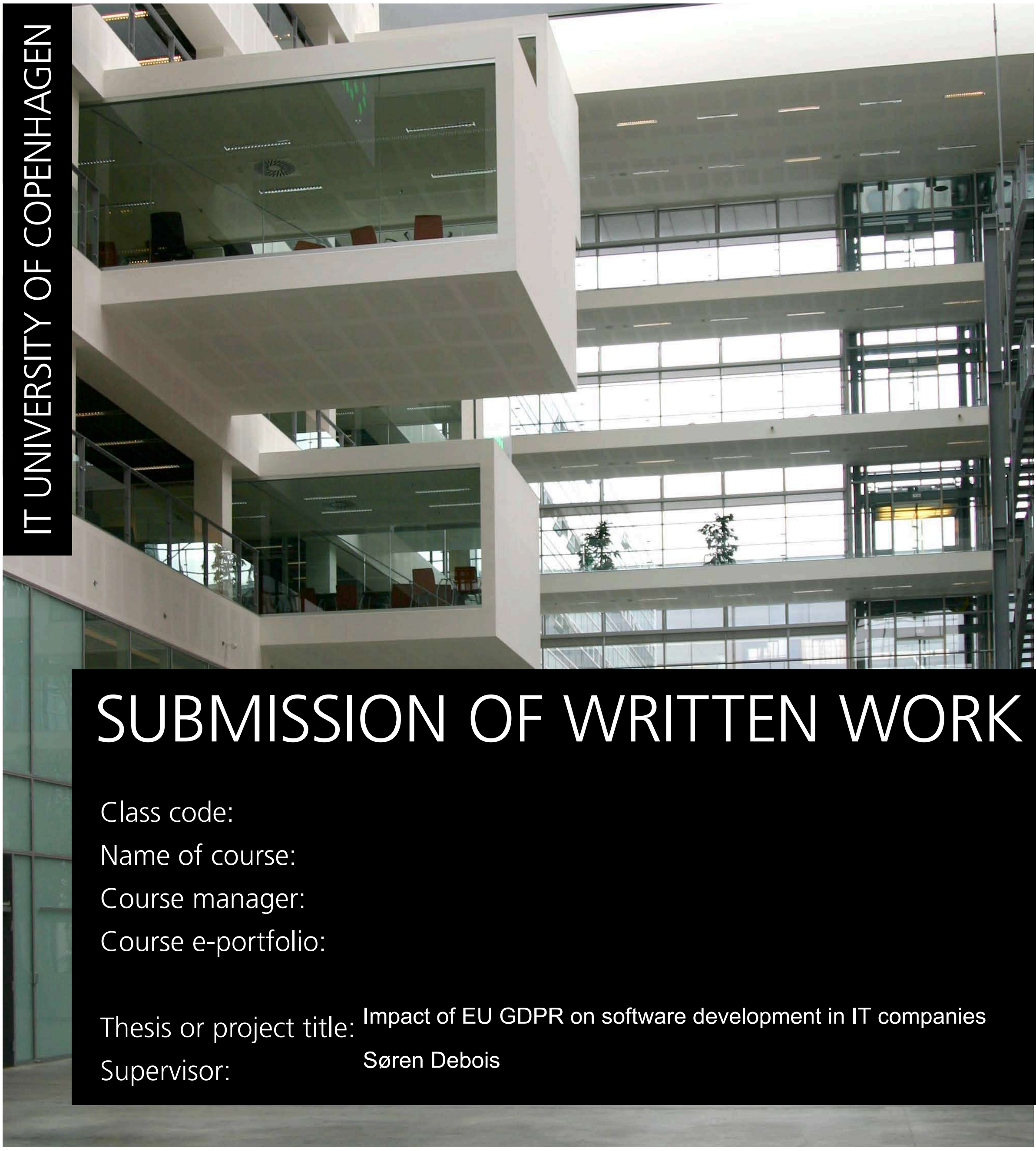
IT UNIVERSITY OF COPENHAGEN



SUBMISSION OF WRITTEN WORK

Class code:
Name of course:
Course manager:
Course e-portfolio:

Thesis or project title: Impact of EU GDPR on software development in IT companies
Supervisor: Søren Debois

	Full Name:	Birthdate (dd/mm-yyyy):	E-mail:
1.	Thor Valentin Aakjær Nielsen Olesen	14/02-1995	tvao@itu.dk
2.	Dennis Thinh Tan Nguyen	01/04-1993	dttn@itu.dk
3.			@itu.dk
4.			@itu.dk
5.			@itu.dk
6.			@itu.dk
7.			@itu.dk

Impact of EU GDPR Laws on software development in European IT companies

Thor V.A.N. Olesen & Dennis Thinh Tan Nguyen

Contents

1 Introduction

The General Data Protection Regulation (GDPR) is a regulation issued and ratified by the European Commission in 2016 to be complied with by international companies in May 2018 [6].

The regulation is comprised of a set of data protection rules that intend to strengthen and unify data protection for individuals within the European Union (EU) as well as outside EU. The objectives of the EU GDPR is to give citizens back the control of their personal data through the establishment and unification of globally accepted industry standards for data security and compliance within the EU.

Further, the GDPR has introduced additional duties that organizations are to carry out, which may change how an organization is structured as well as how business processes are conducted. By way of example, the GDPR requires that large companies appoint a data protection officer whose main responsibility is to advise and ensure that the business complies with the laws outlined in the GDPR [12, p.55].

Also, the GDPR require companies that directly process personal data to determine the purpose and means of processing this personal data [12, p.33]. Thus, they become responsible for ensuring that any business process that handles personal data is compliant according to the GDPR. In the pursuit of compliance, such responsibilities may prove to be challenging for existing companies where present personal data is intertwined and unmonitored.

This is an important matter because of the potential infringement of personal data and privacy in today's digital era where data is increasingly exposed. Ultimately, the GDPR can penalize companies up to 20M Euros, or 4% of their annual revenue for breaches where the companies do not comply with the data protection rules.

In this regard, a crucial matter is to help IT companies understand the legal requirements outlaid in the legislation, identify data breaches in their software and ultimately assist them in creating software that complies with the GDPR in the future.

1.1 Problem Definition

Altogether, the EU GDPR objectives are threefold in that it strives to unify the data protection legislation across the European nations, give back citizens control over their personal data as a universal human right, and finally impose more severe sanctions on companies that do not comply as a way to proactively safeguard personal data belonging to European individuals.

From a business perspective, the rules help them better align with the current technologies (i.e. the Internet and Big Data) and ”free” consumer services that increasingly process personal data of individuals, sometimes without their awareness or consent. In this regard, the EU GDPR will impact the data protection requirements across all areas of companies including their staff, work processes, products, documentation and so forth.

In this regard, this paper aims to look at how IT companies may cope with the EU GDPR. Specifically, it strives to provide software developers in IT companies with a better overview of the EU GDPR from a technical point of perspective and showcase what technical requirements exist, and how they may be implemented as part of the software development. As a result, no effort will be made on elaborating the ”managerial” and organizational aspects of the legislation including work processes, documentation, and internal compliance. Based on the above focus area, the following research question has been defined (see next page):

Research Question: which changes will the EU General Data Protection Regulation bring and how may IT companies cope with it when developing software?

Sub-questions to answer the main question:

- 1) What is the GDPR and how does it affect business practice in IT companies?
- 2) Which considerations should be taken into account when developing software that has to comply with the rules outlined in the GDPR?
- 3) How can one implement an IT system that may cope with the data protection rules described in the GDPR?
- 4) Does the GDPR involve any potential security breaches that allow companies to bypass security measures like confidentiality of data, integrity of data and availability of data?

1.2 Report Structure

This section provides an overview of the report and the topics that each chapter addresses.

Chapter 1 (Introduction)
Provides an introduction and definition of the problem that is investigated in the project about the EU GDPR.

Chapter 2 (Background)
Describes the background of the EU GDPR initiative. Further, the chapter puts light on some of the ways, in which business practice will be affected.

Chapter 3 (Theory)
Describes some of the security goals and security principles as outlined in the Security Course at ITU. These will be used for comparison with the EU GDPR requirements in Chapter 6 (Discussion: see section 8) to see what goals and principles the EU GDPR addresses and which ones it omits.

Chapter 4 (Proposed System)
Describes the domain requirements of the sample fitness web application that is developed throughout this project to showcase technical GDPR requirements.

Chapter 5 (Analysis)
Describes the technical requirements that may be derived from the EU GDPR, when trying to develop compliant software. This will be used as a springboard for the sample web application that has been developed in the project.

Chapter 6 (Implementation)
Showcases how the EU GDPR requirements have been incorporated in the sample web application developed in the project. This is based on the analysis and an ongoing experimental approach of interpreting and translating the legal requirements into technical terms.

Chapter 7 (Sample Application Compliance Review)
Evaluates whether the sample application developed in this project is compliant with the technical requirements derived from the EU GDPR.

Chapter 8 (Discussion)
Discusses any gray areas and problems in the EU GDPR that were experienced during the development of the sample web application. Further, it evaluates how the EU GDPR addresses the security goals and principles described in the Theory section 3.

Chapter 9 (Conclusion)
The conclusion sums up the findings of the project and describes the main points that have been derived from the analysis of the EU GDPR on how to create compliant software.

2 Background of the EU GDPR initiative

Before the introduction of the new data protection reform, legislation on data protection was already in place since 1995 [5]. The legislation provided general rules that guaranteed the fundamental rights related to data protection and were to be interpreted and implemented by each member of the European Union.

Consequently, this has lead to different implementations of the legislation, each based on their own complex and inconsistent interpretation of the laws. This legal uncertainty increased administrative costs and introduced different legal complications, which could potentially be exploited and affect the trust of individuals negatively including the competitiveness of businesses within the EU.

Also, the current legislation was introduced during a time where many contemporary technologies did not exist. The creation of cloud systems, social networks, location-based services, big data and so forth has increased the amount of personal data that is collected and processed, which has caused new implications that did not exist back in 1995.

Arguably, this may render the current legislation out of date and has created a demand of a reformation of the data protection legislation, in order to protect the people's right to personal data protection. As a result, the proposal of a General Data Protection Regulation was introduced in January 2012, ratified to take action on the 25th of May, 2018 [4].

2.1 Principles of the General Data Protection Regulation

In order to better understand what is required by companies according to the new legislation, one may want to analyze the five fundamental rights that it strives to provide individual EU citizens, shown below. The GDPR strives to provide these rights to the individual EU citizen that will give them more control over their personal data and make it easier for them to access their personal data. In addition, the reform is designed to guarantee that the personal data of an individual is protected regardless of its location and where it is being processed.

- **Right to be forgotten** [12, Article 17 - p. 43]
- **Right of access by the data subject** [12, Article 15 - p. 43]
- **Right to know when data has been compromised**[12, Article 17 - p. 43]
- **Right to data portability** [12, Article 34 - p. 52]
- **Right to restriction of data processing** [12, Article 18 - p. 44]

The right to be forgotten
The goal of this right is to protect the privacy of an individual. Thus, upon request, if a person does not wish his personal data to be processed or stored, all his or her personal data must be deleted if there are no legitimate reasons to retain it.

The right of access by the data subject
The goal of this right is to make it easier for the individuals to access their personal data. Besides easier accessibility of personal data, the individuals will also be provided more information about how their data is processed in a clear and understandable way. By way of example, if a service is provided for children, then the information must be presented in a way, so that even a child understands how their personal data is processed.

The right to know when one's data has been compromised
The goal of this right is to let the user know, in time, when their data has potentially been compromised. In the event of data breaches, companies and organizations must notify a national supervisory authority, so that the individuals who use the system may take timely and appropriate measures to reduce the risk of their personal data being compromised any further.

The right to data portability
The goal of this right is to let the user maintain free will of choice and remain independent of a given service or provider. Upon request, a user can retrieve all of his or her data from a service in a portable format that other providers or services may consume. This makes it easier for an individual to transmit their data between providers and services and helps stimulate transparency across different providers.

Data protection by Design and Default
The reform has also defined a rule named 'Data Protection by Design and by Default'. This requires appropriate data protection safeguards to be implemented in products from the earliest stage of development and throughout the whole development lifecycle, also known as 'Privacy By Design'. Further, privacy protection measures should be enabled by default (i.e. Privacy By Default). Thus, it has introduced an additional complexity, which must be taken into consideration, when designing and implementing a piece of software, service or system.

Compliance and consequences
With the introduction of the new regulation, a stronger enforcement of rules has also been included. If companies are not in compliance with the regulation, data protection authorities will be able to fine the companies based on the severity of the incident. Severe noncompliance with GDPR may potentially result in fines up to 4 percent of an annual turnover [12, Article 83 - p. 83], which may severely incapacitate companies if not being compliant.

3 Security Theory

The following section highlights the traditional security goals and principles described in the book: Applied Information Security [8]. The book presents 12 overall security principles to understand what security is and how it can be measured. The theoretical principles will be used to evaluate, which security measures the GDPR addresses and help clarify which security measures it does not address.

3.1 What is security?

To understand what the legislation strives to protect, one must first agree on what ”security” is. Arguably, security is the art of making things not possible. In this case, we are seeking to protect personal data within computers from any threats.

As a software developer, we are developing a piece of software based on the assumptions on how the system should work. In this regard, security becomes highly relevant because it considers all assumptions that have been made, which may potentially be exploited by an adversary.

Conceptually, an adversary is a malicious entity whose aim is to prevent users of a system from achieving their goal (primarily privacy, integrity, and availability of data) [25]. In this regard, a software developer has to protect the software against all possible attacks, whereas, an adversary may only need to find one flaw that works.

Based on this, the goal of security is not to achieve perfect security but rather to establish how many resources is required from an adversary to break into a system. For this project, the security of the software is measured based on the requirements outlaid in the EU GDPR rules but also based on how well it complies with traditional security goals and principles.

3.2 Security Goals

Overall, one may address four security goals when measuring the security of a system [8, chapter 1], where security measures try to address at least one of them:

- Confidentiality: prevent unauthorized access to information
- Integrity: prevent unauthorized altering of information
- Availability: authorized users should have access to the system
- Accountability: actions of a principal may be traced back uniquely

These security goals act as a guide when securing information within a piece of software and will thus be described in detail.

3.2.1 Confidentiality

The security goal of confidentiality strives to limit the disclosure[1] and access of information to authorized users only and aims to prevent unauthorized users from access. In general, one may in part adhere to this goal by using encryption and access controls. Encryption of data ensures that data is transmitted in a form that can be understood by authorized people only. Access controls are used to set standards and policies when accessing information and resources in a system. For this project, one might use the HTTPS protocol over HTTP to encrypt data and avoid eavesdropping[2] between the client and web application and implement certain authorization features across the web application, dictating access to different functionalities.

3.2.2 Integrity

The security goal of integrity strives to prevent unauthorized altering of information and thus maintain the integrity of data. For example, an adversary may try to masquerade or impersonate users with a fake identity to gain unauthorized access to information or tamper with messages on the network. In this case, the goal is to maintain data consistent, accurate and trustworthy. By way of example, this may be done by hashing passwords (supplied by a random salt to avoid brute attacks) of users in the system, ultimately making it very hard for the adversary to invert them. Also, one may use digital signatures and certificates to verify that a user is who he or she claims to be.

3.2.3 Availability

The security goal of availability strives to keep the access to the system open for authorized users (at any time). Thus, this security measure may be subject to DDoS (Distributed Denial of Service) attacks where the system is rendered unavailable because of too many clients trying to connect to the system. A way of approaching this goal may be to employ redundancy and multiple defense layers in the system. The notion of redundancy is mainly based on keeping the system up and running even in the absence of an important component. For example, one might have multiple servers instead of one server (i.e. single point of failure) to host the system. Further, the defense in depth concept tries to place multiple layers of security controls in the IT system to provide redundancy in the event of a failure or when a vulnerability is exploited [26].

[1]a secret that is made known

[2]eavesdropping or network sniffing is an attack where the adversary aims to capture information transmitted over the network

3.2.4 Accountability

The goal of accountability strives to trace actions of a principal uniquely back to that principal. In other words, actions in the system should be traceable to the user who performed the action. One way of doing this may be to log actions and events in the system. This becomes necessary if a user is compromised or if the data of a user is changed unwillingly. For example, someone might change your bank balance, and it is vital to trace how, when and by who it happened.

3.3 Security Principles

The following 12 security principles correspond to the good rule of thumbs when trying to construct ”secure” IT systems that adhere to the four security goals:

- **1 Simplicity**: simple systems contain fewer flaws than complex systems
- **2 Open Design**: the security of a system should not rely on the secrecy of protection mechanisms
- **3 Compartmentalization**: organize resources into isolated groups of similar needs to avoid compromising the whole system upon failures
- **4 Minimum exposure**: minimize the attack surface to the adversary of a system by e.g. disabling unnecessary system components and services
- **5 Least privilege**: any component should operate using the least set of privileges required for a particular task to avoid compromising the rest of the system
- **6 Minimum trust and maximum trustworthiness**: minimize trust and turn assumptions into validated properties to avoid assumption abuse
- **7 Secure fail-safe defaults**: the system should start and return to a secure state in the event of a failure to minimize downtime and increase availability
- **8 Complete mediation**: access to any object must be monitored and controlled to enforce access control and avoid privilege escalation
- **9 No single point of failure**: build redundant security mechanisms whenever feasible to establish defense in depth and maintain system availability
- **10 Traceability**: log security-relevant system events to verify actions and help pinpoint a potential adversary
- **11 Generating Secrets**: maximize the entropy (order) of secrets to keep them sufficiently random and hard enough to guess for an adversary
- **12 Usability**: design security mechanisms that are easy to use to avoid users from circumventing them

Altogether, these security goals and principles will become highly relevant when measuring the security of the web application in this project and evaluate the security measurements required in the EU GDPR legislation. Ultimately, this allows us to determine which security measures that the EU GDPR addresses and neglects in the discussion section 8.

4 Proposed System Requirements

This section defines the requirements of the sample application that it must support to satisfy the domain features. In particular, the sections considers the user stories, functional and non-functional requirements related to the domain. The requirements in the EU GDPR are addressed in the Analysis (see section 5).

4.1 Scope of the system

The scope is based on functionality that features some of the scenarios where personal data is processed and the GDPR becomes relevant. The fitness application shall provide people that exercise with functionality to help them keep track of their training progress in the gym. The application will allow individuals to register their progress (i.e. repetitions and weight) for a predefined set of exercises. Further, the application will allow individuals to register a personal account that stores their personal information and workout progress. Finally, users should be able to compare and share their results with each other, opening up for scenarios where data is exchanged and security measures become relevant.

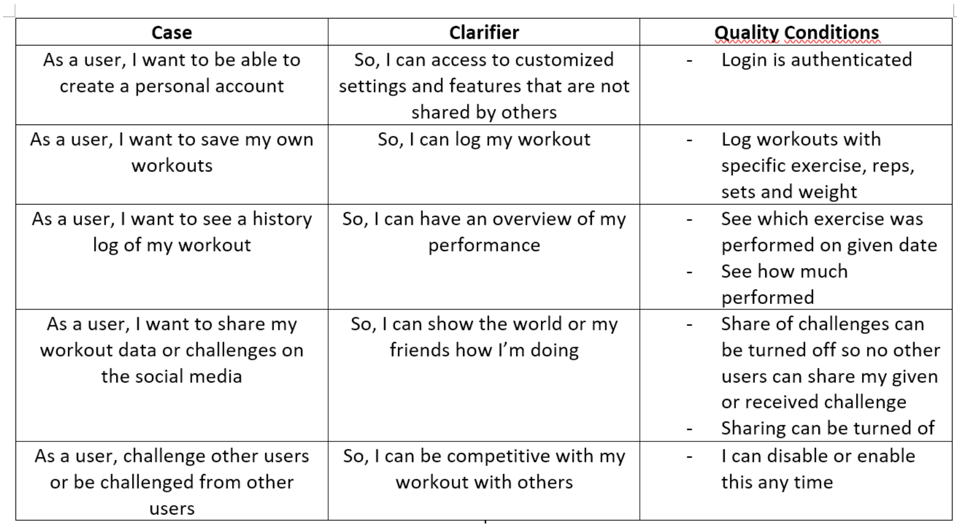
4.2 Purpose of the system

The purpose of the system is to showcase some of the core features required in the EU GDPR legislation and help provide a sample application that may be used by other software developers as a reference point, when trying to implement IT systems that should comply with the EU GDPR rules.

Thus, an application within the domain of fitness will be developed where personal data, influenced by the EU GDPR, is present. Specifically, users should be able to create an account based on personal data, while also being able to exchange and share data with other users on the application. In this way, the application will help illustrate some of the important decisions that have to be made as a developer when implementing IT systems affected by the EU GDPR due to the presence of personal data.

4.3 User Stories

A set of core user stories have been defined to address the needs of the user and the functionality that will satisfy them as a minimum.



Case	Clarifier	Quality Conditions
As a user, I want to be able to create a personal account	So, I can access to customized settings and features that are not shared by others	- Login is authenticated
As a user, I want to save my own workouts	So, I can log my workout	- Log workouts with specific exercise, reps, sets and weight
As a user, I want to see a history log of my workout	So, I can have an overview of my performance	- See which exercise was performed on given date - See how much performed
As a user, I want to share my workout data or challenges on the social media	So, I can show the world or my friends how I'm doing	- Share of challenges can be turned off so no other users can share my given or received challenge - Sharing can be turned of
As a user, challenge other users or be challenged from other users	So, I can be competitive with my workout with others	- I can disable or enable this any time

Figure 1: Core user stories

These user stories have been chosen based on some of the features within the fitness domain that a fitness enthusiast might want to use during workouts. Also, these user stories constitute user scenarios where personal data is collected, processed and may be exchanged between users. The presence of personal data becomes highly relevant when trying to comply with the GDPR from a software developer's point of view. Consequently, the user stories may help showcase examples on how one can comply with some of the core requirements in the EU GDPR, which are analyzed in the analysis section 5.

4.4 Requirements

Based on the core requirements listed in the table (Figure: 1), the following functional requirements and non-functional requirements have been derived to be implemented in the sample application[3].

Functional Requirements

- The system must be able to perform CRUD operations on relevant entities such as users, exercises, workouts and challenges.
- The system must contain a predefined set of exercises, which a user can look up and use when he logs data.
- The system must be able to store information related to user progress based on weight and reps of a given exercise in a database.
- The system must be able to create relevant relations in a database between user entities and user data so data sharing capabilities are made possible
- The system must be able to let users share their workout data with other users or social media.
- The system must be able to authenticate a claimed user before giving access to its features and components.
- The system must be able to validate if a user is authorized to perform a given interaction in the system.
- The system must validate and sanitize any form of user input to make it resistant to malicious input
- The system must create distinct LOGS on any activity that involves data interaction/manipulation and system failures.

[3]Note that the technical GDPR requirements are analyzed in the analysis section 5.1

Non-Functional Requirements

- **Availability** - The system and its features must strive to be available to the user at all times if no server maintenance is undertaken. In the event of failure, the users must be notified with a reason of the downtime within a single day.

- **Accessibility** - The system should be accessible as a web-application in a browser. No assistance-devices, motion trackers or other fitness hardware devices will be considered. Also, the application does not consider the availability for disabled individuals, which is out of the scope of the proposed sample web application.

- **Regulations** - This application will primarily strive to comply with the EU GDPR proposed by the European Union. Other regulations will not be considered, unless stated by other stakeholders such as the supervisor.

- **Backup** - The system should have backup procedures in the event of system failures. This is assumed to be regulated by a third-party backup provider due to time constraints.

- **Extensibility** - The system should have a modular design that supports the extension of additional features that may be added after the development of bare-bones version of the sample application.

- **Fault-tolerance** - The system must be resistant to malicious input or system failures. Thus, the system must be able to regain a fail-safe state of default and resume normal operation in the event of failures.

- **Interoperability** - The system must be able to run cross-platform on Windows, Mac, and Linux. The application should support popular browsers like Chrome, Edge, Firefox and Internet Explorer.

- **Usability** - The system must be easy to use, easy to learn and work without delay when used by the target group defined as fitness people. They may want to use the system during a workout. Thus, the application must be intuitive and efficient for its purpose of logging workout results. Note that the usability considerations have been deemed less important due to the purpose of the application.

5 Analysis

This section aims to give an overview of the requirements in the EU GDPR when trying to develop compliant software. These requirements are separate from the domain requirements in the previous chapter and are defined to assure that the proposed software will be developed in compliance with GDPR. The section will try to analyze and derive the technical requirements in the EU GDPR. Further, possible solutions to them will be described for use in the sample fitness web application. The Implementation section (see 6) will show how to implement some of the core requirements by using the ASP .NET Core technology stack.

5.1 EU GDPR Requirements

The EU GDPR contains a range of articles that describe how an organization and their internal work processes should comply with the EU GDPR. However, these have not been deemed important for the sake of this project that aims to provide help for software developers only. In this regard, the following chapters and articles in the GDPR were deemed to be relevant as a software developer:

Chapter 1 - General Provisions
In particular Article 4, which defines various terms such as personal data, data processing and pseudonymization. The definitions provide important knowledge to developers, since they help define the data flow within the system, as well as how data should be processed. Misinterpreting what the definitions constitute, may result in data being incorrectly processed or exposed in a way that lead to a non-compliant system.

Chapter 2 - Principles and Conditions
This chapter and its articles are relevant to a software developer because it describes the various conditions of user consent and how they are requested. As a front-end developer, one must be able to design a user interface that presents what data is collected, how it is being used as well as how consent is requested. On the other hand, a backend developer should support mechanisms in the underlying logic to process consent conditions and authorize consent requests.

Failing to understand these conditions, may result in a user interface that does not satisfy the requirements on how consent should be requested by the user, or a backend that fails to validate the consent requirements within any relevant system components.

Chapter 3 - The registered user's rights
This chapter and its articles are relevant to a software developer because they describe the rights of a European citizen that is part of the European Union. By way of example, a user has the right to be forgotten (i.e. deleted). A developer must, therefore, consider this requirement within the system and implement the necessary logic needed to address the right of the user. Failing to understand these rights and conditions may result in software that does not support the rights of a user and ultimately lead to software that is not in compliance with GDPR.

Chapter 4 - Data Controller officer and data process officer duties
In particular, Article 25 proposes mechanisms of data protection by design and by default. Further, Article 32 proposes the conditions related to security during data treatment and processing. As a developer, these articles are relevant as they create additional steps to be addressed during the software development life cycle, which were previously not required to address. By way of example, a developer must now consider how personal data is protected from the beginning of the development cycle instead of during the later stages. This may affect how current software methodologies are applied and how software requirements are defined during the very early stages of software development.

NB: the GDPR articles and sections do not always clearly define its requirements. Thus, some conditions stated in the GDPR will be subjectively interpreted when deriving suggestions on how to develop compliant software.

5.2 Important definitions

In this subsection, relevant definitions listed in Article 4 will be used to clarify the fundamental terms that a software developer must understand to correctly interpret the laws set forth in the GDPR and ultimately apply it to the implementation of GDPR compliant software.

Data Controller
A data controller is a legal person, authority or agency that determines the purpose and methods for processing personal data. The conditions of processing are based on the conditions outlined in the EU GDPR [12, p. 33].

Data Processor
A data processor is a legal person, authority or agency that processes personal data based on the data controller's conditions and requirements [12, p. 33].

Personal Data
"Personal data" means any information relating to an identified or identifiable natural person or "data subject"; an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person" [12, p.33].

By definition, all data that can be used to directly refer to an individual or combined with other partial data to refer to a person is regarded as personal data and must, therefore, be treated and processed based on the guidelines that the GDPR presents. The sample fitness web application will use a combination of direct identifiers constituted of data that directly describes a person such as a name, and indirect identifiers, that may indirectly link to a person (e.g. email).

Direct identifiers within the sample application

- Name
- Birth Date (Age)
- Sex
- Country

Indirect identifiers within the sample application

- Unique Identifiers of users such as Ids and GUIDs
- Display name (Username that others can see)
- Email
- Challenge (A challenge between two users are unique)

We assume that workouts and their respective entries do not refer to a user because they do not contain any personally identifiable information (i.e. PII). In extent to this, any other user may have created the same workout as well. Thus, workouts are not regarded as personal data.

Sensitive Data
The GDPR also introduces special categories of personal data. According to Article 9, 'sensitive' data may be classified as: *Processing of personal data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, or trade union membership, and the processing of genetic data, biometric data for the purpose of uniquely identifying a natural person, data concerning health or data concerning a natural person's sex life or sexual orientation shall be prohibited.* [12, p. 38]

As opposed to personal data, 'sensitive' data is not permitted of processing, unless special conditions are in place. For example, the data subject might have given consent and no national laws prohibit such consent from taking place [12, paragraph A - p. 38]. Additional conditions can be found in Article 9.

Issues and assumptions with personal data
Personal data may be presented in ambiguous ways that allow it to be interpreted as both direct or indirect personal data. For example, a username or an email may be classified as indirect, if it does not refer directly to the user as a person. Rather, it relates to a context that is generic, fictional or nonsensical, such as "coolUsername" or "mymaillive.com." However, the user may choose to provide data that exposes their name within an email address, e.g. "DennisNguyen@live.com". This is a scenario that one must consider and deal with as well.

Another problem is caused by the fact that a user may input fake personal data, such as a name that seems genuine but in reality does not correspond to the person's real name. Thus, the data, which seems otherwise genuine, does not refer to the person. Nonetheless, false data and analyzing the legality of such information is outside the scope of this project but it is still an important factor to have in mind, when one is defining what is regarded as personal data within a system.

For this project and the proposed sample application, it is assumed that emails and usernames are based on a context that do not directly refer to a person. That is, we assume that the user will not input data that may directly relate to them.

Data Processing
"'processing' means any operation or set of operations which is performed on personal data or on sets of personal data, whether or not by automated means, such as collection, recording, organisation, structuring, storage, adaptation or alteration, retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, restriction, erasure or destruction;"[12, p.33]

Thus, as a software developer, one must identify any features, components, modules or code that will access and process personal data. These should follow certain constraints set forth in the GDPR. Based on the sample application, personal data will be dealt with in the following areas.

Features that process personal data:

- CRUD operations on users (e.g. registration, login and logout)
- Challenge feature that utilizes data linked between two users
- Sharing feature that may transfer data to a third party (e.g. Facebook)

Pseudonymization

According to the EU GDPR: *"'pseudonymisation' means the processing of personal data in such a manner that the personal data can no longer be attributed to a specific data subject without the use of additional information, provided that such additional information is kept separately and is subject to technical and organisational measures to ensure that the personal data are not attributed to an identified or identifiable natural person;"*[12, p.33]

As a software developer, one must avoid using personal data such as a personal citizen number (CPR-number in Denmark) to identify a given user entity within the system. Thus, other identifiers should be employed in the system like GUIDs (i.e. globally unique identifier), integers or other data types used as a reference to a user. These references can be used to look up users and their data in the database and any other relevant data structures, without exposing their personal data unnecessarily.

This definition may seem trivial and can be considered to be more relevant in organizational processes, such as when staff members exchange reports by email, containing personal data. However, a software engineer must still be aware of how personal data and users are referred in the system. Thus, the sample application will not use any personal data to identify users. Instead, pseudonymized identifiers will be employed to refer to users across the different system layers.

Consent

According to the EU GDPR, *" 'consent' of the data subject means any freely given, specific, informed and unambiguous indication of the data subject's wishes by which he or she, by a statement or by a clear affirmative action, signifies agreement to the processing of personal data relating to him or her;"*[12, p.34]

Thus, as a software developer, one must design a consent model that may be applied to any areas where personal data is processed. Particularly, a software developer must consider how consent is validated server-side and a front-end developer must consider how it is presented client-side and ensure that the request of consent is presented in a clear, easy to understand and unambiguous way.

5.3 Privacy By Design and Privacy By Default

According to Article 25 in The EU GDPR (see [12, p. 48], companies need to employ 'data protection by design and by default'. This requires companies to explicitly recognize the concepts of 'privacy by design'[4] and 'privacy by default'[5]. As a software developer, this means security measures should be made throughout the software development lifecycle where personal data is processed (privacy by design). Further, they should only process the minimum personal data required for each purpose where data processing takes place (privacy by default).

Specifically, Article 25 in the EU GDPR first suggests that "*the risks of varying likelihood and severity for rights and freedoms of persons posed by the data processing...*" (Article 25, Paragraph 1: [12, p. 48]) should be considered. This could be done by documenting a risk analysis with a risk assessment matrix mapping potential threats to their likelihood and impact. However, this is not done in this project where the focus lies on the technical endeavors only.

Further, the article describes the idea of 'privacy by design' where the "*data controller shall implement appropriate technical and organisational measures such as pseudonymization ... to implement data-protection principles, such as data minimisation effectively ... to integrate the necessary safeguards into the processing in order to meet the requirements of this Regulation and protect the rights of data subjects*" (Article 25, Paragraph 1: [12, p. 48]).

Finally, the concept of "privacy by default" is expressed: "*the controller shall implement appropriate technical ... measures for ensuring that, by default, only personal data which are necessary for each specific purpose of the processing are processed*" (Article 25, Paragraph 2: [12, p. 48]).

These requirements may be deemed somewhat vague, ambiguous and broad in their interpretation. In the project, the following questions have stirred up confusion as in what to do as a software developer.

Firstly, does the responsibility of 'privacy by default' belong to the developer? Further, does it even make sense to pseudonymize personally identifiable information (i.e. PII) when additional information may be used to identify the data subject anyway? In practice, numerous examples proof that individuals may be re-identified from anonymized data by using additional information[28].

[4]Software Engineering where privacy is taken into account throughout the whole process
[5]Only the necessary personal data for each specific purpose of processing is processed

Also, if the data subject has control over their personal data and may delete it at any time, how do you make their actions traceable through logging mechanisms that rely on some form of identification? Further, do you have to explicitly request consent to the processing of all personal data? Arguably, the application may be rendered useless if it does not have access to certain data. These are just some of the questions that may lead to confusion amongst developers.

5.4 Data Processing Principles

The following paragraphs will try to elaborate on some of the previous questions and the technical safeguards that may be employed in the processing of personal data. The two concepts of ’privacy by design’ and ’privacy by default’ employ the 6 following data processing principles when embedding privacy into a system:

- **1) Purpose Limitation**: only required data is used for each purpose of data processing
- **2) Data Minimization**: no more data than necessary for a purpose is processed
- **3) Data Accuracy**: data that is processed is kept up to date, and deprecated data is either updated or completely removed
- **4) Storage Limitation**: personal data that may identify a data subject should only be stored for a limited time
- **5) Data Integrity and Confidentiality**: personal data is protected against unauthorized access and unlawful alteration
- **6) Lawful Processing**: processing of personal data may be classified as lawful (i.e. compliant) if the data subject has given consent for this processing

Moreover, section 2 (i.e. Security of personal data) and Article 32 (i.e. Security of processing) in the GDPR (see [12, p. 52]) includes the following security measures to ensure the protection of personal data:

- **1) Pseudonymization and Encryption of Personal Data**
- **2) Ensure confidentiality, integrity, availability and resilience of system**
- **3) Restore the availability and access to personal data in the event of a physical or technical incident**
- **4) A process for evaluating the effectiveness of technical measures**

NB: The last bullet point (4) is not addressed in the project but penetration testing could be a process used to measure the security of the system, by trying to exploit vulnerabilities.

Data Processing Suggestions for the Application

Based on the described principles and requirements, the following technical requirements have been derived to protect personal data:

- 1) Processing of personal data should only have one purpose
- 2) The minimum required data should be used to fulfill any given purpose
- 3) Personal data should be kept up to date and accurate
- 4) Personal data should not be not stored longer than needed
- 5) The communication of personal data should be encrypted
- 6) Passwords leading to personal information should be hashed
- 7) Pseudonyms (i.e. artificial identifiers) should be used instead of PII (i.e. personally identifiable information) to identify the data subject
- 8) User input should be validated and error pages should be used for invalid requests (resilience)
- 9) Multiple servers should be used to host the application (availability through redundancy)
- 10) A backup service should be used to protect data integrity

The above suggestions have been proposed as good practices to be followed when processing personal data and striving to embed 'privacy by design' into system. Needless to say, this is a suggestion and not a final answer on how to integrate 'privacy by design' and 'privacy by default' into the software development process. The suggestions have simply been evaluated in this project to be tangible ways of addressing the processing of personal data in software.

The following sections will describe how the different data processing principles may be addressed in the sample fitness web application.

Data Minimization Suggestions

The web application in the project should only use the absolute minimum data required each place where personal data is processed, and the purpose of each type of data processing should be clearly separated. For example, upon registration of an account, only the email and password should be registered to authenticate the data subject. Thus, any optional data will be omitted to better adhere to the data minimization principle. Instead, optional data may be set in the account management settings, if the user wishes to unlock other features (e.g. access to the workout and challenge page).

Purpose Limitation Suggestions
Further, every time personal data is processed with a new purpose, consent from the data subject should be requested by the system for the new purpose of data processing. For example, if the data subject wishes to use the workout functionality, he or she must first give consent to allow the processing of workout specific data and then provide that data.

Data Accuracy Suggestions
Based on this principle, all personal data related to the data subject should be held modifiable at any times for correction. To support this, input validation should be used to keep data provided by the data subject accurate and free from errors. For example, regular expressions may be used to validate the email input upon registration/login, and numerical ranges may be used to determine, if the user provides valid workout data (e.g. by not providing negative numbers when adding an exercise with a given weight, repetitions, and sets).

Also, the HTML forms used by the data subject to provide data may employ a unique cookie value, that is validated when posting data to the server. This is to prevent cross-site request forgeries where hidden content from another malicious site is added to the server. Further, the personal data stored by the data subject should be made easily accessible in the account management page. In that way, the data subject may easily correct and update his or her personal data to keep it accurate.

Storage Limitation Suggestions
The personal data stored in the web application is required to be "*kept in a form which permits identification of data subjects for no longer than is necessary for the purposes for which the personal data are processed*". By way of example, the personal login data should only be stored to authenticate the data subject in the application.

Further, the domain-specific workout and challenge data should only be kept, if the data subject is actively using these features. Otherwise, the data subject should immediately be able to request the removal of this data in account settings. The personal data is only kept in an identifiable form during temporary data transfers between the system components in the application (e.g. DTO[6]).

[6]Data transfer object used to transfer information between system layers

Data Integrity and Confidentiality Suggestions
According to the GDPR, the application must use appropriate security safeguards like encryption and pseudonymization to help protect the confidentiality and integrity of personal data. Thus, all communication of personal data between the data subject and web application should be encrypted, which may be done using HTTPS as a default protocol for handling HTTP requests. Moreover, all HTTP requests may redirected to the HTTPS port for safe communication.

Further, the web application should use ’pseudonymization’, by using a unique identifier to identify the data subject instead of using personally identifiable information (i.e. PII). In this regard, one of the challenges may be to keep all personal information pseudonymizable, which is covered in the discussion (see section 8.

Moreover, the password of the data subject may be hashed with a random salt in the database to avoid a potential adversary from compromising the password. Otherwise, if the password is compromised in its plaintext form, one may ultimately get access to all of the data subject’s data, breaching with the requirement of data confidentiality. For this purpose, it might be suggested to use a password that is sufficiently random and complex for an adversary to guess.

Most importantly, a backup solution may help protect against data loss but has been omitted, since there are numerous backup providers with better options. As a developer though, one may ensure that the system validates input to prevent inaccurate data entries from happening and keep data accurate and consistent.

Availability and Resilience Suggestions
Amongst others, these security goals may be protected through backups, input validation and hosting on multiple servers. Backups allow the restoration of data in case of failure. Input validation handles erroneous user input to avoid the system from crashing or storing inaccurate data. Hosting the application on multiple server prevents having a single point of failure in the event of a natural disaster or denial of service attack. As a developer though, one should focus on input validation. For this purpose, invalid requests should make the system display an error message or redirect to an error page, explaining what went wrong. As a result, the system encourages accuracy and remains in a fail-safe default without crashing.

Another good practice may be to separate the system architecture into multiple layers. Consequently, if one layer is compromised, it does not necessarily affect the remaining system components. For example, the sample application will have a dedicated layer for the database, business logic, and web server.

5.5 Consent

According to Article 7 (i.e. Conditions for consent) in the EU GDPR (see [12, p.37]), "*the controller shall be able to demonstrate that the data subject has consented to the processing of his or her personal data*". Further, "*the request for consent should be presented in a clearly distinguishable form, using clear and plain language*". To address this requirement, one must comply with the conditions described below.

Consent Conditions

According to Article 7 in the EU GDPR (see [12, p.37]), the following four conditions are required to demonstrate consent:

- 1) Where processing is based on consent, the system should be able to proof that the data subject has given consent to the processing of his or her personal data
- 2) Request for consent and other policies must be clearly separated and distinguishable in a readily available form, using clear and plain language.
- 3) The data subject shall have the right to withdraw his or her consent at any time
- 4) Consent must be easy to give and withdraw

As a result, it is no longer possible to combine the terms of agreements with the consent, which are two different policies. Further, the user should be able to withdraw his or her consent at any time. Consequently, as a software developer, one will have to address the design implications that arise from this new requirement.

Consent Design

The design of the consent model is mostly bound to front-end developers that need to provide a client-side user interface (UI) where the user may easily understand and provide consent. Further, it must be clear how the user withdraws consent, which could be done by toggling a check-box between on and off.

Thus, the usability aspect becomes highly relevant when designing consent into the application and may play a crucial role to ensure that consent is easily accessible and understood. Further, as a backend developer, the consent must be validated in certain system components by features that require consent.

Arguably, the user should not have access to data processing features if he has not given consent for this particular processing, which can be perceived as a matter of authorization (i.e. having access to pages with data processing based on consent).

Access Control Approach

From a technical point of perspective, one way of addressing the consent requirement, may be to interpret it as an access control problem. By definition, access control is a security mechanism used to restrict access to a place or resource. In the context of the sample web application, access to a particular page or resource may be given based on a set of permissions (i.e. authorization).

In terms of consent, that means the application should only give access to pages with data processing functionality, if the data subject has given his or her consent for that particular processing of data. By way of example, the workout page may require some sort of ”workout” consent (i.e. workout access policy) of the data subject to be provided before authorizing access to the workout page.

This approach has been used to solve the consent conditions described according to Article 7 in the EU GDPR legislation [12, p. 37]. In simple terms, the processing of data in the application requires the consent of the user. Thus, the sample application may only give access to data-processing features, if the user has already given consent for this data to be processed. For example, the workout page may only be available if the user has given consent to let the application process his or her workout data. Please consult the authorization section in implementation section for more details on the actual implementation of this: 6.2).

This approach was adopted because, without it, the application could potentially end up processing personal data belonging to the data subject without their consent, and end up not complying with the EU GDPR consent conditions. Thus, it has been deemed fit to only give access to data processing components if consent for this processing has already been given. In addition, it helps create an incentive for the data subject to actually use the web application and provide valuable data thay may be used by the business behind the application.

Consent Suggestions
This section will present a range of technical suggestions used to address the conditions of consent.

Access Control through Claims-based Authorization
As described previously, the requirement of consent may be solved as an access control problem. In practice, one way of employing this may be through the use of claims-based access control (i.e. authorization). In this context, a claim is a name-value pair that represents what the subject is (i.e. not what the subject can do). For example, your driver's license is issued by a local driving license authority and has your date of birth on it. Thus, this claim name is the date of birth, and the claim value is the actual date of birth, e.g. 10th February 1942.

In claim-based authorization, the application checks the value of a claim and allows access to a resource based on that value. For example, the application may only provide access to the account settings if the user is authenticated (i.e. has a claim confirming that the user is authenticated) and the account has been confirmed by email (i.e. has a claim that the email is confirmed and valid).

Another example of this in the sample application might be that the user is requested consent by the application to process his or her workout data in order to use the workout feature. If he or she consents, a workout claim associated with that particular user is issued by the application. By having this claim, the user now complies with a policy giving access to the workout page. In this regard, the user may be assigned one or more claims issued by some kind of trusted party [3].

Consent Settings
Upon registration, the data subject should be prompted to provide consent to different data processing functionality in the application. This may be done through a series of check boxes, that help the data subject provide consent to the processing of his or her workout, challenge or sharing data respectively. Note, that it is optional to give this consent and one should be able omit all of them. In this case, the consent and data processing features should be disabled by default (i.e. privacy by default).

Arguably, this may render the application useless, since the data subject would not be able to access any of the data processing pages. However, this has been deemed a good compromise between giving the data subject more control, while also taking into account the business interest in having the data subject actually use the application and provide data.

As mentioned previously, checkboxes may be used to request consent for the processing of personal data cross different data processing features in the application. In addition, one may display the purpose of the consent request and data processing (i.e. what data is used for what purpose) in a pop-up dialog when the user checks a given checkbox to provide consent. Again, this may be deemed a good solution to adhere to the consent requirements, without compromising the usability of the application. Alternatively, one might use one large matrix to map all personally identifiable information (i.e. PII) and request consent based on each identifier.

However, this has been deemed far too impractical as it would potentially end up hindering the data subject from even wanting to use the application. Most importantly, this might prevent the data subject from reading through the description of the consent request, which is required to establish a ”meaningful” consent that complies with the GDPR. The consent check box settings is a way of grouping consent in logical entities that each cover the processing of data related to a particular functionality on a given page.

Also, the GDPR states that consent *could include ticking a box when visiting an internet website and silence, pre-ticked boxes or inactivity should not constitute consent* ([p.6][12]). Thus, the check box consent is disabled by default and require a clear action by the user that needs to affirm the requested consent in a pop-up dialog.

Consent Pop-up Dialog

When the data subject provides consent, he or she should be presented with a dialog pop-up, clearly explaining what data is processed and for what purpose. The description of the consent request and its implications should be described in a simple, clear and easy-to-understand language.

Terms of Agreement

As stated in the conditions, the consent policy should not be intertwined with other policies when requesting approval of the data subject. For example, upon registration, the request for consent and terms of agreements should be separated from each other completely. Thus, the data subject should address the request for consent and the terms of agreement, separately.

Consent Withdrawal Pop-up Dialog
If the data subject, wants to withdraw his or her consent, this should be possible at any time. This may be enabled by letting the user change his or her choices of consent inside something corresponding to his or her account management settings. The user might access his or her settings page and disable his or her ”workout” consent by unchecking a checkbox. In this case, a dialog pop-up should be presented and prompt the user to confirm the withdrawal, explaining what will happen from this action of withdrawal in a clear and understandable way.

Further, the data subject might have the possibility of deleting all data that has been processed so far, as part of that particular functionality (e.g. the workout feature). From a company perspective, this might not be appropriate if the data is somewhat valuable and opens up for other business opportunities. However, this has been deemed necessary in order to comply with the **Storage Limitation** data processing principles that states personal data should only be kept as long as it is used for a specific purpose. In this case, the workout data becomes obsolete, since the data subject will no longer be using the workout page - and should thus be deleted.

Arguably, this might have been done differently. For example, many companies anonymize their data so that it may be used for other business opportunities (i.e. big data). In this regard, one may argue whether the workout data may even be used to identify the data subject at a later point in time. If anonymized properly (separated from personally identifiable information), the data subject should maybe not have the option of deleting all of that data, since it may no longer be used to identify the data subject and does not constitute a threat to his or her privacy.

Nevertheless, this is an open area of discussion where this project primarily takes its point of departure in the EU GDPR and the rights of the data subject. Thus, the data processing principles will be implemented when possible instead of being strictly restricted to the processing of personal data only.

5.6 User Rights

The EU GDPR providers users with a set of additional rights used to achieve greater control over their data. As mentioned in section 2.1, the following rights have been deemed relevant to address in the software development process and in the sample application or any other software [12, p.39].

The right to be forgotten - Article 17
"The data subject shall have the right to obtain from the controller, the erasure of personal data concerning him or her without undue delay, and the controller shall have the obligation to erase personal data without undue delay"[12, p.43]. Thus, the user has a right to get all personal data removed if there are no legal grounds or purpose in retaining it.

Reasons of removal

- Personal data is no longer necessary to the purpose or process
- User has withdrawn consent, and no legal grounds are in place
- User objects to the processing and no legal grounds are in place
- The personal data has been unlawfully processed
- The personal data has to be erased to comply with other legal obligations

Thus, a system must be capable of removing all personal data upon request and if any data is retained, one must have legal grounds for retaining the data. Based on the proposed sample application, there are no legal grounds to retain any data. Consequently, the sample application and its database must be designed in a way to support deletion of all personal data of a user, upon request by the data subject.

The right of access by the data subject - Article 15
"The data subject shall have the right to obtain from the controller confirmation as to whether or not personal data concerning him or her are being processed, and, where that is the case, access to the personal data"[12, p.43]

This includes information regarding:

- Purpose of processing
- Categories of personal data
- Recipients to whom data have been disclosed
- Timestamps of data stored
- If the personal data was not directly collected by the controller, then one must provide information about the source
- Information shall be provided in a commonly used electronic form.

As a result, a system must be capable of giving easy insight and access to all the personal data of a user upon request. In the sample application, a user may be able to export all his or her personal data in a format that is easy to access and formatted in a way that it is easy to read. In this case, a popular format like a PDF document with appropriate headlines may be used to format such personal data.

The right to restriction of processing - Article 18
"The data subject shall have the right to obtain from the controller restriction of processing"[12, p.44], including the following conditions:

- Accuracy of personal data is deemed inaccurate by user
- Processing is unlawful but user requests restriction instead of erasure
- Controller no longer needs the personal data, but user requires it for any legal claims
- The user has objected the processing of his or her personal data, which according to Article 21[12, p.45], requires the controller to stop the data processing unless any legal grounds are in place

Consequently, a system must be capable of halting and restricting any personal data processing upon request. Therefore, the sample application may support a feature that allows users to enable or disable partially or completely, any features that may process personal data in the system, upon request. Arguably, if a partial disable feature is not possible, then the system must disable the whole feature.

The right to know when one's data has been compromised - Article 33
The Data protection officer (DPO) is mainly responsible for notifying the data protection authority upon any breaches, but to support the DPO, the system may implement categorized logging functionality that logs different action based on how critical an action was. Thus, a DPO or any staff member who is responsible for the security of the system may recognize, search and look up any unusual activities and decide whether personal data has been compromised or not. For the sample application, a simple logging feature may be required to demonstrate such feature. However, it is not an explicit requirement in the GDPR.

The right to data portability - Article 20
With the right to data portability according to Article 20: *"The data subject shall have the right to receive the personal data concerning him or her, which he or she has provided to a controller, in a structured, commonly used and machine-readable format and have the right to transmit those data to another controller without hindrance from the controller to which the personal data have been provided"*[12, p.45]

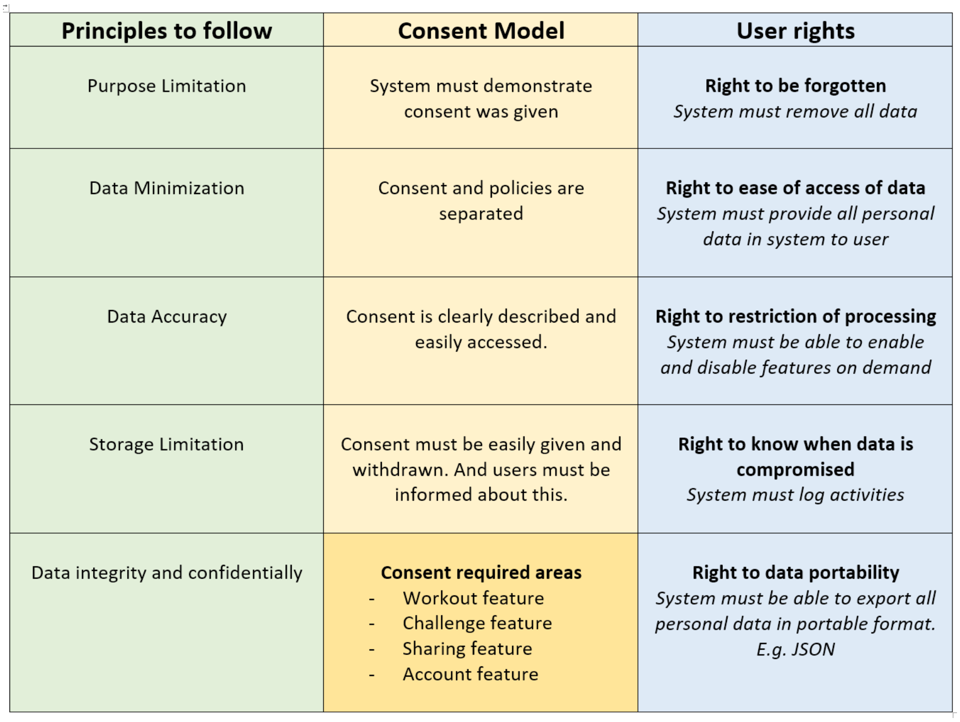
Also according to Article 20 subparagraph 1B and paragraph 2:

- The processing is carried out by automated means.
- Where technically feasible, data must be able to transmitted directly from on controller to another

Thus, a system must be able to support full personal data extraction in a portable format, which may be used by other providers upon request from the user. The portable data must be in a format that can easily be ported by the other providers when the user provides it and if feasible done directly between the two controllers. Thus, the sample application must support such a feature that can retrieve all data and form it in a portable machine-readable format (e.g. JSON) and provide the data through an API.

Overview of GDPR requirements

The GDPR has proposed many definitions, constraints and requirements, which a software developer must bring into consideration when designing a system that is in compliance with GDPR. To sum it all up, figure 2 provides an overview of the most important requirements that should be addressed in the sample application.



Principles to follow	Consent Model	User rights
Purpose Limitation	System must demonstrate consent was given	**Right to be forgotten** *System must remove all data*
Data Minimization	Consent and policies are separated	**Right to ease of access of data** *System must provide all personal data in system to user*
Data Accuracy	Consent is clearly described and easily accessed.	**Right to restriction of processing** *System must be able to enable and disable features on demand*
Storage Limitation	Consent must be easily given and withdrawn. And users must be informed about this.	**Right to know when data is compromised** *System must log activities*
Data integrity and confidentially	**Consent required areas** - Workout feature - Challenge feature - Sharing feature - Account feature	**Right to data portability** *System must be able to export all personal data in portable format. E.g. JSON*

Figure 2: Overview of GDPR constraints

6 Implementation

This section describes the implementation of the sample fitness web application that is to comply with the EU GDPR and its technical requirements.

Technology: the application has been developed in C# and ASP .NET CORE as described in the appendix section 10.1.

System Architecture: the system architecture follows the onion architectural design pattern. For a full explanation on this design choice and its implementation, please refer to the appendix subsection 10.4 .

Data Model and database design: the system uses Entity Framework Core to access the database for permanent storage of domain specific entities that is described together with the data model in the appendix section 10.5.

User Manual: a user manual with screenshots from the sample fitness web application may be found in the appendix section. 10.8.

Identity Framework: the application uses the Identity framework to add secure login functionality to the application and enable claims-based authorization. A description on how Identity is set up may be found in the appendix section: 10.2.

6.1 Authentication (who you are)

6.1.1 Data protection

As mentioned in the GDPR section 5.2, it requires companies to follow *appropriate security measures against unauthorized or unlawful processing and accidental loss, destruction or damage of personal data*, adhering to the security goal of Integrity and Confidentiality in Article 5 [12, p. 36]. Further, the EU GDPR requires *"the existence of appropriate safeguards including encryption..."* in Article 6 [12, p. 37].

In this regard, two popular security measurements have been applied to adhere with the security goals of Integrity and Confidentiality as described in the Theory section 3.2. Namely, the HTTPS (SSL on top of HTTP) protocol has been used to encrypt data communication between the client and web application. Further, identity passwords have been hashed by default. While the EU GDPR requirement on this matter remains broad and somewhat vague, these two security measurements have been deemed to be important security best practices when trying to protect personal information in a software application.

Firstly, to enable HTTPS in your application, one has to consult the **Startup** class [22] again:

```
public void ConfigureServices(IServiceCollection services)
{
  services.Configure<MvcOptions>(options =>
  {
  if (Environment.IsProduction())
  options.Filters.Add(new RequireHttpsAttribute());
  });
}
```

Code 1: Enable HTTPS in Startup

In this code snippet, HTTP is enabled in the **ConfigureServices** method that is called during runtime to add services to the application.

A filter is added to the request pipeline that ensures HTTPS is set up to receive client requests in the production environment. In this way, SSL is used to encrypt data transferred between the client and application. Further, all HTTP requests are automatically redirected to HTTPS for secure communication in the **Configure** method that is called during runtime, specifying how to respond to requests:

```
public void Configure{...}
{
        var options = new RewriteOptions().AddRedirectToHttps();
        app.UseRewriter(options);
}
```

Code 2: Redirect to HTTPS

As a result, the information communicated between the client and web application is now encrypted providing the benefit of confidentiality, as required in the EU GDPR article 5 [12, p. 36]. In this way, a potential adversary may not intercept and read the contents of the information passed between the client and app when exposing personal data like a username or password for authentication.

Finally, password hashing is used to hash passwords in the database as described in the introductory Identity section 10.2. This is used in the **AccountController** that is responsible for registration, login and logout operations of users.

For this purpose, it uses the Identity framework that contains a **UserManager** service for creating and managing users and a **SignInManager** service for logging users in and out. The services are injected into the constructor as follows:

```
[Authorize]
public class AccountController : Controller
{
    private readonly UserManager<ApplicationUser> _userManager;
    private readonly SignInManager<ApplicationUser> _signInManager;

    public AccountController(
        UserManager<ApplicationUser> userManager,
        SignInManager<ApplicationUser> signInManager, ...
    {
        _userManager = userManager;
        _signInManager = signInManager;
    }
```

Code 3: UserManager and SignInManager dependency injection in Startup class

The **UserManager** service may be used to create a user in the **Register** method:

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Register(RegisterViewModel model)
{
        ...
    await _userManager.CreateAsync(user, model.Password);
}
```

Code 4: AccountController Register method

The AccountController calls the **CreateAsync** method in the **Identity.UserManager** class, passing in a User object and the password to be hashed and saved. In this regard, the UserManager uses an underlying instance of an **IPasswordHasher<TUser>** to hash the password, an instance of **PasswordHasherOptions** to set the iterations to use when creating the hash (default is 10.000) and an instance of a **RandomNumberGenerator**. Altogether, this along with some other parameters on the size of the salt, hashed password, and hash function (SHA256 default) help create a hashed password and random salt used to generate the password and save it securely to the database [10].

This will not be described further in detail as it is out of the scope of this project [10]. Most importantly though, hashing the passwords makes it harder for potential adversaries to guess what the possible plain-text alternatively would be for the hashed password. Further, hashing them with a unique random salt prevents common attacks on the database like brute force, dictionary, and rainbow table attacks [29]. As a result, passwords may not be compromised, and personal data may not be altered in the application unwillingly, adhering to the goal of confidentiality and integrity respectively.

6.1.2 Authentication

Authentication is **the process of verifying the identity of a user**. In this particular application, the username (i.e. email) and password are used in the process of authenticating a user. This section strives to explain how authentication is performed throughout the application by using the Identity framework.

As shown already, Identity adds functionality to register, log in and log out an account by using the **UserManager** and **SignInManager** services in the **AccountController**. The Identity framework introduces a property called **User** on the **HttpContext**[7] of type **ClaimsPrincipal**, implementing **IPrincipal** that represents the current user of a given HTTP request. As the name suggests, claims are used authenticate and verify the identity of a user.

In this regard, **a claim is a statement about, or a property of, a specific identity**. The statement is comprised of a name and a value. For example, you might have a passport as a claim to verify your identity, and it might have an expiration date as a value, to determine whether it is valid or not. Another practical example might be when taking a flight - you would have to provide the claims you have on your first name and last name. For this particular purpose, your passport (i.e. identity) verifies those claims so that you can receive a boarding pass.

By default in ASP.NET Core, the **UserName** claim (i.e. email) is used to authenticate a user and login to the system by using the **SignInManager**.

[7]The HttpContext encapsulates all HTTP-specific information about an individual HTTP request

In the implementation, both the **Email** and **Password** property claim is used to authenticate a user if they have a confirmed email:

```
[HttpPost][AllowAnonymous][ValidateAntiForgeryToken]
public async Task<IActionResult> Login(LoginViewModel loginViewModel)
{
        if (ModelState.IsValid)
    {
        var user = await
            _userManager.FindByNameAsync(loginViewModel.Email)
        if (user != null)
        {
                if (await _userManager.IsEmailConfirmedAsync(user))
            {
                await
                    _signInManager.PasswordSignInAsync(loginViewModel.Email,
                    loginViewModel.Password,loginViewModel); ...
            }
        }
    }
}
```

Code 5: AccountController Login Authentication Example

The User inherits these claims from a ClaimsIdentity holding on to all the claims as follows (simplified version):

```
public class ClaimsIdentity: IIdentity
{
    public string AuthenticationType { get; }
    public bool IsAuthenticated { get; }
    public IEnumerable<Claim> Claims { get; }

    public Claim FindFirst(string type) { /*...*/ }
    public Claim HasClaim(string type, string value) { /*...*/ }
}
```

Code 6: ClaimsIdentity Properties

The main properties including **Claims** contains all the claims associated with the Identity, and the utility methods may be used for authorization purposes.

As mentioned previously, the User in the application is represented as a ClaimsPrincipal on the HttpContext that encapsulates all information on an HTTP request. Note that the class has an **Identities** property which returns an **IEnumerable<ClaimsIdentity** so a single User (i.e. ClaimsPrincipal) may have several identities.

Again, this makes sense because an identity may have some claims that comprise different forms of identity. For example, you might have a passport and a driving license as two forms of identity. In this case, the claims principal inherit all the claims from these identities:

```
public class ClaimsPrincipal :IPrincipal
{
    public IIdentity Identity { get; }
    public IEnumerable<ClaimsIdentity> Identities { get; }
    public IEnumerable<Claim> Claims { get; }

    public bool IsInRole(string role) { /*...*/ }
    public Claim FindFirst(string type) { /*...*/ }
    public Claim HasClaim(string type, string value) { /*...*/ }
}
```

Code 7: ClaimsPrincipal Properties

Thus, the point is that a principal in the application may have multiple **Identities** and these identities may have multiple claims that all are inherited by the **ClaimsPrincipal**.

6.2 Authorization (what you are allowed to do)

This section strives to explain how ASP.NET Core authorization is used in the application to solve the consent model requirement in the EU GDPR legislation. By definition, authorization is the process of allowing an authenticated user to access certain resources, by verifying whether the user has certain access rights to the system. In other words, authorization is the process of verifying that you have access to something. Thus, authorization helps control access rights by granting or denying specific permissions to an authenticated user.

First of all, the ASP.NET Core authorization package in **AspNetCore.Authorization** provides numerous authorization techniques including simple authorization, role-based authorization, and claim-based authorization.

6.2.1 Simple Authorization

At its simplest, one may use the **Authorize** attribute in the MVC controllers to limit access to the controller or action to authenticated (i.e. logged on) users only:

```
[Authorize]
   public class AccountController : Controller
   {
           [AllowAnonymous]
       public ActionResult Login()
       {
       }
           [Authorize]
       public ActionResult Logout()
       {
       }
   }
```

Code 8: Simple Authorization in AccountController

As shown above, the authorization is applied to the whole controller and the Logout action. As a result, only authenticated users can access the logout function. The **AllowAnonymous** attribute allows access by non-authenticated users. For example, the **Login** action should be accessible to everyone, regardless of their authentication status [21].

6.2.2 Role-based authorization

An identity may belong to one or more roles. For example, one person might have an administrator and user role, whereas another person might only have a user role. These roles may be used to limit access to controllers and their actions in the same way as shown previously:

```
[Authorize(Roles = ”Administrator”)]
public class AdministrationController : Controller
{
}
```

Code 9: Role-based Authorization Example 1

As a result, the controller is only accessible to users who are members of the **Administrator** role [17]. However, this particular project has deemed claim-based authorization to be more scalable and ”generic”, than issuing out roles to users, which the following hypothetical example might help explain

You might have an action for creating a customer, and you want people in an ”HR” role to be able to do that:

```
[Authorize(Roles=”HR”)]
public ActionResult CreateCustomer()
{
    return View();
}
```

Code 10: Role-based Authorization Example 2

Later, you realize that some people from the ”Marketing” role should also be able to create a customer and need to update your action:

```
[Authorize(Roles=”HR”, ”Marketing”)]
public ActionResult CreateCustomer()
{
    return View();
}
```

Code 11: Updated Role-based Authorization Example 2

Now you realize that not all Marketing people must be able to create a Customer, but it is not possible to assign a different role for those people in Marketing. Thus, you are forced to allow all marketing people to create Customers. Further, you have to update all the controller MVC action methods, and their authorize attributes each time you introduce new changes.

Altogether, this is not very scalable, and the level of abstraction is somewhat inappropriate for the purpose of authorization to become scalable when an application grows. For this purpose, claim-based authorization has been chosen [18].

6.2.3 Consent using Claim-based authorization

Please refer to the analysis chapter for an in-depth explanation of how claims-based authorization has been used to address the requirement of consent. Now, to use claim based authorization in ASP.NET Core, one must add claim checks in the code against a controller or action within a controller. This will specify the claims that the current user must possess and optionally a value that the claim must hold to access a requested resource (e.g. page).

The claim requirements are policy based, and these policies must be built and registered to express what claims may be required to access certain recourses. At its simplest, a claim policy looks for the presence of a claim without checking the value. For example, a claim policy might be added to check if a user should have access to the workout page. The policy is built and registered in the Authorization service configuration, residing inside the **ConfigureServices()** method in the **Startup** class:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();

    services.AddAuthorization(options =>
    {
        options.AddPolicy("WorkoutAccessPolicy",
                          policyBuilder => policyBuilder.
                             RequireClaim(ClaimType.WorkoutAccess.ToString()));
    });
}
```

Code 12: Registration of Policy in Startup class

In this example, the **WorkousAccessPolicy** policy checks for the presence of a claim with the name, **WorkoutAccessPolicy**, on the current identity. Now, the policy may be applied using the **Policy** property on the **Authorize** attribute to specify the policy name in the **WorkoutController** (see next page):

```
[Authorize(Policy = "WorkoutAccessPolicy"]
public class WorkoutController : Controller
{
...
}
```

Code 13: WorkoutController Claim Authorization

In this example, the **Authorize** attribute is applied to the entire controller. Thus, only identities matching the policy will be allowed access to the actions on the controller. However, you may use the **AllowAnonymous** attribute to allow anonymous access to certain actions that do not require any claims in the policy. Apart from the policy shown above, two other policies are used to determine if users may have access to the challenge and share feature but they have been omitted for clarity, since the approach is similar to the one just shown.

6.2.4 View-based Authorization

View based authorization is used to allow one to show, hide or modify the UI, based on the current user identity and their respective claims. They are accessed by the authorization service within MVC views via dependency injection. To inject the service into a Razor view, the **@inject** directive is used: @inject IAuthorizationService AuthorizationService. The service requires the library to be added as well: @using Microsoft.AspNetCore.Authorization [24]. A particular example in the code is the **EditWorkout.cshtml** Razor view file:

```
@if (await
    AuthorizationService.AuthorizeAsync(User,"ChallengeAccessPolicy"))
{ <a asp-controller="Workout"
  asp-action="Challenge"
  asp-route-entryId="@workoutEntry.Id"
  asp-route-workoutId="@Model.Id"
  class="btn btn-warning">
  Challenge
  </a> }
```

Code 14: View Based Authorization

The authorization service is used to check if the user satisfies the **ChallengeAccessPolicy** by calling the **AuthorizeAsync** method. This policy is only met if the user has given consent to the application to let it process challenge data. For this particular example, the challenge button should only be shown within workouts if the user has enabled the challenge feature (i.e. given consent).

6.3 Input Validation

According to the EU GDPR Chapter 2, Article 5: *Personal data shall be accurate and, where necessary, kept up to date; every reasonable step must be taken to ensure that personal data that are inaccurate are erased or rectified without delay (accuracy)* [12, p. 35]. To meet this particular requirement, validation rules have been enforced in the web application, and user input is validated in all input fields across the web application.

The ASP.NET MVC structure of the web layer in the application encourages you to follow the DRY ("Don't Repeat Yourself") design principle, by specifying functionality or behavior only once, and then have it reflected everywhere in the application. As a result, less code is written, and the code becomes less error prone, easier to test and easier to maintain (adhering e.g. to the security principle of simplicity). One example of this is the validation mechanisms provided by the MVC and Entity Framework Core. Specifically, these have been used to add validation rules to the models in the application (i.e. view models in the web layer and entity models in the data layer):

```
public class RegisterViewModel
  [Required][EmailAddress]
  [Display(Name = "Email Address")]
  public string Email { get; set; }

  [Required][DataType(DataType.Password)]
  public string Password { get; set; }
}
```

Code 15: RegisterViewModel Validation Example

As shown above, the DataAnnotations provides a built-in set of validation attributes that have been applied declaratively on the view model properties. The validation attributes specify behavior to be enforced on the model properties. For example, the **[Require]** attribute indicates that a property must have a value. Thus, when the user tries to register an account, they will be enforced to provide e.g. a password value. As a result, the application becomes more robust and ensures that user input is validated and you do not inadvertently let bad data into the database.

On the UI side, validation errors are reflected based on the validation that has been added to the model above. Thus, when a user fills out a form with invalid values, the errors will be detected and display error messages accordingly.

The web application uses HTML forms to receive user input, which automatically renders appropriate validation error messages for each field containing an invalid value. The errors are enforced both on the client-side using JavaScript and jQuery and on the server-side (if e.g. JavaScript is disabled). This is done in the HTML form by using the validation message tag helper that displays the error message:

```
<Register.cshtml> view file
@model RegisterViewModel
<form asp-controller="Account" asp-action="Register" method="post">
    <div asp-validation-summary="ModelOnly"></div>
    Email:  <input asp-for="Email" /> <br />
    <span asp-validation-for="Email"></span><br />
    Password: <input asp-for="Password" /><br />
    <span asp-validation-for="Password"></span><br />
    <button type="submit">Register</button>
</form>
```

Code 16: Register View File Validation Example

Inside the **AccountController**, the first (HTTP GET) **Register** action method displays the initial Register form inside the **Register.cshtml** view file. The second **Register** method (HTTP POST) calls the **ModelState.IsValid** method to check whether the registration data in the **RegisterViewModel** has any validation errors:

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Register(RegisterViewModel model)
{
        if (ModelState.IsValid)
    return View(model); // Retry
}
```

Code 17: ModelState Controller Validation

The action method evaluates the posted data from the HTML form and re-displays the form if it contains any errors along with an informative error message based on the model state passed back.

In addition to the validation of user input, ASP.NET Core uses annotations called ”Tag Helpers”[8] to create robust HTML forms used to post data back to the server. The form tag helper generates a standard HTML form as shown below. The form is accompanied by an action attribute, mapped to a particular MVC controller action on the server-side that dynamically generates a hidden ”Request Verification Token” to prevent cross-site request forgery. This is checked for with the **[ValidateAntiForgeryToken]** attribute in its respective HTTP Post action method.

```
<form asp-controller="Account" asp-action="Register" method="post">
    <!-- Input and Submit elements -->
</form>
```

Code 18: Html Tag Helper Form Example

The above Form Tag Helper generates the following HTML:

```
<form method="post" action="/Account/Register">
     <!-- Input and Submit elements -->
     <input name="__RequestVerificationToken" type="hidden"
         value="<removed for brevity>" />
    </form>
```

Code 19: Html Tag Helper Form - Hidden Code Generated

Overall, this ensures that the user does not execute unwanted actions in the application while they are authenticated. In general, the main benefit of this validation setup in ASP Net Core is that you do not need to change any existing code in the **AccountController** or in the **Create.cshtml** view file to activate the validation in the UI. In other words, the code adheres well to the SOLID[9] principle: *Open for Extension, Closed for Modification*, because the existing code is never changed but the validation can be added continuously when extending the model and still be reflected across the whole application.

Reference Article 5: [12, p. 35]

[8]Tag helpers enable server-side code to participate in creating and rendering HTML elements in Razor view files

[9]Acronym of five basic principles that help create a system that is easier to maintain and extend

6.4 Implementation of User Rights

In this subsection, the implementation of the user rights defined in the GDPR will be showcased. The rights and their practical implications will be discussed in the discussion section 8.

As mentioned in section 5.6, the following rights have been deemed relevant for a developer to implement in a system:

- Right to be forgotten
- Right to data portability
- Right to ease of access to one's personal data
- Right to restriction of processing

6.4.1 Right to be Forgotten

As mentioned in the analysis section 5.6, the right to be forgotten requires that the system support the erasure of all personal data related to a given user upon his or her request. Firstly, the database of the sample application must be designed to able to remove all data and its dependencies related to a given user. This is done by defining how the model entities are related to each other and how they are processed when dependent entities are removed. When implementing the database context, overriding the **OnModelCreating(DbModelBuilder builder)** method allows one to customize the relationship between entities, their attributes and how they react based on deletion.

The code snippet 20 on the next page, shows the **Context** class that is used to interact with data as objects from the database. Specifically, the code shows how the user entity is configured to become a parent entity to the workout and challenge entity. The relationship is defined by chaining the methods **HasMany** and **WithOne**, which denotes that a user entity may have many workout and challenge child entities associated. Reversely, a challenge or workout may have one parent user entity associated. The workout and challenge entity depends on the user entity as a parent, which is defined by using the **HasForeignKey** method to provide the child entity with a foreign key to the parent user entity, namely the userID attribute.

```
public class Context : IdentityDbContext<User>
{
     protected override void OnModelCreating(ModelBuilder
         modelBuilder)
        {
            modelBuilder.Entity<User>()
                .HasMany(pt => pt.Workouts)
                .WithOne(pt => pt.User)
                .HasForeignKey(pt => pt.UserId)
                .IsRequired()
                .OnDelete(DeleteBehavior.Cascade);
            modelBuilder.Entity<User>()
                .HasMany(pt => pt.Challenges)
                .WithOne(p => p.ChallengerUser)
                .HasForeignKey(pt => pt.ChallengerId)
                .OnDelete(DeleteBehavior.Restrict);
        }
}
```

Code 20: Entity relations and cascading

As a result, the foreign key may be configured to support cascade delete, meaning that if a record in the parent user table is deleted, then the corresponding records in the child workout and challenge tables will automatically be deleted. In practice, if a parent entity is deleted, one may use **OnDelete(DeleteBehavior.Cascade || .SetNull ||.restrict)** to define how the child entity is processed upon dropping its relationship with its parent. Since the sample application does not have any legal grounds for retaining any personal data, all data must thus be removed upon request. Thus, all child entities within the applications will cascade delete by using **.OnDelete(DeleteBehavior.Cascade)** to ensure that all related data is removed.

Based on code snippet 20 of the **Context** class, the system will not remove a given challenge if the user issuing a challenge, deletes his or her account. Instead, the system will set the foreign key to be null because the delete action has been configured to **DeleteBehavior.Restrict**. The reason for retaining challenges is to ensure that the one being challenged may still complete even if the person who originally created the challenge has been removed. Any references to the user issuing a challenge and his or her personal data are removed so no personal reference can be made to the deleted user from such given challenge. Thus, no personal data is exposed.

6.4.2 Right of Access

As mentioned in the analysis section 5.6, this right requires that the system is able to provide all personal data of a user in a readable and easy-to-understand format at any time. For this purpose, an export service was implemented with the responsibility of retrieving all personal data for a given user and convert it into an exportable format that is user-friendly. Based on the code snippet 21 below, the method **ExportToPortableObjectAsync** is responsible for converting personal data to a file that the user may download (e.g. PDF).

```
public class ExportService : BaseService, IExportService
{   // Logging removed for clarity
    public async Task<FileDto> ExportToFileAsync(int userId)
    {
        if (userId < 1) return null; // Validate user
        var user = await _userRepository.GetUser(userId);
        if (user == null) return null;

                // Portable data object to export
        var portable = MakePortableUserDto(user);
        if (portable == null)   return null;

        var file = await _fileProvider.MakeFileAsync(portable);
        if (file == null || file.FileContent == null ||
            file.FileContent.Length == 0) return null;

        return file;
        }
}
```

Code 21: Export to file (e.g. PDF)

The method will call another method called **MakePortableUserDto(user);**, which will retrieve all user data and convert it into a portable DTO, which contains all personal data of a user that may be used for further processing. In this case, the portable DTO is transferred to a fileProvider by calling the **MakeFileAsync(dto)** method, which will process the portable DTO and return a file containing the personal data. The chosen file provider generates a PDF file, but may easily be replaced with another provider, by injecting it into the export service.

By way of example, the code snippet 22 below illustrates how the methods **MakeFileAsync** and **ExecutePdfGeneration** are used to generate the PDF file:

```
public class PdfProvider : IFIleProvider<PortableUserDto>
{
    public async Task<FileDto> MakeFileAsync(PortableUserDto input)
    {
        //Raw content converted to PDF bytes
         var rawContent = FormatContent(input);
        var pdfContent = await ExecutePdfGeneration(rawContent);
        var dto = new FileDto
       {
            FileName = $"LiftLog_StoredData_(...).pdf",
            FileExtention = "pdf",
            FileContent = pdfContent
       };
       return dto;
    }
}
private async Task<byte[]> ExecutePdfGeneration(string rawData)
{
        var fileContent = await
            _nodeServices.InvokeAsync<byte[]>("./pdf", rawData);
        return fileContent;
}
```

Code 22: PDF file generation

Since the application uses the latest ASP .NET CORE technology, no PDF generator framework existed during the implementation of the sample application. Thus, NodeJS[10] and **jsreport**, which is a Javascript module used for report generation, are used to generate the PDF file containing personal data. The NodeJs service is used in the private **ExecutePdfGeneration(string rawData)** method that returns a byte array containing the content of the PDF. The content is then returned and added to a file DTO that contains the content, filename and file extension. This file DTO is then returned all the way back to the web layer and then back to the user as a PDF to be downloaded. An example of such PDF may be found in the appendix subsection 10.9

[10]Node.js is an open-source, cross-platform JavaScript run-time environment for executing JavaScript code server-side.

6.4.3 Right to Data Portability

As mentioned in the analysis section 5.6, the right to data portability requires, that the system is able to provide all personal data related to a user in a machine-readable format to be used by other providers. In this regard, one crucial aspect to consider is, how a user may provide the data in a common machine-readable format to another provider in an easy, safe and secure manner.

Thus, as mentioned in section 6.4.1, an export service was implemented that is responsible of retrieving all personal data. This is supported by an open **Export-Controller** API that may be used by third-party providers to retrieve such data. Firstly, each user is given a unique export token GUID that is used to refer to the user, when an export of their data is requested. Upon request, this token is combined with a generated URL in the **ManageController** that resides in the web layer, which is showcased in code snippet 23 below.

```
public class ManageController : Controller
{
         public async Task<IActionResult> GetExportLink()
         {
            var user = await GetCurrentUserAsync();
        var token = await _exportService.GetExportTokenAsync(user.Id);
        var pageAddress = HttpContext.Request.Host;
        var apiPath =
            $"https://{pageAddress}/api/export?token={token}";
        return RedirectToAction(nameof(Index), new ParamsDto {ApiPath
            = apiPath});
         }
}
```

Code 23: Generating export link

The generated export URL will take form as:
https://www.liftlog.dk/api/export?token=e80c7279-7885-4d2f-903e-bf10ff30b12a
The link refers to the **ExportController** API, which the user may hand over to another third-party provider to retrieve all of his or her personal data.

By using the link, the provider calls the open API and the GUID will be used by the **ExportService** to return a portable DTO, containing all personal data of the requested user as shown in code snippet 24 below.

```
public class ExportService : BaseService, IExportService {
    public async Task<PortableUserDto>
        ExportToPortableObjectAsync(Guid userToken) {
        var user = await _userRepository.GetUserWithToken(userToken);
        if (user == null) return null;
        var dto = MakePortableUserDto(user);
        if (dto == null) return null;
        await RefreshToken(user.Id);
        return dto; } }
```

Code 24: Generating portable DTO with token

The portable DTO is then converted to a common format such as JSON[11], which the sample application uses before returning the data back to the provider. An example of such result is found in code snippet 37, in the appendix. In order to secure the personal data of the user, each token is disposed upon each usage and a new token is generated. Thus, the URL and the old token, which is now exposed to the world, is rendered obsolete and may not be used again. Thus, the user must explicitly request for a new URL if another export of personal data is to happen. The token is refreshed in the **RefreshToken** method shown in snippet 25 below:

```
public class ExportService : BaseService, IExportService {
        private async Task<bool> RefreshToken(int userId) {
        var user = await _userRepository.FindAsync(userId);
        if (user == null) return false;
        user.ExportToken = Guid.NewGuid();
        var isSuccess = await _userRepository.UpdateAsync(user);
        return isSuccess; } }
```

Code 25: Refresh export token

6.4.4 Right to Data Restriction

Please consult section 6.2 to see how claims-based authorization is used to restrict data processing through the application.

[11]JSON (JavaScript Object Notation) is a lightweight data-interchange format that is easy for humans to read and write and easy for machines to parse and generate.

7 Sample Application Compliance Review

This section tries to evaluate whether the sample application is representative as a EU GDPR compliant piece of software based on what has been analyzed and implemented.

7.1 Privacy by Design and Privacy by Default

Throughout the development life cycle of the application, thoughts on privacy and the protection of personal data have been included as required in the GDPR and addressed in the analysis: 5.

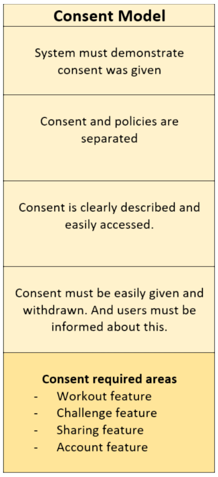
Further, personal data is pseudonymized where possible. Consequently, users are only referred through artificial IDs within the system or represented through a display name to other users that refer to them. By default, when a user creates an account, only the minimum amount of data is collected. That is, their email, password and an optional display name.

Also, all the consent settings are disabled by default to address the requirement of ”Privacy by Default”. Thus, no personal processing may occur unless consent is given during sign up or trough the account management page.

Finally, all data that may be collected in the account management page, such as name, age, sex and so forth are set as optional only. However, optionally collected data must still have a purpose in which the data can be processed. This may lead to additional complications discussed in section 8.3.

7.2 Consent

The requirements related to consent are illustrated in figure 3 below. Based on this, the system addressed the consent requirement as an access control problem where claim based authorization has been used as described in section 5.5. The consent in the system is easy to give and withdraw by pressing a check-box and nothing prevents the user from giving or withdrawing his or her consent at any time. Further, the description between consent and other policies like terms and conditions (TaS) have been clearly separated during registration.



Consent Model

System must demonstrate consent was given

Consent and policies are separated

Consent is clearly described and easily accessed.

Consent must be easily given and withdrawn. And users must be informed about this.

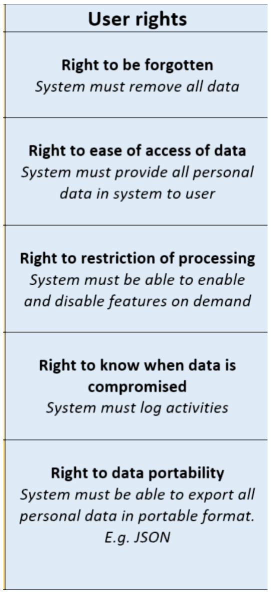
Consent required areas
- Workout feature
- Challenge feature
- Sharing feature
- Account feature

Figure 3: Consent requirements

The design of the consent model that was implemented, strives to be straightforward and easy-to-understand by the target audience. However, no usability tests were performed to verify if the majority of the target audience find this true. Despite this, it is assumed to be true, since the language used to request consent have been conveyed in a very simple manner, describing exactly what data is being processed and for what purpose. Nevertheless, based on the discussion in section 8.1, one may argue that a usability test may only address the needs of certain users. Thus, some users may be left out (i.e. not understand the request for consent), which makes the requirement of consent mostly or partly satisfied in the sample application. In other words, even though the application does not actually proof the consent to be easily understood by all users, it is assumed so based on the efforts made in keeping the language simple.

7.3 User rights

Based on the requirements of the user rights illustrated in figure 4, a technical solution has been proposed and implemented to satisfy such requirements.



User rights
Right to be forgotten *System must remove all data*
Right to ease of access of data *System must provide all personal data in system to user*
Right to restriction of processing *System must be able to enable and disable features on demand*
Right to know when data is compromised *System must log activities*
Right to data portability *System must be able to export all personal data in portable format. E.g. JSON*

Figure 4: Rights requirements

Right to be forgotten: The sample application partly satisfies this right by having a data model where related personal data can be deleted through cascading. Thus, all personal data may be removed from the system upon request. However, as discussed in subsection 8.2.1 and subsection 8.2.1, certain complications exist. Specifically, the issue with notifying third parties upon the event where a user invokes the right to be forgotten. In this case, the system does not inform or remove any personal data that the user has shared via third-party providers like Facebook, upon withdrawing consent or deleting an account. Consequently, this part of the right to be forgotten has not been satisfied.

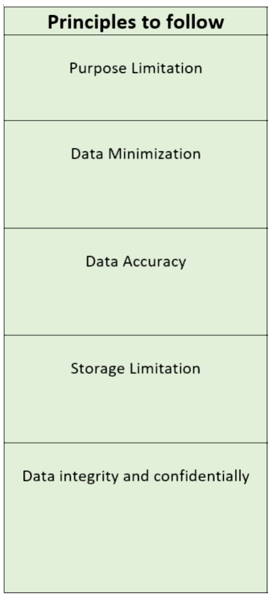
Right to data portability: The sample application satisfies this right by implementing an API, which may be accessed through a unique user generated link that may only be used once. Any provider may use the link and retrieve all personal data in JSON, which is a common machine-readable format. However, as discussed in section 8.2.2, this solution may satisfy smaller systems but may not be scalable when huge quantities of data are to be transferred to many providers.

Right of access by the data subject: The sample application satisfies this right by implementing a service that allows the user to export all personal data in a text format that is easy to use and understand, namely the PDF format. Nevertheless, as discussed in section 8.2.3, the implementation may satisfy the requirements but still not be scalable when vast quantities of personal data are to be exported or presented in a easy and readable format .

Right to data restriction: The sample application satisfies this right by using the access control mechanisms provided in the Identity framework from Microsoft. Specifically, claims-based authorization is used to restrict the data processing upon user request. Thus, users may, on demand, restrict or unrestrict features that process personal data.

7.4 Data Processing Principles

Based on the principles illustrated in figure 4 below, a technical solution has been proposed and implemented to satisfy such requirements.



Principles to follow
Purpose Limitation
Data Minimization
Data Accuracy
Storage Limitation
Data integrity and confidentially

Figure 5: Principles

Purpose limitation: Arguably, the application does not satisfy this principle entirely, since the application does allow the collection of some personal data, that may end up not being used. Specifically, it was decided to create a ranking feature where users would be able to compare their workout results. This feature relies on some of the optional data in the account settings but this data is not currently used because the feature was not finished.

As discussed in section 8.3, one may argue whether retaining data for future use is a purpose. Arguably, retaining data for future use is not really a legitimate purpose. Nevertheless, this is a gray area, which the GDPR does not address. However, from the risk involved with preserving such data, it has been assumed that the application does not satisfy this principle completely. Thus, it should rather have omitted this optional data collection that currently has no purpose.

Data minimization: Arguably, the sample application satisfies this principle, since each feature (e.g. challenge and workout) only uses the minimum required personal data. By way of example, when challenging another user, only the display name is shown together with the workout data to be shared. This is the minimum required data that a challenge requires to exist in practice.

Data Accuracy: The sample application satisfies this principle, since it provides means for users to easily access and update their personal data through the account management page Further, the application actively validates all user input, which helps increasing the accuracy of data in the system and prevent errors from happening. If errors do occur, the user is also prompted to correct them based on helpful messages.

Storage limitations: Arguably, the sample application satisfies this principle because it only displays personal data in a format that may directly identify a user per user session. In other words, when a user accesses the account management, the personal data is only showed and retained based on the lifetime of that particular page or user session. As a result, when the user leaves the page, this data is no longer stored in a form that may potentially help identify the user.

Integrity and Confidentially: The sample application partly satisfies these security principles because it does employ different security measures as shown in the implementation section 6. However, the sample application does not provide any backup mechanisms to restore personal data in the event of data loss. Arguably, this may be perceived as not being part of the developer's main responsibilities, since better backup options are often bought through outside providers.

7.5 Security of Data Processing

As mentioned in the analysis (see 5), the GDPR suggests four requirements to secure the processing of personal data.

Pseudonymization and Encryption: As mentioned already, the sample application uses pseudonyms across the system layers to reference users instead of personally identifiable information. Further, all communication between the client and web application is encrypted by default using HTTPS for communication.

Ensure confidentiality, integrity, availability and resilience of system: The application employs a range of security safeguards to address these security goals like encryption, pseudonymization, authentication, authorization, password hashing, input validation and proper error handling through redirects.

However, the application does not address the goal of availability and hardware-related issues. Arguably, the goal of availability may be addressed by using a remote backup solution and hosting the application across multiple servers. Thus, this goal may be a matter of buying a commercial solution that facilitates the ongoing availability of the application. Consequently, this is not directly perceived as the responsibility of the software developer.

Nevertheless, input validation and redirects to custom error pages have been implemented across the whole application to ensure the correctness of data and avoid the system from becoming unavailable due to malformed input or any other exceptions.

Restoring availability in the event of a physical or technical incident: From a technical perspective, this requirement is addressed through the active use of input validation and redirecting the user upon errors by default. Thus, the application does not simply crash upon errors. Nevertheless, physical incidents are not addressed in this project and might be included by buying services to keep the application available across multiple servers.

A process for evaluating the effectiveness of technical measures: As mentioned in the analysis (see 5), this requirement is not directly addressed in the project. However, one might imagine doing a range of security and penetration tests to determine the effectiveness of technical security measures used to protect personal data.

7.6 Compliance Summary

Based on the content of the discussion, the sample application does satisfy most of the conditions and requirements outlined in the GDPR with a particular focus on the data processing and consent requirements. However, some open areas are up to debate on whether the implementation meet their conditions.

Firstly, the GDPR requires security goals like Integrity, Confidentiality and Availability to be addressed without actually describing what specific safeguards should be implemented to adhere with them (except pseudonymization and encryption). Thus, these security goals have been open for interpretation during the project and the existing safeguards in the sample application only represent a range of common best practices when protecting personal data. Arguably, the Availability goal may be addressed through a remote backup and hosting solution.

Also, some scenarios in terms of consent and user rights have not been catered for directly. Specifically, the scenario where consent is withdrawn after having exchanged data via a third party is not addressed. Also, some optional personal data is collected for future purposes (i.e. ranking feature) even though it currently has no purpose (contradicts purpose limitation). Further, the 'right to be forgotten' is hard to employ in practice because it also requires the notification of other potential data processors to delete personal data that has been requested removed.

All together though, the sample application may still be used as a reference model to showcase how a software developer may address the most important requirements in the GDPR. Notably, this mainly applies to new systems whereas old legacy systems are not addressed. In that regard, the data model may not support the complete deletion of personal data to put the right to be forgotten into play.

8 Discussion

This section strives to discuss some of the afterthoughts on the EU GDPR requirements and the implementation of the sample web application. This includes any ambiguity, gray areas and problems experienced when interpreting the legislation as developers.

Arguably, the implementation of the GDPR requirements defined in section 5, such as 'Privacy By Design', 'Consent', 'Pseudonymization' and the numerous rights may not be implemented in a universal way and the EU GDPR does not directly explain how to do it.

Thus, the chosen GDPR requirements that will be discussed have been selected due to their stated importance in the GDPR, their high impact on this project and the high impact they may have on how software developers should create software in the future.

Further, any problems that existed during the implementation of the sample web application, as well as the problems that still remain afterward, will be discussed. Some of the challenges that emerged will also be considered from an organizational standpoint, as their impact differs from the technical ones, which may be worth looking into.

Finally, the discussion will draw a comparison between the EU GDPR and the security principles described in the section 3 to evaluate in which extent the EU GDPR addresses these security concepts.

8.1 Consent Implications

Consent is one of the key concepts that the GDPR requires a company to incorporate within their business processes and IT-systems when working with personal data. Thus, this subsection will discuss the implementation of consent in the sample applications, the problems experienced, as well as any implications and areas not touched upon in the sample application.

Consent is not universal

A number of challenges may emerge when front-end developers or usability designers try to create a consent model that fits their usability goals while complying with GDPR. According to Article 7, consent must be presented in a manner which is in an intelligible and easily accessible form, using clear and plain language [12, p.73]. However, the GDPR does not state what straightforward and plain easy to understand implies.

Thus, one of the key challenges is to design an interface for requesting consent that is ”easy” to use and understand and may therefore not be regarded as universal but must be tailored to the target group of an application.

In this project, the sample application was designed to be used by an audience of fitness enthusiasts. Arguably, using technical or complex terms on describing how the application uses personal data would not have been deemed appropriate.

However, since the audience is primarily adults, their linguistic skills may be broad enough to understand simple paragraphs such as *”Your country indication will only be processed for statistical purposes”* and they may have sufficient IT-skills to use a web application. Hence, a design was proposed and implemented as showcased in section 6, yet no usability tests have been carried out. Despite of this, it is assumed that the majority find it easy-to-understand.

However, even though one may satisfy the majority of users, it may not satisfy all users within the target group. In this regard, the GDPR does not directly specify anything about addressing the minority of the users. Instead, it only requires that consent is provided in a plain and understandable format.

As a consequence, legal actions related to the understandability of a proposed consent model may be complex and challenging, since it may be deemed impossible to target all user types and hard to agree on when consent was requested in a meaningful way.

Regardless, a key point is to be very clear about the target audience and who the consent model addresses. Based on this, usability tests may be carried out to test if a proposed consent model satisfies the majority of the users. However, the question whether usability tests are sufficient remains unanswered. These are some of the challenges and questions that developers may experience during the implementation of consent. Ultimately, some of the described gray areas and scenarios may force even developers to seek legal advice.

8.2 Rights and their implications

As a developer, one must implement software in a manner that satisfies the user rights and enable the invocation of them in the system. Failing to do so, may ultimately result in certain violations of the GDPR.

As mentioned in section 5.1, the introduction of the GDPR has provided EU citizens with additional rights to gain better control over their personal data [12, p.39]. These rights cover both technical aspects of a system, as well as organizational aspects. In this regard, this section will discuss the rights in the GDPR based the design choices of the sample application, the implications of the rights, as well as the general solutions suggested to their technical implementation.

8.2.1 Right to be Forgotten

This section will clarify some of the challenges implicated by the right to be forgotten. As mentioned in the analysis section 5.6, this right requires personal data to be erased if no legal grounds exist for retaining it.

Technical Standpoint

Based on a technical point of view, this constraint may be boiled down to a functional requirement that requires any personal data to be presented in a relational format that supports the partial as well as full deletion of personal data in the database.

As a software developer, it may not be relevant to know about the cause of the deletion of personal data. For example, whether the data is to be removed because of legal obligations, inaccuracy or withdrawal of consent. Instead, it is mainly important to develop a system that supports the complete deletion of personal data related to a subject. To do tihs, the system needs to be able to distinguish between data to be kept and data to be removed and map data in a way where personal data may easily be found and removed.

For example, this is illustrated in the sample application and its implementation of the right described in section 6.4. On one hand, the user may remove all personal data by deleting his or her entire account. On the other hand, the user may remove only some personal data, by withdrawing consent to a given data processing feature. By way of example, the user may choose to delete all workouts and their related data when withdrawing consent but keep any other remaining data.

Existing VS New Systems
One important implication to consider is the fact that the project mainly considers how to implement the GDPR requirements in fairly new software. However, it becomes highly relevant in legacy systems to address the right to be forgotten by designing a data model where all personal data is mapped to respective accounts in a way that allows for such deletion of data to take place.

Yet, the project does not demonstrate the problem that developers may face with existing systems. Not all personal data that is retained in an existing system might be mapped to an account or person. Therefore, when the right to be forgotten has been invoked, the key challenge as a developer may be to track down all personal data and map it in a way to ultimately support the removal of it.

This may create a significant problem in present systems where vast quantities of personal data are to be removed. Arguably, one might even ask if it is possible to remove all personal data in legacy systems? Without a doubt, existing systems (especially legacy systems) will find it challenging to employ the technical means and mapping of data in the database that satisfy the right to be forgotten in practice.

Regardless, these are some of the obstacles that a developer may need to address when designing the database in existing systems.

Notification of Third Party Companies
One of the main challenges faced during the implementation, arise from how a software developer should deal with data transferred to another third party. In particular, after consent has been withdrawn or when the right to be forgotten has been invoked. Based on the sample application, when a user deletes his or her account, only personal data in the system will be removed.

Yet, according to Article 17, when a user invokes the right to be forgotten, any third party who have links to or copies of the personal data must be informed that the user has invoked the right to be forgotten [12, p.44 - recital 2]. This includes technical measures to ensure that the third party is notified.

In this regard, the sample application does not support any technical means to inform Facebook or any other third party that certain personal data ought to be removed. Thus, it may not be able to fully satisfy the "right to be forgotten" after it has been invoked.

For this particular scenario, a software developer may want to develop an open API that allows third-parties to notify, whichever personal data has been flagged to be removed. This may require some security considerations to validate if a given request is valid. One might even imagine that companies will create an open standard API to accomodate for this need and help notify each other of personal data that should be removed.

Alternatively, the organization (i.e. data controller) may have to use other means to notify third party data processors, which may be cumbersome. In light of such controversy, the key point is to carefully consider what third-party processors are in play and how they may be informed about changes in personal data being processed, possible through an open API standard.

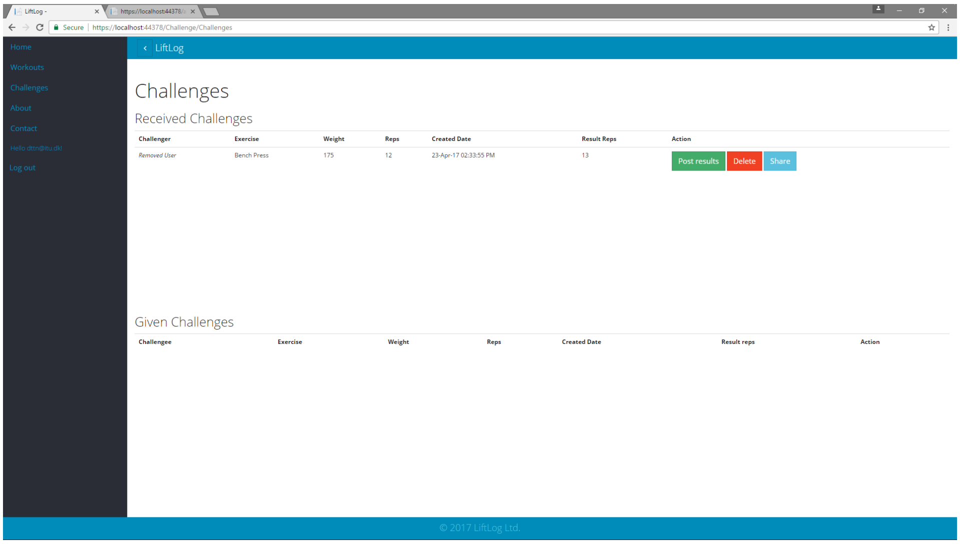
Data Sharing Implication
The right to be forgotten poses a certain challenge in scenarios where personal data is being shared between users. This issue is illustrated and partially solved in the sample web application when two users challenge each other.

Assume that user A challenges another user B. Later on, user A deletes his or her account. Since user B has been challenged; user B may still wish to complete such challenge, even if user A has deleted his or her account. Simply removing the challenge from user B's account may not be optimal in regards to usability.

To solve this scenario, any references that lead back to the challenger are removed or pseudonymized. Thus, user B will only see *"Removed User"* under the challenger display name, instead of the real display name of user A (Figure 6).

Hence, user B is still able to complete the challenge even though user A is no longer referred to, which ultimately satisfies both user needs.

However, this solution may only work on personal data that is pseudonymized. Other personal data may require additional technical precautions.



Challenges

Received Challenges

Challenger	Exercise	Weight	Reps	Created Date	Result Reps	Action
Removed User	Bench Press	175	12	23-Apr-17 02:33:55 PM	13	Post results Delete Share

Given Challenges

Challengee	Exercise	Weight	Reps	Created Date	Result reps	Action

Figure 6: Removed challenger reference

Right to be forgotten VS Statistical Data
Arguably, allowing users to erase all personal data may result in valuable business data being lost, making the company miss out on other business opportunities. This is particularly relevant for personal data that is not stored in a format, allowing for more than just temporary storage.

For example, the application was supposed to have a ranking features used to compare workout progress with other users. For this purpose, a requirement was to be able to compare workout progress based on sex, geography and age. However, if this data is kept in a form that may potentially identify a user, it may not be available for statistical purposes because of the right to be forgotten.

However, based on the current sample application, it does not depend on any data for this kind of statistical purpose in the event of all data being erased. On the other hand, if the sample application implemented a feature that process personal data (e.g. ranking feature) to provide some valuable statistical insight about the users, deleting some or all of that data may result in less accurate statistical insights. Consequently, this may render the feature useless and result in false insights about the users.

8.2.2 Right to Data Portability

As mentioned in the analysis section 5.6, the right to data portability require personal data to be portable so that users may transfer personal data to another provider without hindrance.

Technical Implications

The developer must consider three aspects when implementing a system that supports data portability. Namely data format, scalability and security.

Firstly one must ensure that the data is returned in a commonly used, machine-readable format. In this regard, the sample application uses JSON as a machine-readable format when transmitting the personal data. The key point is to choose a data format, which is commonly used and may be compatible with other providers upon transmission of personal data.

Secondly, one must consider how the data model is designed and how the system may retrieve all personal data to transmit it in a scalable way. Based on the sample application, all personal data related to a user is retrieved from the database and returned as JSON through an open API, which another provider may connect to. This solution works fine on less data but may not prove to be working well on huge quantities of data.

In this regard, the main issue is how to design the system to support transmission of personal data in a scalable way. Assume that a system has huge amounts of personal data belong to multiple users that wish to export all their personal data to another provider. This leads to a question on, how the system may transfer such vast amount of data to other providers in a concurrent and scalable manner.

The key point is that massive datasets may not simply be converted into a compact and portable format that can be transferred directly. Therefore, if a system ought to be working with massive datasets or expanded in the future, a developer must consider, which feasible and possible design choices that exist to satisfy data portability of such datasets in a scalable way.

Finally, the developer must consider the security of exporting personal data. The system must be safeguarded against unlawful data-extractions, while also allowing data to be easily transferred to another provider automatically upon request.

Based on the sample application, each user may generate an export link that contains a unique token. The link can then be sent to another provider, which may use the link to connect to an open API in the web application and retrieve all his or her personal data in a JSON format.

However, the link can only be used once, since the token is updated upon each use. This design choice was made to prevent the exposed link to be misused. Arguably, this method may not be optimal in scenarios where the other provider needs to retrieve the data multiple times, since the link will be rendered obsolete for each usage.

Consequently, the user must generate a new link for each request on behalf of the other provider. In the end, this becomes a hindrance to the user and the other provider. Ultimately, this implicates whether the data portability is satisfied completely because of the usability trade off.

One way of approaching this, may be to increase the number of times a link can be used before the token is refreshed. However, by doing so, one may increase the risk of the link being misused. Assume that a link may be used three times but was only used once. Consequently, the link may still be used twice, which an adversary may potentially misuse and compromise personal data.

The key point and challenge towards a developer, is to consider the balance between security and usability when it comes to transferring and accessing personal data of another user between different data providers.

8.2.3 Right of Access

As mentioned in the analysis section 5.6, the right of access by the data subject, allows users to access all their personal data, see what data is being processed, when it is being processed and see which third parties that have access to their data.

Technical considerations

Based on a technical standpoint, the developer must consider how the user may easily gain access to his or her personal data stored in the system and present all information describing the processing of the personal data.

Based on the sample application, the user may directly download a PDF file that contains all personal data and describes how it is being used in a readable format. An example of such generated PDF may be found in the appendix 10.9. This works well with small sets of data but leads to a scalability issue for larger quantities of personal data. Therefore, scalability is something to consider and this problem is similar to the scalability problem within the right to data portability.

A proposed solution when providing copies of huge sets of personal data, may be to employ a "divide-and-conquer"[12] approach where the data is split into different categories open for download. Further, one may create a dashboard on the application that allows a user to quickly look up any specific personal data and its usage to minimize the need of downloading soft copies.

Regardless, the key point is to consider the quantity of personal data to return, the time it takes to retrieve and how the data may be presented structurally, in order for the average user to get a proper overview of their personal data.

The implication of legacy data

Another challenge caused by the right of access, is managing where all personal data is located, how it is used and map it to the right users. This is particularly the case in large existing companies that already have systems with a huge user base.

In these kind of systems, it may be difficult to track down all personal data and its location. Also, if the personal data is not directly collected by the company but provided by a third party, then the company must provide information on such source.

[12]A larger complex problem is divided into smaller sub-problems solved independently

However, legacy data[13] may not contain information about the source, which make it challenging to accomplish the task of tracking down personal data and its source. Thus, the company may need to inform the users that the source is unknown upon request. Otherwise, the company is compelled to seek legal advice.

There is no simple solution on this issue in existing companies. Arguably, developers may need to change the existing systems in such a way that all personal data is mapped to a data model that contains information abouts its source and other related information (e.g. timestamp).

Pseudonymization Implications

In the sample application, pseudonymization has been used to identify users without using directly identifiable data. By way of example, users are only identified by an integer ID, when the system is processing personal data related to a given user. However, a display name is used as a pseudonym by other users to identify a given user, instead of using personally identifiable data such as the user's real name.

In practice, the act of pseudonymization may provide challenges in scenarios where the user is given some degree of pseudonymization control. For example, the user may define the display name used in the sample application and potentially enter directly identifiable data exposed to all other users.

Personal data may be regarded as disclosed in scenarios like these where a user may expose directly identifiable data by entering it into a public field in the system. Consequently, the disclosed personal data may be compromised and this leads to the question on whether it is the user or creators of the system that are responsible?

In this regard, one may argue that it is the user's responsibility, since the user chose to enter personally identifiable information in a public field. The system might even explain to the user that the data will be exposed publicly. Arguably, the organization behind the system may not be blamed for such actions based on user mistakes or misuse of services.

[13]Information stored in an old or obsolete format or computer system that is difficult to access

By way of comparison, a person may buy a car and decide to drive recklessly, which results in an accident with another vehicle. In this case, the company who provided the vehicle should not be blamed for the reckless driving. Hence, if the user decides to enter personal data into public fields while being informed about it, then the user might ultimately be deemed responsible for disclosing his or her personal data to the world.

On the other hand, according to the GDPR, an organization must protect the user and his personal data. Thus, even if their system provides some degree of control to the user when pseudonymizing data, the organization may ultimately be held responsible for the consequences that may follow. In this case, the software developers must ensure that appropriate technical measures are employed to prevent or handle such consequences.

Firstly, one may use use regular expression or other means to validate data provided by the user against their personal data. For example, the display name field in the sample application may be compared with other existing personal data related to that user. For example, one may check if the display name contains parts of any personal data already stored in the system.

Yet, this may not always be possible. Especially, if the system does not have any data to compare with, which may complicate this matter even further. Thus, the organization may also need to educate their users on the misuse of a given service that may lead to compromising their personal data and provide guidelines on how to use such service in a safe manner.

Regardless, pseudonymization is an advised method to protect personal data from being compromised. However, if some degree of control in the act of pseudonymisation is given to the users, then one must be aware of any consequences that may follow and consider the risks. As a software developer, one may only accommodate for this issue by explicitly telling the user when the data is exposed publicly and validate his or her input against any personal data already presented in the database.

At least, it may be too hazardous to assume that the responsibility of misuse lies on the users' shoulders only. Also, the GDPR requires one to implement safeguards that protect personal data. Arguably, this may also include when users expose personal data in public fields.

8.3 Purpose Limitation VS Future Proofing

Another challenge experienced during the implementation of the sample application was caused by the collection of personal data that was not used. Initially, a ”Ranking” feature was intended to be made, allowing the system to rank users based on personal data such as country, age, weight and sex. However, the feature was not implemented in time to be included in the final version of the program

However, the system still allows users to enter such personal data, which is collected without being used. In other words, this personal data may be collected without having an actual purpose, ultimately violating with the data processing principles of purpose limitation, which is discussed in section 5.1. Consequently, this personal data without a purpose may be required to be removed.

Suppose that the ”ranking feature” would be included in a later release of the sample application. Arguably, the collection of this personal data might suddenly have a purpose again. However, does this actually qualify as a legitimate purpose of collection? This is not addressed by the GDPR and its data processing principles. Thus, one may argue that the collection of personal data for later use is a valid purpose.

On the other hand, a counter argument may be the fact that the personal data is not used for a legitimate purpose at the moment. Thus, one may not claim that it serves any purpose until it is actually processed or that the means for processing such data, is in place. Hence, by collecting and retaining data without the means of satisfying a purpose, may be equal to having no purposes, ultimately violating with the principle of purpose limitation.

This controversy does not have a clear answer of what is right but the risk of violating the GDPR by retaining such data for future purposes, may be too great. Therefore, one may be required to remove any collected data and disable any features that collects personal data not being used with a purpose. The key point of this controversy may be to only collect what you need and what you can process.

8.4 Security Theory Comparison

This section strives to evaluate to which degree the EU GDPR satisfy the security goals and principles as outlaid in section: 3. Specifically, the section will try to cover which principles the EU GDPR addresses and help uncover any security measures that are left out in the legislation.

Security Goals Comparison

As stated previously, the EU GDPR directly addresses the security goals of confidentiality, integrity, and availability (CIA) in Article 32 and states that a data processing system should ensure that they are continuously maintained [12, p.52]. Specifically, the GDPR requires one *to ensure the ongoing confidentiality, integrity, availability and resilience of a system [12, p.52]*. However, it does not address the security goal of **Accountability**.

This stirred up quite some confusion during the project about whether to use any logging mechanisms to log the actions of the data subject when using the sample web application. It seems as if the security principle of **Traceability** has not been addressed in the EU GDPR.

According to Article 33 in the EU GDPR, companies only seem to have the responsibility of notifying about personal data breaches within 72 hours [12, p.52] without actually having to track the cause of the data breach. Arguably, the lack of expectations within this matter might be deemed to be a serious infringement of an otherwise important security measurement. The ability to track and verify the actions of a data subject and their history might be important in the actual investigation of serious data breaches.

As a final note, the suggested use of pseudonymization in the EU GDPR also contradicts this security practice because it ultimately strives to remove all identifiers of the data subject. Thus, it hinders the software developers from keeping a tracked history of actions performed within the application and makes it harder to hold actions accountable to certain users. Overall though, the EU GDPR addresses the classical 'CIA' security goals.

Security Principles Comparison
The EU GDPR and its technically derived requirements address quite a wide range of the listed security principles in the theory section (see section 3). As pointed out already, the 12 listed security principles are good rule of thumbs when trying to construct secure IT systems that adhere with the security goals.

The table below gives an overview of which security principles that the EU GDPR articles addresses:

Security Principle	**GDPR Requirement**
1: Simplicity	Not Addressed
2: Open Design	Not Addressed
3: Compartmentalization	Purpose Minimization [12, Art 25, p. 48]
4: Minimum Exposure	Data Minimization [12, Article 25, p. 48]
5: Least Privilege	Data Minimization [12, Article 25, p. 48]
6: Minimum Trust and Maximum Trustworthiness	Not Addressed
7: Secure Fail-Safe Defaults	Availability [12, Article 32c, p. 52]
8: Complete Mediation	Not Addressed
9: No Single Point of Failure	Availability [12, Article 32, p. 52]
10: Traceability	Not Addressed
11: Generating Secrets	Not Addressed
12: Usability	Consent Conditions [12, Article 7, p. 37]

Based on the table above, one can see that roughly half (7 out of 12) of the security principles are addressed in the EU GDPR. However, it should be noted that the EU GDPR only focuses primarily on the security measurements that may help with the data protection of personal data and user privacy. This may explain why some of the security principles are not directly addressed as part of the legislation.

In other words, many of the security principles not addressed do not necessarily have a direct effect on the compromise of personal data. For example, the **Simplicity** principle states that a simple system contains fewer flaws than a complex one. This, however, does not directly imply the compromise of personal data (but maybe only increases the likelihood that it may happen). The principle of **Open Design** also does not directly say anything about the protection of personal data.

The principle of **Least Privilege** extends the GDPR requirement of minimum exposure, since it implies that the data subject should only have access to the information and resources that are necessary for its legitimate purpose.

The three principles of **Complete Mediation**, **Traceability** and **Generating Secrets** are not addressed in the EU GDPR even though they may have a significant impact on the protection of personal data. Firstly, the principle of **Generating Secrets** state that secrets should have a certain entropy (i.e. order or size) to keep them sufficiently random and hard enough to guess. Otherwise, an adversary might compromise the account of a data subject and their personal data.

Secondly, the principle of **Complete Mediation** states that access to any object must be monitored and controlled to enforce access control and avoid privilege escalation. This is not addressed in the EU GDPR but partially applied in the sample web application that uses claims-based authorization to control access to the web application. In extent to this, the principle of **Traceability** is not addressed as well. Thus, the aspect of monitoring and tracing events and actions are not carefully addressed.

Consequently, the EU GDPR does not necessarily make it technically easier to investigate personal data breaches after they happen but only require companies to notify of and create awareness about them as mentioned previously.

As a software developer, however, one might want to employ certain logging mechanisms to monitor and log security-relevant system events along with the actions performed by the data subject. This history may help provide traces in the event of a data breach and potentially help in the investigation of tracking down the source of the incident.

8.5 Key points in discussion

Based on the discussion, the GDPR has seemingly introduced many new challenges across organizations and to developers. This includes how a front-end developer may design a consent model, and which means a developer may consider when proving that the consent model satisfies the conditions of consent.

Also, the developer must consider how a system may technically satisfy the user rights when they are invoked. All the rights require that a back-end developer considers how to design the database in a way so that all personal data may be mapped to their respective owner.

This is crucial when a right has been invoked, since the system must be able to track down all personal data and perform the given request, such as the data erasure used to satisfy the right to be forgotten.

In this regard, the conditions of the GDPR may require developers to implement open APIs across organizations that allow providers and third-parties to connect and perform various actions. This includes the notification of erasure of data, changes of disclosed personal data or when data should be transmitted to another provider. Thus, developers may consider the possibility of having such open APIs that satisfy these operations in an easy but secure manner.

In addition, the developer must also consider the amount of personal data that the system should be able to process and how this can be done in a secure and scalable way. This is mainly important when addressing the right to data portability and the right to data access.

Further, the developer must consider the privacy of their users by default. This includes employing safeguards like pseudonymization of their personal data to avoid exposing it unnecessarily. Also, ensuring that all components, modules, and features processing personal data must distinguish between data that may be processed (e.g. by consent) and data that is restricted from processing (e.g. right of restriction).

Finally, the GDPR has stated some security goals which must be addressed such as integrity and confidentially. However, to fully protect the organization and its users, the developers may want to address other security principles that are not addressed in the GDPR, including Complete Mediation, Traceability, and Generating Secrets.

To sum up, developers must consider some of these design choices and scenarios when developing EU GDPR compliant software. Some of the requirements in the GDPR have been easy to derive technically. However, conditions such as purpose limitation, the right to be forgotten, consent and pseudonymisation imposed a series of challenges that may require further legal advice.

9 Conclusion

The project set out to clearly investigate what changes the EU GDPR will bring and how software developers may comply with it.

In this pursuit, it was found that the new regulation strives to strengthen and unify the data protection rules across the EU. Companies now have to comply to avoid the increased financial sanctions of up 20 million or 4% of a company's yearly turnover. Thus, the protection of personal data of EU citizens has now become an important worldwide matter that should be addressed according to the requirements in the GDPR.

In this regard, new roles like a Data Protection Officer, Data Controller, Data Processor and Data Subject followed by concepts like Privacy by Design, Privacy by Default and Consent have been introduced. A software developer must now consider these terms along with what personal data is being processed, when it is processed and how it is processed in software.

Thus, privacy and the protection of personal data should be considered throughout the whole software lifecycle (i.e. Privacy by Design). Further, privacy should be a default option, meaning that processing of personal data should be disabled by default (i.e. Privacy by Default).

Also, the processing of personal data should take into account the six data processing principles (article 25) and the processing of personal data should be secured by following the four secure processing requirements (article 32).

Further, the processing of personal data should only happen on behalf of the consent of the data subject (i.e. user), which has led to a series of conditions on how to design and present consent (article 7). Specifically, the requirement of consent was translated into an access-control problem solved with claims-based authorization. In other words, access to data processing functionality is granted based on consent.

In addition, the GDPR provides EU citizens with a range of rights that should be enforced throughout the software (article 15, 17, 18 and 34). Altogether, these requirements were derived in the analysis and implemented in an ASP .NET Core sample application where relevant personal data is collected, processed and exchanged. This helped illustrate how to address the technical GDPR requirements in practice.

Finally, the implementation of GDPR does not happen without the presence of any obstacles or trade-offs. As a developer, one must consider many design choices addressing all the conditions in the EU GDPR to be interpreted and complied with. This involves the design of consent, the implementation of user rights, the appropriate processing of data and the implementation of privacy and security safeguards.

Moreover, the developer must consider how the system handles and tracks personal data. This becomes especially important when working with huge quantities of data where scalability and performance may become a major challenge to overcome. Also, the maintenance of existing systems and legacy data constitute a major challenge of itself, which this project does not consider.

To sum up, the findings of this project and the sample application do not constitute a final answer on how to develop GDPR compliant software. As such, developers may still need to seek legal advice in areas of the GDPR that are open for interpretation. However, the project has helped illustrate many of the technical aspects found in the GDPR, as well as best practices that may provoke change of thoughts amongst software developers in the future.

10 Appendix

10.1 ASP NET Core

The web application has been build in the ASP.NET Core cross-platform framework, which is a technology stack issued by Microsoft using C# as a primary language. The framework is a significant redesign of the previous ASP.NET framework that has resulted in a much leaner and modular framework. This framework has been chosen for the development of the web application because of its new redesign. Specifically, the ASP Net Core framework provides modular components based on a set of granular and well factored NuGet packages. As a result, the framework has become much leaner and modular. The framework has been chosen because of the architectural change that allows you to optimize your program by including the NuGet packages you need. Consequently, the web application exposes a smaller surface area and includes tighter security.

Thus, the framework adheres well to the security principles of minimum exposure and compartmentalization. In other words, the framework only exposes packages that your application actually need, ultimately minimizing the possibilities for a potential adversary to attack the system by exploiting any vulnerabilities in the packages.[8, p.5] Further, the packages are clearly isolated meaning that the impact of an attack on a given component (i.e. package) does not entail the compromise of other components.[8, p.4] Finally, the Microsoft .NET ecosystem provides a broad range of mature and well-established packages with tools that help build security measures into an application (e.g. the Identity framework). Thus, the choice of ASP Net Core has been deemed well-suited for this particular project and its security focus.[7]

10.2 Introduction to the Identity Framework

The web application uses ASP.NET core Identity to add login functionality to the application. The Identity framework has been used to add functionality to register, log in and log out a user. In this regard, the database has been configured to use the Identity framework to store Users, Roles, Claims and Logins for authentication and authorization. The project contains the **Microsoft EntityFrameworkCore** package, which is used to persist identity data and schema to the SQL Server using Entity Framework Core and **Microsoft.AspNetCore.Identity** for the identity system. To enable the Identity services, one has to add it to the **ConfigureServices** method in the **Startup** class that is used as a dependency injection container for all packages used in the application:

```
// Add services to the container during runtime
public void ConfigureServices(IServiceCollection services)
{
    services.AddIdentity<ApplicationUser, IdentityRole>()
        .AddEntityFrameworkStores<ApplicationDbContext>()
        .AddDefaultTokenProviders();
}
```

Code 26: Identity service added to the Startup class

As a result, the Identity services are now available across the whole application through dependency injection.[14]
Now, to actually enable and use Identity one must call **UseIdentity** in the **Configure** method of the Startup class. By default, doing this adds cookie-based authentication (See code snippet in Appendix 10.3) to the request pipeline[15] providing tokens to uniquely identify a user session.[16]:

```
public void Configure(IApplicationBuilder app)
{
        app.UseIdentity();
}
```

Code 27: Enable Identity in Startup class

At this point, authentication and identity features have been configured, but the Database context[17] and User model have not yet been migrated. In order to store identities, one must add the identity database tables and schema to the existing database and make the application user inherit these properties:

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.Data.Entity;
public class User : IdentityUser {}
...
public class DbContext : IdentityDbContext<User>{}
```

Code 28: Identity Database Migration

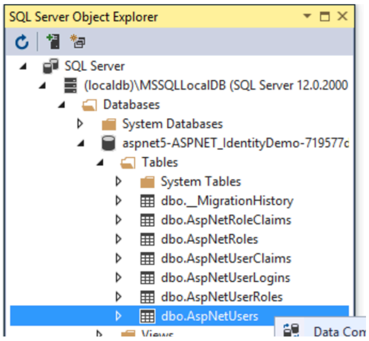
As a result, the following database tables have now been added:

[14]DI is a technique for achieving loose coupling between objects and their dependencies where dependencies are not directly instantiated in the class but provided outside the class in Startup.cs

[15]Mechanism used to handle HTTP requests

[16]The session of activity that a user with a unique IP spends on the Web site in a specific period of time

[17]Class that is responsible for interacting with the database



```
SQL Server
  (localdb)\MSSQLLocalDB (SQL Server 12.0.2000
    Databases
      System Databases
      aspnet5-ASPNET_IdentityDemo-719577c
        Tables
          System Tables
          dbo.__MigrationHistory
          dbo.AspNetRoleClaims
          dbo.AspNetRoles
          dbo.AspNetUserClaims
          dbo.AspNetUserLogins
          dbo.AspNetUserRoles
          dbo.AspNetUsers
        Views
```

Figure 7: Identity Database Tables

Specifically, the **AspNetUsers** table contains the application users and stores, amongst other things, the username and password of a user that has been hashed automatically with a random salt. The identity framework automatically converts the variable-length passwords into cryptic fixed-length passwords by first generating a salt (unique random number) and then hashing the password using a cryptographic hashing function on the password combined with the salt to help ensure that the passwords may not be compromised even if the database is. In this way, users may be authenticated without keeping and therefore risking the plaintext password if the data is compromised. As a side note, it should be mentioned that the **AspNetRoles**, **AspNetUserClaims**, **AspNetRoleClaims** and **AspNetUserClaims** are used to store roles and claims used for role- or claim-based authorization in the application, which will be further elaborated in the following sections.

10.3 Other ASP.NET Core Security Measurements

Safe storage of secrets during development As part of the ASP .NET Core Framework, a built in Secret Manager tool in development has been used to keep secrets out of the code. In general, it is bad practice to store passwords or other sensitive data in the source code. Instead, the **Configuration** system has been used as a way of configuring the application based on a list of name-value pairs stored using the **SecretManager** tool.[15] The way this works is that secrets are stored as environment variables that may then override existing configuration values during runtime of the program. To use this, one needs to reference the **Microsoft.Extensions.SecretManager.Tools** NuGet package and may then use the secret manager tool to set a secret in a console window within the path of the project containing the **Startup** class:

```
dotnet user-secrets set <MySecret> <ValueOfMySecret>
```

Code 29: Secret Manager

As a result, one can now access the Secret Manager secret through the configuration system in the **Startup** class and avoid exposing any application secrets:

```
public Startup(IHostingEnvironment env)
{
        if (env.IsDevelopment())
    {
        builder.AddUserSecrets<Startup();
    }
}
```

Code 30: User Secrets in Startup class

Now, one can access application secrets in the **ConfigureServices** method when running the application:

```
public IConfigurationRoot Configuration { get; }

        public void ConfigureServices(IServiceCollection services)
        {
            _testSecret = Configuration["MySecret"];
        }
```

Code 31: Secret Configuration Example

In the project application, this is used to store the API configuration values used for Facebook integration and email verification (SendGrid).

Cookie Authentication Management

```
public void ConfigureServices(IServiceCollection services)
{
        services.Configure<IdentityOptions>(options =>
    {
        // Password settings
        options.Password.RequireDigit = true;
        options.Password.RequiredLength = 8;
        options.Password.RequireNonAlphanumeric = false;
        options.Password.RequireUppercase = true;
        options.Password.RequireLowercase = false;

        // Lockout settings
        options.Lockout.DefaultLockoutTimeSpan =
            TimeSpan.FromMinutes(30);
        options.Lockout.MaxFailedAccessAttempts = 10;

        // Cookie settings
        options.Cookies.ApplicationCookie.ExpireTimeSpan =
            TimeSpan.FromDays(150);
        options.Cookies.ApplicationCookie.LoginPath =
            "/Account/LogIn";
        options.Cookies.ApplicationCookie.LogoutPath =
            "/Account/LogOff";

        // User settings
        options.User.RequireUniqueEmail = true;
    });
}
}
```

Code 32: Cookie Authentication configured in Startup class

10.4 System Architecture

The web application follows an onion architecture that is intended to address some of the common challenges like coupling and separation of concerns. Specifically, the application is comprised of 4 layers: a core layer, a data layer, a service layer and a web layer. The fundamental rule is that all code can depend on layers more central, but the system cannot depend on layers further out from the core. For example, the uppermost web layer has access to all the other layers, but the innermost Core layer does not have access to the remaining layers on top of itself. As shown on the figure 8, the Core layer resides at the very center and contains the domain model representations used throughout the whole project.

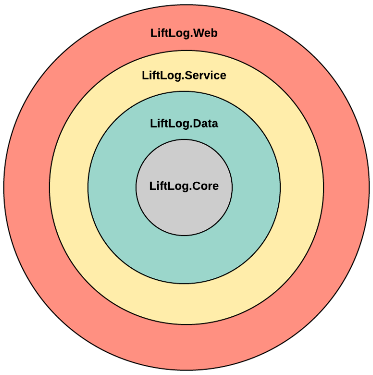


Figure 8: Onion System Architecture

Thus, all other services above have access to this layer used to represent data across the layers. For example, data transfer objects[18] are used to communicate data between the different layers where as each layer uses a specific representation of data(e.g. the web project uses view models where as the data layer uses database specific entities). On top of this is the data layer that represents the data access layer (i.e. DAL) used to provide access to data in persistent storage. This layer uses a set of repositories as an abstraction layer between the data access layer and business layer (i.e. service layer). Specifically, the repositories allow to store and load domain objects from a local Microsoft SQL Server using Entity Framework as a data access tool. On top of the data, layer resides a service layer that represents the business logic layer or domain of the application. This layer is comprised of a set of services that provide the functionality of the fitness application.

[18]DTO: an object that carries data between processes

Finally, the outermost web layer represents the presentation layer in the application used for the user interface and handling client requests. The web layer is an ASP MVC project that follows the architectural MVC design pattern with three main logical components: the model, the view, and the controller. In this regard, the model represents the state of the application data using view models. The controller handles the client interactions with the web application and updates the model to reflect changes in the application that are passed to the view for rendering. The view receives information from the controller based on requests and renders the user interface to the user. Ultimately, this is the way web applications are developed in ASP .NET Core and the MVC design pattern helps decouple the components allowing for better code reuse and separation of concerns.

The final and actual implementation of the onion architecture and its dependencies are illustrated in a package diagram depicted in figure 9. The web layer in the blue package utilizes the services in the orange package which then uses the repositories in the gray package. The core layer and it's DTOs are stored in a separate package which is depicted as the red folder and is used to transfer data between these layers. On top of the whole architecture, the identity framework provided by Microsoft illustrated by the Turkish blue package is used by every layer to perform necessary actions related to access control and authentication.

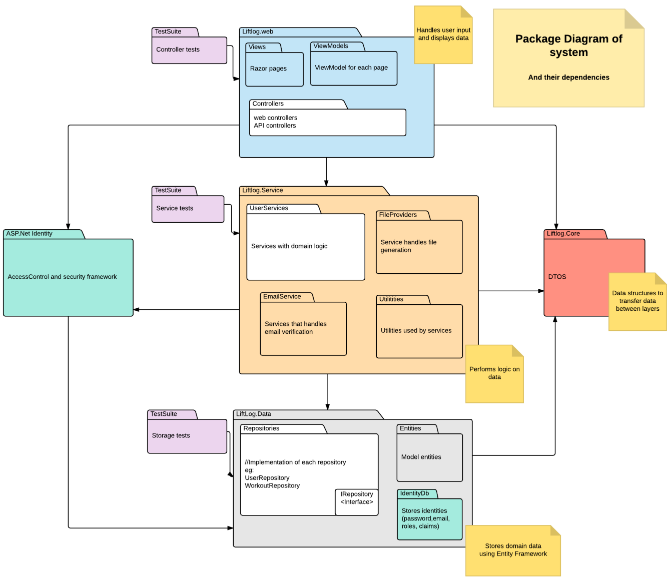


Figure 9: Actual architecture

10.5 Data Model and Database

This subsection outlines the data model of the application and how it was designed and implemented.

Domain models and entities

Since the sample application involves the fitness domain, one is required to determine which domain objects are relevant for the scope of the system to design a data model to satisfy the application requirements from section 4. The following models have been derived and will be regarded as persistent models and domain entities:

- User
- Workout
- WorkoutEntry
- Exercise
- Challenge

User A User represents a single unique account bound to a person. In regards to the domain, the entity contains personal data such as age, sex, name, height, weight and are mainly used as a parent reference to the other entities such as workout and challenge. However, it must be noted that attributes related account details and access control such as password, email, role, claims are inherited by Microsoft's Identity Framework. These attributes have not been explicitly included in the Entity Relationship diagram depicted in figure 10 as the main focus point is to illustrate the domain models and their relations and not the relations within the identity framework.

Exercise An Exercise represents a single unique fitness exercise which a person can perform during a workout. Thus, an Exercise Entity consists of a name such as "Benchpress" and a description of how to perform such exercise.

WorkoutEntry A WorkoutEntry represents the event when a person puts an exercise into a context, where a number of reps combined with an amount of weight are performed. Thus, the WorkoutEntry entity contains a rep and weight attribute with foreign keys to an exercise and a workout.

Workout A Workout represents a specific session where a set of exercises has been performed. Thus, a Workout entity contains a set of workoutEntries, a name, timestamp of when it has occurred and a foreign key to a user.

Challenge A Challenge represents a competitive event between two persons, namely a challenger and the challenged (challengee), that is based on a given exercise performed by the challenger. Thus, a challenge entity consists of attributes that describe a challenge; a timestamp, reps, weight, and a foreign key to an exercise and attributes that describe the results if the challenge such as ”result reps”. The result reps specify the number reps a challengee has managed to perform and can be used as comparison variable between the challenger and challengee.

Relationship between models

Based on the derived models the entity relationship diagram (figure 10) has been created to showcase which entities exists, their attributes and their relations.

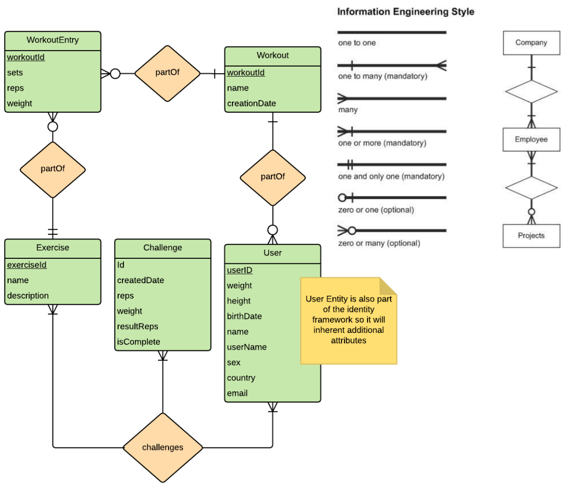


Figure 10: Entity relation diagram of sample application

The user entity has a zero-to-many relationship with the workout and challenge entity since a user does not require to have any workouts or challenges to exist. The workout entity has a one-to-one relationship with a user because a workout can not exist without having a user who defined the workout. Also, a workout entity has a zero-to-many relationship with the workoutEntry entity since a workout does not necessarily need a workoutEntry to exist.

However, the WorkoutEntity has a one-to-one relationship with a Workout and Exercise. Based on the definition of a workout entry, no workout entries can exist without being part of one exercise, and one workout since a workout entry represents an event consisting of a performed exercise, that is part of a workout session.

Finally, The challenge entity has a one-to-one relationship with an exercise and a many-to-many relationship with users. Specifically, a challenge in this domain requires exactly two users and one exercise to exists.

Implementation of Database

The sample application utilizes Microsoft's Entity Framework(EF) Core. EF Core is a lightweight and cross-platform object-relational mapper (O/RM) that enables .NET and C# developers to work with a database without specifying boilerplate code related to data-access code.[19] EF Core utilizes model classes that describes entities and a derived context that represents a session with the database which one can perform query actions on. The model classes are just plain C# classes that defines how a given model looks like. By way of example a workout entity in the sample application is defined as depicted in code snippet 33

```
public class Workout
{
        public int Id { get; set; }
    public int UserId { get; set; }
    public virtual User User { get; set; }
    public string Name { get; set; }
    public DateTime CreationDate { get; set; }

    public virtual ICollection<WorkoutEntry>
                          WorkoutEntries { get; set;}
}
```

Code 33: Workout model class

Based on the properties within a model, EF core will automatically create a table with the attributes that are mapped to the properties and their types. Any parent-child relationships can be represented by included other models in the class. Based on code snippet33 A reference to the user entity is done by included the User class in a property and also a foreign property. The naming convention for a foreign key is "Model"Id where "Model" is the name of the class and Id is a postfix annotation denoting a foreign key. By example "UserId". A parent-child relation example is shown in code snippet 34.

```
public int UserId { get; set; }
public virtual User User { get; set; }
```

Code 34: Parent-child relationship

The example in code snippet 34 demonstrates a one-to-one relationship, however EF core can also automatically define one-to-many relationships by including a collection of the models that ought to be in a one-to-many relationship. By way of example, based on code snippet 35 a workout has a reference to a collection of workout entries and a workout entry has a reference to a workout.

```
public class Workout
{
        //One-to-many with workout entries
        public virtual ICollection<WorkoutEntry> WorkoutEntries
            {get; set;}
}

public class WorkoutEntry
{
        //One-to-one with workout
        public int WorkoutId { get; set; }
        public virtual Workout Workout{ get; set; }
}
```

Code 35: One-To-Many relationship

EF Core will then be able to automatically retrieve and include all related entities a model has when queries are performed such as a retrieve all query.

The Database context as mentioned earlier is a derived class that represents a session with the database in which queries can be performed on but also how the database is defined. EF core will automatically create a database based on how the context class is defined. By way of example in code snippet36:

```
public class Context : IdentityDbContext<User, IdentityRole<int>,
    int>, IContext
    {
        // Entities of data model represented as a set (table)
        public DbSet<Exercise> Exercises { get; set; }
        public DbSet<Ranking> Rankings { get; set; }
        public DbSet<Workout> Workouts { get; set; }
        public DbSet<WorkoutEntry> WorkoutEntries { get; set; }
        public DbSet<Challenge> Challenges { get; set; }

}
```

Code 36: Part of sample application database context

Each ”**Dbset<TEntity>**” property defines a set for a given model class ”TEntity” such as a DbSet of Workouts. EF Core will use these DBsets to define the required tables, use the properties from the models that are bound to a DbSet and define the attributes and foreign keys needed in the given table. By way of example, EF Core will take DbSet<Exercise> and create a table called ”Exercise” with attributes based on the properties of the Exercise model class.

To access the database and perform CRUD operations on the database, one has to create a new instance of a context which now represents a session with the database. Then one can query against the Dbset based on which table or model to access and EF core will then perform the actions on the database. The way data is accessed is done through a repository pattern where each model has its own repository that is derived from a generic repository for which each CRUD operation is defined. Each repository is then provided a context through dependency injection and the CRUD operations are then performed on the injected context. By utilizng a repository pattern, the system is not coupled by Entity Framework Core but is also open for other ORMs or databases as well as making the code more testable due to dependency injecting.

Note that the sample application context derives from a IdentityDbContext from the Identity Framework and not from EF Core standard context because the IdentityDbContext contains additional features and databases related to access control which is not part of the standard context. How the Identity framework and EF Core uses IdentityDbContext will not be described as this is out of the scope.

10.6 JSON Example

```
{
   ”retrievalDate”: 2017-04-23T13:58:51.3946954+02:00,
   ”email”:abc@abc.dk,
   ”displayName”: pseudonym,
   ”creationDate”:2017-04-23T00:00:00,
   ”name”: John Doe,
   ”birthDay”: 1993-03-01T00:00:00,
   ”sexType”: Male,
   ”countryName”: Denmark,
   ”bodyWeight”: 65,
   ”height”: 180,
   ”workouts”:[
      {
         ”name”: Workout 1,
         ”creationDate”: 2017-04-23T13:57:11.4521736,
         ”workoutEntryDtos”:
         [
            {
               ”exerciseName”: Shoulder press,
               ”set”:1,
               ”weight”:5,
               ”reps”:30
            }
         ]
      }
   ],
   ”challengeGiven”:0,
   ”challengeReceived”:0
}
```

Code 37: Exported JSON

10.7 Final Review of Technical Findings by Omada

Ómada Review Meeting Keypoints - 3rd of May 2017

Actions

1) Omada Technical Presentation: Dennis
2) Meeting Notes : Thor
3) Adjust Discussion : Thor
4) Conclusion : Thor
5) Proof Read and adjust program : Dennis

Presentation

- Show full pictures with flow (fade)
- Start introduction with explaining goal of “best practices” for software developers
- Data Accuracy: Mention and show input validation (red jquery validation message)
- Security of processing - mention examples of evaluation process (penetration, risk)
- Proof of Consent - mention logging of timestamps and consent
- Show action confirm flow in presentation

Report

- Data Accuracy: Mention input validation
    - What we do: 1) validation and 2) facilitate easy access to data
- Security of processing - mention examples of evaluation process (penetration, risk)
- Availability: not a developer concern
    - Buy commercial, e.g. azure hosting and remote hosting
- Consent alternative: only request consent when actually needed?
- Consent - mention logging of timestamps and consent
- **Problems outside developer scope -> seek legal advice**
    - Purpose limitation vs Future proof -> seek legal advice
    - Consent VS Data sharing -> seek legal advice
    - Data portability -> consent at receiver company -> seek legal advice
- Data portability -> **Facilitate transport of data (enabler) -> not judicial role**
    - **new api for notification between companies and access/deletion**
- Pseudonymization: 1) add validation on existing personal data and 2) tell user how used
- Pseudonymization Problem: validation on personal data is not possible if not present
- Right to be forgotten problem -> notification of deletion to third parties
    - **We have responsibility to notify third party of deletion as a COMPANY**
    - **Maybe companies need a new “notification” API for notification, access and deletion of data**

Program

- Settings instead of “Hello user x…”
- Lawful Processing: replace “Access Denied” with “Sorry, you have not given consent to this service”
- Pseudonymization: 1) add validation on existing personal data and 2) tell user how used
    - Problem: validation on personal data is not possible if not present

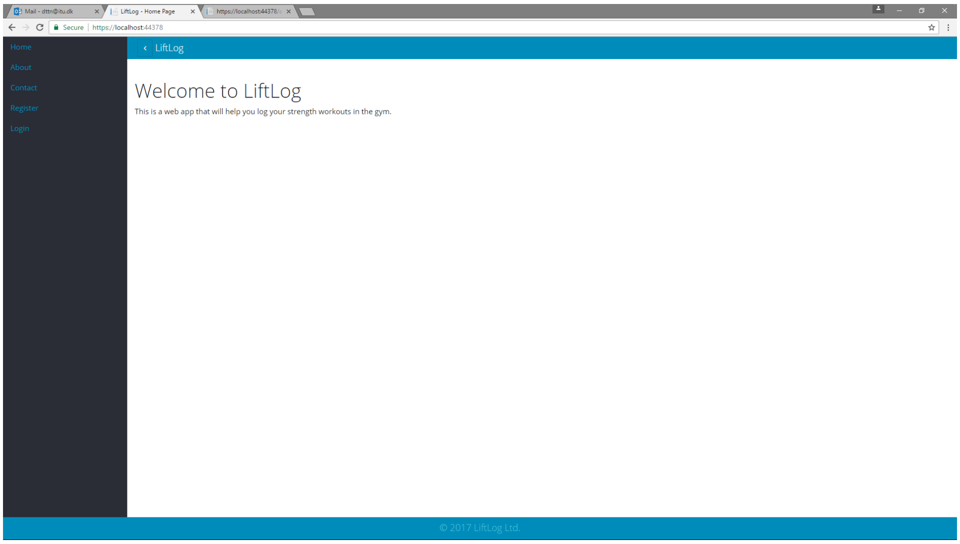
10.8 User Manual

The next couple of subsection will illustrate how to use the sample application, namely Liftlog and its features. The application supports the following use cases and features:

- Creation, Deletion, and Modification of an account
- Workout management and creation of specific workout entries
- Challenge features
- Sharing capabilities on social media
- Export of personal data as PDF or JSON
- Enable or disable features based on user consent.

Overview and main page

When launching the web application, the user will initially be presented with the main page. If the user is not signed in, the user will only have a limited set of pages to access as illustrated in figure 11.



Welcome to LiftLog

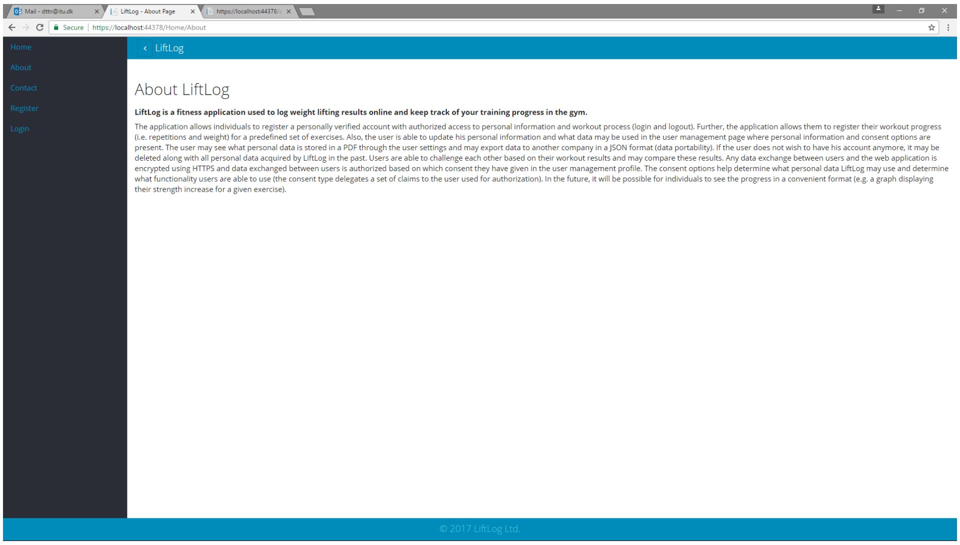
This is a web app that will help you log your strength workouts in the gym.

Figure 11: Limited main page

On the left side of the page, a navigation bar can be accessed and used for navigating around the website:

- The "home" tab will return the user back to the main page
- The "About" tab will redirect the user to a page explaining what LiftLog is and its purpose (Figure 12)

- The ”Contact” tab will redirect the user to a page where one can get in touch with the site administrator (Figure 13)
- The ”Register” tab will redirect the user to a page where one can create a new account.
- The ”Login” tab will redirect the user to a page where one login with an existing account



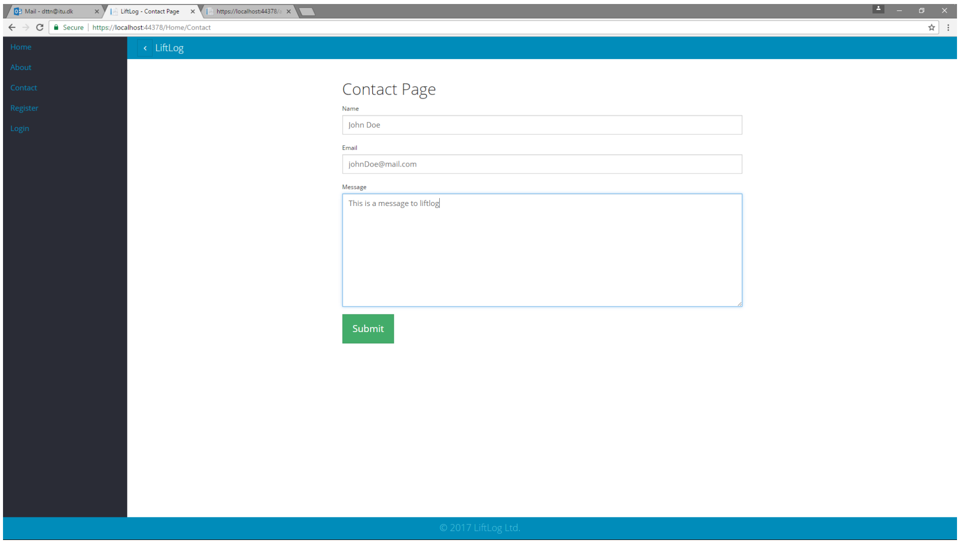
About LiftLog

LiftLog is a fitness application used to log weight lifting results online and keep track of your training progress in the gym.

The application allows individuals to register a personally verified account with authorized access to personal information and workout process (login and logout). Further, the application allows them to register their workout progress (i.e. repetitions and weight) for a predefined set of exercises. Also, the user is able to update his personal information and what data may be used in the user management page where personal information and consent options are present. The user may see what personal data is stored in a PDF through the user settings and may export data to another company in a JSON format (data portability). If the user does not wish to have his account anymore, it may be deleted along with all personal data acquired by LiftLog in the past. Users are able to challenge each other based on their workout results and may compare these results. Any data exchange between users and the web application is encrypted using HTTPS and data exchanged between users is authorized based on which consent they have given in the user management profile. The consent options help determine what personal data LiftLog may use and determine what functionality users are able to use (the consent type delegates a set of claims to the user used for authorization). In the future, it will be possible for individuals to see the progress in a convenient format (e.g. a graph displaying their strength increase for a given exercise).

© 2017 LiftLog Ltd.

Figure 12: About page



Contact Page

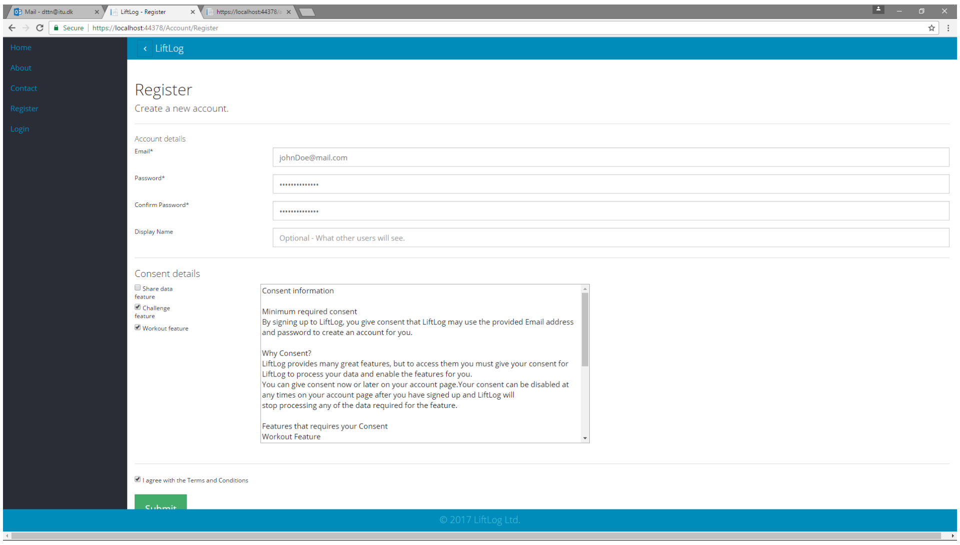
Name: John Doe

Email: johnDoe@mail.com

Message: This is a message to liftlog

Figure 13: Contact page

Registration
To create a new account, the user must go to the register page as illustrated in figure 14. Here one must enter an email address used as a username and a password to sign in with. The length of the password must be at least eight characters long and contain special characters as well, to be valid. The user also has the choice to define a display name, which is the name other users in the application will see. If none is given, a generic name will automatically be used instead. Also, the user will be able to give consent on which features to enable and can also read what personal data is used inside the text box of the consent area. To confirm and sign up, the user must accept terms and conditions and press on the green submit button



Register

Create a new account.

Account details

Email* johnDoe@mail.com

Password* ••••••••••••••

Confirm Password* ••••••••••••••

Display Name Optional - What other users will see.

Consent details

- [ ] Share data feature
- [✓] Challenge feature
- [✓] Workout feature

Consent information

Minimum required consent
By signing up to LiftLog, you give consent that LiftLog may use the provided Email address and password to create an account for you.

Why Consent?
LiftLog provides many great features, but to access them you must give your consent for LiftLog to process your data and enable the features for you.
You can give consent now or later on your account page.Your consent can be disabled at any times on your account page after you have signed up and LiftLog will stop processing any of the data required for the feature.

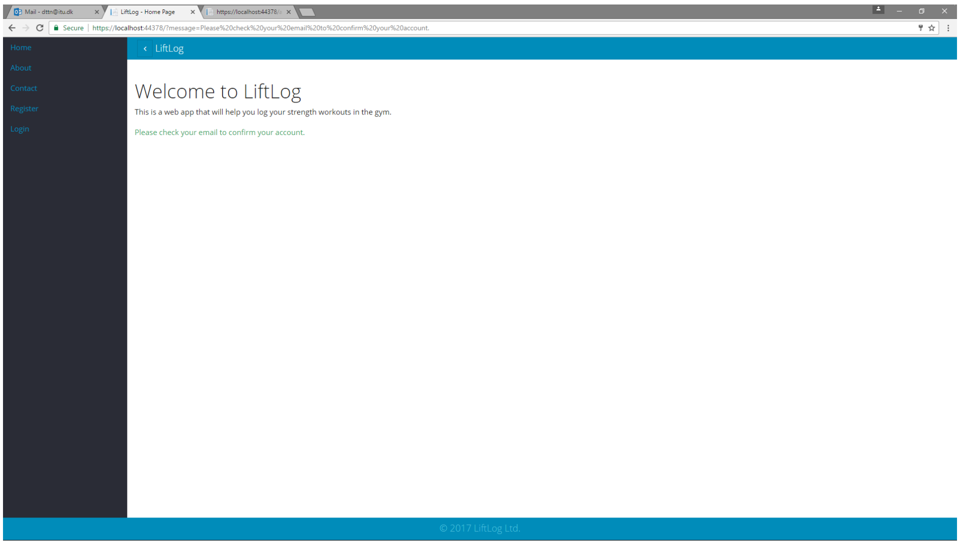
Features that requires your Consent
Workout Feature

- [✓] I agree with the Terms and Conditions

© 2017 LiftLog Ltd.

Figure 14: Register page

When the submit button is pressed, the user will be notified that he or she must confirm the registration by pressing on the confirmation link sent to the email used during registration. This is illustrated in figure 15



https://localhost:44378/?message=Please%20check%20your%20email%20to%20confirm%20your%20account.

Welcome to LiftLog

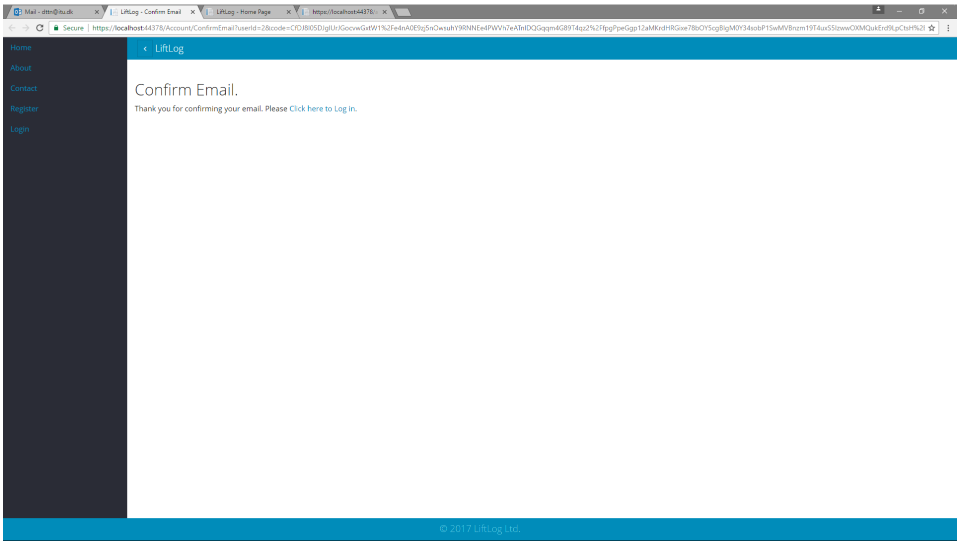
This is a web app that will help you log your strength workouts in the gym.

Please check your email to confirm your account.

© 2017 LiftLog Ltd.

Figure 15: Notifcation of email confirmation

Once the account is confirmed, the user will be redirected to a page explaining the account has been verified and is ready to be used. This is illustrated in figure 16.


Mail - dttn@itu.dk
LiftLog - Confirm Email
LiftLog - Home Page
https://localhost:44378/
Secure
https://localhost:44378/Account/ConfirmEmail?userId=2&code=CfDJ8I05DJglUrJGocvwGxtW1%2Fe4nA0E9zj5nOwsuhY9RNNEe4PWVh7eATnIDQGqqm4G89T4qz2%2FfpgPpeGgp12aMKrdHRGixe78bOY5cgBlgM0Y34sobP1SwMVBnzm19T4uxS5IzwwOXMQukErd9LpCtsH%2I
Home
About
Contact
Register
Login
LiftLog

Confirm Email.

Thank you for confirming your email. Please Click here to Log in.

© 2017 LiftLog Ltd.

Figure 16: Account Confirmed page

Login

To log into LiftLog with an existing user account, one must enter the email and password in the login page and then press the green login button. Moreover, one can login with an existing Facebook account by pressing on the gray ”Facebook” button in the ”Use another service to login” section. In the event of a lost password, a user can create a new password by pressing the ”Forgot your password?” link below the login button. An email will be sent with a link that redirects the user to a page where one can reset the password of the account. The login page is illustrated in figure 17

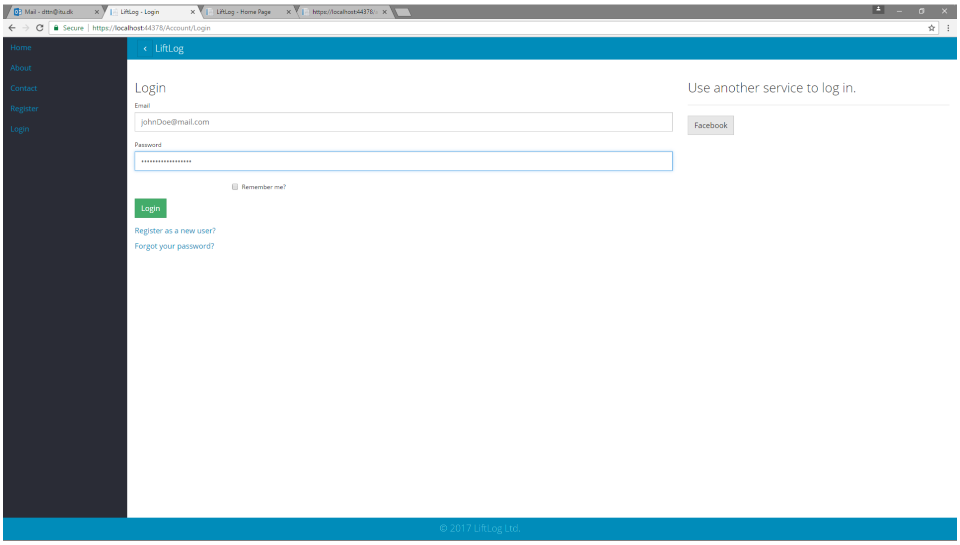


Figure 17: Login page

Once logged in, the user will be redirected back to the main page, but will now have full access to the whole site as illustrated in figure 18.

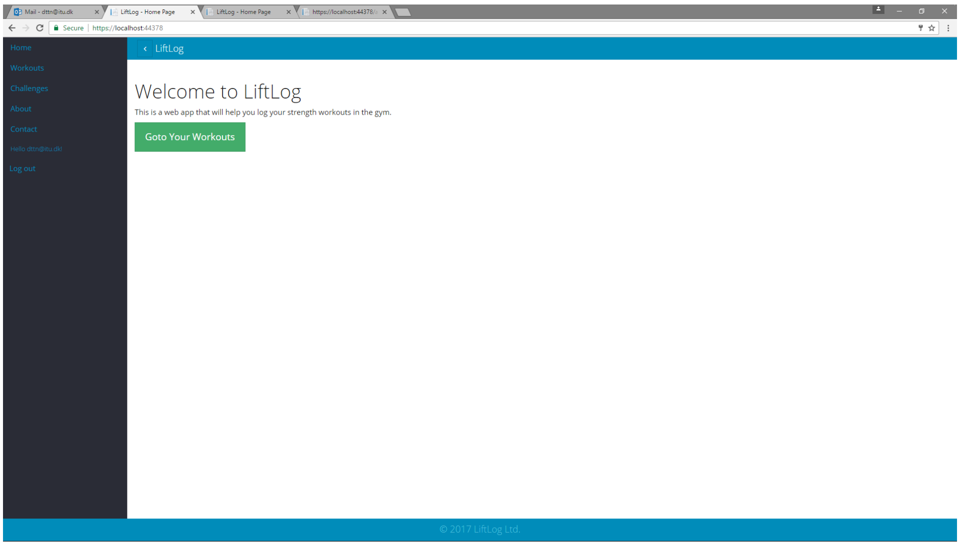
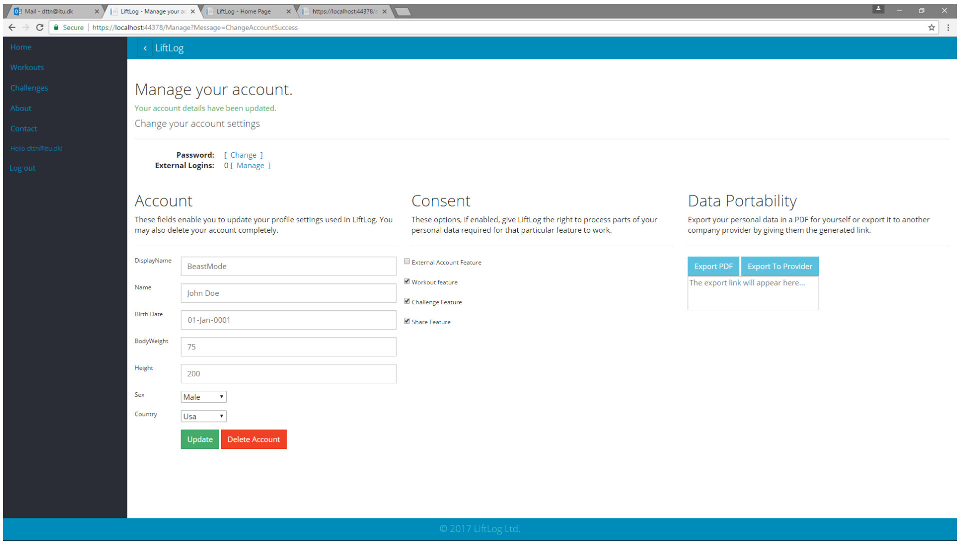


Figure 18: Full main page

Account Management

A user can customize or change his or her account on the Account Management page, which can be accessed by pressing on the "Hello XXXX" tab on the navigation bar located on the left side of the page. The page is illustrated in figure 19.



Manage your account.

Your account details have been updated.

Change your account settings

Password: [Change]
External Logins: 0 [Manage]

Account

These fields enable you to update your profile settings used in LiftLog. You may also delete your account completely.

DisplayName	BeastMode
Name	John Doe
Birth Date	01-Jan-0001
BodyWeight	75
Height	200
Sex	Male
Country	Usa

Update Delete Account

Consent

These options, if enabled, give LiftLog the right to process parts of your personal data required for that particular feature to work.

- [ ] External Account Feature
- [x] Workout feature
- [x] Challenge Feature
- [x] Share Feature

Data Portability

Export your personal data in a PDF for yourself or export it to another company provider by giving them the generated link.

Export PDF Export To Provider

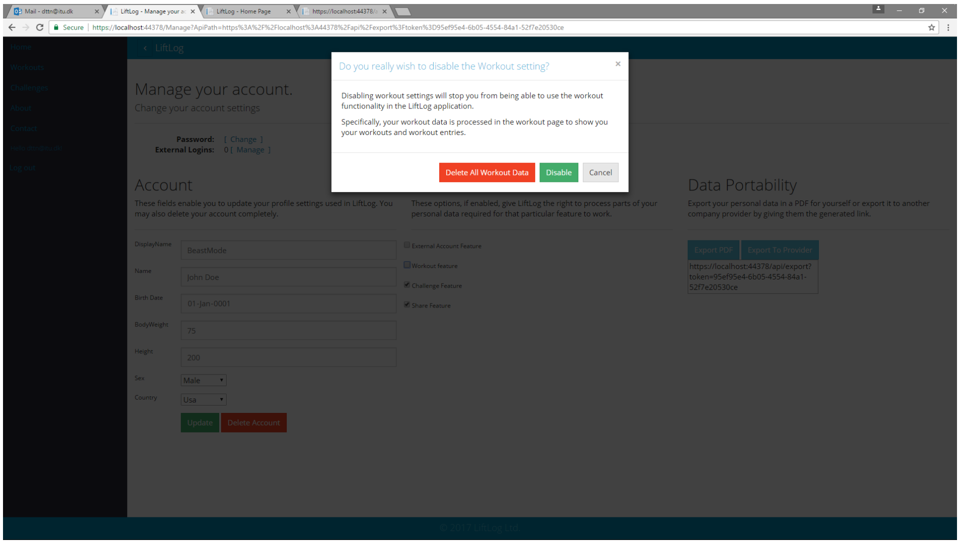
The export link will appear here...

© 2017 LiftLog Ltd.

Figure 19: Account management page

Modify account details The user can modify any personal information bound to the account under the ”Account” section. Further, the user can add, change or remove the display name, real name, birth date, height, weight, sex, and country. To confirm the changes, one must press the green update button. To change the password, one can press the blue ”change” link next to the bolded password label above the ”Account section”.

Consent The user can change his or her consent settings under the ”Consent” section. When giving or disabling consent to a feature, a pop-up dialog will be displayed on the screen with information about the consent and the consequences of giving or retrieving consent. This is illustrated in figure 20.



Do you really wish to disable the Workout setting?

Disabling workout settings will stop you from being able to use the workout functionality in the LiftLog application.

Specifically, your workout data is processed in the workout page to show you your workouts and workout entries.

Manage your account.

Change your account settings

Password: [Change]
External Logins: 0 [Manage]

Account

These fields enable you to update your profile settings used in LiftLog. You may also delete your account completely.

DisplayName	BeastMode
Name	John Doe
Birth Date	01-Jan-0001
BodyWeight	75
Height	200
Sex	Male
Country	Usa

These options, if enabled, give LiftLog the right to process parts of your personal data required for that particular feature to work.

- ☐ External Account Feature
- ☐ Workout feature
- ✓ Challenge Feature
- ✓ Share Feature

Data Portability

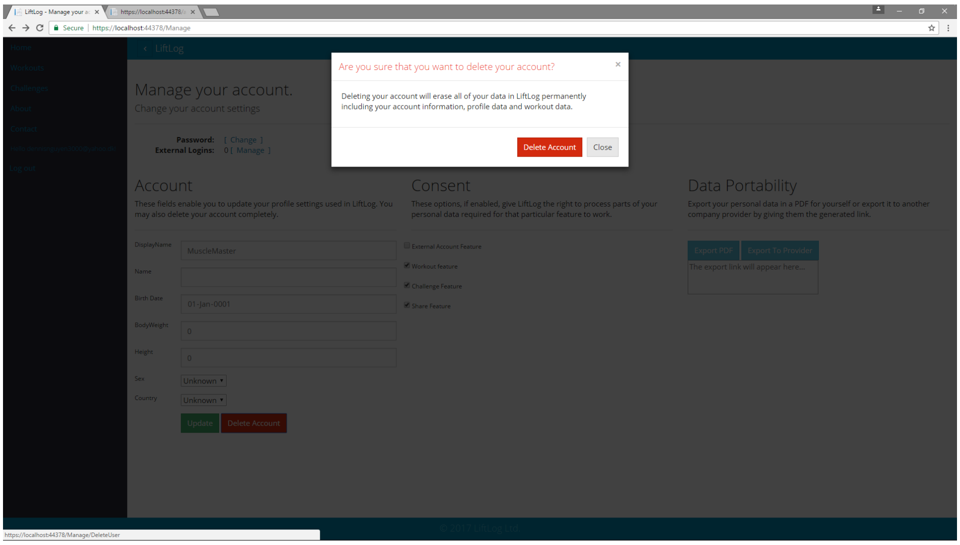
Export your personal data in a PDF for yourself or export it to another company provider by giving them the generated link.

https://localhost:44378/api/export?token=95ef95e4-6b05-4554-84a1-52f7e20530ce

Figure 20: Withdrawing consent pop-up

When withdrawing consent, the user has the option to delete any personal data related to that given feature, by pressing on the red ”Delete all XXXX data” button. Notice that this action cannot be undone.

Delete account The user can delete the created account at any times by pressing the red ”Delete Account” button in the Account Management page. A pop-up dialog box will be shown to inform the user and confirm the deletion. This is illustrated in figure 21. Note that this action cannot be undone.



Are you sure that you want to delete your account?

Deleting your account will erase all of your data in LiftLog permanently including your account information, profile data and workout data.

Manage your account.
Change your account settings

Password: [Change]
External Logins: 0 [Manage]

Account
These fields enable you to update your profile settings used in LiftLog. You may also delete your account completely.

DisplayName: MuscleMaster
Name:
Birth Date: 01-Jan-0001
BodyWeight: 0
Height: 0
Sex: Unknown
Country: Unknown

Consent
These options, if enabled, give LiftLog the right to process parts of your personal data required for that particular feature to work.

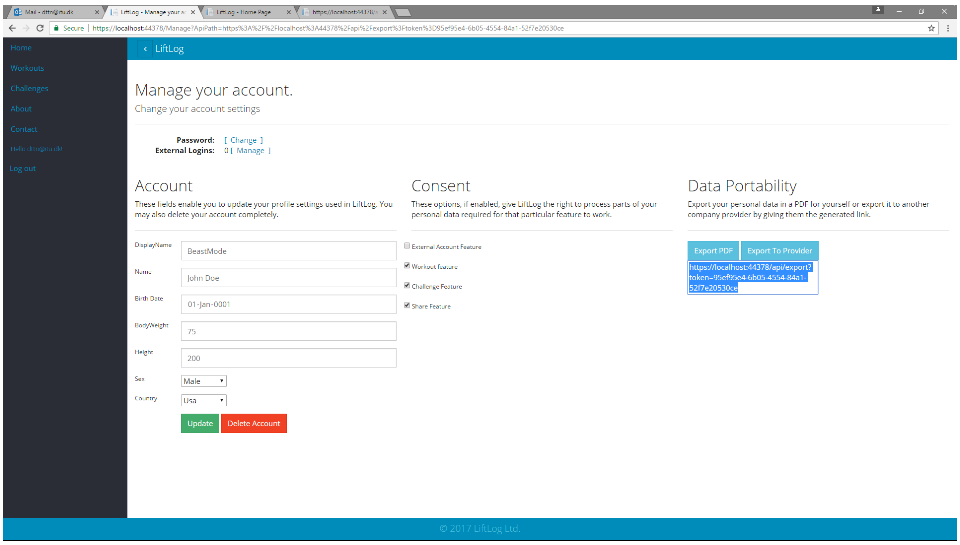
☐ External Account Feature
☑ Workout feature
☑ Challenge Feature
☑ Share Feature

Data Portability
Export your personal data in a PDF for yourself or export it to another company provider by giving them the generated link.

The export link will appear here...

Figure 21: Delete Account pop-up

Export Data To export all personal data, one can either export it as a PDF file for personal use or as a link that can be used to access all personal data in a machine readable format (JSON), which can be found in the ”Data Portability section”. To export as PDF, the user must press the blue ”Export PDF” button, which will then prompt the user where to save the PDF file. To get the export link, one must press the blue ”Export to provider” button. This will generate a link in the text box below the two blue buttons. Note that the link can only be used once and therefore one must generate a new link for each usage. This is illustrated in figure 22.


Mail - dttn@itu.dk | LiftLog - Manage your ac | LiftLog - Home Page | https://localhost:44378/a

Secure | https://localhost:44378/Manage?ApiPath=https%3A%2F%2Flocalhost%3A44378%2Fapi%2Fexport%3Ftoken%3D95ef95e4-6b05-4554-84a1-52f7e20530ce

Home
Workouts
Challenges
About
Contact
Hello dttn@itu.dk!
Log out

LiftLog

Manage your account.
Change your account settings

Password: [Change]
External Logins: 0 [Manage]

Account
These fields enable you to update your profile settings used in LiftLog. You may also delete your account completely.

DisplayName	BeastMode
Name	John Doe
Birth Date	01-Jan-0001
BodyWeight	75
Height	200
Sex	Male
Country	Usa

Update | Delete Account

Consent
These options, if enabled, give LiftLog the right to process parts of your personal data required for that particular feature to work.

- ☐ External Account Feature
- ✓ Workout feature
- ✓ Challenge Feature
- ✓ Share Feature

Data Portability
Export your personal data in a PDF for yourself or export it to another company provider by giving them the generated link.

Export PDF | Export To Provider

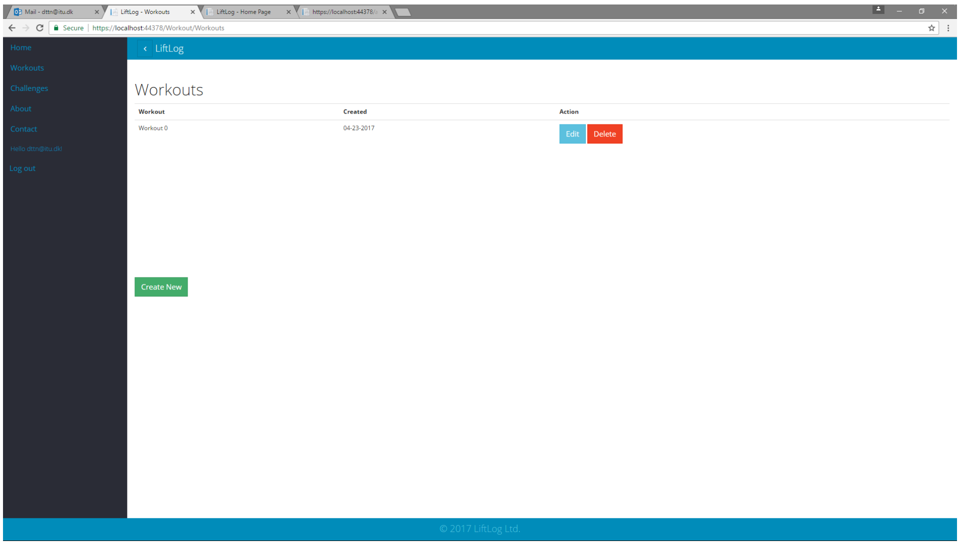
https://localhost:44378/api/export?token=95ef95e4-6b05-4554-84a1-52f7e20530ce

© 2017 LiftLog Ltd.

Figure 22: Generated Export Link

Workout Management

To manage or create new workouts, one must navigate to the workout page by pressing on the "Workouts" button on the navigation panel. The page allows a user to create, edit or delete a workout as illustrated in figure 23.


Mail - dttn@itu.dk | LiftLog - Workouts | LiftLog - Home Page | https://localhost:44378/a

Secure | https://localhost:44378/Workout/Workouts

Home
Workouts
Challenges
About
Contact
Hello dttn@itu.dk!
Log out

LiftLog

Workouts

Workout	Created	Action
Workout 0	04-23-2017	Edit Delete

Create New

© 2017 LiftLog Ltd.

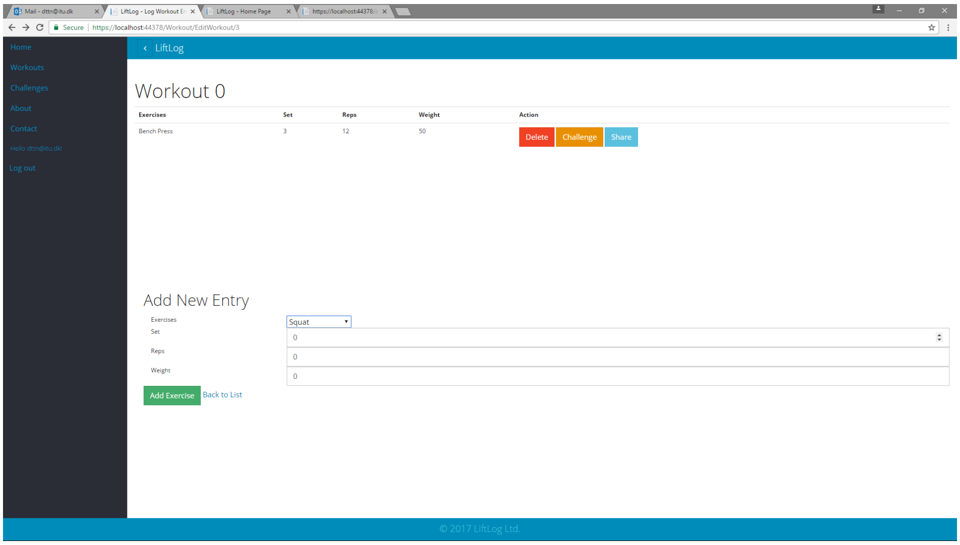
Figure 23: Workout page

Creating a workout To create a workout, press the green "Create New" button. The user will then be redirected to a new page that allows one to add workout entries. This is explained in subsection 10.8.

Deleting a workout To remove a workout, press the red "Delete" button next to the given workout that is to be removed.

Workout Entries Management

To modify a given workout and add or remove entries, press the blue "Set" button which is located next to a workout that is to be modified, located on the workout page. When pressed, the user will be redirected to a page that is used to create, remove workouts as well as share entries or challenge users. The workout entry page is illustrated in figure 24.



Workout 0

Exercises	Set	Reps	Weight	Action
Bench Press	3	12	50	Delete Challenge Share

Add New Entry

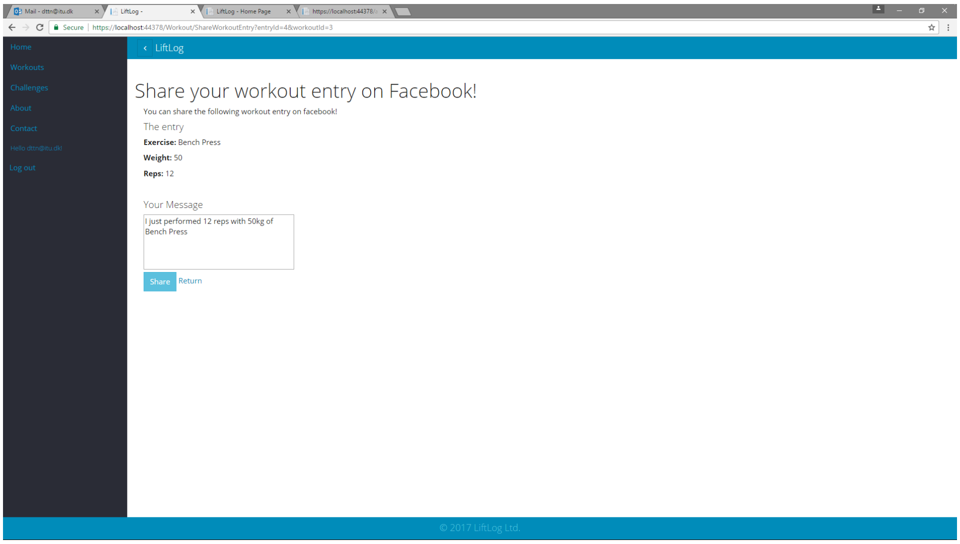
Exercises	Squat
Set	0
Reps	0
Weight	0

Figure 24: Specific workout entry page

Create Entry To create a new workout, one must select an exercise, enter a number of reps, sets and a given weight. After that, the green "Add Exercise" button is pressed to add the new entry to the workout.

Delete Entry Deletion of any entries is done by pressing the red "Delete" button next to the entry that is to be removed.

Share Entry If sharing consent is given, one can share a given challenge by pressing the blue "Share" button next to the entry that is to be shared. The user will then be redirected to a page that allows the to customize a message and see which exercise that are to be shared. This is illustrated in figure 25.



Share your workout entry on Facebook!

You can share the following workout entry on facebook!

The entry

Exercise: Bench Press

Weight: 50

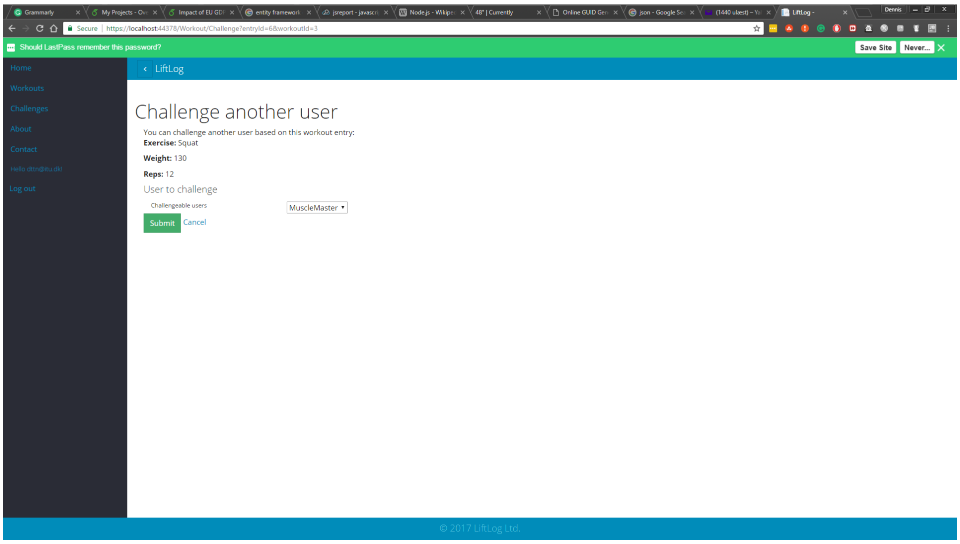
Reps: 12

Your Message

I just performed 12 reps with 50kg of Bench Press

Figure 25: Share workout entry page

Challenge user with entry If challenge consent is given, an entry can be used to challenge another user. By pressing the orange "Challenge" button, the user will then be redirected to a page that allows the user to select another user to be challenged. This page is illustrated in figure 26.



Challenge another user

You can challenge another user based on this workout entry:

Exercise: Squat

Weight: 130

Reps: 12

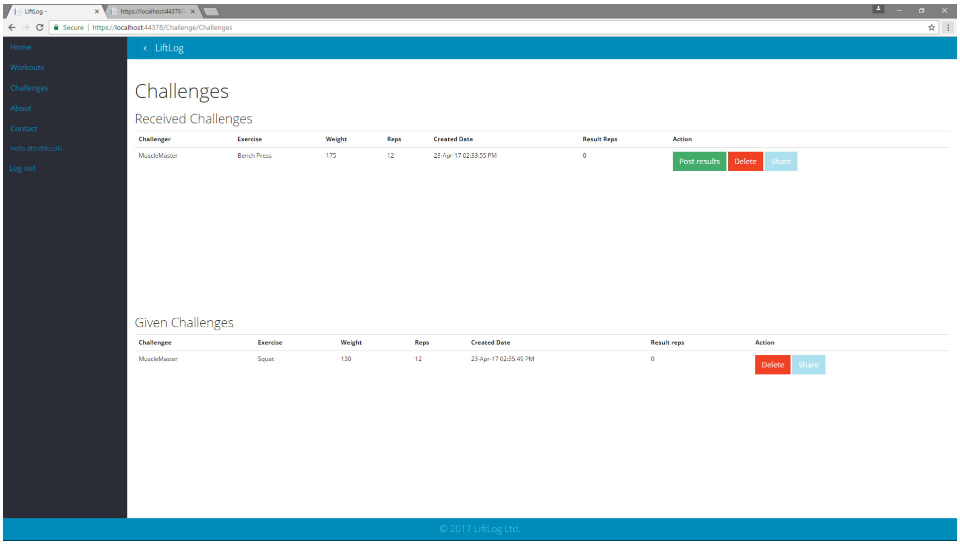
User to challenge

Challengeable users MuscleMaster

Figure 26: Challenge workout entry page

Challenges Management

If challenge consent is given, the user can manage all his or her given or received challenges in the Challenge Management page. This page can be accessed by pressing the ”Challenges” tab in the navigation panel. The page is illustrated in figure 27



Challenges

Received Challenges

Challenger	Exercise	Weight	Reps	Created Date	Result Reps	Action
MuscleMaster	Bench Press	175	12	23-Apr-17 02:33:55 PM	0	Post results Delete Share

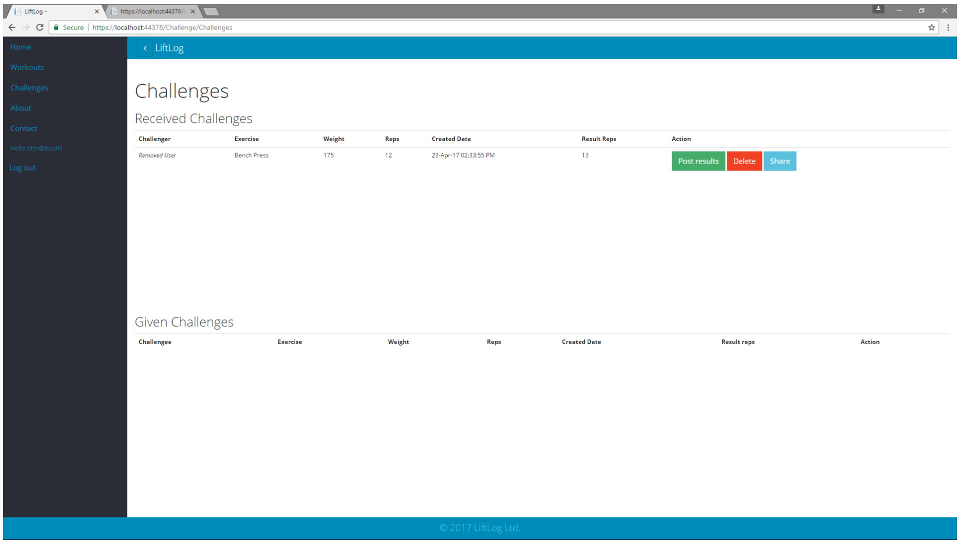
Given Challenges

Challengee	Exercise	Weight	Reps	Created Date	Result reps	Action
MuscleMaster	Squat	130	12	23-Apr-17 02:35:49 PM	0	Delete Share

© 2017 LiftLog Ltd.

Figure 27: Challenge Management page

Delete Challange Any challenge that is given or received can be deleted by pressing the red ”Delete” button next to the given challenge that is to be removed. Note that a given challenge is still accessible by the challengee (i.e. person being challenged) after its deletion. However, the challenger will not be shown in the challenge but a ”removed user” is shown instead. This is depicted in figure 28



Challenges

Received Challenges

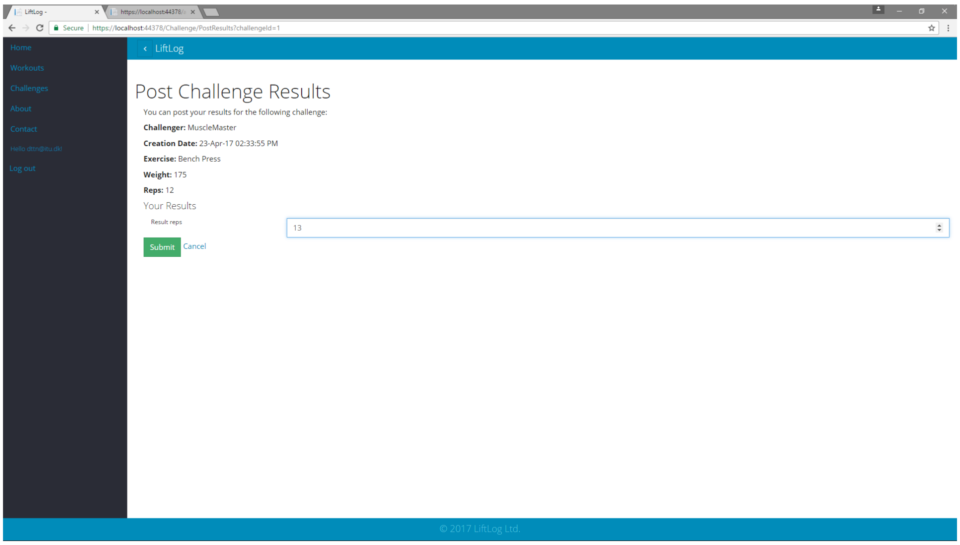
Challenger	Exercise	Weight	Reps	Created Date	Result Reps	Action
Removed User	Bench Press	175	12	23-Apr-17 02:33:55 PM	13	Post results Delete Share

Given Challenges

Challengee	Exercise	Weight	Reps	Created Date	Result reps	Action

Figure 28: Removed user as display name

Post results If the user wishes to post a result for a given received challenge, one must press the green "Post Result" button next to the challenge which results are to be posted. The user will be redirected to a page where the results can be posted. This is illustrated in figure 29.



Post Challenge Results

You can post your results for the following challenge:

Challenger: MuscleMaster

Creation Date: 23-Apr-17 02:33:55 PM

Exercise: Bench Press

Weight: 175

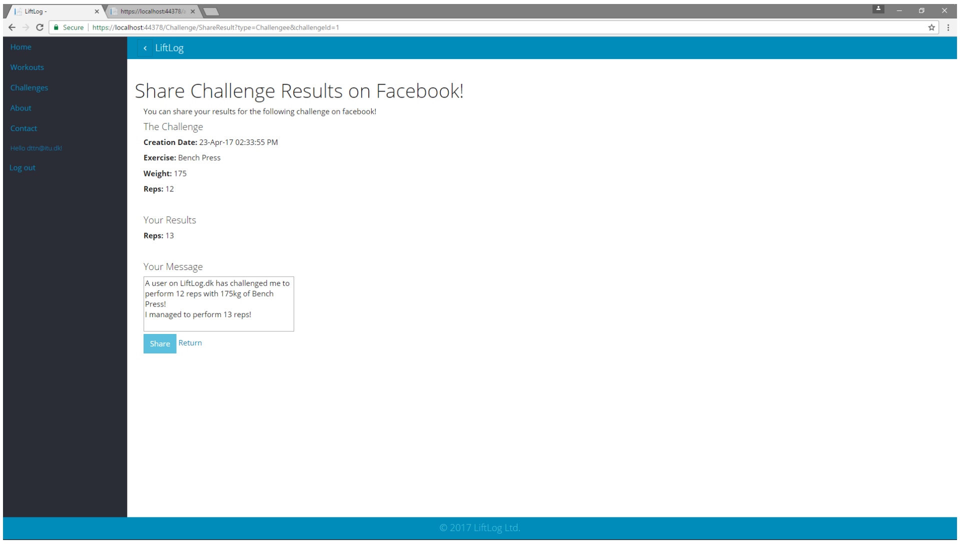
Reps: 12

Your Results

Result reps 13

Figure 29: Post challenge result page

Sharing Challenge results If sharing consent is enabled on both users, the challenger or person being challenged can share the challenge results on the social media. This is done by pressing the blue ”Share” button next to the challenge that is to be shared. The user will then be redirected to a page where the challenge can be confirmed, and a message can be customized. This is depicted in figure 30.



Share Challenge Results on Facebook!

You can share your results for the following challenge on facebook!

The Challenge

Creation Date: 23-Apr-17 02:33:55 PM

Exercise: Bench Press

Weight: 175

Reps: 12

Your Results

Reps: 13

Your Message

A user on LiftLog.dk has challenged me to perform 12 reps with 175kg of Bench Press!
I managed to perform 13 reps!

Figure 30: Share challenge result page

10.9 Exported report example

"

Liftlog - Your personal data

Description

This document contains all of your current personal data stored in Liftlog.
This document was generated on **Thursday, April 6, 2017 05:40:45 PM**

Personal Data

This sections contains all personal data that describes who you are.

- **Name:** Dennis
- **Birth date:** 06-Apr-17
- **Sex:** Sex
- **Weight:** 65
- **Height:** 178
- **Country:** Denmark

Account Data

This sections contains all information about your account at LiftLog

- **Creation date:** 06-Apr-17
- **Display name:** LeetUser
- **Email:** abc@cde.com

Challenge data

This sections contains information on your challenges. We cannot provide you details on thechallenges because they contain personal data of others as well.But rest assured that they onlycontain an exercise created or received and a reference to another user.

- **Challenge Given:** 1
- **Challenge Received:** 2

Workout Data

This sections contains all your workouts and their entries stored at LiftLog
- **Total workouts:**2

Workout: First ever

- **Creation Date:**06-Apr-17

Entries
- Total Entries: 1

Exercise: Benchpress
- Set: 2
- Weight: 145
- Reps: 12

Workout: Second

- **Creation Date:**06-Apr-17

Entries
- Total Entries: 1

Exercise: Squad
- Set: 1
- Weight: 200
- Reps: 12

"

Liftlog - Your personal data

Description

This document contains all of your current personal data stored in Liftlog. This document was generated on **{DateTime.Now:F}**

Personal Data

This sections contains all personal data that describes who you are.

- **Name:** {user.Name}
- **Birth date:** {user.BirthDay:d}
- **Sex:** {user.Sex}
- **Weight:** {user.BodyWeight}
- **Height:** {user.Height}
- **Country:** {user.Country}

Account Data

This sections contains all information about your account at LiftLog

- **Creation date:** {user.CreationDate:d}
- **Display name:** {user.DisplayName}
- **Email::** {user.Email}

Challenge data

This sections contains information on your challenges. We cannot provide you details on the challenges because they contain personal data of others as well. But rest assured that they only contain an exercise created or received and a reference to another user.

- **Challenge Given:** {user.ChallengeGiven}
- **Challenge Received:** {user.ChallengeReceived}

Workout Data

This sections contains all your workouts and their entries stored at LiftLog

- **Total workouts:** {user.Workouts.Count}

Workout: {workout.Name}

- **Creation Date:** {user.DisplayName}

Entries

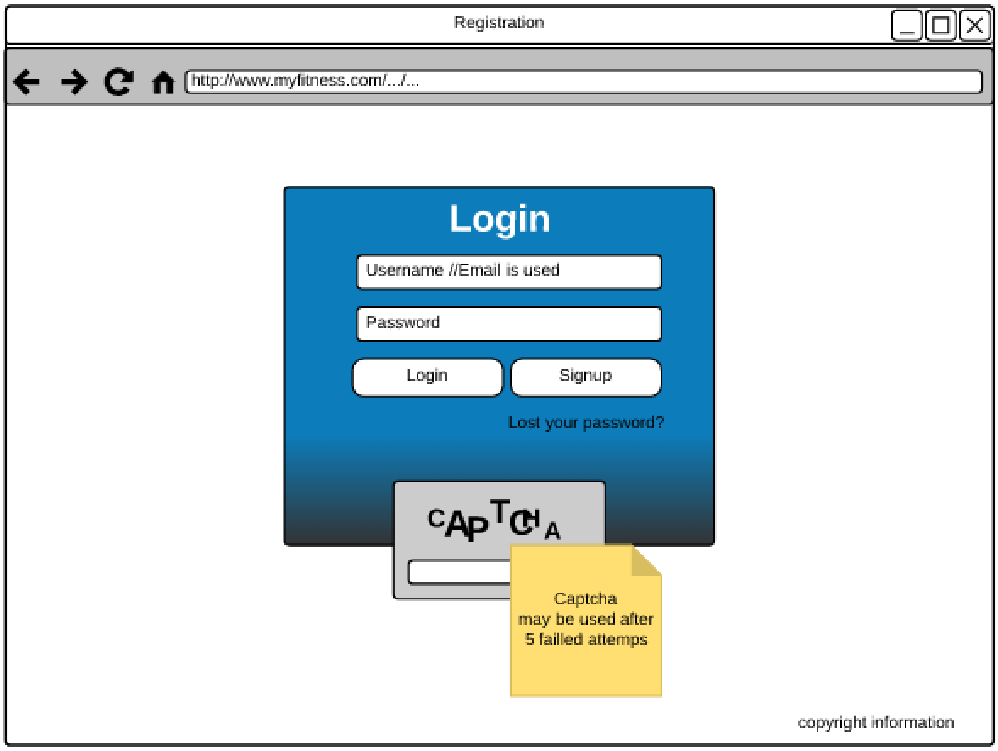
- **Total Entries:** {workout.WorkoutEntryDtos.Count}

Exercise: {WorkoutEntryDto.ExerciseName}

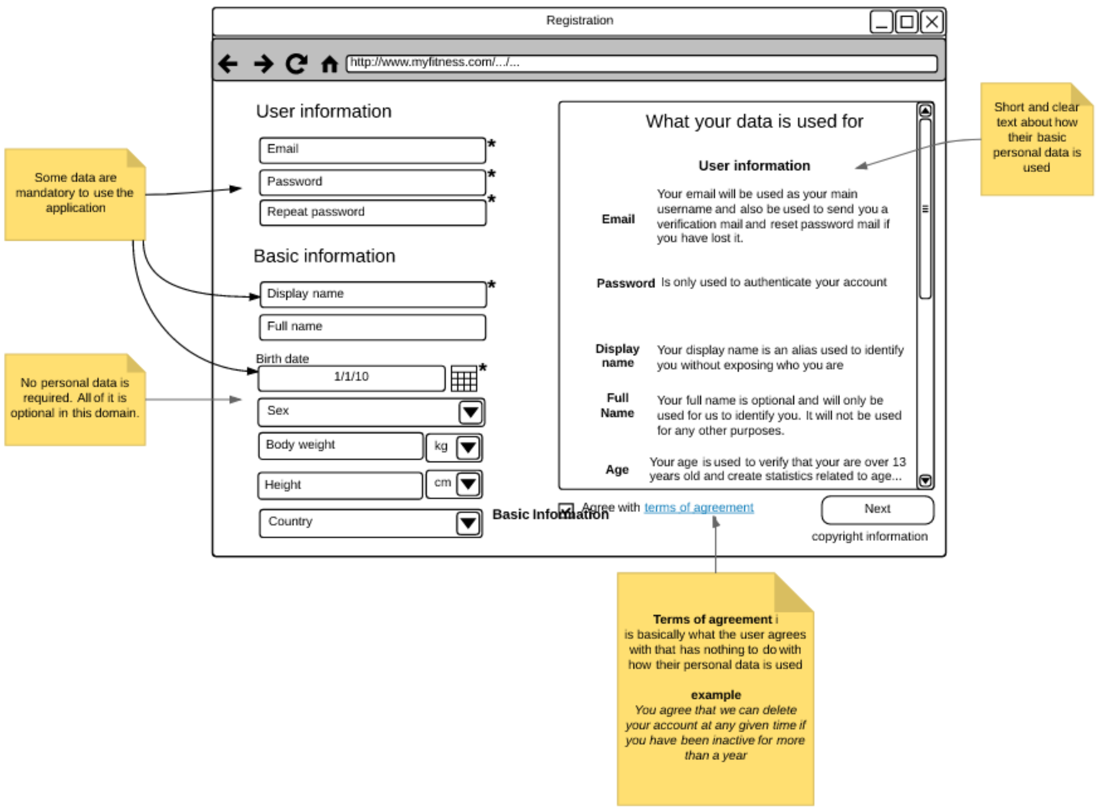
- **Weight:** {workout.WorkoutEntryDtos.Weight}
- **Set:** {workout.WorkoutEntryDtos.Weight}
- **Reps:** {workout.WorkoutEntryDtos.Weight}

10.10 Sample application UI mocks

Mocks -
Login screen



Registrate account screen



What your data is used for

User information

Email	Your email will be used as your main username and also be used to send you a verification mail and reset password mail if you have lost it.
Password	Is only used to authenticate your account
Display name	Your display name is an alias used to identify you without exposing who you are
Full Name	Your full name is optional and will only be used for us to identify you. It will not be used for any other purposes.
Age	Your age is used to verify that your are over 13 years old and create statistics related to age...

Some data are mandatory to use the application

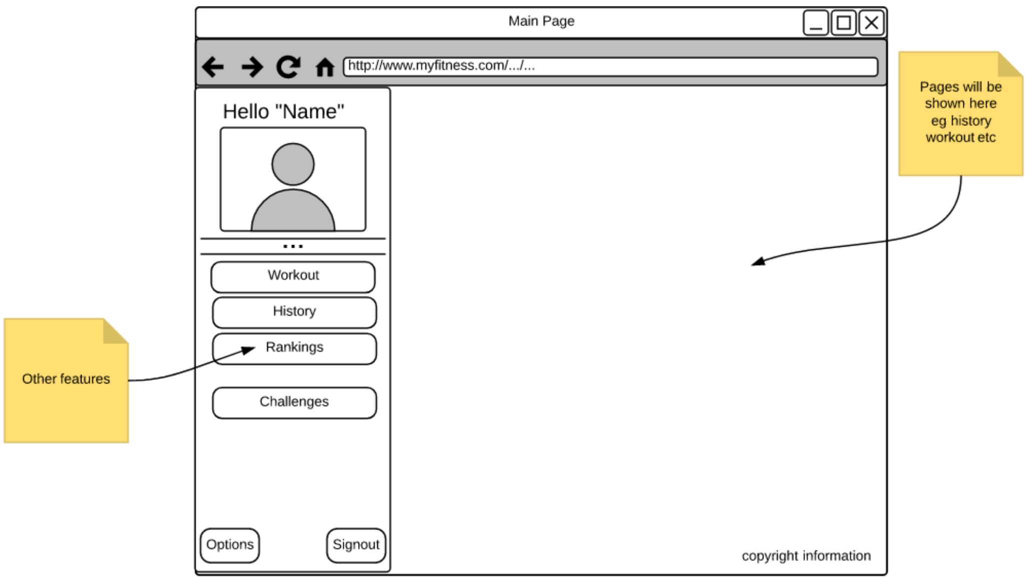
No personal data is required. All of it is optional in this domain.

Short and clear text about how their basic personal data is used

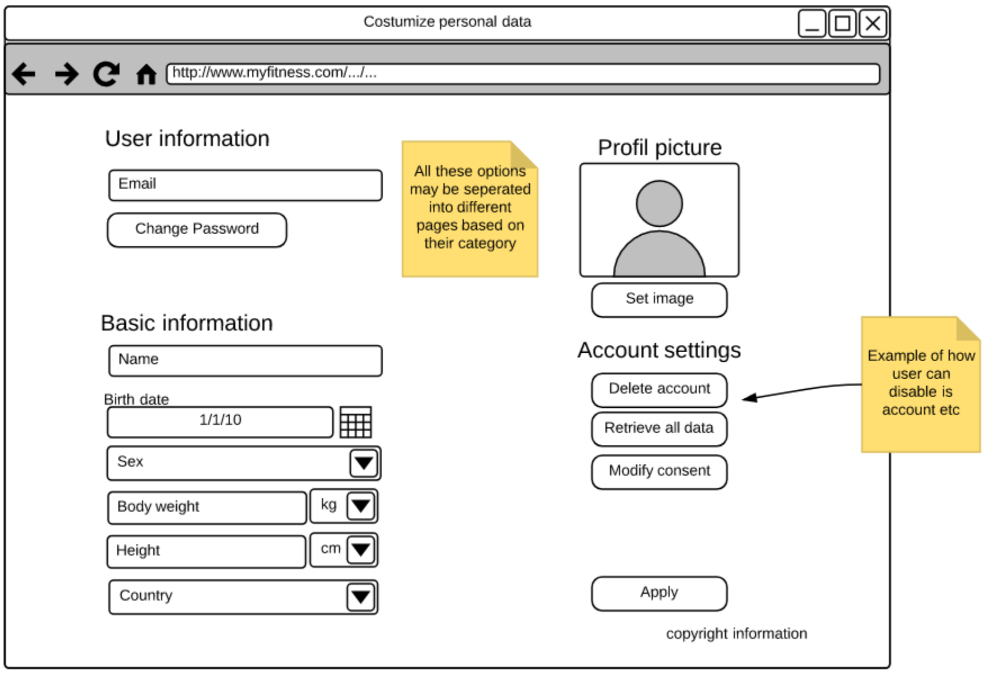
Terms of agreement i is basically what the user agrees with that has nothing to do with how their personal data is used

example
You agree that we can delete your account at any given time if you have been inactive for more than a year

Main page screen

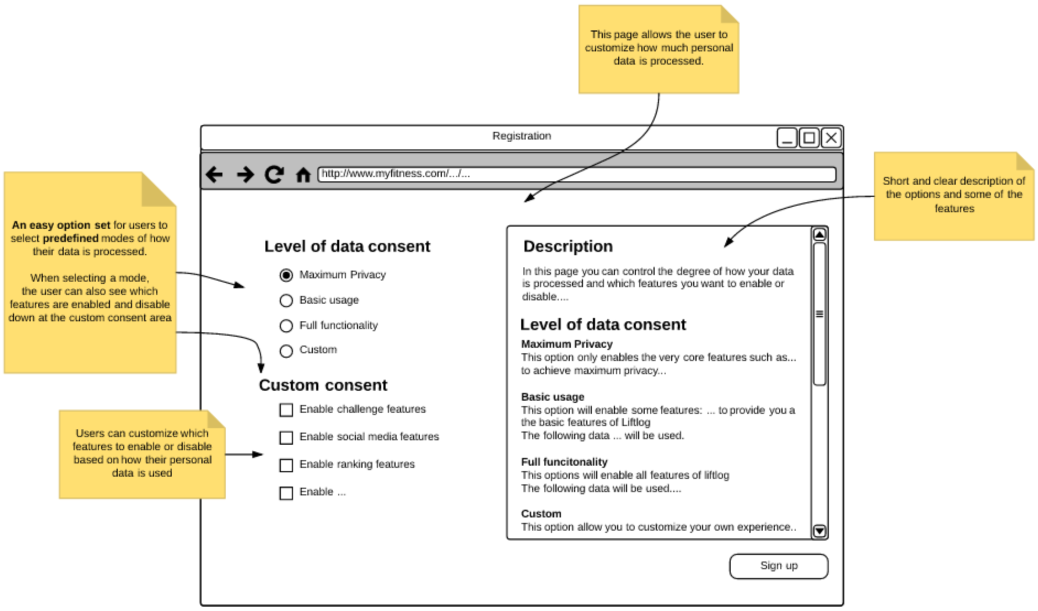


Manage account screen



`

Registrate consent screen



This page allows the user to customize how much personal data is processed.

Registration

http://www.myfitness.com/.../...

An easy option set for users to select **predefined** modes of how their data is processed.

When selecting a mode, the user can also see which features are enabled and disable down at the custom consent area

Level of data consent

- ● Maximum Privacy
- ○ Basic usage
- ○ Full functionality
- ○ Custom

Custom consent

- ☐ Enable challenge features
- ☐ Enable social media features
- ☐ Enable ranking features
- ☐ Enable ...

Users can customize which features to enable or disable based on how their personal data is used

Short and clear description of the options and some of the features

Description

In this page you can control the degree of how your data is processed and which features you want to enable or disable....

Level of data consent

Maximum Privacy
This option only enables the very core features such as... to achieve maximum privacy...

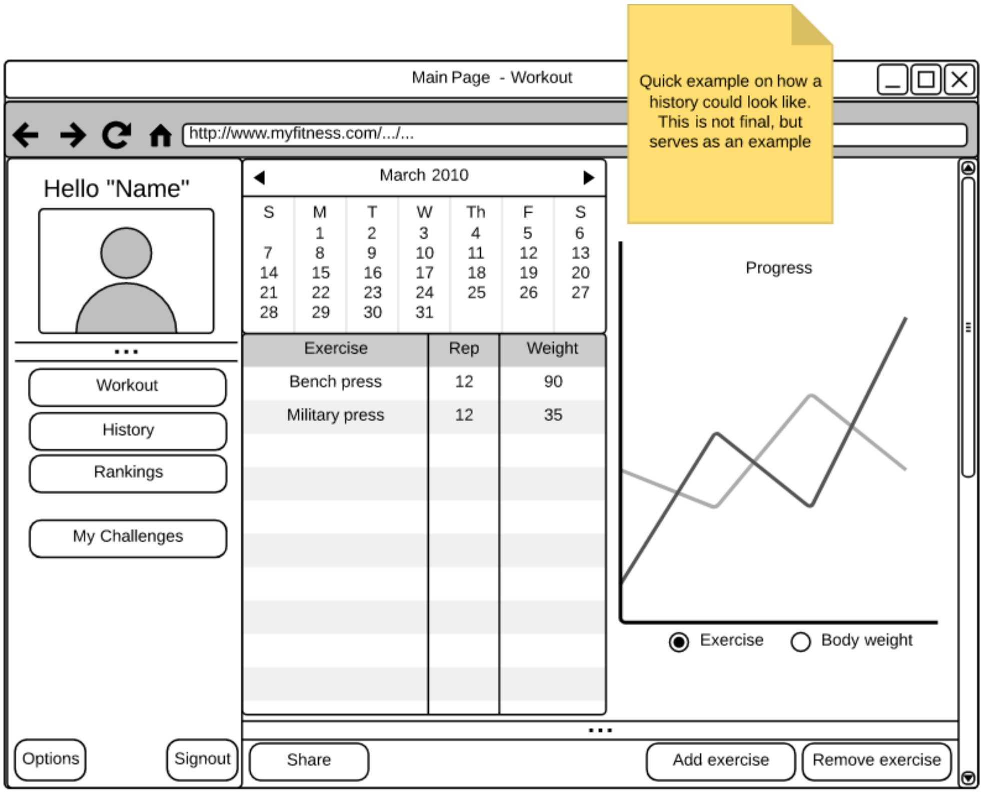
Basic usage
This option will enable some features: ... to provide you a the basic features of Liftlog
The following data ... will be used.

Full funcitonality
This options will enable all features of liftlog
The following data will be used....

Custom
This option allow you to customize your own experience..

Sign up

Workout screen



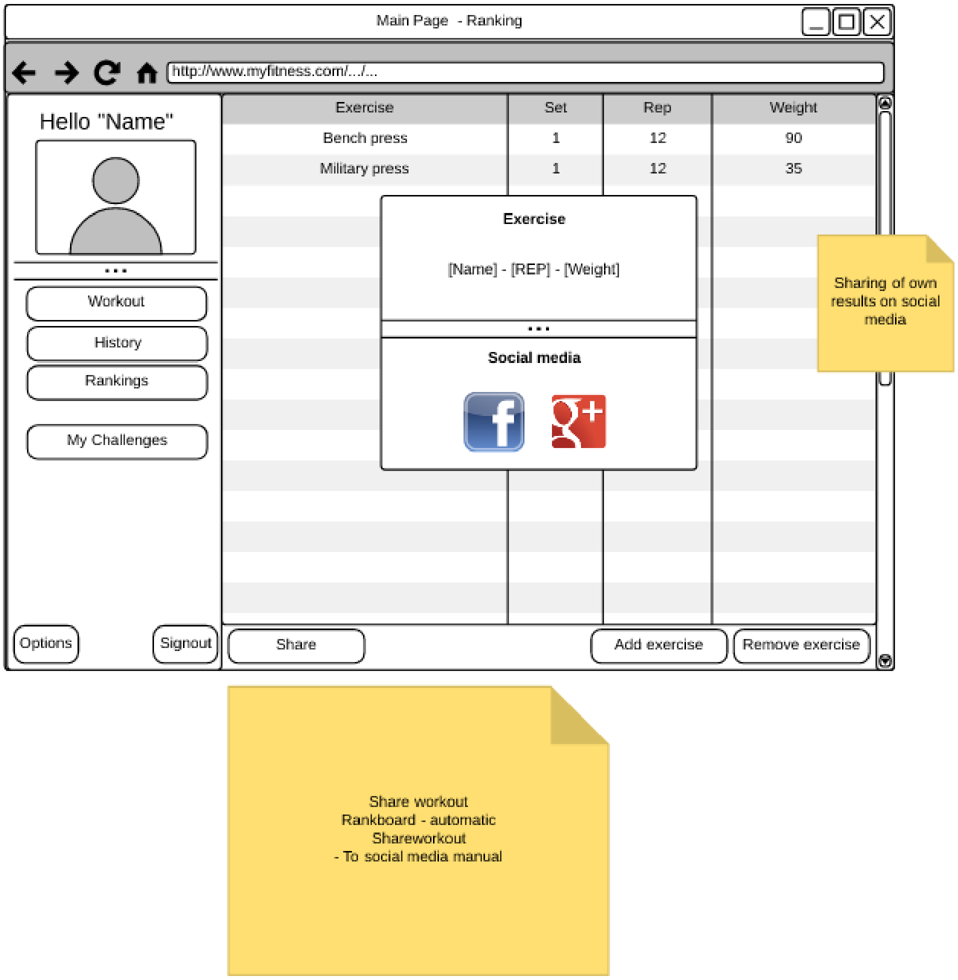
Quick example on how a history could look like. This is not final, but serves as an example

S	M	T	W	Th	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Exercise	Rep	Weight
Bench press	12	90
Military press	12	35

[✓ Exercise] Body weight

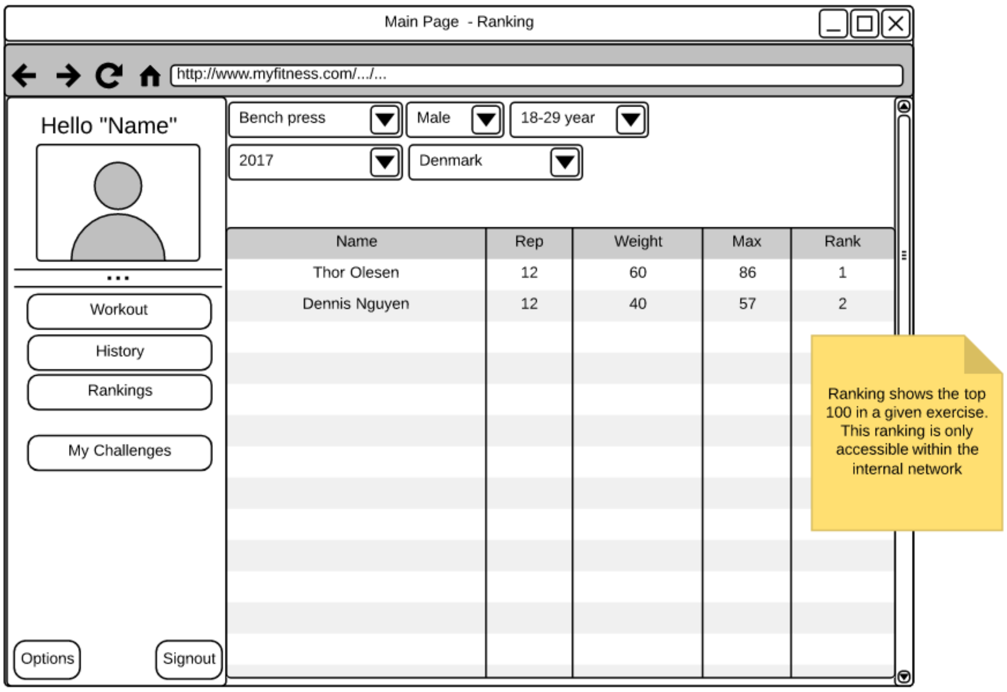
Ranking screen - Share



Exercise	Set	Rep	Weight
Bench press	1	12	90
Military press	1	12	35

Share workout
Rankboard - automatic
Shareworkout
- To social media manual

Ranking screen



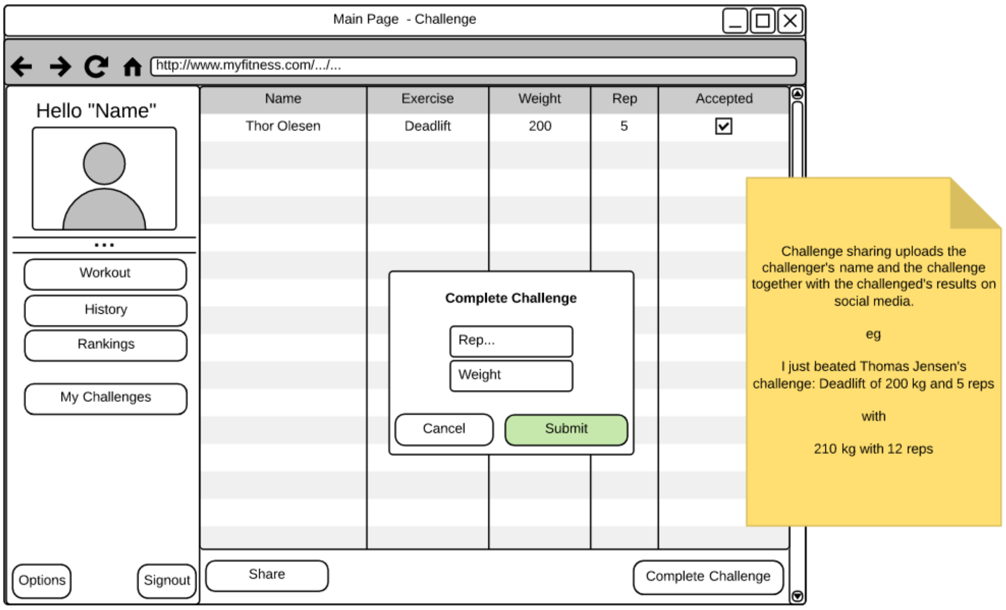
Main Page - Ranking

http://www.myfitness.com/.../...

Name	Rep	Weight	Max	Rank
Thor Olesen	12	60	86	1
Dennis Nguyen	12	40	57	2

Ranking shows the top 100 in a given exercise. This ranking is only accessible within the internal network

Challenge screen – post results



Name	Exercise	Weight	Rep	Accepted
Thor Olesen	Deadlift	200	5	✓

Challenge sharing uploads the challenger's name and the challenge together with the challenged's results on social media.

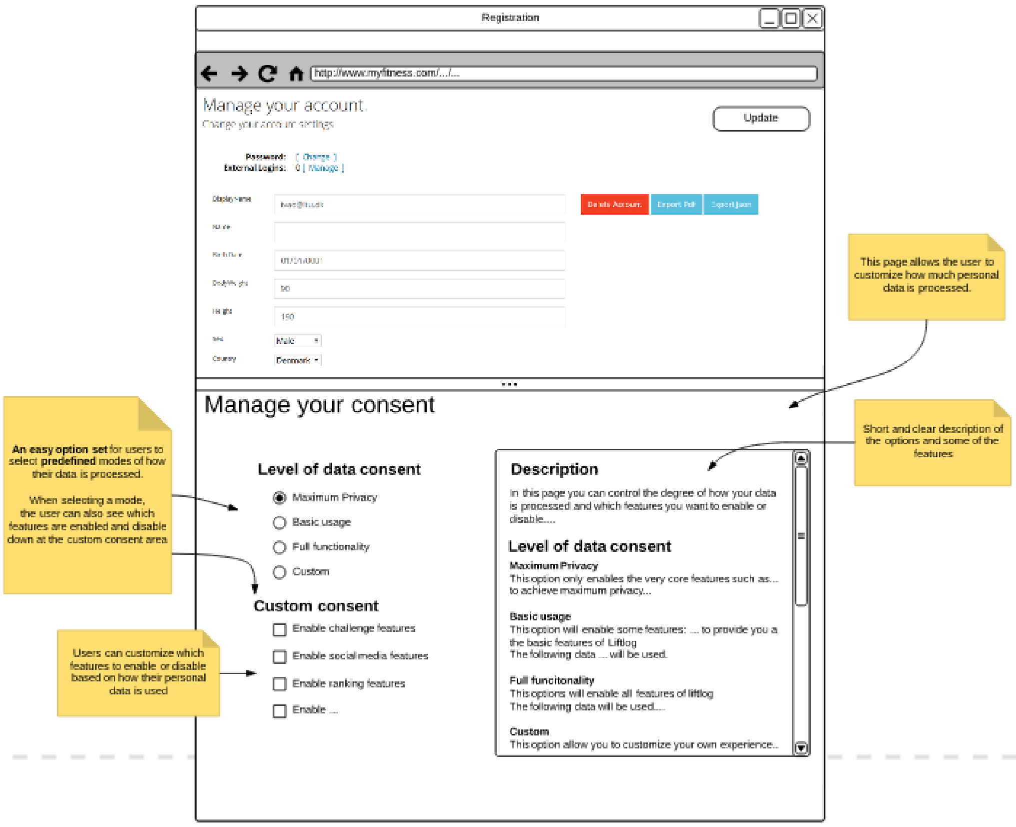
eg

I just beated Thomas Jensen's challenge: Deadlift of 200 kg and 5 reps

with

210 kg with 12 reps

Manage account with consent management



Level of data consent: [✓ Maximum Privacy] Basic usage / Full functionality / Custom

Description

In this page you can control the degree of how your data is processed and which features you want to enable or disable....

Level of data consent

Maximum Privacy
This option only enables the very core features such as... to achieve maximum privacy...

Basic usage
This option will enable some features: ... to provide you a the basic features of Liftlog
The following data ... will be used.

Full funcitonality
This options will enable all features of liftlog
The following data will be used....

Custom
This option allow you to customize your own experience..

This page allows the user to customize how much personal data is processed.

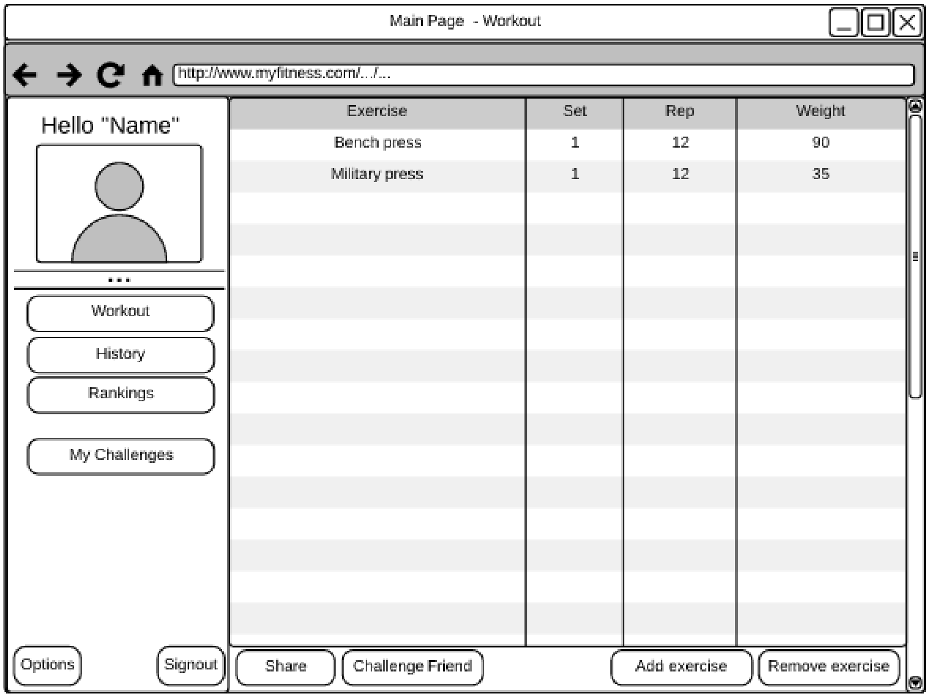
Short and clear description of the options and some of the features

An easy option set for users to select **predefined** modes of how their data is processed.

When selecting a mode, the user can also see which features are enabled and disable down at the custom consent area

Users can customize which features to enable or disable based on how their personal data is used

Workout screen 2



Exercise	Set	Rep	Weight
Bench press	1	12	90
Military press	1	12	35

Share workout
Rankboard - automatic
Shareworkout
- To social media manual

References

[1] Rick Anderson. *Adding Validation.* https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/validation. [Online; accessed 21-April-2017]. 2017.

[2] Rick Anderson. *Tag Helpers.* https://docs.microsoft.com/en-us/aspnet/core/mvc/views/tag-helpers/intro. [Online; accessed 21-April-2017]. 2016.

[3] *Claim-Based Authorization.* https://docs.microsoft.com/en-us/aspnet/core/security/authorization/claims. [Online; accessed 21-April-2017]. 2016.

[4] European Commission. *Protection of personal data.* http://ec.europa.eu/justice/data-protection/. [Online; accessed 27-April-2017].

[5] European Commission. *Questions and Answers - Data protection reform.* http://europa.eu/rapid/press-release_MEMO-15-6385_en.htm. [Online; accessed 27-April-2017]. 2015.

[6] European Commission. *Reform of EU data protection rules.* http://ec.europa.eu/justice/data-protection/reform/index_en.htm. [Online; accessed 27-April-2017].

[7] Rick Anderson Daniel Roth and Shaun Luttin. *Introduction to ASP.NET Core.* https://docs.microsoft.com/en-us/aspnet/core/. [Online; accessed 20-April-2017]. 2016.

[8] Michael Schläpfer David Basin Patrick Schaller. *Applied Information Security.* Springer-Verlag, 2011.

[9] Søren Debois. “Introduction - Security Principles and Security Goals”. In: Security (BSc), IT University of Copenhagen. 2017.

[10] Steve Gordon. *ASP.NET Identity Core (Under the Hood).* https://www.stevejgordon.co.uk/asp-net-identity-core-under-the-hood-part-1. [Online; accessed 21-April-2017]. 2016.

[11] Andrew Lock. *Introduction to Authentication with ASP.NET Core.* https://andrewlock.net/introduction-to-authentication-with-asp-net-core/. [Online; accessed 21-April-2017]. 2016.

[12] THE EUROPEAN PARLIAMENT and THE COUNCIL OF THE EUROPEAN UNION. *General Data Protection Regulations.* Official Journal of the European Union, 2016.

[13] Tom Dykstra Pranav Rastogi Rick Anderson. *Introduction to Identity.* https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity. [Online; accessed 20-April-2017]. 2016.

[14] Daniel Roth Rick Anderson and Scott Addie. *Introduction to using tag helpers in forms in ASP.NET Core.* `https://docs.microsoft.com/en-us/aspnet/core/mvc/views/working-with-forms\#the-validation-tag-helpers`. [Online; accessed 21-April-2017]. 2017.

[15] Daniel Roth Rick Anderson and Scott Addie. *Safe storage of app secrets during development.* `https://docs.microsoft.com/en-us/aspnet/core/security/app-secrets`. [Online; accessed 21-April-2017]. 2016.

[16] Steve Smith Rick Anderson Mark Michaelis and Daniel Roth. *Configuration in ASP.NET Core.* `https://docs.microsoft.com/en-us/aspnet/core/fundamentals/configuration`. [Online; accessed 21-April-2017]. 2016.

[17] *Role based Authorization.* `https://docs.microsoft.com/en-us/aspnet/core/security/authorization/roles`. [Online; accessed 21-April-2017]. 2016.

[18] *Role-Based vs Claim-based Authorization on Stack.* `https://stackoverflow.com/questions/22814023/role-based-access-control-rbac-vs-claims-based-access-control-cbac-in-asp-n`. [Online; accessed 21-April-2017]. 2016.

[19] Martin Milan Rowan Miller. *Entity Framework Core.* `https://docs.microsoft.com/en-us/ef/core/index`. [Online; accessed 21-April-2017]. 2016.

[20] *Security.* `https://docs.microsoft.com/en-us/aspnet/core/security/`. [Online; accessed 20-April-2017]. 2016.

[21] *Simple Authorization.* `https://docs.microsoft.com/en-us/aspnet/core/security/authorization/simple`. [Online; accessed 21-April-2017]. 2016.

[22] Steve Smith and Tom Dykstra. *Application Startup in ASP.NET Core.* `https://docs.microsoft.com/en-us/aspnet/core/fundamentals/startup`. [Online; accessed 21-April-2017]. 2017.

[23] Ken Schwaber & Jeff Sutherland. “The Scrum Guide”. In: (July 2013).

[24] *View Based Authorization.* `https://docs.microsoft.com/en-us/aspnet/core/security/authorization/views`. [Online; accessed 21-April-2017]. 2016.

[25] Wikipedia. *Adversary (cryptography).* [Online; accessed 21-April-2017]. 2017. URL: `%5Curl%7Bhttps://en.wikipedia.org/wiki/Adversary_(cryptography)%7D`.

[26] Wikipedia. *Defense in depth (computing).* `https://en.wikipedia.org/wiki/Defense_in_depth_(computing)`. [Online; accessed 13/04/17]. 2017.

[27] Clyde Williamson. *PSEUDONYMIZATION VS. ANONYMIZATION AND HOW THEY HELP WITH GDPR.* `http://www.protegrity.com/pseudonymization-vs-anonymization-help-gdpr`. [Online; accessed 26-April-2017]. 2017.

[28] Clyde Williamson. *PSEUDONYMIZATION VS. ANONYMIZATION AND HOW THEY HELP WITH GDPR.* `http://www.protegrity.com/pseudonymization-vs-anonymization-help-gdpr/`. [Online; accessed 11-May-2017]. 2017.

[29] Afzaal Ahmad Zeeshan. *Hashing Passwords in .NET Core With Tips.* `http://www.c-sharpcorner.com/article/hashing-passwords-in-net-core-with-tips/`. [Online; accessed 21-April-2017]. 2016.