

Turning CGM Data into Something People can Actually Understand: Flask-Based Approach to Diabetes Data Analysis

Saumil Tripathi¹

Publication Date: 2026/10/07

Abstract: Continuous glucose monitors (CGMs) have changed how people with diabetes interact with their glucose data. Instead of relying on occasional finger-stick measurements, people can now see glucose values throughout the day and night. Over time, this creates an enormous amount of information about how the body responds to meals, exercise, medication, sleep, stress, and everyday life. The challenge is no longer simply collecting glucose data. For many people, the harder part is figuring out what all of that data actually means. This project addresses that problem through a Flask-based web application designed to analyze continuous glucose monitoring data, particularly exports from systems such as Dexcom. The application takes a glucose data file, cleans and organizes the readings, calculates relevant metrics, identifies recurring patterns, analyzes glucose behavior by time of day, compares glucose changes with recorded events, and presents the results through a visual web interface. An artificial intelligence layer is also used to translate the findings into simpler language and generate topics and questions that users can discuss with healthcare professionals. The goal is not to diagnose diabetes-related conditions or replace a physician. Instead, the application acts as an interpretation layer between raw CGM data and the person trying to understand it. By turning thousands of individual readings into recognizable trends, the application can reduce some of the manual work involved in reviewing glucose data and help make conversations between patients and healthcare providers more focused. More broadly, the project shows how software engineering, data analysis, web development, and generative artificial intelligence can work together to address a real-world healthcare problem. It also demonstrates an important idea in health technology: software does not always need to make medical decisions to be useful. Sometimes its most valuable job is simply making existing information easier for people to understand.

How to Cite: Saumil Tripathi (2026) Turning CGM Data into Something People can Actually Understand: Flask-Based Approach to Diabetes Data Analysis. *International Journal of Innovative Science and Research Technology*, 11(9), 3150-3156. <https://doi.org/10.38124/ijisrt/26sep1349>

I. INTRODUCTION

Diabetes is one of the largest chronic health challenges in the United States. According to the Centers for Disease Control and Prevention's 2026 National Diabetes Statistics Report, an estimated 40.1 million Americans had diagnosed or undiagnosed diabetes in 2023, representing approximately 12% of the U.S. population. The report also estimated that 2.1 million people in the United States had diagnosed type 1 diabetes, including more than 300,000 children and adolescents. [1] Those numbers show how widespread diabetes is, but they do not fully explain what managing the condition actually looks like. For many people with diabetes, management is not something that happens once a day or once a week. It can involve constantly thinking about glucose levels, meals, exercise, insulin, medication, sleep, stress, and the possibility of glucose becoming dangerously high or low. Continuous glucose monitoring has reduced some of the burden associated with traditional glucose testing by providing much more frequent information. Modern CGMs can show current glucose levels, trend directions, historical graphs, time in range, and other statistics over periods ranging from several days to several months. [2] That improvement creates another problem, though: there can be a lot of

information to process. A CGM can generate thousands of measurements over several weeks. Looking at a graph might show that glucose went high, but it does not necessarily explain whether the same thing happened repeatedly, whether it happened around a certain time of day, whether it was connected to a person's usual meal schedule, or whether the pattern was significant enough to bring up at a medical appointment. That is the problem this project attempts to address. I developed a web-based CGM analysis system using Python, Flask, HTML, CSS, JavaScript, data-processing libraries, visualization tools, and a generative AI API. Instead of simply displaying glucose readings, the application attempts to turn raw CGM exports into a structured report that highlights the most important patterns. The basic idea is simple: Instead of asking a person to interpret thousands of glucose readings, let software identify the structure in that data and present the important patterns clearly. The application is therefore not another glucose meter or a replacement for a CGM. It is an additional analytical layer built on top of information that already exists.

➤ The Problem with Raw CGM Data

A continuous glucose monitor is useful because it provides a continuous view of glucose behavior. However,

having continuous data does not automatically make that data easy to understand. Imagine that someone has 90 days of CGM readings. Their export could contain thousands of individual measurements. Manually inspecting the entire dataset would be unrealistic. Even if someone had enough time to look at every reading, individual values would not necessarily reveal the most important patterns. For example, a single glucose value of 220 mg/dL tells a very different story from glucose reaching approximately 220 mg/dL after dinner on 70% of the days in a three-month dataset. The first is an observation. The second is a pattern. That difference is one of the central ideas behind this project. Current diabetes technology already provides many important measurements. Dexcom, for example, provides information such as time in range, average glucose, glucose management indicators, logged events, and customizable time periods through its trends tools. [2] International consensus recommendations also emphasize measurements such as time in range, time above range, time below range, mean glucose, and glucose variability when interpreting CGM data. [3] The problem is not that existing diabetes technology lacks useful information. The problem is that a person may still have difficulty turning all of that information into a coherent story. Diabetes is also highly individual. A pattern that occurs consistently for one person may not occur for another. Glucose behavior can change depending on meals, exercise, medication, sleep, stress, illness, and many other circumstances. Because of this, the application focuses on finding patterns within an individual's own data rather than assuming that one explanation applies to everyone.

II. WHAT THE APPLICATION DOES

The process begins when a user uploads a CGM data file. The Flask backend receives the file and passes it through a data-processing pipeline. The pipeline identifies relevant timestamp and glucose columns, removes unusable records, converts timestamps into a consistent format, converts glucose values into numerical values, and creates a standardized dataset that the rest of the application can analyze. This preprocessing step is more important than it might initially seem. Real-world healthcare exports are rarely perfectly consistent. Different files can use different column names, contain missing values, include information that is not needed for the analysis, or represent certain measurements differently. A useful analysis system cannot assume that every uploaded file will have exactly the same structure. The application therefore attempts to handle some of that variation instead of requiring the user to manually reorganize their entire dataset. Once the data has been cleaned, the application calculates several categories of information.

➤ Time in Range

One of the main measurements is time in range. The application calculates the percentage of readings within the selected target range, along with the percentages above and below that range. A commonly used target range for many adults is 70–180 mg/dL, although individual targets can differ. [3][4] Time in range is useful because an average glucose value can hide important differences. For example, two people could have the same average glucose while one

person spends most of their time close to the target range and the other repeatedly moves between high and low glucose levels. An average alone therefore cannot fully describe glucose behavior.

➤ Average Glucose

The application also calculates the average glucose across the available readings. Average glucose gives a useful high-level picture of the dataset, but the application does not treat it as the complete story. Instead, it is presented alongside time in range, time above range, time below range, variability, and time-based patterns. This is intentional. The project is designed around the idea that a meaningful summary should describe several dimensions of glucose behavior instead of reducing a person's data to one number.

➤ Glucose Variability

The application also analyzes glucose variability. One important measurement of variability is the coefficient of variation (CV). International CGM recommendations have identified a CV of approximately 36% or less as a commonly used benchmark for many people with diabetes. [3] The application can use variability to identify whether glucose values are relatively stable or fluctuate substantially. Again, the goal is not to diagnose a medical problem from a single number. Variability is simply another piece of evidence that helps describe the overall dataset.

➤ Detecting Patterns Instead of Isolated Readings

One of the most important features of the application is its ability to identify repeated glucose behavior. The software groups readings into periods and looks for recurring patterns. This helps distinguish between a single unusual event and something that happens repeatedly. For example, a user might have several high glucose values throughout a 90-day period. On its own, that information may not be especially useful. However, if the application finds that elevated glucose repeatedly occurs during the afternoon and evening, the result becomes much more meaningful. The report could communicate the finding as: Glucose levels are frequently higher during the afternoon and evening. That is much easier to understand than thousands of individual readings. The application can also identify periods above and below the selected glucose range. Importantly, these are treated as episodes rather than simply counting every abnormal reading. For example, ten consecutive high readings could represent one prolonged high-glucose episode rather than ten completely separate events. The software therefore attempts to preserve the timing and structure of glucose data instead of treating every abnormal measurement as an independent event.

➤ Time-of-Day Analysis

Another major part of the application is time-of-day analysis. The software divides glucose readings into periods such as overnight, early morning, morning, midday, afternoon, evening, and late night. It then calculates glucose behavior within those periods.

This allows the report to answer questions such as:

- Is glucose generally higher in the morning?
- Does glucose rise during the afternoon?
- Are nighttime readings significantly different from daytime readings?
- Does glucose variability appear concentrated during a particular period?
- Are high or low episodes more common at certain times of day?

This feature matters because diabetes data is inherently connected to time. A glucose value without its timestamp loses much of its context. For example, a glucose value of 180 mg/dL may have a different analytical meaning depending on whether it repeatedly occurs overnight, shortly after a recorded meal, or during physical activity. The application does not automatically assume what caused those differences. Instead, it uses time as another way to organize the evidence.

➤ *Meal-Time Pattern Analysis*

One of the more interesting parts of the project is the distinction between usual meal times and actual recorded meals. That distinction was deliberately built into the application because a schedule is not the same thing as an event. Suppose a user says they normally eat dinner around 8:00 PM. The application can examine what tends to happen to glucose around 8:00 PM across multiple days. However, it does not claim that the person actually ate dinner at exactly 8:00 PM every day. That prevents the software from turning an assumption into a fact. Instead, the application can report that glucose frequently rises around the user's usual dinner time. That observation can then become a question for further investigation. If the user provides timestamped meal data, the application can treat those events differently. It can compare glucose immediately before a recorded meal with the highest glucose observed during the following period.

This creates two separate categories:

- *Schedule-Based Analysis:*
“What tends to happen around the time this person usually eats?”
- *Event-Based Analysis:*
“What happened after this specific recorded meal?”

Keeping these categories separate makes the analysis more responsible because the software is not pretending to know something that was never actually recorded.

➤ *The Visualization Layer*

After analyzing the dataset, the application presents the results through a web-based dashboard.

The interface was built around a straightforward principle: The user should not need to understand the code or the underlying data structure to understand the result.

The report begins with high-level metrics such as:

- Time in range
- Average glucose
- Time above range
- Time below range

It then displays the glucose graph across the analyzed period.

Additional sections show recurring episodes, time-of-day behavior, meal-time patterns, and supporting evidence. This creates a hierarchy of information:

Overview → Trends → Patterns → Supporting evidence

That structure matters because medical data can become overwhelming quickly. Research on CGM experiences has found that people can benefit from greater control, convenience, reassurance, and awareness, but users can also experience frustration related to device complexity, alerts, technology, and feeling overwhelmed by data. [5] This project attempts to address one part of that burden: the difficulty of interpreting large amounts of information.

III. THE ARTIFICIAL INTELLIGENCE LAYER

The most experimental part of the project is the AI-assisted review. After the deterministic analysis is complete, the application can send a limited evidence report to a Gemini model. The AI does not receive the entire raw CGM dataset.

Instead, the application creates a smaller representation of the analysis containing information such as:

- Target glucose range
- Data quality
- Measured metrics
- Time-of-day observations
- Usual meal-time observations
- Recorded event associations
- Episode counts
- Known limitations

➤ *This Design is Intentional*

There is little reason for a language model to receive thousands of raw readings when the Python analysis layer has already calculated the relevant measurements. The Python backend therefore acts as the quantitative analysis engine, while the AI primarily acts as a communication layer.

The architecture can be summarized as:

Raw data → deterministic analysis → structured evidence → AI explanation

This separation is one of the strongest architectural choices in the project. The application does not ask an AI model to independently calculate the entire glucose report.

Instead, the model receives information that has already been processed and structured by the application.

➤ *Guardrails Against Medical Hallucination*

Because this application operates in a healthcare-related context, unrestricted AI-generated medical advice would be irresponsible. The AI prompt therefore includes explicit restrictions.

The model is instructed not to:

- Diagnose medical conditions
- Prescribe medication
- Recommend insulin changes
- Tell the user what they should eat
- Invent glucose readings
- Invent dates
- Invent patient history
- Assume that a meal occurred
- Claim that a specific factor caused a glucose change
- Provide treatment instructions

Instead, the AI is instructed to describe observations and frame possible explanations as topics that could be discussed with a healthcare professional.

For example, the application should not say: “Your evening glucose is high because you eat too many carbohydrates.” That would be an unsupported causal claim. A safer interpretation would be: “Glucose levels are frequently higher during the evening. Meal timing, activity, or medication timing may be worth discussing with your care team, but the available CGM data alone cannot determine the cause.” The second statement stays much closer to the evidence. The AI also returns a predefined JSON structure rather than unrestricted text. The backend validates the returned structure before displaying it. This is another example of treating AI as one component inside a software system rather than allowing the model to control the entire application.

➤ *Why the AI is Useful*

The AI layer has a different purpose from the numerical analysis. Python is good at calculating. A language model is good at communicating.

The Python system can determine that:

- Afternoon average glucose is elevated.
- Evening time in range is lower.
- Several episodes occur after a certain time.
- Glucose variability is elevated.

However, a list of measurements is not necessarily easy for someone to understand.

The AI can turn those findings into a short explanation, such as: “Your glucose tends to run higher during the afternoon and evening, with repeated increases around your usual meal times.” The AI is not creating new medical

evidence. It is organizing existing evidence into language that is easier to understand. That distinction is important because the strongest role for AI in this project is translation rather than diagnosis.

➤ *Questions Instead of Answers*

One of the most important design decisions in the project is that the application can produce questions for a care team instead of pretending to know the answer. For example, a report might identify repeated evening glucose elevations. Rather than telling the user what they should do, the application could generate a question such as: “Could my repeated evening glucose elevations be worth discussing in relation to my meals, activity, or medication timing?” This creates a bridge between a person’s personal data and their healthcare appointment. That could be useful because diabetes care often involves communicating a large amount of information between the person managing the disease and the healthcare professional interpreting it. The application can help the patient arrive at that conversation with a more organized understanding of their own data. It does not replace the conversation. It helps prepare for it.

IV. POTENTIAL IMPACT ON THE DIABETES COMMUNITY

The potential impact of a project like this goes beyond making a better glucose graph. Diabetes management is already highly data-driven. The American Diabetes Association describes diabetes technology as including devices and software that support self-management, glucose monitoring, and treatment. Its 2026 Standards of Care also emphasize that diabetes technology should be individualized based on a person’s circumstances, preferences, needs, and skills. [6] That creates an opportunity for software that improves the usability of information people already have.

➤ *Reducing Information Overload*

A person may have weeks or months of CGM readings but only a limited amount of time to review them. The application attempts to compress thousands of data points into a smaller number of meaningful findings. Instead of asking someone to inspect every glucose spike, the system asks: What patterns repeat? That change in perspective can make large datasets less intimidating.

➤ *Helping Patients Prepare for Appointments*

A typical medical appointment does not provide unlimited time to manually review every glucose reading.

The application could help a patient arrive with a concise summary of:

- Average glucose
- Time in range
- Recurring high and low periods
- Time-of-day patterns
- Recorded meal associations
- Questions they want to ask

That could make the conversation more focused. Existing CGM systems already support sharing data with healthcare professionals, and Dexcom specifically describes its trends information as something that can support discussions during appointments. [2] The difference in this project is the additional analytical layer placed between the raw data and that conversation.

➤ *Making Data More Understandable for Young People*

This is especially relevant for teenagers and young adults. A young person may understand that their glucose “keeps going high,” but understanding exactly when it happens and how consistently it happens can be much harder. A visually clean report can turn an abstract statement into something concrete: “Your glucose was higher during the evening on most of the analyzed days.” That makes diabetes data feel less like an endless stream of numbers and more like information that can actually be interpreted.

➤ *Supporting Patient Agency*

The project is built around the idea that patients should be able to understand their own data. That does not mean patients should independently diagnose themselves. Instead, better information can help someone ask better questions. That distinction matters. Healthcare technology should not make a patient completely dependent on an algorithm. It can instead give the patient a clearer understanding of their own information before they interact with the people responsible for their medical care.

➤ *The Psychological Side of Diabetes Data*

The impact of diabetes is not purely physiological. The National Institute of Diabetes and Digestive and Kidney Diseases describes diabetes distress as the emotional burden associated with living with and managing a demanding chronic condition. Research has also connected diabetes distress with difficulties in self-management and other diabetes-related outcomes. [7] This matters for a data-analysis application because giving someone more information is not automatically helpful. More data can sometimes create more stress. A systematic review of qualitative studies examining CGM experiences found that users described benefits such as greater control, convenience, reassurance, and confidence. At the same time, some users reported feeling overwhelmed by data, frustrated by technology, burdened by alerts, and limited by access barriers. [5] That creates an important design challenge: The application needs to simplify the data without oversimplifying the person. That is why the interface focuses on patterns instead of flooding the user with every statistic the software can calculate. The goal is not to create another screen that someone has to constantly monitor. The goal is to create a report that someone can actually look at and understand.

V. TECHNICAL ARCHITECTURE

From a software engineering perspective, the project combines several different layers.

➤ *Frontend*

• *The Frontend Uses:*

- ✓ HTML
- ✓ CSS
- ✓ JavaScript
- ✓ Plotly-generated visualizations
- ✓ Jinja templating through Flask

The interface displays the processed report and allows users to download their analyzed JSON data.

➤ *Backend*

• *The Backend Uses Flask to Handle:*

- ✓ File uploads
- ✓ Report generation
- ✓ Data processing
- ✓ AI requests
- ✓ Rendering the final report

➤ *Data Processing*

• *The Analysis Pipeline:*

- ✓ Loads the uploaded dataset.
- ✓ Identifies relevant columns.
- ✓ Cleans invalid or missing records.
- ✓ Converts timestamps.
- ✓ Converts glucose measurements into numerical values.
- ✓ Calculates summary metrics.
- ✓ Detects glucose episodes.
- ✓ Analyzes time-of-day patterns.
- ✓ Evaluates meal-time associations.
- ✓ Produces a structured evidence report.

➤ *AI Layer*

• *The AI Layer:*

- ✓ Receives the structured evidence report.
- ✓ Removes unnecessary raw data.
- ✓ Sends the limited evidence to Gemini.
- ✓ Requests a structured JSON response.
- ✓ Parses the response.
- ✓ Validates the expected fields.
- ✓ Displays the result only if the response matches the expected structure.

This architecture creates a clear separation between data processing and language generation. That separation makes the system easier to test, debug, and improve.

➤ *Why a Rule-Based Analysis Layer Matters*

One of the biggest lessons from building this project is that not everything needs AI. In fact, many of the most important calculations should not depend on AI at all. If the application needs to calculate time in range, Python can calculate it deterministically. If it needs to calculate an average, Python can calculate it. If it needs to determine whether a timestamp falls within an evening period, Python can handle that as well. None of those tasks require a language model. AI becomes more useful after those calculations have already been completed. This approach also makes the application easier to debug. If the AI produces an incorrect explanation, the underlying numerical report can be inspected separately. If a metric is incorrect, the data-analysis layer can be tested separately. That modularity is a major software engineering advantage.

➤ *Data Quality and Limitations*

A major limitation of the application is that the quality of its analysis depends on the quality and completeness of the uploaded data. If a CGM export contains missing readings, the application cannot reconstruct what happened during those missing periods. International consensus recommendations emphasize that sufficient CGM data coverage is important for meaningful interpretation. [3] The application therefore includes data-quality information in its report rather than pretending every dataset is equally reliable. Another limitation is that CGM data alone cannot explain causation. The data can show that glucose frequently rises after a certain time. It cannot automatically prove why. The reason could involve meals, activity, medication, stress, sleep, illness, timing, or factors that were never recorded. This is why the application separates observed evidence from possible explanations to discuss. That distinction is central to the project's safety philosophy.

➤ *The Application is Not a Medical Device*

The project also needs to be understood within the limits of what it actually is. It is a student-built software application and has not undergone clinical validation, regulatory review, prospective clinical trials, or formal evaluation of diagnostic accuracy. It should therefore not be represented as a replacement for a physician, diabetes educator, endocrinologist, or other healthcare professional. It should also not be used to determine insulin doses or make medication changes. The safest description of the application is that it is an analytical and educational tool designed to organize CGM information and help users identify questions to discuss with their healthcare team. Acknowledging that limitation does not make the project less meaningful. If anything, it demonstrates a more mature understanding of healthcare technology. Building software for a medical context requires knowing not only what the software can do, but also what it should not do.

➤ *Future Development*

There are several directions in which the application could eventually become more capable.

➤ *Integration with More Data Sources*

The current system primarily analyzes CGM information and optional event data.

• *Future Versions could Incorporate:*

- ✓ Exercise
- ✓ Sleep
- ✓ Heart rate
- ✓ Insulin doses
- ✓ Medication timing
- ✓ Meal composition
- ✓ Stress or symptom logs

Combining these datasets could make pattern analysis much more detailed. However, additional data would also create additional privacy and security responsibilities.

➤ *More Advanced Pattern Detection*

• *The Application could Eventually Identify Patterns Such as:*

- ✓ Repeated overnight drops
- ✓ Recurring morning increases
- ✓ Post-meal glucose responses
- ✓ Day-of-week differences
- ✓ Exercise-related changes
- ✓ Persistent evening elevations
- ✓ Relationships between sleep duration and glucose variability

Machine-learning techniques could eventually be explored for detecting patterns that are difficult to capture using fixed rules. However, these systems would need careful validation before they could support meaningful clinical claims.

➤ *Personalized Question Generation*

A future version could make the AI interview more interactive. Instead of generating several questions at once, the application could ask one question, interpret the response, and use that information to determine the next most useful question.

- For example:
- Application:

“Your glucose appears to rise frequently after dinner. Do these increases happen after meals that are larger than usual?”

The user's response could determine the next question.

This would transform the application from a static report into a more personalized investigation system.

The important part would be keeping the system focused on identifying missing context rather than allowing the AI to jump to conclusions.

➤ Privacy and Security

Healthcare data is especially sensitive. A future production version of this application would need strong privacy protections, including secure storage, encrypted connections, authentication, careful logging practices, data deletion controls, and strict limits on what information is transmitted to external APIs. The current AI architecture already demonstrates one privacy-oriented design choice by creating a minimized evidence payload instead of sending the entire raw dataset to the language model. That approach could be expanded further. Ideally, as much processing as possible should happen locally before information is sent to an external service. The application should also clearly explain when information leaves the application and which services receive it.

➤ Broader Significance

The larger lesson of this project extends beyond diabetes. Modern healthcare is generating enormous amounts of data. Wearable devices, CGMs, smartwatches, fitness trackers, medical records, laboratory tests, and patient-generated logs can all produce information that is technically available but difficult to interpret. The challenge is increasingly becoming: How do we turn data into understanding? This project represents one possible answer. The software does not attempt to replace medical expertise. Instead, it organizes information before that expertise is applied. That concept could eventually be useful in many areas of healthcare. A patient's raw data could be summarized before an appointment. Repeated patterns could be identified. Important questions could be highlighted. Large datasets could be translated into language that patients actually understand. The technology becomes useful not because it knows everything, but because it helps people make sense of what they already know.

VI. CONCLUSION

The CGM analysis application began as a software project, but the deeper problem it addresses is much more human: living with diabetes means constantly interacting with information. A person can have thousands of glucose readings and still not know what those readings mean as a whole. This application attempts to close that gap. It takes raw CGM data and processes it into meaningful measurements. It identifies recurring episodes, examines glucose behavior across different times of day, analyzes patterns around usual and recorded meal events, visualizes the results, and uses a carefully constrained AI layer to explain the findings in simpler language. Most importantly, it does not pretend that software can determine everything about a person's diabetes. Instead, it treats the data as evidence. It separates observation from speculation, schedules from actual events, and explanation from diagnosis. That philosophy may be more important than any individual feature. The potential impact of the application is therefore not simply that it can calculate time in range faster. Its potential impact is that it can help someone look at a large and intimidating dataset and walk away with a clearer understanding of what is actually happening. For someone managing diabetes, that could mean fewer numbers to sift

through, clearer patterns to recognize, and better questions to bring to a healthcare professional. For the broader diabetes community, projects like this show how software does not always need to invent a new medical device to be useful. Sometimes the more practical innovation is taking technology that already exists and building a better bridge between the data it produces and the people who have to live with that data every day. The application is a tool for turning a massive amount of glucose data into something a person can actually understand; and that is the problem it was built to solve.

REFERENCES

- [1]. Centers for Disease Control and Prevention. *National Diabetes Statistics Report: Estimates of Diabetes and Its Burden in the United States*. 2026.
- [2]. Dexcom. *What Can I Find on the Trends Tab of My Dexcom G7 App?* 2026.
- [3]. Battelino, T., Danne, T., Bergenstal, R. M., et al. "Clinical Targets for Continuous Glucose Monitoring Data Interpretation: Recommendations From the International Consensus on Time in Range." *Diabetes Care*, 2019.
- [4]. American Diabetes Association. "Glycemic Goals and Hypoglycemia: Standards of Care in Diabetes." *Diabetes Care*.
- [5]. Systematic review of qualitative studies. "Patient Experiences of Continuous Glucose Monitoring and Sensor-Augmented Insulin Pump Therapy for Diabetes." 2023.
- [6]. American Diabetes Association. "Section 7: Diabetes Technology — Standards of Care in Diabetes 2026."
- [7]. National Institute of Diabetes and Digestive and Kidney Diseases. "Diabetes Distress and Depression."
- [8]. American Diabetes Association. "Diabetes Technology: Improve Diabetes Outcomes with ADA Tools."
- [9]. American Diabetes Association. *Standards of Care in Diabetes 2026*.
- [10]. Recent research on diabetes distress, CGM metrics, and psychosocial outcomes in people with diabetes.