UDC: 004.8:004.855:005.94:331.108 JEL: M15, M12, O33

AUTOMATED RESUME FORMATION USING GENERATIVE ARTIFICIAL INTELLIGENCE

Vilkhivska O., Teslia O.

Introduction and Relevance of the Problem. Artificial intelligence systems serve as a driving force of contemporary progress and exert a substantial influence on virtually all spheres of human life, including education, medicine, finance, electronic commerce, and the labour market. Owing to their capacity to analyse large volumes of data and automate complex processes, such systems have become effective tools for supporting decision-making and enhancing productivity.

One of the pressing challenges in the modern labour market is the rapid search for qualified personnel and the effective presentation of candidates' professional competencies. The quality of a resume's structure, content, and formatting largely determines success at the initial stage of vacancy selection. At the same time, the preparation of an effective resume requires not only knowledge of generally accepted standards but also the ability to describe one's professional skills concisely and persuasively, to adapt the text to a specific position, and to take into account the requirements of automated candidate selection systems.

The rapid advancement of generative artificial intelligence and large language models, particularly technologies accessible via the OpenAI API, has created new opportunities for the automation of textual document creation. Such models can analyse a brief description of a user's professional experience and generate logically structured, grammatically correct, and meaningful resumes that are tailored to a specific vacancy and optimised for Applicant Tracking Systems (ATS). The use of generative language models enables the automation of professional text formation, improves its quality, and significantly reduces the time required to prepare the document.

Significant contributions to the development of generative artificial intelligence technologies have been made by both Ukrainian and foreign scholars, including Yu. Stativka, O. Gorokhovatskyi, S. Pereiaslavska, O. Smagina, V. Buslaiev, O.

Ihnatenko, Yu. Paniv, Aidan Gomez, Ashish Vaswani, Noam Shazeer, Jakob Uszkoreit, Wayne Xin Zhao, Kun Zhou, Mina Lee, Percy Liang, and Qian Yang.

Ukrainian researchers focus on both fundamental and applied aspects of machine learning and natural language processing. Their work encompasses the development of algorithms for intelligent data processing, modelling of linguistic interaction, creation and adaptation of language models, as well as the improvement of methods for text classification and analysis [1–4].

In turn, the authors A. Vaswani, A. Gomez, N. Shazeer, J. Uszkoreit, and colleagues, in their seminal paper “Attention Is All You Need,” established the architectural foundations of modern large language models, which have defined the subsequent development of generative artificial intelligence [5].

The systematisation of contemporary approaches to large language models (LLMs) is presented in the works of Wayne Xin Zhao and Kun Zhou, who provide a comprehensive overview of training methods, scaling techniques, and applications of LLMs. Practical aspects of using generative AI for automated resume creation are examined in studies devoted to the application of GPT- and Llama-type models for generating personalised documents in accordance with vacancy requirements [6].

A distinct research direction is represented by the works of Mina Lee, Percy Liang, and Qian Yang, who explore language models as tools for human–AI collaborative writing, thereby enhancing the quality and efficiency of text creation. Additionally, applied Ukrainian developments – such as the adaptive resume generation system CViWantCV and the HR AI platform HURMA AI – demonstrate the practical implementation of AI technologies in the automation of professional document preparation, including the analysis, structuring, and personalisation of candidate data. Collectively, these studies form the scientific and technical foundation of modern generative AI systems capable of performing the full cycle of text processing – from analysis and information extraction to generation, editing, and adaptation of documents to specific usage contexts [7].

Of particular relevance is the development of proprietary software solutions that integrate modern web technologies with the capabilities of artificial intelligence.

Such systems provide content personalisation, multilingual support, integration with databases, and scalability for use in HR platforms, career services, and educational portals.

Thus, research focused on the automated generation of resumes using generative artificial intelligence is timely and practically significant. The developed software module offers broad prospects for application in electronic recruitment systems and platforms supporting users' professional development.

Main Body of the Article. In the modern digital world, intelligent assistants have become an integral part of human interaction with information systems. They are a separate class of software systems that interact with the user in natural language, analyze queries and perform actions in accordance with the tasks set.

An intelligent assistant is a software system that is able to interact with the user in natural language, analyze queries, store context, learn from experience and perform actions on behalf of the user. Such systems are based on a complex of artificial intelligence technologies, natural language processing methods and large language models [8–10].

The main goal of an intelligent assistant is to reduce the cognitive and time load on the user by automating routine or complex tasks, ranging from information search to the generation of text documents, including professional resumes.

The main characteristics of intelligent assistants include [11]:

the ability to process natural language, the ability to perceive queries in a form familiar to a person without using special commands;

adaptability and learning, improving responses and actions taking into account the behavior of a specific user;

automation of processes, execution of complex sequences of actions;

contextuality, preservation and use of the history of interaction, interests and user situation;

modularity, architecture in the form of a set of interacting components, each of which is responsible for a separate function [12–13].

Depending on the functional capabilities, method of interaction, level of autonomy and implementation architecture, intelligent assistants differ significantly. Therefore, it is advisable to consider their classification according to the main criteria.

According to their functional purpose, they are: universal assistants, specialized, educational, business assistants (Fig. 1).

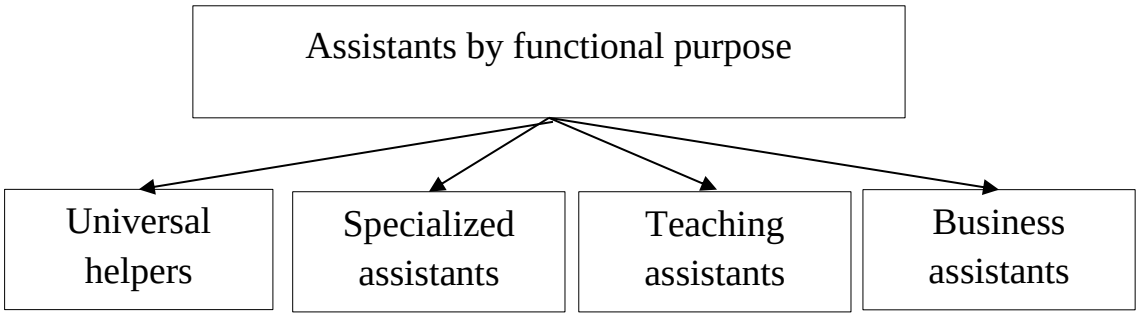


Figure 1. Assistants by functional purpose

Universal intelligent assistants, such as Google Assistant, Apple Siri and Amazon Alexa, are the most common representatives of this class of systems and are designed to perform a wide range of everyday tasks. Their key features are high integration with the ecosystems of the respective companies, support for multimodal interaction using voice, text, visual interface and the ability to understand context. These assistants provide proactive assistance to the user: from searching for information, controlling the smart home and navigation to creating reminders and managing the calendar.

Google Assistant has powerful search capabilities, Siri is distinguished by its emphasis on privacy and local data processing, and Alexa has a developed ecosystem of third-party skills and dominance in the IoT segment. Despite their high versatility, such systems are inferior to specialized assistants in deep personalization and the quality of performing narrow-profile professional tasks, in particular, generating structured documents such as resumes [14].

Specialized AIs are focused on performing specific tasks in a specific subject area, which allows them to provide significantly higher quality and relevance of results compared to universal systems. Unlike universal AIs, they have deep

knowledge in a narrow field, are able to work with domain-specific terminology and adapt to professional requirements. Examples of such systems are HR bots for personnel selection, financial consultants, legal assistants and assistants for creating career documents. In the framework of this study, a specialized text-based AI for automated generation of professional resumes is of particular interest.

Such systems use large language models to analyze the input prompt, create structured content, adapt style and tone to the industry and position, which allows generating personalized, ATS-friendly resumes based on minimal information from the user [15].

Educational AIs are a specialized class of systems focused on supporting the educational process. They help users learn, explain complex concepts, create personalized learning paths, and prepare for exams. Unlike universal assistants, such systems have a deep understanding of pedagogical methods, are able to analyze the user's level of knowledge, identify gaps, and adapt the complexity of the material in real time.

A striking example is the intelligent modules of the Duolingo platform, which, based on the analysis of the student's success, form personalized lessons, offer the optimal sequence of tasks, and adjust the complexity depending on progress. Modern learning assistants also actively use large language models to generate explanations, tests, exercises, and adaptive content, which makes them powerful tools for both self-study and use in educational institutions.

Business assistants are a specialized type of intelligent systems designed to automate and optimize business processes in a corporate environment. They integrate with internal company systems, such as CRM, ERP, accounting, project management and email systems, and perform tasks related to data analysis, reporting, resource planning, document processing and support for management decision-making.

Unlike universal assistants, business assistants have a deep understanding of business logic, corporate terminology and specific industry requirements. They are able to automate routine operations, generate analytical reports, forecast indicators and provide recommendations based on company data. Modern business assistants,

built on the basis of large language models, can also prepare commercial offers, analyze contracts, form KPIs and support communication with customers, which significantly increases employee productivity and the quality of business processes.

By implementation architecture, assistants are divided into local, cloud and hybrid. Local ones provide a higher level of privacy, cloud ones provide more powerful computing capabilities, and hybrid ones combine the advantages of both approaches.

The evolution of intelligent assistants has gone from simple rule-based systems to complex neural networks. A particularly significant breakthrough was the introduction of large language models, such as GPT-4, which demonstrate high quality natural text generation, contextual awareness, and flexibility. This has opened up new prospects for creating highly specialized assistants, in particular for the automated creation of professional documents [16–18].

Despite significant progress, intelligent assistants still face a number of challenges: possible generation errors, data confidentiality issues, language barriers, and dependence on an Internet connection.

Awareness of these limitations and existing gaps in the functionality of existing services has led to the need to create a specialized intelligent tool.

The developed smart assistant module is built on the principles of client-server architecture, which provides a clear separation of responsibilities, scalability and ease of system support. The client part is responsible for collecting input data, providing an interactive user interface, displaying the history of generated resumes and downloading finished documents in PDF format. The server part performs the main business logic: user authentication and authorization, query processing, interaction with external APIs, in particular OpenAI, content generation, creation of PDF documents, data storage in the database and access rights control.

The diagram of the components of the developed IS is shown in Fig. 2.

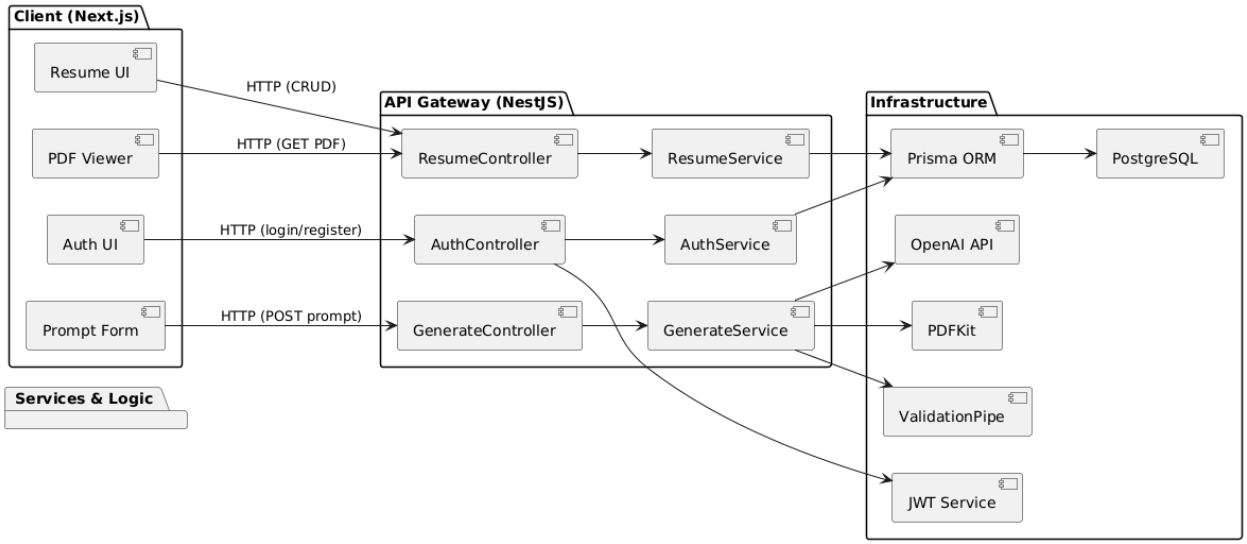


Figure 2. Diagram of system components.

Each request coming from the Next.js client part is processed by the controller of the corresponding module. The project structure is built according to a modular approach, where the main functional blocks are logically separated:

The auth module is responsible for user registration, authentication using JWT, token storage and route protection.

The resume module handles CRUD operations on the resume: create, retrieve, view and delete.

The generate module performs content generation using the OpenAI API, and also forms a PDF document based on the generated text.

The prisma module initializes the client to access the PostgreSQL database via the Prisma ORM.

The input data is checked using ValidationPipe and the class-validator library, which makes it impossible to process invalid objects. All critical routes are protected by middleware that checks the presence and validity of the JWT token. Authorization is implemented via the @nestjs/passport package using the JWT strategy.

The resume content is generated by passing a text request received from the client to the OpenAI API. The response is processed and formatted as structured text. At the next stage, the system generates a PDF file using the PDFKit library. The created document is saved to a buffer using get-stream and written to a database with a link to a specific user.

PostgreSQL is used as a storage, access to which is implemented via Prisma ORM. All configuration variables, access keys to OpenAI, tokens, and database parameters are stored in the .env file and read via the dotenv package [18-20].

All routes related to creating, viewing, downloading, or deleting a resume are inaccessible to unauthorized users. Token verification is performed on each request. Access is granted only in the case of a valid JWT.

The module works in the following sequence of actions. After registering or logging in, the user goes to the form for entering a brief description of their activities, experience, or desired position. This request is sent to the server, where content generation is performed using the OpenAI API and GPT-4. The resulting text is formatted into a resume structure with sections: Summary, Skills, Experience, Education, etc. The document is then stored in the database and simultaneously sent to a PDF generator, the result of which the user can view and download [21].

The class diagram of the developed assistant is shown in Fig. 3.

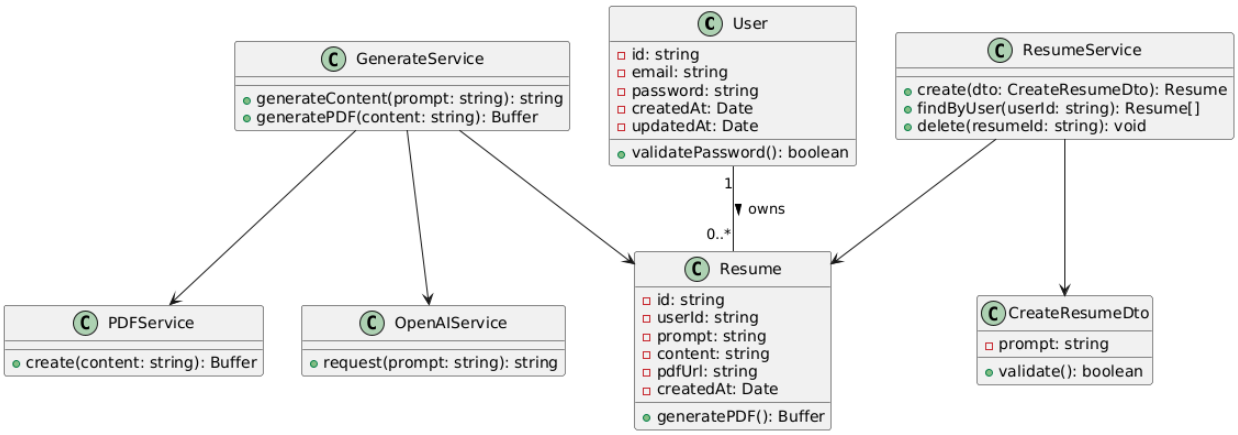


Figure 3. System class diagram

The system provides the functionality of saving the history of generations, which allows the user to return to previously created documents. A mechanism for deleting or updating an existing resume is also implemented, which provides convenient data management and flexibility in preparing for applying for various vacancies.

An important aspect of the task is to take into account modern requirements for UX/UI design. The module interface should be intuitive, minimalistic, and adapted for use on different devices. This allows for comfortable interaction with the system

even for users without technical experience. The project uses modern Next.js, Tailwind CSS, and Shadcn UI frameworks for this, which allow implementing a high-quality adaptive frontend [22-24].

To visualize the structure, logic of operation, and behavior of the developed smart assistant module for creating a resume, four types of UML diagrams were created. Their construction was carried out in accordance with the UML 2.x standard and reflects both static and dynamic aspects of the system (Fig. 4.)

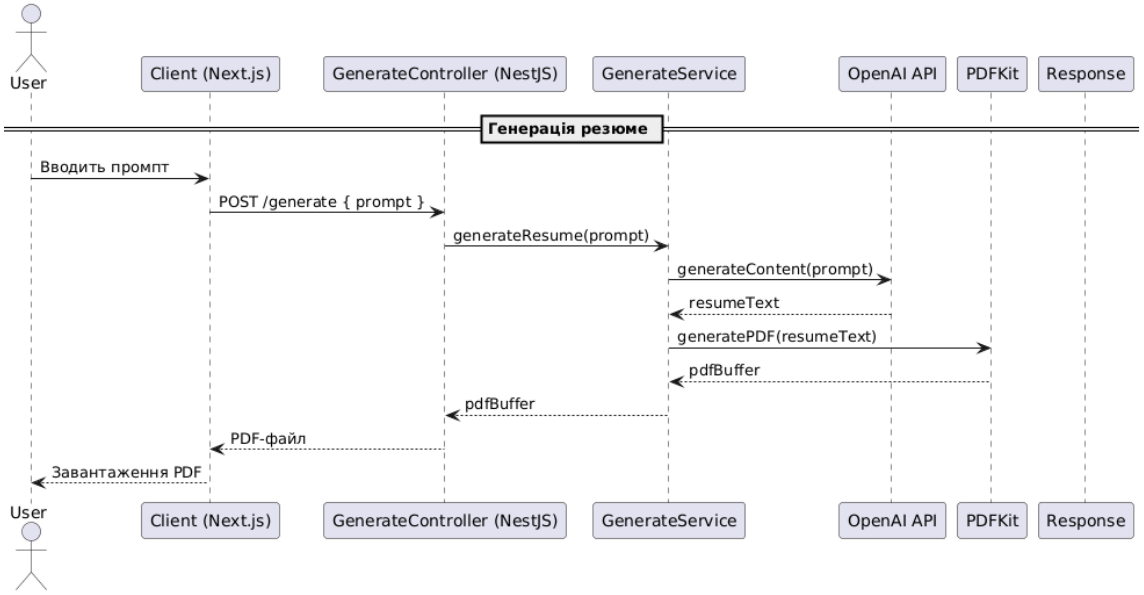


Figure 4. Resume generation flow chart

The assistant allows you to not only generate template text, but also create a full-fledged professional document that is adapted to the industry, position, and level of experience of the candidate. This means that the system must form the content of the resume not only from the point of view of filling in the structure, but also take into account the writing style, tone, key terms, relevant skills, and wording that meet market requirements. These are the opportunities provided by the integration of generative artificial intelligence.

To implement the task, a technology stack is used that provides both flexibility in development and a high level of security, typing, and scalability. TypeScript on the client and server sides allows you to reduce the number of errors, provides a clear code structure, and allows you to implement new functionality faster. NestJS is used as a backend framework that implements a strict architectural model with controllers,

modules, and services. Prisma provides typed ORM work with the PostgreSQL database, which allows you to conveniently store user objects and generated resumes.

The problem formulation also takes into account the need for authorization and data protection. Authentication is implemented based on JWT tokens, routes are protected by middleware that checks the user session. All sensitive data is transmitted using the secure HTTPS protocol. Password hashing using bcrypt is used when saving accounts [25-26].

The project provides support for a multi-user environment with a separate administrator interface. The administrator has the ability to view the list of registered users, the number of created resumes, and also monitor suspicious activity. This allows you to increase the stability and security of the system when expanding its audience. The system usage diagram is shown in Fig. 5.

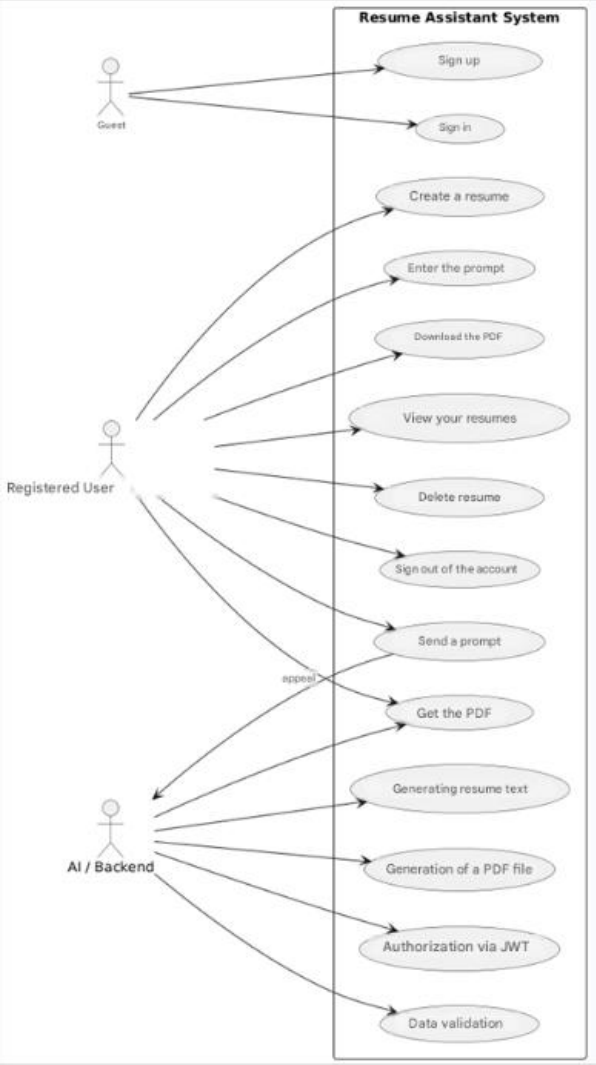


Figure 5. Diagram of system use cases

The developed assistant provides for the implementation of the full lifecycle of working with a document: from entering a query to saving it, reviewing it again, editing it, and exporting it. Accordingly, the development includes not only the creation of a generation core, but also the design of interfaces, the construction of an architecture with separation of responsibilities, the organization of data storage and scaling, integration with external services, and ensuring data security.

In the structure of a client-server application, requirements are distinguished for the client part (frontend), the server part (backend), as well as for the database, network interaction, and external integrations. Particular attention is paid to the protection of personal data, ensuring fault tolerance, logging of critical events, and scalability of the solution.

The Next.js framework was used to develop the client part, which combines the advantages of server and client rendering, which ensures high performance, SEO optimization, and improved page loading [27-29].

Tailwind CSS is used for styling, which allows you to create responsive, lightweight, and scalable interfaces without excessive dependence on third-party templates. The Shadcn UI component library is used to build modern interface elements with accessibility in mind.

Form validation is implemented via React Hook Form in combination with Zod for type-safe input data validation. Global state management is carried out via Redux Toolkit, which provides a centralized structure, convenient scaling, and support for asynchronous requests. React Query (TanStack Query) is used for caching and query management, which minimizes the number of server calls and provides automatic data updates.

The client part must support work on all modern browsers, namely: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari in their current versions. The interface must be fully responsive and display correctly on devices with different resolutions.

The server part is implemented using the NestJS framework for building scalable server applications based on Node.js and TypeScript. It provides a modular

architecture, a clear separation of responsibilities, and a convenient organization of controllers, services, middleware, and guards [30-36].

The main server logic includes the following functional blocks:

processing authentication and authorization (JWT, Passport.js);

receiving a user request (prompt);

interacting with the OpenAI API (using REST requests to GPT-4);

forming the structure of the generated resume;

generating a PDF document (using PDFKit or similar libraries);

saving the result in the database;

logging user actions and system errors;

handling third-party service errors (timeout, rate limit, network fail).

All API endpoints must be protected according to the principle of access delimitation. For private routes, access token verification and user role verification are provided. In case of lack of authorization, the server returns the corresponding status code (401 or 403) with a message.

A PostgreSQL relational database is used to store users and created resumes. ORM Prisma provides type-safe interaction with the database, allows you to implement a clear schema with integrity constraints and relationships between entities (Fig. 6).

Two main tables are provided:User: id, email, password_hash, created_at;

Resume: id, user_id, prompt, content, pdf_url, created_at.

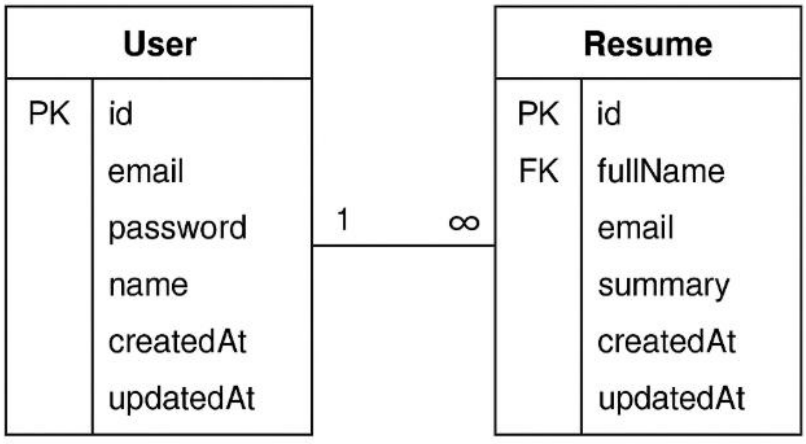


Figure 6. SQL schema of the Resume Assistant database

User data is stored in encrypted form. For passwords, hashing with salt is used, for example, via bcrypt. All foreign keys provide cascading deletion or restriction, according to the rules of business logic.

All critical actions related to generation, authorization, interaction with the API must be recorded in the event log. Logs are stored either in a separate table in the database, or through a third-party system, for example, Winston File, Graylog, Logstash.

The developed module works as follows. After successful authorization, the user goes to the form for entering a brief description of their professional experience, skills or desired position. This text prompt is transmitted to the server, where, using the GPT-4 - OpenAI API model, structured resume content is generated with the main sections: Summary, Skills, Experience, Education and others. The resulting text is stored in the database, a PDF document is generated in parallel, after which the user can view or download the result.

The system supports full document management: storing the history of generations, viewing, editing and deleting previously created resumes. This approach provides flexibility in preparing materials for different vacancies and allows the user to return to previously generated options.

Special attention is paid to the UX/UI user experience. The interface is designed to be minimalistic, intuitive and fully adaptive. Modern technologies are used to implement the client part, namely the Next.js framework, Tailwind CSS and the Shadcn/UI component library, which ensures high interface quality, speed and accessibility even for users without technical experience.

An intelligent tool has been developed that is capable of generating a full-fledged professional document adapted to a specific industry, position and level of experience of the candidate. The assistant takes into account the writing style, tone, key terms and relevant competencies, which significantly increases the quality and competitiveness of the resume.

User actions such as registration, login, resume generation, history viewing, and document upload are available only after authentication.

The registration page is designed to create a new account. The form contains three required fields: email, username, and password. The "Create Account" button initiates a request to the server API with subsequent data validation and saving of the new user in the database. At the bottom of the form, a text link is available to go to the authorization page for already registered users.

The windows of the new user registration pages with validation of the main fields and user authorization with the login form are shown in Fig. 7, 8.

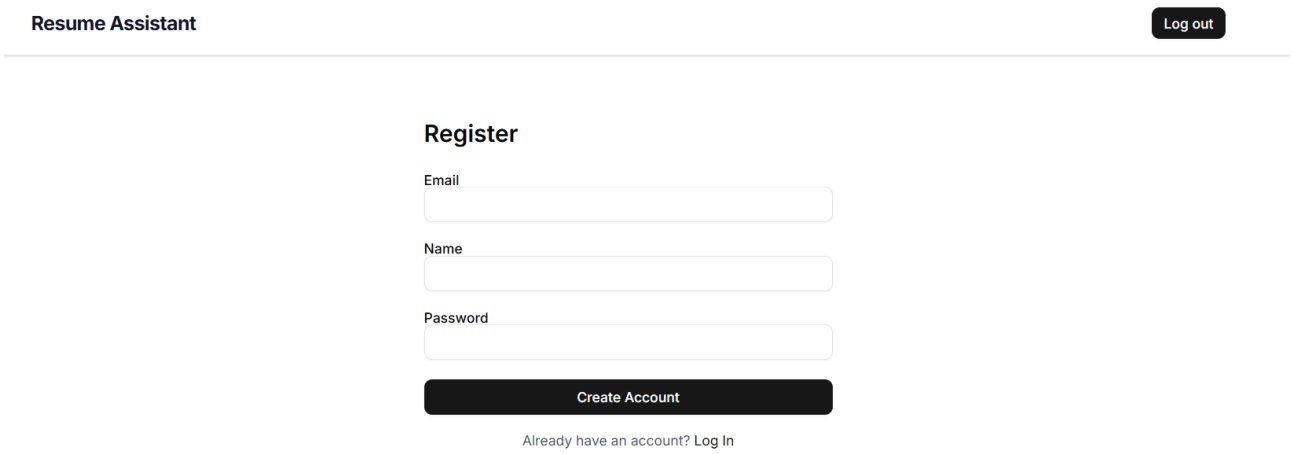


Figure 7. New user registration window

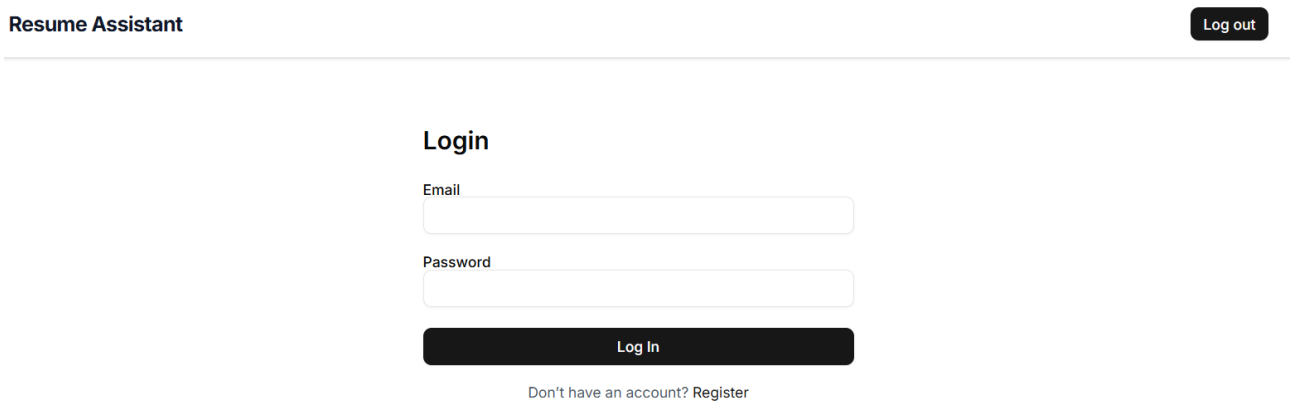


Figure 8. User authorization window

The authorization window includes required fields: email and password. After filling out the form, it sends a POST request to the API to verify the credentials and obtain a JWT token. The "Log In" button activates the authentication process. At the bottom of the page there is a navigation link to go to the registration form in case you do not have an account.

The window of the administrative table of users in the database, the User model, is shown in Fig. 9.

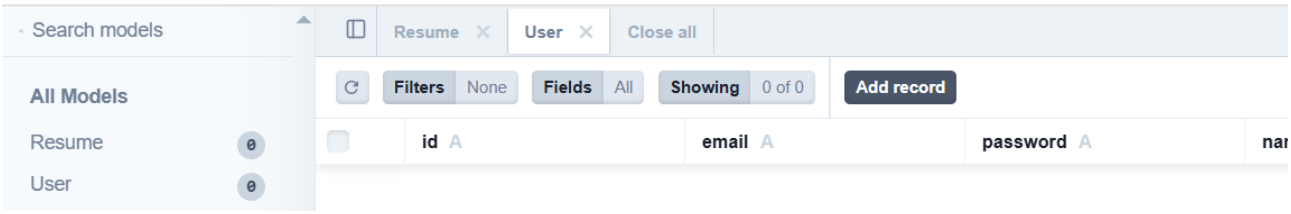


Figure 9 – Administrative user table window

The administrative panel manages the User model. The table stores information about registered users. The main fields are: id (user ID), e-mail (email address), password (encrypted password), name (user name), as well as additional fields such as registration date, user role, tokens, or other parameters defined in the database schema.

The administrative panel provides basic data management functions: viewing, filtering, sorting, adding new ones. All actions are performed within the scope of protected access, as evidenced by the restriction status.

The secret keys required for signing tokens and interacting with external services, in particular OpenAI, are stored in the .env configuration file, which is not published with the source code and is processed using the dotenv module.

Thus, the system guarantees: user identification with identity confirmation; preventing access to data by other persons; protecting critical information during storage and transmission; delimiting access rights.

After successful authorization, the client verifies the token, initiates requests to the corresponding API endpoints of the server side, and displays the response in the corresponding components. The main window of the client side with a description of the main purpose of the system and a call to generate a resume is shown in Fig. 10.

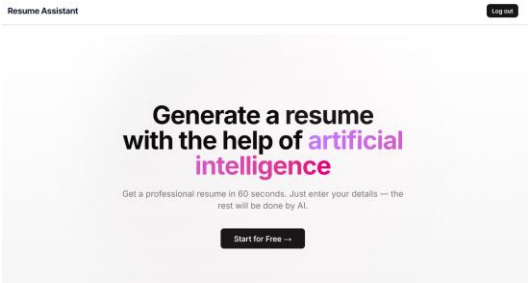


Figure 10. Main window of the client part

The main page displays information about the service, the message in the window focuses on the key functionality of the system. The interface is minimalistic, adaptive, with an emphasis on the central action, namely the launch of the generation process. The top panel contains the logo and the login button. The main call to action “Start for free” activates the transition route to the form of interaction with the system.

The information block with the characteristics of the system: speed, personalization, confidentiality is presented in Fig. 11.

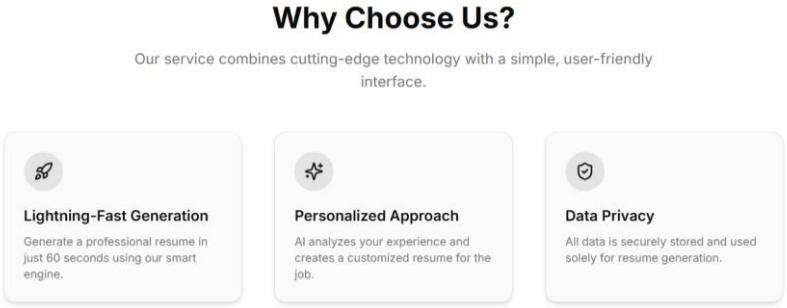


Figure 11. System characteristics information block

The information block contains the key advantages of the system: generation speed, personalized approach to creating a resume and secure storage of personal data. This block performs the function of informing the user about the main technological and practical differences of the system. The next window provides instructions for user interaction with the system, namely: entering basic information, automatic generation of resume content and downloading the generated PDF file. Each step is accompanied by an icon and a short text description, which provides a quick user familiarization with the basic logic of the service (Fig. 12).

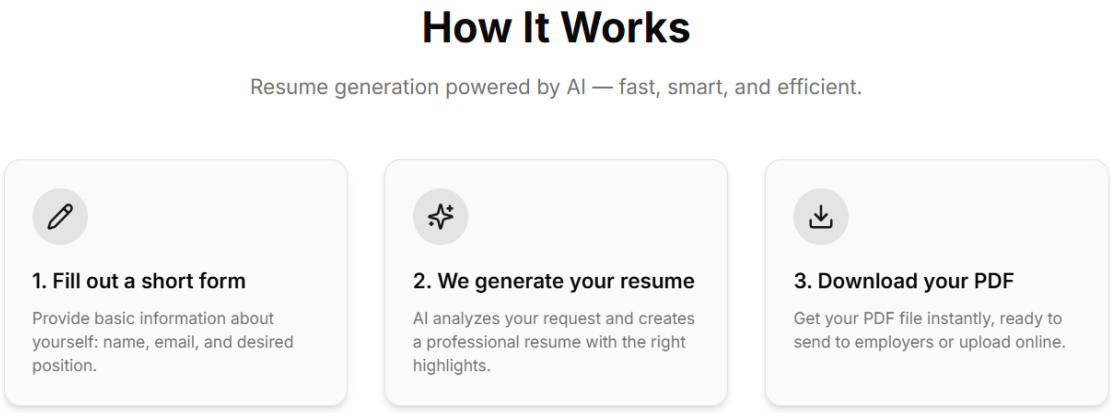


Figure 12. Data entry window, generation, upload resume

The next step opens a preview window of the generated resume with upload options and an explanation of the processing stages. The window includes blocks such as full name, position, short description, work experience, education, skills and languages. On the right are functional elements: a preview button and a PDF upload button. The interface implements visualization of the result before saving it (Fig. 13).

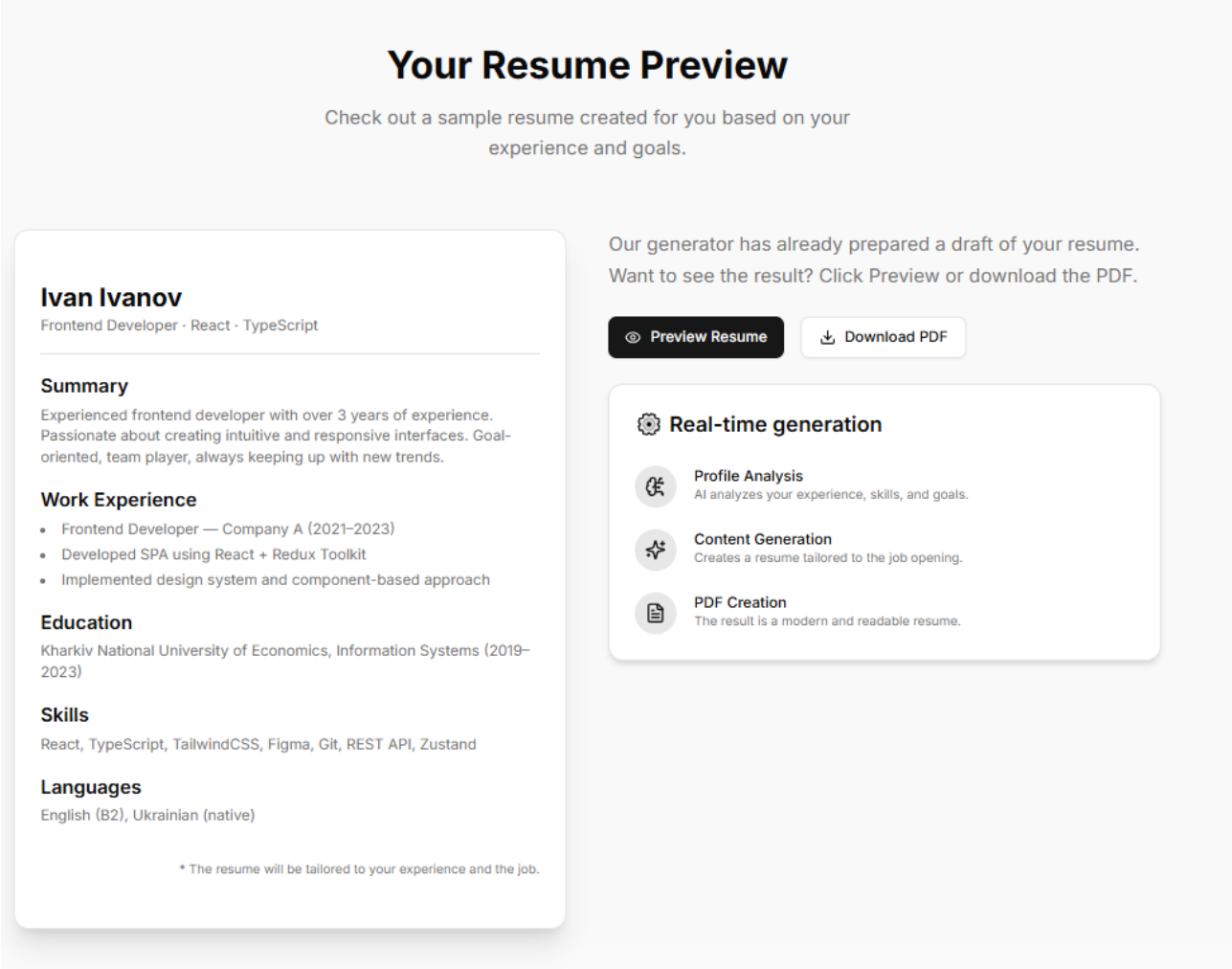


Figure 13. Preview window of the generated resume

The final block implements a sequence of steps: entering a prompt, automatically creating a resume based on templates and recommendations, and the ability to edit and download the final PDF file (Fig. 14).

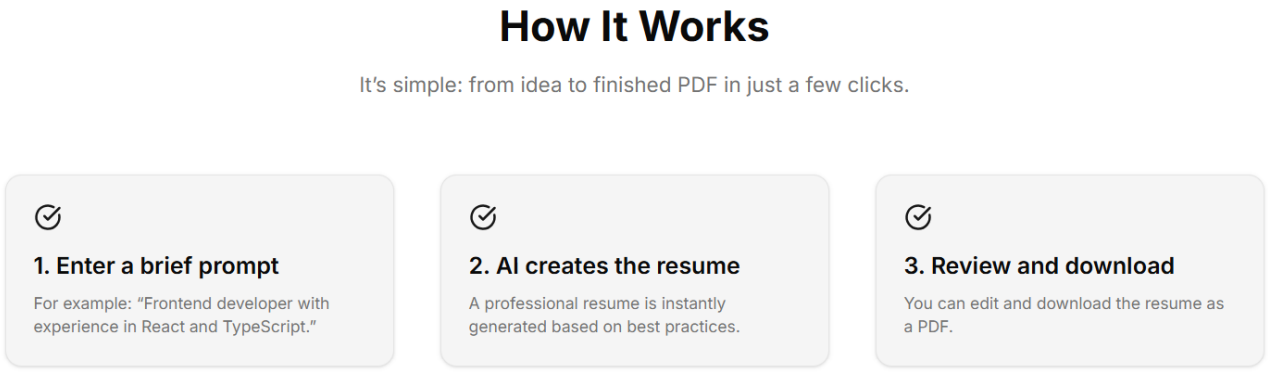


Figure 14. Request input, generation, PDF download window

All generated resumes are stored in the Resume table. The model is linked to the users table via a foreign key and contains the main data that is displayed in the final document: contact information, name, position, as well as technical metadata. Access to the table is restricted, editing and viewing are possible only in protected mode. (Fig. 15).

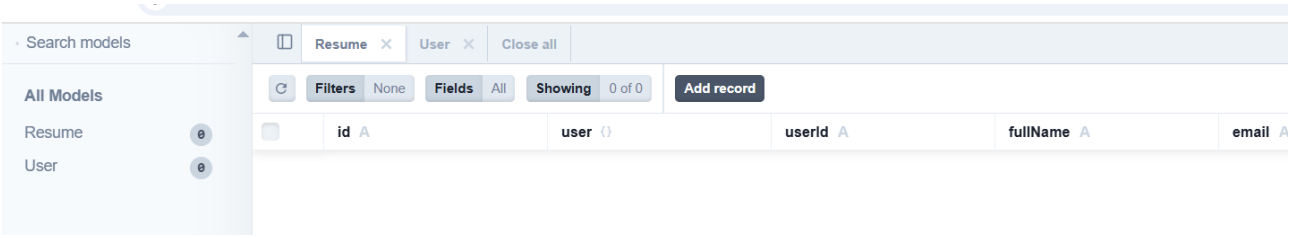


Figure 15. Generated resumes window in the administrative panel

The system implements a full-fledged user authentication and authorization mechanism in compliance with security requirements for the storage and transmission of personal data. To ensure controlled access to functionality, a token-oriented scheme based on JWT is used.

The authorization process is implemented using the Passport.js + @nestjs/passport connection, where a token is generated upon successful user authentication on the server side and transmitted to the client. In the future, the client is required to include this token in the request headers for access to protected endpoints. Verification occurs on each request by checking the signature and expiration date of the token.

User data is stored in the PostgreSQL database in encrypted form. Passwords are hashed using a cryptographic library (bcrypt or similar), which eliminates the possibility of storing or transmitting account information in plain text.

All routes related to viewing, creating, downloading or deleting a resume are protected by JWT Guard, which blocks unauthorized requests. When attempting to access without a token or with an invalid token, the user receives an access denied message (Fig. 16).

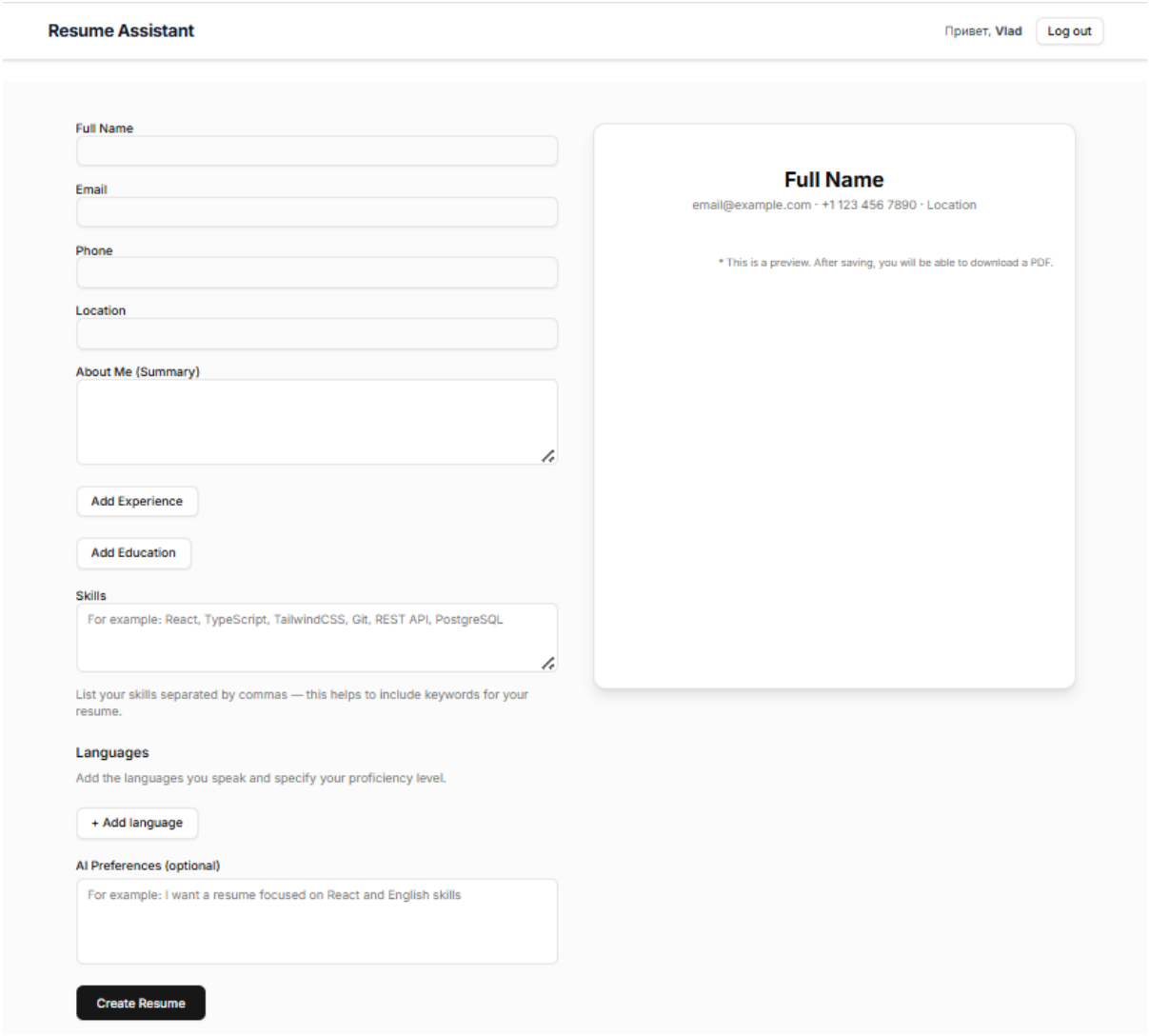


Figure 16. Interface of the form for filling in input data for generating a resume

The developed assistant also includes a feedback page with a demonstration of ratings and real experience of using the service Fig. 17.

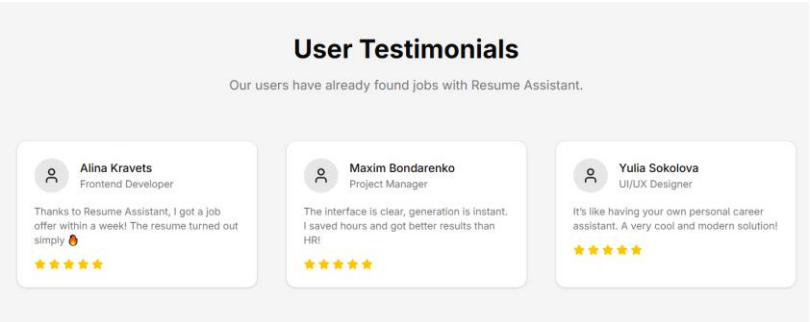


Figure 17. Testimonials Block Window

The final element of the home page serves as a final call to action. It reinforces the main goal of the application – to help the user stand out from other candidates by creating a professional resume in minutes (Fig. 18).

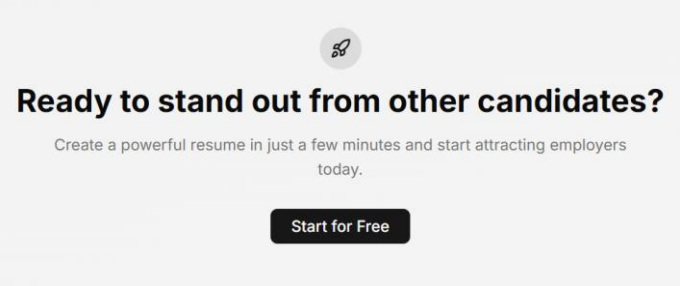


Figure 18. Assistant completion window

The task of generating summary content in the developed system is reduced to calculating the conditional probability of a sequence of symbols or tokens that form a logically completed text block based on the user's input query. From a formal point of view, the language model can be considered as a stochastic function:

$$G : P \rightarrow T, G(p) = \hat{t},$$

where P – set of valid incoming requests,

T – a set of potential resume texts,

$\hat{t} \in T$ – text that has maximum probability under the condition $p \in P$.

The goal of generation is to find the following sequence of tokens $t = (\omega_1, \omega_2, \ldots, \omega_n)$, which maximizes the conditional probability:

$$\hat{t} = \arg \max_{t \in T} P(t|p).$$

The estimation of P(t|p) is performed by a large-scale neural transformer model trained on general and professional text corpora. In our case, we use the OpenAI GPT-4 API, which performs predictive text generation using a pre-trained probability distribution function.

The final result is presented in the form of a text with a defined structure, which corresponds to the formats of modern resumes. The model not only generates a consistent text, but also performs logical-semantic addition based on key terms in the query (profession, skills, experience, etc.).

Thus, content generation in the system is formalized as a task of sequential language transformation, where the output depends on the internal assessment of the

context of the entered prompt and the language model with the function of conditional distribution.

Within the framework of the operation of the resume generation system, the processing of input and output documents plays a key role, which ensure full communication between the user, the client part and the server logic. These documents are the basis of all logical transformations that occur when working with information.

Input documents are formed based on the data provided by the user when filling out an interactive form. The user enters information related to personal data, professional skills, experience, education, languages and other parameters that characterize his qualifications. Such documents act as a source of input requests that are passed to the backend for further processing. The data is not limited to a rigid structure, but the system expects the information to conform to the general logical format adopted for building a resume.

Before being transferred to the server, the input information undergoes initial validation on the client side and is also checked by the server logic. This allows filtering out incorrect or empty requests even before the external or internal generation mechanisms are contacted. Thus, only relevant, complete and interpretable information by model mechanisms is allowed for processing.

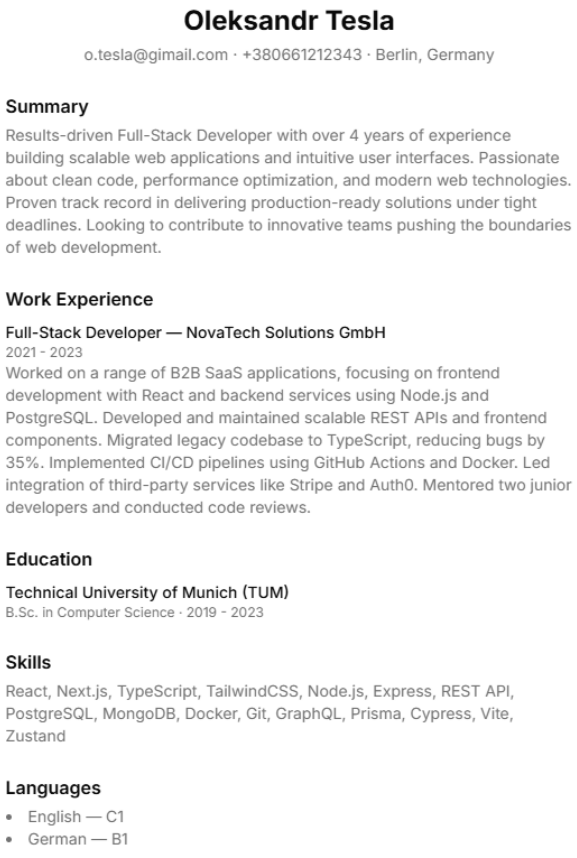
As a result of processing the input document, the system generates an output document, which is a structured final result. This document displays the generated summary - formed on the basis of the input information using natural language processing, formatting and structuring methods. The output document can be presented in various formats, but the main one is a document in PDF format, suitable for viewing, transferring or storing in external environments (Fig. 19).

The developed module has wide possibilities for practical application both within individual use and as an element of larger software complexes. Its flexible architecture, modern technology stack, and adaptability to various usage scenarios allow solutions to be integrated into an ecosystem of digital services related to the

labor market, educational platforms, recruiting systems, and other information portals.

From a technical point of view, the system can be implemented both on the company's local server and in the cloud. The use of Docker containerization allows you to quickly deploy the system in an isolated environment with fixed dependencies. At the same time, the use of providers such as Vercel, Heroku, AWS or Railway allows you to minimize infrastructure costs for the first stages of launch.

One of the main advantages of the module is the ability to scale in a corporate environment. In case of expanding the functionality, the system can be deployed as part of the company's internal portal, providing automated generation of resumes for employees, interns or candidates in the personnel reserve. In addition, it can be adapted for integration into human resource management systems (HRM/HRIS), which opens up prospects for automating the process of preparing supporting documents within the recruitment procedure.



Oleksandr Tesla

o.tesla@gmail.com · +380661212343 · Berlin, Germany

Summary

Results-driven Full-Stack Developer with over 4 years of experience building scalable web applications and intuitive user interfaces. Passionate about clean code, performance optimization, and modern web technologies. Proven track record in delivering production-ready solutions under tight deadlines. Looking to contribute to innovative teams pushing the boundaries of web development.

Work Experience

Full-Stack Developer — NovaTech Solutions GmbH
2021 - 2023
Worked on a range of B2B SaaS applications, focusing on frontend development with React and backend services using Node.js and PostgreSQL. Developed and maintained scalable REST APIs and frontend components. Migrated legacy codebase to TypeScript, reducing bugs by 35%. Implemented CI/CD pipelines using GitHub Actions and Docker. Led integration of third-party services like Stripe and Auth0. Mentored two junior developers and conducted code reviews.

Education

Technical University of Munich (TUM)
B.Sc. in Computer Science · 2019 - 2023

Skills

React, Next.js, TypeScript, TailwindCSS, Node.js, Express, REST API, PostgreSQL, MongoDB, Docker, Git, GraphQL, Prisma, Cypress, Vite, Zustand

Languages

- English — C1
- German — B1

Figure 19. Example of a resume for a software development specialist

Security, stability, performance. The system must operate over HTTPS, and all requests to third-party services must have timeout restrictions, retry attempts, and clear error handling (with a message returned to the user).

It is also important to provide scalability: as the number of users increases, the server must withstand at least 1000 simultaneous requests. For this, it is recommended to use an architecture with division into microservices or use load balancing in the future.

Stability requirements include ensuring:

response time < 500 ms under normal load;

no data loss in the event of a request failure;

the presence of re-generation mechanisms in the event of a third-party service failure.

Requirements for integration with the OpenAI API.

Using GPT-4 requires registering an API key with an appropriate tariff plan. Requests to the model have restrictions on the number of characters, the speed of access, and the cost of generation. Therefore, the module must contain:

response timeout control;

error handling type 429 (rate limit);

caching results for identical prompt requests (optional).

In addition, it is necessary to implement a mechanism for informing the user about the status of the request for resume generation. The user must have a clear idea of what stage the processing is at: the request has been sent, the response is expected, the generation is in progress, the result has been generated, or an error has occurred. In case of failures, for example, exceeding the request limit, API errors, lack of Internet connection. The system must provide a detailed and understandable message about the reason for the failure and, if possible, offer options for further actions, for example, repeating the request after a certain time or changing the wording of the prompt. The presence of such feedback increases the user's trust in the system, minimizes confusion and ensures predictability of the application's behavior even in cases where the external service is temporarily unavailable.

Thus, the structure of the developed system clearly defines the requirements for the client and server parts, database, network interaction and external integrations. Special attention is paid to the protection of personal data, error handling, logging of critical events, response time speed of less than 500 ms under normal load and scalability of the solution.

The developed module has significant practical potential. It can be used as an independent service, integrated into corporate HR systems, employment platforms, educational portals for student career development or be the basis for a SaaS product. Flexible architecture, modern technological stack and orientation to the real needs of users create the prerequisites for further development of the system, in particular adding cover letter generation, adapting resumes to specific vacancies and expanding analytical capabilities.

The service can also be used within educational institutions as part of a digital platform for student career development. In particular, it can enable students to create modern, structured resumes without the need for in-depth language or design skills. This is important in the context of shaping the competitiveness of graduates in the labor market.

Another area of potential implementation is the creation of a SaaS product with a subscription or tariff model. In this format, the system can offer basic resume generation for free, and access to advanced features (editing, style selection, exporting multiple options, multilingual support, etc.) is provided by subscription. This creates conditions for commercializing the solution, attracting investors, or launching a startup.

References

1. M. Bondarenko, S. Lushnei, Y. Paniv, O. Molchanovskyi, M. Romanyshyn, Y. Filipchuk, A. Kiulian Sovereign Large Language Models: Advantages, Strategy and Regulations [Electronic resource] - Access mode: https://www.researchgate.net/publication/389694178_Sovereign_Large_Language_Models_Advantages_Strategy_and_Regulations / DOI:10.48550/arXiv.2503.04745

2. V. Gorokhovatskyi, I. Tvoroshenko, O. Yakovleva, M. Hudáková and O. Gorokhovatskyi, "Application a Committee of Kohonen Neural Networks to Training of Image Classifier Based on Description of Descriptors Set," in IEEE Access, vol. 12, pp. 73376-73385, 2024, doi: 10.1109/ACCESS.2024.3404371

3. Переяславська, С., & Смагіна, О. (2025). Вивчення методів обробки природної мови для створення інтелектуальних систем, таких як чат-боти та системи автоматичного перекладу. Herald of Khmelnytskyi National University. Technical Sciences, 349(2), 100-108. https://doi.org/10.31891/2307-5732-2025-349-14

4. Методи обробки природної мови: Феномен ChatGPT [Текст]: навч. посіб. для здобувачів ступеня бакалавра за освітньою програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці» / автор: Ю. І. Стативка; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 621 кбайт). – Київ: КПІ ім. Ігоря Сікорського, 2023. – 67 с.

5. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin Attention Is All You Need [Electronic resource] - Access mode: https://arxiv.org/abs/1706.03762 https://doi.org/10.48550/arXiv.1706.03762

6. Wayne Xin Zhao, Kun Zhou, Junyi Li Tianyi Tang, Ji-Rong Wen Large Language Models [Electronic resource] - Access mode: https://content.e-bookshelf.de/media/reading/L-26457430-31ac3ab496.pdf ISBN 978-981-96-6259-3 (eBook) https://doi.org/10.1007/978-981-96-6259-3

7. Mina Lee, Percy Liang, Qian Yang CoAuthor: Designing a Human-AI Collaborative Writing Dataset for Exploring Language Model Capabilities / CHI '22: Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems Article No.: 388, Pages 1 – 19, https://doi.org/10.1145/3491102.3502030

8. Штучний інтелект і нейромережі: від теорії до практики на залізничному транспорті: Навч. посібник / В. О. Сотник, С. О. Змій, А. С. Панченко та ін. – Харків: УкрДУЗТ, 2025. – 324 с., рис. 54, табл. 11.

9. Сініцин І.П. , Рогушина Ю.В. , Юрченко К.Ю. Інтеграція великих мовних моделей із засобами семан-тичної обробки як інструмент цифровізації знань // Проблеми програмування, 2025, № 2, С.63-76 DOI: https://doi.org/10.15407/pp2025.02.063

10. Каверинський В. В., Літвін А. А., Палагін О. В. Зворотний синтез природномовних висловлювань на основі їх онтологічного представлення з використанням великої мовної моделі. Проблеми програмування. 2024. № 2-3. С. 359-366. https://doi.org/10.15407/pp2024.02-03.359

11. Q. Bai, S. Li, J. Yang, Q. Song, Z. Li, and X. Zhang, “Object detection recognition and robot grasping based on machine learning: A survey,” IEEE Access, vol. 8, pp. 181855–181879, 2020, doi: 10.1109/ACCESS.2020.3028740.

12. Terisa Roberts; Stephen J. Tonna, "Explaining Artificial Intelligence, Machine Learning, and Deep Learning Models," in Risk Modeling: Practical Applications of Artificial Intelligence, Machine Learning, and Deep Learning , Wiley, 2022, pp.55-70, doi: 10.1002/9781119824961.ch4.

13. Y. Cao, "Design and Implementation of an Intelligent Machine Learning System Based on Artificial Intelligence Computing," 2023 2nd International Conference on Data Analytics, Computing and Artificial Intelligence (ICDACAI), Zakopane, Poland, 2023, pp. 707-711, doi: 10.1109/ICDACAI59742.2023.00141.

14. Aakanksha et al. Assessing Vulnerabilities in Voice Assistants: Comparative Analysis of Google Assistant, Siri, and Alexa. AdvancesinKnowledge-BasedSystems,DataScience,andCybersecurity AdvancesinKnowledge-Based Systems, DataScience, andCybersecurity.Research 2025;2(1):10. Pages-197-214 https://cybersecurityjournal.info// | April 2025

15. I.P. Sinitsyn, J.V. Rogushina, K.Yu. Yurchenko Integration of large language models with semantic processing tools as an instrument for knowledge digitization / Problems in programming 2025; №2: P. 63-76 https://doi.org/10.15407/pp2025.02.063

16. Y. Sun, Z. Li, X. Li, and J. Zhang, “Classifier selection and ensemble model for multi-class imbalance learning in education grants prediction,” Appl. Artif. Intell., vol. 35, no. 4, pp. 290–303, Feb. 2021, doi: 10.1080/08839514.2021.1877481.

17. C. C. Aggarwal, “Machine learning with shallow neural networks,” in Neural Networks and Deep Learning. Cham, Switzerland : Springer, 2023, pp. 73–117.

18. S. Zhang, J. Wang, X. Tao, Y. Gong, and N. Zheng, “Constructing deep sparse coding network for image classification,” *Pattern Recognit.*, vol. 64, pp. 130–140, Apr. 2017, doi: 10.1016/j.patcog.2016.10.032.

19. Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., Diakopoulos, N. Designing the User Interface: Strategies for Effective Human–Computer Interaction. 6th ed. Pearson, 2016. – 616 с. – ISBN 978-0134380384.

20. Fowler, M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Addison-Wesley, 2003. – 208 с. – ISBN 0321193687.

21. NestJS Documentation. [Електронний ресурс]. – Режим доступу: https://docs.nestjs.com – Дата звернення: 09.05.2025.

22. Prisma ORM Docs. [Електронний ресурс]. – Режим доступу: https://www.prisma.io/docs – Дата звернення: 10.05.2025.

23. PostgreSQL Documentation. [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs – Дата звернення: 09.05.2025.

24. OpenAI API Documentation. [Електронний ресурс]. – Режим доступу: https://platform.openai.com/docs – Дата звернення: 09.05.2025.

25. Next.js Documentation. [Електронний ресурс]. – Режим доступу: https://nextjs.org/docs – Дата звернення: 09.05.2025.

26. Tailwind CSS Documentation. [Електронний ресурс]. – Режим доступу: https://tailwindcss.com/docs – Дата звернення: 10.05.2025.

27. React Query (TanStack) Docs. [Електронний ресурс]. – Режим доступу: https://tanstack.com/query – Дата звернення: 10.05.2025.

28. Redux Toolkit Docs. [Електронний ресурс]. – Режим доступу: https://redux-toolkit.js.org – Дата звернення: 09.05.2025.

29. React Hook Form Docs. [Електронний ресурс]. – Режим доступу: https://react-hook-form.com – Дата звернення: 09.05.2025.

30. Zod Validation Library. [Електронний ресурс]. – Режим доступу: https://zod.dev – Дата звернення: 10.05.2025.

31. Axios Docs. [Електронний ресурс]. – Режим доступу: https://axios-http.com/docs/intro – Дата звернення: 10.05.2025.

32. JWT.IO – JSON Web Tokens. [Електронний ресурс]. – Режим доступу: https://jwt.io/introduction – Дата звернення: 09.05.2025.

33. Passport.js Documentation. [Електронний ресурс]. – Режим доступу: http://www.passportjs.org/docs – Дата звернення: 10.05.2025.

34. Microsoft Docs: REST API Design. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design – Дата звернення: 08.05.2025.

35. PDFKit Documentation. [Електронний ресурс]. – Режим доступу: https://pdfkit.org/docs – Дата звернення: 09.05.2025.

36. Shadcn UI Documentation. [Електронний ресурс]. – Режим доступу: https://ui.shadcn.com/docs – Дата звернення: 10.05.2025.

References

1. M. Bondarenko, S. Lushnei, Y. Paniv, O. Molchanovskyi, M. Romanyshyn, Y. Filipchuk, A. Kiulian Sovereign Large Language Models: Advantages, Strategy and Regulations [Electronic resource] - Access mode: https://www.researchgate.net/publication/389694178_Sovereign_Large_Language_Models_Advantages_Strategy_and_Regulations / DOI:10.48550/arXiv.2503.04745

2. V. Gorokhovatskyi, I. Tvoroshenko, O. Yakovleva, M. Hudáková and O. Gorokhovatskyi, "Application a Committee of Kohonen Neural Networks to Training of Image Classifier Based on Description of Descriptors Set," in IEEE Access, vol. 12, pp. 73376-73385, 2024, doi: 10.1109/ACCESS.2024.3404371

3. Pereyaslavska, S., & Smagina, O. (2025). Study of natural language processing methods for creating intelligent systems, such as chatbots and automatic

translation systems. Herald of Khmelnytskyi National University. Technical Sciences, 349(2), 100-108. https://doi.org/10.31891/2307-5732-2025-349-14

4. Natural language processing methods: ChatGPT phenomenon [Text]: a study guide for bachelor's degree applicants in the educational program "Software engineering of intelligent cyber-physical systems in energy" / author: Yu. I. Statyvka; Igor Sikorsky Kyiv Polytechnic Institute. – Electronic text data (1 file: 621 kbytes). – Kyiv: Igor Sikorsky Kyiv Polytechnic Institute, 2023. – 67 p.

5. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin Attention Is All You Need [Electronic resource] - Access mode: https://arxiv.org/abs/1706.03762 https://doi.org/10.48550/arXiv.1706.03762

6. Wayne Xin Zhao, Kun Zhou, Junyi Li Tianyi Tang, Ji-Rong Wen Large Language Models [Electronic resource] - Access mode: https://content.e-bookshelf.de/media/reading/L-26457430-31ac3ab496.pdf ISBN 978-981-96-6259-3 (eBook) https://doi.org/10.1007/978-981-96-6259-3

7. Mina Lee, Percy Liang, Qian Yang CoAuthor: Designing a Human-AI Collaborative Writing Dataset for Exploring Language Model Capabilities / CHI '22: Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems Article No.: 388, Pages 1 – 19, https://doi.org/10.1145/3491102.3502030

8. Artificial Intelligence and Neural Networks: From Theory to Practice in Railway Transport: Textbook / V. O. Sotnyk, S. O. Zmiy, A. S. Panchenko et al. – Kharkiv: UkrDUZT, 2025. – 324 p., fig. 54, tab. 11.

9. Sinitsyn I.P. , Rogushyna Yu.V. , Yurchenko K.Yu. Integration of large language models with semantic processing tools as a tool for knowledge digitization // Programming Problems, 2025, No. 2, P.63-76 DOI: https://doi.org/10.15407/pp2025.02.063

10. Kaverinsky V. V., Litvin A. A., Palagin O. V. Reverse synthesis of natural language utterances based on their ontological representation using a large language model. Programming Problems. 2024. No. 2-3. P. 359-366. https://doi.org/10.15407/pp2024.02-03.359

11. Q. Bai, S. Li, J. Yang, Q. Song, Z. Li, and X. Zhang, “Object detection recognition and robot grasping based on machine learning: A survey,” IEEE Access, vol. 8, pp. 181855–181879, 2020, doi: 10.1109/ACCESS.2020.3028740.

12. Terisa Roberts; Stephen J. Tonna, "Explaining Artificial Intelligence, Machine Learning, and Deep Learning Models," in Risk Modeling: Practical Applications of Artificial Intelligence, Machine Learning, and Deep Learning, Wiley, 2022, pp.55-70, doi: 10.1002/9781119824961.ch4.

13. Y. Cao, "Design and Implementation of an Intelligent Machine Learning System Based on Artificial Intelligence Computing," 2023 2nd International Conference on Data Analytics, Computing and Artificial Intelligence (ICDACAI), Zakopane, Poland, 2023, pp. 707-711, doi: 10.1109/ICDACAI59742.2023.00141.

14. Aakanksha et al. Assessing Vulnerabilities in Voice Assistants: Comparative Analysis of Google Assistant, Siri, and Alexa. AdvancesinKnowledge-BasedSystems,DataScience,andCybersecurity AdvancesinKnowledge-BasedSystems,DataScience,andCybersecurity.Research 2025;2(1):10. Pages-197-214 https://cybersecurityjournal.info// | April 2025

15. I.P. Sinitsyn, J.V. Rogushina, K. Yu. Yurchenko Integration of large language models with semantic processing tools as an instrument for knowledge digitization / Problems in programming 2025; No. 2: P. 63-76 https://doi.org/10.15407/pp2025.02.063

16. Y. Sun, Z. Li, X. Li, and J. Zhang, “Classifier selection and ensemble model for multi-class imbalance learning in education grants prediction,” Appl. Artif. Intel., vol. 35, no. 4, pp. 290–303, Feb. 2021, doi: 10.1080/08839514.2021.1877481.

17. C. C. Aggarwal, “Machine learning with shallow neural networks,” in Neural Networks and Deep Learning. Cham, Switzerland : Springer, 2023, pp. 73–117.

18. S. Zhang, J. Wang, X. Tao, Y. Gong, and N. Zheng, “Constructing deep sparse coding network for image classification,” Pattern Recognit., vol. 64, pp. 130–140, Apr. 2017, doi: 10.1016/j.patcog.2016.10.032.

19. Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., Diakopoulos, N. Designing the User Interface: Strategies for Effective Human–Computer Interaction. 6th ed. Pearson, 2016. – 616 p. – ISBN 978-0134380384.

20. Fowler, M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Addison-Wesley, 2003. – 208 p. – ISBN 0321193687.

21. NestJS Documentation. [Electronic resource]. – Access mode: https://docs.nestjs.com – Date of access: 09.05.2025.

22. Prisma ORM Docs. [Electronic resource]. – Access mode: https://www.prisma.io/docs – Access date: 10.05.2025.

23. PostgreSQL Documentation. [Electronic resource]. – Access mode: https://www.postgresql.org/docs – Access date: 09.05.2025.

24. OpenAI API Documentation. [Electronic resource]. – Access mode: https://platform.openai.com/docs – Access date: 09.05.2025.

25. Next.js Documentation. [Electronic resource]. – Access mode: https://nextjs.org/docs – Access date: 09.05.2025.

26. Tailwind CSS Documentation. [Electronic resource]. – Access mode: https://tailwindcss.com/docs – Access date: 10.05.2025.

27. React Query (TanStack) Docs. [Electronic resource]. – Access mode: https://tanstack.com/query – Access date: 10.05.2025.

28. Redux Toolkit Docs. [Electronic resource]. – Access mode: https://redux-toolkit.js.org – Access date: 09.05.2025.

29. React Hook Form Docs. [Electronic resource]. – Access mode: https://react-hook-form.com – Access date: 09.05.2025.

30. Zod Validation Library. [Electronic resource]. – Access mode: https://zod.dev – Access date: 10.05.2025.

31. Axios Docs. [Electronic resource]. – Access mode: https://axios-http.com/docs/intro – Access date: 10.05.2025.

32. JWT.IO – JSON Web Tokens. [Electronic resource]. – Access mode: https://jwt.io/introduction – Access date: 09.05.2025.

33. Passport.js Documentation. [Electronic resource]. – Access mode: http://www.passportjs.org/docs – Access date: 10.05.2025.

34. Microsoft Docs: REST API Design. [Electronic resource]. – Access mode: https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design – Access date: 08.05.2025.

35. PDFKit Documentation. [Electronic resource]. – Access mode: https://pdfkit.org/docs – Access date: 09.05.2025.

36. Shadcn UI Documentation. [Electronic resource]. – Access mode: https://ui.shadcn.com/docs – Access date: 10.05.2025.

Анотація. У статті розглянуто процес розроблення інтелектуальної системи автоматизованого створення резюме на основі генеративного штучного інтелекту (ГШІ) та сучасних вебтехнологій. Актуальність дослідження зумовлена зростаючою потребою у швидкому та якісному формуванні професійних резюме, адаптованих до вимог конкретних вакансій і систем автоматичного відбору кандидатів.

Метою роботи є створення програмного модуля, який дає змогу користувачеві сформувати структуроване резюме на основі короткого текстового опису досвіду, навичок і професійних досягнень.

У роботі проведено аналіз існуючих сервісів для створення резюме, визначено їхні переваги та недоліки, сформульовано функціональні та нефункціональні вимоги до системи. Для реалізації програмного продукту використано мову програмування TypeScript, фреймворки NestJS і Next.js, систему керування базами даних PostgreSQL та ORM Prisma. Генерація текстового вмісту здійснюється за допомогою API OpenAI, а формування документів у форматі PDF із використанням PDFKit.

Запропонована система забезпечує персоналізацію змісту, багатомовну підтримку, збереження історії документів та експорт готового резюме.

Результати тестування підтвердили стабільну роботу модуля, високу якість згенерованого тексту та суттєве скорочення часу підготовки документів.

Розроблене рішення може бути використане як самостійний вебсервіс або інтегроване в HR-платформи та системи електронного працевлаштування.

Ключові слова: інтелектуальна система, генеративний штучний інтелект, великі мовні моделі, автоматизоване створення резюме, вебзастосунок, TypeScript, NestJS, Next.js, PostgreSQL, Prisma, OpenAI API, ATS-оптимізація, PDF-генерація.

Abstract. The article examines the development process of an intelligent system for automated resume generation based on generative artificial intelligence and modern web technologies. The study's relevance is driven by the growing need for fast, high-quality creation of professional resumes tailored to specific job requirements and to applicant tracking systems (ATS).

The aim of this work is to develop a software module that enables users to generate a structured resume from a brief textual description of their experience, skills, and professional achievements. The study analyzes existing resume-building services, identifies their advantages and disadvantages, and formulates both functional and non-functional system requirements.

The software product is implemented using TypeScript, the NestJS and Next.js frameworks, PostgreSQL, and Prisma ORM. Text generation is performed using the OpenAI API, while document generation in PDF format is implemented using PDFKit.

The proposed system provides content personalization, multilingual support, document history storage, and resume export. Testing results confirm the module's stable performance, high-quality generated text, and a significant reduction in document preparation time. The developed solution can be used as a standalone web service or integrated into HR platforms and electronic employment systems.

Keywords: intelligent system, generative artificial intelligence, large language models, automated resume generation, web application, TypeScript, NestJS, Next.js, PostgreSQL, Prisma, OpenAI API, ATS optimization, PDF generation.

Відомості про авторів

Вільхівська Ольга Володимирівна - кандидат економічних наук, доцент, доцент кафедри інформатики та комп’ютерної техніки, Харківський національний економічний університет імені Семена Кузнеця

Контактна інформація: 096-542-99-07, olha.vilkhivska@hneu.net

Тесля Олександр - магістр з інформаційних систем та технологій.

Контактна інформація: oleksandr.teslia@hneu.net, +380962735184

Офіційне написання англійською мовою:

Vilkhivska Olga V.

Teslia O. O.

Контактна особа - Вільхівська О. В.

Наданий матеріал раніше не публікувався та в інші видання не надсилався.