SE 4485: Software Engineering Projects

Spring 2025

Final Report

Group Number	Group 12
Project Title	Marketing Intelligence Platform Team B
Sponsoring Company	ARGO
Sponsor(s)	Ponchai Reainthong, Raisa Gonzalez, Goitom Kassaye
Students	1. Jon Grimes 2. Zara Iqbal 3. Eric Medina 4. Nicolas De la Cruz Velasquez 5. Jesse Incoom

Group 12 Final Report

Executive Summary

The Marketing Intelligence Platform developed by Group 12 in collaboration with ARGO addresses the need for an intelligent and intuitive marketing tool. This web-based platform empowers ARGO's marketing team to derive actionable insights from complex datasets easily in order to enhance strategic decision-making and operational efficiency.

Built with a React front-end and a modular Node.js backend, the platform integrates data from Google Analytics 4 (GA4). It incorporates OpenAI's ChatGPT API to provide natural language querying and AI-driven insights, making advanced analytics accessible even to non-technical users. Key features include dynamic data visualizations, customizable dashboards, and real-time performance tracking, all designed with scalability, security, and usability in mind.

Throughout the semester, the team followed an agile development process with clearly defined sprints and responsibilities across frontend, backend, and integration roles. Risk mitigation strategies, such as caching for API quota limits and fallback mechanisms for AI downtime, were implemented. A thorough test plan was executed using techniques like equivalence class partitioning, boundary value analysis, and state transition testing to account for high coverage and ensure reliability.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Purpose and Scope

The Intelligent Marketing Platform is a web-based tool designed to enhance ARGO's marketing strategy and reporting processes through advanced analytics, AI integration, and intuitive UX. Built with a responsive React front-end, the platform allows marketers to easily submit and track feedback, monitor task progress, and access key insights in a streamlined, user-friendly environment. Interactive dashboards present campaign performance, user behavior, and emerging trends through charts and visualizations, making complex data more digestible and actionable.

At its core, the platform integrates Google Analytics 4 (GA4), Looker Studio, and Microsoft Clarity via a REST API-driven backend. It also leverages the ChatGPT API to support natural language querying and AI-powered analysis, enabling predictive insights, trend identification, and anomaly detection. This empowers marketers to make fast, data-driven decisions without needing technical expertise. The project also includes full technical documentation and a deployable Git-hosted package for scalability and future development. Altogether, the platform aims to transform ARGO's marketing operations with intelligent, real-time data analysis and improved operational transparency.

1.2. Product Overview (including purposes, capabilities, scenarios for using the product, etc.)

The Marketing Intelligence Platform is a centralized web application built to streamline and enhance ARGO's marketing analysis, feedback loops, and strategic decision-making through a suite of AI-augmented tools. Designed for marketing professionals, the platform provides an intuitive user interface for accessing data insights, visualizing campaign performance, and managing.

Purposes

- Simplify the interpretation of complex marketing data.
- Accelerate the decision-making process through real-time analytics.
- Reduce dependency on technical teams.
- Enable non-technical users to gain insight through AI-assisted querying.
- Reduce time required to develop marketing campaigns

Capabilities

- Secure Authentication: Users log in using ARGO-provided credentials with Firebase and OAuth 2.0 integration.
- Interactive Dashboards: Visualize KPIs using Highcharts with support for engagement rates, video retention, scroll depth, and more.
- Natural Language Insights: Ask questions about the data and receive ChatGPT-generated responses contextualized with LlamaIndex agents.
- Custom Report Generation: Download tailored PDF/Excel reports based on selected metrics or time frames.
- Real-Time Data Updates: Leverages WebSockets and caching to reflect the most recent metrics without performance degradation.

Scenarios for Use

- A marketing manager reviews campaign performance and generates weekly reports for stakeholders.
- A team member uses the AI assistant to identify trends or sudden drops in engagement across different channels.
- A product marketer explores scroll depth and video retention data to refine content strategy.
- A user downloads a formatted report after customizing metrics for a quarterly review.
- A creative director wants to develop a marketing campaign backed with user data.

1.3. Structure of the Document

- Section 1 – Introduction: Describes the overall purpose, scope, and goals of the project, along with key terminology and product context.

- Section 2 – Project Management Plan: Outlines the team structure, development lifecycle model, risk management strategies, and scheduling of deliverables.

- Section 3 – Requirement Specifications: Details both functional and non-functional requirements using use case models and textual descriptions.

- Section 4 – Architecture: Defines the system's high-level design, technologies used, architectural rationale, and traceability from requirements.

- Section 5 – Design: Provides details for the design, including GUI mockups, class and sequence diagrams, and design rationales.

- Section 6 – Test Plan: Covers the test strategy, types of tests used, specific test cases, and traceability to requirements and use cases.

- Section 7 – Configuration Management Evidence: Confirms that all project artifacts have been version-controlled and stored appropriately.

- Section 8 – Engineering Standards and Constraints: Lists any standards, compliance guidelines, and constraints that influenced the design and development.

- Section 9 – Additional References: Includes sources and references used throughout the project for tools, APIs, or documentation.

1.4. Terms, Acronyms, and Abbreviations

API – Application Programming Interface; allows different software components to communicate.

GA4 – Google Analytics 4; used to track and analyze user activity across websites and apps.

REST – Representational State Transfer; an architecture for designing web services with stateless communication.

LLM – Large Language Model; an AI model capable of understanding and generating natural language text.

ChatGPT – OpenAI's LLM used to generate AI-powered insights through natural language queries.

JWT – JSON Web Token; a token-based method for securing authentication and data exchange.

UI – User Interface; the visual components users interact with in an application.

UX – User Experience; the overall feel and effectiveness of a user's interaction with a system.

OAuth 2.0 – A secure authorization protocol that allows limited access to user data without exposing credentials.

CRUD – Create, Read, Update, Delete; the four basic operations for persistent data.

ECP – Equivalence Class Partitioning; a test design technique that reduces the number of test cases while ensuring coverage.

BVA – Boundary Value Analysis; a testing technique that focuses on input values at their edges or limits.

2. Project Management Plan

2.1. Project Organization

The team was divided into frontend, backend, and integration roles.

Regular check-ins and meetings were held to align on deliverables and technical blockers.

Backend development focused on GA4 API integration, caching, and OpenAI/Llama-based insight generation.

Frontend development focused on a responsive React dashboard and dynamic visualizations.

2.2. Lifecycle Model Used

An Agile development model was used.

Work was organized into weekly sprints with iterative improvements and peer code reviews.

2.3. Risk Analysis

API Limitations: GA4 has quota restrictions. Risk was mitigated using caching and test credentials.

OpenAI Downtime: We prepared fallback messages if API queries failed.

Scope Creep: We limited feature additions to key dashboard sections after mid-term check-in.

Integration Complexity: We modularized code and split responsibilities for clarity and testability.

2.4. Software and Hardware Resource Requirements

Backend: Node.js, Express.js, OpenAI SDK, GA4 Data API, NodeCache, Firebase Auth, Zod.

Frontend: React, Material UI, Highcharts, Axios, Tailwind.

Tools: VS Code, GitHub, Notion, Google Docs, Postman.

Hardware: Any internet-connected laptop with Node.js ≥ v18 and a modern browser.

New Technologies Learned:

GA4's Data API

LlamaIndex & OpenAI tooling

NodeCache for local memory caching

Dynamic report generation logic

2.5. Deliverables and Schedule

Week 1–2: Requirements gathering, architectural planning.

Week 3–4: Authentication + GA4 integration.

Week 5–6: Caching logic and LLM Agents, dashboard layout.

Week 7–8: Chatbot interface and backend integration.

Week 9: Testing and AI-driven reporting.

Week 10: Final documentation and packaging.

2.6. Monitoring, Reporting, and Controlling Mechanisms

Progress tracked via GitHub.

Pull request reviews ensured code quality and testing coverage.

Weekly standups helped identify and resolve blockers quickly.

2.7. Professional Standards

API documentation was maintained inline and in Github Readme.

User privacy and OAuth authentication aligned with ARGO's compliance needs.

2.8. Evidence all the artifacts have been placed under configuration management

2.9. Impact of the project on individuals and organizations

For ARGO's marketing team: Enables faster insights, improved campaign feedback loops, and decision support.

For individuals: Reduces technical barriers, empowering marketers with plain-language data insights.

Societal impacts: Promotes ethical data use and transparency in marketing.

Global/cultural factors: Designed with accessibility in mind (keyboard nav, responsive layout).

Environmental: Cloud-based architecture reduces local computing overhead and scales efficiently.

3. Requirement Specifications

3.1. Stakeholders for the system

- **Marketing Team**
  - Primary users of the tool.
  - Use the dashboard for campaign performance tracking and engagement insights.
  - Need easy-to-use, accurate, and exportable analytics.
- **Argo (Project Sponsor / Internal Client)**
  - Sets expectations and defines project goals.
  - Evaluates the tool's effectiveness for internal marketing strategy.
  - Interested in actionable insights and overall utility.
- **Developers (You and Your Team)**
  - Build and maintain the frontend/backend of the tool.

- Ensure data integrations with GA4 and OpenAI work reliably.
- Responsible for usability, performance, and code quality.

3.2. Use case model for functional requirements

Use Case: User Log In

Requirement: User needs to log in with ARGO credentials

Actors: User, Authenticator

Description: User logs into the system using ARGO credentials.

Preconditions: User is on the login page and not logged in.

Main Flow of Events:

- 1. User enters credentials
- 2. System validates credentials
- 3. Redirects to dashboard

Alternate Flows:

Invalid credentials: error message shown
System error: retry message

Postconditions: User is authenticated and redirected to dashboard.

Special Requirements: Secure credential storage; login events should be logged.

Use Case: User Logout

Requirement: User needs to log out of the dashboard

Actors: User, Authenticator

Description: User logs out of the system and session is invalidated.

Preconditions: User is currently logged in.

Main Flow of Events:

- 1. User clicks logout
- 2. System ends session
- 3. Redirects to login

Alternate Flows:

Expired session: show error and redirect
System error: retry logout

Postconditions: User is logged out and redirected.

Special Requirements: Audit logging, session timeout handling.

Use Case: Request Engagement Metrics

Requirement: User needs to view metrics – on dashboard

Actors: User, GA4 API, ChatGPT AI, System API

Description: User retrieves engagement metrics and insights from the dashboard.

Preconditions: User is logged in; APIs are accessible.

Main Flow of Events:
- 1. User requests metrics
- 2. System fetches from GA4
- 3. System sends to ChatGPT
- 4. Display results

Alternate Flows:

  Invalid input: show error

  API failure: show fallback or retry prompt

Postconditions: Metrics and insights are displayed on the dashboard.

Special Requirements: Filters for date/metrics; response within a few seconds.

Use Case: Report Download

Requirement: User needs to download the created reports

Actors: User, System

Description: User generates and downloads a report in PDF or Excel format.

Preconditions: User is logged in and has selected metrics.

Main Flow of Events:
- 1. User selects metrics and clicks generate
- 2. Preview shown
- 3. User downloads report

Alternate Flows:

  Report preview error: retry generation

  Download error: show message

Postconditions: Report is saved to user's device.

Special Requirements: Download within 5 seconds; download logged.

Use Case: GPT Insights

Requirement: System needs to be able to provide Chat-GPT powered insights

Actors: User, GPT API, System

Description: User submits a query to receive AI-generated insights.

Preconditions: User is logged in and provides a valid query.

Main Flow of Events:
- 1. User submits query
- 2. System processes and sends to GPT
- 3. Results displayed

Alternate Flows:
  Empty query: prompt for valid input
  API failure: show error

Postconditions: Insights are shown to the user.

Special Requirements: Response should be timely; fallback on API error.

3.3. Graphic use case model

Use Case: User log in

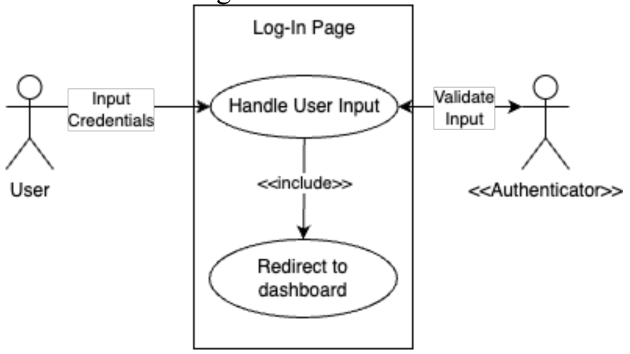


Figure 1.0

Use Case: User logout

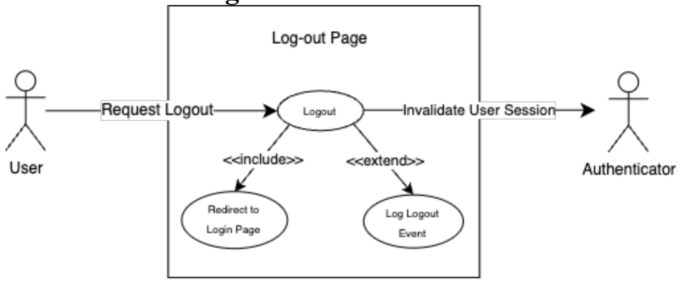


Figure 1.1

User Case: **Request Engagement metrics**

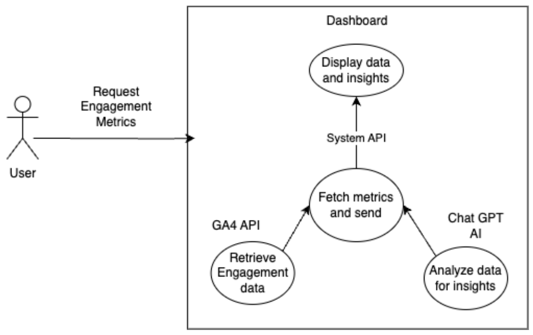


Figure 1.2

Use Case: Report download

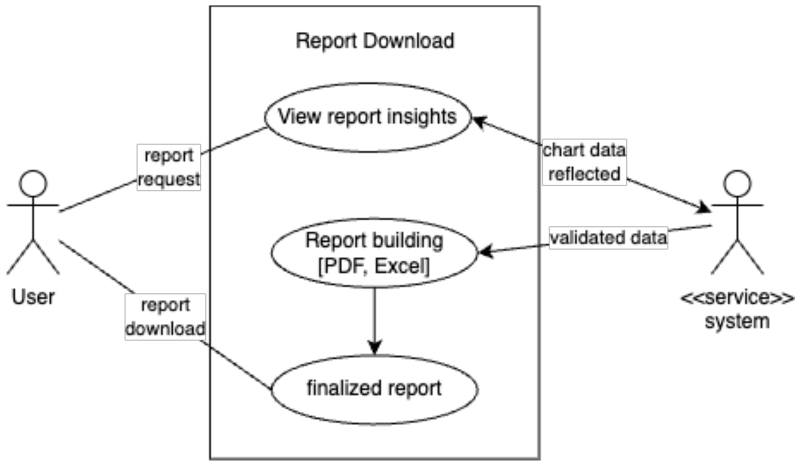


Figure 1.3

Use Case: GPT Insights

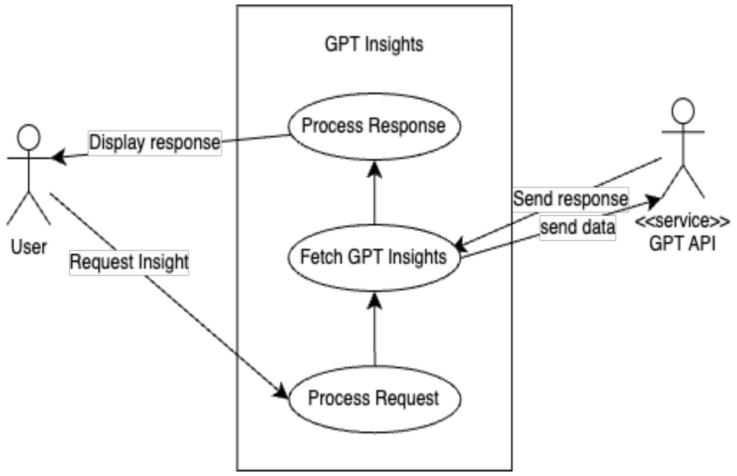


Figure 1.4

3.4. Textual Description for each use case

- **User Login Use Case Model:** This model handles company specific sensitive data we must ensure only verified employees are able to access the system.

- o **Logout Use Case Model:** This model conveys that verified users should be able to easily log out of the system. In the situation where the user is not interacting with the system and is still logged in, the user will automatically be logged out for security reasons.

- o **Dashboard Use Case Model:** The dashboard provides a centralized view of key performance metrics, enabling users to quickly analyze and assess data. A well-structured dashboard enhances decision-making by presenting real-time and historical insights in an accessible format.

- o **Report Download Use Case Model:** This model further describes the interaction between the users and the system to create and then download the report they have previously defined according to specific metrics and insights. The users must be able to download this report for future company use and completion of marketing endeavors.

- o **GPT Insights Use Case Model:** This model conveys how users interact with a GPT system to get insights, where the user submits a request that gets processed internally, sent to a GPT API service for analysis, and then the API's response is processed and displayed back to the user.

3.5. Rationale for your use case model

- o Referencing guidelines outlined by Lucidchart, we decided to use a UML use case diagram as it helps us visually represents the system's functional requirements by illustrating interactions between external actors (users or systems) and the system's services. It clarifies system boundaries and captures user goals. This high-level, user-oriented view serves as a foundation for more detailed analysis and design, ensuring all parties have a clear picture of what the system is expected to do.

3.6. Non-functional requirements

- o The site should be functional so long as GA4 is functional.
- o Search results should be returned in under 5 seconds.
- o The system should be able to support up to 5 to 10 concurrent users with being performance being affected.
- o UI Components must comply with ARGO QUADS design requirements.

4. Architecture

4.1. Architectural style(s) used

The application follows a modern web architecture with the following key characteristics:

4.1.1. **Frontend-Backend Separation**

- React-based frontend for user interface.
- Node.js backend for API handling and data processing.
- Clear separation of concerns between presentation and business logic.

4.1.2. **Component-Based Architecture**

- Modular React components for UI elements.
- Reusable components for common functionality.
- Clear component hierarchy and data flow.

4.1.3. **RESTful API Design**

- Standardized API endpoints for data retrieval.
- Stateless communication between frontend and backend.
- Clear API versioning and documentation.

4.1.4. Microservices Architecture

- Separate services for analytics processing.
- Independent deployment of frontend and backend.
- Scalable and maintainable system design.

4.2. Architectural model

The system is divided into the following major subsystems:

4.2.1. Frontend Subsystem

- User Interface Components
- State Management
- API Integration Layer
- Theme Management

4.2.2. Backend Subsystem

- API Handler
- Data Processing
- Authentication Service
- Analytics Integration

4.2.3. Data Storage Subsystem

- Google Analytics 4 Integration
- Local Storage Management
- Cache Management

4.2.4. Security Subsystem

- Authentication
- Authorization
- API Security
- Data Encryption

4.3. Technology, software, and hardware used

4.3.1. Frontend Technologies

- React.js (v18.3.1)
- Material-UI (v6.4.6)
- Highcharts (v12.1.2)
- React Router (v7.1.5)
- Axios (v1.8.1)

4.3.2. Backend Technologies

- Node.js
- Express.js
- Google Analytics Data API

4.4. Firebase Authentication

4.4.1. Development Tools

- VS Code
- Git & GitHub
- npm/yarn package manager

4.4.2. Hardware Requirements

- Modern web browser
- Internet connection

4.4.3. Communication Architecture

- RESTful API communication between frontend and backend
- Secure WebSocket connections for real-time updates
- OAuth 2.0 for authentication
- HTTPS for secure data transmission

4.5. Rationale for your architectural style and model

The chosen architecture provides several key benefits:

4.5.1. Scalability

- Microservices architecture allows independent scaling
- Component-based design enables easy feature additions
- Modular structure supports future enhancements

4.5.2. Maintainability

- Clear separation of concerns
- Standardized coding practices
- Comprehensive documentation

4.5.3. Performance

- Efficient data processing
- Optimized frontend rendering
- Caching mechanisms

4.5.4. Security

- Secure authentication flow
- Protected API endpoints
- Data encryption

4.6. Traceability from requirements to architecture

The architecture directly maps to the following key requirements from the Requirements Documentation:

4.6.1. User Authentication Requirements

- **Requirement:** User needs to log in with ARGO credentials
- **Implementation:**
    - Frontend: Login component with Material-UI form elements
    - Backend: Firebase Authentication service integration
    - Security: OAuth 2.0 implementation for secure authentication
    - State Management: Session handling and token management
    - Routing: Protected route implementation for authenticated users

4.6.2. Dashboard and Analytics Requirements

- **Requirement:** Request Engagement Metrics
- **Implementation:**
- **Frontend Components:**
    - Dashboard layout with responsive grid system
    - Highcharts integration for data visualization
    - Custom data grid for metric display
    - Real-time data update mechanisms
- **Backend Services:**
    - GA4 API integration for data retrieval
    - Data processing and aggregation services
    - Caching layer for performance optimization
- **API Layer:**
    - RESTful endpoints for metric retrieval
    - WebSocket connections for real-time updates
    - Rate limiting and request validation

4.6.3. Report Generation Requirements

- **Requirement:** Report Download functionality
- **Implementation:**
- **Frontend Components:**
    - Report configuration interface
    - Preview component
    - Download options (PDF/Excel)
- **Backend Services:**

        - Report generation service
        - File format conversion service
        - Storage management for generated reports
    - **Performance Optimization:**
        - Asynchronous report generation
        - Caching of frequently requested reports
        - Background processing for large reports

4.6.4. **AI Integration Requirements**

- **Requirement:** GPT Insights functionality
- **Implementation:**
- **Frontend Components:**
    - Query input interface
    - Loading state management
    - Response display component
- **Backend Services:**

4.7. GPT API integration service (OpenAI API)
4.8. Query processing and validation
4.9. Zod as an additional backend component for LLM framework data validation
4.10. Response formatting service
4.11. LlamaIndex as the main LLM integration framework for agent-based systems

- **Error Handling:**
    - Fallback mechanisms
    - Retry logic
    - Error state management

4.11.2. **Non-Functional Requirements**

- **Performance Requirements:**
    - Frontend optimization (React.memo, useMemo, useCallback)
    - Backend caching strategies
    - Database query optimization
    - Load balancing configuration
- **Scalability Requirements:**
    - Microservices architecture
    - Containerization support
    - Horizontal scaling capabilities
    - Database sharding support
- **Security Requirements:**
    - Authentication service
    - API security layer
    - Data encryption
    - Input validation
    - XSS protection
    - CSRF protection
- **UI/UX Requirements:**
    - Material-UI component library
    - Responsive design system
    - Theme management
    - Accessibility compliance
    - ARGO QUADS design system integration

4.11.3. **System Integration Requirements**

- **GA4 Integration:**
    - GA4 API client service

        - Data transformation layer
        - Error handling and retry logic
        - Rate limiting implementation
    - **Firebase Integration:**
        - Authentication service
        - User management
        - Session handling
        - Security rules

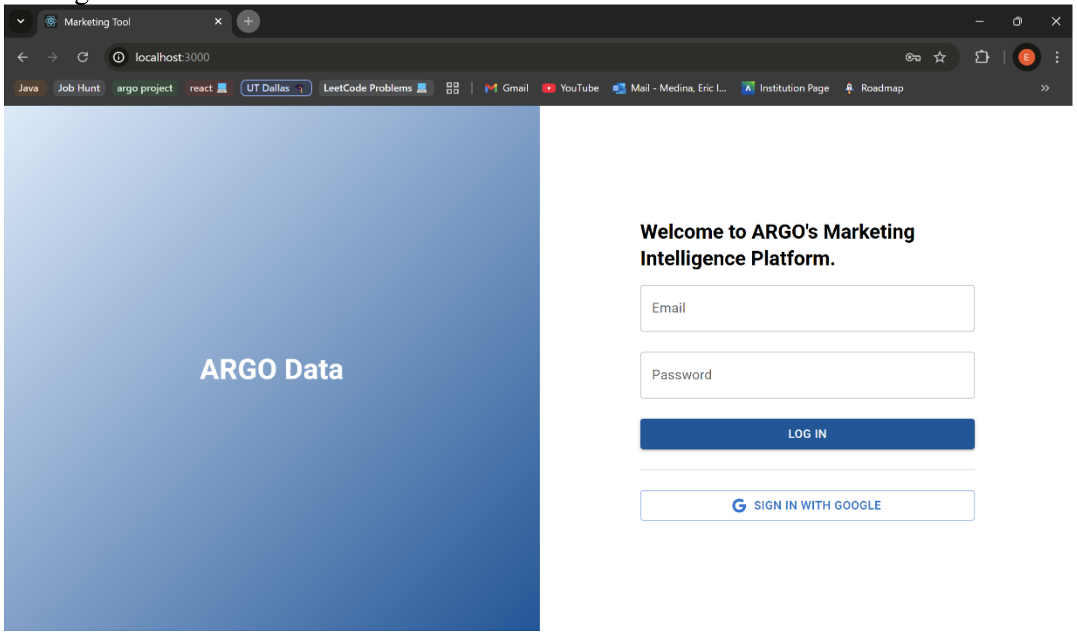
4.11.4. **Data Management Requirements**

- **Data Storage:**
    - Local storage for user preferences
    - Cache management for frequently accessed data
    - Session storage for temporary data
    - IndexedDB for offline capabilities
- **Data Processing:**
    - Data transformation services
    - Aggregation pipelines
    - Real-time processing capabilities
    - Batch processing for large datasets

5. Design

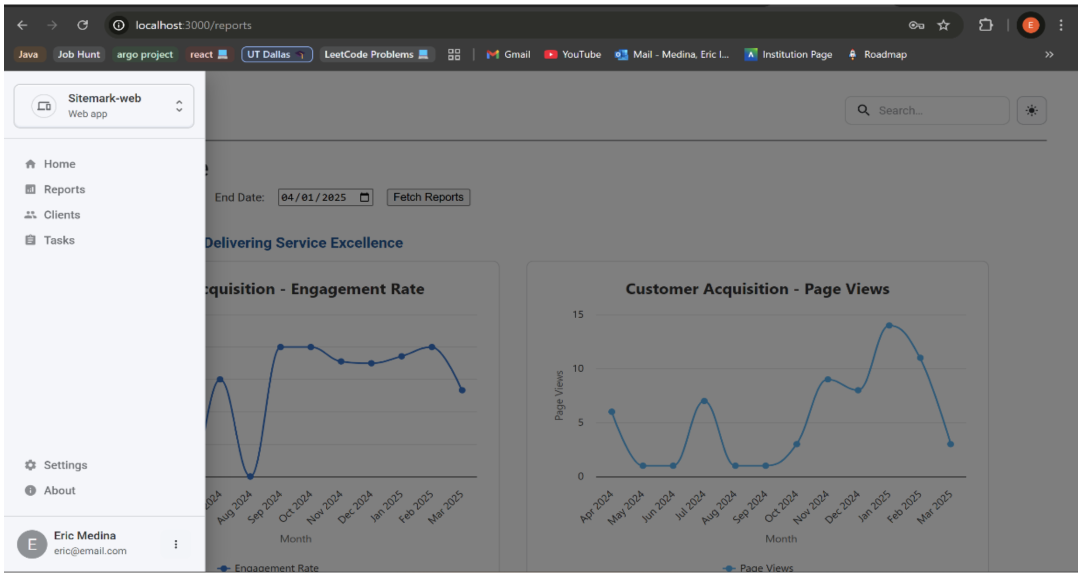
5.1. GUI (Graphical User Interface) design

- Login screen

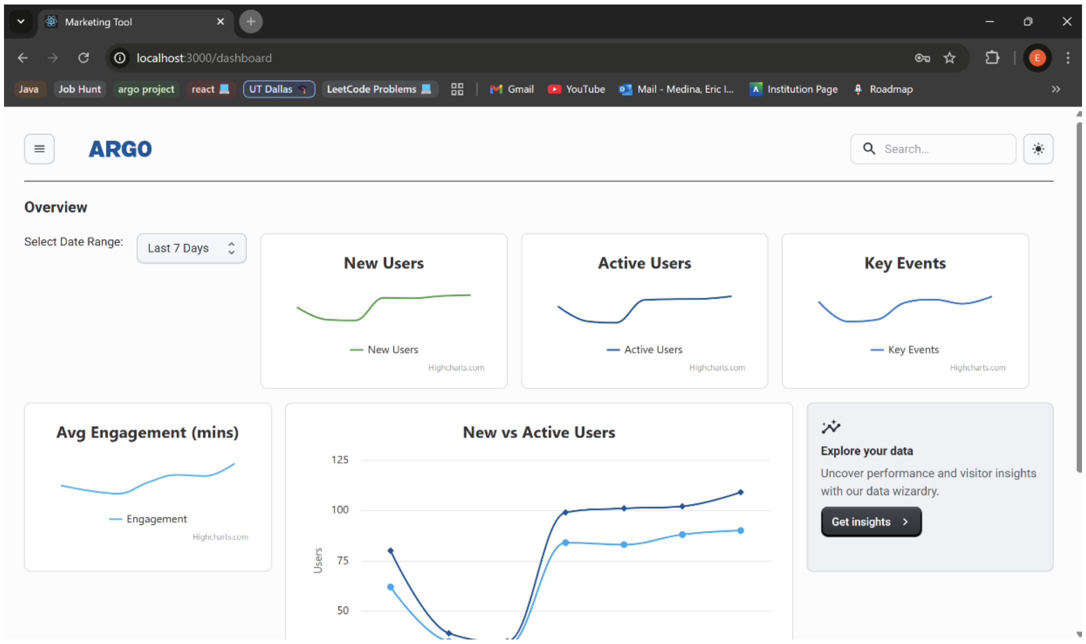


Welcome to ARGO's Marketing Intelligence Platform.

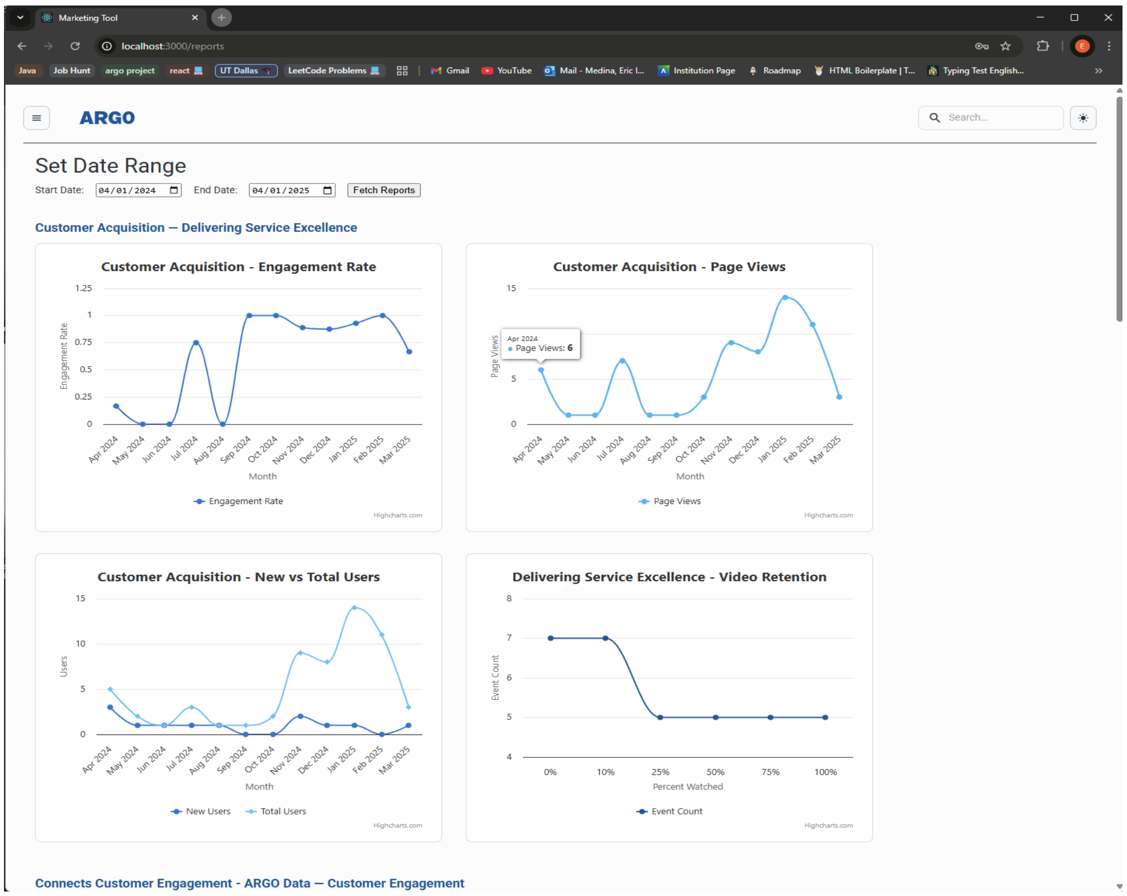
- Side menu screen



- Dashboard screen



- Reports screen



Set Date Range

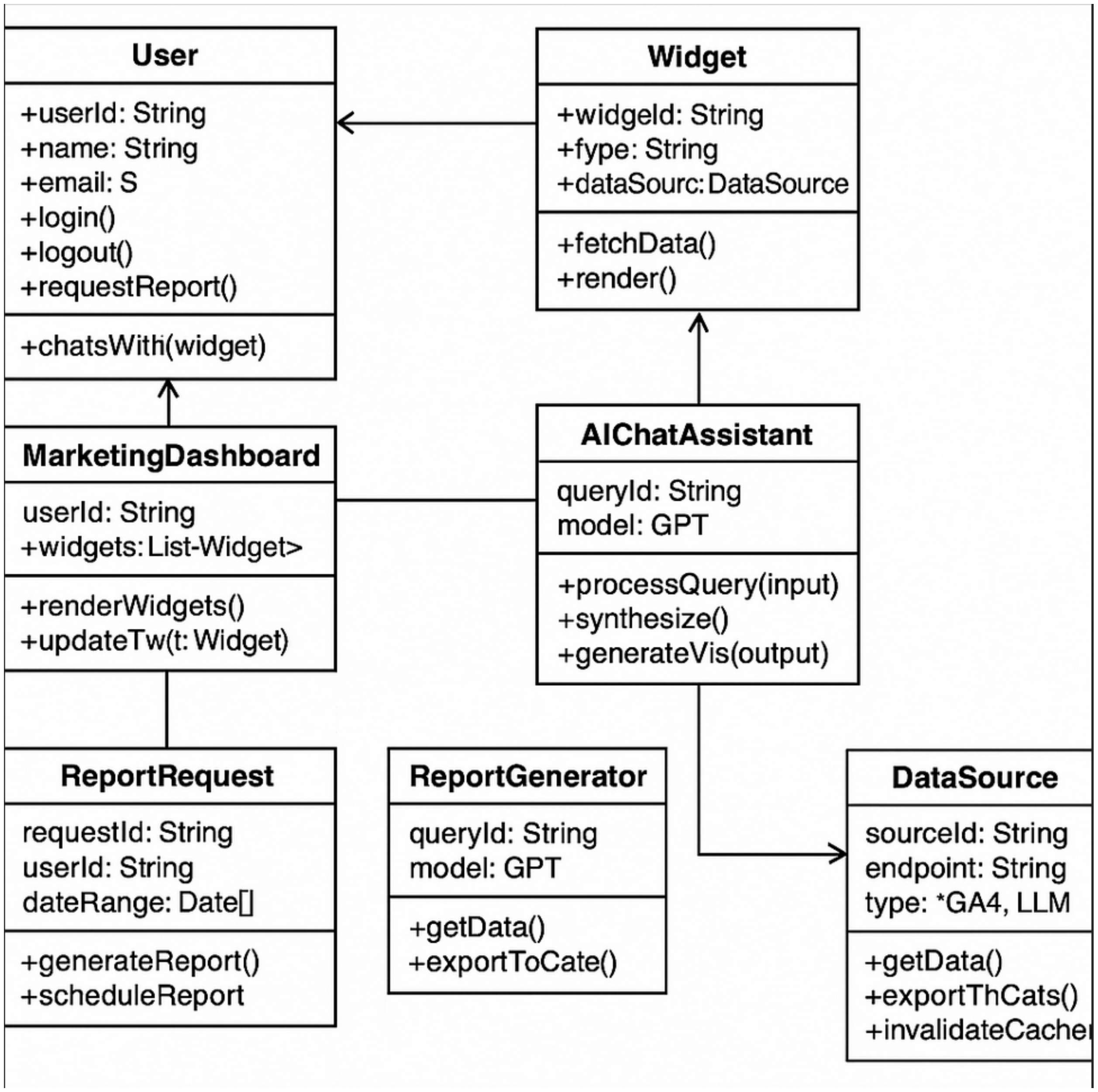
Start Date: 04/01/2024 End Date: 04/01/2025 Fetch Reports

Customer Acquisition — Delivering Service Excellence

Connects Customer Engagement - ARGO Data — Customer Engagement

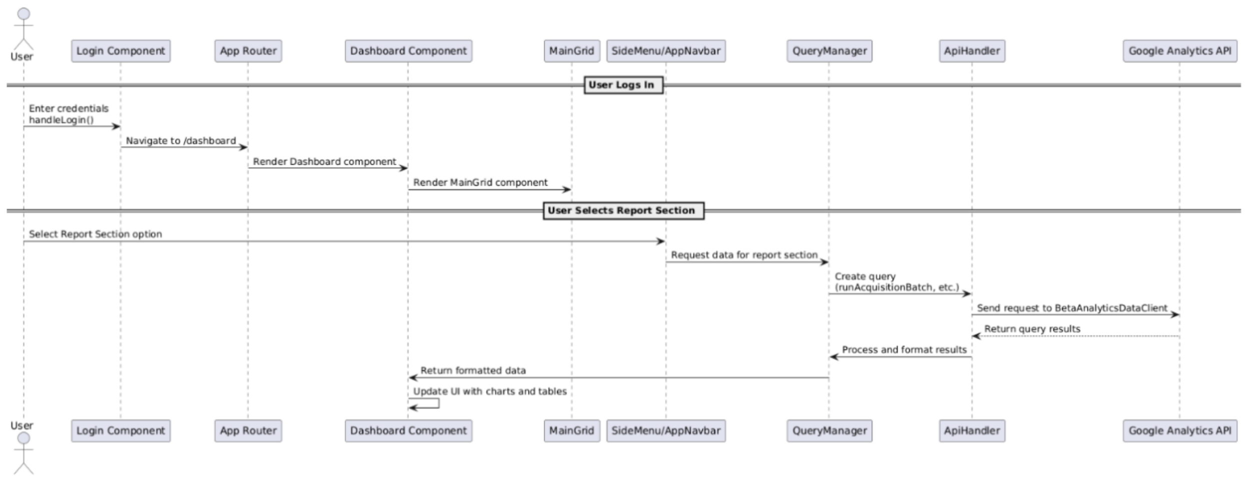
5.2. Static model – class diagrams

- captured in Rose (other tools are also allowed)

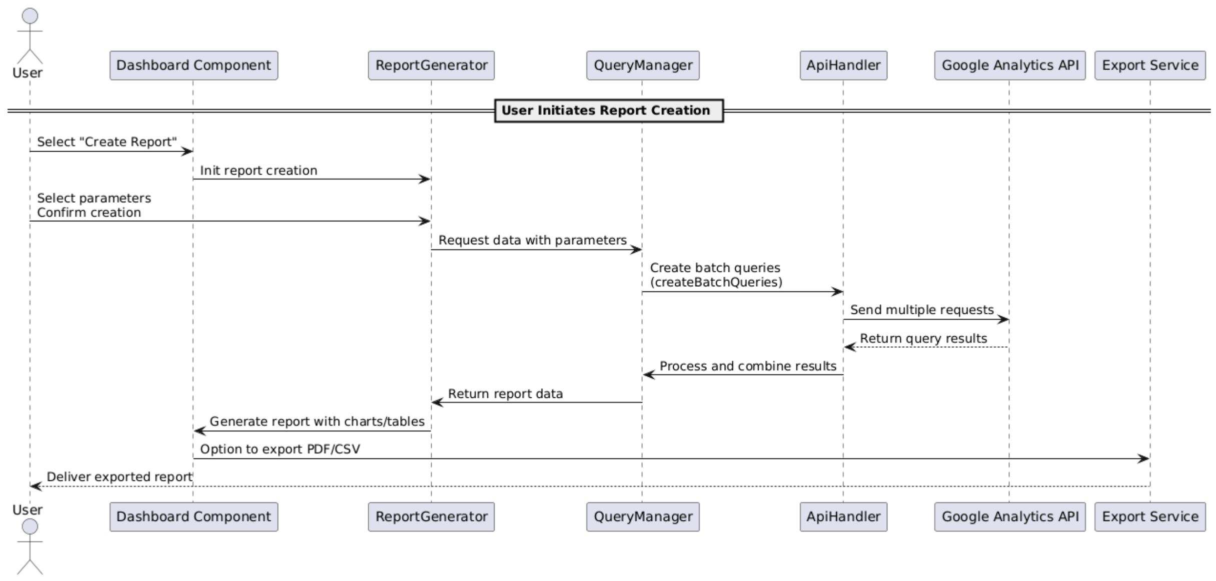


5.3. Dynamic model – sequence diagrams

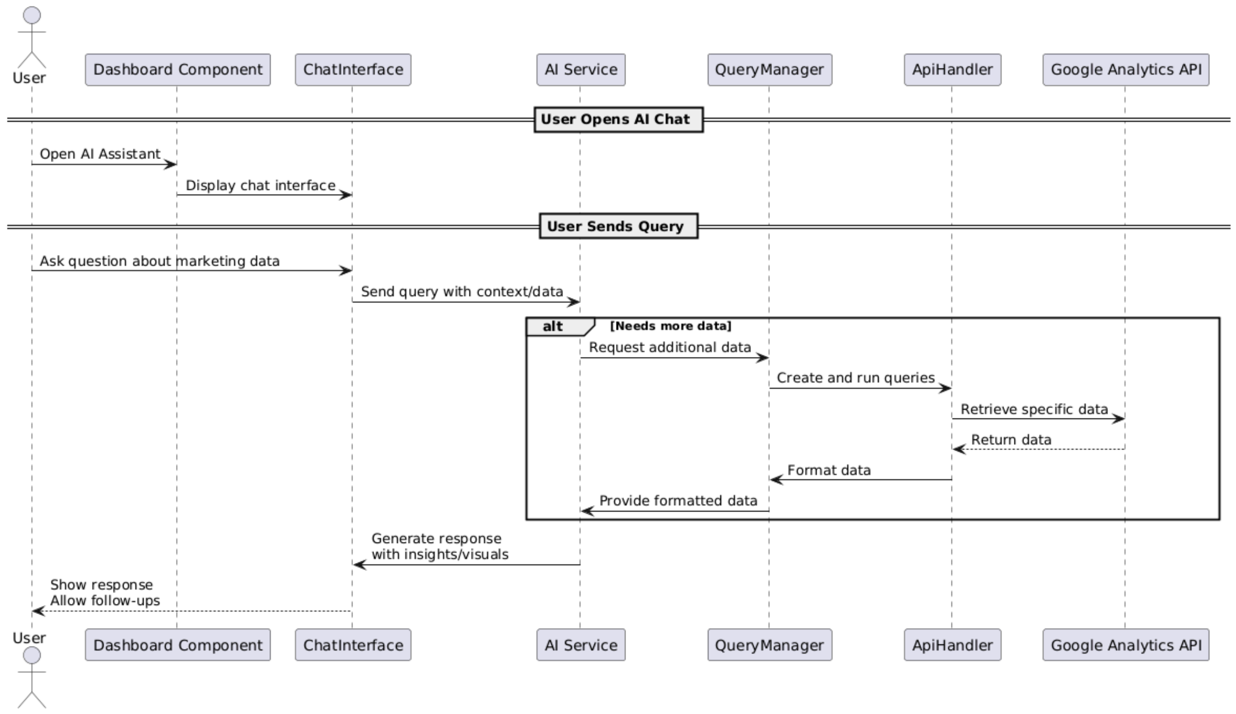
- Query Info for Reports Section


User
Login Component
App Router
Dashboard Component
MainGrid
SideMenu/AppNavbar
QueryManager
ApiHandler
Google Analytics API
User Logs In
Enter credentials
handleLogin()
Navigate to /dashboard
Render Dashboard component
Render MainGrid component
User Selects Report Section
Select Report Section option
Request data for report section
Create query
(runAcquisitionBatch, etc.)
Send request to BetaAnalyticsDataClient
Return query results
Process and format results
Return formatted data
Update UI with charts and tables

- Create Report



- Chat With AI Assistant



5.4. Rationale for your detailed design model

- Designed to meet both functional and non-functional requirements (scalability, maintainability, usability).
- Uses modular, microservice-based architecture for separation of concerns.
- Integrates LlamaIndex (LLM framework) as a customizable, lightweight alternative to LangChain.
- Closely aligns data models with GA4 schema for seamless ingestion.
- Includes a chatbot UI to enable natural language interaction with marketing data.
- Supports real-time queries and scheduled report generation.
- Designed with extensibility in mind for adding future analytics or data sources.

5.5. Traceability from requirements to detailed design model

• The following demonstrates how each requirement from the Requirements Documentation is addressed in the detailed design:

Requirement ID: REQ-1 - User needs to log in with ARGO credentials
• Design Components:
  - AuthController class
  - Login UI component
  - Authentication API endpoints
  - Sequence diagram: Query Info for Report Section
• Implementation Details:
  - JWT-based authentication
  - Login UI with error handling
  - Google OAuth integration
  - Authentication token validation

Requirement ID: REQ-2 - User Logout
• Design Components:
  - AuthController class

- Logout UI component
    - Session management service
- Implementation Details:
    - Token invalidation
    - Session state cleanup
    - Audit logging
    - Redirect to login screen

Requirement ID: REQ-3 - Request Engagement Metrics
- Design Components:
    - ApiHandler class
    - QueryManager class
    - Dashboard UI components
    - MainGrid component
    - Class diagram: Backend Classes
- Implementation Details:
    - GA4 API integration
    - Data transformation services
    - Metric visualization components
    - Caching for frequently accessed data

Requirement ID: REQ-4 - Report Download
- Design Components:
    - ReportGenerator class
    - ReportManager class
    - Export Service
    - Sequence diagram: Create Report
- Implementation Details:
    - PDF/Excel export functionality
    - Report templates
    - Report caching
    - Export formats with user selection

Requirement ID: REQ-5 - GPT Insights
- Design Components:
    - AiService class
    - ChatInterface component
    - AI API endpoints
    - Sequence diagram: Chat With AI Assistant
- Implementation Details:
    - OpenAI API integration
    - Context-aware queries
    - Natural language processing
    - Fallback mechanisms for API failures

Requirement ID: NF-REQ-1 - System operational when GA4 API is available
- Design Components:
    - ApiHandler error handling
    - Monitoring services
    - Fallback data caching
- Implementation Details:

  - Health check mechanisms
  - Monitoring dashboards
  - Graceful error handling
  - Data caching strategies

Requirement ID: NF-REQ-2 - Search results in under 5 seconds
- Design Components:
  - Query optimization
  - Result caching
  - Performance testing framework
- Implementation Details:
  - Indexed database queries
  - Multi-tier caching strategy
  - Load testing benchmarks
  - Performance monitoring

Requirement ID: NF-REQ-3 - Support 5-10 concurrent users
- Design Components:
  - Load balancing design
  - Auto-scaling architecture
  - Cloud infrastructure
- Implementation Details:
  - Elastic scaling policies
  - Resource optimization
  - Connection pooling
  - Request queuing

Requirement ID: NF-REQ-4 - UI compliant with ARGO QUADS
- Design Components:
  - Component library
  - UI theme providers
  - Design system integration
- Implementation Details:
  - Standardized component usage
  - Theme consistency
  - Design token implementation
  - Responsive layouts

Requirement-to-Design Mapping Process

- Authentication Requirements (REQ-1, REQ-2)
  - The design incorporates the AuthController class with specific methods for authenticating users, validating tokens, and revoking sessions
  - The login and logout sequence flows are clearly defined in the sequence diagrams
  - Security design includes JWT tokens and secure password handling as specified

- Data Processing Requirements (REQ-3)
  - The ApiHandler and QueryManager classes are designed specifically to handle GA4 API integration
  - Data transformation components handle the conversion of raw analytics data into usable formats
  - Caching mechanisms are implemented to optimize performance for frequently accessed metrics

• Reporting Requirements (REQ-4)
- The ReportGenerator and ReportManager classes provide the functionality for generating and exporting reports
- The sequence diagram for report creation shows the complete flow from user selection to delivery
- Export options support both PDF and Excel formats as required

• AI Integration Requirements (REQ-5)
- The AiService class manages integration with the OpenAI GPT API
- The Chat with AI Assistant sequence diagram outlines the process for natural language queries
- Error handling mechanisms address potential API failures as specified

• Non-Functional Requirements
- Performance requirements (NF-REQ-2, NF-REQ-3) are addressed through caching, load balancing, and cloud architecture
- UI compliance (NF-REQ-4) is ensured through the implementation of ARGO's QUADS design system
- System availability (NF-REQ-1) is maintained through monitoring and fallback mechanisms

6. Test Plan

6.1. Requirements/specifications-based system level test cases

Test Case ID	Description	Input	Expected Output	Priority
TC01	Engagement rate chart renders	Valid pageReport JSON	Chart shows engagement rate over time	High
TC02	Video retention chart renders	Valid videoReport JSON	Chart displays retention at 10%, 25%, 50%, 75%, 100%	High
TC03	Scroll depth chart appears	Valid scrollPercent custom dimension	Chart reflects scroll interaction	Medium
TC04	API fetch success	Valid user and GA4 parameters	API returns expected metrics	High
TC05	Chart expansion toggle works	User clicks "Expand"	All charts in the section are shown	Low
TC06	User login with ARGO credentials	Valid login credentials	User is authenticated and redirected to dashboard	High
TC07	User logout	User clicks logout	User is redirected to login page and session ends	High

TC08	View metrics on dashboard	Page and video report data	All charts render correctly in each section	High
TC09	Generate Chat-GPT insights	Valid data	AI-generated insights are shown under each section	Medium
TC10	Render customer acquisition charts	Valid data	Customer Acquisition section displays 4 charts	High
TC11	Render customer engagement charts	Valid data	Customer Engagement section displays 4+ charts	High
TC12	Render fulfillment charts	Valid data	Fulfillment section displays 4 charts	High
TC13	Render customer experience charts	Valid data	Customer Experience section displays 4 charts	High
TC14	Render risk management charts	Valid data	Risk Management section displays 4 charts	High
TC15	Render sales management charts	Valid data	Sales Management section displays 4 charts	High
TC16	Render service charts	Valid data	Service section displays 4+charts	High
TC17	Download Report	User clicks download	Report is saved as file	Medium

Table 1.0

6.2. Techniques used for test generation

In this project, we applied both black-box and white-box testing techniques to ensure that the Marketing Intelligence Platform behaves correctly at the system and code levels.

Test Generation Techniques Used

- Equivalence Class Partitioning (ECP):
  We divided input data into valid and invalid partitions to reduce the number of test cases while maintaining coverage. For instance, we tested typical, boundary, and invalid date ranges for

campaign filters.

- Boundary Value Analysis (BVA):
  This technique helped us identify potential off-by-one or edge-related errors. We applied BVA to inputs like date ranges, numerical thresholds for campaign metrics, and pagination values.

- Decision Table Testing:
  Used for validating business logic involving combinations of filters and user roles. This ensured that all logical outcomes for complex inputs were covered.

- State Transition Testing:
  We used this to verify dynamic UI components such as tab switching, user authentication states, and system responses after login or both user-defined and automatic data updates.

- Control Flow Testing (White-box):
  Applied during unit and integration testing to ensure that key backend functions covered all logical branches.

- Error Guessing:
  Based on our understanding of the platform and common errors, we manually created test cases targeting likely failure points.

6.3. Assessment of the goodness of your test suite

To evaluate the effectiveness and completeness of our test suite, we used the following quality criteria:

- Code Coverage:
  We used manual inspection and console-based logs to ensure major backend functions were tested, especially those responsible for filtering and computing marketing metrics.

- Requirement Coverage:
  Each functional requirement from the Requirements Document was linked to at least one test case. For example, requirements related to user roles, report generation, and chart accuracy were mapped directly to UI and API test cases.

- Traceability:
  All test cases were traceable to specific user stories or use cases. This mapping was documented to demonstrate full coverage of system functionality.

- Defect Detection Rate:
  During each testing iteration, we tracked the number of defects found relative to the number of tests executed to monitor how effective our tests were at identifying issues early.

- Test Case Effectiveness:
  We reviewed how many test cases resulted in the discovery of real defects (as opposed to false positives or redundant checks), refining low-yield test cases as needed.

- Automation Potential:
  While most tests were executed manually, we identified repeatable tests (e.g., checking default

chart rendering) that could be automated in future sprints to support regression testing in a CI/CD pipeline.

6.4. Traceability of test cases to use cases

- **Use Case: User Log In**
  - **Mapped Test Case(s):** TC06
  - **Description:** This use case is covered by TC06, which verifies that users can log in using valid ARGO credentials. The test confirms successful authentication and redirection to the dashboard, ensuring the system meets the requirement *"User needs to log in with ARGO credentials."*
- **Use Case: User Logout**
  - **Mapped Test Case(s):** TC07
  - **Description:** TC07 confirms the logout functionality. When a user clicks logout, they are redirected to the login page and the session is terminated. This test ensures the system fulfills the requirement *"User needs to log out of the dashboard."*
- **Use Case: Request Engagement Metrics**
  - **Mapped Test Case(s):** TC01, TC02, TC03, TC04, TC08
  - **Description:** These test cases verify that metrics are correctly fetched from the backend (TC04) and that individual visualizations for engagement rate (TC01), video retention (TC02), and scroll depth (TC03) display accurately. TC08 confirms that all charts render correctly across the dashboard. Together, they ensure the system meets the requirement *"User needs to view metrics – on dashboard."*
- **Use Case: Report Download**
  - **Mapped Test Case(s):** TC17
  - **Description:** TC17 validates the system's ability to generate and save a report when a user initiates a download. It ensures that a properly formatted file (e.g., PDF or CSV) is created, satisfying the requirement *"User needs to download the created reports."*
- **Use Case: GPT Insights**
  - **Mapped Test Case(s):** TC09
  - **Description:** TC09 tests the integration of Chat-GPT powered insights. It checks that when valid dashboard data is available, the system displays relevant AI-generated commentary under each section. This fulfills the requirement *"System needs to be able to provide Chat-GPT powered insights."*

7. Evidence the Document Has Been Placed under Configuration Management

8. Engineering Standards and Multiple Constraints

- IEEE Std 830-1998: Software Requirements
- IEEE Std 1016-1998 (Revision-2009): Software Design
- IEEE Std 1471-2000: Software Architecture
- IEEE Std 29148: Requirements Engineering
- IEEE Std 829-1983: Software Testing
- IEEE P3123: Artificial Intelligence and Machine Learning (AI/ML) Terminology and Data Formats
- ISO/IEC/IEEE Std 29148-2018: Systems and Software Engineering – Life Cycle Processes, Requirements Engineering
- ISO/IEC/IEEE Std 29119-1 (Rev. 2022): Software Testing – General Concepts

- ISO/IEC/IEEE Std 29119-2 (Rev. 2021): Software Testing – Test Process
- ISO/IEC/IEEE Std 29119-3 (Rev. 2021): Software Testing – Test Documentation
- ISO/IEC/IEEE Std 29119-4 (Rev. 2021): Software Testing – Test Techniques
- ISO/IEC/IEEE Std 42030:2019: Software, Systems, and Enterprise Architecture Evaluation Framework
- IEEE Std 12207:2017: Systems and Software Engineering – Software Life Cycle Processes

9. Additional References

- Lamsweerde, A.V., 2009. Requirements Engineering: From System Goals to UML Models to Software Specifications. John Wiley
- Larman, C., 2012. Applying UML and Patterns: An Introduction to Object Oriented Analysis and Design and Iterative Development. Pearson Education
- Hyman, B., 1998. Fundamentals of Engineering Design. Prentice Hall
- Simon, H.A., 2014. A Student's Introduction to Engineering Design. Pergamon Unified Engineering Series (Vol. 21), Elsevier
- Bass, L., Clements, P., Kazman, R., 2003. Software Architecture in Practice. Addison-Wesley
- Lattanze, A.J., 2008. Architecting Software Intensive Systems: A Practitioner's Guide. CRC Press
- Jorgensen, P.C., 2013. Software Testing: A Craftsman's Approach. Auerbach Publications
- Paul C. Jorgensen, 2021. Software Testing: A Craftsman's Approach (5th Ed.). Auerbach Publications
- Mathur, A.P., 2013. Foundations of Software Testing (2nd Ed.). Pearson Education
- QUADS Design System. quads.argodata.com
- UML Use Case Diagram Tutorial. Lucidchart. www.lucidchart.com/pages/uml-use-case-diagram
- React Documentation
- Material-UI Documentation
- Google Analytics Documentation

Acknowledgment

We would like to express our sincere gratitude to our project sponsors at ARGO — Raisa Gonzalez, and Goitom Kassaye — for their guidance, feedback, and support throughout the development of this project. We also thank Professor Wong and the SE 4485 teaching staff for providing us with the framework, resources, and mentorship to complete this capstone successfully. Finally, we appreciate the dedication and collaboration of our teammates, whose efforts made this project both rewarding and impactful.