Carb Capture

A Glucose & Carbohydrate Integrated Tracking Software

November 24, 2024

Members:
Cornelius Pavlic
Nassim Marhari
Jiaxing Lu
Cole Kaminski

Table of Contents

Introduction.. **4**

Problem Statement

Solution

Users

Environment

Requirements Elicitation.. **7**

System Functionality

Data

Physical Environment

Interfaces

End-Users

Performance

Usability and Human Factors

Security

Reliability and Availability

Maintainability

System Architecture.. **13**

Software Components Diagram

Render Functions

Dexcom API Functions

OpenAI API Functions

Data Storage and Processing Functions

User Types

System Capacity and Performance

Context Diagram

Key Interactions

User Benefits and Functional Role

UML Diagram

Component Interactions

Functional Workflow

Gantt Chart

User Interfaces... **20**

Login Screen

Home Page

Set Target Glucose Screen
Settings Page

Usage Cases.. 23
Dexcom Login
Updating Carb-Insulin Ratio
Receiving Insulin Estimate
Additional Use Cases

Conclusion.. 26
Possible Updates and Expansion
Summary

GitHub Repository
https://github.com/nmarhari/sdp/tree/main

Introduction

Problem Statement

The challenge of managing diabetes effectively revolves around achieving a balance between blood sugar levels, insulin dosage, and food intake. This requires a diabetic person to continuously monitor their blood glucose levels, estimate carbohydrate content in meals, and calculate appropriate insulin dosages. However, this process can be overwhelming and prone to errors, especially when dining out or consuming non-homemade meals with unknown nutritional information.

People with diabetes need to carefully manage their insulin dosage based on their current blood sugar levels and the carbohydrate content of their meals. Often when eating meals, especially ones that are not homemade, diabetics must guess the amount of carbs that are in the food they are eating. Along with this, despite the availability of continuous glucose monitoring (CGM) devices like Dexcom, there is a need for an application that integrates this real-time data to assist users in determining the precise insulin dosage needed before meals.

Currently, Continuous Glucose Monitoring (CGM) devices like Dexcom provide real-time blood glucose readings, offering users valuable insight into their current glucose levels. However, these devices do not inherently provide detailed guidance on how to use this data to make accurate, context-specific insulin dosage decisions.

Solution

Carb Capture is a diabetes management app that streamlines insulin dosing by combining meal recognition through photos with real-time glucose data. Users can upload pictures of their meals to get accurate carbohydrate estimates, which are then integrated with data from continuous glucose monitoring (CGM) devices to calculate precise insulin dosages. The app allows users to set their target glucose range, ensuring personalized recommendations that align with their health goals. Additionally, a meal tracker records food intake, insulin dosages, and glucose levels, providing insights into patterns that can help users and healthcare providers make data-driven adjustments to treatment plans.

By reducing the guesswork involved in carbohydrate estimation and insulin dosing, Carb Capture alleviates the cognitive burden on users, leading to more accurate management of blood glucose levels. This holistic approach to meal tracking and insulin calculations not only simplifies daily diabetes care but also has the potential to improve glycemic control and overall health outcomes.

Users

The primary users of Carb Capture are individuals with diabetes, particularly those who require insulin management. This includes people with Type 1 diabetes, as well as some individuals with Type 2 diabetes who rely on insulin therapy. In the United States alone, approximately 2 million people live with Type 1 diabetes. Using a conservative estimate, if 10% of this population adopts the app, the initial user base could reach around 200,000 individuals. However, the global potential is much larger, with millions of diabetics worldwide who could benefit from a streamlined, cost-effective, and integrated digital solution. Carb Capture is especially suited for those already using Continuous Glucose Monitoring (CGM) devices like Dexcom or those seeking advanced tools to simplify their health management.

Carb Capture is designed to be accessible to a wide range of users, regardless of educational background or technical expertise. Users are expected to have basic smartphone skills, such as downloading and navigating an app, and the ability to read simple on-screen instructions. The app is inclusive and caters to all age groups, from tech-savvy teenagers managing their diabetes to older adults adapting to new tools for their health. Its intuitive interface ensures that even individuals with minimal technical proficiency can easily upload photos of their meals, log glucose trends, and optionally link their CGM accounts for seamless data integration. This broad usability makes Carb Capture a valuable tool for anyone seeking to reduce the complexities of diabetes management.

Environment

Hardware: Carb Capture is designed to run on mobile devices, specifically smartphones and tablets. Users will require a device capable of running either the Android or iOS operating systems to access the app.
Operating System: The application is compatible with Android and iOS, ensuring broad accessibility for users regardless of their preferred platform.
User Environment: Carb Capture is intended for casual, day-to-day use. Users can interact with the app wherever they are, typically during meal times, whether at home, at work, or dining out. Its functionality is optimized for convenience, allowing users to quickly and easily log meals and monitor glucose levels in real-time.

Standards: The app aims to deliver reliable health estimates in a user-friendly and convenient manner. It prioritizes accuracy and accessibility to provide actionable insights that assist users in managing their diabetes effectively.
Assumptions/Risks: Several assumptions and risks are associated with Carb Capture's use:

- It assumes that the CGM devices (e.g., Dexcom) provide accurate blood glucose readings.
- The app assumes users input their carb-to-insulin ratios correctly for accurate insulin dosing recommendations.
- There is a risk of providing slightly incorrect or imprecise carbohydrate estimates due to limitations in image recognition technology or user input errors.
- Connectivity and service availability are assumed to be constant; interruptions could affect real-time data integration.
- A significant risk lies in providing incorrect medical advice, emphasizing the need for users to consult healthcare professionals for critical decisions and view the app as a supplemental tool.

Requirements Elicitation

System Functionality

1. **What will the system do?**
   The system will use API's and user entered information, such as carb-to-insulin ratios and an image of a food item, to give the user an estimation of how much insulin they should take for that meal.
2. **When will the system do it?**
   The system will be able to do it whenever the user provides a photo of a food item, whenever they are eating.
3. **Are there several modes of operation?**
   No, there is just one mode of operation, one where the user will enter information. They must enter a carb-to-insulin ratio, then, it is optional whether they want to submit a photo whenever they please of the food item they would like to get an estimation for.
4. **What kinds of computations or data transformations must be performed?**
   The system will have to communicate with Dexcom's API and OpenAI's API in order to get blood glucose levels and the estimated carbohydrates, respectively. The data received from these API's may need to be transformed from text to integers for computation. It will then need to multiply the amount of carbohydrates in the food item by the Dexcom blood glucose level and the carb-to-insulin ratio that the user provided to give an estimate.
5. **What are the appropriate reactions to possible stimuli?**
   There should be two main stimuli in the application. For the first one, it would be entering in a carb-to-insulin ratio. The application should accept this number or decimal and store it for later, letting the user know that they successfully updated their information. The second one would be taking a photo of food to send to get an estimate. The application should take this photo, use OpenAI's API to decipher the photo and get an estimate from the food, do the computation to convert it to an insulin amount, then send back the estimated insulin amount. The application should receive blood glucose levels from Dexcom, but should not need to manually enter these numbers if a user has a compatible sensor.

Data

1. **For both input and output, what should be the format of the data?**
   For input, the data should be in the format of an integer or decimal for the carb-to-insulin ratio, a and a photo (jpg, png, etc.) of the food item.
2. **Must any data be retained for any period of time?**
   Data may be retained for the user to keep a log of their meals and consult their doctor for updated carb-to-insulin ratios. This is optional.
3. **Design Constraints**
   The design should be user friendly, able to be used by a variety of ages from teenagers to middle ages. It should be intelligent enough for doctors to derive enough information to provide educated advice.

Physical Environment

1. **Where is the equipment to be located?**
   The app will run on user smartphones or mobile devices.
2. **Is there one or several locations?**
   The app can be used anywhere the user carries their phone.
3. **Are there constraints on size of the system (Handheld/Server/PC etc)?**
   Yes, the system must be optimized for mobile devices.
4. **Are there any constraints on programming language, OS because of existing software components?**
   The app should be developed using cross-platform frameworks like Flutter or React Native to support both Android and iOS.

Interfaces

1. **Is input coming from one or more other systems (“upstream”)?**
   Yes, input is coming mainly from Dexcom’s API and OpenAI’s API. These systems are assumed to be working at all times for the use of the application.
2. **Is output going to one or more other systems (“downstream”)?**
   No, there are no systems outside our application that should be receiving the data from our application, as it deals with health information.
3. **What is the protocol for the upstream and downstream systems?**

The upstream data will most likely be received through a secure channel like HTTPS, using GET, POST, or other requests relevant to this protocol.

End-Users

1. **Who will use the system?**
   The primary users are people with diabetes, particularly those requiring insulin management (Type 1 and some Type 2 diabetes patients).
2. **Will there be several types of users?**
   Yes, there will be users from all demographics, health backgrounds, and intentions.
3. **What is the skill level of each user?**
   They must be capable of reading text and have basic smartphone knowledge to open the application. However, the app should also be accessible to those with minimal technical expertise.

Performance

1. **Are there constraints on execution speed, response time or throughput?**
   There are no constraints on execution speed. The application's most resource intensive and time constraining feature will be the API pulling data from OpenAI. This process is relatively quick, and the user should not be in a life-or-death situation requiring immediate output from the application.
2. **How much data will flow through the system.**
   For the first version of our application, the data being input into the system is an image, or multiple images depending on the user, and DexCom text data. Further versions may allow for a user log-in text inputs, data reading from a user's connected DexCom device, and healthcare provider interactions (which may involve images, videos, or text). The return data will be through our connected API, pulling text from OpenAI.
3. **How often will data be received or sent?**
   Data will be collected and received at the user's discretion, meaning every time they upload an image. Data will be sent via an image upload and the corresponding output will be received shortly after. DexCom data will either be uploaded to the app via background pulls, upon image processing execution, or

upon app startup. There will likely be a limiting factor, preventing the user from uploading a picture before the previous input returns data.

Usability and Human Factors

1. **What kind of training will be required for each type of user?**
   The application will not require training to get an understanding of the functions. An optional tutorial or introduction can be provided by the healthcare provider.
2. **How easy should it be for a user to understand and use the system?**
   It should be fairly easy for a user to understand and use the system. There will be very minimal interaction needed to fulfill the purpose of using the application.

Security

1. **Must access to the system or information be controlled?**
   Access to the system does not need to be controlled. The application will take a photo from the user and output how many carbs are in the food. There is no private health information on file or other sensitive information. If an additional logging system is added to the app, we plan to store that data on the user's phone.
2. **Should each user's data be isolated from the data of other users?**
   The user's data will always remain independent of other users' data.
3. **Should user programs be isolated from other programs and from the OS?**
   No, the user program does not need to be isolated from other programs and the OS. There is no sensitive information flowing into or out of the application.
4. **Address Cyber security issues**
   With our current plans, we expect no cybersecurity issues as a threat to the user. This will change if we decide to implement a log-in feature, meal or health record loggings, or healthcare provider information.

Readability and Availability

1. **Must the system detect and isolate faults?**

No, due to the nature of our application using API's to retrieve information from OpenAI, a fault detection system is not necessary. We can implement fault detection code into the API call, like outputting "Please try another image, the provided picture is not recognized as food" if the user uploads an inevaluable image. The results provided from the application are approximate and have minimal chance to cause harm.

2. **What is the prescribed Mean Time between Failures?**
   Our estimated mean time between failures is estimated to be no more than 24 hours. The system is using Dexcom and OpenAI's APIs in order to ensure maximum uptime and reliability.
3. **Is there a maximum time allowed for restarting the system after a failure?**
   The maximum time allowed for restarting the system after a failure should be around 30 minutes. This is because when making a request to the Dexcom and OpenAI API's there is usually a message that prompts you to try again in a few minutes should there be any errors. To ensure the failure is resolved, there should be a maximum of 30 minutes allowed for restarting the system.
4. **How often will the system be backed up?**
   Our initial version of the system will not require a back up because no data needs to be stored. If data needs to be stored in the future, for example, interactions between a user and a healthcare provider, a daily system backup should be enough to minimize data loss.
5. **Must backup copies be stored at a different location?**
   Backup copies can be stored in the cloud to prevent user error backing up data to their device and limit resource demand. Storing backups to the cloud provides added security, scalability, and accessibility.

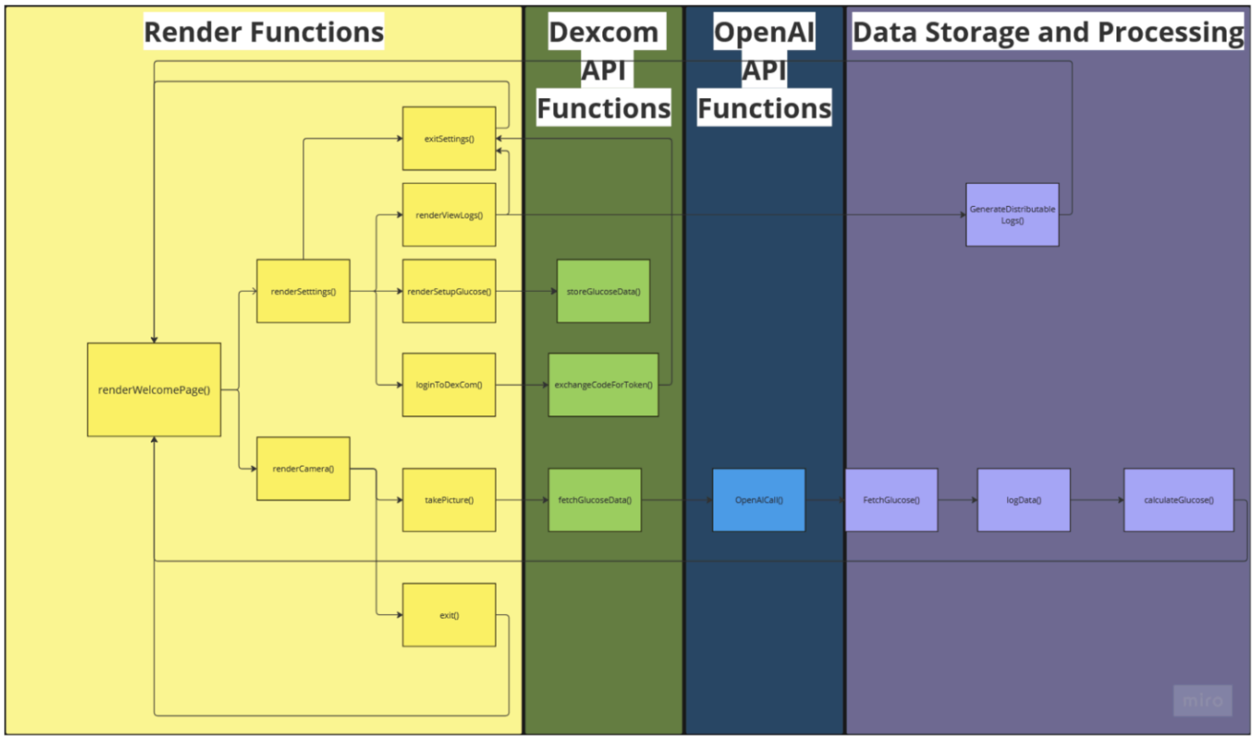
Maintainability

1. **When and in what ways might the system be changed in the future?**
   Some possibilities for expansion in the future include the implementation of a user log-in, Dexcom connectivity, healthcare provider access, and a meal or health logging system. There is not a current time to expect any changes. Changes will be made according to user demand and resource availability.
2. **How easy should it be to add features to the system?**
   It should be relatively easy to add features to our system. We would implement different pages or functionality within the app, making a seamless transition between version updates. As well as our first product will cover the basic goals of our idea, it will be easy to expand the complexity.

3. **How easy should it be to port (or migrate) the system from one platform to another?**
   It will be easy to port the system across platforms. The team is going to use software that requires little-to-none effort transitioning the system across other platforms.

Systems Architecture

Software Components Diagram



Render Functions

Each of these render functions will be React components. They handle UI rendering tasks.

Components:

renderWelcomePage()

Utility: The main page that the user interacts with when launching the application. It provides an entry point to other features. As well as displaying the user's data after it is returned.

renderSettings() and renderCamera() are subcomponents of renderWelcomePage()

renderSettings()

Utility: Renders the settings page where users can configure options such as Dexcom integration or other preferences.

Sub-components:

exitSettings(): Used to exit the settings page and return to the previous page.

renderViewLogs(): Displays logs, of Meals and glucose data

renderSetupGlucose(): Allows the user to set up glucose monitoring by entering relevant data.
loginToDexCom()

Utility: Logs the user into Dexcom services, enabling them to access glucose data.

renderCamera()

Utility: Renders a camera interface, allowing users to take a picture of their food.

Sub-components:

takePicture(): Allows users to capture a picture within the camera interface.

exit(): Exits the camera interface and returns to the previous screen.

Dexcom API Functions

These functions are responsible for communicating with the Dexcom API to manage glucose data.

Components:

storeGlucoseData()

Utility: Stores glucose data received from the Dexcom API into the database.

fetchGlucoseData()

Utility: Fetches glucose data from the Dexcom API for use in the app.

exchangeCodeForToken()

Utility: Exchanges authentication codes for an access token to interact with Dexcom's API.

OpenAI API Functions

These functions are responsible for integrating with OpenAI to process data.

Components:

OpenAICall()

Utility: Sends a request to OpenAI's API to process for amount of carbs in the picture

Data Storage and Processing Functions

These functions handle data storage and backend processing.

Components:

GenerateDistributableLogs()
Utility: Creates a CSV from the local database of glucose and insulin.
FetchGlucose()
Utility: Fetches glucose data from the database for processing and analysis.
logData()
Utility: Logs data into the database, likely in structured form, for future retrieval.
calculateGlucose()
Utility: Calculates glucose levels based on data fetched from the Dexcom API or stored data, potentially providing insights or predictions.

User Types

1. **Admin User:**
   Administers application settings and configurations, defines access groups and user permissions, handles system backups, analyzes application performance metrics and user engagement, configures and monitors API integrations, and oversees all aspects of the application and infrastructure.
2. **End User (Subscriber):**
   Uploads food images to receive carbohydrate and glucose-insulin ratio feedback, views historical data and recommendations, and authenticates Dexcom account.

System Capacity and Performance

Server Capacity

a. Computational cost is handled by Dexcom and OpenAI servers for them to fulfill requests, independent from our Data Processing and Storage.
b. The number of external API requests (Dexcom and OpenAI) may be limited based on service agreements or excessive token costs.

Storage Capacity
The database will have a finite amount of storage for user data, image uploads, and historical records. Once the limit is reached, data management strategies (like archiving or purging old data) can be needed.
Query Performance
As the amount of data grows, query performance may degrade, especially if the database isn't optimized. This can lead to slower response times for users retrieving historical data logs or recommendations.
Upload/Download Speeds

Users' internet speeds can affect how quickly they can upload images and receive feedback. Slow connections may lead to timeouts or a poor user experience.

Image Size

Larger image files may take longer to upload and process, impacting user experience. There may also be limits on file sizes imposed by the API.

Processing Time

The time taken by the OpenAI API to analyze images and return results can introduce delays, especially during peak usage times.

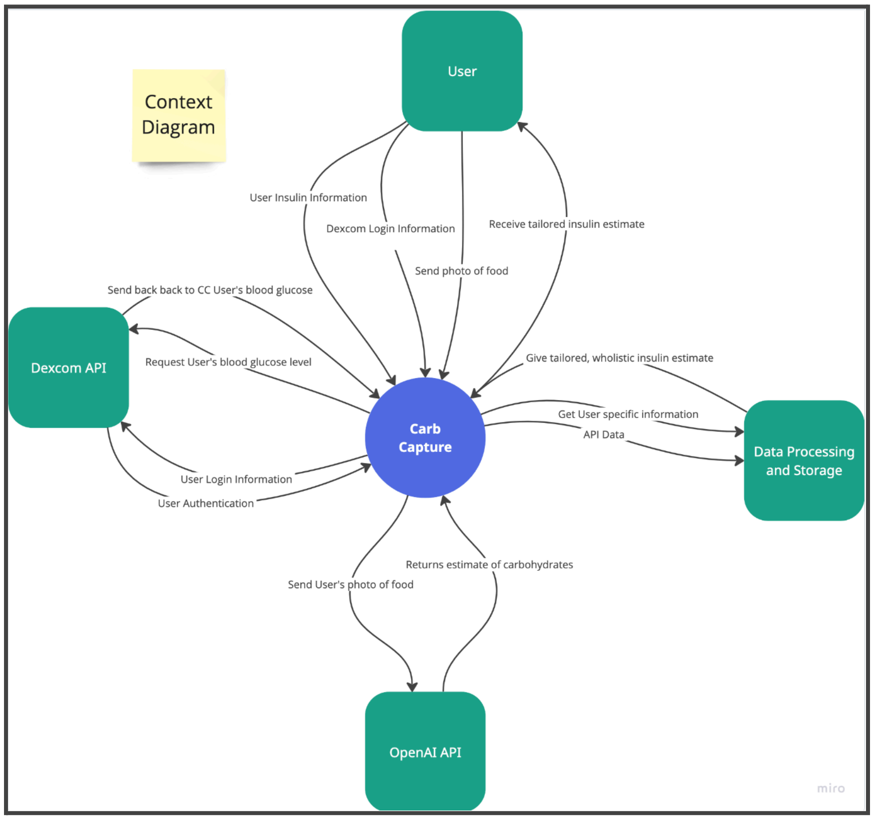
Device Performance

The app's performance can vary based on the user's device capabilities (CPU, RAM). Older or less powerful devices may struggle with processing or displaying data efficiently. Our application is not resource intensive and should work efficiently on modern devices.

Storage Space

Users may have limited storage space on their devices for app data and images, potentially affecting the app's functionality, like meal and insulin logging.

Context Diagram


Context Diagram
User
User Insulin Information
Dexcom Login Information
Receive tailored insulin estimate
Send photo of food
Send back back to CC User's blood glucose
Dexcom API
Request User's blood glucose level
Give tailored, wholistic insulin estimate
Carb Capture
Get User specific information
API Data
Data Processing and Storage
User Login Information
User Authentication
Returns estimate of carbohydrates
Send User's photo of food
OpenAI API
miro

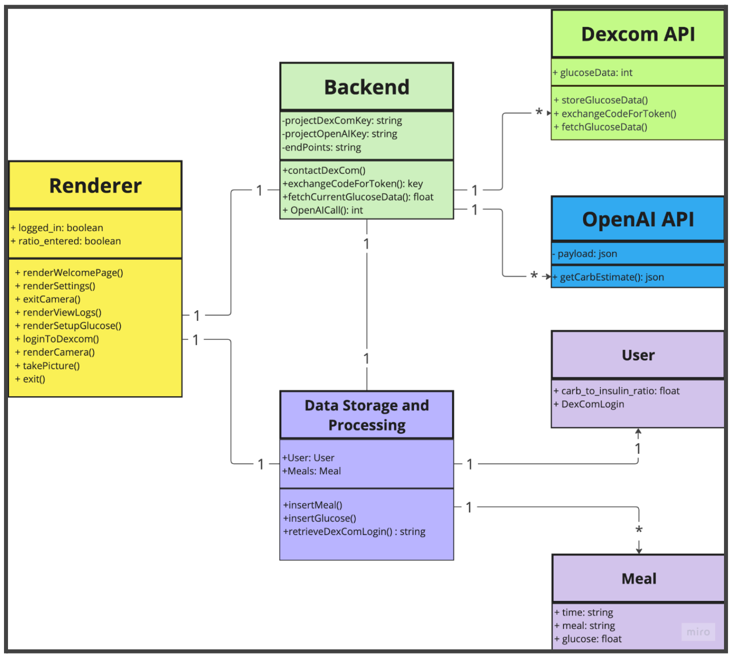
Key Interactions

The Dexcom API manages user authentication and retrieves blood glucose levels, sending the data back to the Carb Capture Application for further analysis. Similarly, the OpenAI API interprets user-submitted food photos to estimate carbohydrate content, which is then processed by the application. Internally, the Carb Capture system relies on data processing and storage components to compute insulin recommendations and retain relevant information, ensuring seamless integration between user inputs and external data sources.

User Benefits and Functional Role

The Carb Capture Application enhances the user experience by combining advanced technologies with a functional, user-friendly interface. It simplifies complex diabetes management tasks, such as calculating carbohydrate intake and determining insulin requirements, into a streamlined process. By leveraging data from external APIs and internal computations, the application provides accurate, personalized, and efficient recommendations, improving the quality of life for users managing diabetes.

UML Diagram



The UML diagram illustrates the high-level architecture of the Carb Capture Application, showcasing interactions between its core software components and external APIs. The system is built around a modular design with key elements, including the Renderer, Backend, Data Storage and Processing, and external APIs such as the Dexcom API and OpenAI API. The primary user interacts with the Renderer, which acts as the interface for navigating and submitting data, while the Backend handles API communication, meal data processing, and carb-to-insulin ratio calculations.

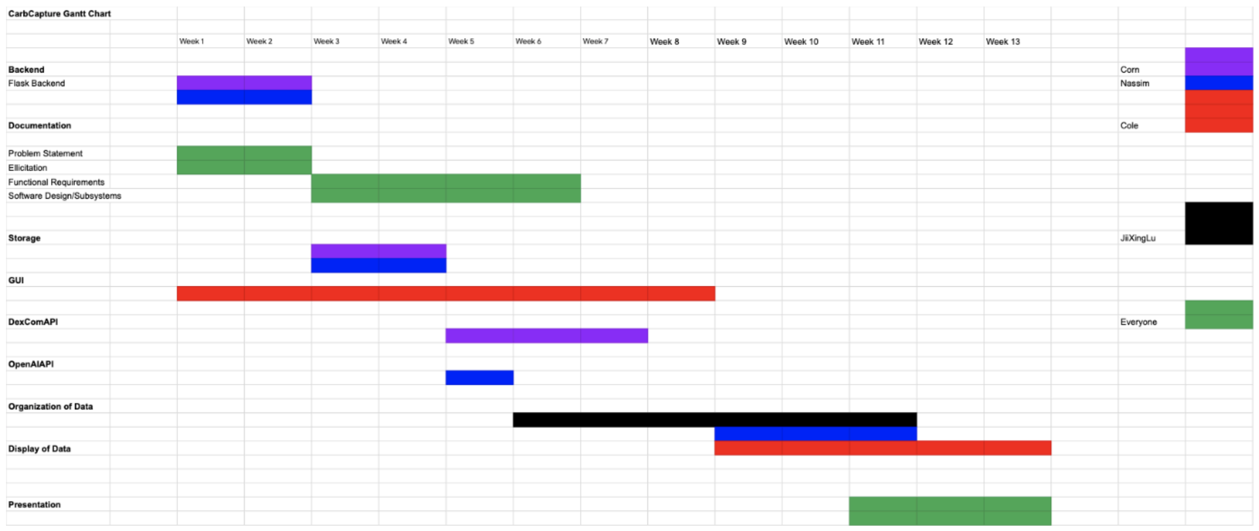
Component Interaction

The Renderer is the user's point of interaction, managing functions like logging in, taking pictures, and rendering settings or glucose data. The Renderer communicates with the Backend, which integrates with the Dexcom and OpenAI APIs. The Dexcom API retrieves glucose levels, while the OpenAI API estimates carbohydrate content from user-submitted food data. Processed results are stored and managed in the Data Storage and Processing component, which maintains user and meal information and ensures continuity in data retrieval.

Functional Workflow

The system begins with the user inputting data through the Renderer, which forwards it to the Backend for further processing. The Backend uses the APIs to fetch external data, combines it with user-provided information, and performs calculations like carb-to-insulin ratio adjustments. Results are stored in the Data Storage and Processing unit, linking user accounts and meal data for future reference. This design ensures seamless interaction, robust data handling, and accurate health recommendations for the user.

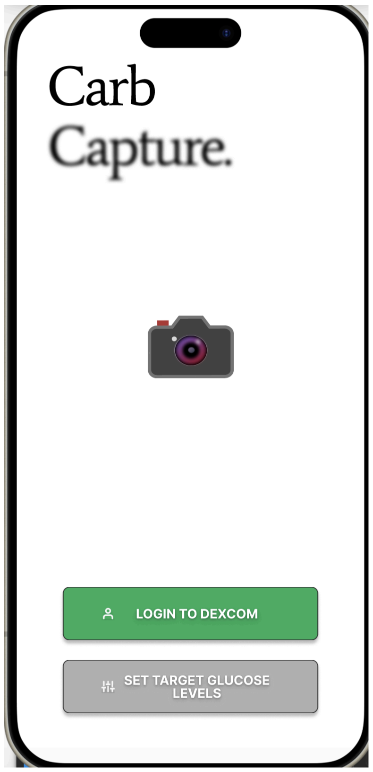
Gantt Chart



The Gantt chart shows the timeline and main tasks for developing the CarbCapture Application. Each row represents a part of the project, like backend development, GUI design, or API integration, while the colored bars indicate when tasks are worked on and who is responsible.
The project begins with backend development to build the core functionality, alongside documentation tasks to define the problem and system requirements. The GUI (Graphical User Interface) is developed to provide the user-facing part of the app.
API integrations with Dexcom and OpenAI ensure the app can fetch glucose data and analyze food images. Afterward, focus shifts to organizing and displaying the data effectively. The project concludes with a presentation to showcase the completed app.

User Interfaces

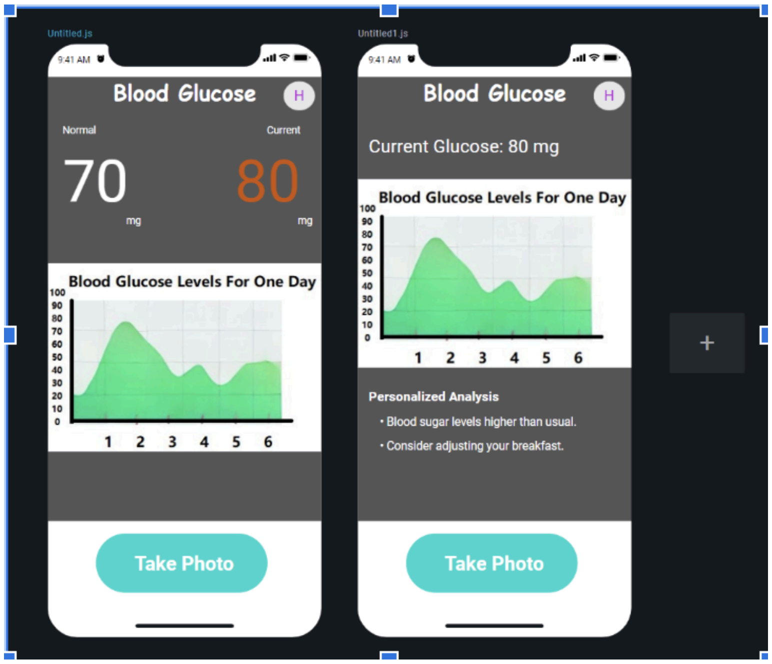
Login Screen



The login screen of the Carb Capture app serves as the starting point for users to access its features. At the top, the app's name, "Carb Capture," is prominently displayed, establishing branding and purpose. The screen includes a central camera icon, indicating the app's focus on capturing food photos, a core functionality for estimating carbohydrate content.

Below the camera icon, two buttons guide user interaction. The first button, "Login to Dexcom," allows users to authenticate their Dexcom account, enabling the app to fetch real-time blood glucose data. The second button, "Set Target Glucose Levels," lets users configure their desired glucose range, providing a personalized experience for insulin recommendations. The clean and minimalist design ensures ease of navigation while focusing on essential functions.

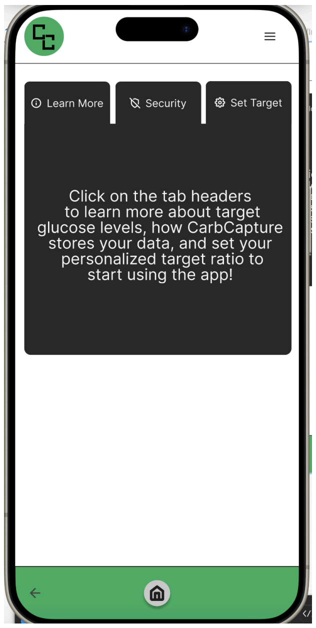
Home Page



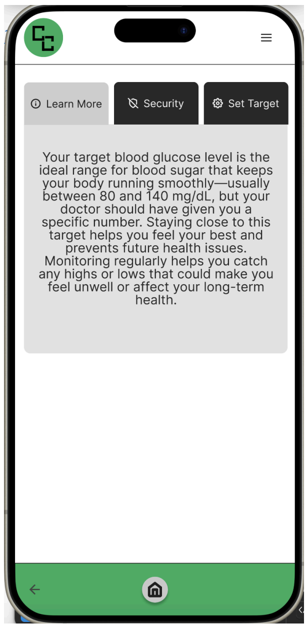
The Home Page of the CarbCapture app provides users with an overview of their blood glucose levels and a clear call-to-action for capturing meal data. The screen prominently displays both the current blood glucose level (e.g., 80 mg/dL) and a comparison to the normal range (e.g., 70 mg/dL), ensuring users can quickly assess their status. Below the glucose values, a graph visually represents blood glucose trends over a day, offering insights into fluctuations and patterns.

Additionally, the app may include a Personalized Analysis section, which provides actionable feedback based on the user's glucose levels, such as dietary recommendations or suggestions to adjust insulin. The Take Photo button at the bottom serves as the primary action, allowing users to capture and analyze meal photos, making it simple to log food intake and correlate it with glucose changes. This screen is designed for clarity and ease of use, keeping users informed and engaged with their health metrics.

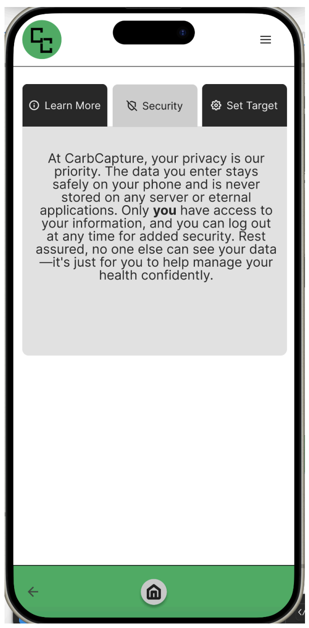
Set Target Glucose Screen



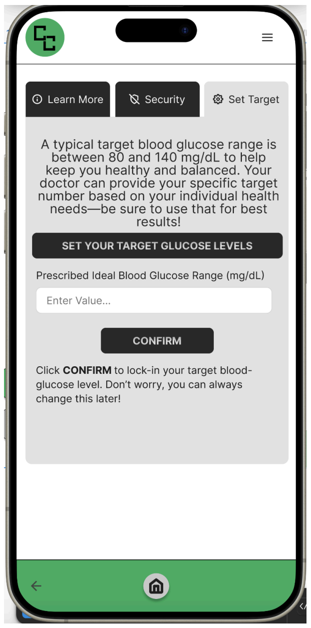
Click on the tab headers to learn more about target glucose levels, how CarbCapture stores your data, and set your personalized target ratio to start using the app!



Your target blood glucose level is the ideal range for blood sugar that keeps your body running smoothly—usually between 80 and 140 mg/dL, but your doctor should have given you a specific number. Staying close to this target helps you feel your best and prevents future health issues. Monitoring regularly helps you catch any highs or lows that could make you feel unwell or affect your long-term health.



At CarbCapture, your privacy is our priority. The data you enter stays safely on your phone and is never stored on any server or eternal applications. Only **you** have access to your information, and you can log out at any time for added security. Rest assured, no one else can see your data —it's just for you to help manage your health confidently.



A typical target blood glucose range is between 80 and 140 mg/dL to help keep you healthy and balanced. Your doctor can provide your specific target number based on your individual health needs—be sure to use that for best results!

Prescribed Ideal Blood Glucose Range (mg/dL)

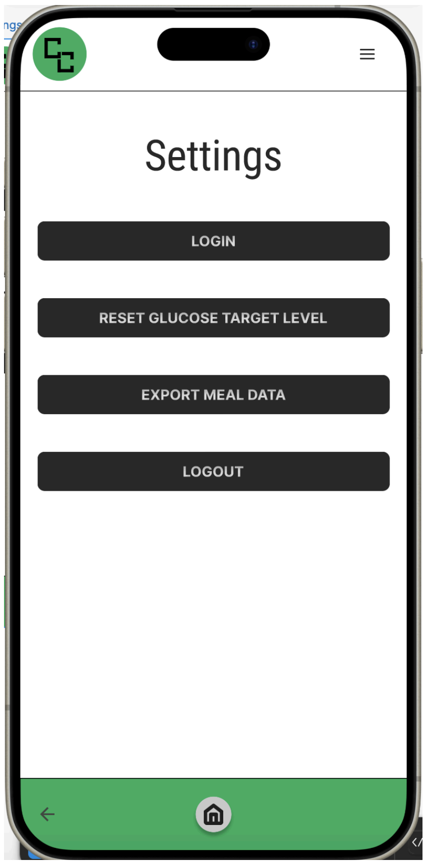
Enter Value...

Click **CONFIRM** to lock-in your target blood-glucose level. Don't worry, you can always change this later!

The Set Target Glucose Screen allows users to personalize their blood glucose target within the app. This feature ensures that users can input their doctor-recommended target glucose level, which helps tailor the app's recommendations to their specific health needs. The screen includes three informational tabs: Learn More, Security, and Set Target, providing a simple and informative user experience.
The Learn More tab educates users on the importance of maintaining a healthy glucose range, typically between 80 and 140 mg/dL, emphasizing how it helps prevent health issues. The Security tab reassures users that their data is securely stored locally on their devices, ensuring privacy and peace of mind. The Set Target tab includes an input field where users can enter their prescribed glucose range and confirm it to lock in their

settings. By combining education, privacy assurance, and functionality, this screen equips users with the tools and knowledge to confidently manage their glucose levels.

Settings Page

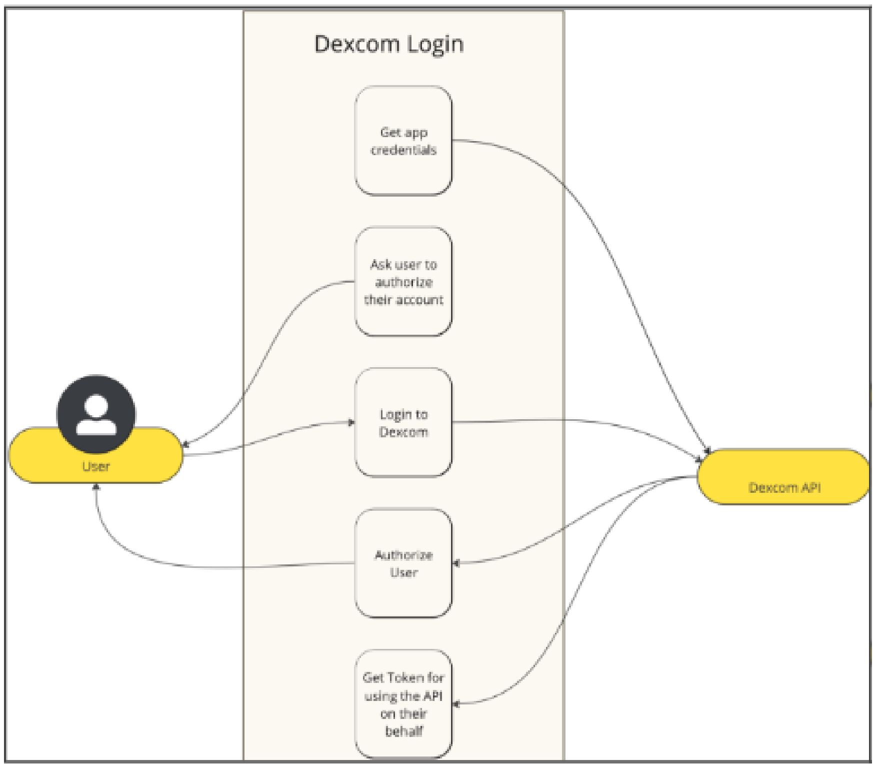


The Settings Page in the CarbCapture app provides users with key controls to manage their account and data within the app. It features four primary buttons, each corresponding to a specific function. The Login button allows users to securely log into their Dexcom account or reauthenticate if needed.

The Reset Glucose Target Level button gives users the flexibility to update or modify their target glucose range, ensuring that their health metrics remain accurate and relevant. The Export Meal Data feature enables users to download or share their recorded meal information for personal tracking or healthcare purposes. Finally, the Logout button allows users to securely exit their account, protecting their data and ensuring privacy. This screen offers a simple and intuitive layout for managing critical app functionalities.

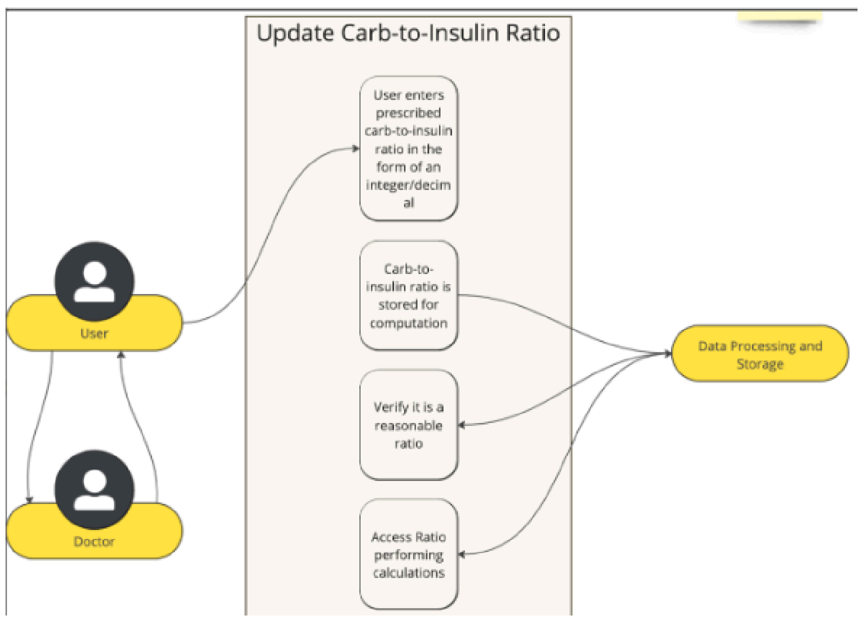
Usage Cases

Dexcom Login


Dexcom Login
Get app credentials
Ask user to authorize their account
Login to Dexcom
Authorize User
Get Token for using the API on their behalf
User
Dexcom API

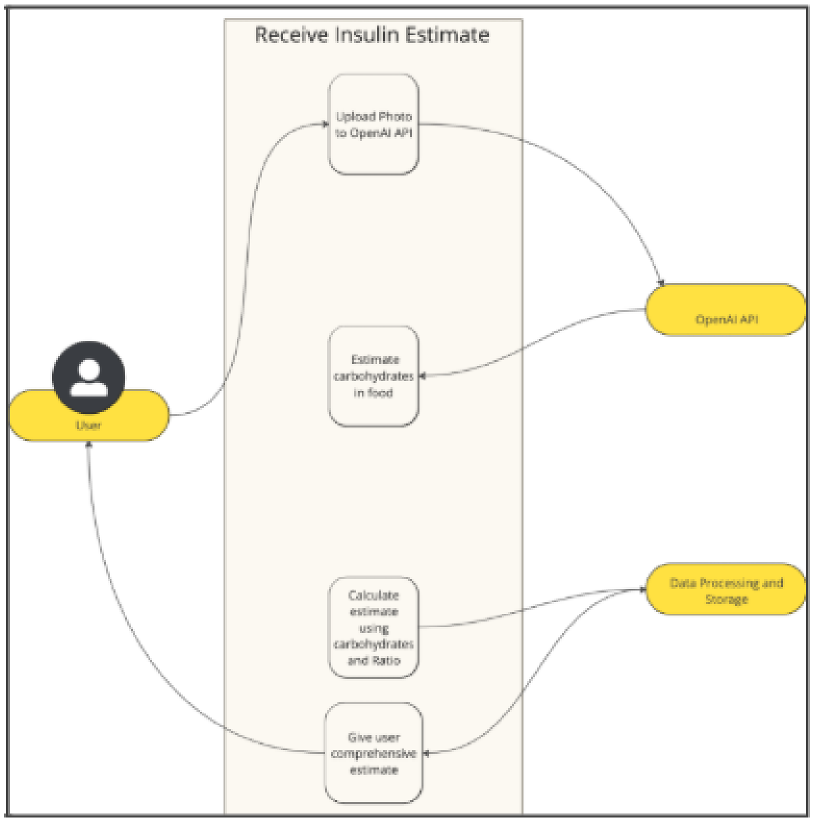
This use case diagram illustrates the Dexcom login process, where a user connects their account to the Dexcom API for glucose tracking. The process begins with the app obtaining app credentials to initiate the connection. The user is then prompted to authorize their account, granting permission for the app to access their data. Next, the app facilitates the user's login to Dexcom, where the Dexcom API verifies and authorizes the user. Once authorization is complete, the Dexcom API provides a token that allows the app to securely access the user's glucose data on their behalf. This flow ensures secure and user-approved integration with the Dexcom API.

Updating Carb-Insulin Ratio


Update Carb-to-Insulin Ratio
User enters prescribed carb-to-insulin ratio in the form of an integer/decimal
Carb-to-insulin ratio is stored for computation
Verify it is a reasonable ratio
Access Ratio performing calculations
User
Doctor
Data Processing and Storage

This use case diagram explains the process of updating the carb-to-insulin ratio in the application. The user enters their prescribed carb-to-insulin ratio, typically provided by their doctor, in the form of an integer. The ratio is then stored in the Data Processing and Storage system for future computations. The system verifies whether the entered ratio is reasonable to ensure accurate calculations. Once verified, the stored ratio is accessed and used for performing various calculations related to insulin recommendations, helping the user effectively manage their blood glucose levels. This process ensures that the ratio is personalized, accurate, and securely stored for ongoing use.

Receiving Insulin Estimate


Receive Insulin Estimate
Upload Photo to OpenAI API
OpenAI API
Estimate carbohydrates in food
User
Calculate estimate using carbohydrates and Ratio
Data Processing and Storage
Give user comprehensive estimate

This use case diagram outlines the process of receiving an insulin estimate based on a food photo and glucose readings. The user uploads a photo of their meal, which is sent to the OpenAI API to estimate the carbohydrate content in the food. Once the carbohydrate estimate is received, the Data Processing and Storage system uses the stored carb-to-insulin ratio to calculate an appropriate insulin estimate. Finally, the system provides the user with a comprehensive insulin recommendation, tailored to their specific health metrics and the meal's carbohydrate content. This process ensures accurate, data driven insulin recommendations for effective blood glucose management.

Additional Use Cases

Track Blood Glucose Trends
Description: Users can view and analyze their blood glucose trends over a specified period, currently set to daily, to identify patterns and anomalies.

Components: User, Data Processing and Storage.
Purpose: Helps users monitor long-term glucose control and make informed adjustments to their diet or insulin regimen.

Generate Nutritional Reports
Description: Users can generate detailed reports of their carbohydrate intake, meal log, and blood glucose level for a given time period.
Components: User, Data Processing and Storage.
Purpose: Self reflection. Provides users with insights to share with healthcare providers for better diabetes management.

Upload Food Photos for Carb Estimation
Description: Users upload photos of their meals to estimate carbohydrate content using AI-based analysis.
Components: User, OpenAI API, Data Processing and Storage.
Purpose: Provides users with an easy and accurate way to log carb intake for better insulin management.

Update and Store Carb-to-Insulin Ratios (Pictured Above)
Description: Users can enter or update their carb-to-insulin ratio, which is securely stored in the system for future calculations.
Components: User, Data Processing and Storage.
Purpose: Ensures accurate insulin recommendations based on user-specific ratios.

Receive Insulin Dose Estimates (Pictured Above)
Description: Users receive personalized insulin dose recommendations based on their blood glucose levels, carb-to-insulin ratio, and meal carbohydrate content.
Components: User, OpenAI API, Data Processing and Storage.
Purpose: Provides actionable guidance for managing insulin dosage effectively.

Dexcom Login for Blood Glucose Data (Pictured Above)
Description: Users log in to their Dexcom accounts to sync real-time blood glucose data with the app.
Components: User, Dexcom API, Data Processing and Storage.
Purpose: Integrates continuous glucose monitoring data to enhance accuracy and convenience for the user.

Conclusion

Possible Updates and Expansion

- **Offline Data Logging:** Enable users to manually log meals, blood glucose levels, and insulin doses when offline, with automatic syncing once back online.
- **Multiple User Profiles:** Allow caregivers or parents to manage profiles for dependents, such as children or elderly family members.
- **Custom Meal Suggestions:** Provide meal or snack recommendations based on current blood glucose levels, carb-to-insulin ratios, and dietary preferences.
- **Blood Glucose Alerts:** Implement real-time notifications to alert users when their blood glucose levels are outside their target range.
- **Goal Setting and Tracking:** Introduce a feature for users to set health-related goals and visually track their progress over time.
- **Integration with Wearable Devices:** Sync with wearable devices like Fitbit or Apple Watch for continuous monitoring of glucose levels, physical activity, and calories burned.
- **Advanced Reporting Features:** Expand reporting capabilities with trend analytics, comparative reports, and projections, exportable in formats like PDF or CSV.
- **Educational Content:** Include a library of educational resources and tips on topics like carb counting, exercise effects, and managing glucose fluctuations.
- **Integration with Insulin Pumps:** Sync with compatible insulin pumps to automate dose calculations and delivery based on glucose and carb data.
- **Improved AI-Based Meal Recognition:** Enhance AI to better recognize mixed dishes and provide detailed nutritional information, including protein and fat estimates.
- **Expanded API Integrations:** Integrate with additional services or APIs, such as nutrition databases or other CGM systems, to broaden compatibility.
- **Cloud Backup and Data Portability:** Add secure cloud storage for backing up data, allowing users to access it across multiple devices or transfer it to other health platforms.

Summary

The CarbCapture app is a tool designed to help users effectively manage their blood glucose and insulin needs by integrating AI analysis, real-time data, and user-friendly interfaces. The app's primary features include the ability to upload food photos for AI-based carbohydrate estimation, track blood glucose trends using Dexcom

API data, and calculate personalized insulin dose recommendations based on stored carb-to-insulin ratios. These functionalities are supported by a well defined architecture, combining a Flask-based backend, API integrations, and a streamlined user interface for seamless navigation.

The app architecture involves components like the backend for API communication and data processing, integration with Dexcom for real-time glucose monitoring, and the OpenAI API for analyzing food images. Users interact with the system via a GUI that facilitates tasks such as logging in, setting glucose targets, and exporting nutritional data. Data storage ensures that user inputs, such as carb-to-insulin ratios and logged meals, are securely maintained for accurate future recommendations. Together, these components create a cohesive system that balances technical complexity with ease of use.

Requirement elicitation played a crucial role in the app's development, focusing on identifying user needs such as reliable carb estimation, intuitive interfaces, and secure data handling.

The app supports several use cases, including meal photo analysis, personalized insulin recommendations, tracking glucose trends, and generating nutritional meal logs. These features provide users with actionable insights to manage their diabetes effectively and share relevant data with healthcare providers when needed.

Future expansions, such as offline data logging and enhanced AI capabilities, could further improve the app's usability and scope. However, in its current state, CarbCapture is a functional and efficient tool that bridges the gap between real-time data, advanced AI, and user focused design to support effective diabetes management.