Used

Technical specification NekoShop

Development of online shop in Telegram messenger using Python

4-proxy
21.08.2024

Table of Contents

Table of contents

1. Specifications of terminology and design solutions

1.1. Special provision on terms:

- Project – A collection of program code, documentation, databases, images and other files included in the development process;
- System – A system refers to the set of program code, embedded algorithms and operational scripts that ensure the functionality and performance of a software;
- Application – The final development object, a program that is a Telegram shop;
- Administrator – A person who influences the structure and operation of an application or who is involved in the development of a project;
- Seller – A person who has *CRUD*[1] transaction rights for their own product in the application catalog;
- User – The regular client of the application, i.e. the Telegram user (buyer);

1.2. Special provision for design solutions:

1. *Semantic versioning*[2] will be used to specify the version of the project and individual packages/modules (For more details see "Appendices" section, item: 2. Semantic versioning);

2. When working with project documentation in Microsoft Word, the document "Documentation Standard - Recommendations for Formatting Project Documentation in Microsoft Word" will be applied. (For more details see "Appendices" section, item: 1. Documentation standard);

[1] **CRUD** – create (C), read (R), update (U), delete (D).
[2] **Semantic Versioning** – A specification on how to assign versions to software releases.

3. The structure of this Technical Specification for the “NekoShop” project is based on recommendations from “GOST 34.602-2020” and “GOST 19.201-78”. (For more details see “Appendices” section, items: 3. GOST 34.602-2020, 4. GOST 19.201-78);

4. The document “Documentation Standard” shall be given priority in the design of this Technical Specification. (For more details see “Appendices” section, item: 1. Documentation standard).

That is, in case of conflicts with other documents regarding the design of content elements, the “Documentation Standard” is preferred;

1.3. General provision for the technical specification of the project:

As the “NekoShop” project is a personal and non-commercial development, the Technical Specification will be developed with the objective of overall system architecture design, creating a design and establishing clear intentions for the developer/programmer.

This means that the content of the Technical Specification will be limited to certain frameworks that are not redundant and are relevant to the development area.

The structure and content of these Technical Specification may also be subject to change during the project development process;

2. General information

2.1. The full name of the project and designation:

Project name: “Development of online shop in Telegram messenger using Python”;

Designation: “NekoShop”;

2.2. Information about of the developer, if exists:

Developer: 4proxy (4-proxy);

Ways of communicating:

- https://github.com/4-proxy (GitHub);
- 4proxy.cleanhouse@proton.me (Public Email);

2.3. Regulatory and technical documents regulating the creation of the application:

External RTDs:

- GOST 34.602-2020;
- GOST 19.201-78;

Internal RTDs:

- Documentation standard;
- Semantic versioning;

3. Project goals and purpose

3.1. Project creation goals:

Creating a user-friendly interface: developing a user-friendly and intuitive interface for users to ensure comfortable interaction with the application;

Ensuring payment security: guaranteeing payment security that will allow users to shop online with confidence in the safety of their financial data;

Simple control of the application: providing simple application management to allow for prompt response to errors, make improvements, and maintain quality of operation;

Availability of catalog extension: providing a user-friendly interface for sellers to provide CRUD transaction, which will give easy control over up-to-date data of their own products;

3.2. Purpose of the project:

Development and creation of the application, which is an online shop selling virtual products (for example, paid content in online games or paid subscriptions to services) in the messenger Telegram.

The application is planned to have a mascot in the role of *Neko*[3], to provide a unique feature. As a consequence, such a solution will require certain functionality to display and highlight this feature when the user interacts with the application interface;

[3] **Neko** (*Catgirl*) – is a girl who has cat ears, a tail, or other features of the cat family, but otherwise has a fully human body. A fictional character often seen in anime and manga, as well as in various comic books and video games.

4. Characterization of the automation object

4.1. General information about the automation object:

Title: Shop into Telegram messenger;

Type of object: Online shop;

Subject of activity: Selling products and services via Telegram messenger;

Main functions: Providing information about products, taking orders, processing payments, organizing delivery, feedback and managing user base;

4.2. Conditions of exploitation of the automation object:

Hardware requirements: Internet connection/access;

Mode of operation: Online, round the clock;

User support: Feedback, real-time order processing, simple and friendly user interface;

Security: Protection of user's data and secure payment systems;

4.3. Environmental Characteristics:

Target audience: Telegram users interested in purchasing products or services provided in the application catalog;

Development Trends: Opportunities to introduce new technologies/features to improve user experience and system efficiency, creating an ecosystem that includes an online shop branch in the implementation of the website;

5. Project requirements

5.1. Requirements to database structure components:

5.1.1. Requirements to the structure of database tables:

Table 5.1.

Table of fields of the table “Category”

Name	Value type	Flags
id	INT	PK; NN; AI;
name	VARCHAR(60)	NN; UQ;
description	VARCHAR(160)	NN;
is_blocked	TINYINT(1); DEFAULT 0	NN;

Table 5.2.

Table of fields of the table “Service”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;
category_id (FK)	INT	NN;
name	VARCHAR(60)	NN; UQ;
description	VARCHAR(255)	NN;
image	BLOB	NN;
is_blocked	TINYINT(1); DEFAULT 0	NN;

Table 5.3.

Table of fields of the table “Product”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;
owner_id (FK)	INT	NN;
service_id (FK)	INT	NN;
title	VARCHAR(60)	NN;
description	VARCHAR(255)	NN;
image	BLOB	NN;
price	DECIMAL(6,2)	NN;

End of Table 5.3.

is_blocked	TINYINT(1); DEFAULT 0	NN;
is_infinite	TINYINT(1); DEFAULT 0	NN;
quantity	INT; DEFAULT 0	NN;

Table 5.4.

Table of fields of the table “Order”

Name	**Value type**	**Flags**
id (PK)	INT	PK; NN; AI;
product_id (FK)	INT	NN;
customer_id (FK)	INT	NN;
status	ENUM('open', 'pending', 'canceled', 'completed'); DEFAULT 'open'	NN;
total_price	DECIMAL(6,2)	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
updated_at	DATETIME; DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP	
closed_at	DATETIME	

Table 5.5.

Table of fields of the table “User”

Name	**Value type**	**Flags**
id (PK)	INT	PK; NN; AI;
phone_number	VARCHAR(20)	NN; UQ;
email	VARCHAR(255)	UQ;
is_email_notification	TINYINT(1); DEFAULT 0	NN;
is_banned	TINYINT(1); DEFAULT 0	NN;
is_admin	TINYINT(1); DEFAULT 0	NN;
is_seller	TINYINT(1); DEFAULT 0	NN;

End of Table 5.5.

interface_language_code (FK)	VARCHAR(3); DEFAULT 'eng'	NN;

Table 5.6.

Table of fields of the table “SupportedInterfaceLanguage”

Name	**Value type**	**Flags**
code (PK)	VARCHAR(3)	PK; NN;
name	VARCHAR(60)	NN;

Table 5.7.

Table of fields of the table “Ticket”

Name	**Value type**	**Flags**
id (PK)	INT	PK; NN; AI;
creator_id (FK)	INT	NN;
responsible_admin_id (FK)	INT	
creator_type	ENUM('user', 'seller'); DEFAULT 'user'	NN;
status	ENUM('open', 'pending', 'resolved', 'canceled'); DEFAULT 'open'	NN;
subject	VARCHAR(100)	NN;
description	TEXT	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
updated_at	DATETIME; DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP	
resolved_at	DATETIME	
canceled_reason	TEXT	

Table 5.8.

Table of fields of the table “TicketSolution”

Name	Value type	Flags
ticket_id (PK)(FK)	INT	PK; NN;
solution	TEXT	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
updated_at	DATETIME; DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP	

Table 5.9.

Table of fields of the table “RequestBlockProduct”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;
product_id (FK)	INT	NN;
responsible_admin_id (FK)	INT	
status	ENUM('open', 'pending', 'closed', 'canceled'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
updated_at	DATETIME; DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP	
closed_at	DATETIME	
canceled_reason	TEXT	

Table 5.10.

Table of fields of the table “RequestUnblockProduct”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;
product_id (FK)	INT	NN;
responsible_admin_id (FK)	INT	
status	ENUM('open', 'pending', 'closed', 'canceled'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
updated_at	DATETIME; DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP	
closed_at	DATETIME	
canceled_reason	TEXT	

Table 5.11.

Table of fields of the table “RequestDeleteProduct”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;
product_id (FK)	INT	NN;
responsible_admin_id (FK)	INT	
status	ENUM('open', 'pending', 'closed', 'canceled'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
updated_at	DATETIME; DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP	
closed_at	DATETIME	

End of Table 5.11.

canceled_reason	TEXT	

Table 5.12.

Table of fields of the table “RequestEditProduct”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;
product_id (FK)	INT	NN;
responsible_admin_id (FK)	INT	
status	ENUM('open', 'pending', 'closed', 'canceled'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
updated_at	DATETIME; DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP	
closed_at	DATETIME	
canceled_reason	TEXT	

Table 5.13.

Table of fields of the table “EditProductData”

Name	Value type	Flags
request_id (PK)(FK)	INT	PK; NN;
new_title	VARCHAR(60)	
new_description	VARCHAR(255)	
new_image	BLOB	
new_price	DECIMAL(6,2)	
is_infinite	TINYINT(1)	
quantity	INT	

Table 5.14.

Table of fields of the table “RequestAddProduct”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;
service_id (FK)	INT	NN;
owner_id (FK)	INT	NN;
responsible_admin_id (FK)	INT	
status	ENUM('open', 'pending', 'closed', 'canceled'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
updated_at	DATETIME; DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP	
closed_at	DATETIME	
canceled_reason	TEXT	

Table 5.15.

Table of fields of the table “AddProductData”

Name	Value type	Flags
request_id (PK)(FK)	INT	PK; NN;
title	VARCHAR(60)	NN;
description	VARCHAR(255)	NN;
image	BLOB	NN;
price	DECIMAL(6,2)	NN;
is_infinite	TINYINT(1); DEFAULT 0	NN;
quantity	INT; DEFAULT 0	NN;

Table 5.16.

Table of fields of the table “AdminPrivilege”

Name	Value type	Flags
admin_id (PK)(FK)	INT	PK; NN;
is_technical_support	TINYINT(1); DEFAULT 0	NN;
is_request_management	TINYINT(1); DEFAULT 0	NN;
is_category_management	TINYINT(1); DEFAULT 0	NN;
is_service_management	TINYINT(1); DEFAULT 0	NN;
is_product_management	TINYINT(1); DEFAULT 0	NN;
is_delete_product	TINYINT(1); DEFAULT 0	NN;
is_seller_management	TINYINT(1); DEFAULT 0	NN;
is_admin_management	TINYINT(1); DEFAULT 0	NN;
is_user_ban	TINYINT(1); DEFAULT 0	NN;
is_user_unban	TINYINT(1); DEFAULT 0	NN;

Table 5.17.

Table of fields of the table “AdminRequestAddAdmin”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;
responsible_admin_id (FK)	INT	NN;
added_admin_user_id (FK)	INT	NN;
status	ENUM('open', 'pending', 'closed'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
closed_at	DATETIME	

Table 5.18.

Table of fields of the table “AdminRequestAddSeller”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;
responsible_admin_id (FK)	INT	NN;
added_seller_user_id (FK)	INT	NN;

End of Table 5.18.

status	ENUM('open', 'pending', 'closed'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
closed_at	DATETIME	

Table 5.19.

Table of fields of the table “AdminRequestKickAdmin”

Name	**Value type**	**Flags**
id (PK)	INT	PK; NN; AI;
responsible_admin_id (FK)	INT	NN;
kickable_admin_user_id (FK)	INT	NN;
status	ENUM('open', 'pending', 'closed'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
closed_at	DATETIME	

Table 5.20.

Table of fields of the table “AdminRequestKickSeller”

Name	**Value type**	**Flags**
id (PK)	INT	PK; NN; AI;
responsible_admin_id (FK)	INT	NN;
kickable_seller_user_id (FK)	INT	NN;
status	ENUM('open', 'pending', 'closed'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
closed_at	DATETIME	

Table 5.21.

Table of fields of the table “AdminRequestBanUser”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;
responsible_admin_id (FK)	INT	NN;
banned_user_id (FK)	INT	NN;
status	ENUM('open', 'pending', 'closed'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
closed_at	DATETIME	

Table 5.22.

Table of fields of the table “AdminRequestUnbanUser”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;
responsible_admin_id (FK)	INT	NN;
unbanned_user_id (FK)	INT	NN;
status	ENUM('open', 'pending', 'closed'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
closed_at	DATETIME	

Table 5.23.

Table of fields of the table “AdminRequestBlockCategory”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;
category_id (FK)	INT	NN;
responsible_admin_id (FK)	INT	NN;
status	ENUM('open', 'pending', 'closed'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;

End of Table 5.23.

closed_at	DATETIME	

Table 5.24.

Table of fields of the table “AdminRequestUnblockCategory”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;
category_id (FK)	INT	NN;
responsible_admin_id (FK)	INT	NN;
status	ENUM('open', 'pending', 'closed'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
closed_at	DATETIME	

Table 5.25.

Table of fields of the table “AdminRequestEditCategory”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;
category_id (FK)	INT	NN;
responsible_admin_id (FK)	INT	NN;
status	ENUM('open', 'pending', 'closed'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
closed_at	DATETIME	

Table 5.26.

Table of fields of the table “AdminEditCategoryData”

Name	Value type	Flags
request_id (PK)(FK)	INT	PK; NN;
new_name	VARCHAR(60)	
new_description	VARCHAR(160)	

Table 5.27.

Table of fields of the table “AdminRequestAddCategory”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;
responsible_admin_id (FK)	INT	NN;
status	ENUM('open', 'pending', 'closed'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
closed_at	DATETIME	

Table 5.28.

Table of fields of the table “AdminAddCategoryData”

Name	Value type	Flags
request_id (PK)(FK)	INT	PK; NN;
new_name	VARCHAR(60)	NN;
new_description	VARCHAR(160)	NN;

Table 5.29.

Table of fields of the table “AdminRequestBlockService”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;
service_id (FK)	INT	NN;
responsible_admin_id (FK)	INT	NN;
status	ENUM('open', 'pending', 'closed'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
closed_at	DATETIME	

Table 5.30.

Table of fields of the table “AdminRequestUnblockService”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;

End of Table 5.30.

service_id (FK)	INT	NN;
responsible_admin_id (FK)	INT	NN;
status	ENUM('open', 'pending', 'closed'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
closed_at	DATETIME	

Table 5.31.

Table of fields of the table “AdminRequestEditService”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;
service_id (FK)	INT	NN;
responsible_admin_id (FK)	INT	NN;
status	ENUM('open', 'pending', 'closed'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
closed_at	DATETIME	

Table 5.32.

Table of fields of the table “AdminEditServiceData”

Name	Value type	Flags
request_id (PK)(FK)	INT	PK; NN;
new_name	VARCHAR(60)	
new_description	VARCHAR(160)	
new_image	BLOB	

Table 5.33.

Table of fields of the table “AdminRequestAddService”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;
category_id (FK)	INT	NN;

End of Table 5.33.

responsible_admin_id (FK)	INT	NN;
status	ENUM('open', 'pending', 'closed'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
closed_at	DATETIME	

Table 5.34.

Table of fields of the table “AdminAddServiceData”

Name	Value type	Flags
request_id (PK)(FK)	INT	PK; NN;
new_name	VARCHAR(60)	NN;
new_description	VARCHAR(160)	NN;
new_image	BLOB	NN;

Table 5.35.

Table of fields of the table “AdminRequestDeleteProduct”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;
product_id (FK)	INT	NN;
responsible_admin_id (FK)	INT	NN;
status	ENUM('open', 'pending', 'closed'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
closed_at	DATETIME	

Table 5.36.

Table of fields of the table “AdminRequestBlockProduct”

Name	Value type	Flags
id (PK)	INT	PK; NN; AI;
product_id (FK)	INT	NN;
responsible_admin_id (FK)	INT	NN;

End of Table 5.36.

status	ENUM('open', 'pending', 'closed'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
closed_at	DATETIME	

Table 5.37.

Table of fields of the table “AdminRequestUnblockProduct”

Name	**Value type**	**Flags**
id (PK)	INT	PK; NN; AI;
product_id (FK)	INT	NN;
responsible_admin_id (FK)	INT	NN;
status	ENUM('open', 'pending', 'closed'); DEFAULT 'open'	NN;
created_at	DATETIME; DEFAULT CURRENT_TIMESTAMP	NN;
closed_at	DATETIME	

5.1.2. Requirements for the purpose of database tables:

Table 5.38.

Table description to assignment of database tables

Name of table	**Assignment Description**
Category	The “Category” table is used to manage actual categories of application services. For example: “Game”, “Film”, “Music”
Service	The “Service” table is used to manage the actual services of a application category. For example, the “Gift Card”: “Amazon”, “Xbox”, “Microsoft”, “Apple”.

Continuation of Table 5.38.

Product	The “Product” table is used to manage the actual products of a application service. For example, the service “Steam wallet”: “Refill your wallet for 5$”, “Refill your wallet for 10$”, “Refill your wallet for 25$”
User	The “User” table is used to manage all application accounts. Not dividing the account privileges into separate tables. Privileges mean the status of the account, for example: “Admin”, “Seller”.
Order	The “Order” table is used to manage product orders in any state. Order status means its status: pending, canceled, returned, completed.
SupportedInterfaceLanguage	The “SupportedInterfaceLanguage” table is used to manage the actual language localizations of the application interface.
Ticket	The “Ticket” table is used to manage application technical support tickets (requests).
TicketSolution	The “TicketSolution” table is used to manage solutions (responses) to tickets sent to technical support.
RequestBlockProduct	The “RequestBlockProduct” table is used to manage requests to block a product in the application catalog.
RequestUnblockProduct	The “RequestUnblockProduct” table is used to manage product unblock requests in the application catalog.
RequestDeleteProduct	The “RequestDeleteProduct” table is used to manage requests to delete a product in the application catalog.

Continuation of Table 5.38.

RequestEditProduct	The “RequestEditProduct” table is used to manage requests to edit product data in the application catalog.
EditProductData	The “EditProductData” table is used to store request data for editing product data in the application catalog.
RequestAddProduct	The “RequestAddProduct” table is used to manage requests to add a new product in the application catalog.
AddProductData	The “AddProductData” table is used to store request data for adding a new product to the application catalog.
AdminPrivilege	The “AdminPrivilege” table is used to describe the available privileges/permissions for each individual admin.
AdminRequestBanUser	The “AdminRequestBanUser” table is used to store requests from administrators to block users.
AdminRequestUnbanUser	The “AdminRequestUnbanUser” table is used to store requests from administrators to unblock users.
AdminRequestAddAdmin	The “AdminRequestAddAdmin” table is used to store requests from administrators to upgrade the specified account to administrator.
AdminRequestKickAdmin	The “AdminRequestKickAdmin” table is used to store requests from administrators to downgrade another administrator account to a regular user.
AdminRequestAddSeller	The “AdminRequestAddSeller” table is used to store requests from administrators to upgrade a specified account to a seller.

Continuation of Table 5.38.

AdminRequestKickSeller	The “AdminRequestKickSeller” table is used to store requests from administrators to downgrade a seller account to a regular user.
AdminRequestBlockCategory	The “AdminRequestBlockCategory” table is used to store requests from administrators to block a category.
AdminRequestUnblockCategory	The “AdminRequestUnblockCategory” table is used to store requests from administrators to unblock a category.
AdminRequestEditCategory	The “AdminRequestEditCategory” table is used to store requests from administrators to edit a category.
AdminEditCategoryData	The “AdminEditCategoryData” table is used to store category edit request data from the administrator.
AdminRequestAddCategory	The “AdminRequestAddCategory” table is used to store requests from administrators to add a new category.
AdminAddCategoryData	The “AdminAddCategoryData” table is used to store the request data from the administrator to add a new category.
AdminRequestBlockService	The “AdminRequestBlockService” table is used to store requests from administrators to block an application category service.
AdminRequestUnblockService	The “AdminRequestUnblockService” table is used to store requests from administrators to unblock an application category service.
AdminRequestEditService	The “AdminRequestEditService” table is used to store requests from administrators to edit an application category service.

End of Table 5.38.

AdminEditServiceData	The “AdminEditServiceData” table is used to store the data of a request from the administrator to change an application category service.
AdminRequestAddService	The “AdminRequestAddService” table is used to store requests from administrators to add a new service to an application category.
AdminAddServiceData	The “AdminAddServiceData” table is used to store request data from the administrator to add a new service to an application category.
AdminRequestBlockProduct	The “AdminRequestBlockProduct” table is used to store requests from administrators to block a product of a specific application service.
AdminRequestUnblockProduct	The “AdminRequestUnblockProduct” table is used to store requests from administrators to unblock a product of a specific application service.
AdminRequestDeleteProduct	The “AdminRequestDeleteProduct” table is used to store requests from administrators to delete a product from a specific application service.

5.1.3. Visual representation of the database model:

The visual representation as well as a finished model for viewing and identifying relationships between tables can be found in the project repository at the link: https://github.com/4-proxy/NekoShop.

A more precise location, defined by: /templates/database;

5.2. Requirements for functional characteristics:

5.2.1. Functional requirements for the keyboard interface in Telegram:

This subsection describes the functional requirements of this technical specification for the elements of the current section “5. Project requirements”, subsection: 5.3.3. Requirements for interface elements in Telegram.

5.2.1.1. Functional requirements at the regular user level:

Table 5.39.

Table of functional requirements “Account”

Functional requirement	Detailed description
Registering an account in the application database	For initial registration, the user must share his/her Telegram account phone number in chat with the bot. The bot will check for a duplicate number in the database, if such a phone is not registered, the bot will create a new record and add the specified phone to the corresponding field.
Binding an e-mail address (email)	When registering or using a registered account, be possible to bind an email to the account. The bot will check if the specified email is registered in the database, if it is not, the bot will send a message with a confirmation code to the specified email. After that, the user will need to send the received code in chat with the bot for verification.
Authorization via phone number	When starting a new chat with a bot, the user should be able to authorize by sharing the phone of the Telegram account in the chat. Provided that the account with the specified phone number is registered in the database.

End of Table 5.39.

Authorization via e-mail address (email)	When starting a new chat with the bot, the user must be able to authorize by receiving and sending a confirmation code from the specified email. Provided that the email specified by the user is registered behind the account.
Viewing the order history	The user can request their order history, after which the bot will send the user's order records in a ".pdf" file for the last year.
Deletion of account	The user must be given an ability to delete his/her account from the application database. To confirm the deletion, it is necessary to send the text "Confirm".
Changing e-mail (email)	The user should be able to change the linked email to the account by specifying a new email and sending the confirmation code received to it in chat with the bot.
View pending orders	When requesting information about pending orders (product paid for but not received yet), the bot will provide information about the orders the user is expecting.
Selecting the interface language	The user should be able to select the language of the application interface.

Table 5.40.

Table of functional requirements "Catalog"

Functional requirement	**Detailed description**
Displaying service categories	When navigating to the catalog, the system should display all available and relevant service categories to the user. Blocked categories should not be displayed to the regular user.

End of Table 5.40.

Display available services of the selected category	When selecting a category in the catalog, the system should display available and relevant services for selecting and purchasing goods in them. Blocked services should not be displayed to a regular user.
Display available products of the selected service	When selecting a service, the system needs to display the available products as well as the actual data about them to the user.
Search for a product or service in the catalog	The user should be able to search for a product or service in the application catalog using keywords. The user will enter keywords that will be used to search for the required product or service in the catalog database.

Table 5.41.

Table of functional requirements “About Us”

Functional requirement	**Detailed description**
Displaying application data	The system will provide and display the available application data to the user in a chatbot.

Table 5.42.

Table of functional requirements “Technical Support”

Functional requirement	**Detailed description**
Report a problem	The user should be able to notify technical support of faults in the application as a whole or in individual components. An appropriate interface is required, coupled with a form for filling in the data.
View frequently asked questions (FAQ)	The system should be able to provide an up-to-date list of FAQs and their corresponding answers for further review.

5.2.1.2. Functional requirements at the seller level:

Table 5.43.

Table of functional requirements “Catalog of own products”

Functional requirement	Detailed description
CRUD operations for products	The seller should be presented with a user-friendly, clear and simple interface to provide CRUD operations on the goods in his possession. Any changes concerning the data on the goods will be made by filling in special forms by the Seller, thus creating a request for a certain operation to change the goods.
View active requests	The system should be able to provide a view of active change requests as well as a view of completed forms that the vendor has submitted/completed.
Cancel request	The seller should have the ability to cancel an active change request for an item in case they change their mind.
Edit request	The seller must have the ability to edit a submitted item change request.
Product blocking	The system should provide the seller with the functionality to block its own goods from being purchased by normal users.
Product unblocking	The system should provide functionality for the merchant to unblock their own product that is blocked for purchase.

The functional requirements for application support feedback correspond to the functional requirements at the regular user level with the observation that any request has a field indicating the role of the sender. (For more details see “5. Project requirements” section, subsection: 5.2.1.1. Functional requirements at the regular user level , table 5.42.)

5.2.1.3. Functional requirements at the administrator level:

Table 5.44.

Table of functional requirements “Change requests for products”

Functional requirement	Detailed description
Receiving a product change request	Administrators are required to provide functionality to handle requests.
Approval of a request for change of products	When an administrator approves a request, the system should change the status of the request to closed and mark the date the request was closed.
Editing a request to change products	When editing the data in the received request forms, the system will need to send the modified forms back to the vendor for further confirmation from the vendor.
Rejection of a request for change of products	When an administrator chooses to reject a request, he/she must indicate the reason why the request was rejected. This reason must be assigned to the current request in the database for further sending to the seller. When a request is rejected, the system is required to notify the merchant that their request has been rejected. Attach the reason for rejection to the notification message.

Table 5.45.

Table of functional requirements “Technical Support”

Functional requirement	Detailed description
Obtaining a ticket for technical support	The system shall provide functionality for administrators to receive and work on a ticket sent to technical support.
View tickets	Administrators need the ability to retrieve a random or defined ticket using its ID.

End of Table 5.45.

Seller's Ticket	The administrator must have the option to view the ticket from the vendor.
Accepting a ticket	If an administrator accepts a ticket, the system should provide functionality to create a response that will be forwarded to the sender of the request. The system will also mark the ticket as accepted and indicate the accepting administrator. (If a ticket is considered accepted, other administrators cannot receive it)
Rejection ticket	If the administrator rejects a ticket, he/she needs to specify the reason for rejection. The system will change the ticket status to canceled and notify the sender by attaching the reason for rejection.

Table 5.46.

Table of functional requirements “Work with catalog”

Functional requirement	**Detailed description**
Adding a category	The system shall provide the administrator with a form to add a new service category to the application database.
Edit the category	The administrator must have access to the ability to edit the service category data in the application database.
Blocking a category	The system must provide the administrator with the functionality to block a category without allowing access to the services of the blocked category.
Unblocking a category	The system shall provide the administrator with the functionality to unblock a blocked category of application services.

End of Table 5.46.

Adding a service	The system shall provide the administrator with a form to add a new service to the selected application category.
Edit the service	The administrator must have access to edit the service data of the selected category in the application database.
Service blocking	The system should provide the administrator with the functionality to block the service to hide it from the users of the application.
Unblocking the service	The system shall provide the administrator with the functionality to unblock a blocked service, in order to resume access by users of the application.
Product Search	The system shall provide functionality to search for products in the application database by record ID as well as keywords.
Product blocking	The system must provide item locking functionality to the administrator.
Product unblocking	The system must provide the administrator with the functionality to unlock a blocked product.
Edit product	The administrator must have access to the ability to edit the item data. This method is manual, as opposed to approving a completed item change request form from the seller.
Deleting a product	The administrator must have access to the ability to remove a product from the application database.

Table 5.47.

Table of functional requirements “Work with catalog”

Functional requirement	**Detailed description**
Adding an administrator	An administrator with certain privileges should be able to elevate a regular user account to an administrator.
Configuring administrator privileges	An administrator with certain privileges must be able to customize the privileges of other administrators. The privileges for customization do not exceed the available privileges of the administrator initiating the customization.
Revocation of administrator rights	An administrator with certain privileges should be able to downgrade the administrator account to a regular user if the privilege level of the account is not higher than the initiator.
Adding a seller	An administrator with certain privileges should be able to elevate a regular user account to a merchant account.
Kick of the seller	An administrator with certain privileges should be able to downgrade a seller account to a regular user. The seller's items are locked and cannot be modified.
User Blocking	An administrator with certain privileges should be able to lock an account in the application database using an ID, phone number or email.

5.2.2. Other functional requirements:

Table 5.48.

Table of functional requirements for application logging

Functional requirement	Detailed description
Logging levels	Logging should support different levels such as DEBUG, INFO, WARNING, ERROR and CRITICAL so that it is possible to filter messages by importance.
Message format	Logs shall be recorded in a single standardized format to ensure readability and ease of handling.
Data privacy	Logs should not contain sensitive information such as passwords or personal data.

Table 5.49.

Table of functional requirements for bot state processing

Functional requirement	Detailed description
Bot going online	When the application is launched, the system should process this event. Then send a message to the personal chat of the bot owner with the appropriate text about going online.
Bot going offline	When the application is offline, the system should handle this event. After that send a message to the personal chat of the bot owner with the corresponding text about going offline.

5.3. Information and software compatibility requirements:

5.3.1. Requirements for programming languages and software tools:

5.3.1.1. Requirements for programming languages:

Main programming language: Python 3.12.X;

Auxiliary languages: PowerShell, SQL, Bash, Jenkinsfile, Dockerfile;

5.3.1.2. Requirements for software tools:

Selected development environments: VSCode, PowerShell ISE, MySQL Workbench 8.0 CE;

Supporting tools: MySQL Community Server 8.4.X (GPL), Docker Engine 27.1.X, Jenkins 2.462.X (LTS);

Main used libraries and language frameworks: Aiogram 3.13.X, MySQL Connector/Python 9.0.X;

Development platforms and environment: Windows 10, Windows Server 2019, Ubuntu Server 24.04.X (LTS), DockerImage - python:3.12-slim;

5.3.2. Source code requirements:

Readability and code structure: Source code should be written clearly and concisely, using understandable variable, function, and class names. The code should be structured using indentation, comments and division into logical blocks;

Compliance with coding standards: Source code must comply with established coding standards for the language used, including code layout rules, styling and naming conventions;

Documentation: Source code should be accompanied by detailed documentation describing its functionality, structure, input and output data, and API specifications, if applicable;

Testing: The code should be written with testing capabilities in mind;

5.3.3. Requirements for interface elements in Telegram:

This subsection describes the requirements for interface elements for the “NekoShop” project in Telegram.

Since the application is presented as a Telegram chat bot, it will use built-in and provided Telegram interface elements to interact with the user, such as the keyboard and buttons.

This subsection uses keyboard-specific notations such as [...], indicating the use of a separate key for the keyboard, and [...]*, reporting the personal structure and undefined number of elements that individual keys are;

5.3.4.1. Type of keypads for the regular user:

- New chat (not authorized chat with bot)
    - [Personal Account]
        - [Sign up]
        - [Log in]

- Home (authorized chat with bot)
    - [Personal Account]
        - [Delete Account]
        - [Change Mail]
        - [Order History]
        - [Orders Pending]
        - [Interface Language]
    - [Catalog]
        - [Search]
        - [Category]*
            - [Service]*
                - [Product]*
    - [About Us]
    - [Technical Support]

        - [Report a Problem]
        - [FAQ]

5.3.4.2. Type of keypads for the seller:

- Special office (authorized chat with bot)
    - [Catalog of own products]
        - [View Products]
        - [View active requests]
            - [Edit request]
            - [Cancel request]
        - [Create Request]
            - [Add product]
            - [Edit product]
            - [Delete product]
        - [Product Status]
            - [Block]
            - [Unblock]
    - [Technical Support]
        - [Report a Problem]
        - [FAQ]

5.3.4.3. Type of keypads for the administrator:

- Special office (authorized chat with bot)
    - [Requests for change of products]
        - [Get request from a seller]
            - [Accept]
            - [Edit request]
            - [Reject]
    - [Technical Support]
        - [Random Ticket]

        - [Accept]
        - [Reject]
    - [Ticket by ID]
        - [Accept]
        - [Reject]
    - [Ticket from seller]
        - [Accept]
        - [Reject]

- Administrator panel (privilege-based)
    - [Catalog work]
        - [Services Categories]
            - [Category]*
                - [Add Category]
                - [Edit Category]
                - [Category Status]
                    - [Block]
                    - [Unblock]
                - [Category Services]
                    - [Service]*
                        - [Add Service]
                        - [Edit Service]
                        - [Service Status]
                            - [Block]
                            - [Unblock]
        - [Products]
            - [Product Search]
                - [By ID]
                - [By Keywords]

- [Product Status]
  - [Block]
  - [Unblock]
- [Delete product]

- [User Management]
  - [Admin]
    - [Add Admin]
    - [Customize privileges]
    - [Remove from administrations]
  - [Seller]
    - [Add Seller]
    - [Remove from sellers]
  - [Block account]
    - [By ID]
    - [By phone number]
    - [By email]
  - [Unblock account]
    - [By ID]
    - [By phone number]
    - [By email]

6. Stages and phases of development

1. Planning
    - Defining the main idea of the project, goals and objectives. Formation of a short description of the project and its possibilities;
    - Search and familiarization with the required information and documents to further develop the set of documents required for the preparation of the primary documentation for project development;
    - Formation and compilation of the basis of the technical specification, which includes related activities, for example: definition of functional requirements, analysis of the automation object, analysis and formation of requirements for software tools, etc.;

2. Prototyping
    - Search, study and analyze information, software tools and solutions in software code that can be further used to develop;
    - Designing and creating initial/test prototypes and mockups for testing certain functionality of the future system;
    - Analyze the developed prototypes and further merge them to test individual functional areas of the future system;
    - Finalization of technical specification;
    - Designing the architecture of the future system on the basis of the developed prototypes and formed technical specification;

3. Development
    - Prepare, install and customize workspace components and workspace environment for further development;
    - Development of basic core functionality of the system and creation of initial customization;
    - Formation of the structure and skeleton of the system;
    - Development of core functionality and timely testing of developed code elements;
    - Compilation of project documentation;
    - Adjusting the system settings and bringing them to a finalized form;
    - Development of additional functionality;
    - Actualization system settings;
    - Actualization of the settings of auxiliary software tools;
    - Actualization of project documentation;

4. Testing
    - Analyze and formulate the main directions, settings and situations for comprehensive system testing;
    - Preparation of required software solutions for comprehensive testing;
    - Conducting comprehensive testing through manual testing and testing using software solutions;
    - Analyzing and evaluating test results, making appropriate changes and adjustments to the project;

Appendices

1. Documentation standard

- The full title – “Documentation Standard - Recommendations for formatting project documentation in Microsoft Word”.
- Annotation – The document contains recommendations and rules for formatting documents in “Microsoft Word”.
- Purpose - Used to format .docx documents of the “NekoShop” project.
- Resource reference - 4-proxy/SpecificDocumentation
- Additional features:
    - Date of publication from – 19.06.2024
    - Version – 1.0.0
    - Unique identifier – x04

2. Semantic versioning

- The full title – “Semantic Versioning 2.0.0”.
- Annotation – The document contains an approach to software version control.
- Purpose - Used to manage source code versions in a project “NekoShop”.
- Resource reference - 4-proxy/SpecificDocumentation
- Additional features:
    - Version – 2.0.0
    - Unique identifier – x01

3. GOST 34.602-2020

- The full title – “Information technology. Set of standards for automated systems. Technical assignment for developing of automated system”.
- Annotation – The document establishes requirements for the content of the technical specification for the creation of an automated system.
- Purpose – Used to formulate the structure of the technical specification of the project “NekoShop”.
- Resource reference - 4-proxy/SpecificDocumentation
- Additional features:
    - Unique identifier – x03

4. GOST 19.201-78

- The full title – “Unified system for program documentation. Technical specification for development. Requirements for contents and form of presentation”.
- Annotation – The document establishes the procedure for the construction and execution of technical specifications for the development of a program or software product for computing machines, complexes and systems regardless of their purpose and field of application.
- Purpose – Used to formulate the structure of the technical specification of the project “NekoShop”.
- Resource reference - 4-proxy/SpecificDocumentation
- Additional features:
    - Unique identifier – x02