COMENIUS UNIVERSITY BRATISLAVA

FACULTY OF MATHEMATICS, PHYSICS, AND INFORMATICS

AI Content Generator

Project Specification

Aidas Jankauskas
aidas.jankauskas@mif.stud.vu.lt

BRATISLAVA 2023

Contents

Introduction

The application specified in this document enables users to generate various content in multiple languages. It is designed to provide users with the ability to generate articles, posts, essays and other texts in multiple languages that are suitable for both research and commercial purposes. The application uses OpenAI API to enable users are to generate unique and high-quality content that is tailored to their needs.

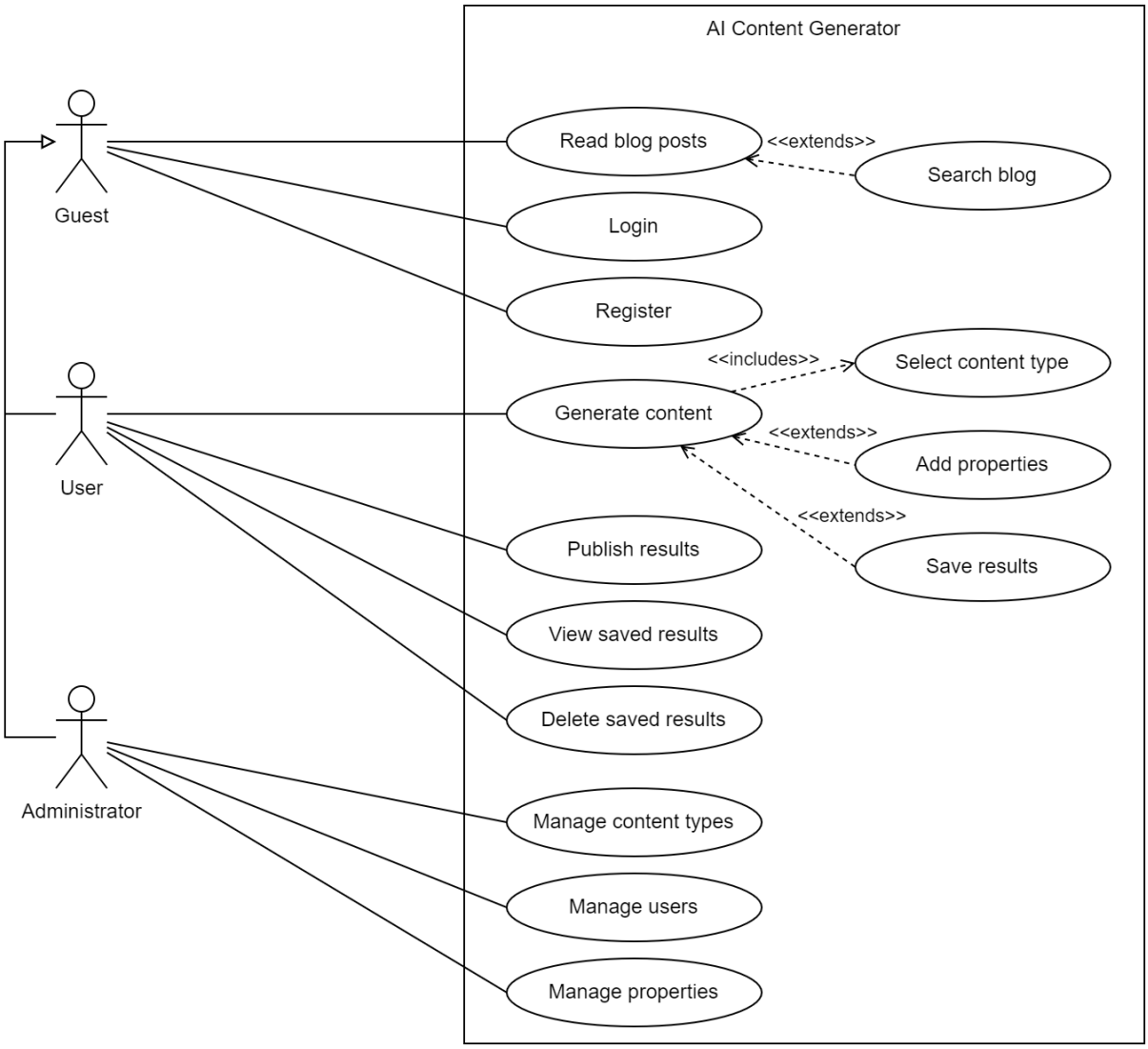
The web application requires users to sign up to access its features. After successful registration, a user can log in to the application to access its features. Website allows users to choose from different content categories, topics, and styles to generate custom content. Furthermore, the users can add custom keywords that are considered during the generation process. The application supports multiple user types: administrators and standard users. Administrators have access to manage categories, subcategories, and additional available properties of texts to be generated. Standard users, on the other hand, can generate texts, save them, and share them on the blog.

The application is implemented using client-server architecture. It consists of front-end web application and back-end server that serves requests and communicates with OpenAI API. The back-end application also stores and manages data in the database.

This application is targeting two main groups of users: individuals (students, researchers, bloggers, writers), and businesses that need high quality content for marketing and advertising purposes. The application offers a wide range of content categories, topics, and styles, making it suitable for various business needs. Individuals might need the generated content for their personal projects, research, or study purposes.

User Requirements

The functionality of the system is represented in the following use-case diagram.

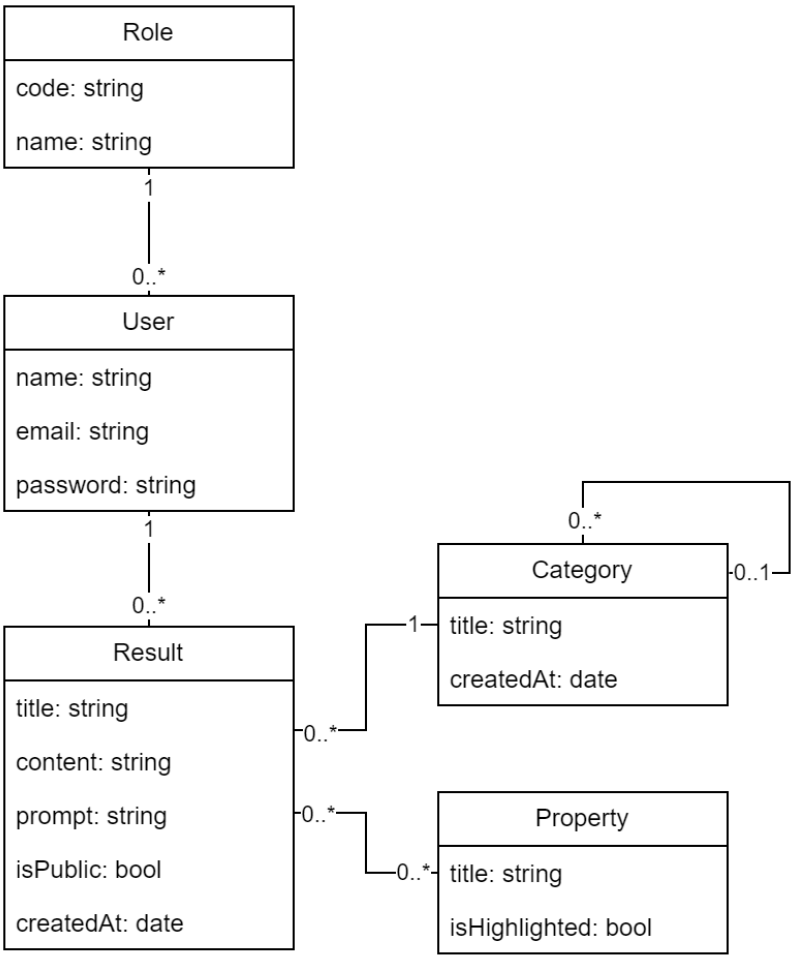


Guests (unauthenticated users) can visit blog and read generated content. Every guest is also able to login and register to the application. Primary functionality that is available for users is content generation. Users can generate content by selecting content type and optionally adding keywords. After generating the content, user can save and share the results on the blog. Users can also manage and delete their saved results. Administrator is able to manage content types and manage users by blocking malicious content creators.

Role	Description
Guest	Unauthorized user that has an intention to read blog, login, and register
User	Logged in standard user (primary target group). Is able to: • Generate content by selecting its type, adding keywords • Publish generated content to blog • Manage created blog posts
Administrator	User with elevated permissions that is responsible for managing content types and users.

Data Model

The data model is represented in the following class diagram.



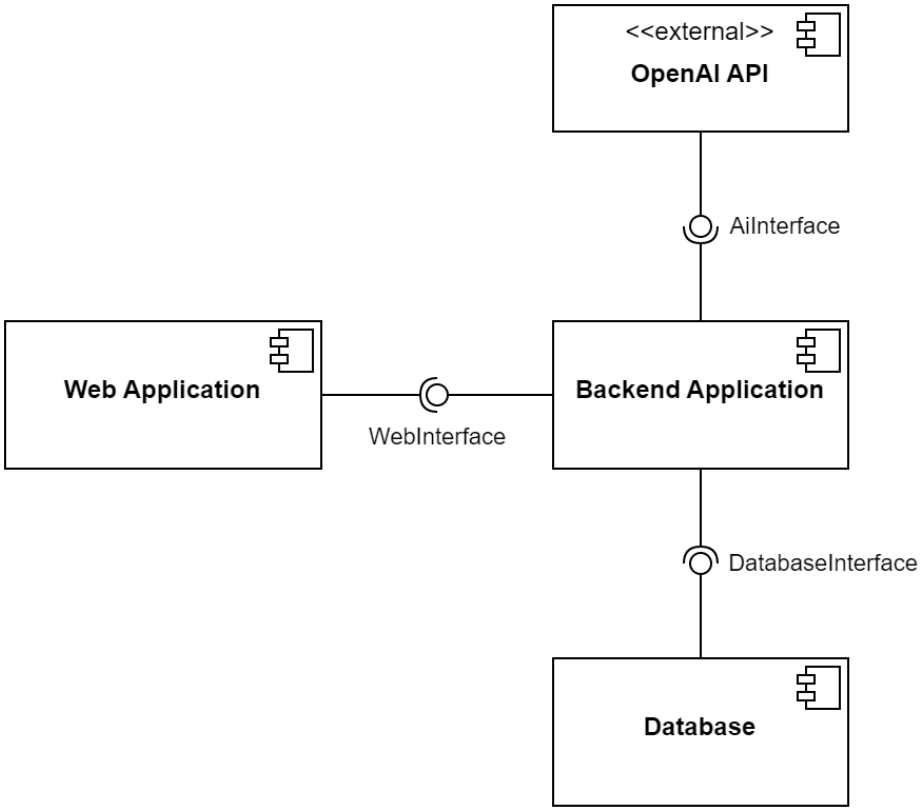
Data model consists of 5 entities:

- Category – list of content types, managed by administrator.
  - Title: the title of the category
  - CreatedAt: creation date
- Result – results/posts that are generated by AI and optionally published by users.
  - Title: the title of the post
  - Content: the content of the post that is generated by AI
  - IsPublic: defines if the post is publicly available in the blog
  - CreatedAt: creation date

- Property – custom properties and keywords that are available for content generation.
    - Title: the title of the property
    - IsHighlighted: defines if the property is displayed while generating content.
- User – list of users in the system
    - Name: name of the user
    - Email: email of the user that is used to login
    - Password: hashed password that is used for authentication
- Role – list of roles within the system
    - Code: system code used to identify the role
    - Name: the name of the role

Solution Architecture

The architecture of the application is represented in the following component diagram.



Application consists of the following components:

Component	Description
Web Application	Application that provides user interface and works in client browser
Backend Application	Application that serves requests from web application and interacts with OpenAI API and database to fetch data
Database	Storage for the application data
OpenAI API	External component that is used to generate content from prompts

Components in the application are connected by the following interfaces:

Component	Description
WebInterface	HTTPS interface for communication between web application and backend application
DatabaseInterface	TCP/IP interface for communication between backend application and database
AiInterface	HTTPS interface for communication between backend application and OpenAI API

Technical Requirements

Technical requirements are specified in the following table.

Client-side	**Language:** TypeScript **Library:** React 18
Server-side	**Environment:** Node.js 19 **Framework:** Express 4
Hosting	**Docker support:** Yes **Memory:** 1GB **Disk:** 20GB **Throughput:** 1TB
Database	**DBMS:** MongoDB 6
Supported Browsers	Chrome, Firefox

Timeline

The project timeline is provided in the following table.

Date	Milestone
2023-03-26	Initial project and local environment setup
2023-04-02	Authentication
2023-04-09	Administrator interface, managing categories and users
2023-04-16	Integration with OpenAI API to generate content
2023-04-23	Interface for standard user, generating content by categories and properties, blog
2023-04-30	Final report

Future Work

This document covers only beta version of the project. Plans for future might include:

- In order to launch a production version, the infrastructure needs to be upgraded to support at least 10 000 daily users. The infrastructure should be scalable to handle usage peaks.
- Monetization strategy should be developed to cover expenses of hosting and usage of OpenAI API. SaaS model would likely work the best, users could be charged monthly or based on the usage of the service.
- Image rendering, based on the content of generated textual content could be developed as an additional feature in future.