János Kodolányi University

Department of Informatics

PUBLICATION

Application of Artificial Intelligence in Social Media

Social AI - Part 2 – Implementation

Supervisors :

Dr. László Pitlik (https://orcid.org/0000-0001-5819-0319),

Dr. János Rikk (https://orcid.org/0000-0002-3846-6661)

Created by : István Vancsura (https://orcid.org/0000-0002-5402-8186)

Budapest

2023

Table of Contents

LIST OF ABBREVIATIONS

- DevOps: Development and Operations
- CI: Continuous Integration
- CD: Continuous Deployment
- API: Application Programming Interface
- DNS: Domain Name System (Domain Name Server)
- ID: Identifier
- OAuth: Open Authorization
- CORS: Cross-Origin Resource Sharing
- NPM: Node Package Manager
- PIP: Package Installer for Python
- URL: Uniform Resource Locator
- CSS: Cascading Style Sheets
- ENV: Environment
- JS: JavaScript (Programming Language)
- JSX: JavaScript XML
- JSON: JavaScript Object Notation
- YAML: YAML Ain't Markup Language
- CNAME: Canonical Name
- TTL: Time to Live
- WWW: World Wide Web
- UUID: Universally Unique Identifier
- SDK: Software Development Kit
- GSI: Google Sign-In
- DB: Database
- NS: Name Server
- HTTP: Hypertext Transfer Protocol
- HTTPS: Hypertext Transfer Protocol Secure
- HTML: Hypertext Markup Language
- AI: Artificial intelligence

- PaaS: Platform as a Service
- On-prem: On-premises (Local infrastructure)
- IaaS: Infrastructure as a Service
- SaaS: Software as a Service
- GTC: General Terms and Conditions
- GDPR: General Data Protection Regulation
- ChatGPT: Chat Generative Pre-trained Transformer

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Social AI - Part 1 – Planning: https://miau.my-x.hu/miau/305/Social_AI_Part1.pdf

The aim of my thesis is to demonstrate how to develop software containing artificial intelligence (ChatGPT) based features for managing social media platforms (e.g. , Facebook), partially automation content creation , and deploying it using cloud technology (Azure).

I will develop a web application, as it can be accessible across various operating systems (e.g. , Windows, iOS, Android).

Throughout the development , I will utilize and integrate several Microsoft (eg ., Azure Functions, Azure DevOps), Google (eg ., Identity Platform), Meta (eg ., Meta for Developers) products and services , detailed in Table 1 and elaborated in section 1.1.

In development process , I will employ numerous programming (eg ., Python, JavaScript), descriptive (eg ., HTML, CSS), and query (eg ., SQL) languages , summarized in Table 1. Subsequent chapters will extensively cover software design (see Chapter 1), software development (see Chapter 2), and deploying the software in the Microsoft Azure public cloud infrastructure (see Chapter 3). Emphasis on security will be maintained throughout both software development and deployment due to me specialization in IT security (refer to Chapter 4).

Furthermore , I aim to keep the operational costs of the application low (approximately 0 HUF/ month). During deployment , I will apply DevOps methodology , including various practices such as CI, CD, and necessary tools like Azure DevOps, GitHub.

At the conclusion of the documentation , I will discuss the testing results (refer to Chapter 5) and outline potential avenues for further development (see the " Conclusion " chapter).

1. THE DESIGN OF THE SOFTWARE

1.1. Creating a local development environment

The process of setting up the local development environment is to be interpreted in the case of the Windows 10 operating system.

The following MSI installer must be installed on the operating system, this installer contains the Azure Functions Core Tools in order to be able to start the Azure Function in the local development environment:
https://go.microsoft.com/fwlink/?linkid=2174087

The version of Python 3.10, which can be downloaded from the following page, must be installed on the operating system, which is also a dependency for starting the Azure Function:
https://www.python.org/downloads/release/python-3100/

The version of Node.js v20.2.0 that can be downloaded from the following page must be installed in order to be able to start the React Frontend application on the local development environment:
https://nodejs.org/en

You need to clone or download the React Frontend application source code from the private Azure DevOps repo below:
https://dev.azure.com/antarax/_git/OpenAI%20React

To start the React Frontend application, you need to run the following 3 commands:

1. npm install
2. npm run build
3. npm run start

You need to clone or download the Python Azure Function source code from the private Github repo below:
https://github.com/Anttarax/AzureFunction

To start the Python Azure Function, you need to run the following 2 commands:

1. pip install -r requirements.txt
2. func start

Source :
https://learn.microsoft.com/en-us/azure/azure-functions/functions-run-local?tabs=macos%2Cisolated-process%2Cnode-v4%2Cpython-v2%2Chttp-trigger%2Ccontainer-apps&pivots= programming-language-python

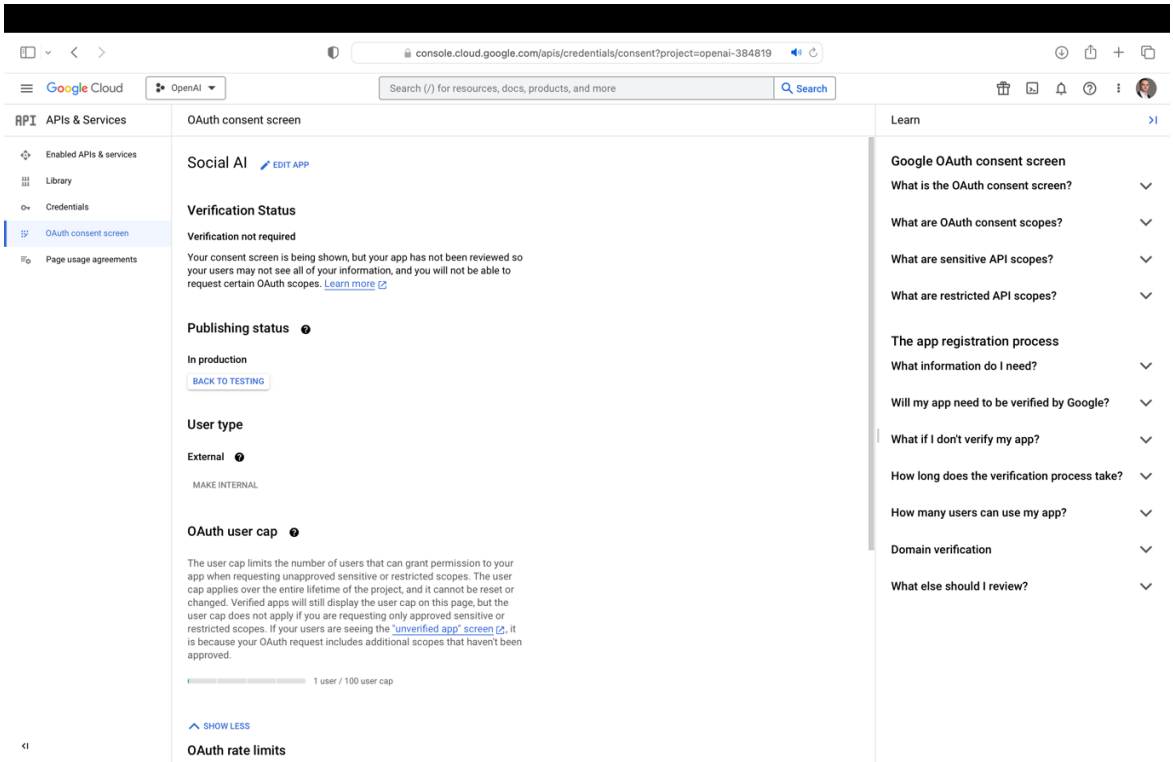
https://create-react-app.dev/docs/getting-started/

1.2. Google Cloud setup

In order to implement Google authentication and to allow users to log in to the application with their Google account, an OAuth consent screen had to be created on the Google Cloud Platform, and then an OAuth 2.0 Client ID had to be created. Google Cloud generated the Client ID and the associated Client secret during creation. During implementation, I placed them in the source code of the Frontend React application.

The page where Google authentication was implemented: https://socialai.progeurope.hu/login

In the settings of the OAuth consent screen, the Publishing status had to be set to the production status, as this allows anyone (not only developers and testers) to be able to log in. Since I only ask users for the most common, basic information (public name, public email) when logging in, there was no need to perform a verification process for the application, as Google does not require this in such cases.



1. Figure: Google Cloud OAuth consent screen settings

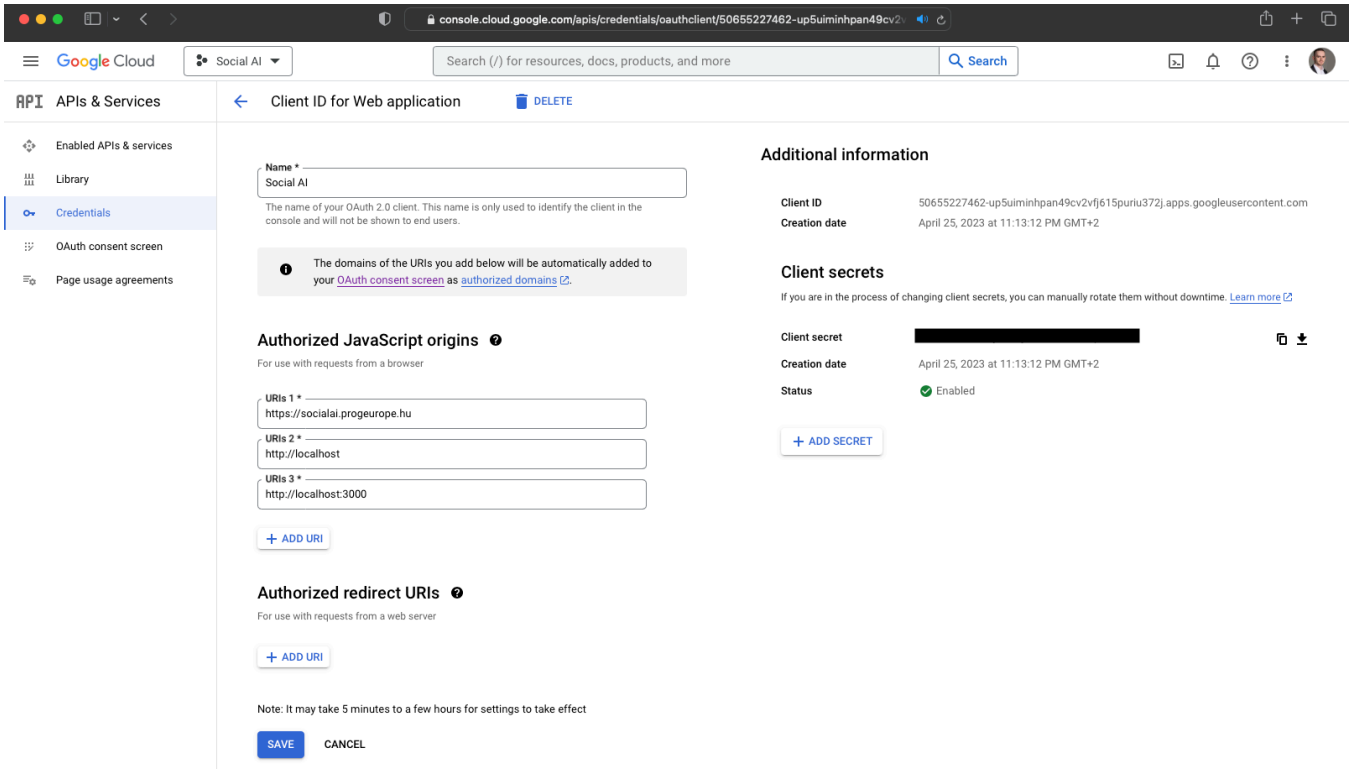
Source: https://console.cloud.google.com/apis/credentials/consent?project=openai-384819

After that, I was able to start creating the Credentials for the previously created OAuth consent screen. When creating it, you had to select the type of application, which in this case is Web application type.

Since it was implemented on the Frontend side, the URLs of the different environments had to be added to the following authorization list:

Authorized JavaScript origins:

- URIs 1: https://socialai.progeurope.hu (Live environment)
- URIs 2: http://localhost (Local development environment)
- URIs 2: http://localhost:3000 (Local development environment)



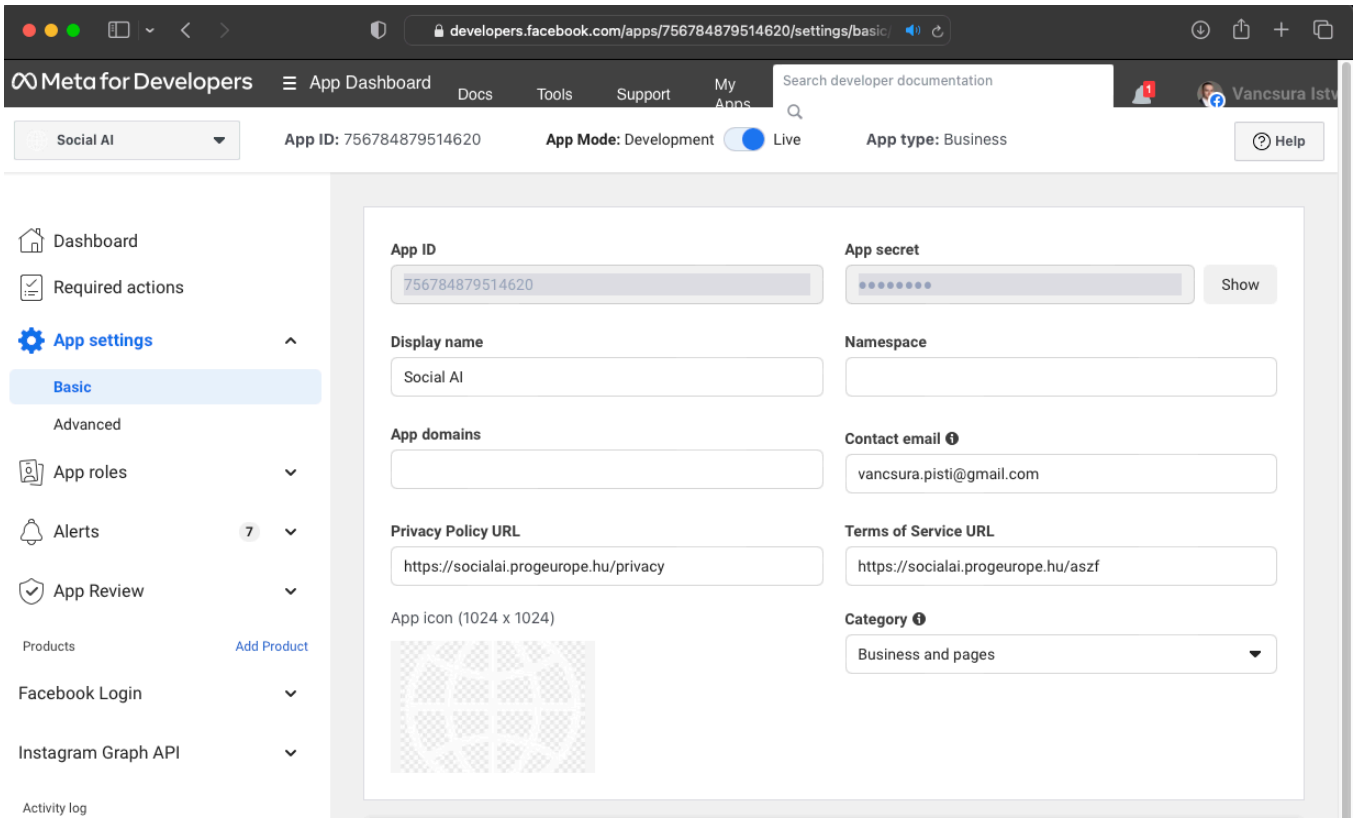
2. Figure: Google Cloud OAuth 2.0 Client ID settings

Source: https://console.cloud.google.com/apis/credentials/oauthclient/50655227462-up5uiminhpan49cv2vfj615puriu372j.apps.googleusercontent.com?hl=hu&project=openai-384819

1.3. Meta for Developers

In order to implement the feature that allows the application to create and publish content on behalf of the user to Facebook , I implemented the previously mentioned Facebook SDK in the Frontend React application. In order for this implementation to work properly, as with Google authentication, a Client (App) ID and the corresponding Secret are also required here, which I also placed in the source code. During creation, I chose the Business type App and made the Facebook Login settings detailed below.

- Privacy Policy URL: https://socialai.progeurope.hu/privacy
- Terms of Service URL: https://socialai.progeurope.hu/aszf
- Website Site URL: https://socialai.progeurope.hu/
- Contact email and Data Protection Officer contact information: I have filled in my own personal information.



3. Figure: Meta for Developers APP settings

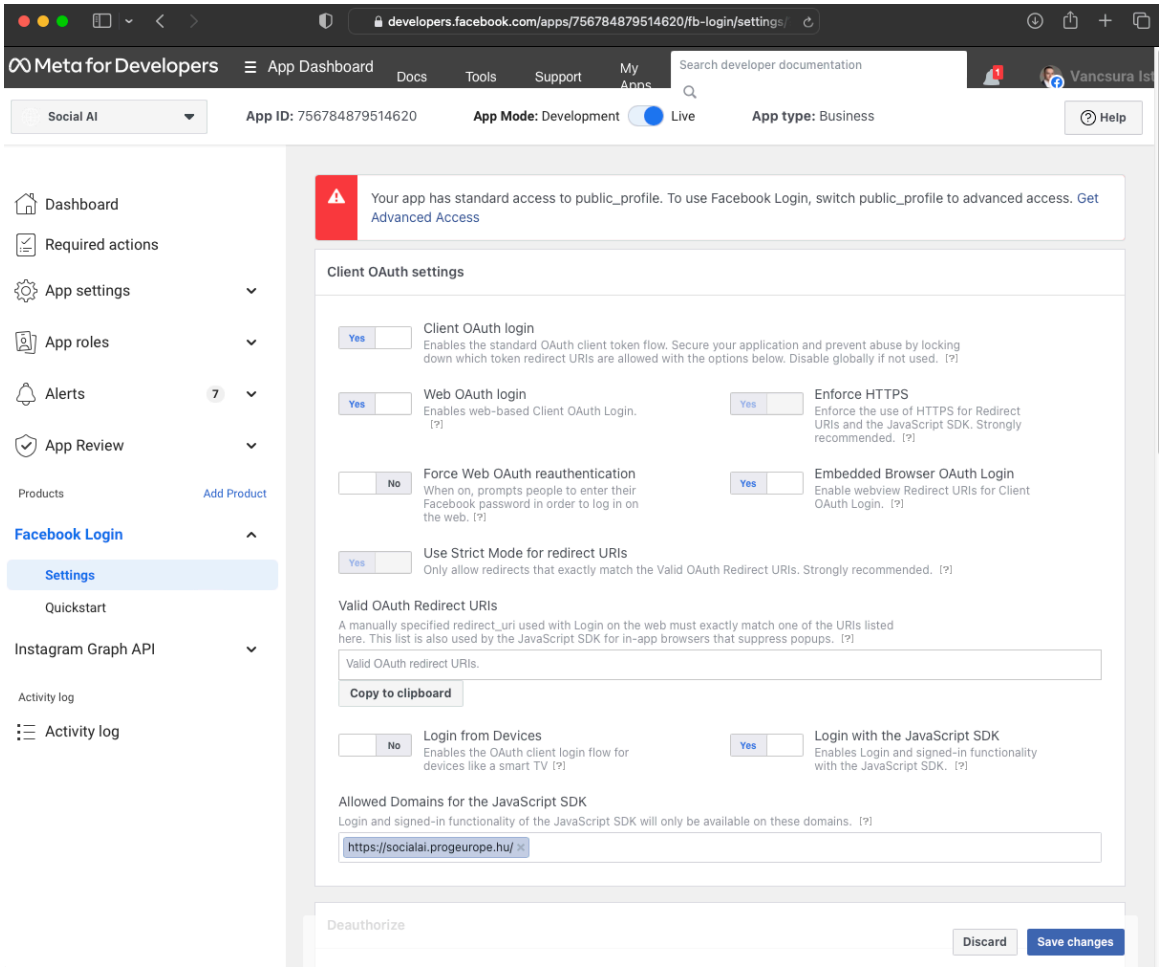
Source:
https://developers.facebook.com/apps/756784879514620/settings/basic/?business_id=758453229336454

Allowed Domains for the JavaScript SDK:

- https://socialai.progeurope.hu/ (Live environment)

Unlike the Google authentication settings, the local development environment does not need to be added to the whitelist, as it is mandatory to enter HTTPS (encrypted) URLs, however, if the App Mode is set to Development, it works, but in this case the live environment is not good. There are 2 potential solutions for this in the future:

1. We take the local development environment URL as https and issue a certificate locally, and then implement it in the local development environment.
2. We create a separate Meta for Developers APP for the developer environment, which remains in Development mode, while the original one serves the production environment in Live mode.



4. Figure: Meta for Developers APP settings

Source: https://developers.facebook.com/apps/756784879514620/fb-login/settings/?business_id=758453229336454

1.4. ChatGPT and Pexels.com implementation

I implemented OpenAI and Pexels.com API in the app. pexels.com is a platform for images that are legally free to use. The platform provides an open database, a search engine for searching and querying images, and OpenAI provides a language model called ChatGPT, which enables the generation of texts.

All of these implementations allow users to generate posts in the application, and then publish them directly from the application to their own page thanks to the Facebook integration described in the previous chapter.

The OpenAI and Pexels.com APIs have been implemented in the following two APIs of the application:

1. PostGen

Below I explain in detail the logic behind how it works:

If the HTTP request method is GET, then:

1. Based on the received data (data from Facebook integration supplemented with the user's inputs, which can be viewed in more detail in the "API documentation of the application"), the OpenAI API generates a social media post text and the keywords associated with the text for searching Pexels.com images.
2. It then randomly selects an image via the Pexels.com API based on the generated keywords. (which keywords specifically belong to the text generated in the previous step, only randomly selected after the targeted search, from the results).
3. It returns the URL of the selected image and the generated text to the Frontend application in JSON format
4. Based on the e-mail address, it queries the user from the database and increases the balance. (on which an invoicing system can be built later, which is included in the further development options)

If the HTTP request method is POST, then the received data, which contains the text of the final social media post and the URL of the image corresponding to the text (full data in the chapter "API documentation of the application") publishes the post on the Facebook page.

2. ImgGen

It was necessary to create a separate ImgGen API, as there may be cases where the text generated by PostGen is accepted by the user, but the image is not. The user can also edit the generated text, which is why a "Regenerate Image" function has been created, along with the ImgGen API endpoint.

Below I explain in detail the logic behind how it works:

1. Based on the received data (which includes the text of the final social media post, (full data in the "API documentation of the application") section, the OpenAI API generates the received keywords for the text to search for Pexels.com images.
2. It then randomly selects an image via the Pexels.com API based on the generated keywords. (which keywords specifically belong to the text generated in the previous step, only randomly selected after the targeted search, from the results).
3. It returns the URL of the selected image to the Frontend application in JSON format.

The "Generate post" and "Regenerate image" functions can be repeated indefinitely according to the user's decision, and at the end of the process, the "Finalize post and send" function is used, which means calling the POST method of the PostGen API endpoint detailed above.

I created the OpenAI API key on the following page and then placed it in the source code of the Backend system: https://platform.openai.com/account/api-keys . I did the implementation based on the documentation on the following page: https://platform.openai.com/docs/api-reference

I created the Pexels.com API key on the page below and then placed it in the source code of the Backend system: https://www.pexels.com/api/new/ . I did the implementation based on the documentation on the following page: https://www.pexels.com/api/documentation/

1.5. Support role of ChatGPT

The following examples show the versatility of ChatGPT. Using the capabilities of machine intelligence can help developers in a variety of ways (as described in the subsections below). I also attach appendices to the thesis about the sub-chapters, which contain logs of real, completed communications.

1.5.1. Swagger documentation generation

ChatGPT can analyze program code. By interpreting the codes of the REST APIs, you recognized what the endpoints are, what parameters they expect and what responses they return. It was then able to generate Swagger documentation in YAML format from the recognized information, which syntactically met the Swagger documentation requirements. Related Annex No. 8.

1.5.2. Fix algorithm syntax errors

ChatGPT is capable of identifying algorithm errors. When the code produced incorrect results, ChatGPT explained in detail where the error might be, such as pointing out missing parentheses or incorrect variable names in the code. He then made suggestions for correcting the errors, such as showing the correct syntax. Related Appendix No. 9.

1.5.3. Consulting on programming libraries.

When the project needed a spreadsheet library, ChatGPT helped me make the right choice. ChatGPT has listed possible spreadsheet libraries and explained their advantages. Related Appendix No. 10.

1.5.4. A translation with an explanation

In the case of professional documentation in a foreign language (e.g. English), ChatGPT was able not only to translate the foreign language text into Hungarian, but also to provide assistance in its interpretation and summary, thus shortening the development time. Related Appendix No. 11.

1.6. Fontend files and components

- .env : This file is used to store environment variables for local development that we cannot write directly in our frontend code, such as API keys.
- .gitignore : This file contains a list of files and folders that we want (or should) exclude from version control, for example the node_modules folder, in which the source files of the frontend dependencies are installed locally.
- index.html: This file is stored in the public folder and its main task is for the react application to render the content to be displayed to the user in the html tag in this file. In addition, its important task is to use the <script> members used in the code to connect to the Facebook and Google APIs so that we can use them in the frontend program.
- index.js: This component is the starting component of the React application, its main task is to load the other components. The html rendered by the components (which we display to the user on the website) will be displayed inside the html tag in the index.html file.
- App.jsx: This component is responsible for defining the Routes created in the React app, as well as for associating the "page" components with the given routes.
- index.css: This file defines the css classes that we use in the other components to format our frontend application.
- Footer.jsx: In this file, the Footer react component displayed at the bottom of each subpage is declared, the main function of which is to display the links to the General Terms and Conditions, Data Management Policy and the Contact page.
- Navbar.jsx: This component displays the navigation menu at the top of each subpage.
- Account.jsx: This react component is responsible for displaying the content in the "/" path. On this subpage, the user can view the main data associated with the Google account (name, email address, profile picture).
- Aszf.jsx: This component displays the Terms and Conditions in the "/aszf" path.
- Privacy .jsx: This component displays the Privacy Policy in the path "/privacy".
- Billing.jsx: This component is responsible for displaying the payment options that will be potentially implemented later in the "/billing" path.

- Facebook.jsx: This component is responsible for logging into the user's Facebook account, as well as displaying the data of logged-in users and available Facebook pages in an ordered list using the "react-table" npm package.
- Login.jsx: This component is responsible for logging in the user using the user's Google account in the "/login" path. Any other page of the application will redirect us to this "/login" route if we are not yet logged in.
- Post.jsx: This component is responsible for displaying the form needed to generate Facebook posts in the "/post" path. This is perhaps the most important component of the application, as this component is responsible for creating specific Facebook posts with the help of artificial intelligence. On this subpage, the user can generate personalized posts, modify them, and then, by pressing a button, actually share them from this app using the Facebook API.

The parts of the navigation menu and the footer and the routes they link to:

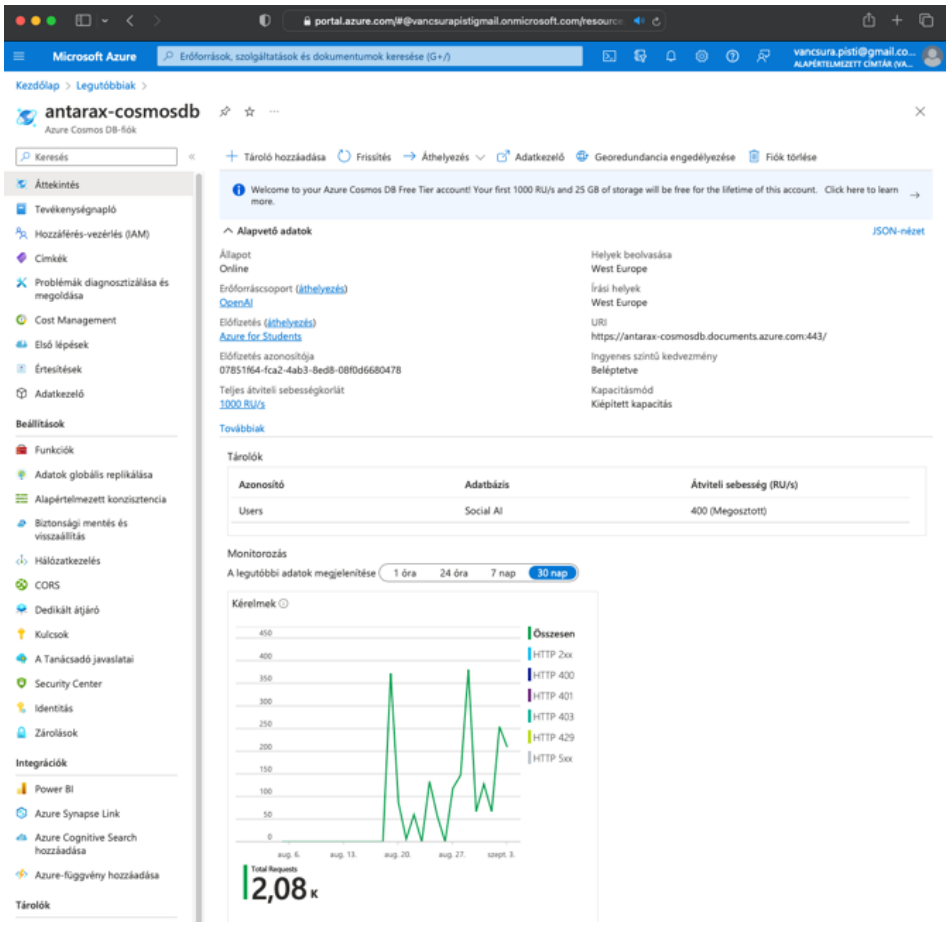
- My account: "/"
- Billing: "/billing"
- AI post generation "/post"
- Facebook master data: "/facebook"
- Data management policy: "/privacy"
- General terms and conditions: "/aszf"
- Contact: " https://progeurope.hu/hu/contact/with-map/ " (the contact page of the company, which was created during an independent project.)

2. DEPLOYING A PRODUCTION ENVIRONMENT

2.1. Create and configure Azure CosmosDB

Created Azure Cosmos DB account parameters:

- Location: Western Europe
- URI: https://antarax-cosmosdb.documents.azure.com:443/
- Total transfer rate limit: 1000 RU/s
- CORS Allowed source locations: *



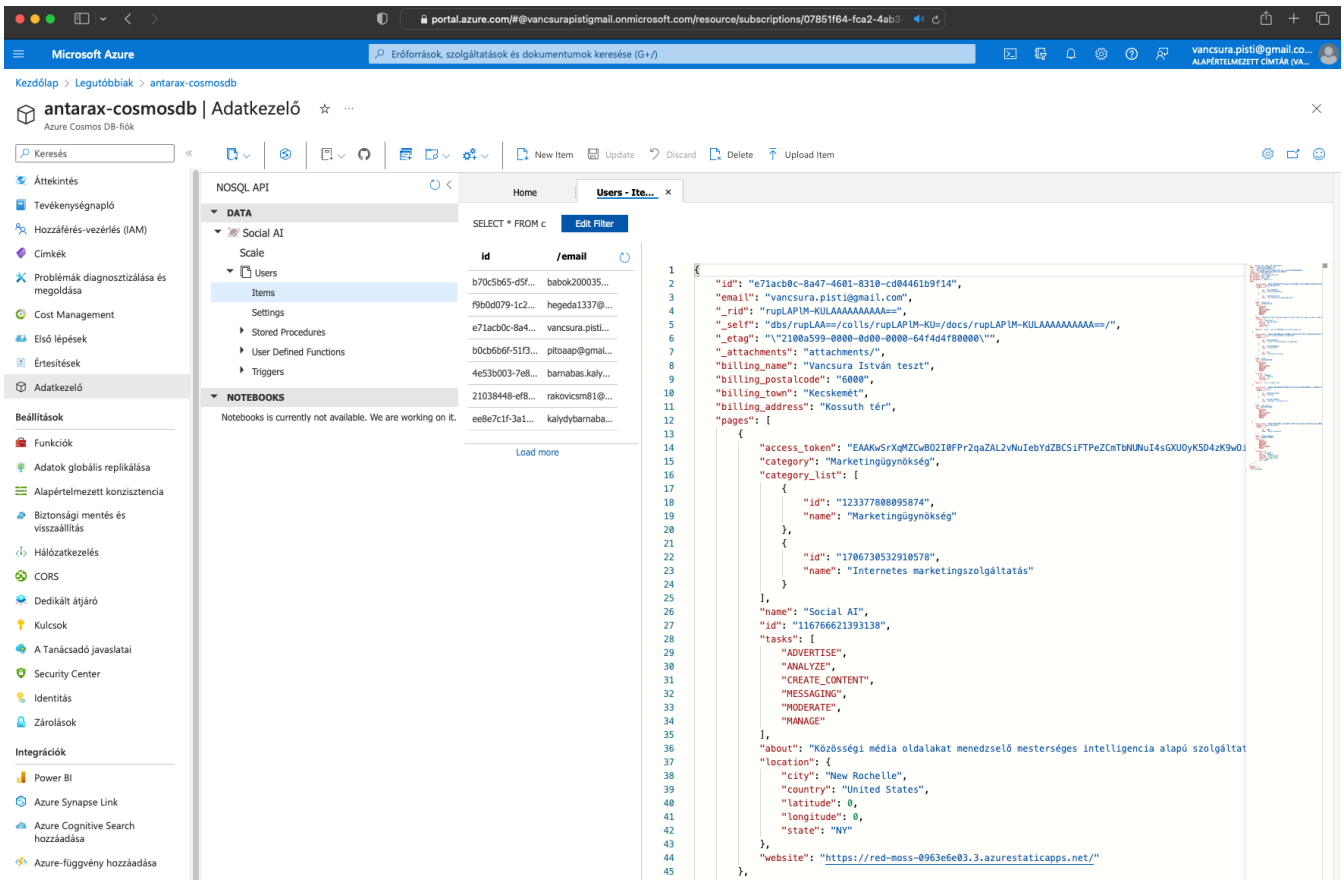
5. Figure: Azure Cosmos DB account overview

Source:
https://portal.azure.com/#@vancsurapistigmail.onmicrosoft.com/resource/subscriptions/07851f64-fca2-4ab3-8ed8-08f0d6680478/resourceGroups/OpenAI/providers/Microsoft.DocumentDb/databaseAccounts/antarax-cosmosdb/ overview

A database called Social AI has been created in the Azure Cosmos DB account with the following parameters:

- Maximum reception speed limit: 400 RU/s
- Scaling: Automatic

In the created database called Social AI, a container called Users was created, in which container the application can store users and data belonging to users in JSON format. A separate JSON document is created for each user. The storage partition key is the following attribute: /email



6. Figure: Azure Cosmos DB data manager

Source:

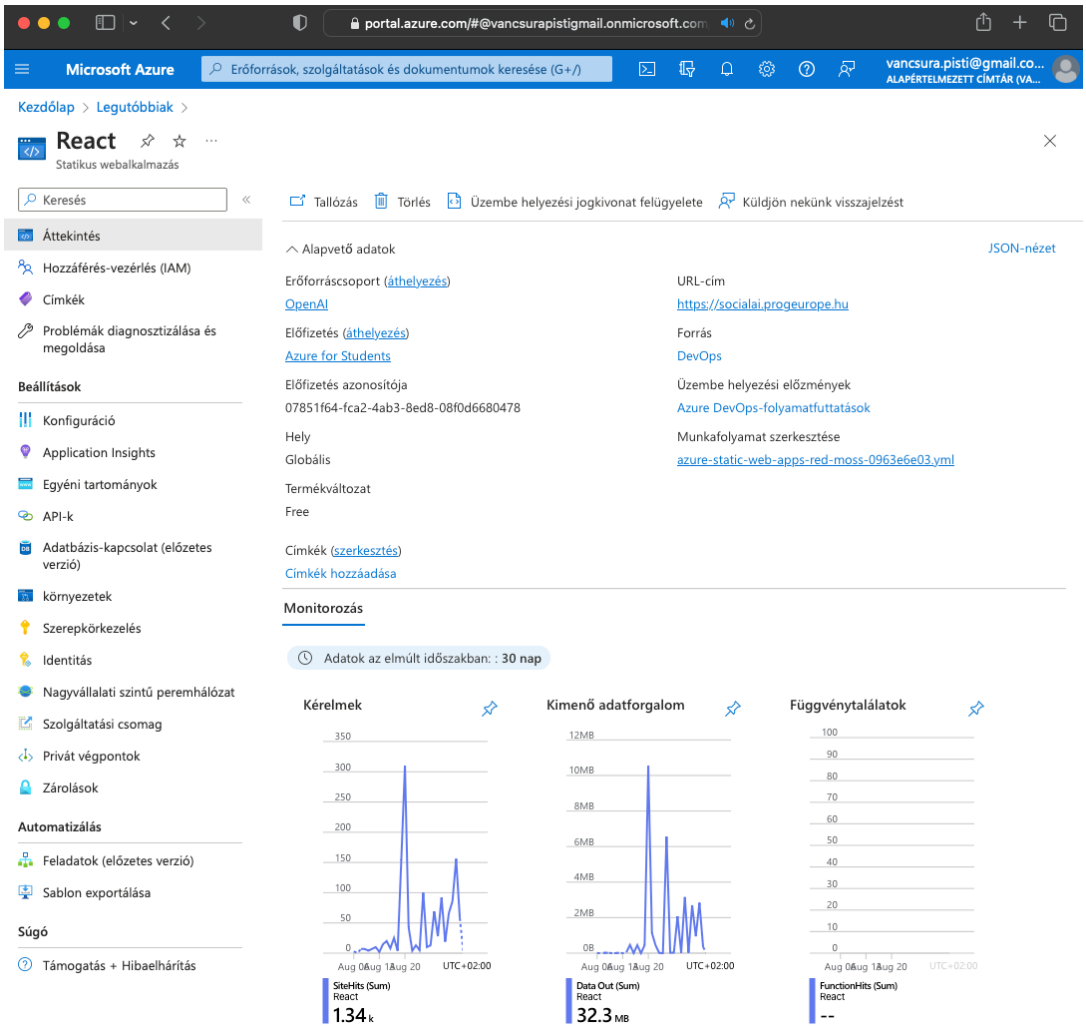
https://portal.azure.com/#@vancsurapistigmail.onmicrosoft.com/resource/subscriptions/07851f64-fca2-4ab3-8ed8-08f0d6680478/resourceGroups/OpenAI/providers/Microsoft.DocumentDb/databaseAccounts/antarax-cosmosdb/ dataExplorer

2.2. Create and configure Azure Static Web Apps

- Location: Western Europe
- URL address: https://socialai.progeurope.hu

Custom domains:

- red-moss-0963e6e03.3.azurestaticapps.net (Automatically generated)
- www.socialai.progeurope.hu
- socialai.progeurope.hu (Default)



7. Figure: Azure Static Web Apps overview

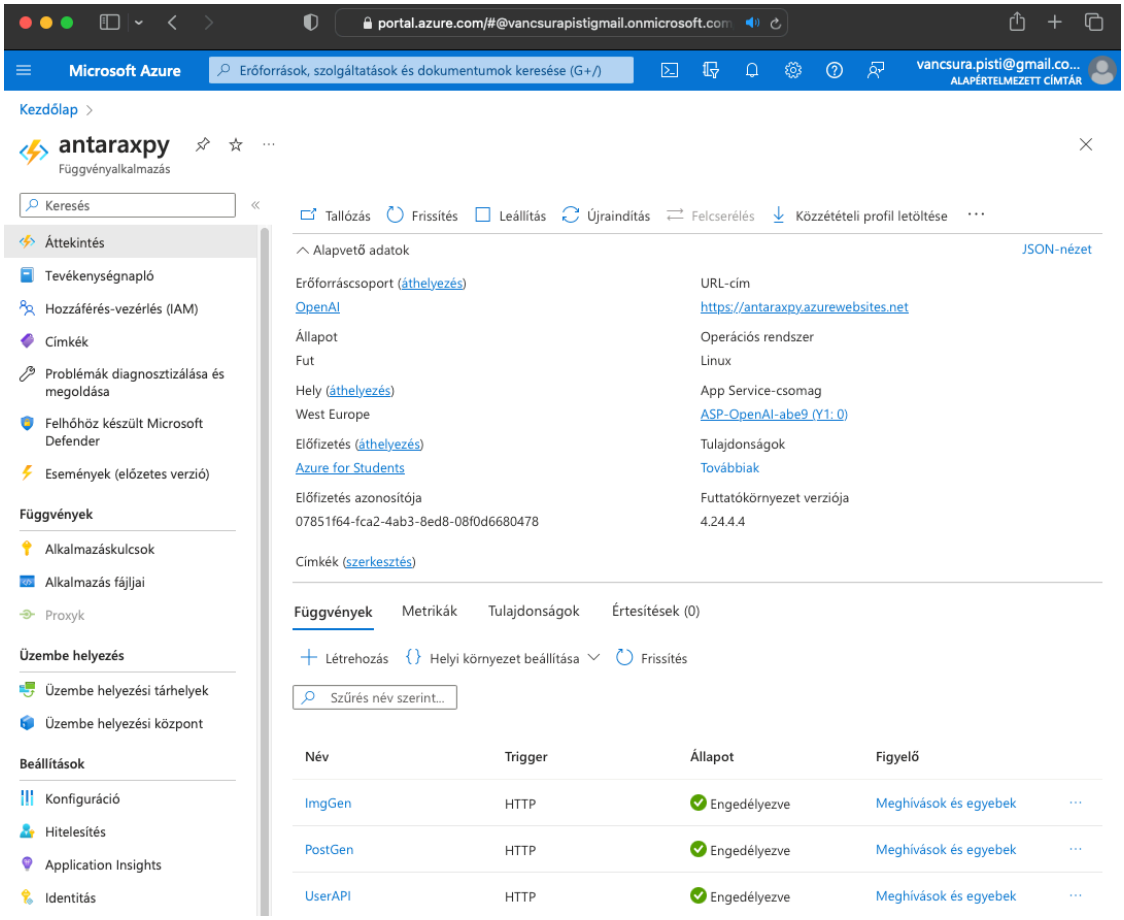
Source:
https://portal.azure.com/#@vancsurapistigmail.onmicrosoft.com/resource/subscriptions/07851f64-fca2-4ab3-8ed8-08f0d6680478/resourceGroups/OpenAI/providers/Microsoft.Web/staticSites/React/staticsite

2.3. Create and configure Azure Functions

- Location: Western Europe
- Operating system: Linux
- URL: https://antaraxpy.azurewebsites.net
- CORS Authorized Origins: *

Created functions according to API plans and Swagger documentation:

- https://antaraxpy.azurewebsites.net/api/ImgGen
- https://antaraxpy.azurewebsites.net/api/PostGen
- https://antaraxpy.azurewebsites.net/api/UserAPI



8. Figure: Azure Functions overview

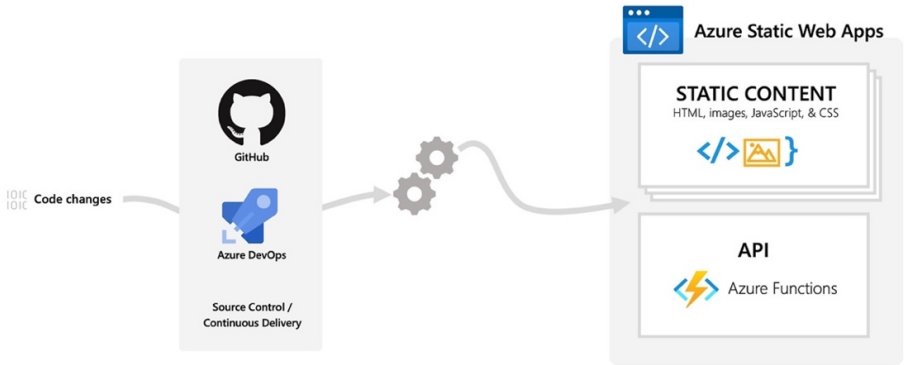
Source:
https://portal.azure.com/#@vancsurapistigmail.onmicrosoft.com/resource/subscriptions/07851f64-fca2-4ab3-8ed8-08f0d6680478/resourceGroups/OpenAI/providers/Microsoft.Web/sites/antaraxpy/appServices

2.4. Documentation of GitHub Actions and Azure DevOps CI/CD processes

In this subsection, I document the CI/ CD .YAML definition files of the application components, and the complete .YAML files are included in the Source Codes.zip attachment of the thesis.

The definition files can be found at the following accesses:

- Social AI React Frontend/azure-static-web-apps-red-moss-0963e6e03.yml
- Social AI Functions Python Backend/.github/workflows/main_antaraxpy.yml



9. Figure: DevOps Architecture Figure

Source: https://azure.microsoft.com/en-us/products/app-service/static

CI/CD pipeline parameters for each component:

- Trigger: Main branch
  So this means that when the developer pushes the source code changes of the given component to the main branch, the CI/CD pipeline starts automatically, i.e. the automated process that prepares (in technical terms, builds) the new version of the program, and then the target environment automatically installs the new version.
- Agent: Microsoft Hosted Agent
  So this means that the automated CI/CD process takes place on a server run and operated by Microsoft. The other option would have been to use a Self-Hosted Agent, so I provide the server environment to run the automations, but this would have been expensive and not justified.
- vmImage: ubuntu-latest
  So the automated process runs on a Linux operating system.

Frontend React Application CI Process Steps:

1. task: NodeTool
    a. versionspec : '20.x'

    Explanation: You are installing Node.js with version 20.x, as it is a basic requirement for a React-type application.
2. script: npm install

    Explanation: Installation of the dependencies of the project, which is the dependency of the next step.
3. script: npm run build

    Explanation: Performs the build process of the project's source code.
4. task: CopyPublishBuildArtifact
    a. CopyRoot: 'build'
    b. Contents : ' *'

    Explanation: After the build process, the created files (located in the build folder) are copied and published to the artifacts used by CI/CD.

Frontend React application CD process (Deploy):

5. task: Deploy Azure Static Web App
    a. Working directory: $(System.DefaultWorkingDirectory)
    b. App location: _React CI CD/build

    Explanation: You are installing the application from the folder containing the React application build.

Azure Functions Python Backend System CI Process Steps:

1. Setup Python version

    Explanation: Sets the Python version to the specified PYTHON_VERSION, then creates and activates a virtual Python environment.
2. Install dependencies: pip install -r requirements.txt

    Explanation: Installs project dependencies based on requirements.txt.

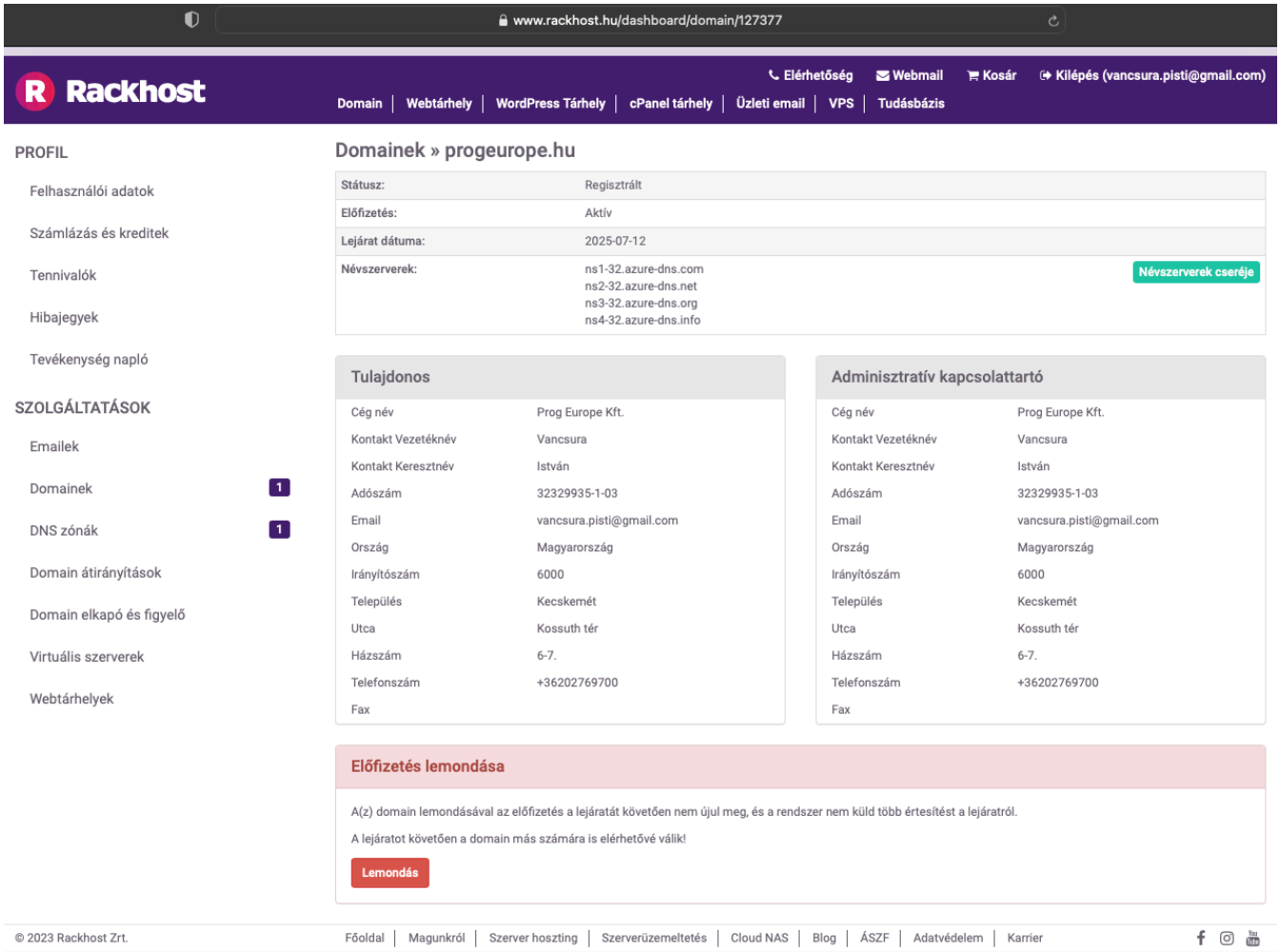
Azure Functions Backend system CD process (Deploy):

3. Downloads the artifact created by the build workflow.
4. Azure Functions will be installed.

2.5. Domain , Azure DNS zone

I purchased the domain progeurope.hu from the Rackhost Zrt. (https://www.rackhost.hu/) domain registrar, and then delegated its control to the Azure DNS zone service by setting the name servers that were given to me by the Azure portal.

- Nameserver#1: ns1-32.azure-dns.com
- Nameserver#2: ns2-32.azure-dns.net
- Nameserver#3: ns3-32.azure-dns.org
- Nameserver#4: ns4-32.azure-dns.info



10. Figure: Domain control interface, Rackhost

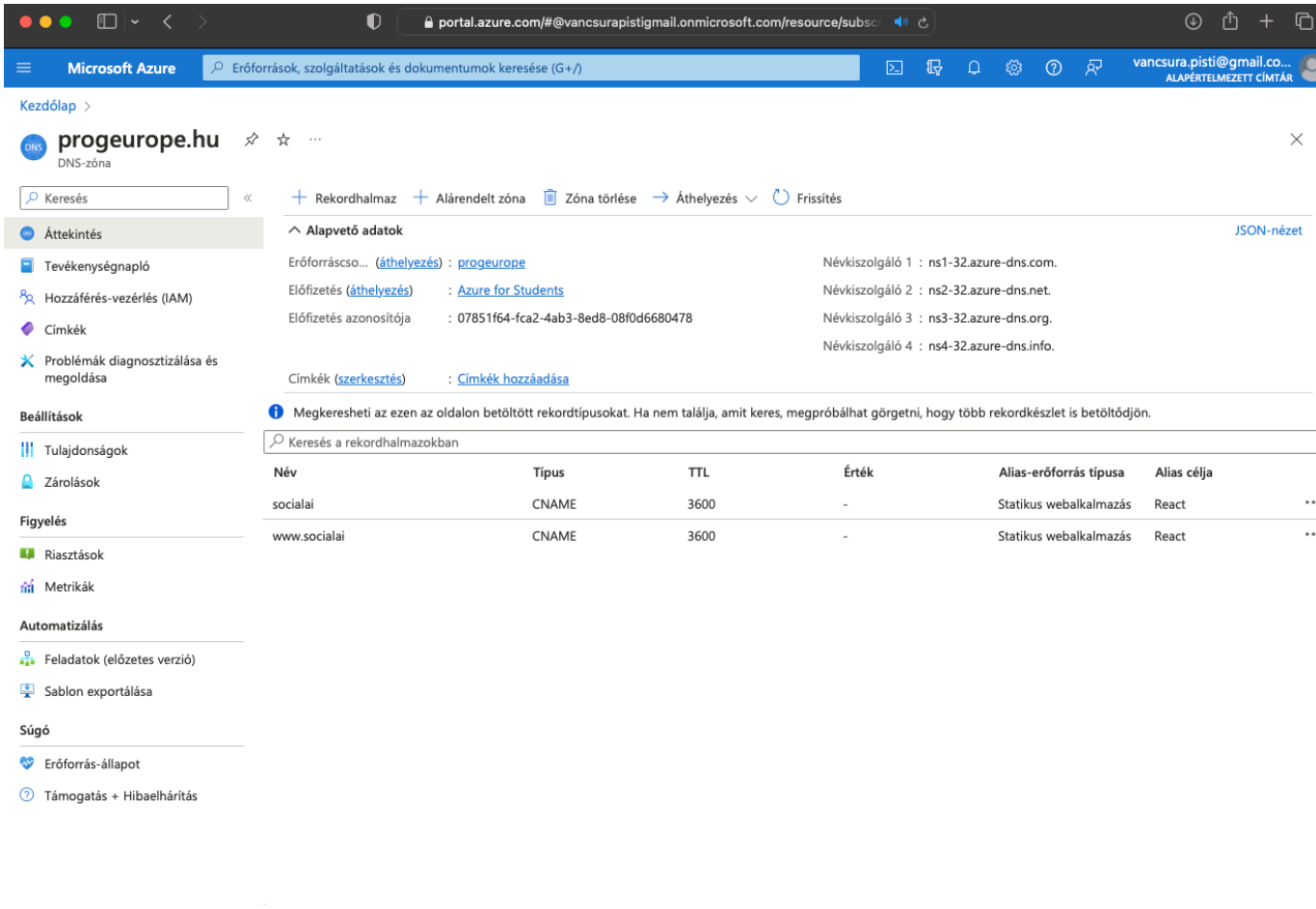
Source: https://www.rackhost.hu/dashboard/domain/127377

2.6. Create and configure an Azure DNS zone

I created a subdomain called socialai for the application by adding the following (see Table 4 and Figure 14) DNS records:

1. Table: Azure DNS zone, configured DNS records

Name	Type	TTL	Value	Alias resource type	Alias purpose
social	CNAME	3600	-	Static web application	React
www.socialai	CNAME	3600	-	Static web application	React



11. Figure: Azure DNS zone settings

Source:
https://portal.azure.com/#@vancsurapistigmail.onmicrosoft.com/resource/subscriptions/07851f64-fca2-4ab3-8ed8-08f0d6680478/resourceGroups/progeurope/providers/Microsoft.Network/dnsZones/progeurope.hu/ overview

SUMMARY

Evaluation of Results :

I consider the following aspects successful :

- The successful deployment of the entire development content (all components , including the Frontend application, Backend system , and database) into a live.
- Managing to keep the deployment costs close to zero (refer to Appendices 3, 4, 5, 6) while ensuring the application's scalability .
- Drunk implementation of DevOps CI/CD processes , automating the installation of the application's new version after the upload of developed source code for further developments (see Chapter 3.4).

I consider the following aspects partially successful :

- Creation of text generation and image selection algorithms (see Chapter 2.4).
- Facebook integration , as currently only test users can use the application until the Facebook Business verification and App Review processes are completed .

Further Development Opportunities :

- Integration of additional social media platforms.
- Implementation and automation of credit cards payment /bank transfer system .
- Outsourcing of Secret information from Backend source code to Azure DevOps/GitHub system , similar to the Frontend system .
- Implementation and automation of billing systems .
- Development of an Autopilot feature allowing fully your vending machine content creation .
- Integration of additional authentication platforms beyond Google accounts for login.
- Further enhancement of the page appearance (eg ., colors , images , layout , logos).
- User- friendly improvements on the page.
- Further development and enhancement of text generation and image selection algorithms .
- Completion of Facebook Business verification and App Review processes .

REFERENCE LIST

- Microsoft: Learn about free Azure services URL: https://azure.microsoft.com/en-us/pricing/free-services/ Retrieved 09/15/2023
- Microsoft: Azure Functions. Execution of an event-driven serverless code function with a full development environment, URL: https://azure.microsoft.com/hu-hu/products/functions/ Downloaded 2023.09.15.
- Microsoft: Azure Functions Guide for Python Developers, 2023-05-25, URL: https://learn.microsoft.com/hu-hu/azure/azure-functions/functions-reference-python?pivots=python-mode-decorators&tabs =asgi%2Capplication-letter Downloaded 15.09.2023.
- OpenAI: Introducing ChatGPT, URL: https://openai.com/blog/chatgpt Downloaded 15.09.2023.
- OpenAI: Libraries, URL: https://platform.openai.com/docs/libraries Downloaded 15.09.2023
- Microsoft: Build native cloud applications with a fast NoSQL database and a free Azure account, URL: https://azure.microsoft.com/hu-hu/free/cosmos-db/ Downloaded 09/15/2023.
- Microsoft: Connecting Azure Functions to Azure Cosmos DB using Visual Studio Code 06/24/2023 https://learn.microsoft.com/en-us/azure/azure-functions/functions-add-output-binding-cosmos-db-vs-code?tabs=in-process%2Cv1&pivots=programming-language-python Downloaded 15.09.2023
- React: The library for web and native user interfaces, URL: https://react.dev Downloaded 2023.09.15.
- Microsoft: Static Web Apps, URL: https://azure.microsoft.com/hu-hu/products/app-service/static Downloaded 15.09.2023.
- Microsoft: Quickstart: Build your first static website with Azure Static Web Apps. 09.07.2023, URL: https://learn.microsoft.com/hu-hu/azure/static-web-apps/getting-started?tabs=react Downloaded 09.15.2023.
- Google: Identity Platform, URL: https://cloud.google.com/identity-platform Downloaded 15.09.2023.

- Microsoft: Azure DevOps : With modern development tools you can plan better, collaborate more efficiently and deliver your finished products faster, URL: https://azure.microsoft.com/hu-hu/products/devops Downloaded 09/15/2023.
- GitHub: GitHub Actions: Automate your workflow from idea to production. https://github.com/features/actions Downloaded 15.09.2023.
- Microsoft: Azure Pipelines: Continuous build, test and deployment for any platform and cloud, URL: https://azure.microsoft.com/hu-hu/products/devops/pipelines Downloaded 15.09.2023.
- Facebook: Long-Lived Access Tokens, URL: https://developers.facebook.com/docs/facebook-login/guides/access-tokens/get-long-lived Downloaded 15.09.2023.
- Microsoft: Local development Azure Functions using Core Tools. 28.08.2023 https://learn.microsoft.com/hu-hu/azure/azure-functions/functions-run-local?tabs=macos%2Cisolated-process%2Cnode-v4%2Cpython-v2%2Chttp-trigger%2Ccontainer-apps&pivots= programming-language-python Downloaded 09/15/2023
- React: Getting started, URL: https://create-react-app.dev/docs/getting-started/ Downloaded 2023.09.15.
- Microsoft: Azure DNS: You can run your DNS domains in Azure with Azure DNS, URL: https://azure.microsoft.com/hu-hu/products/dns Downloaded 09/15/2023.
- OpenAI: Welcome to the OpenAI platform, URL: https://platform.openai.com/overview Downloaded 15.09.2023.
- Pexels: Start building with the power of Pexels, URL: https://www.pexels.com/api/ Downloaded 15.09.2023.
- Facebook: Create a Facebook page, URL: https://hu-hu.facebook.com/business/help/104002523024878 Downloaded 15.09.2023.
- Pexels: API Introduction, URL: https://www.pexels.com/api/documentation/ Downloaded 15.09.2023.
- OpenAI: Introduction, URL: https://platform.openai.com/docs/api-reference Downloaded on 15.09.2023.
- Microsoft: Shared responsibilities in the cloud , URL: https://learn.microsoft.com/hu-hu/azure/security/fundamentals/shared-responsibility Downloaded 15.09.2023.

- Microsoft: Azure Network Architecture. 01.06.2023 https://learn.microsoft.com/hu-hu/azure/security/fundamentals/infrastructure-network Downloaded 15.09.2023.
- Microsoft: Azure infrastructure security. 01.06.2023, URL: https://learn.microsoft.com/hu-hu/azure/security/fundamentals/infrastructure Downloaded 15.09.2023.
- Microsoft: Azure facilities, facilities and physical security. 25.03.2023, URL: https://learn.microsoft.com/hu-hu/azure/security/fundamentals/physical-security Downloaded 15.09.2023
- Edit Szilvia Rubóczki: *"The most revolutionary and perhaps the most popular IT infrastructure solution of the decade is cloud technology.", URL: http://lib.uni-obuda.hu/sites/lib.uni-obuda.hu/files/Ruboczki_Edit_Szilvia_ertekezes.pdf* Page 59 Downloaded 10.07.2023

BIBLIOGRAPHY

- Imre Petkovics, Ármin Petkovics: The future of IT is in the cloud, 2010, URL: http://www.vmtt.org.rs/mtn2010/492_502_Petkovics_A.pdf Downloaded: 2023.10.07.
- József Váradi: Data protection in cloud computing, 2012, URL: http://publikaciok.lib.uni-corvinus.hu/publikus/tdk/20120326135701.pdf Downloaded: 2023.10.07.
- Rubóczki Edit Szilvia: The importance of users' security awareness when introducing public cloud services at large companies, 2019, URL: http://lib.uni-obuda.hu/sites/lib.uni-obuda.hu/files/Ruboczki_Edit_Szilvia_ertekezes.pdf Downloaded 2023.10.07 .
- The Hungarian National Bank 4/2019. (IV.1) on the use of community and public cloud services, URL: https://www.mnb.hu/letoltes/4-2019-felho.pdf Downloaded: 2023.10.07.
- Zsolt Katonka: The role of artificial intelligence in everyday life, 2017, URL: http://dolgozattar.uni-bge.hu/12581/1/katonka_zsolt_2017jún_publikus.pdf. Downloaded on 07.10.2023

- Dr. László Dudás: Artificial intelligence, URL: http://ait.iit.uni-miskolc.hu/~dudas/MIEAok/MIea1.PDF Downloaded 2023.10.07.
- János Rikk, László Pitlik: ChatGPT-experiments-programming, 2023, URL: https://m2.mtmt.hu/gui2/?mode=browse&params=publication;34071055 Downloaded 2023.10.08.
- László Pitlik: Token-based chatGPT3 – Example, 2023, URL: https://m2.mtmt.hu/gui2/?mode=browse&params=publication;34071014 Downloaded 2023.10.08.
- Microsoft: Shared responsibility in the cloud, 29.09.2023, URL: https://learn.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility Downloaded 15.09.2023.
- Microsoft: Azure infrastructure security, 01.02.2023, URL: https://learn.microsoft.com/en-us/azure/security/fundamentals/infrastructure Retrieved 15.09.2023

APPENDIX

1. attachment: Social AI API.yaml

2. attachment: User.json

3. attachment: Microsoft account.pdf

4. attachment: Google account.pdf

5. attachment: Rackhost invoice.pdf

6. attachment: OpenAI invoice.pdf

7. attachment: Source codes.zip

8. attachment: ChatGPT Swagger documentation generation.html

9. attachment: Fix ChatGPT algorithm syntax errors.html

10. appendix: Advice on ChatGPT programming libraries.html

11. attachment: translation with ChatGPT explanation.html