**מגישים:  
איתמר קראוס - 318304763**

**חי פדידה - 318302783**

**אייל כהן - 316148063**

**עידן אייזנברג - 316194315**

**אלעד גולדנברג - 315040519**

# Programmer Manual-ZEBRA

**Version 1.0**

**10.8.2024**

**Table of Contents**

[**1.**](#_heading=h.1fob9te) **Introduction 3**

[1.1](#_heading=h.3znysh7) Introduction to project 3

[1.2](#_heading=h.2et92p0) System Architecture Overview 3

[1.2.1](#_heading=h.tyjcwt) Key Components 3

[**2.**](#_heading=h.3dy6vkm) **Backend Key Functions 5**

[2.1](#_heading=h.1t3h5sf) Firebase Interaction 5

[2.1.1](#_heading=h.4d34og8) initialize\_firebase() 5

[2.1.2](#_heading=h.2s8eyo1) get\_firebase\_reference() 5

[2.2](#_heading=h.17dp8vu) Route Handling 5

[2.2.1](#_heading=h.3rdcrjn) Handle\_log() 5

[2.2.2](#_heading=h.26in1rg) get\_analytics() 5

[2.2.3](#_heading=h.lnxbz9) chat() 5

[2.3](#_heading=h.35nkun2) Application Initialization 6

[2.3.1](#_heading=h.1ksv4uv) main() 6

[**3.**](#_heading=h.44sinio) **Frontend Key Functions 7**

[3.1](#_heading=h.2jxsxqh) File Upload and Analytics Page 7

[3.1.1](#_heading=h.z337ya) displayJsonData(jsonData) 7

[3.1.2](#_heading=h.3j2qqm3) downloadCSV() 7

[3.1.3](#_heading=h.1y810tw) calculatePercentages(data) 7

[3.1.4](#_heading=h.4i7ojhp) filterJsonData(searchTerm) 7

[3.1.5](#_heading=h.2xcytpi) calculateAnalytics(jsonData) 7

[3.1.6](#_heading=h.1ci93xb) updateCharts(userActivity, documentUsage, tabActivity) 8

[3.1.7](#_heading=h.3whwml4) sendMessage() 8

[3.1.8](#_heading=h.2bn6wsx) displayChatMessages(messages) 8

[**4.**](#_heading=h.qsh70q) **FireBase Data Base 9**

## Introduction

### Introduction to project

This manual serves as a guide for developers working on the OnShape analyzer Flask Web Server project. It details the system architecture, backend and frontend functions, and the interaction with Firebase. The application processes JSON logs, performs analytics, and integrates a chatbot for user interactions.

### System Architecture Overview

*The project uses Flask for the backend logic, Firebase for data storage, and Ngrok for exposing the local server. It processes JSON logs, performs data analytics, and includes a chatbot for user interaction.*

#### Key Components

**Flask Backend**:

* **Overview**: Manages server-side logic, API routing, and interactions with Firebase. Handles JSON log processing, analytics, and chatbot functionalities.
* **Key Features**:
  + **Routing**: Defines routes for different API endpoints including log submissions, analytics retrieval, and chatbot interactions.
  + **Middleware**: Implements middleware for handling CORS, request validation, and error handling.
  + **Analytics Processing**: Processes incoming logs, performs analytics, and generates insights based on predefined metrics.
  + **Chatbot Integration**: Interfaces with a chatbot engine to provide conversational capabilities and handle user interactions.

**Firebase Database**:

* **Overview**: Stores JSON logs and analytics data. Chosen for its scalability and real-time capabilities.
* **Key Features**:
  + **Real-Time Updates**: Supports real-time synchronization of data, allowing for immediate updates and interactions.
  + **Scalability**: Scales seamlessly with increasing data volume and user load, ensuring consistent performance.
  + **Data Security**: Provides robust security rules and access controls to protect sensitive data.
  + **Data Structure**: Utilizes a flexible schema to handle varied data types and structures for logs and analytics.

**Chatbot Engine**:

* **Overview**: Powers the conversational interface of the application, providing dynamic responses based on user input.
* **Key Features**:
  + **Natural Language Processing (NLP)**: Analyzes user messages and generates appropriate responses using NLP techniques.
  + **Customizable Responses**: Allows for the creation of custom response patterns and behaviors tailored to user needs.
  + **Integration**: Seamlessly integrates with the Flask backend to handle user interactions and provide real-time responses.

**JSON Log Processor**:

* **Overview**: A component responsible for parsing and processing incoming JSON logs.
* **Key Features**:
  + **Log Parsing**: Extracts relevant information from raw JSON logs for further analysis.
  + **Error Handling**: Implements robust error handling to manage malformed or incomplete log data.
  + **Data Enrichment**: Adds context or additional metadata to logs before storing or processing them.
  + 

**Analytics Engine**:

* **Overview**: Analyzes processed data to generate insights and reports.
* **Key Features**:
  + **Data Aggregation**: Aggregates data from various sources to provide comprehensive insights.
  + **Reporting**: Generates reports and visualizations based on analytical results.
  + **Real-Time Processing**: Provides real-time analytics to support dynamic decision-making.

## Backend Key Functions

### Firebase Interaction

#### initialize\_firebase()

 **Input**: None

 **Output**: Firebase app instance

 **Description**: Initializes the Firebase app with the provided credentials. Handles connection errors and ensures proper setup.

#### get\_firebase\_reference()

 **Input**: None

 **Output**: Firebase database reference

 **Description**: Retrieves a reference to the Firebase database for performing read and write operations.

### Route Handling

#### Handle\_log()

 **Input**: JSON object from the POST request

 **Output**: JSON response with status message

 **Description**: Receives JSON logs, processes them, and stores them in Firebase. Logs errors if any occur during processing.

#### get\_analytics()

 **Input**: None

 **Output**: JSON response with analytics data

 **Description**: Fetches and processes analytics data from Firebase. Provides a summary of log data and insights based on predefined metrics.

#### chat()

* **Input**: JSON object with user message
* **Output**: JSON response with chatbot response
* **Description**: Handles user messages, generates responses based on predefined patterns, and returns the response to the user.

### Application Initialization

#### main()

 **Input**: None

 **Output**: Logs a message indicating that the server is running

 **Description**: Initializes the Flask app, sets up routes, and starts the server. Uses Ngrok to create a public URL for the server.

## Frontend Key Functions

### File Upload and Analytics Page

#### displayJsonData(jsonData)

 **Input:** jsonData (object) - The JSON data to be displayed on the page.

 **Output:** Updates the page with the JSON data.

 **Description:** Processes and displays JSON data in a structured format on the page. Stores and filters data based on user input for further actions.

#### downloadCSV()

 **Input:** None.

 **Output:** Downloads a CSV file containing user activity, document usage, and tab activity data.

 **Description:** Constructs and triggers a download for a CSV file based on the current data in the analytics section. Converts JSON data into a CSV format and initiates a file download.

#### calculatePercentages(data)

 **Input:** data (array) - Array of data entries for which percentages need to be calculated.

 **Output:** Returns an array of percentage values for each data entry.

 **Description:** Calculates the percentage of each data entry relative to the total. Used in table rendering to show relative proportions of data.

#### filterJsonData(searchTerm)

 **Input:** searchTerm (string) - The term used to filter the JSON data.

 **Output:** Filters and updates the table with data matching the search term.

 **Description:** Filters the displayed JSON data based on the search term input by the user. Updates the table with results that match the search criteria.

#### calculateAnalytics(jsonData)

 **Input:** jsonData (object) - Raw JSON data to be processed for analytics.

 **Output:** Updates local storage and UI elements with aggregated data.

 **Description:** Processes raw JSON data to compute various analytics metrics, including user activity, document usage, and tab activity. Updates the UI with the computed data and stores it in local storage.

##### updateTable(tableId, data)

 **Input:** tableId (string), data (array) - The ID of the table to update and the data to populate.

 **Output:** Updates the specified table with new data entries.

 **Description:** Iterates over the provided data and appends rows to the table identified by tableId. Refreshes the table view with the new data.

#### updateCharts(userActivity, documentUsage, tabActivity)

 **Input:** userActivity (object), documentUsage (object), tabActivity (object) - Data for updating different charts.

 **Output:** Updates the charts on the page with new data.

 **Description:** Uses Chart.js to refresh charts with new data for user activity, document usage, and tab activity. Updates visualizations to reflect the latest analytics.

#### sendMessage()

* **Input:** None.
* **Output:** Updates chat with user and chatbot messages.
* **Description:** Handles the submission of user messages, sends them to the backend, and receives responses. Appends both user and chatbot messages to the chat area and scrolls to the latest message.

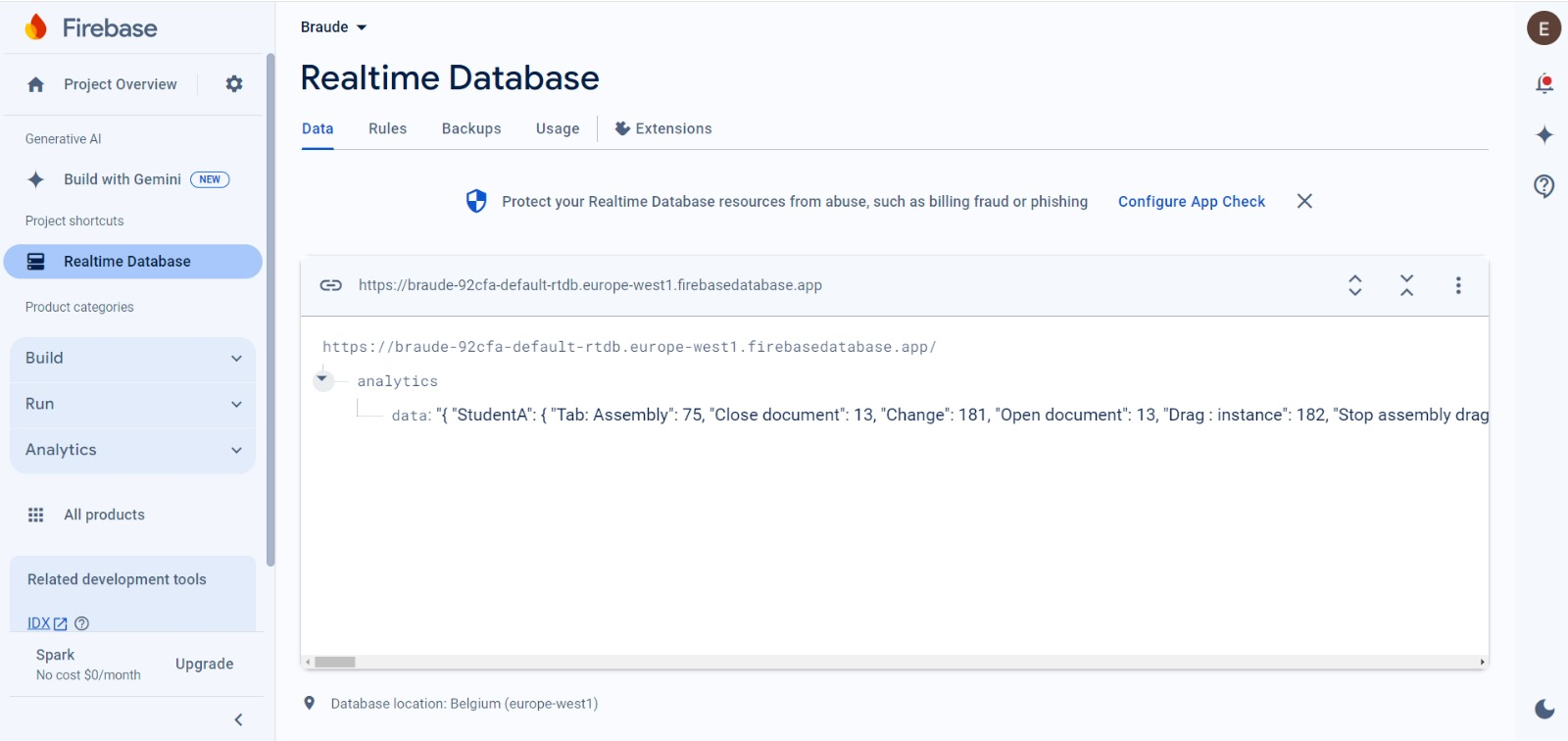
#### displayChatMessages(messages)

 **Input:** messages (array) - Array of message objects to be displayed in the chat area.

 **Output:** Renders chat messages in the chat interface.

 **Description:** Processes and displays an array of chat messages in the chat container. Formats messages and ensures proper display in the chat interface.

## FireBase Data Base

****