .findOne({ _id: ObjectId(req.params.questionAttemptId) });
    console.log(attempt);
    if (quality >= 3) {
      var interval = 0;
      if (attempt.repetitions == 0) {
        interval = 1;
      } else if (attempt.repetitions == 1) {
        interval = 6;
      } else {
        interval = Math.ceil(attempt.interval * attempt.easeFactor);
      }
      attempt.interval = interval;
      attempt.repetitions += 1;
      attempt.easeFactor += 0.1 - (5 - quality) * (0.08 + (5 - quality) * 0.02);
    } else if (quality < 3) {
      attempt.repetitions = 0;
      attempt.interval = 1;
    }
    attempt.easeFactor = Math.max(attempt.easeFactor, 1.3);
    QuestionAttempt.updateOne({ _id: req.params.questionAttemptId }, attempt)
      .then(() => {
        res.send(attempt);
      })
      .catch((error) => {
        res.status(400).json({
          error: error,
        });
      });
  });
};
```

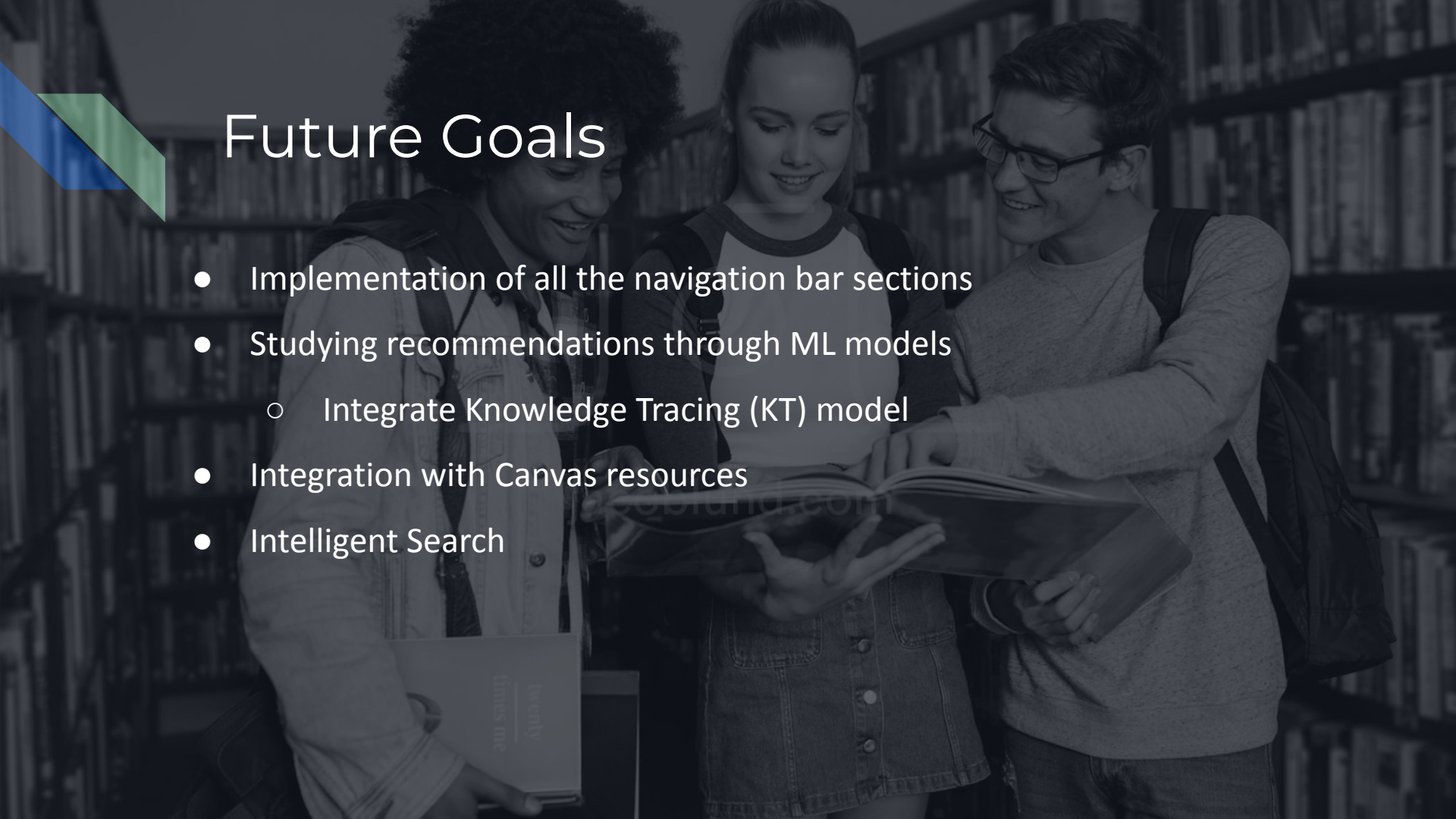


```
app.get('/findQuestionAttempt/:userId/:qId', (req, res) => {
  QuestionAttempt.find({
    userId: req.params.userId,
    questionId: req.params.qId,
  })
    .then((questionAttempt) => {
      if (questionAttempt.length == 0) {
        var newQuestionAttempt = {
          questionId: req.params.qId,
          userId: req.params.userId,
          repetitions: 0,
          easeFactor: 2.5,
          interval: 0,
        };
        console.log(newQuestionAttempt);
        db.collection('QuestionAttempt').insert(newQuestionAttempt);
        res.send(newQuestionAttempt);
      }
      res.send(questionAttempt);
    })
    .catch((err) => {
      res.status(500).send({
        message: 'some error occurred while retrieving messages',
        Error: err,
      });
    });
});
```

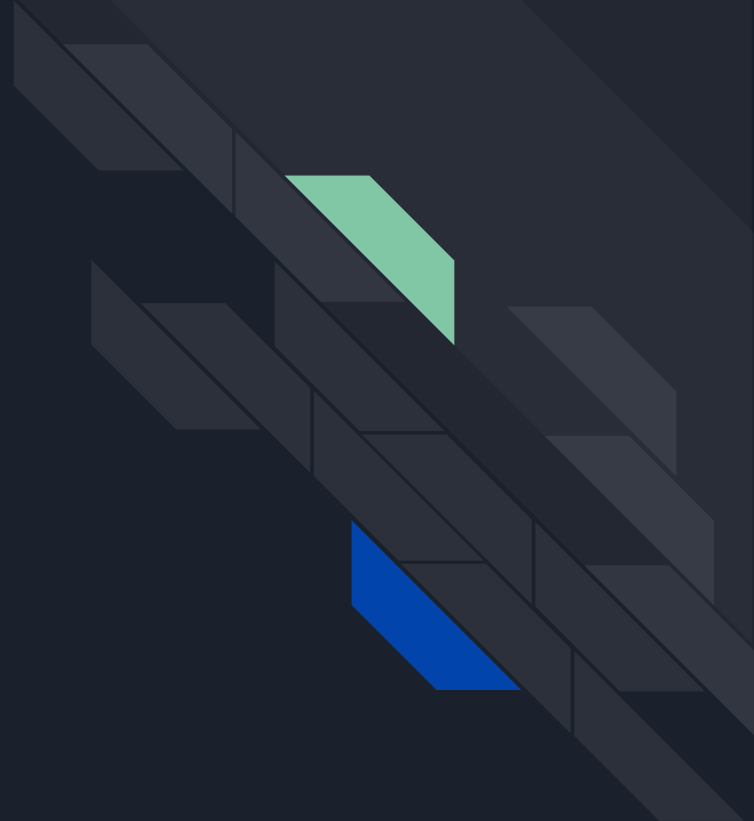
Implements the spaced repetition algorithm on a specific question attempt, given the **quality**.



Future Goals

- Implementation of all the navigation bar sections
 - Studying recommendations through ML models
 - Integrate Knowledge Tracing (KT) model
 - Integration with Canvas resources
 - Intelligent Search
- 

Thank you



Questions?

