Bot Car Search

Software Requirements Specification

04.04.24

<Tetiana Sozanska,Vita Katsydym,Vitalii Dehod,Andrii Balytskiy,Andrii Cheremshinskiy>

Lead Software Engineer

Prepared for

WSU-TC CptS 322—Software Engineering Principles I

Instructor: A. David McKinnon, Ph.D.

Spring 2005

# **Revision History**

| **Date** | **Description** | **Author** | **Comments** |
| --- | --- | --- | --- |
| 04.04.24 | Version 1 | Tetiana Sozanska,Vita Katsydym,Vitalii Dehod,Andrii Balytskiy,Andrii Cheremshinskiy | First Revision |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# **Document Approval**

The following Software Requirements Specification has been accepted and approved by the following:

| **Signature** | **Printed Name** | **Title** | **Date** |
| --- | --- | --- | --- |
|  | Tetiana Sozanska | Team Lead | 04.04.2024 |
|  | Vitalii Dehod | Developer | 04.04.2024 |
|  | Vitalii Dorosh | Instructor | 04.04.2024 |

**Table of Contents**

**REVISION HISTORY II**

**DOCUMENT APPROVAL II**

**1. INTRODUCTION 1**

1.1 Purpose 1

1.2 Scope 1

1.3 Definitions, Acronyms, and Abbreviations 1

1.4 References 1

1.5 Overview 1

**2. GENERAL DESCRIPTION 2**

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Characteristics 2

2.4 General Constraints 2

2.5 Assumptions and Dependencies 2

**3. SPECIFIC REQUIREMENTS 2**

3.1 External Interface Requirements 3

*3.1.1 User Interfaces 3*

*3.1.2 Hardware Interfaces 3*

*3.1.3 Software Interfaces 3*

*3.1.4 Communications Interfaces 3*

3.2 Functional Requirements 3

*3.2.1 <Functional Requirement or Feature #1> 3*

*3.2.2 <Functional Requirement or Feature #2> 3*

3.3 Use Cases 3

*3.3.1 Use Case #1 3*

*3.3.2 Use Case #2 3*

3.4 Classes / Objects 3

*3.4.1 <Class / Object #1> 3*

*3.4.2 <Class / Object #2> 3*

3.5 Non-Functional Requirements 4

*3.5.1 Performance 4*

*3.5.2 Reliability 4*

*3.5.3 Availability 4*

*3.5.4 Security 4*

*3.5.5 Maintainability 4*

*3.5.6 Portability 4*

3.6 Inverse Requirements 4

3.7 Design Constraints 4

3.8 Logical Database Requirements 4

3.9 Other Requirements 4

**4. ANALYSIS MODELS 4**

4.1 Sequence Diagrams 5

4.3 Data Flow Diagrams (DFD) 5

4.2 State-Transition Diagrams (STD) 5

**5. CHANGE MANAGEMENT PROCESS 5**

**A. APPENDICES 5**

A.1 Appendix 1 5

A.2 Appendix 2 5

# 1. Introduction

This document outlines the Software Requirements Specification (SRS) for a car deal search bot. The bot aims to streamline the car searching process for users by automatically monitoring popular car sale websites and notifying them of new listings matching their specified criteria.

## 1.1 Purpose

This SRS serves as a comprehensive guide for software engineers responsible for designing and developing the car deal search bot. It details all functionalities, data sources, user interactions, and technical specifications required to build the bot.

## 1.2 Scope

Included in Scope:

1. Data source: Getting ads from popular car sales websites like Auto.ria.

2. Notification method: Sending notifications to users about new announcements via Telegram.

3. Filters: The bot includes searches based on the filters you choose on Auto.ria. This will allow you to receive notifications only about those ads that meet your needs.  
Excluded from Scope:  
1. Сar buying process: Unless your bot helps users complete the purchase itself, exclude this step.

2. Integration with external services: If the bot won't connect to financing or car history platforms, mention it here.

3. Advanced features: If there are functionalities you might consider for later versions (e.g., price negotiation assistance), list them here.

## 1.3 Definitions, Acronyms, and Abbreviations

* SRS: Software Requirements Specification
* UI: Interface

## 1.4 References

(1)CDC\_UP\_Project\_Charter\_Template.

(2)M101-project-vison-statement-template.

(3) sample-test-plan-template.

## 1.5 Overview

The document consists of an introduction, a general description, specific requirements and appendices.

# 2. General Description

## 2.1 Product Perspective

The Car Deal Search Bot is a standalone software application designed to work independently. It will not directly integrate with other software systems in the initial phase. However, it relies on internet connectivity to access car sale websites and utilize Telegram for notifications.

## 2.2 Product Functions

The core function of the Car Deal Search Bot is to automate car search on behalf of users. It achieves this by:

* Continuously monitoring designated car sale websites.
* Processing new listings and filtering them based on user-defined criteria.
* Sending notifications to users through Telegram messenger when matching listings are found.

## 2.3 User Characteristics

The target users for the Car Deal Search Bot are individuals actively looking to purchase cars. These users are likely:

* Tech-savvy and comfortable using mobile applications.
* Interested in streamlining their car search process.
* Seeking a way to receive timely updates on new car listings.

## 2.4 General Constraints

* The initial development focuses on a single car sale website (Auto.ria) due to resource limitations. Future expansion to other websites may be considered.
* The bot relies on the availability and stability of internet connectivity for functionality.
* Telegram serves as the primary notification channel in the initial phase. Integration with other platforms may be explored later.

## 2.5 Assumptions and Dependencies

* This SRS assumes users possess basic smartphone or mobile device literacy to operate the Telegram application.
* The functionality of the bot depends on the continued operation and public API access of the target car sale websites (e.g., Auto.ria) and Telegram messenger.

# 3. Specific Requirements

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

The bot shall provide a user-friendly interface for users to:

* Add, edit, and delete search filters for car listings (make, model, year, price range, etc.).
* Manage notification preferences on Telegram.
* View their current search filters and notification settings.

### 3.1.2 Hardware Interfaces

The bot shall function on various mobile devices with internet connectivity. (Define minimum OS requirements if applicable)

### 3.1.3 Software Interfaces

* The bot shall utilize the Telegram Messenger API to send notifications to users. (Provide API version if known)
* The bot shall establish an interface with the target car sale website(s) to access new listings data. (Specify API or data scraping method if applicable)

### 3.1.4 Communications Interfaces

The bot shall require a stable internet connection to function effectively.

## 3.2 Functional Requirements

This subsection details the specific functionalities of the Car Deal Search Bot.

### 3.2.1 <Functional Requirement or Feature #1>

3.2.1.1 Introduction

This functionality allows users to manage their search filters for new car listings.

3.2.1.2 Inputs

* User interacts with the bot's UI to:
  + Add new search filters based on available options (make, model, year, etc.).
  + Edit existing search filters by modifying their criteria.
  + Delete unwanted search filters.

3.2.1.3 Processing

* The bot stores and manages user-defined search filters.
* When processing new listings, the bot considers only the active search filters defined by the user.

3.2.1.4 Outputs

* The bot displays a user-friendly interface to view, edit, and delete currently active search filters.

3.2.1.5 Error Handling

* The bot should handle invalid user inputs for search filters (e.g., out-of-range year selection) by providing appropriate error messages.
* The bot should persist user-defined search filters even after application restart or logout (if applicable).

### 3.2.2 <Functional Requirement or Feature #2>

3.2.2.1 Introduction

This functionality provides users with control over the frequency and timing of notifications for matching car listings.

3.2.2.2 Inputs

* User interacts with the bot's UI to set notification preferences:
  + Choose preferred notification frequency (e.g., daily, hourly, or custom schedule).
  + Define specific times for receiving notifications within the chosen frequency (if applicable for custom schedules).

3.2.2.3 Processing

* The bot stores the user's chosen notification schedule.
* When a new listing matches the user's search filters, the bot considers the user's notification schedule before sending an alert.
* The bot queues notifications based on the chosen frequency and timing preferences.

3.2.2.4 Outputs

* The bot displays a user-friendly interface for configuring notification schedules.
* The bot sends automated notifications to the user via Telegram messenger according to their set schedule and when a matching car listing is found.

## 3.2.2.5 Error Handling

* The bot should handle invalid user inputs for scheduling (e.g., selecting a time in the past).
* The bot should notify the user if encountering errors that prevent notification delivery despite a matching listing being found.

## 3.3 Use Cases

### 3.3.1 Inputs

* User interacts with the bot's UI to set notification preferences:
  + Choose preferred notification frequency (e.g., daily, hourly, or custom schedule).
  + Define specific times for receiving notifications within the chosen frequency (if applicable for custom schedules).

### 3.3.2 Notifications

The bot sends automated notifications to the user via Telegram messenger according to their set schedule and when a matching car listing is found.

## 3.4 Classes / Objects

### 3.4.1 <Class / Object #1>

3.4.1.1 Attributes

* make (string): The car make (e.g., Honda)
* model (string): The car model (e.g., Accord)
* year (integer): The car's year of manufacture
* price (float): The car's asking price
* url (string): The URL of the original car listing on the website

3.4.1.2 Functions

matches\_filters(self, filters): This function takes a dictionary of search filters (make, model, year, price range, etc.) and returns True if the car listing matches all or some of the specified criteria

### 3.4.2 <Class / Object #2>

3.4.2.1 Attributes

search\_filters (list): A list of CarListing objects representing the user's active search filters

* notification\_schedule (object): An object representing the user's preferred notification frequency and timing

3.4.2.2 Functions

* add\_filter(self, car\_listing): This function adds a new CarListing object to the user's search\_filters list.
* edit\_filter(self, index, car\_listing): This function edits an existing CarListing object in the user's search\_filters list based on its index.
* remove\_filter(self, index): This function removes a CarListing object from the user's search\_filters list based on its index .
* set\_notification\_schedule(self, schedule): This function sets the user's notification schedule object

## 3.5 Non-Functional Requirements

### 3.5.1 Performance

The bot shall retrieve new car listings from the target website(s) and process them within an acceptable time frame. The target response time to retrieve and process a new listing should be less than 5 seconds on average.

The user interface should respond to user actions (adding filters, changing settings) with minimal delays. The target response time for user interface interactions should be less than 2 seconds.

### 3.5.2 Reliability

The bot shall strive for continuous operation with minimal downtime. The system uptime target is 99.5% on a monthly basis.

The bot shall implement mechanisms to handle errors during communication with external services (car sale website, Telegram API). The bot should attempt to re-establish connections upon encountering temporary errors and notify users about persistent issues.

### 3.5.3 Availability

The bot should be available for users to access and manage their search filters and notification preferences 24/7.

The bot should deliver notifications for matching car listings in a timely manner. The target notification delivery time from finding a matching listing to sending the notification to the user should be less than 1 minute.

### 3.5.4 Security

User credentials (usernames and passwords) must be securely stored using hashing algorithms to prevent unauthorized access.

The bot should implement secure communication protocols for data transmission between the bot, target website(s), and Telegram messenger.

### 3.5.5 Maintainability

The bot's code shall be well-documented, modular, and easy to understand to facilitate future maintenance and updates.

The system architecture should allow for easy integration of additional features or functionalities in the future.

### 3.5.6 Portability

The initial development of the bot shall target mobile devices with Android and iOS operating systems (define minimum version requirements if applicable).

The bot's architecture should consider future potential porting to web platforms or other operating systems with minimal code modifications.

## 3.6 Inverse Requirements

* **The bot should not overwhelm users with excessive notifications.**
  + Implement a mechanism for users to define the maximum number of notifications they receive per day or timeframe (e.g., hourly).
* **The bot should not send notifications for irrelevant car listings.**
  + Allow users to define additional filters beyond core criteria (make, model, year, price) to further refine their search (e.g., mileage range, transmission type).

## 3.7 Design Constraints

* **Development Resources:** The initial development phase should prioritize core functionalities within resource limitations (time, budget, personnel).
* **Target Platforms:** The bot should function on mobile devices with Android and iOS operating systems (define minimum version requirements if applicable). Future expansion to other platforms can be considered later.

## 3.8 Logical Database Requirements

* The bot may utilize a lightweight local database or persistent storage mechanism on the user's device to store user data such as:
  + User login credentials (securely hashed)
  + User-defined search filters
  + Notification preferences (schedule)
* The database should ensure data integrity and prevent unauthorized access.
* Data retention policies can be defined based on user preferences (e.g., allow deleting search history).

## 3.9 Other Requirements

* **Localization:** Consider future internationalization by allowing interface language selection (English by default).
* **Logging and Monitoring:** The bot should implement mechanisms for logging errors and monitoring system performance to facilitate troubleshooting and future improvements.

# 5. Change Management Process

The following process will be used to ensure that the SRS is up-to-date with changes in design or requirements:

1. Initiation of change:

Any member of the project team can initiate a change by submitting a formal Change Request (CR) through the project management system.

2. Evaluation of the change and decision-making:

The Project Manager and Project Director evaluate the impact of the proposed change on the project.

3. Implementation of the change:

After approval, the project team makes changes to the SRS.

4. Notification and documentation:

All changes are documented and stakeholders are notified of SRS updates.

5. Versioning of SRS:

An updated SRS gets a new version number and release date.

## 