CMPE 491/SENG 491

Senior Design Project I

Growth Genius

High-Level Design Report

Group Members:
Dide Sena Çalışkan 16348139678
Nilüfer Şahin 46225905362
Melda Bolat 24652172908
Çağrı Berk Çiftçi 23866341384

Note: Due to some problems, we changed our website link https://growth-genius.vercel.app/

1. Introduction

1.1 Purpose of the System

Growth Genius is an AI-powered business intelligence platform designed to empower small and medium-sized businesses (SMBs). It provides real-time analytics, predictive modeling, and actionable insights to address challenges such as data fragmentation, inefficient processes, and lack of actionable insights. The platform enables businesses to optimize resources, develop sales strategies, and adapt to market changes using cutting-edge technology and AI-powered methodologies.

The goal of Growth Genius is to simplify complex business operations and enable informed decision-making by transforming raw data into actionable intelligence. The ultimate goal of the system is to increase business profitability and sustainability in a dynamic market.

1.2 Design Goals

- **Scalability**: The platform must accommodate increasing volumes of users and data without compromising performance. Horizontal and vertical scaling mechanisms are integral to the architecture.
- **Security**: Data protection is a top priority, achieved through encryption, compliance with GDPR, and robust access control mechanisms.
- **Usability**: A user-friendly interface ensures accessibility to users with varying levels of technical proficiency. The design promotes intuitive interactions and efficient workflows.
- **Performance**: System performance is optimized for minimal latency, high availability, and rapid response times under concurrent loads.
- **Modularity**: The architecture follows a modular design to facilitate independent development, maintenance, and scalability of components.

1.3 Definitions, Acronyms, and Abbreviations

- **GDPR**: General Data Protection Regulation
- **API**: Application Programming Interface
- **RBAC**: Role-Based Access Control
- **KPI**: Key Performance Indicator
- **JWT**: JSON Web Token
- **SME**: Small and Medium-sized Enterprises
- **ML**: Machine Learning
- **UI/UX**: User Interface/User Experience

1.4 Overview

This report details the high-level design of Growth Genius, encompassing subsystem decomposition, hardware/software mapping, persistent data management strategies, access control policies, and handling of boundary conditions. Each section includes comprehensive explanations and design rationales to ensure a thorough understanding of the system.

2. Current Software Architecture

At present, there is no implemented software architecture. SMEs often rely on fragmented tools and manual processes for data analysis and decision-making, resulting in inefficiencies such as:

- **Data Fragmentation**: Business data resides across disparate systems like CRM, ERP, and POS, making integration challenging.
- **Manual Processing**: Time-intensive, error-prone methods limit scalability and adaptability.
- **Limited Insights**: Traditional tools fail to generate actionable recommendations, leaving businesses to interpret raw data without context.
- **Real-Time Deficiency**: Inability to process data dynamically and respond to rapid market changes.

Growth Genius aims to address these limitations by providing an integrated, AI-powered solution.

3. Proposed Software Architecture

3.1 Overview

The Growth Genius platform is a modular, AI-driven web application focused on business intelligence and recommendations. Its architecture adopts a clear client-server model with well-defined responsibilities. The frontend is built in React and maintains responsiveness, while the backend is written in Django and performs data processing, interacts with AI models, and calls APIs securely. This helps with scalability, maintainability of the code, and performance optimization.

3.2 Subsystem Decomposition

The system is divided into the following subsystems:

1. **Frontend Subsystem**:
    - Built using React and JavaScript.
    - Focuses on delivering a seamless user experience with dynamic data visualization and navigation.
    - Interfaces with the backend via RESTful APIs.
2. **Backend Subsystem**:
    - Developed using Django and its REST Framework (DRF).
    - Manages business logic, data processing, and machine learning model integrations (via PyTorch).
    - Handles role-based access control (RBAC) and user authentication using JWT.
3. **Data Management Subsystem**:
    - Powered by PostgreSQL for relational data storage.
    - Includes preprocessing pipelines for data cleaning and feature extraction using NumPy and Pandas.
4. **Recommendation Engine Subsystem**:
    - Integrates AI models for predictive analytics and strategic insights.
    - Provides dynamic updates based on real-time data.
5. **Notification and Report Generation Subsystem**:
    - Implements asynchronous notifications using Celery and Redis.
    - Generates user-customizable reports with visualizations in formats like PDF and Excel.

3.3 Hardware/Software Mapping

- **Frontend Deployment**:
    - Hosted on cloud platforms like AWS or Azure, ensuring high availability.
    - Employs content delivery networks (CDNs) for faster load times and reduced latency.
- **Backend Deployment**:
    - Deployed on a scalable cloud environment with Kubernetes for container orchestration.
    - Uses load balancers to distribute traffic across instances, maintaining optimal performance.
- **Data Storage**:
    - PostgreSQL database deployed with replication for fault tolerance.
    - Encrypted storage for sensitive data ensures compliance with GDPR.

3.4 Persistent Data Management

- **Database**:
  - PostgreSQL serves as the primary database, optimized with indexing for query efficiency.
  - Stores user data, business analytics, and AI model outputs.
- **Data Integrity**:
  - Ensured through validation pipelines for data input and preprocessing.
  - Backup systems provide automated data recovery capabilities in case of failures.
- **Compliance**:
  - Implements GDPR-aligned encryption (AES-256) for data at rest and in transit.

3.5 Access Control and Security

- **User Authentication and Authorization**:
  - Managed via Django's authentication system with bcrypt for password hashing.
  - Supports role-based access control (RBAC) to limit permissions based on user roles (e.g., admin, standard user).
- **Data Security**:
  - Employs HTTPS for secure API communications.
  - Multi-factor authentication (MFA) adds an additional security layer.
- **Monitoring and Alerts**:
  - Tracks user activities and system anomalies, providing real-time alerts for suspicious behaviors.

3.6 Global Software Control

- **Control Strategy**:
  - The backend orchestrates all global interactions, including user sessions, API requests, and database operations.
  - A master controller in the backend ensures synchronization across subsystems.

- **Error Handling**:
  - Implements robust error management, logging issues with ELK Stack (Elasticsearch, Logstash, Kibana) for diagnostics and resolution.

3.7 Boundary Conditions

- **Startup Conditions**:
  - Upon deployment, the system initializes its database connections, preloads AI models, and sets up the API endpoints.
- **Shutdown Conditions**:
  - Ensures data consistency by gracefully terminating active sessions and flushing task queues before shutting down services.
- **Failure Recovery**:
  - Automatic failover mechanisms with replication databases and server clustering minimize downtime.
  - Regular backups and disaster recovery plans safeguard critical data against loss.

4. Subsystem Services

4.1 User Authentication

Registration:

- The system works with registration through the email and password of users
- •Registration password policies of the strongest kind (i.e., min length, special character use) Available option to display error messages to the user if any fields are missing or are invalid.
- Upon registration, the verification link will be sent to the user's registered email ID to validate the registration. The account works only when the verification is done.

Login:

- Users are issued a signed JWT JSON Web Token (JWT) for authentication in subsequent requests once they have logged in successfully.
- Allowing multiple failed login attempts will, in turn, unlock temporary account lockouts, providing a more robust security layer against differential attacks.

Password Management:

- The "Forgot Password" feature allows the user to trigger a password reset.
-
- After that, a reset link will be sent to the registered email address (valid only for some time 24 hours).
- Passwords — bcrypt algorithm (as on new registration) but the same security used during reset
- Resetting passwords are stored in a bcrypt hash with the password encryption standard employed.

Access Control:

- Role-Based Access Control (RBAC) categorizes users into specific roles (Admin, Standard User, Manager).
- Admin users have full access to all system functionalities, whereas Standard users are restricted to their own data, preventing unauthorized access to sensitive information.

4.2 Data Management

Data Input:

- Users can conveniently upload or import data into the system (CSV, JSON, etc.) via API integration.
- As part of data loading, the system discovers issues like null values, wrong format, and schema mismatches while the user gets the alert.

Preprocessing:

- After validation, data is normalized, and missing fields are filled. Advanced feature engineering techniques are applied to prepare data sets for analysis.
- For example, date formats are standardized and categorical data is encoded (e.g., one-hot encoding).

Storage:

- Data is stored in a PostgreSQL database. A normalized schema design prevents data redundancy and improves performance.
- Large datasets are partitioned to enhance query performance and ensure system scalability.

Data Security:

- Data is encrypted both at rest (using AES-256) and in transit (using TLS protocol).
- Access to data is restricted based on the RBAC model, and access logs are maintained for auditing purposes.

4.3 Recommendation System

Model Training:

- Machine learning models are trained on historical data to generate advanced insights. Datasets are regularly updated to improve algorithm accuracy.
- Trained models consider multiple factors such as user behavior, sales trends, and customer segmentation.

Real-Time Recommendations:

- Trained models provide real-time predictions and recommendations via RESTful APIs.
- Recommendations are tailored based on user-provided parameters (e.g., pricing suggestions for a specific product group).
-

Scenario Modeling:

- Users can use strategy changes to make 'what if' scenarios to model potential strategic impacts on the system.
- To illustrate: The system can say what would happen to the customer or the revenue due to say price changes, etc.

4.4 Notifications

Trigger Events:

- The system detects events such as sudden changes in sales performance, subscription renewals, or data anomalies and sends notifications to users.
- Users can configure thresholds for events that trigger notifications (e.g., a specific KPI growth rate).
-

Delivery Channels:

- Notifications are sent via email, SMS, and in-app messages. Users can select one or multiple channels for receiving notifications.
- Notifications are processed asynchronously using Redis and Celery, minimizing delays.

User Preferences:

- Users can customize which notifications they receive and through which channels (e.g., email only). This enables personalized user experiences.

4.5 Reporting

Customizable Reports:

- The system allows users to easily generate the reports they need; users can choose a date range, KPI(s), and other parameters themselves. For example, report examples include highly descriptive sales performance/customer behavior analysis and resource allocation.

Visualizations:

- The reports are enriched with visual tools like graphs, charts, and heatmaps. These visualizations provide better insights to support business decision-making processes.

Export Options:

- Users can export reports in PDF, Excel, or PowerPoint formats.
- Scheduled reporting functionality allows automatic delivery of regular updates via email.

5. Glossary

Refer to Section 1.3 for definitions of key terms, expanded with additional technical terms as required during implementation.

6. References

1. Bruegge, B., & Dutoit, A. H. (2004). *Object-Oriented Software Engineering, Using UML, Patterns, and Java* (2nd ed.). Prentice-Hall.
2. GDPR Guidelines: https://gdpr.eu
3. Cloud Computing Resources: https://www.nist.gov
4. Django Documentation: https://docs.djangoproject.com
5. React Documentation: https://reactjs.org