SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

FINAL WORK br. xxxx

**WEB WORKOUT PLANNER**

Dalijo Vorkapić

Zagreb, mjesec 2022.

[Introduction 1](#_Toc103000604)

[1. Requirements specification 3](#_Toc103000605)

[1.1 Functional requirements 3](#_Toc103000606)

[1.1.1 Actors and their functional requirements 4](#_Toc103000607)

[1.1.2 Use cases 5](#_Toc103000608)

[1.1.3 Use case diagram 16](#_Toc103000609)

[1.2 Non-functional requirements 17](#_Toc103000610)

[2. System architecture and design 17](#_Toc103000611)

# Introduction

In the last few years, there has been a great increasing interest of people in physical activities, such as sports, exercising and training, and they have all become an important part of their lives. With the initial appearance of the coronavirus, people suddenly had much more time and comfort for themselves, and many of them started with the so-called self-care routines. The main part of most of these routines, the one that has been preserved and grown to this day, is individual training. It is popular because first of all, it is very healthy and beneficial for human body and mind, it requires little to no equipment, it can be performed anywhere, at home, outside or in the gym, and it does not necessarily require a personal coach, since all the guides can be found online and the person can come up with its own plan. By that approach, people have to put together their own exercise and workout details, plans and schedule, as well as monitor their progress, usually through tracking the number of repetitions and sets of specific exercise, or by tracking the required training execution time. It takes a lot of time, thought and energy, and the final product, usually turns out to be disorganized, incomplete and difficult to update and keep the track of, taking into account that planning, writing and tracking is done with a classic pen and paper, note application or with an excel spreadsheet.

Since we live in the age of the Internet, where various services are available to the users through web applications, The Web Workout Planner application will serve as a solution to the mentioned problem of monitoring progress and organization of the workouts because it reduces the complexity of performing the mentioned actions and it offers insight into the user statistics of workouts, to help track the progress better.

This work presents the process of developing a Web Workout Planner application, although its template can be applied to any type of personal planning and progress tracking application.

The Web Workout Planner application development process will be presented in stages, and it will be based on the Software development lifecycle (SDLC) process, which consists of a detailed plan describing requirements analysis, requirements specification, architecture designing, product implementation and product testing and deployment.

Requirements specification will be carried out using the defined functional requirements of the application and by combining them with the use cases. Architecture of the application will be divided into the 3 groups: database, backend (“server-side”) development and frontend (“client-side”) development.

As for the necessary tools for application development and documentation, the Astah UML tool will be used to create UML diagrams for requirements specification and application architecture. Regarding the architecture and implementation, Java Spring Boot framework will be used for backend programming, Angular TypeScript framework for frontend programming and PostgreSQL for database. For application deployment, Heroku Server, Git for the implementation versioning and Github for project repository.

The aim of this paper is to present the full process of software development and to deliver a complete and stable user web application.

# Requirements specification

Requirements specification is the first phase of software development. It is a collection of all requirements that are to be imposed on the design and verification of the application. A requirement is a thing that product must do (“system shall do”) or a quality it must have (“system shall be”). It is determined by the requirements engineering process, which analyzes, structures, documents and verifies user-required system services and usage constraints. Based on the content, each requirement specification can be divided into function and non-function requirement. “Differences between functional and non-functional requirements (*Table 1.1*);

Table 1.1: Functional vs. Non-functional requirements

|  |  |  |
| --- | --- | --- |
|  | **Functional requirements** | **Non-functional requirements** |
| Objective | Describe what the product does | Describe how the product works |
| End result | Define product features | Define product properties |
| Focus | Focus on user requirements | Focus on user expectations |
| Documentation | Captured in use case | Captured as a quality attribute |
| Essentiality | Mandatory | Not mandatory, but desirable |
| Origin type | Usually defined by user | Usually defined by developers |
| Testing | Component, API, UI testing | Performance, usability, security testing |
| Types | Interface, authentication,  authorization levels, etc. | Usability, scalability, reliability,  Performance, etc. |

## Functional requirements

A functional requirement is a description of the service that the software must offer to the stakeholder (actor). It describes a software system or its component. A function is nothing but inputs to the software system, its behavior, and outputs.

For this application, we define actors with their corresponding functional requirements. Initiator is the type of actor that directly interacts with the system, while participant does not interact and it only gets passively affected by system.

### Actors and their functional requirements

**Actors:**

* Initiators
  + Administrator
  + User
* Participants
  + Database

**User can:**

* Login to the application
* View the data of:

1. Workouts
2. Exercises
3. Schedule
4. Statistics

* Create, update and delete data of:

1. Workouts
2. Exercises
3. Scheduled workouts

**Administrator can:**

* Everything that user can
* Manage users
  1. Register new users
  2. Remove users

**Database:**

* Saves all data from the application
* Deletes selected data
* Updates selected data

### Use cases

A use case is a written description of how users will perform tasks on the website. It outlines, from a user's point of view, a system's behavior as it responds to a request. Each use case is represented as a sequence of simple steps, beginning with a user's goal and ending when the goal is fulfilled. Use cases add value because they help explain how the system should behave and in the process, they also help brainstorm what could go wrong.  They provide a list of goals and this list can be used to establish the cost and complexity of the system.

In this work, use cases describe a combination of the following elements: primary actor, goal, precondition, basic flow (main success scenario) and alternative paths (variations to the main success scenario).

Description of application use cases:

**UC1 – Registration**

* Primary actor: Administrator
* Goal: Registration of the new user
* Participants: Database
* Precondition: Administrator logged in
* Basic flow:
  1. Administrator clicks on the account icon
  2. Administrator selects “Register new user” from the menu
  3. Registration form pop up window opens
  4. Administrator inserts the new user data
  5. Administrator submits the input
  6. New user added to the database
  7. Pop up window closes
  8. Account menu closes
* Alternative paths:

3. Administrator clicks outside the window

1. Registration pop up window closes

4. Administrator cancels the registration

1. Registration pop up window closes

**UC2 – Login**

* Primary actor: User
* Goal: User login
* Participants: Database
* Precondition: User registered by administrator
* Basic flow:
  1. User clicks selects the Login option
  2. User is being redirected to the Login page
  3. Login form is displayed
  4. User inserts email and password
  5. User submits the input
  6. Database approves the credentials
  7. User is being redirected to the Home page
* Alternative paths:

4. Improper input

1. User is not able to submit the input

5. Given credentials are wrong

1. Login page reopens

**UC3 – Logout**

* Primary actor
  1. User
  2. Administrator
* Goal: User login
* Participants: Database
* Precondition: User / Administrator logged in
* Basic flow:
  1. User clicks on the account icon
  2. User selects “Logout” from the menu
  3. User is redirected to the Login page

**UC4 – Exercises overview**

* Main Actor: User
* Goal: Displaying a list of currently stored exercises
* Participants: Database
* Precondition: Login
* Basic Flow:
  1. On the home screen, User selects the Exercises option
  2. User is being redirected to the Exercise page
  3. List of Exercises is displayed

**UC5 – Filtering exercises**

* Main Actor: User
* Goal: Displaying filtered exercises
* Participants: Database
* Precondition: Login
* Basic Flow:
  1. On the home screen, user selects the „Exercises“ option
  2. User is being redirected to the Exercises page
  3. List of exercises is displayed on the page
  4. User inserts filter in the filter input
  5. Filtered exercises are displayed on the page
* Alternative path:

5. No exercise shown for given filter

**UC6 – Creating new exercise**

* Main Actor: User
* Goal: Adding new exercise
* Participants: Database
* Precondition: Login
* Basic Flow:
  1. On the home screen, User selects the Exercises option
  2. User is being redirected to the Exercises page
  3. User selects „Create new exercise“ option
  4. A new window with the create form pops up
  5. User inserts the data for the new exercise
  6. User submits the input
  7. New exercise created in the database
  8. Pop up window closes
  9. Exercises page refreshes with the changes
* Alternative path

6. Exercise „Name“ input is left empty

1. User is not able to submit the input

7. Exercise name already exists

1. Submit request is rejected

**UC7 – Editing exercise**

* Main Actor: User
* Goal: Editing chosen exercise
* Participants: Database
* Precondition:
  1. User logged in
  2. Exercise existing
* Basic Flow:
  1. On the home screen, User selects the Exercises option
  2. User is being redirected to the Exercises page
  3. User selects the „Edit“ icon of the chosen exercise
  4. The change form window pops up
  5. User changes the desired exercise data
  6. User submits the changes
  7. Exercise is updated in the database
  8. Pop up window closes
  9. Exercises page refreshes with the changes
* Alternative path

4. Exercise „Name“ input is changed to empty

1. User is not able to submit the input

6. Exercise name already exists

1. Update request is rejected

6. User cancels the changes

1. Pop up window closes

**UC8 – Deleting exercise**

* Main Actor: User
* Goal: Deleting chosen exercise
* Participants: Database
* Precondition:
  1. User logged in
  2. Exercise existing
* Basic Flow:
  1. On the home screen, User selects the Exercises option
  2. User is being redirected to the Exercise page
  3. User selects the „Delete“ icon of the chosen exercise
  4. An alert window pops up
  5. User confirms the deletion
  6. Exercise is deleted from the database
  7. Pop up window closes
  8. Exercises page refreshes with the changes
* Alternative path

4. User clicks outside of the pop up window

1. Pop up window is closed

5. User cancels the deletion

1. Pop up window is closed

**UC9 – Workouts overview**

* Main Actor: User
* Goal: Displaying a list of currently stored workouts
* Participants: Database
* Precondition: User logged in
* Basic Flow:
  1. On the home screen, User selects the Workouts option
  2. User is being redirected to the Workouts page
  3. Workouts list is displayed

**UC10 – Filtering workouts**

* Main Actor: User
* Goal: Displaying filtered exercises
* Participants: Database
* Precondition: Login
* Basic Flow:
  1. On the home screen, user selects the „Exercises“ option
  2. User is being redirected to the Exercises page
  3. List of exercises is displayed on the page
  4. User inserts filter in the filter input
  5. Filtered exercises are displayed on the page
* Alternative path:

5. No exercise shown for given filter

**UC11 – Workout details overview**

* Main Actor: User
* Goal: Displaying details for selected workout
* Participants: Database
* Precondition:
  1. User logged in
  2. Workout existing
* Basic Flow:
  1. On the home screen, User selects the Workouts option
  2. User is being redirected to the Workouts page
  3. Workouts list is displayed
  4. User clicks on the desired workout
  5. Workout details are shown on the page

**UC12 – Creating new workout**

* Main Actor: User
* Goal: Adding new workout
* Participants: Database
* Precondition: User logged in
* Basic Flow:
  1. On the home screen, User selects the Workouts option
  2. User is being redirected to the Workouts page
  3. User selects „Create new workout“ option
  4. A new window with the create form pops up
  5. User inserts the data for the new workout
  6. User submits the input
  7. New workout created in the database
  8. Pop up window closes
  9. Workouts page refreshes with the changes
* Alternative path

6. Workout „Name“ input is left empty

1. User is not able to submit the input

7. Workout name already exists

1. Submit request is rejected

**UC13 – Editing workout**

* Main Actor: User
* Goal: Editing chosen workout
* Participants: Database
* Precondition:
  1. User logged in
  2. Workout exists
* Basic Flow:
  1. On the home screen, User selects the Workouts option
  2. User is being redirected to the Workouts page
  3. User selects the „Edit“ icon of the chosen workout
  4. The change form window pops up
  5. User changes the desired workout data
  6. User submits the changes
  7. Workout is updated in the database
  8. Pop up window closes
  9. Workouts page refreshes with the changes
* Alternative path

5. Exercise „Name“ input is changed to empty

1. User is not able to submit the input

6. Exercise name already exists

1. Update request is rejected

6. User cancels the changes

1. Pop up window closes

**UC14 – Deleting workout**

* Main Actor: User
* Goal: Deleting chosen workout
* Participants: Database
* Precondition:
  1. User logged in
  2. Workout exists
* Basic Flow:
  1. On the home screen, User selects the Workouts option
  2. User is being redirected to the Workouts page
  3. User selects the „Delete“ icon of the chosen workout
  4. An alert window pops up
  5. User confirms the deletion
  6. Workout is deleted from the database
  7. Pop up window closes
  8. Workouts page refreshes with the changes
* Alternative path

4. User clicks outside of the pop up window

1. Pop up window is removed

5. User cancels the deletion

1. Pop up window is removed

**UC15 – Schedule overview**

* Main Actor: User
* Goal: Displaying the current schedule
* Participants: Database
* Precondition: User logged in
* Basic Flow:
  1. On the home screen, User selects the Schedule option
  2. User is being redirected to the Schedule page
  3. User schedule is displayed

**UC16 – Adding workout to schedule**

* Main Actor: User
* Goal: Adding workout to schedule
* Participants: Database
* Precondition:
  1. User logged in
  2. Workout exists
* Basic Flow:
  1. On the home screen, User selects the Schedule option
  2. User is being redirected to the Schedule page
  3. User marks the wanted period on the schedule
  4. Workout selection pop up window opens
  5. User selects which workout to put in the schedule
  6. User submits the changes
  7. Pop up window closes
  8. Scheduled workout is added to database
  9. Schedule display updates
* Alternative path:

6. User cancels the changes

1. Pop up window closes

**UC17 – Workout period change**

* Main Actor: User
* Goal: Changing the period of workout
* Participants: Database
* Precondition:
  1. User logged in
  2. Scheduled workout exists
* Basic Flow:
  1. On the home screen, User selects the Schedule option
  2. User is being redirected to the Schedule page
  3. User resizes the workout period box he wants to change
  4. Scheduled workout is updated in the database
  5. Schedule display updates

**UC18 – Deleting scheduled workout**

* Main Actor: User
* Goal: Deleting workout from schedule
* Participants: Database
* Precondition:
  1. User logged in
  2. Scheduled workout exists
* Basic Flow:
  1. On the home screen, User selects the Schedule option
  2. User is being redirected to the Schedule page
  3. User selects the workout period box he wants to delete
  4. User selects the “Delete” icon
  5. Alert pop up window opens
  6. User confirms deletion of the scheduled workout period
  7. Pop up window closes
  8. Scheduled workout is removed from the database
  9. Schedule display updates
* Alternative path:

5. User clicks outside of the window

1. Pop up window closes

6. User cancels the changes

1. Pop up window closes

**UC19 – Statistics overview**

* Main Actor: User
* Goal: Displaying user workout statistics
* Participants: Database
* Precondition: User logged in
* Basic Flow:
  1. On the home screen, User selects the Statistics option
  2. User is being redirected to the Statistics page
  3. User statistics are displayed
* Alternative path:

3. No statistics available

### Use case diagram

A use case diagram is part of UML behavioral diagrams, which depict the elements of a system that are dependent on time and that convey the dynamic concepts of the system and how they relate to each other, such as graphical depiction of a user's possible interactions with a system, which was previously mentioned through the use cases. “Application’s use cases UML diagram (*Figure 1.1*).”

Diagram

Description automatically generated

Figure 1.1: Application's use case diagram

## Non-functional requirements

A non-functional requirement is a specification that describes the software’s operation capabilities and constraints that enhance its functionality, such as performance, security, reliability, usability, scalability and maintainability.

**Application’s non-functional requirements:**

* Performance
  + Single page application
  + Fast communication with database
* Scalability
  + Application can stand high workload
* Availability
  + Application is always online
  + Multiple users possible in real time
* Reliability
  + No unexpected crashes
* Security
  + Authentication
  + Password hashing
  + Input validation
* Usability
  + Minimalistic and simple user interface

# System architecture and design