Glassboard Requirments Document

Team: Connor Babb

Description:

Build a lightweight analytics dashboard for small businesses that helps them track what's happening on their websites without the complexity of tools like Google Analytics. The system would work by giving a business owner a simple JavaScript snippet they can drop into their site, which automatically tracks user actions such as clicks, purchases, and referral sources. This data would be sent to your backend (Python + SQL), stored in a structured database, and displayed in a clean dashboard with charts and metrics like most-clicked items, conversion rates, and revenue trends. The focus is on making it easy to set up and understand, providing small businesses with actionable insights about their traffic and sales in a way that's both accessible and scalable.

Introduction: Purpose of Project

The purpose of Glassboard is to give small business owners an accessible, easy-to-use analytics tool that helps them understand how visitors interact with their websites. Many existing analytics platforms such as Google Analytics are very complex, cluttered, or require significant setup and interpretation skills. Glassboard aims to simplify this by offering a plug-and-play solution: a lightweight JavaScript embeddable snippet that automatically tracks clicks, page visits, and other statistics, sending that data to a backend system that turns it into clear, actionable insights. The goal is to make web analytics intuitive and approachable, even for non-technical users.

This project is solving the problem of visibility and usability in website data for smaller organizations. Small business owners often lack the time, resources, or expertise to configure and analyze enterprise-level analytics tools. Glassboard removes those barriers by combining simple setup with meaningful data visualization, allowing users to quickly identify which pages attract the most attention, what elements drive conversions, and how customers engage with their site. In short, it helps users make smarter, data-driven decisions without too much technical overhead.

Must Have Requirements:

1. **User Account Creation and Login:**
    a. *Success Measure:* Users can register using a valid email and password, and subsequently log in using those same credentials.
    b. *Demonstration:* Test by creating a new account and logging in successfully, verifying that authentication persists across sessions.

2. **JavaScript Tracking Snippet Generation:**
    a. *Success Measure:* The system generates a unique embeddable JavaScript snippet for each user's website.
    b. *Demonstration:* Confirm that upon account creation, a distinct snippet is available for copying, and that it includes a unique tracking ID tied to the user in the database.
3. **Database and Event Storage:**
    a. *Success Measure:* The system captures and stores user events (e.g., clicks, page visits) in an AWS RDS database.
    b. *Demonstration:* Insert the snippet into a test webpage, trigger user events, and verify event data entries appear in the database tables.
4. **Analytics Dashboard Visualization:**
    a. *Success Measure:* The dashboard visually displays tracked data using charts (visitors, most clicked items, referrer sources).
    b. *Demonstration:* Log in as a user and confirm real-time or recent data appears accurately on the dashboard.
5. **Insights Generation:**
    a. *Success Measure:* The system automatically compares data across daily, weekly, and monthly periods to highlight trends.
    b. *Demonstration:* Generate test data across multiple time frames and verify the dashboard correctly displays comparisons and insights.
6. **User Experience Simplicity:**
    a. *Success Measure:* Users can sign up and begin tracking their site within 5 steps or fewer.
    b. *Demonstration:* Ensure the onboarding process requires no more than 5 interactions to complete setup.

Stretch Requirements:

7. **Real-Time Analytics Updates:**
    a. *Success Measure:* Dashboard metrics update live without requiring a page refresh.
    b. *Demonstration:* Simulate user activity on a tracked site and confirm the dashboard auto-refreshes with new data within seconds.
8. **Custom Event Tracking**
    a. *Success Measure:* Users can define and track custom events (e.g., "Add to Cart" or "Form Submission") through a simple interface.
    b. *Demonstration:* User creates a custom event via the dashboard, adds the corresponding snippet to their site, and sees data populate in the metrics view.
9. **Data Export Functionality:**
    a. *Success Measure:* Users can export analytics data as CSV or PDF reports.
    b. *Demonstration:* From the dashboard, trigger an export and verify that the downloaded file contains accurate and formatted data from the database.
10. **Multi-Website Management:**

a. *Success Measure:* A single account can manage and view analytics for multiple websites.
b. *Demonstration:* Create multiple site profiles under one user account, each generating a distinct snippet and dashboard section, confirming that data remains separated and accurate.

Overview of the Product:

3.1 Workflow

The workflow describes how a user interacts with the analytics dashboard from start to finish:

1. **Account Creation (Input X):**
   The user signs up by providing an email and password twice. Upon successful registration, the system generates a unique user ID and stores the credentials in the authentication database.

2. **Snippet Generation (Process A):**
   Once logged in, the system automatically generates a unique JavaScript snippet tied to that user's account.
   Output: A copyable snippet with a tracking ID (e.g., user_12345).

3. **Website Integration (Input Y):**
   The user pastes this snippet into their website's HTML <head> section. The snippet begins sending event data (clicks, page views, referrers) to the backend API whenever visitors interact with the site.

4. **Event Processing (Process B):**
   The backend (Python + FastAPI) receives event payloads, validates them, and stores them in an AWS RDS database. Data is aggregated and updated periodically to compute metrics like visitor count, most-clicked elements, and referral sources. **Output:** Structured event and metric data stored in the database.

5. **Analytics Visualization (Process C):**
   The user logs into the dashboard, which retrieves the aggregated data through secure API calls and displays it via interactive charts and summaries. Users can filter by time period (daily, weekly, monthly) and view auto-generated insights. **Output:** Actionable visual reports.

6. **(Optional Stretch)** – Users may define custom events, download CSV/PDF reports, or manage multiple websites from one account.

3.2 Resources

Frontend (Client Tier):

- **Technologies:** React.js, Tailwind CSS, Chart.js (or Recharts).
- **Purpose:** Provides the user interface for registration, login, and data visualization. Handles all interactions and data display.
- **Connection:** Communicates with backend via RESTful API calls (JSON over HTTPS).

Backend (Application Tier):

- **Technologies:** Python + FastAPI.
- **Purpose:** Handles authentication, event ingestion, data aggregation, and insight generation.
  Runs periodic background tasks to compute metrics.

Database (Data Tier):

- **Technology:** AWS RDS (PostgreSQL).
- **Purpose:** Stores persistent user data and event logs. Supports both raw and aggregated tables for efficient querying.

Third-Party & Supporting Tools:

- **AWS RDS:** Managed relational database service.
- **JWT / OAuth:** For secure session management.
- **Chart.js / Recharts:** For rendering dashboard charts.
- **Pandas / NumPy:** For backend data analysis and insights generation.

3.3 Data at Rest

- **User Profile Data:** Stored in a Users table containing:
  - UserID (Primary Key).
  - Email (Unique).
  - Password Hash (bcrypt).
  - Date Created.
  - Associated Website(s).

- **Application (Event) Data:** Stored in Events and Aggregates tables:
  - Events: raw user interactions (timestamp, event type, referrer, element clicked).
  - Aggregates: summarized metrics (daily/weekly/monthly visitors, clicks, conversions).

All data is stored in AWS RDS (PostgreSQL) with encryption at rest (AES-256). Access control and IAM roles ensure only the backend API has write privileges.

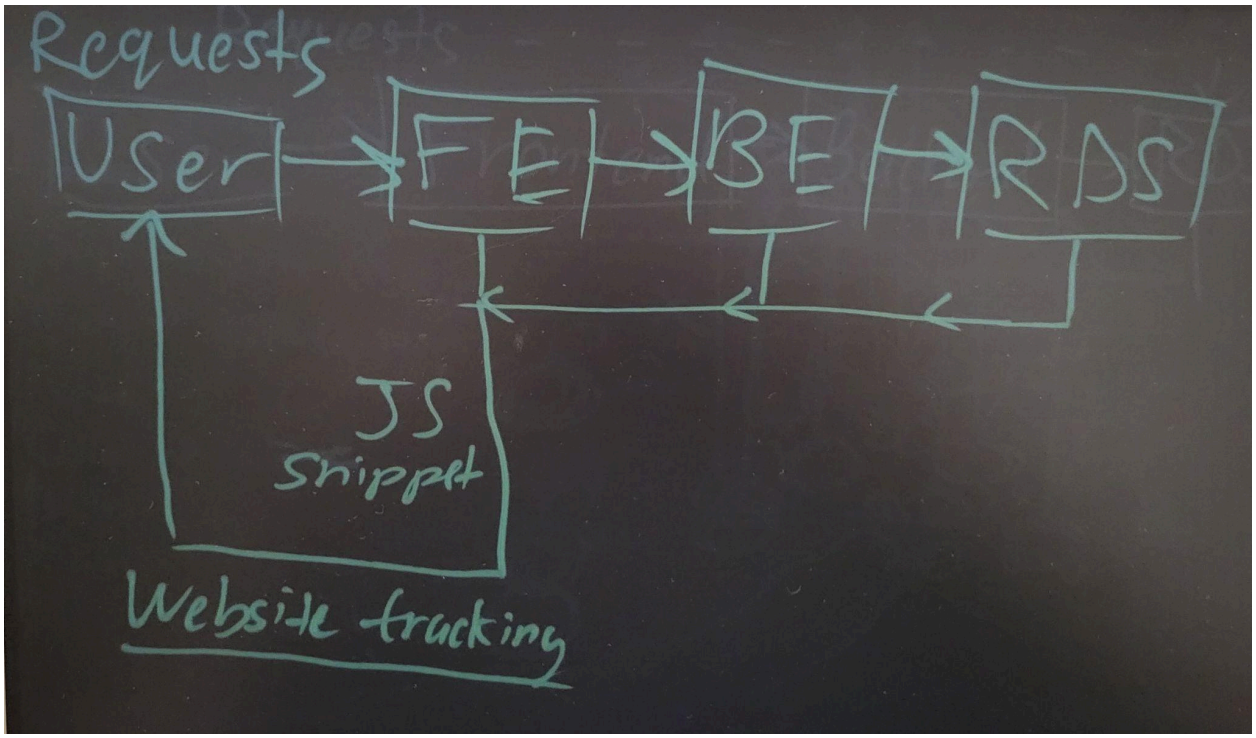
3.4 Data on the Wire

- **Transmission:**
    1. All communication uses HTTPS to encrypt data in transit.
    2. The JavaScript snippet sends event payloads via POST requests (JSON) to the backend API endpoint.
    3. The frontend dashboard fetches analytics data from the API using authenticated requests with JWT tokens.

- **Process:**
    1. JavaScript snippet captures user interaction (e.g., button click).
    2. Sends JSON payload → Backend API.
    3. Backend validates data and inserts it into AWS RDS.
    4. Dashboard requests aggregated metrics → API returns processed JSON data for display.

3.5 Data State (Flow Diagram)

Below is a simplified data flow representation:



This flow shows:

- User interacts via dashboard and website.
- Data travels through snippet → backend → database → back to dashboard.

3.6 HMI / HCI / GUI Prototype

Below is a conceptual mock-up of the user interface (you can sketch or wireframe this using Figma, Excalidraw, or even draw by hand for your report).

Dashboard Mockup (Description)

Top Navigation Bar:

- Logo (top-left)
- “Dashboard”, “Insights”, “Settings” tabs
- User profile icon (top-right)

Main Dashboard View:

- **Metrics Summary Cards** (at top):
  - Visitors Today
  - Top Clicked Item
  - Bounce Rate
  - Conversion Rate
- **Charts Section:**
  - Line Chart – Visitors over Time (Daily/Weekly/Monthly toggle)
  - Bar Chart – Top 5 Clicked Elements
  - Pie Chart – Referrer Sources
- **Insights Panel:**
  - Displays text-based summaries such as “Visitors increased 12% this week compared to last.”

Snippet Page:

- Code block showing the unique JavaScript snippet with a “Copy to Clipboard” button.

Verification: How will the project be tested:

4.1 Demo Plan

Show a video demo of each part of the software working from start to finish.

1. **User Registration & Login:**
    - Create a new account using an email and password.
    - Log in to verify authentication and access to the dashboard.

2. **Snippet Generation & Installation:**
    - Display the unique JavaScript snippet generated for the test user.
    - Copy and paste it into a sample HTML webpage.

3. **Data Collection & Transmission:**
    - Open the webpage in a browser, perform interactions (clicks, refreshes, form submissions).
    - Show event payloads being transmitted to the backend via browser developer tools (network tab).
    - Verify data entries appear in the AWS RDS database.

4. **Dashboard Visualization:**
    - Log into the analytics dashboard.
    - Show live updates of visitor count, click data, and referrer sources appearing in charts.

5. **Insights Comparison:**
    - Switch between daily, weekly, and monthly views to confirm calculated insights update correctly.

6. **Stretch Demonstrations (if completed):**
    - Real-time updates (no manual refresh).
    - Custom event tracking setup and visualization.\
    - Export analytics data to CSV or PDF.
    - Multi-website management demonstration.

4.2 Demo Data Used

- A controlled dataset of simulated user actions to ensure measurable, repeatable test conditions.

Requirement	**Verification Method**	**Success Criteria (Acceptable Range)**	**Failure Condition**
1. User Account Creation & Login	Perform registration and login tests using valid and invalid inputs.	User can register, receive confirmation, and log in successfully. Login	Account cannot be created or authenticated correctly.

		fails with wrong credentials.	
2. JavaScript Tracking Snippet Generation	Inspect generated snippet post-signup; verify it contains unique tracking ID and works when embedded in test HTML.	Snippet correctly identifies user and transmits events. Each user receives a unique snippet.	Duplicate snippet IDs or no event transmission detected.
3. Event Data Collection & Storage	Trigger multiple event types (clicks, views) and query AWS RDS tables.	All events appear in the database within <5 seconds of action. Data fields match schema.	Missing, duplicate, or delayed (>10s) event records.
4. Analytics Dashboard Visualization	Display charts with test data; compare against known database values.	Chart values align (+/-2%) with aggregated database totals.	Dashboard shows incorrect or stale data.
5. Insights Generation	Feed simulated data across time intervals; verify comparisons in dashboard output.	Correctly identifies trends (increase/decrease) between selected periods.	No insight generation or inaccurate comparisons.
6. User Experience Simplicity	Perform a usability test with 3–5 users. Count steps from sign-up to first dashboard view.	≤5 interactions needed; all buttons visually consistent with brand palette.	More than 5 required steps or inconsistent UI styling.
Stretch 1: Real-Time Analytics Updates	Simulate live events while viewing dashboard.	Dashboard auto-refreshes within 3 seconds of new data arrival.	Data only updates after manual refresh.
Stretch 2: Custom Event Tracking	Create and trigger a custom event via dashboard.	Event logs correctly under user-defined label and visualizes on dashboard.	Custom events not captured or misclassified.
Stretch 3: Data Export Functionality	Click export button; verify downloaded CSV/PDF accuracy.	Exported data matches visible dashboard metrics.	Export fails or contains inaccurate/missing entries.

Stretch 4: Multi-Website Management	Add multiple website profiles and test distinct snippets.	Data from each site appears in separate dashboard views with no cross-contamination.	Events from different sites merge incorrectly.

4.3 Verification Tools and Methods

- **Frontend Testing:**
  - *Tool:* Cypress or Playwright for UI automation.
  - *Goal:* Confirm visual components render, charts load, and buttons function.
- **Backend Testing:**
  - *Tool:* Postman for API endpoint testing; Pytest for unit/integration tests.
  - *Goal:* Validate API responses, authentication, and event processing.
- **Database Verification:**
  - *Tool:* SQL queries via AWS RDS console or pgAdmin.
  - *Goal:* Ensure event data integrity, uniqueness, and correct aggregation.
- **Performance Testing:**
  - *Tool:* Locust or JMeter.
  - *Goal:* Validate that the system can handle at least 500 concurrent event submissions with <2s latency.
- **Security & Validation Testing:**
  - Check password hashing (bcrypt).
  - Validate HTTPS data transmission and JWT authentication tokens.

4.4 Verification Summary

- **Success =** User can set up tracking, see accurate analytics, and interact with a reliable, responsive dashboard.

- **Failure =** Any step in that chain (setup -> data collection -> visualization) fails to produce the expected output or breaks defined criteria.