Coach4Health

Analysis and Design Document

Student: Lazăr Denisa Ștefania

**Group: 30238**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[**I. Project Specification** 4](#_Toc5288596)

[**II. Elaboration – Iteration 1.1** 5](#_Toc5288597)

[1. Domain Model 5](#_Toc5288598)

[2. Architectural Design 7](#_Toc5288599)

[2.1 Conceptual Architecture 7](#_Toc5288600)

[2.2 Package Design 8](#_Toc5288601)

[2.3 Component and Deployment Diagrams 9](#_Toc5288602)

[**III. Elaboration – Iteration 1.2** 10](#_Toc5288604)

[1. Design Model 10](#_Toc5288605)

[1.1 Dynamic Behavior 10](#_Toc5288606)

[1.2 Class Design 10](#_Toc5288607)

[2. Data Model 10](#_Toc5288608)

[3. Unit Testing 10](#_Toc5288609)

[**IV. Elaboration – Iteration 2** 10](#_Toc5288610)

[1. Architectural Design Refinement 10](#_Toc5288611)

[2. Design Model Refinement 10](#_Toc5288612)

[**V. Construction and Transition** 11](#_Toc5288614)

[1. System Testing 11](#_Toc5288615)

[2. Future improvements 12](#_Toc5288616)

[**VI. Bibliography** 12](#_Toc5288617)

# Project Specification

Nowadays, everyone has a busy schedule. We are always caught up in long hours at work or univeristy and we feel that time is our biggest enemy. So, unfortunately we tend to neglect our health by leaving physical activities aside and by having an unsuitable diet.

To overcome the negative effects a sedentary lifestyle has on us, I will implement a fitness application that will be like a personal coach at home. This means that whenever its users do not have enough time to go to the gym, they can access the app and get assistance regarding a certain type of workout and different diet plans.

Next, the functionalities and the functional and non-functional requirements of this application called “Coach 4 Health” will be described.

The functional requirements are described in the list below:

* **Account creation** 🡪 in order for someone to use the app, he/she must create a personal account. This is achieved by completing a sign-up form with personal information
* **Choosing a workout plan** 🡪 after creating a personal account or after logging in, any user will be able to choose a workout plan suitable for his/her needs. Workout plans can vary by difficulty, duration and different type of exercises and challenges
* **Choosing a meal plan** 🡪 to benefit even more from this application, users can choose between a few meal plans suggested with each type of workout
* **Progress tracking** 🡪 once a user has completed a workout, he/she can remove it from the list of workouts in the chosen plan and add it to a “completed” list. So, the user can monitor daily progress.
* **User management** **& guidance**🡪 a second type of user for this application will be the fitness coaches. Their main role is to create and provide different types of workouts to be available to regular users. A fitness coach will keep track of which users do a workout provided by him/her, being able to also monitor their ongoing progress and send them suggestions.
* **Daily tips & tricks 🡪** any regular user will receive a daily tip regarding exercises and meals. This tip aims to improve their experience and keep them motivated to progress everyday in living a healthier life
* **Payment 🡪** regular user must pay (by card or bank transfer) for their workout and meal plans
* **30-day money back guarantee 🡪** if within 30 days of purchasing workout & meal plans a user is not satisfied, he/she can ask for a refund

The non-functional requirements are described next:

* **Performance** 🡪 the response time of the system will be kept low meaning because a user should not wait a long time for performing basic activities like creating an account or choosing a workout type and so on
* **Security** 🡪 all the personal data of the users must be kept safe from potential malicious users (data encryption)
* **Maintainability** 🡪 when adding new services/features to the application or when updating existing ones, the effort should be kept at a minimum level
* **Reusability 🡪** different modules of the application will be designed such that they can be used in future projects as well for example the login and sign in modules)
* **Recoverability** 🡪 in case of any kind of crash and/or hardware failure, the application should recover fast, being able to continue doing the data processing and operations before the failure occured
* **Scalability 🡪** the application should be able to keep working just as well as before when it is changed in size and volume, to meet a growing need of its users

# Elaboration – Iteration 1.1

# Domain Model

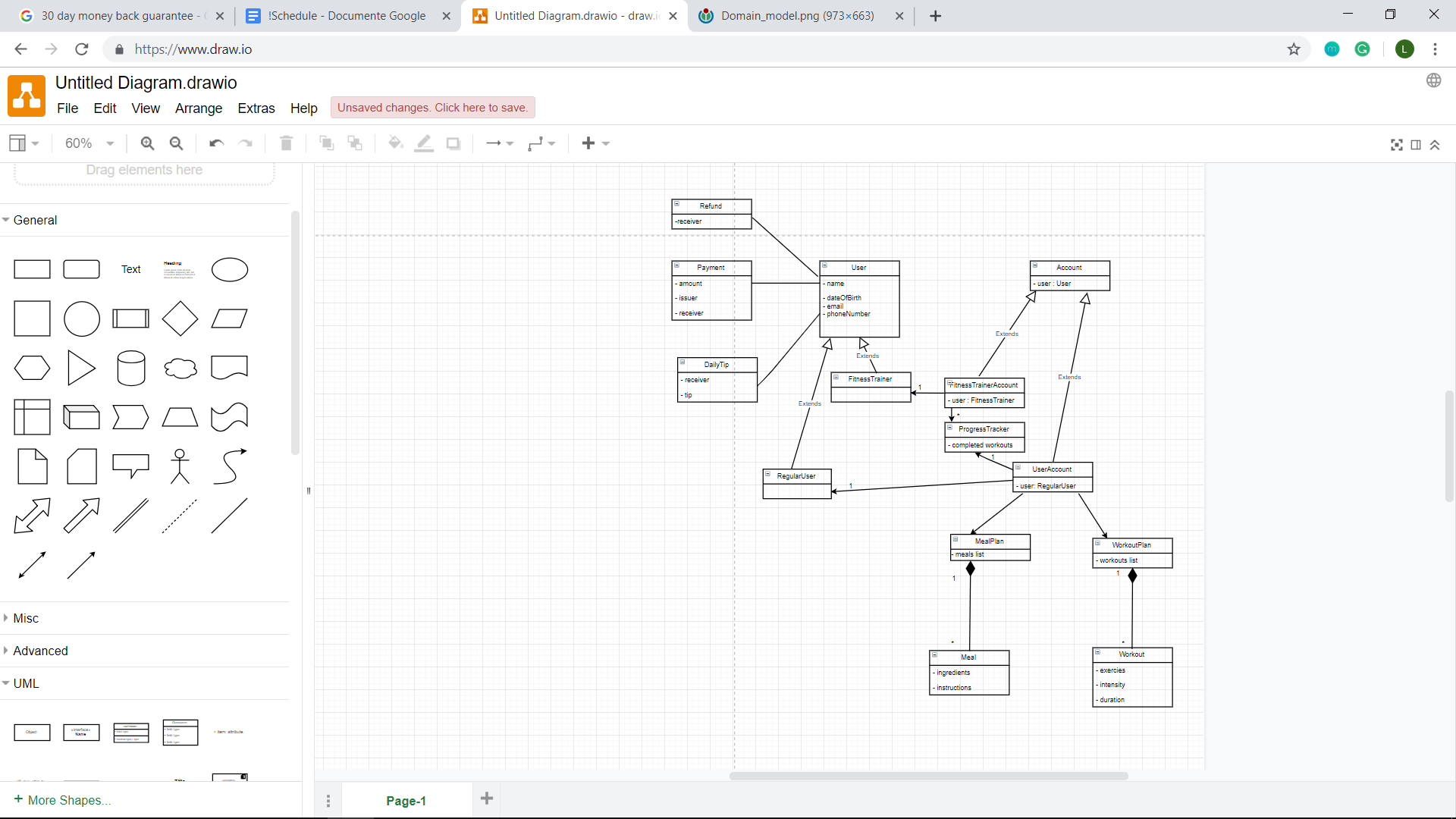
In software engineering, a **domain model** is a conceptual model of the domain that incorporates both behavior and data. It is a structural model of basic domain concepts and the relationships between them and it may contain domain objects, conceptual classes, associations or attributes.

The **main steps in building a domain model** are:

* **Finding conceptual classes** (they represent ideas, things or objects in the domain); this can be achieved by identifying nouns in the system’s description/specification and the filtering them based on different criteria
* **Drawing the conceptual classes as classes in a UML class diagram** (note: a conceptual class includes only a few relevant attributes, not operations)
* **Identifying attributes of each class and the relationships between classes**

Possible conceptual classes that I identified for this application are: ***user***(which can be either a **regular user** or a **fitness trainer**), ***account*** (**user account** or **fitness trainer** account)**, *workout*, *meal, workout plan*** (has more workouts associated to it)*,* ***meal plan*** (has different meals associated to it), **daily tip** (offered to users based on their chosen workout and meal plan),**progress tracker**, ***payment, refund***. Next, I will describe possible attributes for each conceptual class and in the conceptual class diagrams show the relationships that can be established between them (note: during the design process for the app “Coach4Health” they will be subject to modification and improvements)

* **Conceptual class attributes**
* **User :** name, date of birth, email, phone number,
* **Account:** user; based on the user type the account will offer different functionalities
* **Workout:** contains a list of exercises, duration and intensity
* **Workout plan:** a list of different workouts and a cost
* **Meal:** list of necessary ingredients, cooking instructions
* **Meal plan:**  a list of different meals and a cost
* **Daily tip:** receiver(can only be a regular user, not a fitness trainer), the tip itself(can be implemented as a message sent to the user)
* **Progress tracker:** list of completed workouts by a certain user
* **Payment:** total amount(meal plan cost + workout plan cost), issuer (regular user), receiver(fitness coach/trainer)
* **Refund:** receiver(the user who within 30 days, is not satisfied with the meal plan and workout previously chosen)

**Conceptual class diagram**

The diagram above represents an overview of the system as a whole.

# Architectural Design

## Conceptual Architecture

A conceptual architecture represents an initial architectural design of an application. It can be designed by analyzing the requirements (especially the functional ones) so it can provide on overview of the structure that a system has in terms of functionalities.

For the purpose of this application I thought about using a layered (n-tire) architecture. The main reason for this choice is that a layered architecture divides the application in different layers, each of them having a precise functionality. Another advantages of using a layered architecture include: ease of updating a specific layer not affecting too much the other ones (increased maintainability), increased simplicity and scalability. Also the data flow between layers is easy to control.

Next, a diagram showing the main layers of the application, together with a description of the content of each layer will be presented.



* **Presentation layer**

The presentation layer will deal with the graphical user interface. From this layer a user will issue different requests, which will be passed down to the following layers.

* **Business layer**

In this layer the actual data processing will start. Data necessary for different operations will be received from the database layer and after being processed it will be sent back to the presentation layer.

* **Persistence layer**

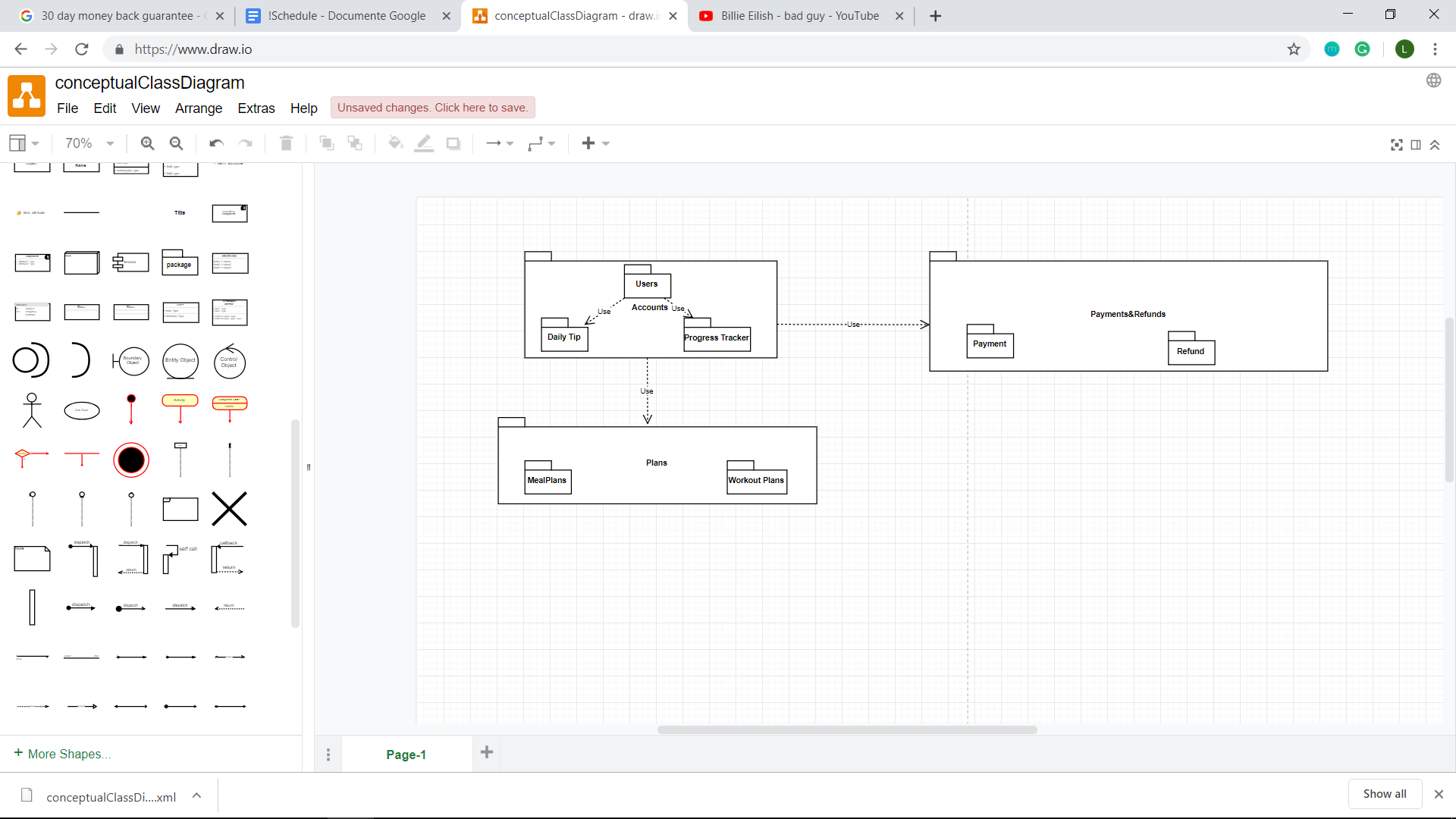
This layer will act as an intermediate layer between the business logic layer and the database layer. It can contain for example, the data access objects (DAO).

* **Database layer**

This layer is responsible for managing the database used to store the application’s data .

## Package Design

The package diagram corresponding to “Coach4Health” application is illustrated below.



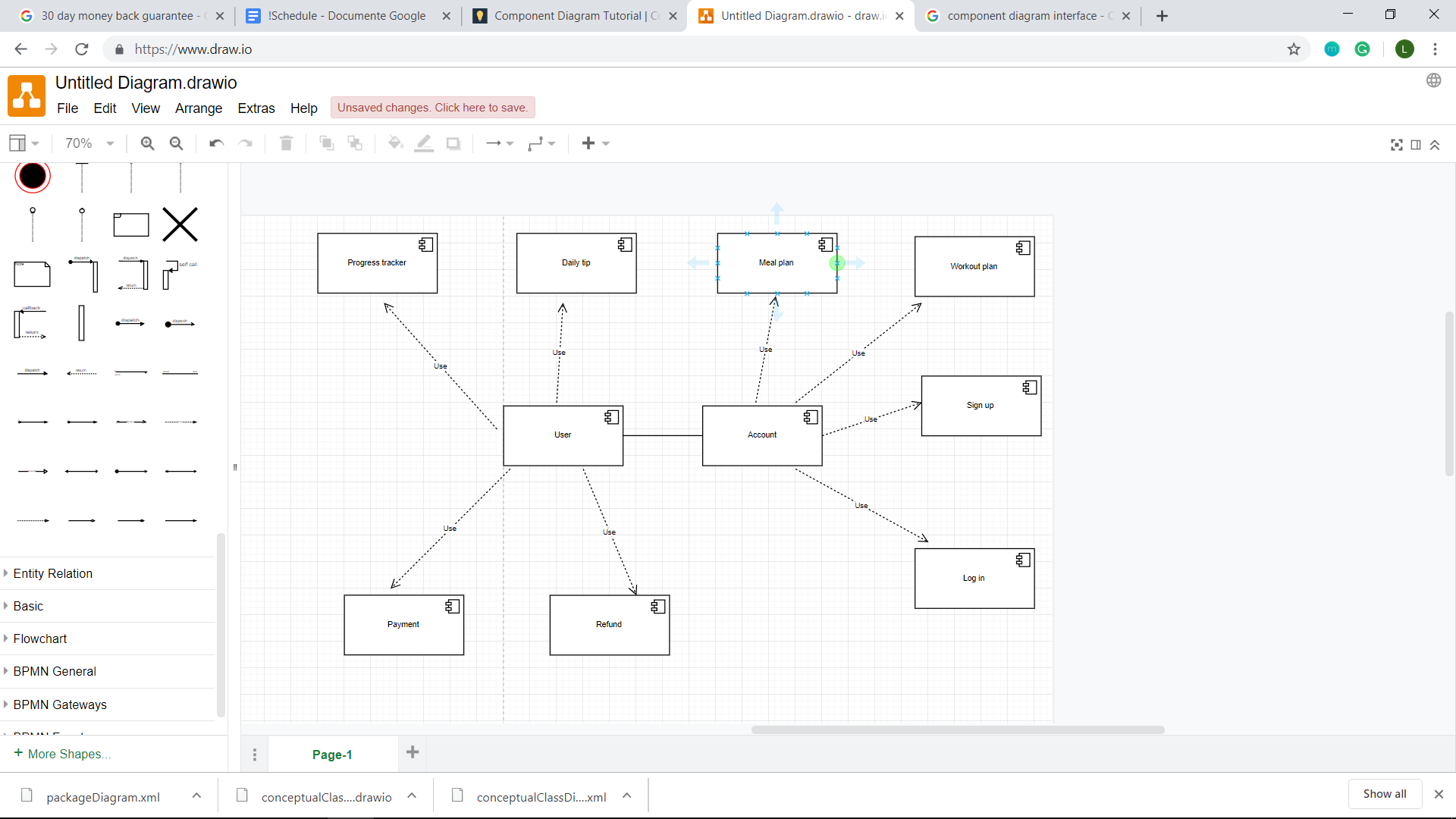
The “Users” package will contain the classes modelling different type of users the application has (regular users and fitness trainers). “Daily Tip” package contains classes responsible for creating tips based on chosen workout and meal plans. “Progress Tracker” package will contain classes that are necessary for the monitoring progress feature. All these packages are sub-packages of a larger one called “Accounts”. This “Accounts” package will contain classes responsible for account creation and facilities associated with different account types.

The “Plans” package contains two sub-packages called “Meal Plans” and “Workout Plans”. The “Meal Plans” package will contain classes modelling meals and creating meal plans. The “Workout Plans” package will contain classes modelling workouts and creating different types of workout Plans.

The “Payment&Refunds” package has two sub-packages called “Payment” and “Refund”. The “Payment” package will contain classes necessary for making a payment once a workout plan and a meal plan have been chosen. The “Refund” package will contain classes modelling the refund process if the user is not satisfied with his/her workout/meal plan.

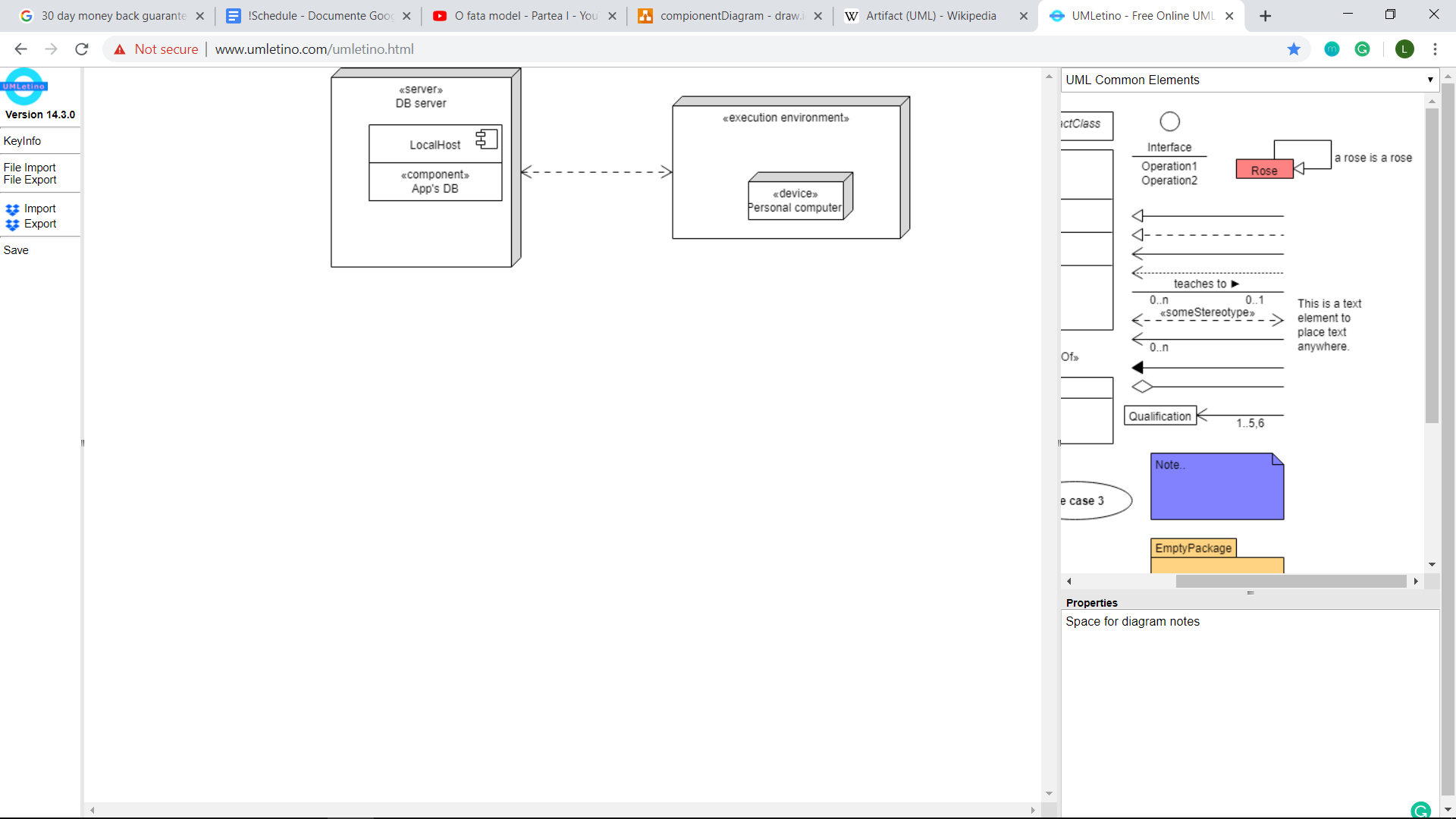
## Component and Deployment Diagrams

* Component diagram



The main components of the application are illustrated above. The Sign Up, Log in, Payment and Refund components will need special interfaces in order to be able to interact with the database.

* Deployment diagram



# In an initial phase, the application will be made to run on personal computers. At a later stage it can be extended to run on mobile devices (using Android and/or iOS). In order for the application to function in a proper manner it needs to work together with a database (where all the data related to the application will be stored) which will reside on localhost initially (something like XAMPP and PhpMyAdmin).

# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior

*[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]*

## Class Design

*[Create the UML class diagram; apply GoF patterns and motivate your choice]*

# Data Model

*[Create the data model for the system.]*

# Unit Testing

*[Present the used testing methods and the associated test case scenarios.]*

# Elaboration – Iteration 2

# Architectural Design Refinement

*[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]*

# Design Model Refinement

## *[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]*

# Construction and Transition

# System Testing

Integration testing represents the testing performed to expose defects in the interfaces and in the interaction between the components of an application. The integration testing is usually performed by the developers themselves or by independent testers.

There exist different approaches for performing integration testing, the most used being: “Big Bang” ,”Top Down”, ”Bottom Up” and “Sandwich/Hybrid”. The last one is a combination between the “Top Down” and “Bottom Up” approaches.

First of all, I will perform unit testing before proceeding with the integration testing using the “Bottom Up” approach.

A few possible test cases are contained in the following list:

1. *Log in testing* 🡪 this test-case will focus on the application’s response when a user introduces a valid username and password. The system must not fail in checking that the username and associated password exist in the databse.
2. *Sign in testing* 🡪 this test case is centered on the creation of a new account inside the application. The data introduced by a future users when creating a new account must be inserted correctly in the database and checked again invalid format (for example a negative age or not providing required information like name and/or email).
3. *Choosing workout or meal plans testing* 🡪 this test case will cover that once a user chooses a specific meal/workout plan, that plan will correctly be assigned to him/her
4. *Secure payment testing* 🡪 Since a user can have the option to pay by using a card, the fact that sensible information (like card number) will be manipulated is indeed. So a rigorous testing of the components dealing with payments and refunds must be performed in order to avoid unpleasant situations in the future.

# Future improvements

Still being in the incipient phase, the system can be improved in a few ways, which will be described in this section.

First of all, at this time the “Coach4Health” is available only as a desktop application running on Windows 10, so it can be extended to be available for more mobile devices and other operating systems as well.

Also, a vital improvement needed is the one regarding data security. Data should be encrypted such that it will become less vulnerable.

Another improvement that could increase the utility of “Coach4Health” is creating a facility where users can rate workouts by their effectiveness. In the future, users should also be able to receive discounts on different gym equipment and workout clothes, but also on workout and meal plans as they progress into doing more advanced workout routines.

# Bibliography

<https://en.wikipedia.org/wiki/Non-functional_requirement>

<https://study.com/academy/lesson/domain-model-vs-data-model.html>

<https://en.wikipedia.org/wiki/Domain_model>

<http://stg-tud.github.io/eise/WS11-EiSE-07-Domain_Modeling.pdf>

<http://softwaretestingfundamentals.com/integration-testing/>