Corevation LLC

PostAgent
Software Requirements Specification

Version 1.0

Revision History

Date	Version	Description	Author
12/02/2025	1.0	Initial Draft	Alyssa Turenne Charley Reavley Ryan Jordan Kayla Fruean
04/10/2026	1.1	Branding and Final Updates	Kayla Fruean

Table of Contents

Software Requirements Specification

1. Introduction

1.1 The Purpose of the PostAgent

PostAgent is an AI-powered content creation and scheduling platform designed to support small businesses, entrepreneurs, and service providers in maintaining a strong and consistent digital presence. By simplifying the workflow of creating, editing, and publishing social media content, PostAgent reduces the time, effort, and marketing expertise required while improving engagement outcomes. Users can generate text and image posts, connect external social accounts, view performance insights, and automate scheduling within a single, intuitive interface.

1.2 The Purpose of this Document

This Software Requirements Specification (SRS) defines the functional and non-functional requirements for PostAgent. It serves as the authoritative source of system expectations for stakeholders including project managers, developers, testers, and support staff. This SRS ensures all parties share a common understanding of system behavior, constraints, and quality goals, enabling accurate implementation, validation, and long-term maintenance.

1.3 Document Conventions

- “The System” refers to the PostAgent software product.
- Requirements are written in verifiable “system shall…” statements.
- Requirement identifiers use the format <Category>-<Number> (e.g., PER-1 for Performance).

1.4 References

- Project Glossary: URL
- Vision and Scope: URL
- Software Architecture: URL
- Use Cases: URL
- Business Rules: URL
- User Interface Wireframe/Prototypes: URL
- Branding Documentation: URL

2. Project Glossary

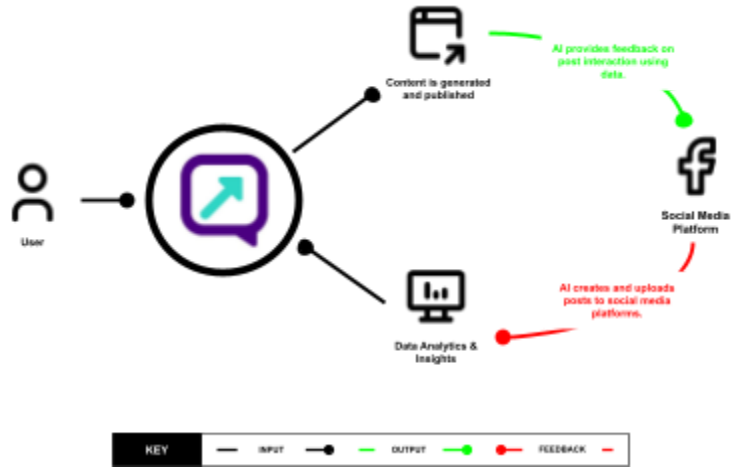
The project glossary is available here: URL.

3. Vision and Scope

The vision and scope document is available here: URL.

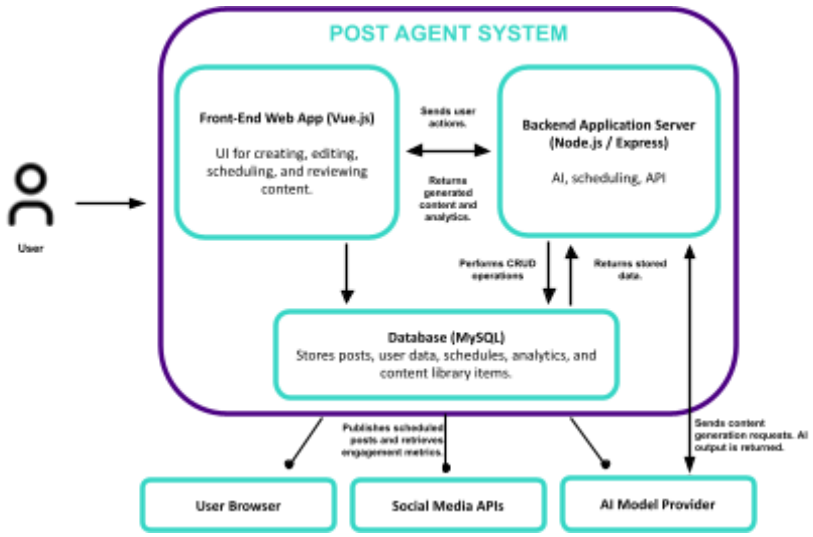
4. Software Architecture

4.1 System Context Diagram



The Level 1: Context Diagram for the PostAgent application provides a high-level overview of its interactions with users and external systems. PostAgent serves as a centralized AI-driven content creation platform, where users provide preferences and instructions through the integrated AI chatbot. The system generates posts and other content based on these inputs and publishes them directly to external social media platforms such as Facebook and LinkedIn. PostAgent also retrieves performance data from these platforms (such as views, interactions, and comparative engagement metrics) and delivers this feedback to the user. This diagram highlights the role of the user as the primary actor, the central functionality of PostAgent as an AI content manager, and its interfaces with social media platforms for publishing and data collection, offering a clear picture of the system's operational scope and interactions.

4.2 Container Diagram



The Level 2: Container Diagram for the PostAgent application provides a detailed view of its internal architecture and illustrates how its core components collaborate to deliver AI-driven content creation, scheduling, publishing, and analytics functionality. The system is composed of three primary containers: the Front-End Web Application, the Backend Application Server, and the Database, all supported by integrations with external social media platforms such as Facebook and LinkedIn. The Front-End, where users interact with the system, provides tools for generating and editing AI-assisted content, organizing posts, and reviewing performance insights. The Backend Application Server manages AI content generation, scheduling logic, publishing workflows, and data analytics while coordinating with the Database for storage and retrieval of user-generated and system-generated content. The Database stores posts, user preferences, scheduling data, analytics history, and internal usage metrics that support personalized recommendations.

Here are the detailed interactions and features enabled by these containers:

1. Access the PostAgent Platform:
   The user begins by accessing the PostAgent web application through their browser, where the Front-End interface is delivered.
2. Generate and Edit Content:
   Through the Front-End, the user can create text or images using AI-powered content generation tools. Wizard-inspired assistants—such as Pixie Polish, Spellbook, and Casting Carousel—guide users by offering creative suggestions and customization options. Generated content can be edited and personalized directly within the application.
3. Schedule and Publish Posts:
   When the user decides to publish, the Front-End sends requests to the Backend Application Server. The Backend handles scheduling logic, enabling users to save posts to a calendar for future publication or publish immediately. The system communicates with third-party platforms like Facebook and LinkedIn to post content on behalf of the user.
4. Store and Manage Content:
   The Backend performs CRUD operations with the Database, storing previously created posts in a centralized content library. Users can organize, retrieve, and search stored posts for later use. Users can also upload their own content, such as photos, text, and videos.
5. Provide Analytics and Insights:
   The Backend collects both internal data (such as user interactions within the app) and external performance metrics retrieved from social media platforms. These analytics are stored in the Database and visualized through the Front-End, offering insights such as engagement trends, optimal posting times, and recommendations for improving content strategy.
6. Training and Support:
   The Front-End provides access to built-in training videos and a helpdesk system. Users can request support, and the Backend manages the processing of helpdesk inquiries and retrieval of training resources.

This sequence of interactions demonstrates how the PostAgent system containers work together to offer seamless AI-assisted content creation, efficient scheduling and publishing, robust data analytics, and user support—providing a comprehensive operational overview consistent with the system's functional goals.

4.3 Operating Environment

OE-1: The system shall operate reliably on modern web browsers, including Chrome, Safari, Firefox, and Edge.
OE-2: The backend application shall run within a cloud-hosted environment, specifically deployed on Microsoft Azure.
OE-3: The system shall be accessible globally via a secure HTTPS connection to ensure encrypted communication and user data protection.

4.4 Design and Implementation Constraints

CO-1: The system must integrate with Facebook (Meta) and LinkedIn APIs for authentication, content publishing, and analytics retrieval.
CO-2: All external platform access shall use secure authentication methods, such as OAuth 2.0, in compliance with platform security policies.
CO-3: AI text and image generation services must adhere to the copyright, licensing, and usage restrictions defined by the underlying AI model providers.
CO-4: The PostAgent front-end shall conform to modern web standards, including HTML5, CSS3, and ECMAScript 6 or higher.
CO-5: The backend must run within the constraints of the Azure cloud environment, using approved technologies and services (e.g., Azure App Service, Azure SQL, Azure Functions).
CO-6: All data storage shall use the project's designated database engine (e.g., Azure SQL or MongoDB Atlas), following organizational security and encryption requirements.

4.5 Assumptions and Dependencies

AS-1: Third-party social media APIs (e.g., Facebook and LinkedIn) will remain available, maintain their current endpoints, and will not revoke the required permission scopes used for authentication, publishing, and analytics.
AS-2: AI model providers (e.g., OpenAI or equivalent services) will maintain stable availability, pricing, and API performance throughout the development and operation of the system.
AS-3: Users will have stable internet connectivity to support real-time content generation, publishing, and analytics retrieval.
AS-4: The Azure cloud environment will remain available and continue to support the services and configurations required by the PostAgent backend.

5. Functional Requirements

5.1 Use Cases

The use cases are available here: URL.

5.2 Non-Use Case Functional Requirements

- When a user creates content via AI generation and they decide not to use the content on a social media post, then the system shall immediately save the content to the content library in the event the user wishes to use the content at a later time.
- When a user hits an upward metrics trend on a social media post or account, such as likes or followers, the system shall immediately notify the user of the achievement.
- If a user does not give a post a title, the system shall default the title to (NO TITLE).
- If created content is scheduled to be posted on the calendar at a later date, the system shall begin the posting process 15 minutes ahead of schedule to suffice for any possible delays in the posting process.
- After a user inputs their website into the brand book, if the system can not find the website for any reason, then after three seconds it will send a pop up error to the user.
- The system shall welcome new users within five minutes of sign-up.

6. Business Rules

The business rules are available here: URL.

7. Data Requirements

7.1 Data Acquisition, Integrity, Retention, and Disposal

DI-1: Data shall be acquired via api calls to third-party platforms (i.e., Facebook, LinkedIn)

DI-2: All data that is acquired is in accordance with the permission guidelines granted by the third-party platform.

DI-3: Data will be stored in databases using MySQL.

8. External Interface Requirements

8.1 User Interfaces

The UI shall follow brand and accessibility guidelines as defined in Vision & Scope requirements and in the following branding guide: URL

Initial UI supports:

- *Enchanter's Controls*: Users review and update brand context that will be used for generation of content.
- *Pixie Polish*: Users review scheduled posts and posts labeled for further review.
- *Sorcerer's Spellbook*: Users can generate posts using underlying and current context, eventually accepting, rejecting, or saving posts to review later.
- *Content library*: Users can view all previously posted posts and export them as needed. Users can also upload and delete content.
- *Scheduling calendar*: Users can view and edit their post schedule in a calendar view.
- *Performance Analytics dashboard*: Users can view insights into connected social media platforms to see how trends have changed after using PostAgent.

8.2 Software Interfaces

SI-1: PostAgent Storage System:

SI-1.1: User data inputted via the brand book will be stored in the PostAgent database and used as

reference throughout the software.

SI-1.2: All data related to any post generated by PostAgent will be stored in the PostAgent database.

SI-1.3: When a third-party social media account is connected, all available data from posts on the user account will be stored in the PostAgent database.

SI-2: PostAgent AI Generation System:

SI-2.1: Create both text and images based upon user specifications.

SI-2.2: Allows for editing of both text and images by the user.

SI-3: PostAgent Review & Post System:

Any post created by the PostAgent AI Generation System will have the following options upon completion:

SI-3.1: Save to the review queue, for edits at a later time.

SI-3.2: Schedule and post, the created post will post at the day and time as dictated by SI-1.

SI-4: PostAgent Content Upload & Media Storage System:

SI-4.1: Users can upload images, videos, and text files through the Uploads interface.

SI-4.2: The system provides secure access for previewing and downloading media.

SI-4.3: Users can delete uploaded files, removing them from both storage and the database.

9. Quality Attributes

9.1 Usability

USE-1: New users will be able to follow the dedicated workflow to generate text and images intuitively.

USE-2: All main features will be available in the side menu.

USE-3: All pages will have a description of what happens there under the title.

9.2 Performance

PER-1: New users should be able to complete the getting started workflow within 15 minutes.

PER-2: User problem tickets should be acknowledged within 2 business days.

9.3 Security

SEC-1: OAuth is used to connect system accounts to Gmail and Microsoft accounts.

SEC-2: Stripe is used to handle secure payments for user subscriptions.

9.4 Safety

SAF-1: The use of OpenAI API provides safeguards that prohibit any responses that contain any negative or dangerous wording.

9.5 Availability

AVL-1: The system shall be available at least 98% of the time between 8:00 AM and 5:00 PM central standard time, and 95% 24/7, local time.

9.6 Robustness

ROB-1: User data is saved until the user completes the process of posting. This data includes wizard input and results, which are stored in local storage; if an error occurs, data is pulled from local storage until the user is finished with the post in the wizard.

10. Deployment

Deployment architecture:
The software is hosted on Microsoft Azure, which provides the compute, networking, and storage resources.

- Azure Functions for running backend application code.
- Azure storage for file and blob storage.
- Azure API Management for managing API access and rate limiting
- OpenAI API for AI-driven features
- Azure networking services such as Virtual Networks, Load Balancers, and Application Gateways

This architecture ensures high availability, built-in security, and cloud-native scalability.

Environments:
The system uses the following environments:

- Development Environment:
  - Runs locally on developer machines. Code is managed through GitHub repositories. Developers use test keys for Azure and OpenAI.
- Staging Environment:
  - Deployed on Azure to mirror production. Used for integration testing, user acceptance testing, and verifying new releases before going live.
- Production environments:
  - The live deployment access by end users will be on a website domain.

Deployment Procedures:
Deployment is managed through a GitHub Actions CI/CD pipeline, which automates:

- Code build and testing
- Packaging and deployment to Azure (Azure Functions)
- Database migrations for MySQL
- Environment-specific configuration updates

Developers trigger deployments by merging code to specific branches:

- main - Production development
- tcu-dev - student development and testing.

Ensures consistent, repeatable releases.

Updates and Rollback Strategies:
Azure provide built-in tools for mitigating deployment risks:

- Slot-based deployments
- Versioned deployments stored in GitHub and Azure
- Ability to quickly rollback by swapping deployment slots or redeploying a previous version from GitHub Actions

If an update fails, the system can revert to a stable version within minutes

Scaling and performance:

Azure services provide several scaling options:

- Autoscaling based CPU, memory, or request load
- Horizontal scaling (adding more instances)
- Vertical scaling (moving to a higher-tier plan)
- MySQL supports configurable compute and storage scaling
- Caching strategies can be added for heavy workloads
- OpenAI API throughput is managed using rate limits and retry policies

This ensures the system can grow to support increased usage.

11. Internationalization and Localization Requirements

The System is currently only suitable for the United States. Future international expansion is ideal, beginning with English-speaking countries, to which cultural differences (e.g. American vs British English) would be taken into account.