PROJECT IN INTELLIGENT INFORMATION SYSTEMS

W2024

**CHATBOT AI (LLM) ASSISTANT**

Amir Suleimenov K12247291

**Table of content**

[Abstract 2](#__RefHeading___Toc6106_2379827945)

[Introduction 2](#__RefHeading___Toc6108_2379827945)

[Brief description of Project: 3](#__RefHeading___Toc6110_2379827945)

[Goals 4](#__RefHeading___Toc6112_2379827945)

[Requirements 5](#__RefHeading___Toc6114_2379827945)

[Summary System Description 5](#__RefHeading___Toc6116_2379827945)

[Stakeholder Identification 6](#__RefHeading___Toc6118_2379827945)

[The functional and nonfunctional requirements 8](#__RefHeading___Toc6128_2379827945)

[Actors and Agents 10](#__RefHeading___Toc6130_2379827945)

[Prototyping Scenario 11](#__RefHeading___Toc6132_2379827945)

[Concept 15](#__RefHeading___Toc6134_2379827945)

[Base architecture diagram 15](#__RefHeading___Toc9354_2379827945)

[Web model 15](#__RefHeading___Toc9356_2379827945)

[Data model 16](#__RefHeading___Toc9358_2379827945)

[User Scenarios 22](#__RefHeading___Toc9360_2379827945)

[References 26](#__RefHeading___Toc9362_2379827945)

## **ABSTRACT**

**The evolution of chatbots has progressed from rule-based systems to advanced AI-driven models powered by large language models (LLMs). This project presents the development of an intelligent chatbot assistant designed for a volunteer web portal, leveraging the Llama 3 model via an Ollama instance. The chatbot aims to streamline user interactions by assisting with profile creation, volunteer onboarding, and task management.  
The system integrates with a Django-based web platform, storing user data in an SQLite database and dynamically generating structured responses in JSON format. By maintaining conversational context and adapting responses based on user input, the chatbot enhances engagement, accessibility, and efficiency. Functional and non-functional requirements were carefully designed to ensure natural language understanding, error handling, and scalability.  
The chatbot serves different stakeholders, including volunteers, nonprofit managers, and administrators, within a structured framework that aligns with European AI regulations. Through a modular and extensible architecture, this project demonstrates how LLM-powered chatbots can optimize volunteer engagement, automate administrative processes, and enhance user experiences in nonprofit digital platforms.**

## **INTRODUCTION**

**Key aspect to consider is the role of volunteering in social cohesion, particularly in Austria. Volunteering is an essential component of the Austrian social system, fostering community engagement and support across various sectors. To track the evolving nature of voluntary activities, Austria regularly conducts the Survey on Volunteering in private households, ensuring that policymakers and organizations have up-to-date insights into participation trends.  
According to the latest data from the first quarter of 2022, approximately 3.73 million individuals in Austria were engaged in volunteering, making up a significant majority of the population. (Statistics Austria) This growing trend highlights the need for a modern and efficient registration system capable of handling a large-scale database of volunteers. Traditional methods struggle to keep up with the increasing demand, reinforcing the importance of digital transformation in volunteer management.**

**A significant advantage of LLM-powered chatbots over traditional web forms is their ability to improve and simplify the registration process. Conventional forms often require users to manually input extensive data in a rigid format, which can be time-consuming, repetitive, and discouraging. In contrast, a chatbot-driven approach provides a more interactive and user-friendly experience, guiding volunteers through the process dynamically, answering questions in real time, and adapting the conversation based on user input. This personalized assistance reduces drop-off rates, enhances user engagement, and ensures faster and more accurate data collection compared to static forms.**

**Chatbots have several types and levels of complexity. They started with ELIZA, developed by Joseph Weizenbaum. ELIZA simulated conversation using pattern-matching techniques and scripts but lacked real understanding, only creating the appearance of comprehension (Weizenbaum, 1966). Based on this idea, a family of rule-based chatbots developed.**

**Rule-based chatbots are basic and easiest to implement. They are made to interact with users by previously determined guidelines and requirements. These solutions initiate with pre-programmed responses when they recognise certain phrases or patterns in user input. Developers manually create and design a rule-based chatbot's rules, which define how the bot will react to different user inputs. That means chatbot conversations always have strict restrictions on the chat flow algorithm. Users can't interact flexibly to get or receive information in a different order (ChatInsight, 2024). In recent years, the development of neural networks and deep learning led to the creation of large language models (LLMs) such as OpenAI’s GPT series. () These models, trained on vast datasets and fine-tuned for natural language understanding, can engage in coherent, context-aware conversations across a wide range of topics (Brown et al., 2020). Unlike earlier models, LLMs utilize transformer architectures, allowing for superior handling of context and generation of nuanced, human-like responses (Vaswani et al., 2017). JP Morgan workers were encouraged to use LLM Suite for “writing, generating ideas, solving problems using Excel, summarizing documents,” among other things, according to an email sent by the bank. JP Morgan employees were encouraged to use the LLM Suite for “writing, generating ideas, solving problems with Excel, summarizing documents,” among other things, according to an email from the bank. At JPMorgan, the chatbot could augment the work with “a hybrid of human and AI analysts, similar to how the intelligence community works,” Igor Jablokov, founder and CEO of AI startup Pryon told Fortune. JPMorgan has more AI chatbots in addition to LLM Suite, like Connect Coach and SpectrumGPT, which are designed for specific business tasks rather than being general-purpose tools (Confino, 2024)**

**This adaptability makes LLM chatbots particularly well-suited for tasks like volunteer onboarding, where they can guide new users by answering questions, collecting relevant information, and recommending opportunities tailored to each user’s profile and preferences.**

### Brief description of Project:

Chatbots have evolved significantly over the years, transitioning from basic rule-based systems to sophisticated AI-powered conversational agents. Their theoretical application for websites has expanded beyond simple user interaction, enabling a new level of engagement and personalization. In the context of modern web applications, a chatbot serves not only as a means of communication but as a core component of the user experience, offering tailored services, guiding users through tasks, and providing dynamic content based on individual needs and preferences.  
The theoretical purpose of a chatbot for a website revolves around streamlining interaction, offering assistance, and improving accessibility. Websites often require systems that can handle common user inquiries, support complex workflows, and assist users in navigating the platform efficiently. A well-designed chatbot can act as a virtual assistant, guiding users through projects. Also chatbot could add specific volunteer information, troubleshooting steps, or even assist with tasks like filling out forms, providing status updates, or offering personalized recommendations.  
One of the primary purposes of a chatbot on a website is to provide real-time volunteer support. Instead of relying on static FAQs or waiting for a human representative to become available, users can get immediate answers to their questions. The chatbot can interact in a conversational manner, making the experience feel more personal and responsive. It can address a wide range of inquiries, from basic questions about the website or volunteering possibilities to more complex issues requiring the chatbot to access and integrate user-specific data, like purchase history or account status.  
In this project, we document the design and development of a chatbot based on these LLM capabilities. The project contains a website, which uses Django framework and stores authorized user profiles in SQLite database with a rendering chat page. Chatbot is powered by a configured Llama LLM model, which runs through an Ollama instance. Llama 3 was chosen as its significance lies in its balance of performance, cost-efficiency, and open accessibility. Meta's experimental evaluation discovered that the Llama3 model performs on par with leading language models such as GPT-4. (Llama Team, 2024). This project not only highlights the potential of LLMs in chatbots to enhance user interactions but also pushes the boundaries of what is possible with AI-driven communication.

## Goals

**Project Goal:**

Develop an intelligent LLM chatbot assistant prototype  
Measure and collect performance based on human conversations about the interaction process

**System(Chatbot) Goal:** Assist web portal users(NPO managers, volunteers):  
The chatbot should consult the user, provide information and help the user to use the web platform (Sub-goals achieving processes)

**Sub-System Goal 1:** Have a specialized chat for user profile creation. Where the user gets a template of authentication and registration profiles. The chatbot should check that information between fields is logically or semantically connected and that fields do not contradict each other. The chatbot after completing a form, responds with a form in JSON format for further website functions.

**Sub-System Goal 2:** Have a specialized chat for volunteer onboarding, after creating a user profile or after user authentication. Where additional user fields such as competencies or additional information such as gender, age, etc. It will allow to get more personalized experience based on provided user interest and characteristics. Also, it will possibly open more features as in career portals. It could allow users to find specific tasks and possibly subscribe to them.

**Sub-System Goal 3:** Have a specialized chat for task creation. Where the user could get a template of a task based on his data. The chatbot should check that information between fields is logically or semantically connected and that fields do not contradict each other.

## REQUIREMENTS

## Summary System Description

(Prototype for) AI-assisted chatbot for Austrian volunteers web portal.

The AI-assisted chatbot for the Austrian Volunteers Web Portal is an intelligent system designed to facilitate seamless interactions between volunteers and nonprofit organizations. The system based on modelfile configuration shall help to users by generating answers for users input. As An Ollama Modelfile is a configuration file that defines and manages models on the Ollama platform. Create new models or modify and adjust existing models through model files to cope with some special application scenarios (GPU-mart). The system have authorization before accessing chatbot for sub-system goal 2 and 3. Built on a robust framework, the chatbot prioritizes security, efficiency, and adaptability. It ensures compliance with AI governance standards while providing scalable performance for multiple concurrent users. The system’s modular design allows for continuous improvements and future enhancements.

Beyond its core functionality, the chatbot operates within a broader ecosystem, interacting with various stakeholders, meeting functional and nonfunctional requirements, and aligning with specific user roles. Through a structured approach, real-world scenarios and use cases demonstrate how the chatbot effectively supports and enhances volunteer engagement.

For a comprehensive understanding of these integrations, the respective sections delve deeper into chatbot interaction characteristics and enhances the chatbot's role within the volunteer web portal.

## Stakeholder Identification

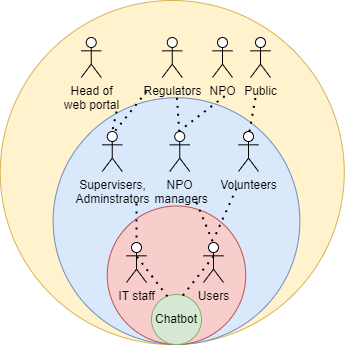


Diagram 1. Onion diagram

System:

The system represents the product or solution that is being delivered by the project (Olson 2013)

* Chatbot

Business System:

This layer represents the Business System and entails not just the final product, but those stakeholders who interact directly with it, like operators. (ConceptDraw)

* IT staff( web master, server technician etc)
* Users

Web application:

Those stakeholders who are functional “beneficiares” according Ian Alexander. These are the other stakeholders within the organization who may not interact directly with the solution who benefit from it. (Alexander, 2003)

* Volunteers
* NPO managers
* Supervisers, Administrators

Public Environment:

Those stakeholders who are wider environment in which the organization operates. This layer is populated with stakeholders who are outside the firm but who are still important. As public or various pseudo-governmental organizations has a crucial stake in most projects. (Alexander 2003):

* Head of web-portal environment
* Non Profit Organization
* Public / Citizens
* Regulators

Regulators of such web site with AI chatbot can contain:

* AI Act of European AI Office (AI Act, 2024)
* The Federal Ministry of Social Affairs, Health, Care and Consumer Protection
* Municipal/ Local offices and representatives of European AI Office and Federal Ministry

Diagram 1 represents diagram provides a visual metaphor where the central core represents the system or product, and the surrounding layers represent different groups of stakeholders who are involved with or impacted by the project. The onion metaphor helps to distinguish between stakeholders based on their level of involvement or influence with the system, showing how close or distant each group is to the core system.

At the core of the onion diagram is the system itself. In this case, the core represents the chatbot—the primary product or solution being delivered by the project. The chatbot is the focus of the entire project and the central entity that all stakeholders are interacting with or working around. Its purpose is to respond to user inputs, understand intentions, and generate responses in natural language, ultimately providing valuable assistance or information to users. This layer is the foundation, as it is the product that all other stakeholders ultimately interact with, either directly or indirectly.  
  
Surrounding the core is the Business System layer. This layer includes the stakeholders who directly interact with the system and are responsible for its day-to-day operation. These are the individuals and teams that ensure the chatbot is functioning correctly and integrated within the larger business or organizational structure. For example, IT staff, such as webmasters and server technicians, are critical in ensuring the system runs smoothly. They handle the technical aspects, such as hosting, maintenance, updates, and troubleshooting. Additionally, the users—those interacting directly with the chatbot—are placed in this layer. These individuals rely on the chatbot to provide responses and perform tasks, making them key players in the success of the system. The chatbot’s ability to respond accurately and efficiently to user inputs directly impacts the experience of these stakeholders.  
  
Next, we have the Web Application layer. This layer encompasses the functional beneficiaries—stakeholders who may not interact directly with the system but benefit from it in some way. These individuals or groups leverage the chatbot’s capabilities to enhance their own work or processes, even if they aren't directly involved with the technical side of the system. For example, volunteers might use the chatbot to receive guidance or answers to frequently asked questions, improving their efficiency and effectiveness. Similarly, NPO managers benefit from the chatbot’s ability to automate certain tasks, such as answering inquiries or providing information, thus reducing the workload of human staff and allowing them to focus on more critical tasks. Supervisors and administrators also fall into this layer, as they oversee the performance and integration of the chatbot within the organization, ensuring it aligns with organizational goals and needs.  
  
Finally, the outermost layer is the Public Environment. This layer consists of stakeholders who exist outside the immediate organization but still hold a significant interest in the system. These are external stakeholders whose influence may not be as direct but who are still impacted by the chatbot’s presence or its interaction with the public. For instance, the head of the web-portal environment represents an external oversight body responsible for the broader technical environment in which the chatbot operates. Non-profit organizations that collaborate with or benefit from the chatbot may also be part of this layer, as they rely on its functionality to serve their communities or carry out their missions. The public or citizens may be affected by the chatbot if it provides public-facing services or resources. Finally, regulators are external stakeholders who ensure that the system complies with legal or industry regulations, such as data protection laws, accessibility standards, or ethical guidelines. By understanding these layers, the project optimize and ensure that all relevant stakeholders, from direct users to external regulatory bodies, are considered and engaged appropriately throughout the development and implementation of the system.

## The functional and nonfunctional requirements

EARS template helps to express system behavior from the user's perspective and simplifies the communication of what a system should do. This method breaks down requirements into components that make the system’s intended actions clear and measurable. Template keeps functional and non-functional requirements clear and structured for all stakeholders involved in the project. Below is a list of requirements for webportal chatbot project.

**List of functional requirements**

Authentification

**FR100:** The system shall allow user create account (registrate) for the purpose of providing conversation for every user

**FR101:** The system shall allow user to his/her login (authentificate) by entering his/her password

**FR102:** The system shall allow user change password based on his/her account

**FR103:** The system shall use SQLite db as DBMS for easier integration with other parts of web portal

Chatbot

**FR200:** The LLM-powered chatbot shall respond to user inputs in natural language.

**FR201:** The chatbot should generate an answer for every message from user input.

**FR202:** The system shall be able to understand the user's intention from their input.

**FR203:** The bot shall extract specific information (entities) from the user’s message, such as dates, names, or task information.

**FR204:** The bot shall have context management and conversation flow

**SFR2040:** The system shall maintain the context of a conversation across multiple turns, remembering prior user inputs and responses.

**SFR2041:** The system should handle ongoing conversations, where it can maintain state and carry information across multiple exchanges.

**SFR2042:** The chatbot should continuously update its internal state to reflect new information from the conversation and respond in a relevant manner based on the current state.

**SFR2043:** The system shall decide the next action (asking for more information, providing an answer, executing a task) based on the current state.

**SFR2044**: The chatbot should use welcome message, in which it start with a greeting, a brief explanation of its functions and goal of chatconversation.

**FR205:** The system shall h**andle misunderstandings and have adaptability**

****SFR2050:** The system shall be able to handle misunderstandings, detect when it cannot understand, and ask clarifying questions.**

****SFR2051:** The chatbot should recognize when it cannot fully understand a user's input and ask clarifying questions to resolve ambiguities.**

**SFR2052: The system shall adapt to individual users by learning preferences over time.**

**List of nonfunctional requirements**

Chatbot

**NFR 200:** The LLM-powered chatbot’s architecture shall allow for updates and bug fixes to be deployed without downtime.

**NFR 201:** The system should handle a large number of concurrent users and scale efficiently without performance degradation.

## Actors and Agents

The Actors and Agents section presents a structured table that outlines the key roles involved in the chatbot system. It categorizes the primary participants in the system, specifying their responsibilities and interactions within the AI-powered chatbot in the web portal.  
It defines primary roles in the level of business system of Diagram 1. Table modulates description of actors during normal daily interaction of chatbot. The chatbot serves as the central AI component, facilitating human interactions while IT staff maintain the backend functionality. By clearly delineating roles, this classification helps in understanding the system’s functional workflow and technical dependencies.

Table 1. Actors in business system level.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Primary Actor** | **Type** | **Description** |
| 1 | User of chatbot | Human | Responsible for   1. registration process via entering personal details in special chat or webpage 2. autentification process before starting chat for sub-system goals 2 and 3 3. entering messages and questions for defining exact topic of user interest |
| 2 | Chatbot Processing Model | AI | Provides core logic for answering on messages in natural language processing   1. Configured as chatbot assistant for Austrian volunteers portal 2. Identifying individual chat and conversation history 3. Fulfill specific goal of sub-system chats |
| 3 | IT staff | Human | Responsible for   1. Maintenance technical equipment such as server equipment 2. Problems and bug solving 3. Help users troubles and questions with interaction to chatbot instance |

## Prototyping Scenario

The Prototyping Scenario section presents structured use case tables that outline interactions between users and the chatbot, which is designed with predefined roles and domain-specific knowledge. These tables define the chatbot’s behavior, user expectations, and the system’s responses to various inputs. All Use cases define goal of user – web site interactions in specific cases. The Use cases also handle exception handling scenario logic. Use cases UC 101 and UC 102 ensure that the chatbot functions as an interactive assistant with structured responses based on its role.

|  |  |
| --- | --- |
| **ID** | UC100-Auth-Login-page |
| **Description** | User logs into app |
| **Actors** | Volunteer/NPO manager as User of chatbot |
| **Stakeholders** | NPO managers, Volunteers |
| **Pre-Conditions** | The system has access to the internet via the device’s WLAN access point |
| **Success end condition:** | The user is logged into the app and into the chatbot in the background |
| **Failure end condition:** | The user is not logged in and is being asked to contact support to verify credentials & permissions |

|  |  |  |
| --- | --- | --- |
| **Main Success Scenario** | | |
| 1 | The user enters his/her user info (user id, password) | |
| 2 | The user submits the user info | |
| 3 | The systems checks whether the user is permissioned to use the chatbot app | |
| 4 | The system creates a session for the user for submitting API requests LLM engine | |
| 5 | The app launches and displays the main screen/menu | |
|  |  | |
| **Alternative Scenarios** | | |
|  |  | |
| **Exception Scenario** | |  |
| A1.1 | Checking the user credentials results in an authentication error (unknown/invalid user) | |
| A1.2 | The system asks the user the verify credentials and to try to login again | |
| A2.1 | Checking the user credentials results in an authorisation error  (user is known but not permissioned to use the app) | |
| A2.2 | The system informs that permission to use the app is missing and to contact support  Before trying to log in again | |
| A3.1 | The system is unable to create a session with the chatbot system API (either authentication, authorisation or general technical error) | |
| A3.2 | The system asks the user to verify network connection and permissions  Inside the web application system before trying to log in again | |

|  |  |
| --- | --- |
| **ID** | UC101-chat-for-user-creation- |
| **Description** | Visitor uses chatbot to create profile |
| **Actors** | User of chatbot |
| **Stakeholders** | Public, NPO |
| **Pre-Conditions** | The system has access to the intranet via the device’s WLAN access point and user has not logged |
| **Success end condition:** | The user created from chat conversation |
| **Failure end condition:** | The website did not created user from chat for user creation |

|  |  |  |
| --- | --- | --- |
| **Main Success Scenario** | | |
| 1 | The chatbot checks whether the user is not logged | |
| 2 | The user starts his conversation with chatbot | |
| 3 | The user sends user details as name, username, last name, email and password | |
| 4 | The chatbot checks the validity of fields | |
| 4 | The system generates JSON file from user entries and volunteer role as default | |
| 5 | The app saves correct json as entity in database | |
| 6 | The app displays authorization page for log in | |
| **Alternative Scenarios** | | |
|  |  | |
| **Exception Scenario** | |  |
| 3.A1.1 | Checking the user not logged | |
| 3.A1.2 | The system returns to main page and show alert message that user alredy logged in | |
| 3.A2.1 | Checking the user entries for relevance in task fields | |
| 3.A2.2 | The system informs that entries does not correspond with fields as password and password confirmation. Chatbot ask user to resend data in appropriate format | |
| 4.A3.1 | The system is unable to create a task entity with the chatbot database API as username or email is exists | |
| 4.A3.2 | The system show system error in chat with appropriate field as it already exists Inside the chatpage system ask to try send other data substitute new data and initializes another user creation function | |

|  |  |
| --- | --- |
| **ID** | UC102-chat-for-volunteer-profile |
| **Description** | User logs into app with volunteer profile |
| **Actors** | Volunteer as User of chatbot |
| **Stakeholders** | Voluunteer, Public |
| **Pre-Conditions** | The system has access to the intranet via the device’s WLAN access point and user has logged as volunteer |
| **Success end condition:** | The user completed volunteer profile from chat conversation |
| **Failure end condition:** | The website did not completed volunteer profile with data from chat |

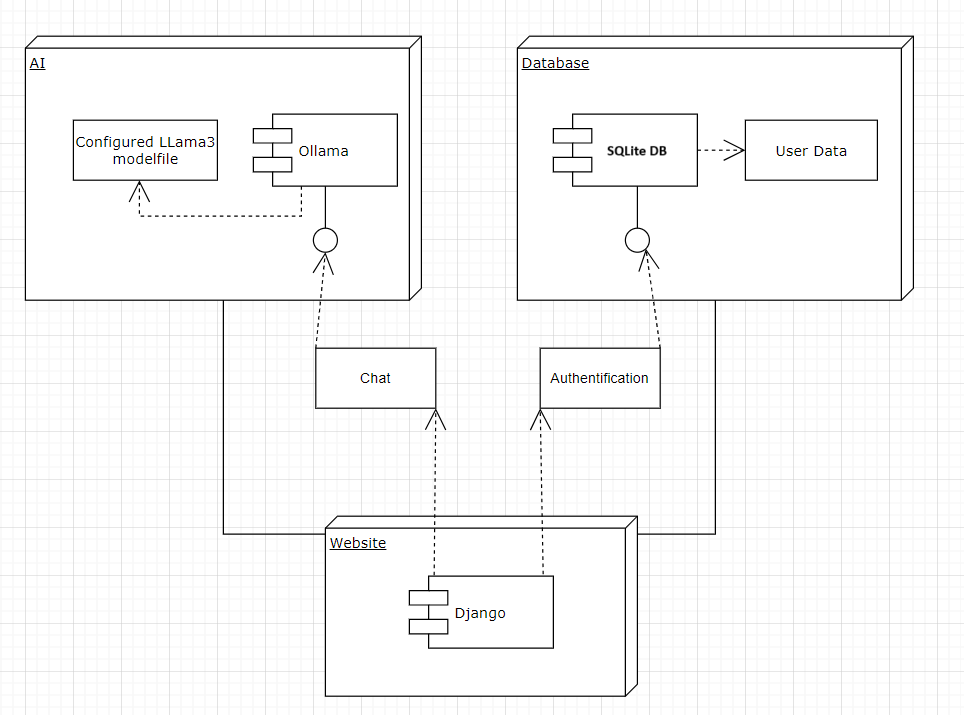
|  |  |  |
| --- | --- | --- |
| **Main Success Scenario** | | |
| 1 | The chatbot checks whether the user logged as volunteer | |
| 2 | The user starts his conversation with chatbot | |
| 3 | The user sends voluunteer details as description, schedule, gender, competences etc. | |
| 4 | The chatbot checks the validity of fields | |
| 4 | The system generates JSON file from user entries. | |
| 5 | The app saves correct json as entity in database | |
| 6 | The app displays user page | |
| **Alternative Scenarios** | | |
|  |  | |
| **Exception Scenario** | |  |
| 3.A1.1 | Checking the user not logged as volunteer | |
| 3.A1.2 | The system returns to main page and show alert message that access is denied as user profile is not Volunteer | |
| 3.A2.1 | Checking the user entries for relevance in task fields | |
| 3.A2.2 | The system informs that entries does not correspond with fields as competences are out of list of competences. Chatbot ask user to resend data in appropriate format | |
| 4.A3.1 | The system is unable to create a task entity with the chatbot database API as data formats is wrong | |
| 4.A3.2 | The system show system error in chat with appropriate field which syntaxis is inappropriate Inside the chatpage system ask to try send json initialization request again to chatbot | |

|  |  |
| --- | --- |
| **ID** | UC103-chat-for-task-creation- |
| **Description** | User logs into app with NPO manager profile |
| **Actors** | NPO manager as User of chatbot |
| **Stakeholders** | NPO managers, NPO |
| **Pre-Conditions** | The system has access to the intranet via the device’s WLAN access point and user has NPO manager profile |
| **Success end condition:** | The NPO managers task created from chat conversation |
| **Failure end condition:** | The website did not created chat for NPO manager |

|  |  |  |
| --- | --- | --- |
| **Main Success Scenario** | | |
| 1 | The chatbot checks whether the user is permissioned to create tasks | |
| 2 | The user starts his conversation with chatbot | |
| 3 | The user sends task details as name, description, startdate, enddate | |
| 4 | The chatbot checks the validity of fields | |
| 4 | The system generates JSON file from user entries. | |
| 5 | The app saves correct json as entity in database | |
| 6 | The app displays page with users tasks | |
| **Alternative Scenarios** | | |
|  |  | |
| **Exception Scenario** | |  |
| 3.A1.1 | Checking the user permission(not NPO manager profile) | |
| 3.A1.2 | The system returns to main page and show alert message that access is denied as profile is not NPO manager | |
| 3.A2.1 | Checking the user entries for relevance in task fields | |
| 3.A2.2 | The system informs that entries does not correspond with fields as dates exepts system restrictions. Chatbot ask user to resend data in appropriate format | |
| 4.A3.1 | The system is unable to create a task entity with the chatbot database API | |
| 4.A3.2 | The system asks the user to verify network connection and permissions  Inside the chatpage system ask to try send json initialization request again to chatbot | |

## CONCEPT

## Base architecture diagram

Diagram 2. Architecture diagram

The second diagram illustrates the base structure of an AI-assisted chatbot system integrated into a Django-based website, utilizing an Ollama-powered LLaMA 3 model for AI interactions and an SQLite database for user data storage.

Django Framework acts as the primary interface for users, handling chat-based interactions and authentication processes. Users interact with the chatbot through the website, which processes requests and routes them to the AI and database systems.

AI System hosts the configured LLaMA 3 model, which serves as the chatbot's core intelligence. Ollama functions as the AI model interface, processing user queries and generating responses. The AI component communicates bidirectionally with both the website and the database, ensuring contextual responses based on stored user data.

Database (SQLite) stores user profile information and other relevant data. Works alongside the authentication module to manage user access and permissions. The AI system retrieves user data when necessary, allowing for personalized chatbot interactions.

The user starts to interact with the website via chat. The Django backend processes the request and forwards it to the AI system (Ollama & LLaMA 3). The AI system may require user data from the SQLite database, such as authentication status or stored preferences. The chatbot responds through the Django web interface, completing the interaction cycle.

This architecture ensures modularity, scalability, and efficiency, enabling an intelligent, context-aware chatbot for volunteer and nonprofit organization engagement.

## Web model

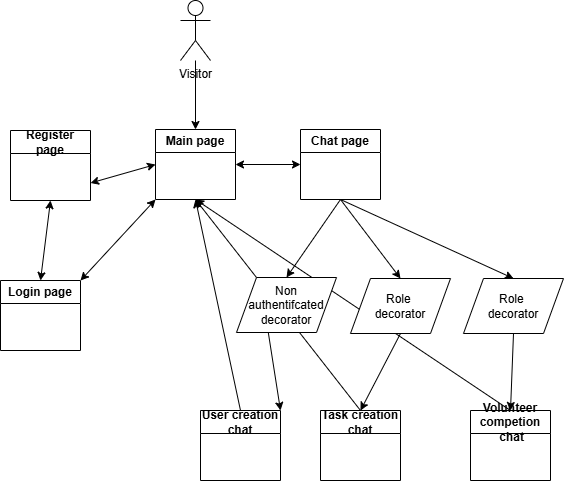


Diagram 3. Representation of Web portal pages and role-based access

This diagram represents the web-based navigation and role-based access model for a chatbot system within a volunteer management platform. It illustrates the interaction flow between users, authentication mechanisms, and chatbot functionalities.

On the left side of web model is located register page and login page. It is old styled part with authentication with representing and sending post forms. Users can access the system through either the Register Page (for new users) or the Login Page (for existing users). Once authenticated, they are redirected to the Main Page, serving as the central hub for further actions.

To interact with chatbot users visit pages on right side of the model. Users navigate from the Main Page to the Chat Page, where they interact with the chatbot. Different chat functionalities are available, such as User Creation Chat, Task Creation Chat, and Volunteer Completion Chat, indicating that the chatbot supports administrative and operational tasks according formulated sub-system project goals. Role Decorators manage user permissions based on their assigned roles, ensuring that only authorized users can perform specific actions. Like, NPO managers create tasks for volunteers, amd volunteer complement their profiles for future matching the tasks. This web model showcases a structured navigation and access control framework for a volunteer management chatbot system. The web model shows that web portal facilitates key user actions like registration, task management, and volunteer coordination, making the system an efficient digital assistant for volunteer organizations.

## Data model

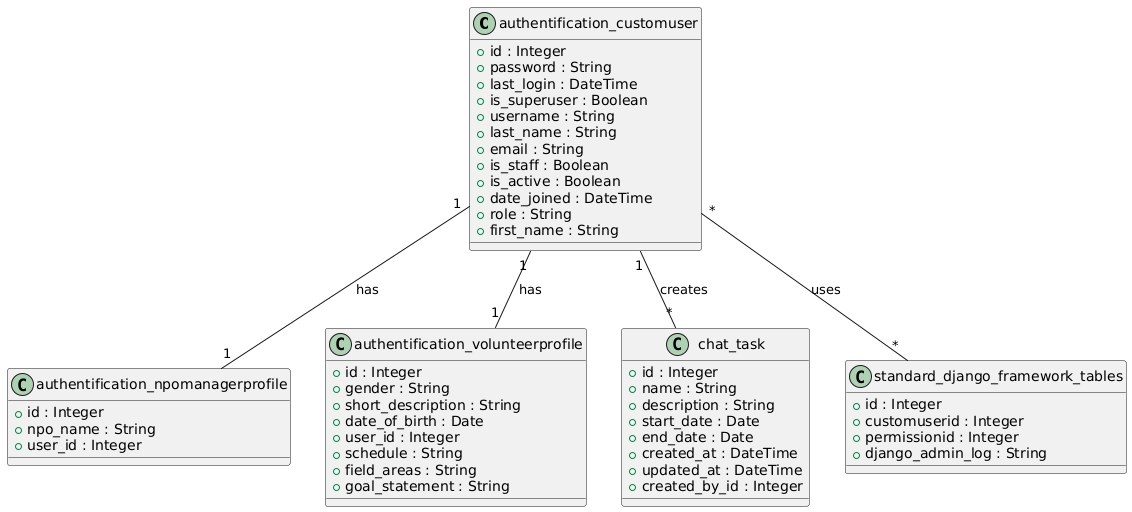
In this section I have the explanation of base models which are used in web application.

I have the User model for all users. It means that volunteers, NPO managers and admin have all general fields. All role specific access, rights and fields are stored in specific profiles.

* Centralized User Management: By using one base User model, the core authentication (e.g., login, password, permissions) remains centralized and consistent across the application.
* Specialized Profiles for Custom Data: The user model could be extended to hold domain-specific fields (e.g., user roles, additional contact details) without cluttering the base User model itself. This keeps the responsibility of the base User model limited to authentication and core properties.
* Dynamic Role Management: Specialized profile models for different types of users (e.g., Volunteer, NPO Manager). These models can hold different sets of data based on their specific needs or business logic, such as varying fields or settings.
* Efficient Rights and Access Control: Permissions and access rights can be customized at the profile level. A common practice is to define different permissions or access rules for each user role, which can be easily handled by specialized profile models.

This approach have several benefits:

* **Easier to Scale User Models:** As application grows to introduce more features (e.g., new user roles or additional fields specific to user types), Users can be extended the profile model without modifying the base User model. This makes it easier to scale app and avoid repeated changes to the user-related codebase.
* **Better Data Organization:** Having separate profiles for different roles makes the database schema more modular, and allows to avoid overloading the base user table with unnecessary fields.
* Data Integrity and Performance: By splitting data into logical profiles for different roles, it improve the integrity and performance of database queries. When accessing a specific profile, app don’t have to retrieve unnecessary information for other types of users.

Diagram 4. UML diagram generated with Plantext UML Editor

Based on UML representation there is several important entities encoded as python classes. First of them is CustomUser class.

User (CustomUser in code) contains:

* Role (volunteer or NPO manger)
* Username (unique for each user)
* Email (unique for each user)
* First name
* Last name (Surname)
* Password

Also database store some internal fields as Id, last\_login, is\_superuser, is\_staff, is\_active, date\_joined

Volunteer profile which extends user contains:

* User id as foreign key to user
* Gender (Male, Female, Other)
* Short description
* Date of birth
* Competencies, which contains predefined list of competencies and values from 0 to 3 (field\_areas)
* Interests
* Schedule (preferred available time)
* Goal statement

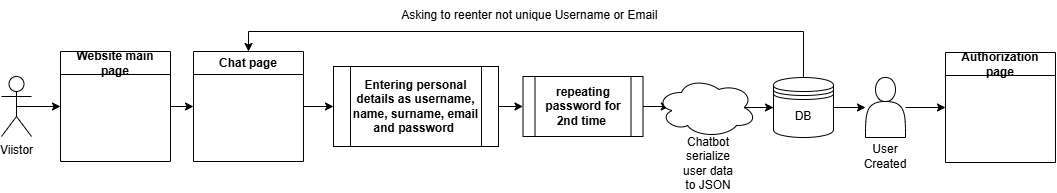
NPO manager contains:

* User id as foreign key to user
* NPO name

Then Task model contains:

* Name
* Description
* Start date
* End date
* Created by which is foreign key to user
* Date of creation (created\_at)
* Date of update (updated\_at)

The table standard\_django\_framework\_tables is schematic representation of User class relations with standard Django framework generated tables

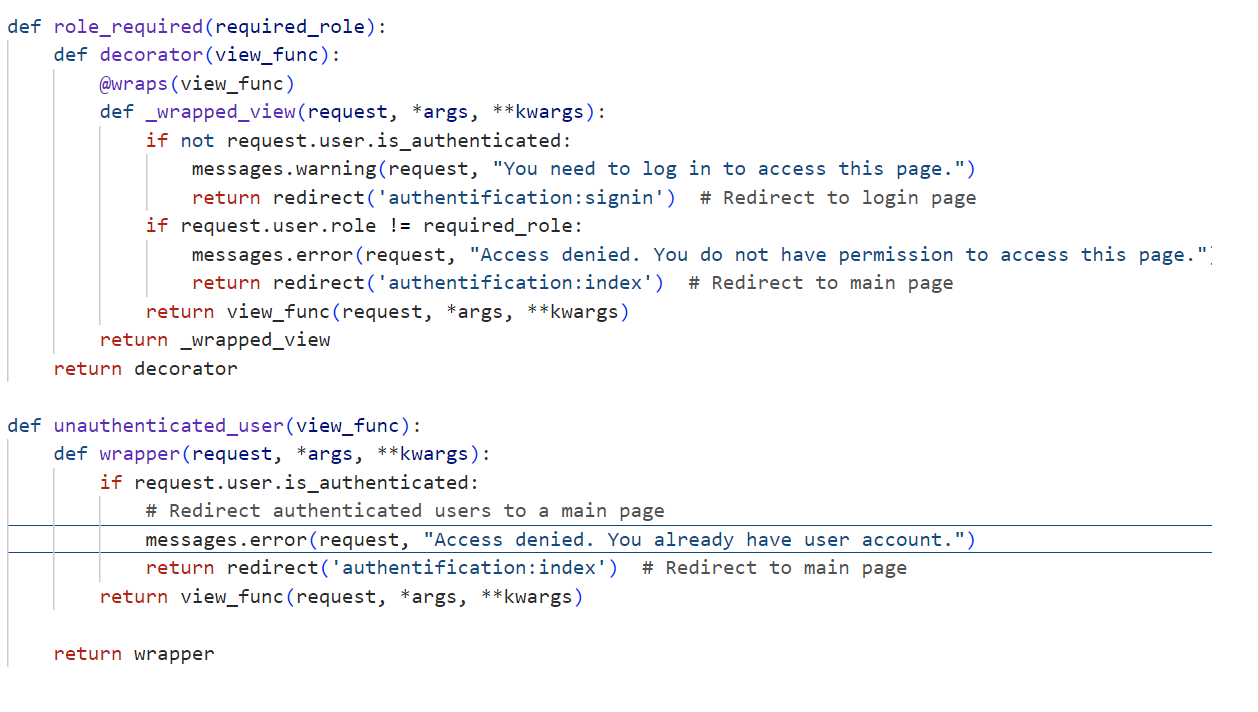
Diagram 5. Pipeline of user creation chat

In diagram 5 represented the flow how user can register in volunterring web portal via smart chatbot conversation. This diagram represents the user registration authentication process for a web-based system, guiding users through the steps required to create an account and gain access to the platform. The user starts from the main page of the website. The user could proceed to a chat-related page to start registration process. The user provides essential registration details, including username, first name, surname, email address and password. After to ensure correctness of password user need to send it again. After bot aggregates all user data and serialize it to JSON format. It sends request to database to ensure that user is new by checking username and email. If username or email is registered chatbot will ask user to add other username or email and will try again. The validated data is stored in a database for future authentication and user management. A new user account is successfully created and stored. Upon successful registration, the user is redirected to the authorization page, where they can log in and access their account.

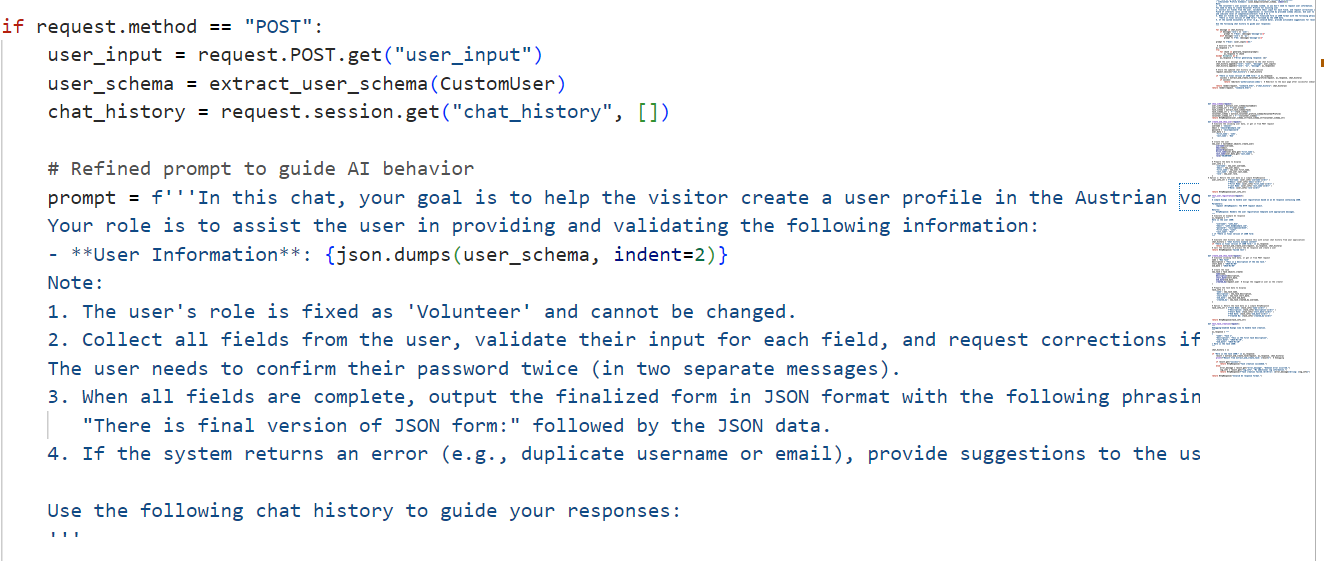
To provide understanding how this data models are work in application, I provide images of major functions implementations for achieving goals of the project.

To avoid already registered user of web portal use user registration chat, web portal tries to check user status. Before access the chat page for user creation Django check with @unauthenticated\_user decorator that user are not logged in. Example how special function will define user unauthenticated user status is shown in image 1.

Similarly, access to volunteer profile and task creation requires users to be authenticated and have the appropriate role. This is enforced through the @role\_required decorator, which checks if a user is logged in and whether their assigned role matches the required permissions. Volunteers must have the "volunteer" role to create their profiles, while NPO managers need the "npo\_manager" role to create and manage tasks. If a user lacks the necessary role, they are redirected to the main page with an error message indicating insufficient permissions.  
  
These custom decorators enhance security and user experience by ensuring that only authorized users can access specific functionalities, reducing errors and streamlining role-based navigation within the system. The implementation of these decorators is depicted in Image 1, demonstrating how Django enforces authentication and authorization in different sections of the platform.

Image 1. Implementation of decorators

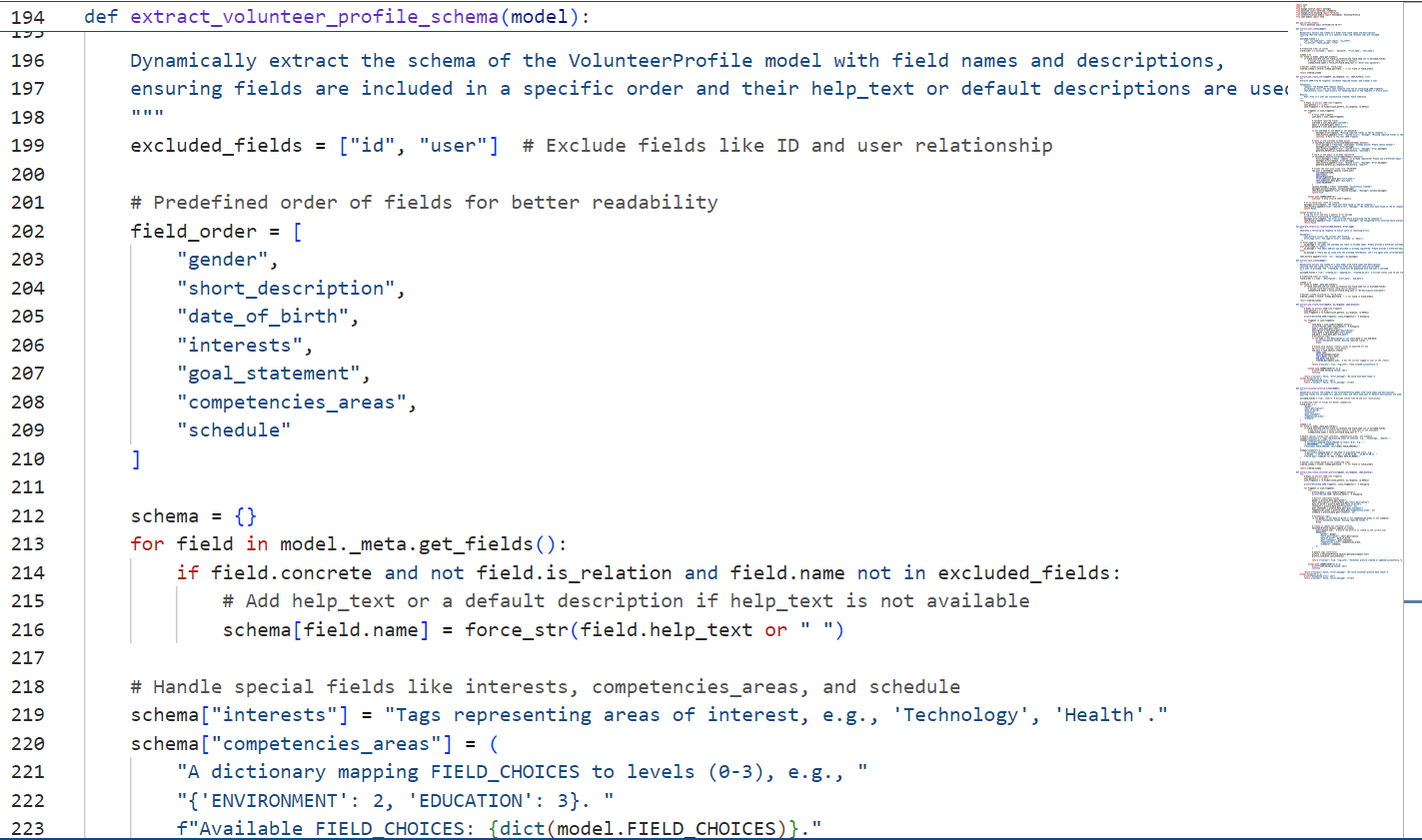
The chat conversations of specific page follows main goal of the chat which is aimed in initial prompt for the ollama bot. In case of user creation page it has such prompt.

Image 2. Prompt initialization

Each prompt contains main goal to the Ollama bot which it need to complete during conversation and main features for what bot need to pay attention as fixed role of the user. Each chat interaction is structured around a clearly defined objective that the chatbot must accomplish during the conversation.  
  
For instance, in the user creation page, the chatbot prompt defines the core purpose of the interaction: guiding the user through the volunteer registration process while ensuring all required data is collected and validated. This prompt establishes a fixed role for the bot, specifying the key aspects it should focus on, such as verifying user input, assisting in the selection of competences (fields of interest), and ensuring adherence to predefined schema constraints.  
  
Additionally, the prompt includes specific conditions that serve as triggers for activating event listeners. These listeners play a crucial role in monitoring user input, detecting when all required information has been gathered, and initiating the JSON-based profile generation. The finalized JSON output is formatted for database storage, enabling seamless integration with the system’s backend. If any validation issues arise, the chatbot proactively provides corrective suggestions, ensuring data integrity and user-friendly error resolution.  
  
This structured approach ensures that each chat session is goal-oriented, context-aware, and optimized for data collection, improving the efficiency and accuracy of the volunteer onboarding process.

For every prompt in the application, the schema of the class is generated dynamically using a specialized function. This dynamic extraction method allows the chatbot to adapt to changes in the class model without requiring manual adjustments to its logic or predefined rules. Unlike rule-based chatbots, which rely on rigidly coded responses and fixed logic, this approach enhances flexibility by decoupling chatbot functionality from the underlying data model. For example there is extracting volunteer profile dynamic schema in Image 3 and 4.

In the case of the volunteer profile, the extract\_volunteer\_profile\_schema function is responsible for retrieving field names, descriptions, and constraints directly from the VolunteerProfile model. This ensures that the chatbot accurately collects, validates, and structures user input according to the most up-to-date data model. The schema is prioritizing fields into logical order, making the onboarding process more intuitive for users.  
  
Additionally, custom handling of complex fields—such as competency levels and schedules—ensures that structured inputs like skill ratings (0-3) and time availability are clearly defined and properly formatted. This method allows to the chatbot’s ability to validate and guide user input dynamically, ensuring compliance with expected data formats.  
  
By automating schema generation, this approach allows the chatbot to remain highly adaptable, supporting future expansions of the volunteer onboarding process while maintaining a seamless integration between data models and chatbot interactions.

Image 3. Extracting schema

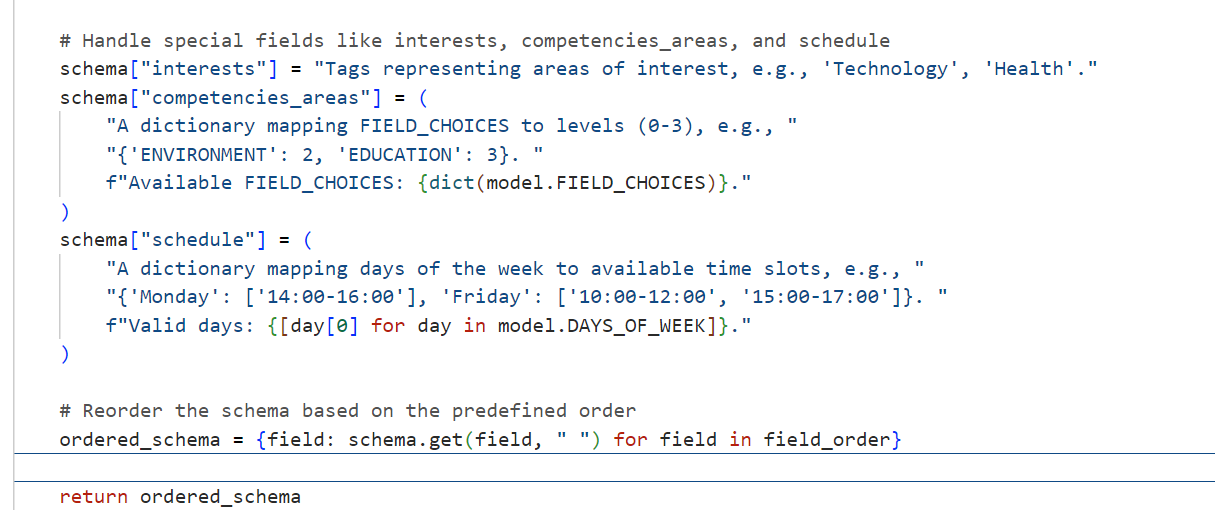


Image 4. Continue function extracting schema

To achieve volunteer profile registration goal, I use complex class attributes, such as a schedule, which are represented as dictionaries in code and stored as strings in the database. I also add help texts to fields to make it easier for the bot to understand their purpose. This way, the bot knows what is expected from the user, and it adjusts the information in each user's message to fill out the required fields. The prompt design follows a structure similar to that of Guangzhi Sun's team. They created a prompt with a general task definition (in my case, a goal definition), an in-context example (which shows how the LLM should behave in specific situations, like my example of a skeleton to fill with values and explanations), LKI (the possible values for some slots, such as restricted end dates or default volunteer roles for user creation), and a query (which evaluates the user input within the app). (Guangzhi Sun et al., 2023)

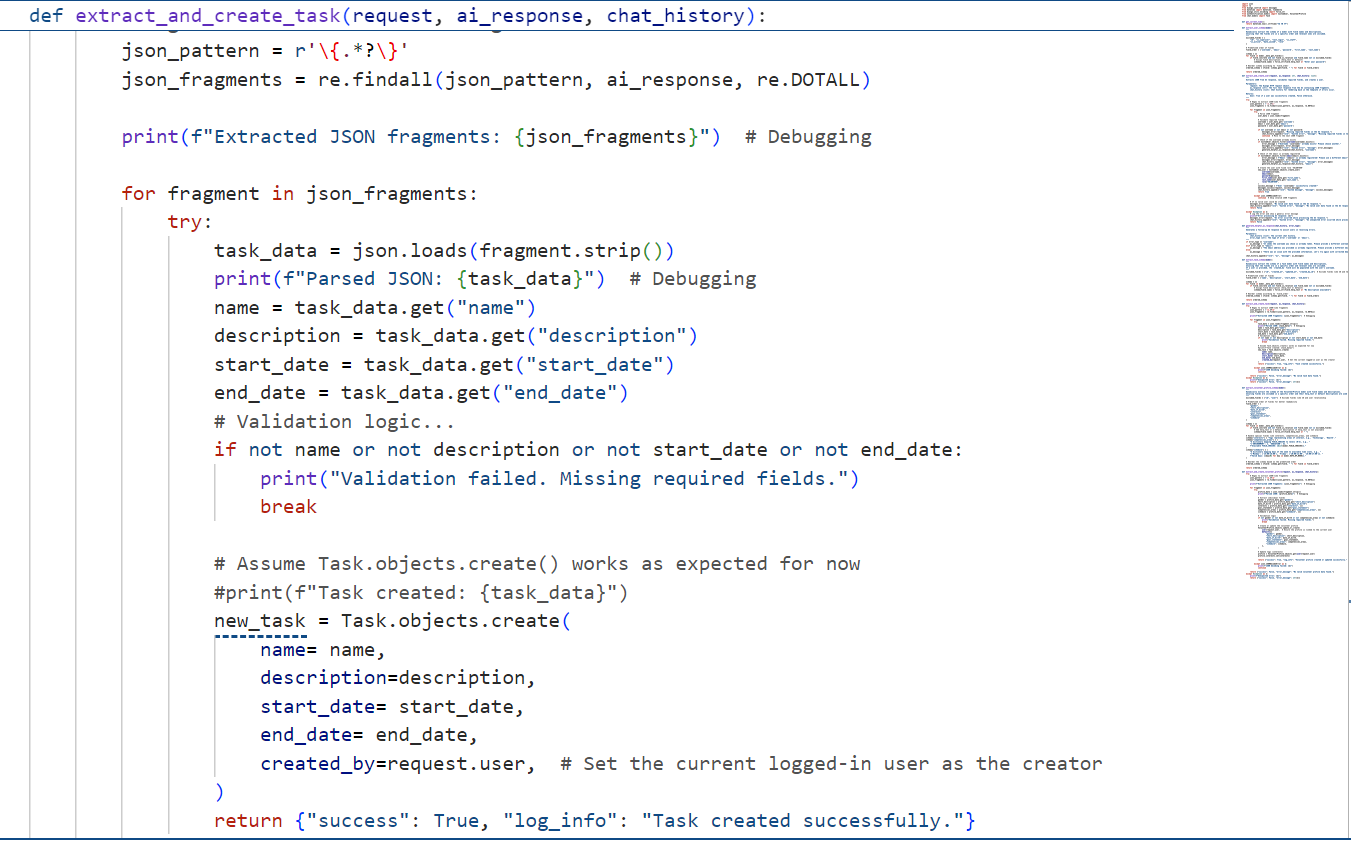
In image 5, chatbot awaits that user fills all required fields filled the calls trigger to try save entity.

Image 5. Trigger to create task

In image 6 the extract\_and\_create\_task function is designed to dynamically extract and validate task-related data from the AI-generated response before creating a new task entry in the system. It plays a crucial role in ensuring that only properly structured and complete task data is accepted, thereby preventing the creation of incorrect or incomplete records.

The function first scans the AI response for JSON-like structures using regular expressions, allowing it to identify, extract, and parse task definitions embedded in the chatbot response. Each extracted JSON fragment undergoes a decoding process to convert it into a structured Python entity. Before proceeding with task creation, the function enforces strict validation rules to ensure that all required fields—such as task name, description, start date, and end date—are properly filled. If any of these attributes are missing or invalid, the function logs an error message and stops execution, preventing incorrect data from being saved. If validation is successful, a new task is created in the system using the Task.objects.create() method. The logged-in user is automatically assigned as the creator of the task, ensuring proper ownership and accountability. The function then logs a success message and redirects the user to their task management page, confirming that the task has been registered correctly.

By automating task extraction, validation, and creation, this function helps automatize the volunteer task management workflow. Users can describe their task in natural language within the chatbot, and the system intelligently extracts and registers the task, reducing manual data entry and improving overall efficiency. This approach enhances usability by allowing volunteers and organizers to manage their responsibilities seamlessly through AI-driven interactions.

Image 6. Extract and creation of task

## User Scenarios

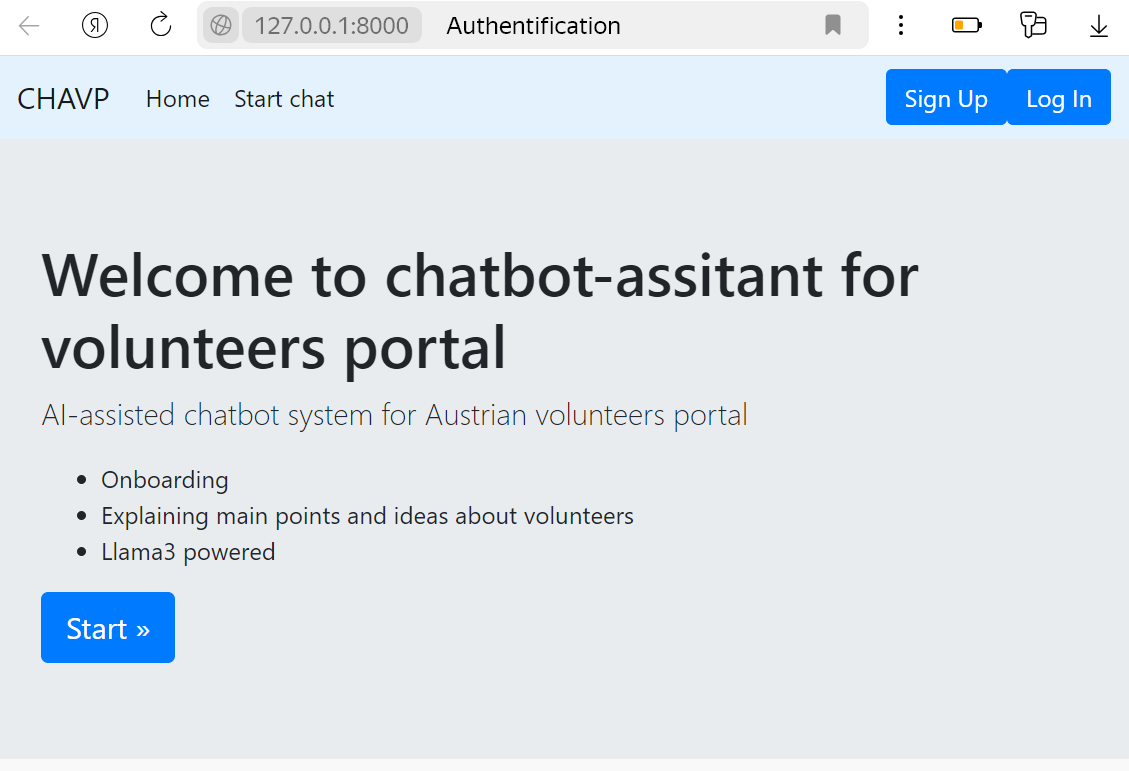
Image 7. Main page

Image 7 is application main page where user can start chat by buttons or log in or register via traditional web pages. It is general entry point where user start interaction with web portal.

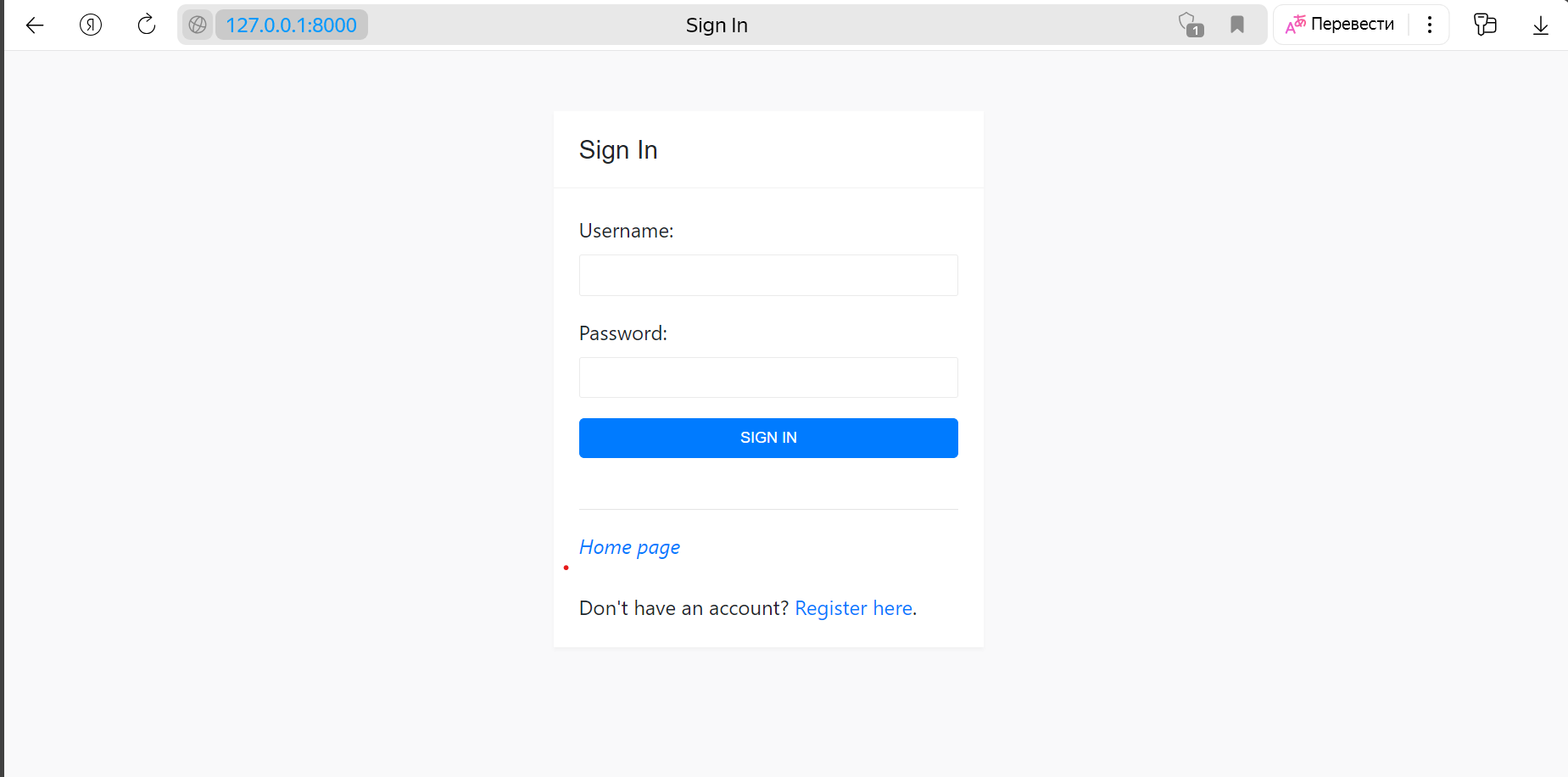
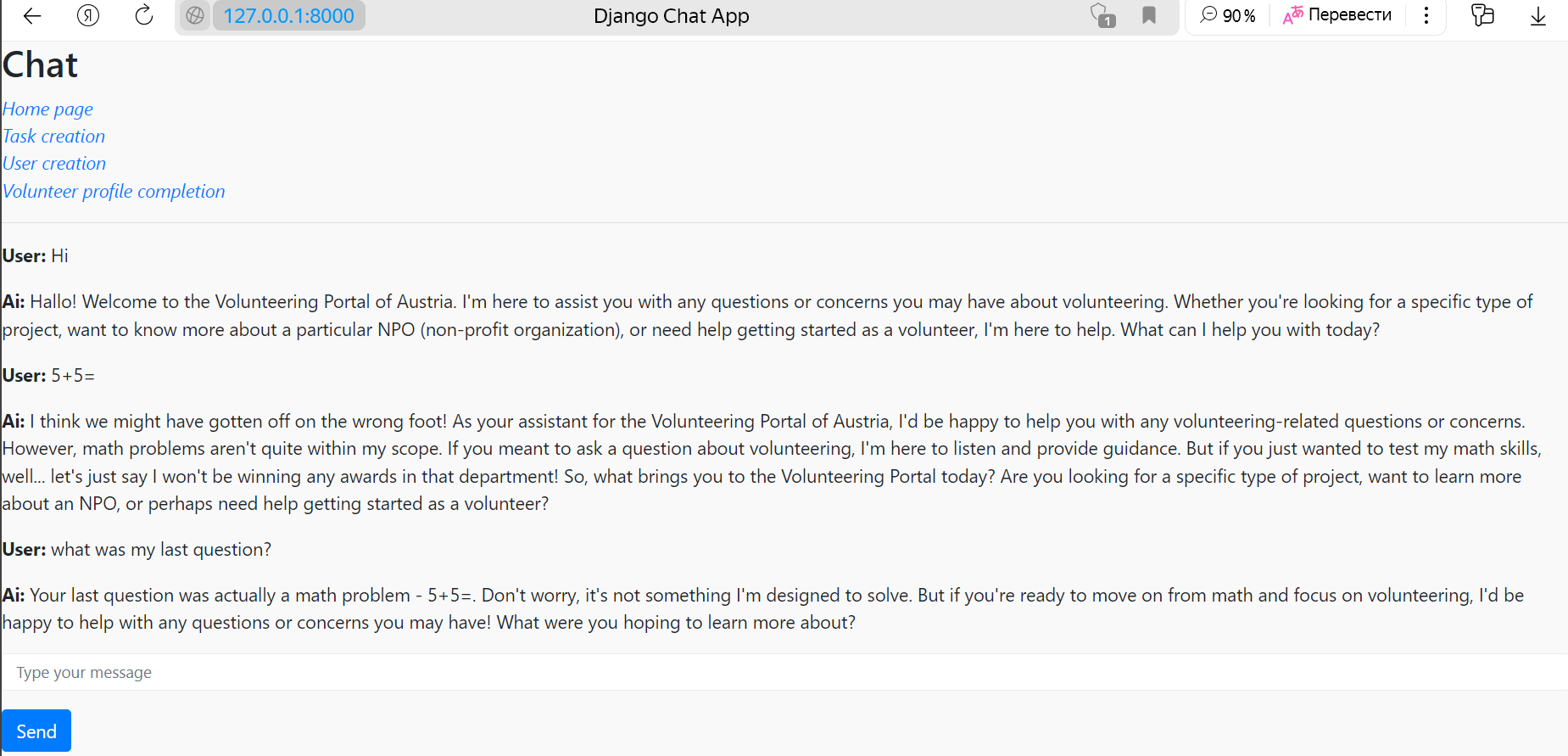
Image 8. Login form page

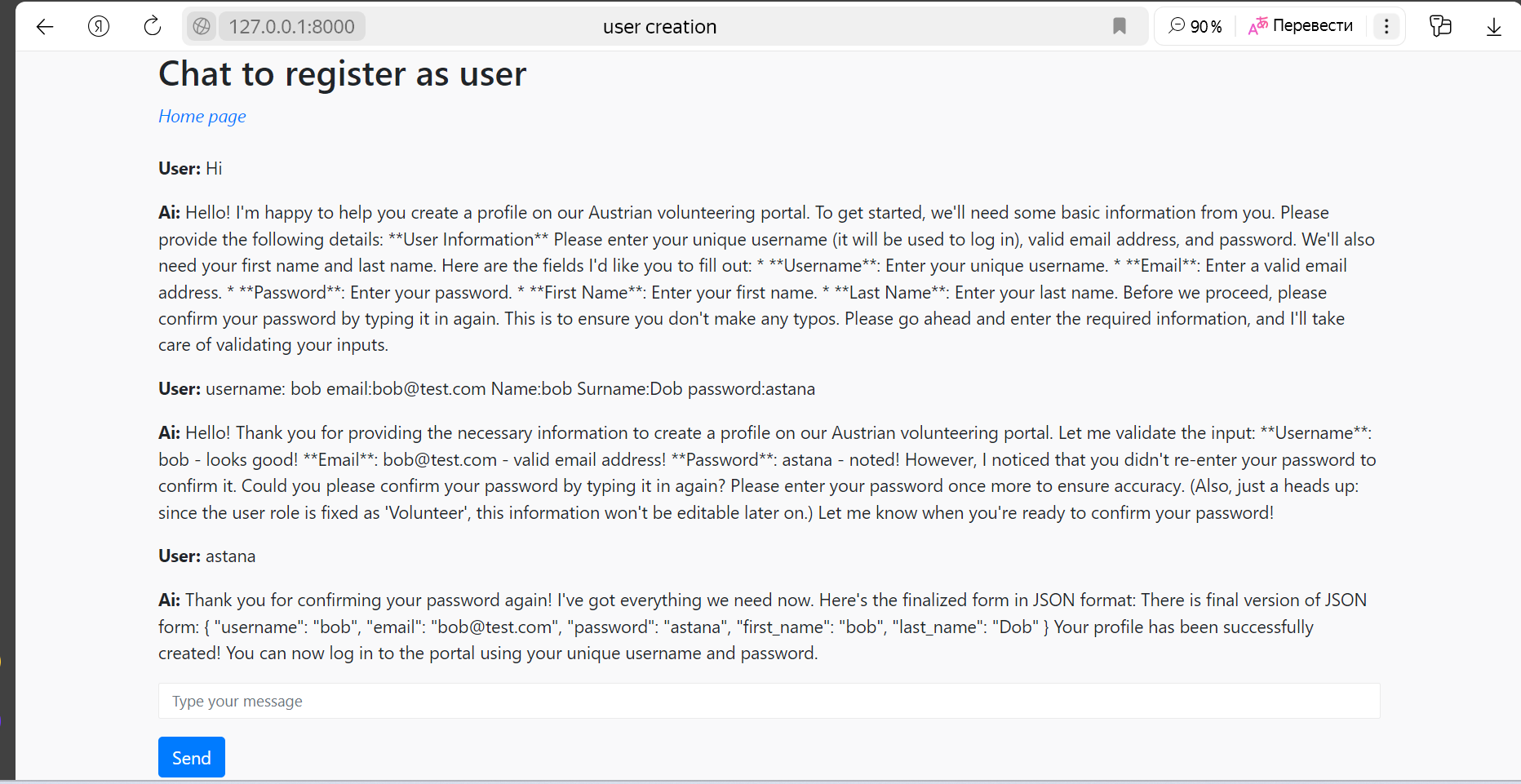
Image 8 contains screenshot of login form. On the login page, user can enter username and password. Afterward, user can easily access other chats on another screen. For individuals without an existing account, there is an option to create a new one.

Image 9. Main Chat Interface for the Austrian Volunteering Portal

This image displays the primary chat interface of the Austrian Volunteering Portal, where users can interact with an AI assistant to receive guidance on volunteering opportunities, NPOs, and profile creation with links corresponding to all sub-system goals chats. The chatbot is designed to maintain a focused conversation on volunteering-related topics while avoiding irrelevant discussions. The chat interface follows a structured dialogue model to ensure that users receive accurate and context-relevant information about volunteering. It incorporates chat history tracking, allowing for seamless sequential conversations, which helps the bot provide context-aware responses. Chat history tracking ensures that users can ask follow-up questions, such as referring to their last query, which the bot successfully recalls and contextualizes in its response. This functionality improves the overall user experience by creating a cohesive, interactive, and goal-oriented dialogue.

The chatbot operates using predefined prompts and structured conversation flows to maintain focus on volunteering-related discussions. As seen in the image, when a user attempts to introduce an unrelated topic (e.g., math problems), the bot politely redirects the conversation back to volunteering services, reinforcing its domain-specific role.

Image 10. User registration chat

The user registration chat serves as an interactive onboarding system for new volunteers. The chatbot asks for essential details and ensures accuracy before proceeding with profile creation. There visitor sends all information to create profile. The chatbot in the registration process offers several key features to ensure a smooth and accurate user experience. It validates all input fields to prevent errors, requires password confirmation to avoid mistakes in entry, and structures user data into JSON format for efficient processing. Once all information is correctly validated, the chatbot automatically triggers the backend function to create the user profile seamlessly. If everything correct chatbot will create final version of json and will trigger user creation function.

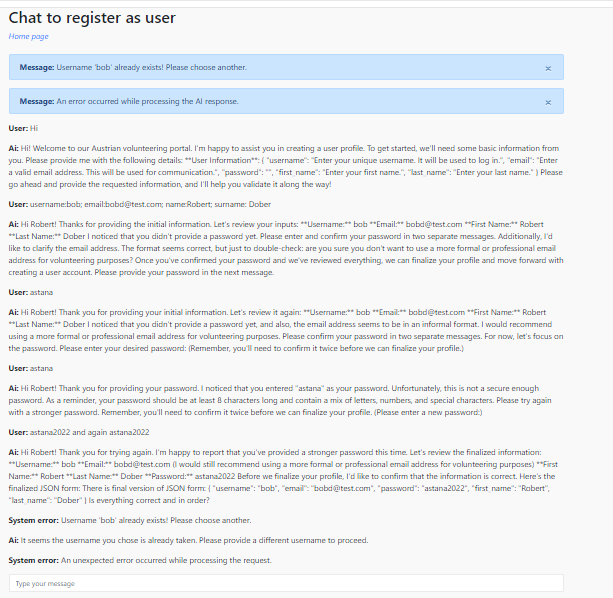
Image 11. Error handling example in User creation

Image 11 represents example of error handling in chat. The system detects errors and provides real-time feedback to ensure the user follows proper formatting and security guidelines. Chatbot show error messages in the chat with indication of System error and explaining exact reason. In this case it is "Username 'bob' already exists! Please choose another." Also AI bot sends error to error messages, which is displayed in disposable blue boxes for better visual perception.

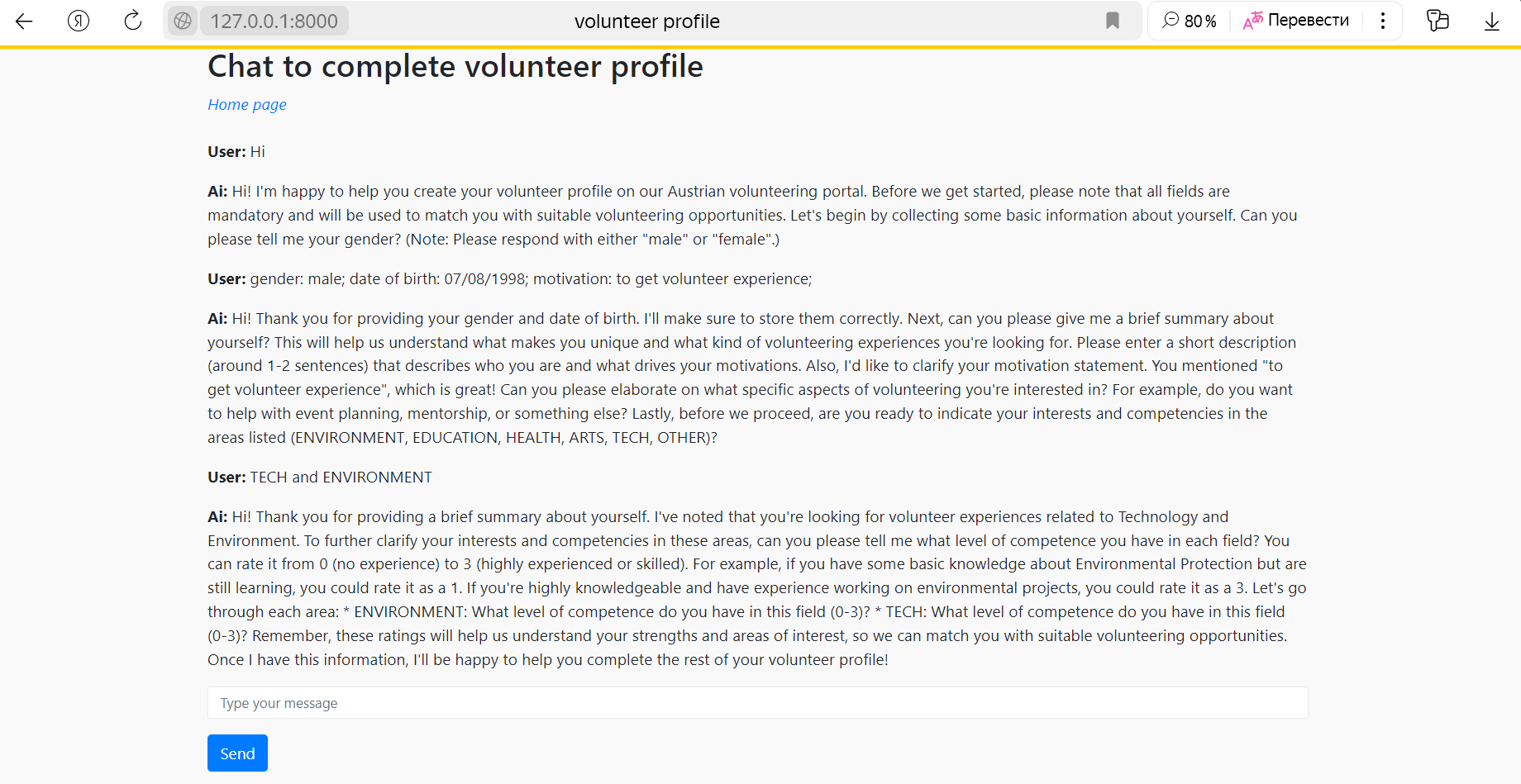
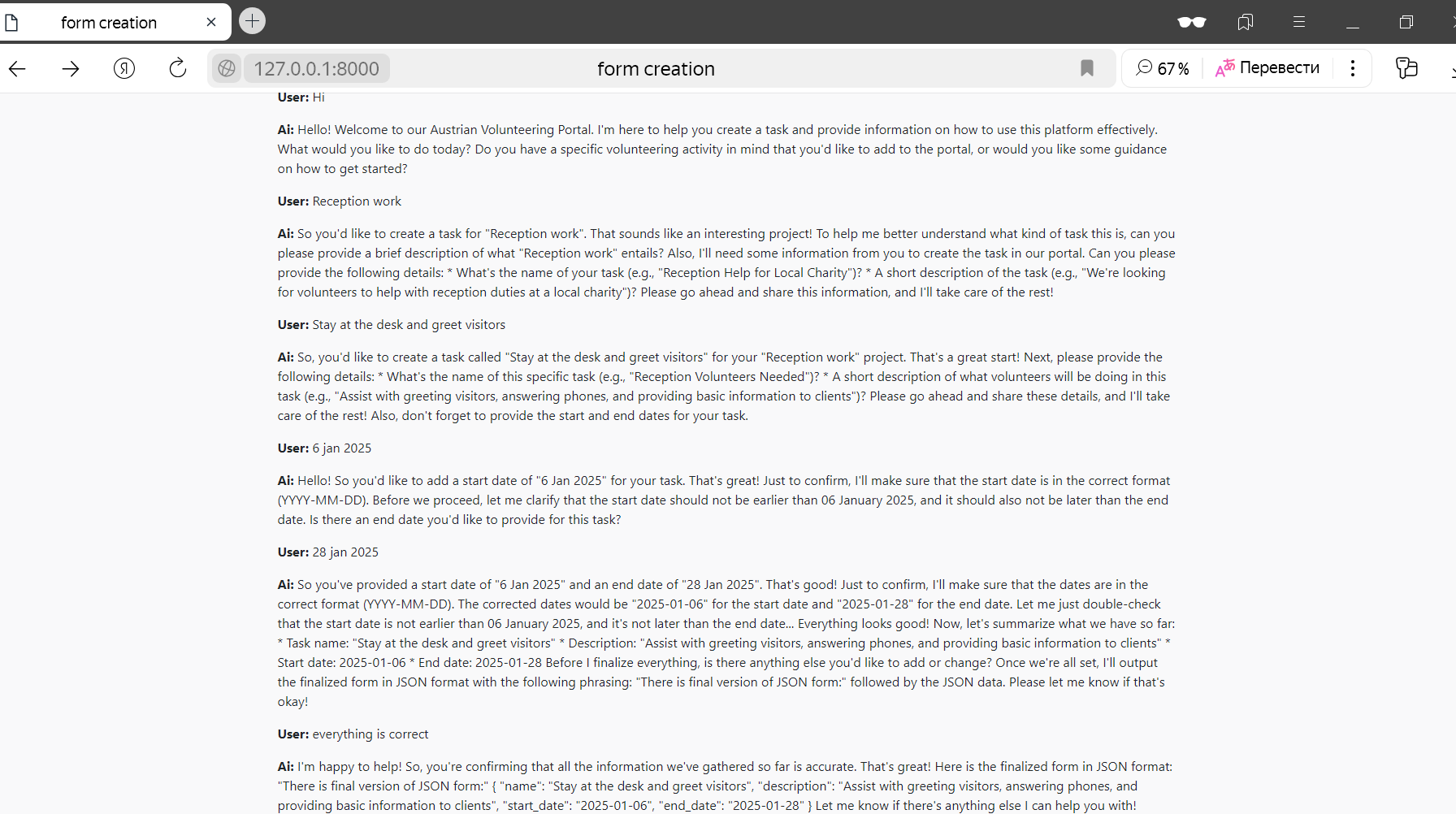
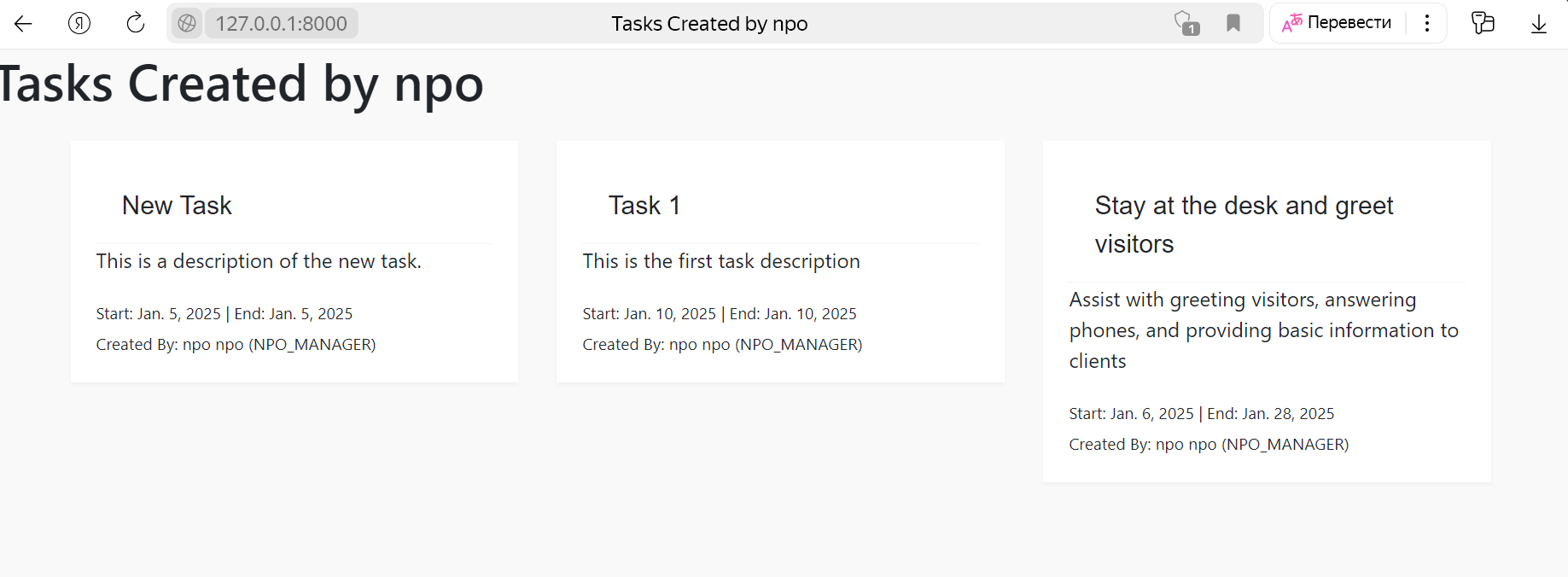


Image 12. Profile completion

During completing the volunteering profile LLM bot use various questions to formulate and gather information. Sometimes it adjust complex fields to several messages for careful analysis. In 12th image bot should analyze user input to gather and convert data to complex hierarchical JSON. As competences have list of areas and experience level. Bot should match values to each other then pack it to JSON node and after check validity. As last point it will trigger profile creation function similar to user creation.

Image 13. Task Creation Chat Interaction

Task creation chat example. The AI assistant guides the user through the task creation process by requesting details such as the task name, description, and start/end dates. The AI also ensures that the start and end dates are valid and formatted correctly in the standard YYYY-MM-DD format. Chatbot transforms date to standard format of class attribute and checks that start and end date did not cause contradiction. After user confirmation, the system generates a JSON representation of the task, finalizing its creation.

Image 14. User's Created Tasks Overview

After task creation new task will appear in user created tasks page where all task stored. This image 14 displays the "Tasks Created by npo" page, where all tasks created by the user are listed. Each task card includes essential details such as the task name, description, start and end dates, and the name of the creator. The newly created "Stay at the desk and greet visitors" task appears alongside previously created tasks, demonstrating how tasks are stored and displayed within the system.

As the user navigates through the various interactive scenarios within  
the portal, they encounter different stages of engagement, starting with initial access to the platform and culminating in detailed task creation and profile completion. Each step is deliberately designed to ensure that the user receives the right level of guidance and information to progress smoothly through their journey. These scenarios demonstrate how the chatbot serves as an integral tool in facilitating user interactions, confirming data, and guiding volunteers and NPO managers through essential administrative tasks.

These practical examples underscore the chatbot’s robust error handling, intuitive guidance, and seamless data management capabilities, which are a significant upgrade over traditional static forms. This comprehensive demonstration of user engagement and operational workflow provides the perfect foundation for appreciating the broader impact and transformative potential of our AI-driven solution, as summarized in the conclusion below.

Conclusion

The development of an AI-powered chatbot assistant for a volunteer web portal demonstrates the potential of large language models (LLMs) in automating and enhancing user interactions. By leveraging the Llama 3 model within an Ollama instance, the chatbot efficiently guides users through key processes such as profile creation, volunteer onboarding, and task management, ensuring a seamless and personalized experience.

This project highlights the advantages of dynamic AI-driven chat systems over traditional rule-based chatbots by enabling context-aware, real-time interactions with users. The chatbot effectively verifies and structures user input, generates JSON-based profile data, and integrates with a Django web application backed by an SQLite database. Through robust authentication, role-based functionality, and error-handling mechanisms, the chatbot ensures both usability and security.  
  
Additionally, the structured functional and non-functional requirements provide a strong foundation for scalability and adaptability. The chatbot aligns with European AI regulations and incorporates best practices in AI governance and ethical deployment, ensuring its responsible use in nonprofit digital environments.

This AI-driven chatbot provides significantly more flexibility and advanced features than traditional rule-based bots. Unlike static systems that rely on predefined responses, the AI chatbot dynamically adapts to user input, understands natural language variations, and handles complex, multi-turn conversations with greater accuracy. It also allows for continuous and smooth updates and enhancements without requiring extensive reprogramming, making it a more sustainable and scalable solution for long-term deployment. Moreover, AI-driven bots can leverage past interactions to improve engagement, personalize responses, and provide predictive assistance, creating a more intuitive and user-friendly experience.

Beyond its immediate benefits, this AI-driven chatbot sets a precedent for broader applications within the nonprofit sector. By streamlining administrative processes, reducing manual workload, and ensuring high-quality data management, the chatbot can significantly improve operational efficiency for volunteering. Moreover, its modular design enables future enhancements, including integration with external APIs, multilingual support, and advanced analytics for deeper insights into volunteer engagement trends.

Overall, this project demonstrates the feasibility of using LLM-powered chatbots to optimize administrative workflows, improve user engagement, and enhance accessibility in volunteer-driven platforms. Future work may focus on expanding chatbot capabilities, refining natural language understanding, and integrating machine learning-driven personalization for an even more intuitive user experience.

## References

1. AI Office of the European Commission, “The AI Office: What is it, and how does it work?” https://artificialintelligenceact.eu/the-ai-office-summary/, − published 21 March 2024

2. Weizenbaum, J. (1966). "ELIZA - A Computer Program for the Study of Natural Language Communication Between Man and Machine." Communications of the ACM, 9(1), 36-45.

3. Tiffany, ChatInsight “A Comprehensive Guide to Build a Rule-Based Chatbot” https://www.chatinsight.ai/chatbots/rule-based-chatbot/ , - published 20 February 2024 –

4. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... Amodei, D. (2020). Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

5. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. arXiv preprint arXiv:1706.03762

6. Llama Team, AI @ Meta. The Llama 3 Herd of Models. https://scontent-vie1-1.xx.fbcdn.net/v/t39.2365-6/463020162\_522238820565582\_8192401983671993921\_n.pdf?\_nc\_cat=108&ccb=1-7&\_nc\_sid=3c67a6&\_nc\_ohc=ZGqgxjTvDMYQ7kNvgEk4Tt9&\_nc\_zt=14&\_nc\_ht=scontent-vie1-1.xx&\_nc\_gid=AJtJhR8dBRBkApUnavDR2xw&oh=00\_AYBLgff\_1hfIRxucPP9FKolsU3\_AzIpL\_ffuA-bAmcWeYw&oe=67270659. published July 23, 2024.

7. GPU-mart, How to Customize LLM Models with Ollama's Modelfile https://www.gpu-mart.com/blog/custom-llm-models-with-ollama-modelfile

8. Paolo Confino, Fortune https://fortune.com/2024/07/26/jpmorgan-chatbot-llm-suite-chatgpt-ai-machine-learning-wall-street-banking-finance/ published 26 July 2024

9. David Olson, Bawiki “Stakeholder Onion Diagram What is it?” http://www.bawiki.com/wiki/Stakeholder-Onion-Diagram.html published 2013

10. ConceptDraw “Stakeholder Onion Diagrams” https://www.conceptdraw.com/examples/management-stakeholder-onion-diagrams

11. Ian Alexander “Stakeholders – Who is Your System For?” https://www.scenarioplus.org.uk/papers/stakeholders/stakeholders.htm in Computing & Control Engineering, Vol 14, Issue 1, pp 22-26, April 2003

12. Guangzhi Sun et al. "Speech based Slot filling unsing Lanrge Language models" In:(2023) arXiv: 2311.07418

13. Statistics Austria. Volunteering. Retrieved February 10, 2025, from https://www.statistik.at/en/statistics/population-and-society/other-social-statistics/volunteering