Gramalytics

The Future of Content Strategy with AI and Data

CSCI 457: Senior Project 2, Fall 2024

Team Members:

Dhara Chandpara

Esther Bilenkin

Farzana Alam

Advisor: Dr. Wenjia Li

Introduction to Gramalytics

- Social media growth challenges on Instagram.
- Need for actionable recommendations to navigate content visibility.
- Analysis of high-performing posts and trends.
- Tailored insights for hashtags, past post performance, and audience preferences.

Existing Systems

Hootsuite:

A social media management platform for scheduling posts, tracking engagement. and managing various platform accounts. The program generally aims to help accounts, especially brand accounts, keep accounts and posts in an organized manner for easy interaction and overview.

Pros:

- Multi-platform support
- post scheduling which ensures posts are published at optimal times.
- detailed analytics to track performance and engagement metrics

Cons:

- Expensive (requires premium plans costing about $1200/yr)
- limited customization options for lower-tier plans
- Doesn’t have direct engagement with followers

Sprout Social:

Cross platform app/web browser which is designed to help grow user businesses across social networks, supporting X, Instagram, LinkedIn, Facebook, Pinterest, and TikTok. The system extrapolates data across all provided accounts to generate insights on performance and online reputation.

Pros:

1. Performs internal analyses on multiple linked accounts.
2. Inbox feature which helps manage messages and interactions across social media accounts.
3. Monitors brand mentions and trends.

Cons:

1. Expensive ($200 per month for standard plan and $300 per month for advanced)
2. Scheduling limits on lower-tier plans.
3. Limited feature integration compared to other competitors (shopify integrations)

Iconosquare:

AI-powered service available both through a web browser and as a phone app. It helps users manage any number of social media accounts across Instagram, X , Facebook, TikTok, and LinkedIn. It works by scraping information from the provided accounts and using AI to generate ideas based on past content.

Pros:

1. Competitor analysis feature.
2. Affordable.
3. Hashtag tracking and optimization tools.

Con:

1. Limited post engagement tools.
2. User interface can be overwhelming for users.
3. Some analytics features (story performance tracking) are only available on premium.

vidIQ:

An AI-powered service intended to help users with growing their YouTube channels with a wide array of features designed to help creators optimize their video content, improve channel performance, and grow their audience.

Pros:

1. Video performance tracking with real-time analytics.
2. Keyword research and optimization tools.
3. Provides various insights related to channel performance to improve growth.

Cons:

1. Limited collaboration tools.
2. Suggestions can be generic and less effective for niche channels.
3. No direct video editing or publication tools.

Proposed System

comprehensive solution for Instagram users to enhance content strategies using actionable insights
originally included:

- data collection by use of the Meta Graph API and Instagram API
- database management using postgreSQL
- AI Integration via OpenAI API
- mobile-friendly interface using React Native
- provides users with keyword optimization, post performance analysis, and niche-specific suggestions

Changes to the System

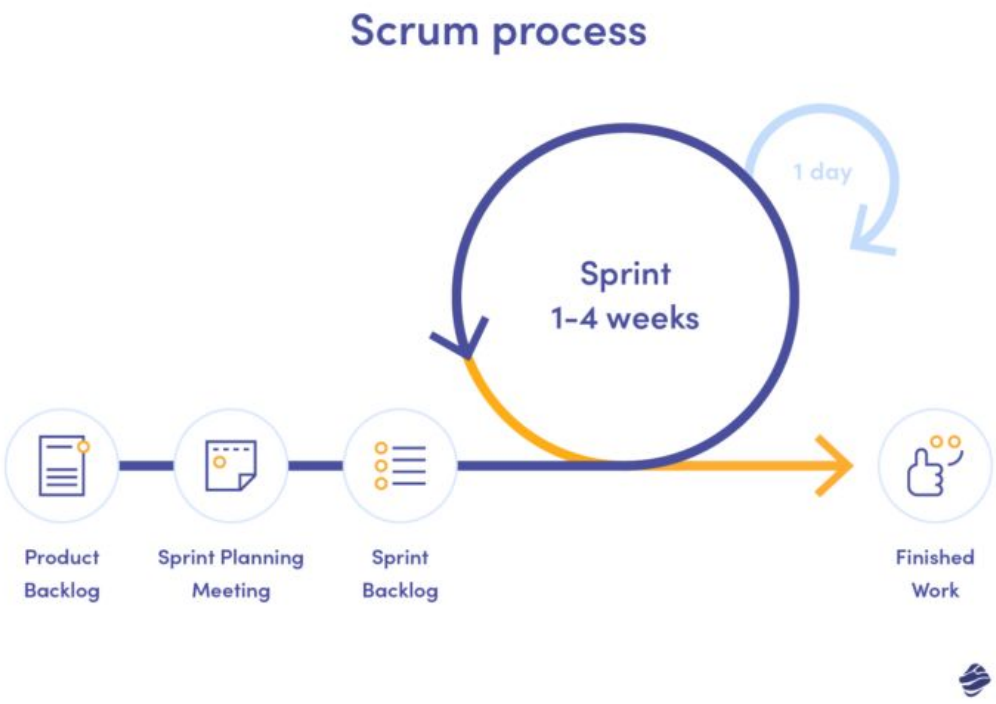
several changes were made to the system during development:

- Instagram API and webscraping were excluded from the final implementation due to time constraints & compliance concerns. mock data was used instead
- switched to Firestore database from PostgreSQL to simplify setup & leverage team's existing experience with Firebase
- Backend logic brought to frontend
- Feature scope reduced
- AI chatbot functionality simplified for deliverable

Software Engineering Model

SCRUM provides an agile framework for development, which allowed our team to work efficiently. The benefits for this method were:

1. Allows for evolving requirements and feedback with incremental releases
2. Use of backlogs and sprints which influenced frequent team collaboration.
3. Tracker for feature updates and implementation.



Functionality and Features

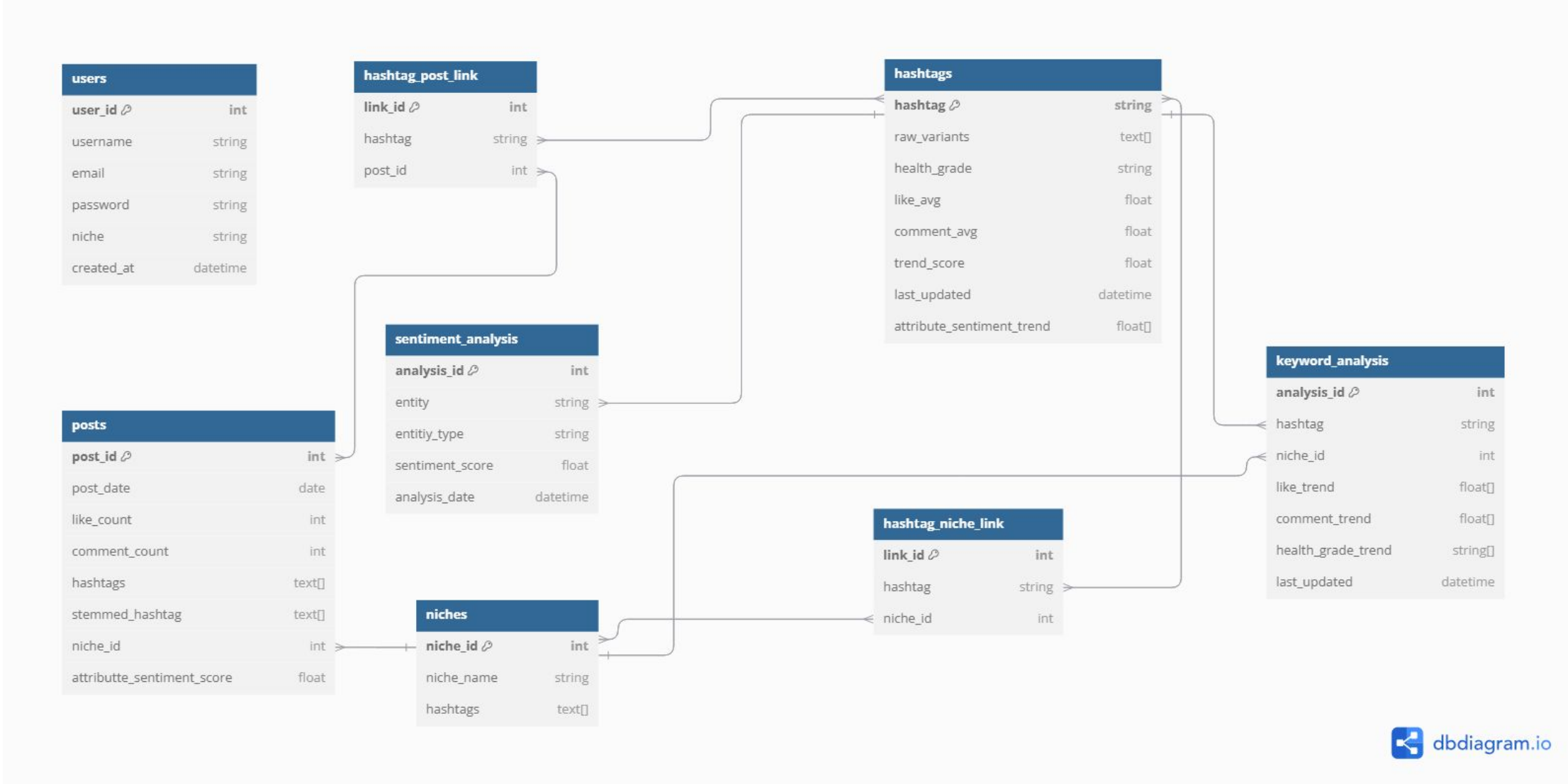
- **Interactive UI**
- **Core Tabs and Functionalities**
- **Cross-Platform Compatibility**
- **Ai Chatbot**:
- **Future-Ready Design**

Technologies and Tools

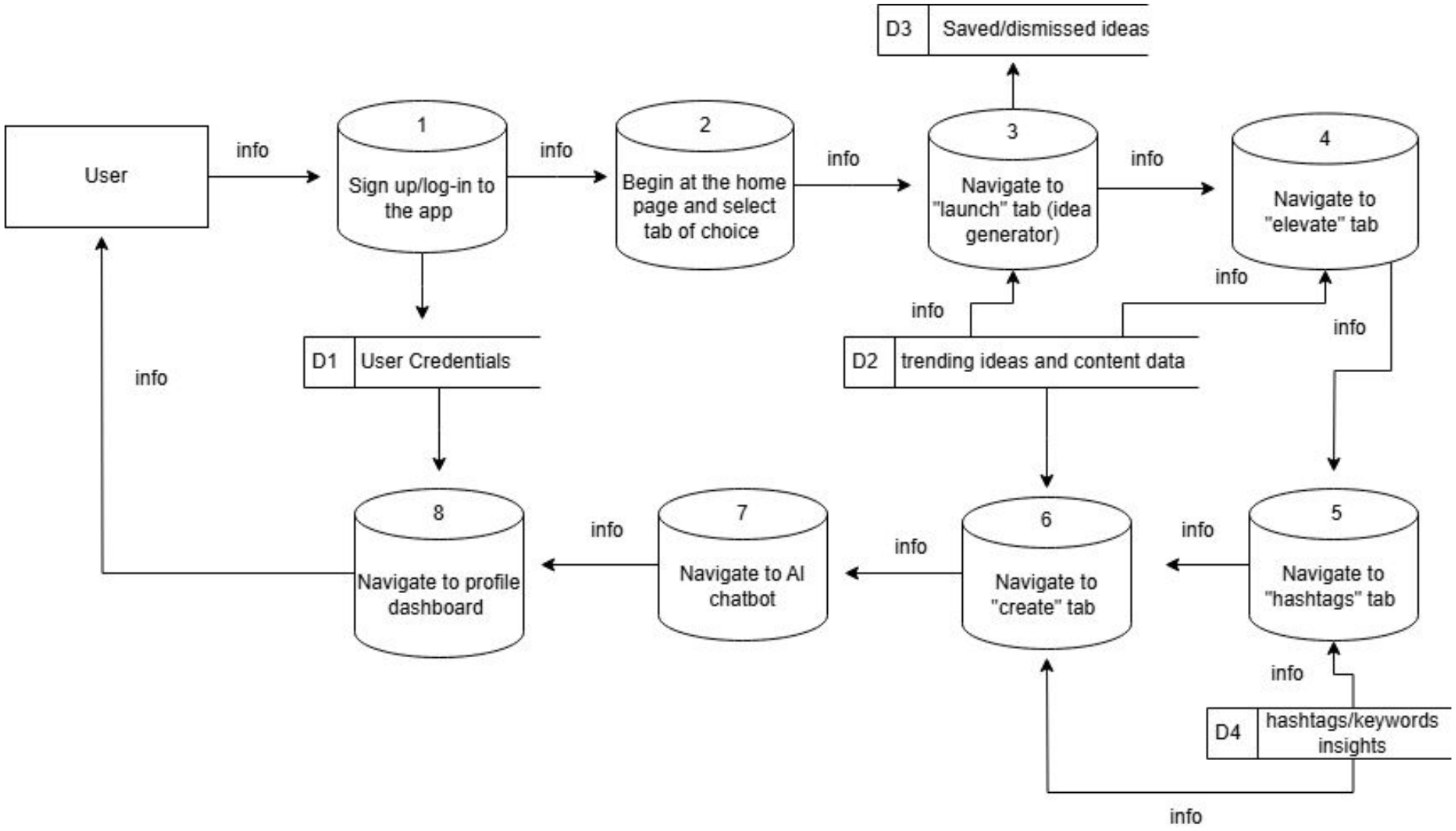
- **Backend**: React Native
- **Database**: Firebase Realtime Database
- **Frontend**: React Native, React, Vue, Expo
- **AI**: OpenAI for chatbot integration.

High Level Design

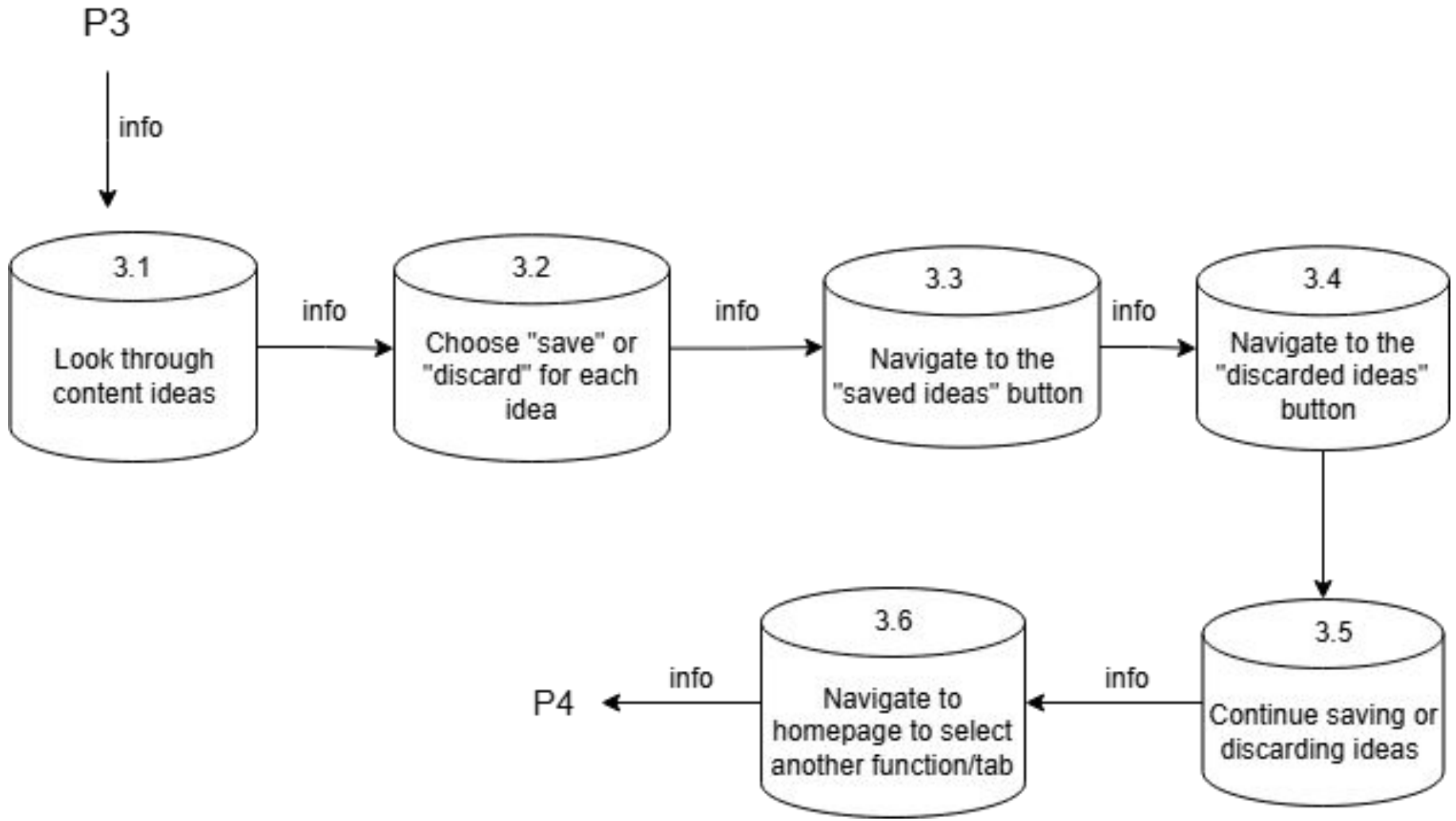
Entity-Relationship (ER) Diagram



Data Flow Diagram



Child Diagram



Implementation

Code Implementation Example

- Star Rating with Emojis

```
const getEmoji = (rating) => {
  switch (rating) {
    case 1:
      return "😞"; // Sad emoji for 1 star
    case 2:
      return "😐"; // Neutral emoji for 2 stars
    case 3:
      return "😊"; // Slightly happy emoji for 3 star
    case 4:
      return "😊"; // Happy emoji for 4 stars
    case 5:
      return "🤩"; // Excited emoji for 5 stars
    default:
      return "⭐";
  }
```

- Secure login

```
const passRegex = /^(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{6,20}$/;
if (!passRegex.test(password)) {
  setError(
    "Password must be 6-20 characters with at least one digit, one lowercase and one u
  );
  return;
}
```

UI and UX Design

launch- include functionalities such as idea generator, new feature

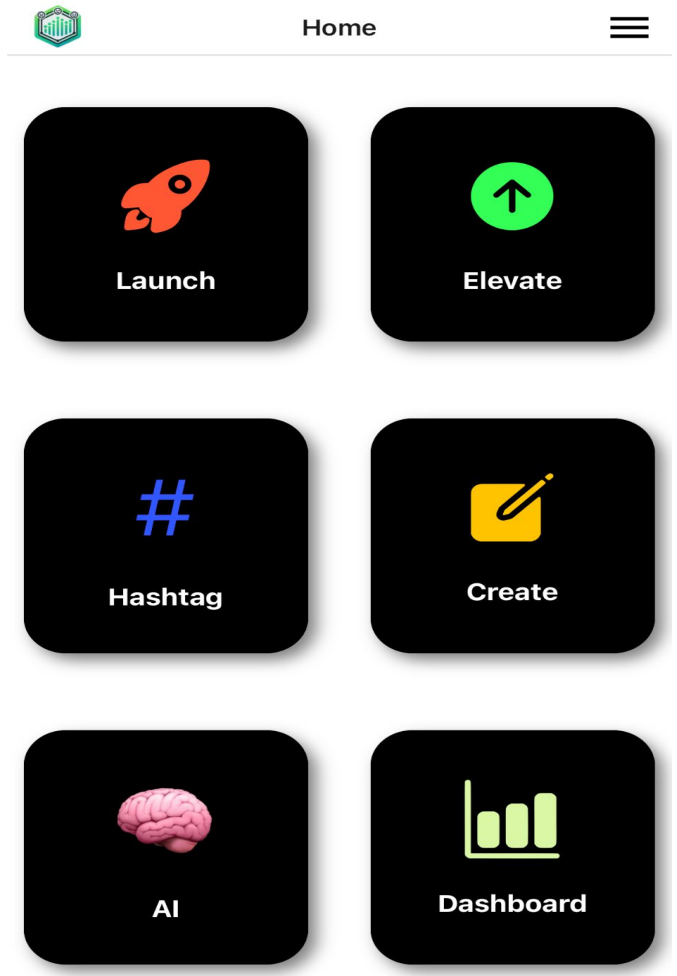
Elevate- insights for improving content

Hashtag- assist users in finding relevant hashtags

Create- A tool to brainstorm idea

AI- direct access to chatbot for assistance

Dashboard- summarizes user's profile



AI Query Handling

queryOpenAI

- process user inputs to the AI chatbot
- returns responses from chatbot
- powered by OpenAI API
- delivers human-readable outputs aligning with goals of the app

```
const handleQuerySubmit = async () => {
  if (query.trim() === "") {
    Alert.alert("Error", "Please enter a query");
    return;
setIsLoading(true);
try {
  // Use main_handler to query OpenAI
  const result = await main_handler("queryOpenAI", query);
  setAiResponse(result);
} catch (error) {
  console.error("Error querying OpenAI:", error);
  Alert.alert("Error", "Failed to get AI response");
} finally {
  setIsLoading(false);
  setQuery("");
```

Backend Design

main_handler

- component of the application responsible for coordinating backend logic
- centralizes kery operations of the app
- lends to a modular, easily extensible design
- design helps isolate communication issues between frontend and backend

```
//main_handler - assignment function which determins which logic is being called
const main_handler = async (action, data = null) => {
  console.log("Action received: ${action}", data);

  switch (action) {
    case "generate": // generate new ideas in Launch
      return await generateIdeas();
    case "save": // save idea from Launch
      return await saveIdea(data);
    case "discard": // discard idea from Launch
      return await discardIdea(data);
    case "getSaved": // Get saved ideas
      return await getSavedIdeas();
    case "getTopicRelatedHashtags": // Get related hashtags for a topic
      return await getTopicRelatedHashtags(data);
    case "getDiscarded": // Get discarded ideas
      return await getDiscardedIdeas();
    case "queryHashtag": // query hashtag from input in Hashtag
      return await queryHashtag(data);
    case "getTrendingHashtags": // Get trending hashtags
      return await getTrendingHashtags();
    case "queryOpenAI": // query OpenAI from query in AI tab
      return await queryOpenAI(data);
    case "generateTrends": // periodically fill out the content of Elevate tab
      return await generateTrends();
    case "semanticSearch": // query from Elevate tab
      return await semanticSearch(data);

    default:
      console.warn("Unknown action: ${action}");
      return { message: "Uknown action" };
  }
```

```
    const response1 = await main_handler('queryHashtag', 'art'); // call ha
    const response2 = await main_handler('queryHashtag', 'technology');
    const response3 = await main_handler('queryHashtag', 'fitness');
    const response4 = await main_handler('queryHashtag', 'travel');
    const response5 = await main_handler('queryHashtag', 'music');
    console.log('Response:', response1);
    console.log('Response:', response2);
    console.log('Response:', response3);
    console.log('Response:', response4);
    console.log('Response:', response5);
};

const testOpenAIQuery = async() => {
    console.log('Testing Query AI...');
    const response = await main_handler('queryOpenAI', 'input'); // call op
    console.log('Response:', response);
}

const testGenerateTrends = async() => {
    console.log('Testing Generate Trends...');
    const response = await main_handler('generateTrends', 'input'); // call
    console.log('Respond:', response);
}

const testSemanticSearch = async() => {
    console.log('Testing Semantic Search...');
    const response = await main_handler('semanticSearch', 'input'); // call
    console.log('Respond:', response);
}
```

Backend Design Cont’

testBackend.js

- simple script for testing individual components of backendLogic.js
- simulates various inputs and outputs
- desired information is firstly, functionality and secondly,, output format
- helped identify and resolve issues early

Security Measures and Testing

Security Measures:

1. Firebase Authentication:
    a. Used for secure login and signup.
2. Real-Time Database Security:
    a. Firebase rules are set up to restrict unauthorized access to sensitive data.
3. Token-Based Access:
    a. Authentication tokens validate user sessions, reducing the risk of attacks.

Testing:

1. Component Testing: The testBackend.js script was employed to verify backend functionality. It focused on:
    a. Data flow between tabs and modules.
    b. Reading from mockup data tables to generate results
    c. Explicit component testing
2. End-to-End Testing: Conducted on an emulator, verifying:
    a. Navigation and feature accessibility.
    b. Data from backend logic was displayed properly.
    c. OpenAI API chatbot functions properly.

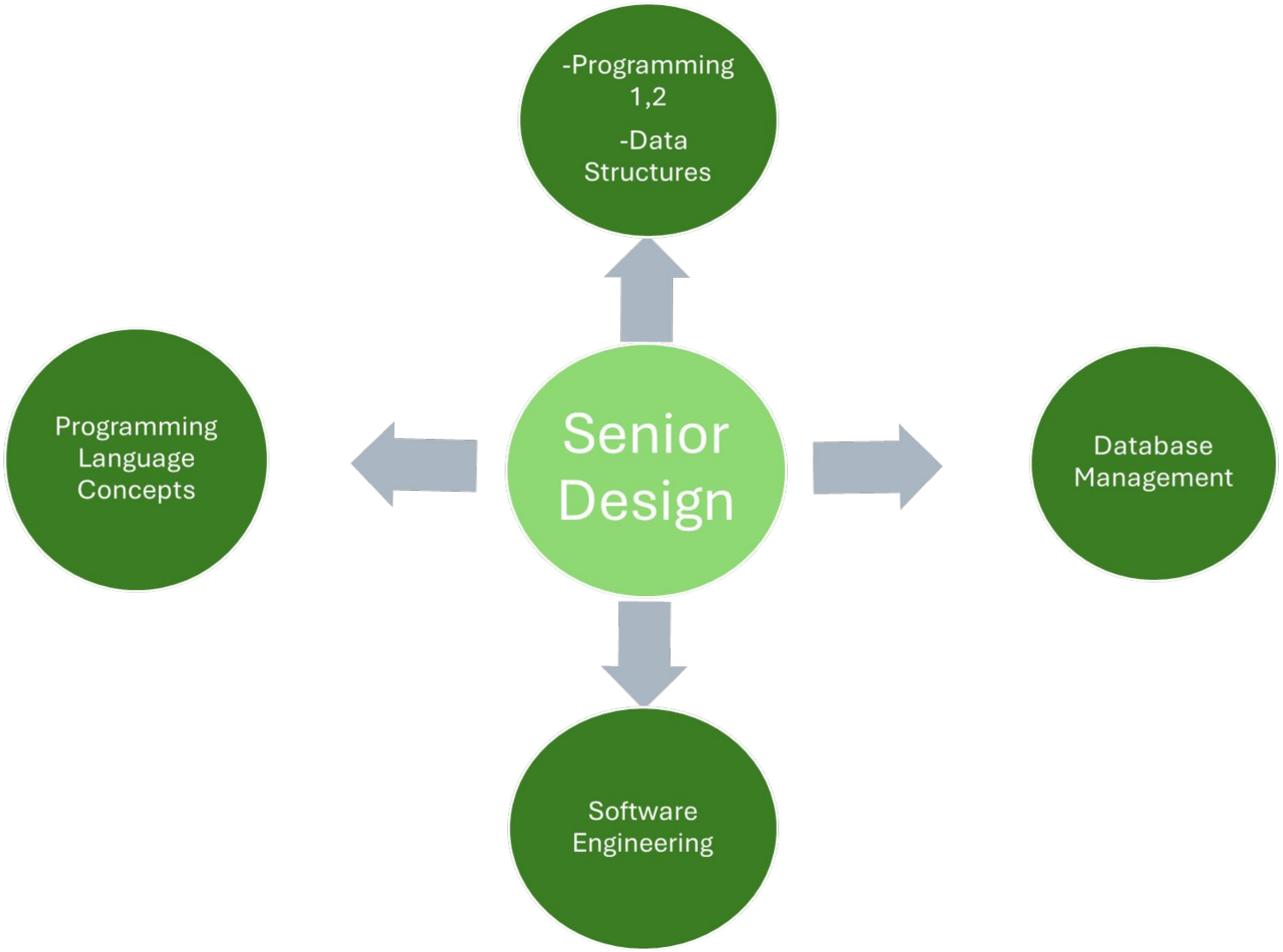
Limitations and Challenges

1. Reduction of team size at the beginning of the development page.
2. Technical challenges regarding backend and frontend implementation.
    a. Difficulties running tools and software cohesively.
    b. Most of our challenges regarded tools/technology, as most of them weren't used as intended.
3. Database Design
    a. The database significantly changed as the team began working though the algorithms and methodology for each feature. The scope increased from four to eight tables to compensate for all the data required for analysis.

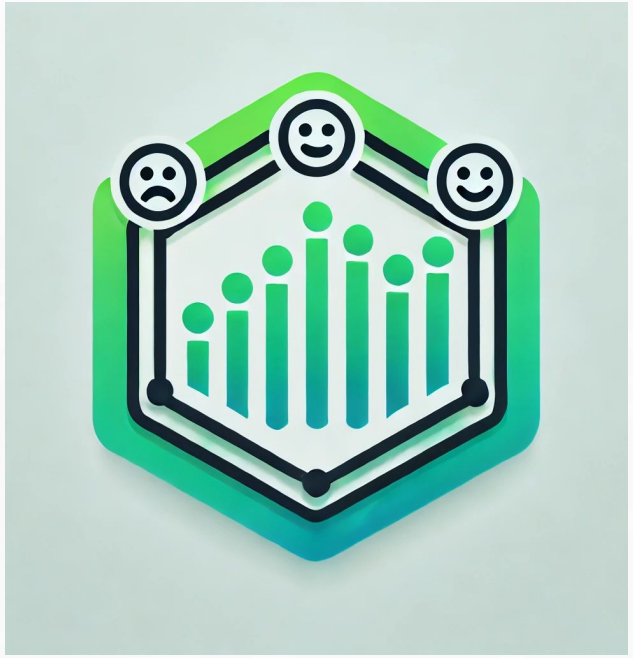
Future Enhancements

1. **Custom Dashboard**
    a. Allow users to build and customize their own analytics dashboard, tailoring content to their specific needs or performance goals.
2. **Implement real time data and analytics**
    a. The implementation of a web scraper, data preprocessing pipeline, and data mining pipeline would enable up-to-date information, allowing the app to provide more precise and relevant insights to users.
3. **Enhancing the AI chatbot**
    a. Allow the AI chatbot to a dedicated in-app assistant rather than just being a general AI query system.
4. **Integrating an image display feature**
    a. Allow the users to post ideas and trending content within the app which allows them to utilize the app more and also provide the backend logic with more data to work with.

What made this project successful?



Demo video



Team Contributions:

Dhara: Frontend Design

Esther: Backend Design

Farzana: Database Implementation

Although each member was responsible for their primary section, the team worked collaboratively and frequently helped each other out.

Any Questions?