Software Design Specifications

for

<Advantage 1.0>

Prepared By:

Harsha Dayani Akula	**SE22UCSE107**
Pranav Yeturu	**SE22UCSE214**
Sahithi Nampally	**SE22UCSE179**
Manasvi Boggarapu	**SE22UCSE053**
Yashaswi Matla	**SE22UCSE300**
Zauq Mohammed	**SE22UCSE172**
Vijaya Sai Chigurupati	**SE22UCSE293**

Document Information

Title: ADvantage SDD	
Project Manager: Team	**Document Version No: 1**
	Document Version Date: 7-04-25
Prepared By: Team	**Preparation Date: 5-04-25**

Version History

Ver. No.	Ver. Date	Revised By	Description	Filename
1	7-04-25	Team	Version 1 draft	Advantage_SDD

Table of Contents

1 INTRODUCTION

1.1 Purpose

This Software Design Specification (SDS) document outlines the architectural and technical design of ADvantage, an AI-powered web platform for generating and deploying contextually relevant advertisements. It is intended for use by developers, testers, and stakeholders as a reference to ensure alignment with the system's functional and non-functional requirements. It ensures consistent understanding and alignment across the development life cycle. The SDS is a critical reference point for ensuring the system's design is robust, scalable, and aligned with the intended product vision.

1.2 Scope

ADvantage leverages AI and real-time data—such as social trends and news data—to help businesses generate highly contextually-relevant ads. These ads can be previewed and deployed through email increasing relevance and engagement. The SDS covers all core modules, interfaces, data models, and system interactions involved in the development.

The scope of this document includes:

- Design and architecture of all major modules and subsystems
- Descriptions of interfaces and APIs
- Data flow and interaction models
- Integration points with third-party services
- Security considerations and constraints

1.3 Definitions, Acronyms, and Abbreviations

- **AI**: Artificial Intelligence
- **UI**: User Interface
- **API**: Application Programming Interface

1.4 References

- Software Requirements Specification Document for ADvantage
- IEEE 1016-2009 Standard for Software Design Descriptions

2 USE CASE VIEW

2.1 Use Case

- Use Case 1: User Login and Access

  Actors: Marketing Team (Paid Users), Non-Users
  Description:
  Users log in using token-based authentication to access the ADvantage platform. Paid users can access the ad generation tools, while non-users are directed to a public landing page with information about the product, demos, and pricing.

  Steps:

  1. User accesses the platform and selects login or sign-up.
  2. Upon authentication, the user is directed to the dashboard.
  3. User access level determines available features.

- Use Case 2: Ad Generation Based on Trend Data

  Actors: Logged-in Marketing Team
  Description:
  Users input information about their company and the product to promote, select the tone, purpose, intent filters and the platform fetches trending topics. The system recommends trend-topic combinations relevant to the business category. Users then select a trend and provide optional inputs for tone and style.

  Steps:

  1. User selects a business category from predefined options.
  2. System fetches real-time national trends.
  3. Trends are filtered and matched with the selected category.
  4. User selects a trend-topic combination and provides an optional prompt.
  5. User selects tone/style from available options.
  6. System uses an LLM to generate trend-based ad content.
  7. Generated ad is displayed to the user.

- Use Case 3: Customize and Re-generate Ads

  Actors: Marketing Team
  Description:
  Users can refine or regenerate ad content based on updated instructions. This supports iterative improvement and creative control over the final advertisement.

  Steps:

  1. User selects the generated ad they want to modify.
  2. Users can update the content manually.
  3. System saves the updated ads and gives access to send ads to clients.

- Use Case 4: Copy and Share Generated Ads

  Actors: Marketing Team
  Description:
  Once satisfied with an ad, the user can select the ad and send the ads to the customers once the data is provided

  Steps:

  1. Users uploaded customer data initially.
  2. Users select the Ads they would like to publish
  3. User hits the Send Ads button.

3 DESIGN OVERVIEW

The design of the *ADvantage* platform is centered around generating and delivering contextually relevant advertisements by leveraging real-time external data such as social media trends, google news, google trends. The system is modular and scalable, built using modern development tools that facilitate smooth integration with AI models and external APIs.

The overall architecture follows a layered and service-oriented approach, ensuring separation of concerns, reusability, and ease of maintenance. This section outlines the design goals, architectural components, external dependencies, and interfaces that make up the *ADvantage* system.

3.1 Design Goals and Constraints

Design Goals

- Enable AI-powered contextual ad generation using real-time data from external sources.
- Provide a simple, intuitive dashboard for businesses to create, manage, and deploy ad campaigns.
- Ensure a modular architecture that supports easy upgrades, third-party integration, and long-term scalability.
- Maintain security and privacy compliance with data protection standards such as GDPR and CCPA.
- Utilize lightweight, asynchronous frameworks for optimal performance and responsiveness.

Design Constraints

- External API limitations (e.g., rate limits or paid tiers) may affect trend data availability.
- Development timeline is fixed (target: May 20, 2025), so we're focusing on building a rock-solid MVP.
- Small team of 7 developers, so feature prioritization is essential.
- Starting with static trend data (mock JSON/CSV) before moving to real-time integrations.

3.2 Design Assumptions

- APIs like OpenAI, Google custom search are reliable.
- Django Auth or JWT-based authentication will be sufficient for secure user access during the MVP.
- Business users will primarily use the web dashboard, while end-users will engage via messaging channels
- AI-generated content based on real-world trends will be relevant and effective for marketing.
- Future updates (e.g., real-time trend scraping or premium cloud scaling) will be easy to integrate.

3.3 Significant Design Packages

- Ad Generation Engine (AI/LLMs)

- User Authentication and Campaign Management
- Deployment Gateway
- Data Ingestion Layer (fetches national level trends)

3.4 Dependent External Interfaces

The table below lists the public interfaces this design requires from other modules or applications.

External Application and Interface Name	Module using the Interface	Functionality/Description
Google Trends (Web Scraping)	Community Trend API	Extracts trending search topics in a given region to inform contextual ad generation.
Google Custom Search API	Community Insight Module	Retrieves recent articles or web content related to user or business categories.
OpenAI API (GPT-4o)	Ad Content Generator Module	Generates AI-driven advertisement content using trends, news, and business data.
SMTP server	Ad Delivery Module	Sends approved ad copies to customers or target recipients via email.
Events Web API / Scraper (Optional)	Local Event Awareness Module	(If used) Scrapes or queries local event data to enhance time-based ad relevance.

3.5 Implemented Application External Interfaces (and SOA web services)

The table below lists the implementation of public interfaces this design makes available for other applications.

Interface Name	Module Implementing the Interface	Functionality / Description
Ad Generation API	backend-core	Accepts user inputs and triggers the OpenAI API for context-aware ad content generation.

Ad Review API	frontend-ui / backend-core	Provides the UI and logic for reviewing, editing, and approving AI-generated advertisements.
Campaign Push API	integration-apis	Pushes finalized ads to email via the publish API for delivery to users.
User Authentication API	auth-service	Manages user sign-up, login, token validation, and session handling (via JWT)

4 LOGICAL VIEW

4.1 Design Model

The system architecture is composed of the following main modules:

1. Input Processing Module

- UserInputHandler
  Captures the Product Description from the user.
- InputValidator
  Validates the the Product Description is meaningful.

2. Data Aggregation & API Integration Module

- DataParser
  Parses and structures data received from APIs into a consistent internal format.
- TrendFetcher
  Used when generating advertisements based on trending topics
- RecommendationEngine
  Filters and matches fetched trends with the user's selected business category.
  Recommends relevant trend-topic combinations.

3. Data Persistence Module

- DatabaseManager
  Handles all interactions with the database, including storing user input and scraped data.
- DataModel classes (e.g., UserQuery, APIScrapedData)
  Represent structured data stored in the database.

4. Advertisement Generation Module

- LLMEngine
  Interfaces with the Large Language Model to generate personalized advertisements using stored data.
- AdReviewManager
  Provides ad previews to users and supports further edits
- AdVersionController.
  Enables iterative editing and creative experimentation.

5. Advertisement Delivery Module

- Email servicce
  Sends approved advertisements to customers using SMTP server and user-provided emails.
- MessageFormatter
  Formats generated ads into WhatsApp-compatible message templates.
- ClipboardService
  Enables easy reuse of generated ad text.

4.2 Use Case Realization

Use Case 1: User Login and Access

Actors: Marketing Team (Paid Users), Non-Users
Modules Involved:

- UserInputHandler
- AuthService
- UserManager
- DashboardController
- LandingPageManager

Realization Steps:

1. UserInputHandler captures login or sign-up credentials.
2. AuthService validates the credentials using token-based authentication.
3. If authentication is successful:
    - UserManager checks the user's role.
    - DashboardController displays the dashboard for paid users.
4. If the user is not registered or is a non-user:
    - LandingPageManager redirects them to a public page with product information, demos, and pricing.

Use Case 2: Ad Generation Based on Trend Data

Actors: Logged-in Marketing Team
Modules Involved:

- TrendFetcher
- CategoryManager
- LLMEngine
- AdGenerator
- UserInputHandler
- RecommendationEngine

Realization Steps:

1. User selects a business category through UserInputHandler.
2. TrendFetcher retrieves real-time national trends.
3. RecommendationEngine filters and matches the trends with the selected category.
4. User selects a trend-topic combination and optionally provides a custom prompt.
5. User also selects tone and style preferences.
6. AdGenerator compiles the inputs and sends them to LLMEngine.
7. LLMEngine generates personalized advertisements.
8. AdGenerator displays the generated ad to the user.

Use Case 3: Customize and Re-generate Ads

Actors: Marketing Team
Modules Involved:

- AdReviewManager
- UserInputHandler
- LLMEngine
- AdVersionController

Realization Steps:

1. User selects an existing ad via dashboard.
2. User enters revised instructions or prompts using UserInputHandler.
3. Can be able to edit the existing Ads and hashtags.
4. AdVersionController presents the new version alongside the previous one for comparison.

5 DATA VIEW

5.1 Domain Model

The domain model below represents the core persistent entities in the ADvantage system and the relationships among them. These entities are primarily domain objects used to manage users, ad generation requests, generated content, campaigns, and usage-based payments.

- User: Represents a registered individual using the platform. Includes both business users and administrators.
- AdRequest: Captures every instance of a user generating an advertisement by submitting a request to the system.
- Advertisement: Stores the actual AI-generated ad content corresponding to a specific ad request.
- Campaign: Allows users to organize multiple ads into scheduled or ongoing marketing campaigns.

5.2 Data Model (persistent data view)

5.2.1 Data Dictionary

Entity	Attribute Name	Description
User	user_id	Primary key. Unique identifier for each user.
	email	Email address used for registration and communication.
	password_hash	Hashed password for secure authentication.
	role	User role type (e.g., admin, business).
	created_at	Timestamp indicating when the user registered.
AdRequest	request_id	Primary key. Unique identifier for each ad generation request.
	user_id	Foreign key referencing the requesting user.
	prompt_input	User's input or prompt to guide ad generation.
	tone	Selected tone/style for the advertisement (e.g., humorous, formal).
	genre	Ad category or industry type (e.g., fashion, tech).
	timestamp	Time when the ad generation request was made.
Advertisement	ad_id	Primary key. Unique identifier for each generated ad.
	request_id	Foreign key referencing the associated AdRequest.
	content	AI-generated advertisement text.
	created_at	Timestamp when the ad content was generated.

6 EXCEPTION HANDLING

6.1 Exception Types and Scenarios

Exception	Scenario	Handling Mechanism	Logging	Follow-Up Action
ValidationError	Invalid user inputs (e.g., empty form fields, password mismatch)	Caught in views; form re-rendered with error message	Logged with WARNING level	Inform the user to correct input
ObjectDoesNotExist	Attempt to access a non-existent user or database record	Try/Except blocks to catch and redirect user to an error page	Logged with ERROR level	Show user-friendly error message; prompt to retry or go back

PermissionDenied	Unauthorized access to a protected view or API	Custom middleware/view decorators to restrict access	Logged with WARNING level	Redirect to login or "Access Denied" page
IntegrityError	Duplicate entries or violation of database constraints	Catch in the database layer; show appropriate message	Logged with ERROR level	Alert user or admin for manual intervention
SMTPException	Email sending failure (OTP, registration)	Caught and fallback shown to user with retry option	Logged with ERROR level	Prompt user to reinitiate action; notify admins if persistent
ConnectionError	Failure to connect to external APIs (e.g., weather, Reddit)	Fallbacks in service layer; notify user with fallback message	Logged with ERROR level	Retry logic or degrade gracefully with static default content
JWTDecodeError / ExpiredSignature	JWT token is missing, invalid, or expired	Custom middleware returns HTTP 401 Unauthorized	Logged with WARNING level	Prompt re-login or redirect to login page
MultiValueDictKeyError	Accessing a missing POST/GET form field	Default fallback values or error message rendered	Logged with INFO level	Prompt user to retry submission
Exception	Unknown server-side issues	Global exception handler logs the error and shows a generic error page	Logged with CRITICAL level	Notify admin via email/log; return user-friendly fallback response

6.2 Logging Strategy

All exceptions are logged using Python's logging module. The logging configuration is defined centrally and supports multiple levels:

- INFO: For general flow-level messages and benign exceptions.
- WARNING: For non-critical issues that should be monitored.
- ERROR: For significant failures that interrupt normal operations.
- CRITICAL: For unexpected failures that may bring the system down.

Logs are stored in log files (logs/app.log) and optionally integrated with monitoring tools for real-time alerts.

7 CONFIGURABLE PARAMETERS

This section outlines the configurable parameters used in the ADvantage application. These parameters are defined in the settings.py file and/or .env file. They control core aspects of the system including security, email integration, database, and file storage. The table below includes each parameter's name, purpose, and whether it is dynamic (can be changed without restarting the application).

Configuration Parameter Name	Definition and Usage	Dynamic?
DEBUG	Controls whether the application runs in development or production mode.	No
SECRET_KEY	A secret string for cryptographic signing. Used for session and token security.	No
ALLOWED_HOSTS	List of allowed domains/hosts the app can serve.	No

EMAIL_HOST_USER	Email address used to send OTP and system notifications (stored in .env).	Yes
EMAIL_HOST_PASSWORD	Password/app key for the above email (stored in .env).	Yes
DATABASES	Configuration for PostgreSQL database (name, user, password, host, port).	No
STATIC_URL	URL prefix for serving static files (e.g., CSS, JS).	No
STATICFILES_DIRS	List of directories to look for static files during development.	No
AUTH_USER_MODEL	Points to the custom user model (CustomUser).	No
JWT_SECRET_KEY (if used)	Secret key used for signing JWT tokens.	No
OTP_EXPIRY_MINUTES (custom)	Time (in minutes) after which the OTP becomes invalid.	Yes
RETRY_EMAIL_SEND (custom)	Number of retries allowed for sending OTP email.	Yes
API_KEYS (in .env)	API keys for 3rd party integrations like OpenAI, Reddit, Weather, etc.	Yes
DEFAULT_FROM_EMAIL	Sender email address for all outbound emails.	Yes

LOG_LEVEL (custom)	Sets logging verbosity (DEBUG, INFO, WARNING, ERROR, CRITICAL).	Yes

8 QUALITY OF SERVICE

This section outlines the quality-related aspects of the ADvantage platform's design, focusing on availability, security, performance, and monitoring in a production environment. These considerations ensure a robust, responsive, and secure experience for both administrators and end users.

8.1 Availability

The ADvantage platform is expected to be available 24/7 to support marketing teams operating across different regions and time zones. To meet this availability requirement:

- All APIs used for scraping data and trends are accessed asynchronously to prevent blocking the main application thread.
- The architecture adopts a modular microservice-style separation, so failure in one module does not affect ad generation or user interaction.
- Redundancy is built into key areas such as database access and third-party API fallback mechanisms to ensure continued functionality during outages.
- Maintenance tasks (e.g., data clean-up, session purges) are scheduled during off-peak hours and designed to be non-blocking.

8.2 Security and Authorization

Security is a critical component of ADvantage, especially due to the use of external APIs, personal user data, and third-party integrations.

- Authentication & Authorization: The platform uses token-based authentication to validate and authorize users. Role-based access ensures that only paid marketing users can access ad generation and delivery features, while guests are restricted to landing pages and public demos.
- Data Access Controls: Sensitive data like phone numbers and user queries are encrypted in storage. Only authenticated sessions can view, copy, or send generated ads.
- User Administration: User role management is handled via a centralized UserManager module. Admins can provision, revoke, or update access rights via a secure dashboard.

- Secure API Communication: All communication with external APIs (e.g., for trend fetching, WhatsApp messaging) occurs over HTTPS with secure tokens.

- Audit Logging: Every ad generation, modification, and delivery action is logged for traceability and accountability.

8.3 Load and Performance Implications

Given the system's real-time use cases and interactive nature, the following performance considerations are addressed in the design:

- Transaction Load: The system is designed to support hundreds of concurrent ad generations. LLM calls are offloaded to a dedicated service that can scale based on demand.

- API Load: API usage is rate-limited and uses caching for repeated queries to reduce redundant external requests.

- Database Growth: Ad drafts, user queries, and trend data are stored per session with periodic archival strategies in place to manage table size and maintain performance.

- Expected Latency:

    - LLM Response Time: ≤ 2 seconds per generation under normal load

    - API Fetching & Parsing: ≤ 1 second on average

    - Ad Review to WhatsApp Delivery: ≤ 3 seconds

- Stress Testing: The system design supports future load testing scenarios simulating high volumes of ad generation and WhatsApp message dispatch to evaluate scalability.

8.4 Monitoring and Control

The ADvantage platform includes built-in mechanisms for monitoring and control of its key services and processes:

- Monitoring Metrics:

    - API success/failure rates

    - LLM generation time
    - Email message delivery status
    - System uptime and user session counts
- Process Control:
    - Ad generation and regeneration are treated as controlled processes, with logs and checkpoints to resume interrupted tasks.
    - Daemons or background workers manage data cleanup, message retries, and trend refresh cycles.