BeiGene AI Chatbot Portal Documentation

Table of Contents

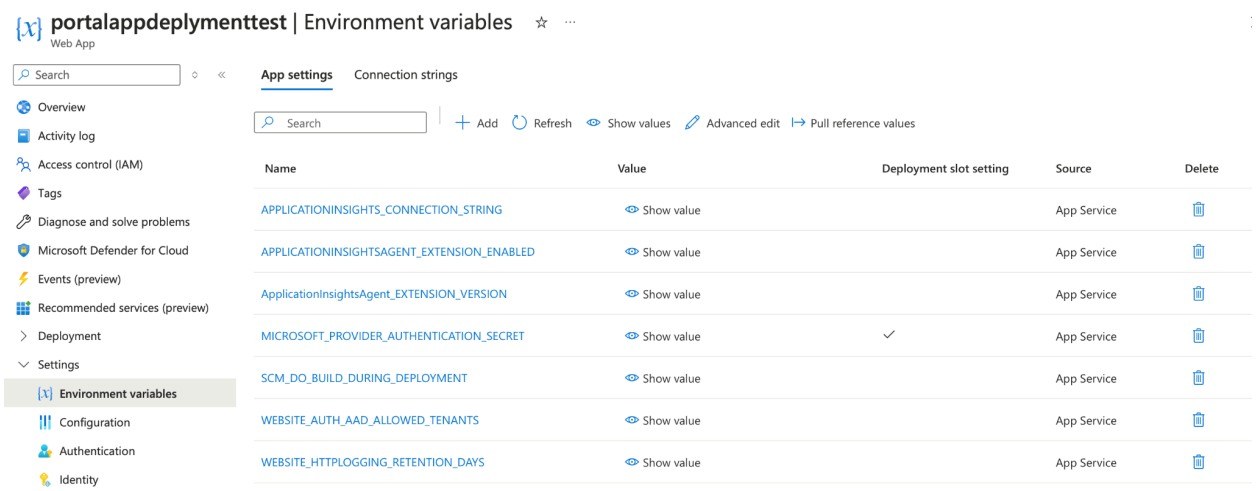
Overview

This is a detailed report about the development and configuration of the Beigene AI Chatbot Portal. It is a website designed to streamline interaction with AI-powered chatbots designed for BeiGene users. The portal is built on a modern tech stack, utilizing Next.js for front-end development, NextAuth for authentication, and Azure Active Directory for identity management. Deployment is handled via Azure App Services, offering seamless integration with GitHub or Visual Studio Code for codebase updates. The portal includes features such as dynamic chatbot filtering, responsive design, and role-based access control to improve usability and accessibility.

Environment Variables

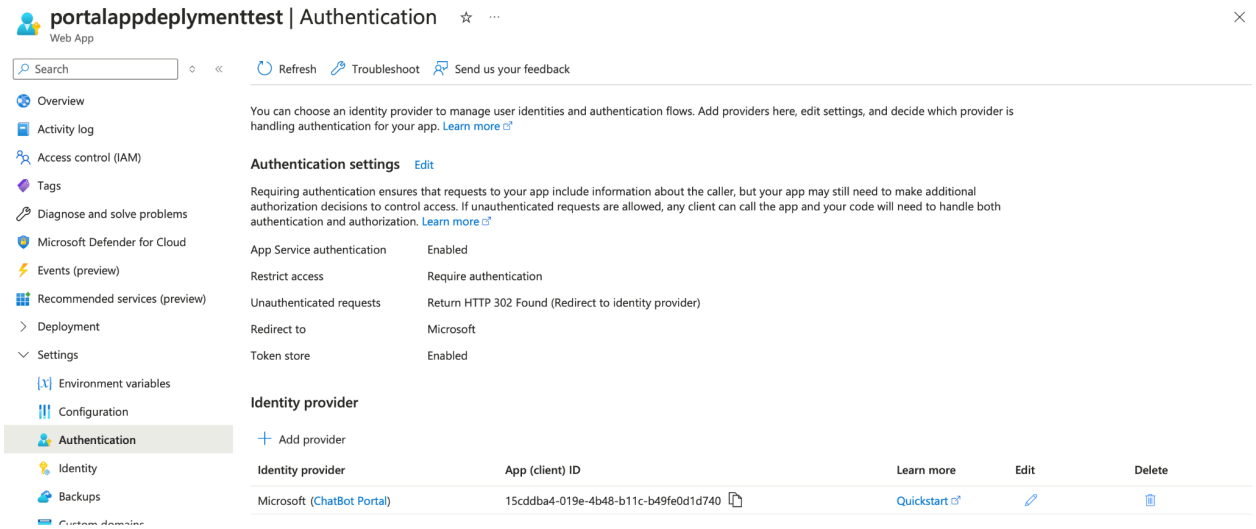
When first downloading the source code, make sure the .env file is populated with the correct environment variables. If a .env does not exist, make a copy of .env.example and fill out the blanks. Change the URL to the one used for deployment. The values of the other variables (AZURE_AD_CLIENT_ID, AZURE_AD_CLIENT_SECRET, and AZURE_AD_TENANT_ID) can be found in

the Azure App Service for the website.



In the Environment variables section, we can get the following variables:

- AZURE_AD_CLIENT_SECRET = MICROSOFT_PROVIDER_AUTHENTICATION_SECRET
- AZURE_AD_TENANT_ID = WEBSITE_AUTH_AAD_ALLOWED_TENANTS



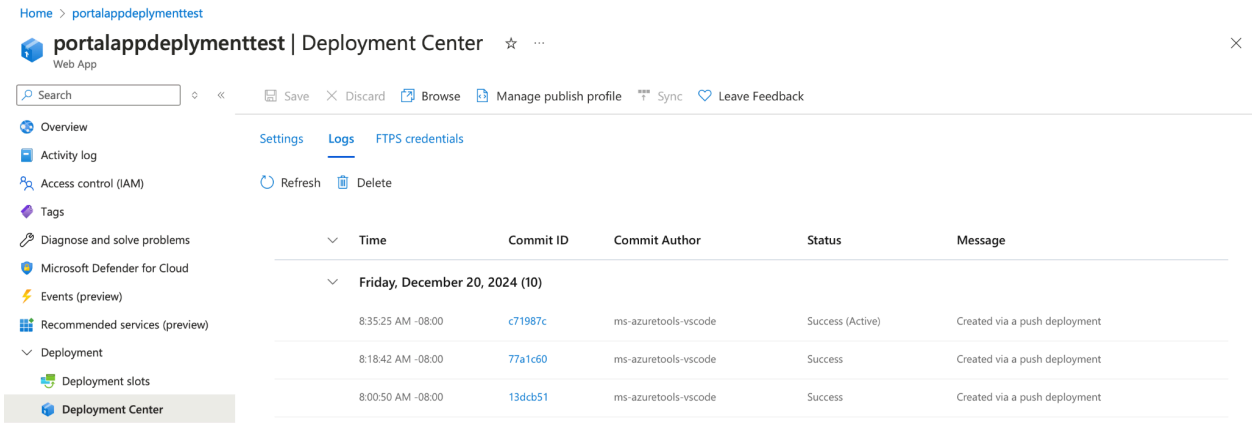
The client ID can be found in the Authentication section under Identity provided. If there is no active identity provider, one must be added and configured.

Deployment

Currently, the application is registered through the Azure portal under the name “portalappdeplymenttest.” Going into the App Service in Azure reveals authentication, service plan, and hosting details. It also shows deployment logs for everytime the codebase is pushed for deployment. There are two recommended ways to deploy a new version of the app.

1. Using GitHub: Select a GitHub repository to connect to on the portal. Every time changes are committed and pushed to the main branch, the app begins the deployment process.
2. Using VSCode: Download the Azure extension on VSCode. After making changes to the code, select the Azure icon in the extension selection menu, and log in with BeiGene credentials. Then, select the App Service you want to deploy it to. Currently, deployments have been pushed to portalappdeplymenttest.

Once you have pushed for deployment, it will take several minutes to complete. The deployment status can be checked on the Azure portal in the app service by going into Deployment Center > Logs. The entry that says "Active" is the one that is currently deployed.



Codebase

There are a few key files in the codebase that are important for understanding the overall architecture, workflow, and areas of future development.

The **pages/index.tsx** file serves as the main landing page, displaying a list of available chatbots and a user-specific navigation sidebar. Upon authentication, users are assigned to one of two artificial privilege groups, which determine their access to certain chatbots. The page features a search bar to filter chatbots by name and reorders them to display accessible chatbots at the top, with inaccessible ones locked and displayed at the bottom. A responsive layout is implemented to ensure proper rendering on both desktop and mobile devices, with a collapsible sidebar for navigation links like Home, Chatbots, and Settings.

The **pages/api/auth/[...nextauth].ts** file handles the authentication flow using Azure Active Directory (Azure AD) with NextAuth. It configures the Azure AD provider with client credentials, tenant details, and necessary Graph API scopes such as https://graph.microsoft.com/.default for user profile and group

information. The file manages tokens using JSON Web Tokens (JWTs), including refreshing expired access tokens through the Microsoft identity platform. The session callback enriches the session object with user-specific data, such as their profile image and group memberships, by making Graph API requests to endpoints like /me/photo/$value and /me/transitiveMemberOf. This ensures the client side can access the authenticated user's details and privileges seamlessly.

The **types/next-auth.d.ts** file extends the default NextAuth typings to include custom properties for sessions and JWTs. It defines additional fields such as profile_image, groups, and access_token_expires_at on the Session.userobject, and properties like accessToken and refreshToken on the JWT object. By customizing these interfaces, the file ensures that TypeScript enforces type safety throughout the application, allowing consistent use of these extended fields.

Finally, the **styles/Home.module.css** file defines the core styling for the home page and its components. It includes responsive layout rules for the main container, sidebar, and content area, as well as grid and hover effects for chatbot cards. The sidebar is styled for smooth transitions when opening or closing, with additional classes for navigation links, user profile sections, and authentication buttons. The file also supports theming with CSS variables, enabling a cohesive appearance for light and dark modes. Together, these styles ensure a polished and user-friendly interface for the entire application.

App Registration

In order to register an app or create a new app service, follow this tutorial: Quickstart: Register an app in the Microsoft identity platform. This should be done for several reasons when pushing the final version of the app:

1. A more polished URL and app name is needed.
2. The Chatbot Portal icon on the BeiGene app portal should direct to this website.
3. Everyone within the organization should have some level of access to the website.

An important thing to note is that most of the GTS team cannot access App Registration controls or create a new one. In order to make any configurations, please consult Raymond Chan. Within the App Registration menu, there's options to add, edit, or delete redirect URLs (used for authentication) and publish an "Enterprise App." Also, make sure the Web App corresponds to the correct App Registration.

Further Development and Troubleshooting

The most crucial next step in the development of this website is the ability to automatically display or hide chatbots based on the logged in user's access. This access can be determined based on the organizational user groups the user is in. Currently, there has been difficulties in

retrieving the user's group membership information through the Graph API. This is the same Microsoft-based API that was used to retrieve the logged-in user's name and email. With this information, the website can be configured so that it displays the chatbots the user has access to at the top while putting a lock icon on the chatbots they don't have access to. Right now, access is randomized for interface demonstration purposes.

The Help and Settings pages need to be updated. The Help page may give details about who to contact if issues or questions about chatbots and access arise. The Settings page can include many more personalization features like a functioning toggle between dark/light mode and accessibility tools.

Some early testers had issues when trying to open the website such as only seeing a blank screen or not being able to log in. If such an issue arises, clear the website data for this specific website on the browser or use private browsing. There may also be issues with HTML requests being too large that may cause the website to crash for some users.