Functional Software Design Description

Template

Version 1.2 ● 14 JAN 2008



Using This Template

The companion tool, **Functional Software Design Description Instructions**, provides detailed direction for completing this template. This and other Framework Extension tools are available on the Framework Web site.

To create a deliverable from this template:

1. Delete the template title page (previous page) and this page.
2. Replace [bracketed text] on the cover page (next page) with your project and agency information.
3. Replace [bracketed text] in the tool header area at the top of page i (Contents page) with the same project and agency information as on the cover page.

Note: Please do not remove or modify content in the footer area.

1. Complete the entire template. Each section contains abbreviated instructions, shown in italics, and a content area. The content area is marked with a placeholder symbol (⇒) or with a table. Relevant text from other project deliverables may be pasted into content areas.

Note: Please do not remove the italicized instructions.

1. Update the table of contents by right-clicking and selecting “Update Field,” then “Update entire table.”

TEXAS PROJECT DELIVERY FRAMEWORK

**FUNCTIONAL SOFTWARE  
DESIGN DESCRIPTION**



**Absolute**

**Car Bot Search**

| VERSION: 1 | REVISION DATE: 13.05.24 |
| --- | --- |

| **Approver Name** | **Title** | **Signature** | **Date** |
| --- | --- | --- | --- |
| Dehod Vitalii | SDLC\_functionalSoftwaredesign\_template |  | 13.05.2024 |
|  |  |  |  |
|  |  |  |  |

Contents

[Section 1.](#_heading=h.3znysh7) Overview 1

[1.1](#_heading=h.2et92p0) Purpose 1

[1.2](#_heading=h.tyjcwt) Scope 1

[Section 2.](#_heading=h.3dy6vkm) System Architecture 2

[Section 3.](#_heading=h.1t3h5sf) Data Dictionary 3

[Section 4.](#_heading=h.4d34og8) Software Domain Design 4

[4.1](#_heading=h.2s8eyo1) Software Application Domain Chart 4

[4.2](#_heading=h.44sinio) Software Application Domains 4

[Section 5.](#_heading=h.17dp8vu) Data Design 5

[5.1](#_heading=h.2jxsxqh) Persistent/Static Data 5

[5.2](#_heading=h.z337ya) Transient/Dynamic Data 5

[5.3](#_heading=h.3j2qqm3) External Interface Data 5

[5.4](#_heading=h.1y810tw) Transformation of Data 5

[Section 6.](#_heading=h.3rdcrjn) User Interface Design 6

[6.1](#_heading=h.26in1rg) User Interface Design Overview 6

[6.2](#_heading=h.4i7ojhp) User Interface Navigation Hierarchy 6

[6.3](#_heading=h.2xcytpi) User Function Categories (or Use Cases) 6

[Section 7.](#_heading=h.lnxbz9) Other Interfaces 8

[Section 8.](#_heading=h.35nkun2) Other Design Features 9

[Section 9.](#_heading=h.1ci93xb) Requirements Traceability Matrix 10

[Section 10.](#_heading=h.3whwml4) References 11

[Section 11.](#_heading=h.2bn6wsx) Glossary 12

[Section 12. Revision History 13](#_heading=h.qsh70q)

[Section 13. Appendices 14](#_heading=h.3as4poj)

# 

# Section 1. Overview

## 1.1 Purpose

*Describe the purpose of the Functional Software Design Description (SDD) and its intended audience.*

The creation of bot telegrams for searching for cars by photo, which are taught on the Auto.ria website, can have significant benefits in various areas:

1. For car buyers:

Quick and convenient search: The bot can help users quickly and conveniently find cars that meet their criteria, without the need to spend time searching independently on Auto.ria.

Filtering by various parameters: The bot can allow users to filter cars by various parameters, such as make, model, price, year of manufacture, mileage, color, body type, transmission, etc.

Obtaining detailed information about the car: The bot can provide users with detailed information about the car they are interested in, for example, description, photos, technical characteristics, reviews.

Contact with sellers: The bot can help users contact car sellers they are interested in directly through Telegram.

2. For car sellers:

Increase the number of potential buyers: The bot can help car sellers find more potential buyers, because their ads will be available not only on Auto.ria, but also on Telegram.

Convenient communication with buyers: A bot can help car sellers conveniently communicate with buyers, answer their questions and arrange appointments.

Optimizing the sales process: A bot can help car sellers optimize the sales process by automating some routine tasks, such as sending out notifications about new cars or answering frequently asked questions.

3. For car dealerships:

Expanding sales channels: The bot can help car dealerships expand their sales channels by offering their cars not only on Auto.ria, but also on Telegram.

Increase customer loyalty: The bot can help car dealerships increase customer loyalty by providing them with convenient and quick access to car information and the ability to communicate with sellers directly through Telegram.

Customer data collection: The bot can help car dealerships collect customer data such as their interests, budget, preferences, which can be used to improve marketing strategies.

⇒

## 1.2 Scope

*Describe the scope of the software to be produced.*

⇒

Scope of software for our project

To determine the amount of software for our project (telegram bot for searching for cars by photo from the Auto.ria website), the following factors must be taken into account:

1. Functional capabilities:

Basic functions:

Search for cars by photo uploaded by the user.

Filter search results by make, model, price, year, mileage, color, body type, transmission, and more.

Providing detailed information about the car that interested the user (description, photos, technical specifications, reviews).

Communication with car sellers directly via Telegram.

Additional features:

Text recognition from car photos (for automatic filling of some search fields).

Search for a car by certain characteristics (for example, the presence of a sunroof, air conditioning, cruise control, etc.).

Saving search results and favorite cars.

Personalization of the bot (setting the language, units of measurement, data format, etc.).

Notifications about new cars that meet the user's criteria.

Integration with other services (for example, Google Maps, Google Translate).

2. Technologies:

Platform: Telegram Bot API.

Programming language: Python.

Libraries: OpenCV (for image recognition), Requests (for HTTP requests), Beautiful Soup (for HTML parsing), SQLAlchemy (for working with databases).

Databases: PostgreSQL.

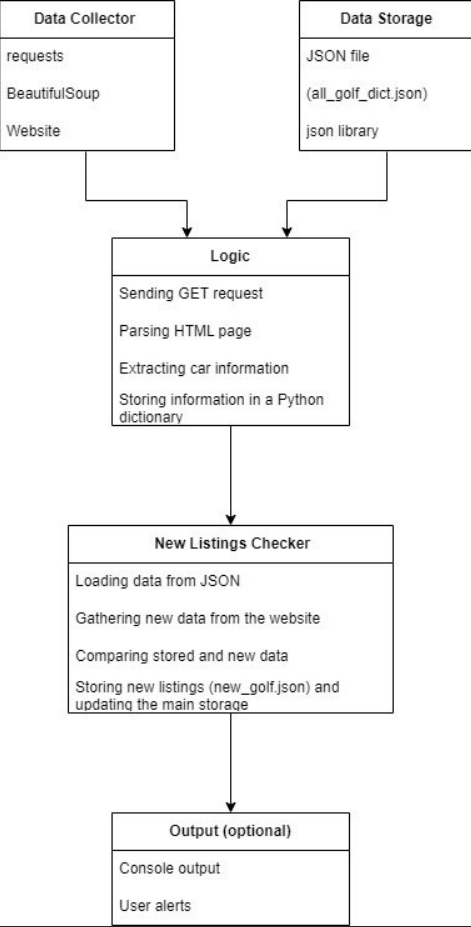
Cloud services: Heroku, Amazon Web Services, Google Cloud Platform.

3. Scope of the code:

Basic functions: about 5,000-10,000 lines of code.

Additional features: + 1,000-5,000 lines of code per feature.

# Section 2. System Architecture

*Provide and describe a figure that depicts the overall system architecture.*

The architecture of the Telegram bot system for searching for cars by photo from the Auto.ria website

Description of the general architecture of the Telegram bot system for searching for cars by photo from the Auto.ria website:

1. User:

Interacts with the bot via Telegram.

Sends a photo of the car or enters a description of the car.

Gets search results and detailed information about the car.

2. Telegram Bot API:

Provides communication between the bot and Telegram.

Receives commands and messages from users.

Sends messages and data to users.

3. Bot:

Processes user commands and messages.

Uploads a photo of the car or receives a description of the car.

Sends requests to the Auto.ria API.

Gets search results and detailed information about the car.

Generates responses for users.

4. Auto.ria API:

Provides access to Auto.ria data.

Receives requests from the bot.

Returns search results and detailed information about the car.

5. Database:

Stores user information, car photos, search results, and other data.

Used by the bot to receive and store data.

# Section 3. Data Dictionary

*Provide a reference to the location of or provide the actual Data Dictionary Table that contains a description of each element in the software application.*

⇒

A database in the form of a json file will be used to save data.

"32693960": {

        "Title": "   Volkswagen Golf VI  2011   ",

        "Price": "  7 550�$���266 213����  ",

        "Url": "https://auto.ria.com/uk/auto\_volkswagen\_golf\_vi\_32693960.html",

        "Location": "  ��������� "

    }

* car\_id (primary key) | Unique identifier for each car | Integer
* Title | Name and model of the car | Text
* Price | Asking price of the car | Decimal
* URL | link to the ad of the car | Text
* Location | car location | Text

# Section 4. Software Domain Design

## 4.1 Software Application Domain Chart

*Provide a figure depicting the set of software application domains showing major components and their relationships.*

**Components:**

* **User Interface (UI):** Handles user interaction through Telegram messages and displays information.
* **Command Handler:** Receives user commands and routes them to the appropriate function.
* **Data Storage (optional):** Stores user preferences, recent car searches (if implemented).
* **Auto Checker:** Checks for new car listings on Auto.ria periodically (using check\_new\_golf function).
* **Car Information Provider:** Retrieves details (title, price, location, URL) about a specific car from Auto.ria.
* **Message Sender:** Sends messages to the user through Telegram.

**Relationships:**

1. **User interacts with UI:** User sends a text message with a command (e.g., "Нові авто").
2. **UI sends command to Handler:** UI identifies the command and forwards it to the Command Handler.
3. **Command Handler calls function:** Based on the command, the Handler calls the corresponding function (e.g., get\_fresh\_car).
   * "Нові авто" - Triggers get\_fresh\_car to check for new listings.
   * "Останні 5 авто" - Retrieves past data from storage (if implemented) or fetches recent listings.
   * "Інформація" - Provides bot information.
   * "Завантажити фото" (not implemented yet) - This functionality is not currently implemented.
4. **Function interacts with Auto Checker (optional):** The function might call check\_new\_golf to retrieve new listings (depending on the implementation).
5. **Auto Checker checks Auto.ria:** The Auto Checker retrieves new car listings from Auto.ria.
6. **Function interacts with Data Storage (optional):** Depending on the function, it might access Data Storage to retrieve past car data or store new information.
7. **Function prepares message:** The function prepares a message containing car details (title, price, location, URL).
8. **Function sends message to Message Sender:** The function sends the prepared message to the Message Sender.
9. **Message Sender sends to User:** The Message Sender delivers the car details message to the user through Telegram.

**Notes:**

* The car\_every\_hour function (commented out) demonstrates a potential approach for sending automatic notifications about new listings. This functionality is not currently active.
* Data Storage usage is optional and depends on the specific implementation (e.g., storing recent searches for "Останні 5 авто").

# Section 5. Data Design

*Customize the following subsections to describe the data contained in databases and other data structures shared between design elements of the software design. Include persistent/static data, transient/dynamic data, external interface data, and transformation of data. Label and title each subsection appropriately.*

### Data Types:

* **Persistent/Static Data:** This data remains constant throughout the bot's operation and is typically stored in configuration files or the code itself. Examples include:
  + **Bot Token:** The unique identifier used by Telegram to recognize the bot.
  + **Auto.ria API Credentials (if applicable):** Information needed to interact with the Auto.ria API (if the bot retrieves data directly).
  + **Default Messages:** Predefined text messages used for greetings, informational responses, and error messages.
* **Transient/Dynamic Data:** This data changes during the bot's execution and is often stored in memory or temporary storage. Examples include:
  + **User Input:** Text commands and messages received from users.
  + **Search Results:** Data retrieved from Auto.ria containing details about new or recent car listings (title, price, location, URL).
  + **User Preferences (if applicable):** User-specific settings like preferred car makes or models (if the bot allows customization).
* **External Interface Data:** This data originates from external sources and is retrieved by the bot for processing. Examples include:
  + **Auto.ria Listings:** Information about cars listed on Auto.ria, obtained through their API (if applicable).

### Data Transformation:

Data transformation occurs when the bot processes or modifies data for different purposes. Here are some examples:

* **User Input Parsing:** The bot might parse user messages to extract commands or search criteria.
* **Car Listing Formatting:** Data retrieved from Auto.ria might be formatted into a user-friendly message containing title, price, location, and URL.

### Data Sharing and Storage:

The way data is shared and stored depends on the specific implementation. Here are some possibilities:

* **In-memory Storage:** Transient data like user input and search results might be stored in memory for quick access during the current session.
* **Configuration Files:** Persistent data like bot tokens and default messages can be stored in configuration files.
* **Database (optional):** A database can be used to store user preferences, recent searches, or historical car data (if the functionality is implemented).

# Section 6. User Interface Design

## 6.1 User Interface Design Overview

**Main Menu:**

* Presents several button options:
  + **Нові авто (New cars):** Triggers a search for new car listings.
  + **Останні 5 авто (Last 5 cars):** Displays information about the most recent five car searches (potentially retrieved from storage).
  + **Інформація (Information):** Provides details about the bot's functionality and its GitHub repository link.
  + **Завантажити фото (Upload photo) (not implemented):** This feature is currently unavailable.

**System Requirements:**

* **Performance:** The user interface should deliver messages promptly to maintain a responsive user experience. Delays in sending or receiving messages can negatively impact usability.
* **Usability:**
  + The menu options should be clear and concise for users to understand.
  + The interface should be easy to navigate using basic text commands.
  + Ideally, the bot should provide informative responses to user queries, including error messages that guide users towards appropriate actions.

**Navigation Hierarchy Diagram:**

The navigation hierarchy for this bot is relatively simple, consisting of a single menu level.

1. User starts a conversation with the bot.
2. The bot presents the main menu with several button options.
3. User selects a desired option by clicking the corresponding button.
   * Clicking "Нові авто" or "Останні 5 авто" initiates a search and displays results.
   * Clicking "Інформація" provides bot info.
   * Clicking "Завантажити фото" (not implemented yet) would likely lead to an informative message explaining the feature's unavailability.
4. User can interact further by selecting new options from the main menu at any point.

# Section 7. Other Interfaces

*Provide a list of all documents and other sources of information referenced in the Software Design Description (SDD) and utilized in developing the Software Design Description. Include for each the document number, title, date, and author.*

| **Document No.** | **Document Title** | **Date** | **Author** |
| --- | --- | --- | --- |
| 1 | M101-project-vison-statement-template | 04.03.2024 | Tetiana Sozanska |
| 2 | CDC\_UP\_Project\_Charter\_Starlight | 04.03.2024 | Tetiana Sozanska |
| 3 | Project-Management-Plan | 13.03.2024 | Balitskii Andrii |
| 4 | Software Requirements Specification\_\_TEMPLATE | 10.04.2024 | Dehod Vitalii |
| 5 | sample-test-plan-template | 15.04.2024 | Dehod Vitalii |

# Section 8. Other Design Features

*Identify changes to the Software Design Description.*

| **Version** | **Date** | **Name** | **Description** |
| --- | --- | --- | --- |
| 1 | 13.05.2024 | SDLC\_functionalSoftwaredesign\_template | First version SDLC\_template |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |