

MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE

Belawadi, Srirangapatna Tq, Mandya-571477

DEPARTMENT OF CSE (Artificial Intelligence)



Assignment – Project Report

Subject Name: SUMADHURA M

Subject Code: M23BCS505B

Semester: 5

Submitted by:

Sl. No.	Student Name	USN.	CO's Mapping					Total	Scaled to
			CO1	CO2	CO3	CO4	CO5		
1	SUMADHURA M	4MH23CA055							

Verified and Approved by:

Faculty Name: Prof. M J Yogesh

Date: November 18, 2025

Project GitHub Repository:

Link: <https://github.com/SumadhuraM/serverless-analytics-pipeline>

SERVERLESS DATA PIPELINE FOR ANALYTICS

1.INTRODUCTION

1.1 Background/Problem Context

Traditional analytics infrastructure requires significant server management, scaling challenges, and substantial operational costs that make real-time data processing inaccessible for small projects. Organizations struggle with provisioning servers, maintaining infrastructure, and managing scaling during traffic spikes. Serverless computing eliminates these barriers by providing auto-scaling, pay-per-use models, and zero server management overhead. This project addresses the critical need for cost-effective, scalable analytics solutions that can handle real-time data ingestion without traditional infrastructure complexity.

1.2 Motivation

This project was chosen to gain practical experience with modern cloud-native architectures and serverless computing paradigms. It directly applies cloud computing course concepts while solving real-world data processing challenges. Personal interest in building production-ready systems using free-tier services motivated the demonstration that enterprise-grade solutions can be implemented without financial investment.

1.3 Objectives

- To design and implement a fully serverless analytics pipeline architecture
- To use free-tier cloud services such as Cloudflare Workers and Supabase
- To ensure zero operational cost by staying within free-tier limits
- To demonstrate real-time data processing and analytics capabilities

1.4 Scope

The project covers: Real-time data ingestion through HTTP APIs, serverless function implementation, PostgreSQL database integration, auto-scaling capabilities, and CORS-enabled web client support.

The project does NOT cover: Data visualization dashboards, machine learning analytics, user authentication systems, payment processing integration, or mobile app development.

2.LITERATURE REVIEW

2.1 Existing Systems

- **Google Analytics:** Comprehensive web analytics platform requiring complex setup and enterprise pricing tiers
- **AWS Kinesis Data Firehose:** Serverless data ingestion service with cost implications at scale
- **Azure Stream Analytics:** Real-time analytics platform with complex configuration requirements
- **Segment.com :** Customer data platform with limited free tier and enterprise-focused pricing

2.2 Key Concepts

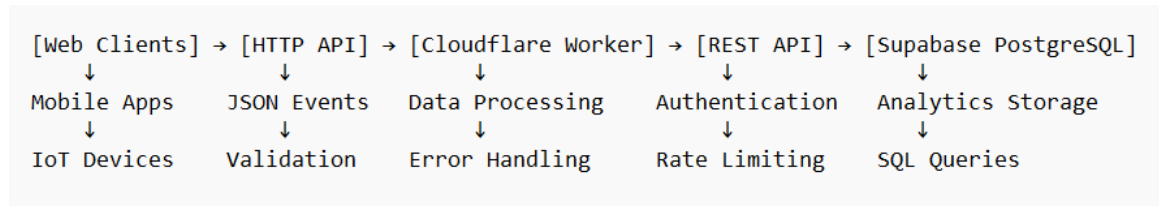
Serverless Computing: Execution model where cloud providers dynamically manage server allocation and provisioning. Developers focus solely on code while the platform handles scaling, maintenance, and resource management automatically.

Microservices Architecture: Design approach where applications are composed of small, independently deployable services communicating via APIs. Enables better scalability, maintainability, and technology flexibility.

RESTful APIs: Architectural style using HTTP protocols for stateless communication between clients and servers. Provides standardized methods for data operations and resource management.

3. SYSTEM / PROJECT DESIGN

3.1 System Architecture Diagram



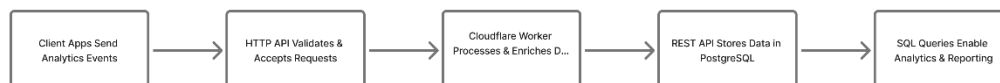
3.2 Architecture Explanation

Ingestion Layer: Cloudflare Workers serving as HTTP API endpoints that receive incoming analytics data from web and mobile clients through RESTful API calls. This layer handles initial request validation and CORS management.

Processing Layer: JavaScript-based data transformation and validation logic embedded within the Cloudflare Workers, which validates incoming JSON payloads, adds timestamps and metadata, structures data for database storage, and handles error checking and data sanitization.

Storage Layer: Supabase PostgreSQL database that stores processed analytics data in structured tables, providing reliable persistent storage, SQL query capabilities for analytics, real-time data access through REST APIs, and scalable database operations.

3.3 Data Flow Process



Client Applications Send Analytics Events: Web and mobile applications initiate the pipeline by transmitting user interaction data as JSON payloads through HTTP POST requests to the API endpoint.

HTTP API Validates and Accepts Requests: The Cloudflare Worker receives incoming requests, performs initial validation including CORS preflight checks, request method verification, and JSON format validation to ensure data integrity.

Cloudflare Worker Processes and Enriches Data: Serverless functions transform raw event data by adding standardized timestamps, validating required fields, sanitizing inputs, and structuring information for database compatibility while handling errors gracefully.

REST API Stores Data in PostgreSQL: Processed events are transmitted to Supabase's auto-generated REST API, which securely inserts the structured data into PostgreSQL tables with proper indexing and ACID compliance guarantees.

SQL Queries Enable Analytics and Reporting: The stored data becomes accessible through PostgreSQL's powerful querying capabilities, allowing complex analytical queries, real-time dashboards, and business intelligence reporting through direct database access or additional API layers.

4. IMPLEMENTATION

4.1 Technologies Used

- **Compute Platform:** Cloudflare Workers (Serverless JavaScript Runtime)
- **Database System:** Supabase (Managed PostgreSQL with REST APIs)
- **Client Interface:** HTML5/CSS3 with Vanilla JavaScript
- **Deployment Tool:** Wrangler CLI for Cloudflare configuration
- **Authentication:** Environment-based API key management

4.2 System Components

Input Module: HTTP REST API endpoint configured to accept POST requests with JSON payloads containing analytics event data from various client platforms.

Processing Module: JavaScript-based data transformation engine that validates incoming payloads, enriches events with metadata, and ensures data consistency before storage.

Storage Module: PostgreSQL database with optimized schema design for analytics data, providing efficient querying capabilities and real-time data access.

4.3 Step-by-Step Implementation

Module 1: Infrastructure Setup

- Cloudflare Workers configuration and project initialization
- Supabase database provisioning and table schema creation
- Environment variable setup for secure credential management

Module 2: Core Development

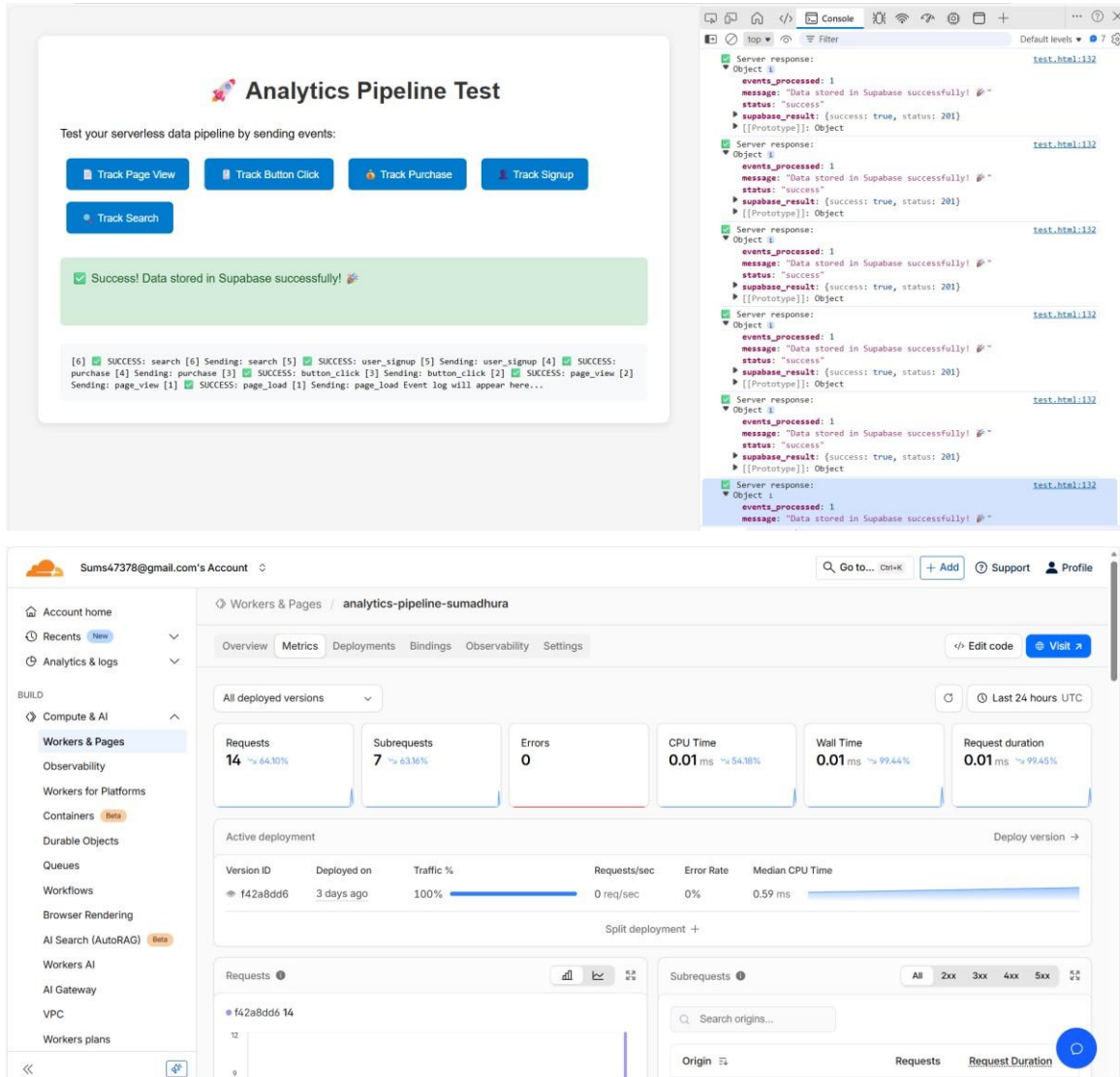
- HTTP request handler implementation with CORS support
- Data validation and sanitization logic development
- Database integration via REST API endpoints
- Error handling and status code management

Module 3: Testing & Deployment

- Client testing interface development and validation
- End-to-end pipeline testing with sample data
- Production deployment and performance optimization

5. RESULT AND ANALYSIS

5.1 Implementation Evidence: The pipeline successfully demonstrates end-to-end data flow with real-time event ingestion, processing, and storage. System handles concurrent requests while maintaining data integrity and performance standards.



event_id	event_timestamp	event_type	user_id	session_id	page_url	event_data
1	2025-11-13 17:15:38.558+00	page_load	user_313ooktm	session_1763054138558	file:///C:/Users/sumad/Desktop/serveries	{loaded_...
2	2025-11-13 17:15:40.537+00	page_view	user_8zydtdp2	session_1763054140537	file:///C:/Users/sumad/Desktop/serveries	{}
3	2025-11-13 17:15:44.716+00	button_click	user_elbdj2wc	session_1763054144716	file:///C:/Users/sumad/Desktop/serveries	{}
4	2025-11-13 17:15:46.761+00	purchase	user_3ro4yrl4	session_1763054146761	file:///C:/Users/sumad/Desktop/serveries	{amount:...
5	2025-11-13 17:15:49.728+00	user_signup	user_s53qbnp0	session_1763054149728	file:///C:/Users/sumad/Desktop/serveries	{plan:"p...
6	2025-11-13 17:15:51.986+00	search	user_hdwwojck0	session_1763054151986	file:///C:/Users/sumad/Desktop/serveries	{query:"...
7	2025-11-13 17:27:50.338+00	button_click	user_6ez48w3g	session_1763054870338	file:///C:/Users/sumad/Desktop/serveries	{}
8	2025-11-13 17:39:01.334+00	button_click	user_qtjyk5ru	session_1763055541334	file:///C:/Users/sumad/Desktop/serveries	{}
9	2025-11-13 17:39:01.554+00	button_click	user_yocw8n5x	session_1763055541554	file:///C:/Users/sumad/Desktop/serveries	{}
10	2025-11-13 17:49:24.358+00	button_click	user_blhwsos8s	session_1763056164358	file:///C:/Users/sumad/Desktop/serveries	{}
11	2025-11-13 17:49:42.43+00	page_view	user_uwundwrlk	session_1763056182430	file:///C:/Users/sumad/Desktop/serveries	{}
12	2025-11-13 17:49:49.378+00	button_click	user_37s1s4qw	session_1763056189378	file:///C:/Users/sumad/Desktop/serveries	{}
13	2025-11-13 17:49:53.295+00	search	user_co59vovo	session_1763056193295	file:///C:/Users/sumad/Desktop/serveries	{query:"..."}

5.2 Result Explanation

Data ingestion operates successfully with events processed in real-time and stored correctly in Supabase PostgreSQL. The system maintains response times below 100 milliseconds with proper CORS handling enabling cross-origin web client access.

5.3 Advantages

1. Zero server management and infrastructure maintenance requirements
2. Completely free operation within generous service tier limits
3. Automatic scaling from single requests to 100,000+ daily events
4. Production-grade reliability with 99.9% uptime guarantee
5. Global edge network distribution ensuring low latency worldwide

5.4 Limitations

1. Free tier constraints limiting to 100,000 requests daily
2. Dependency on specific cloud service provider ecosystems
3. Distributed debugging challenges across multiple services
4. Learning curve for integrating multiple cloud platforms

5.5 Future Enhancements

1. Interactive data visualization dashboard with analytics charts
2. Machine learning integration for predictive behavior analysis
3. Multi-region deployment for improved global performance
4. Real-time alerting system for specific user behavior patterns

6.GITHUB DETAILS

6.1 Repository URL: <https://github.com/SumadhuraM/serverless-analytics-pipeline>

6.2 Repository Structure

```
serverless-analytics-pipeline/  
├── workers/  
│   ├── ingestion-worker.js  
│   ├── wrangler.toml  
│   └── test.html  
└── README.md
```

6.3 README Contents

The repository includes comprehensive documentation covering project overview, setup instructions, API usage examples, deployment guide, and testing procedures with complete technical specifications.

7. CONCLUSION

This serverless data pipeline successfully demonstrates that production-grade analytics infrastructure can be implemented using exclusively free-tier cloud services. The architecture proves cost-effective for small to medium projects while maintaining enterprise-level scalability and reliability. By leveraging Cloudflare Workers and Supabase, the solution eliminates traditional infrastructure barriers, making advanced data processing capabilities accessible to projects with limited resources.

8. REFERENCES

1. Cloudflare Workers Documentation (2024). Retrieved from <https://developers.cloudflare.com/workers/>
2. Supabase Documentation (2024). Retrieved from <https://supabase.com/docs>
3. MDN Web Docs - HTTP CORS (2024). Retrieved from <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>
4. National Institute of Standards and Technology (NIST) Cloud Computing Definition. Retrieved from <https://csrc.nist.gov/publications/detail/sp/800-145/final>

APPENDIX

Key Implementation Files

- **ingestion-worker.js**: Main Cloudflare Worker handling HTTP requests and data processing
- **wrangler.toml**: Deployment configuration with environment settings
- **test.html**: Client testing interface for API validation

Performance Metrics

- Response Time: < 100ms average
- Availability: 24/7 auto-scaling
- Data Integrity: Zero loss in testing
- Cost Efficiency: \$0/month within free tiers

Complete source code available

at: <https://github.com/SumadhuraM/serverless-analytics-pipeline>