Project description:

The application (called SecFit) is a web application to manage your personal fitness life. The application allows users to define exercises and compose workouts. Additionally, users may communicate with fitness coaches through this application. The coaches can use this application to manage workouts for their assigned athletes. A detailed description of functional requirements, roles, and permissions is below.

Functional Requirements:

Users/Roles: A user is a coach if they have one or more athletes on their roster. A user is an athlete if they have a coach. It is possible for a user to be both an athlete and a coach.

- Athlete

An athlete logs daily workouts consisting of one or more exercises. Exercises can either be selected a from a list or created (both on-the-spot and in general). In addition to exercises, the user can enter notes about the workout and upload images and videos. The athlete must all specify the visibility of their workouts (private, visible to coach, public). An athlete can designate another user as their coach at will and communicate with that user via comments on workouts. An athlete can view the details of and comment on public workouts.

- Coach

A coach can view, comment on, and react to their athlete's workouts (assuming the visibility is not private). A coach can also upload files (pdfs) for each of their athletes, such as to present a training plan. A coach can also view the details of and comments on other athletes' (i.e., not their own) public workouts. In order for the coach to have an athlete, the coach has to accept an athlete's request or send a request to an athlete.

Function Requirements ID	Title	Description
FR1	Sign up and sign in	A user must be able to sign in to an authorized session and gain rights to log workouts, create exercises, leave comments, and upload files.
FR2	Create/log workout	The athlete should be able to create/log a workout consisting of a date, name, zero or more exercises, and notes. A visibility level (Public, Coach, or Private) must also be set. Each exercise will have one or more sets of some number of units.
FR3	Upload images and video to workout	The athlete should be able to upload images and/or videos to their workouts.
FR4	Edit workout	The athlete should be able to edit all details of their workouts after they have been created and saved.
FR5	View workout	The user should be able to view all of the details, files, and comments on workouts of sufficient visibility. For athletes, this means that the workout needs to be either their own or public. For coaches, this means that the workout is at least one of their athletes' non-private workouts OR the workout is public. For visitors, this means that the workout needs to be public.
FR6	Create exercise	The user should be able to create a new exercise both inside and outside of workouts. An exercise consists of a name, description, and unit (e.g., seconds, km, reps).
FR7	Upload documents	The coach should be able to upload documents for each of his athletes.
FR8	View coach documents	The athlete should be able to view all of the documents that the coach has uploaded for them.
FR9	Leave comment	The athlete should be able to comment on their own workouts. The coach should be able to comment on their athletes' nonprivate workouts.
FR10	View list of workouts	The athlete should be able to view a list of all of their own work outs. The athlete should be able to view the workouts of the coach.
FR11	View list of athletes' workouts	The coach should be able to view a list of all of their athletes workouts.
FR12	View all public workouts	The user should be able to view a list of all public workouts.
FR13	Filter workouts	The user should be able to filter out workouts from each of the aforementioned lists.

FR14	Sort workouts	The user should be able to sort workouts (e.g., by date or name).
FR15	View athletes/roster	The coach should be able to view a list of all of their athletes.
FR16	Add athlete to roster	The coach should be able to add another athlete by sending a request to that user.
FR17	Remove athlete from roster	The coach should be able to remove athletes.
FR18	Accept/decline coach's offer	The athlete should be able to accept/decline a coach's offer.
F19	Delete athletes	The coach shall be able to delete the athletes from the whole system.
FR20	Set coach	The athlete should be able to specify a coach.
FR21	Remove coach	The athlete should be able to remove a user as their coach.
FR22	Register new user	The user should be able to register a new account with personal information, a username, e-mail address, and password.
F23	Coach permission	The athlete should be able to specify a coach without coach permission.

Exercise 1 – Requirements Engineering (30 points)

Purpose of the exercise:
This exercise is designed to let you practice requirement analysis and learn how to write a good requirement description. Last but not least, the exercise will help to practice the implementation of sustainability at the requirement level.

Exercise design and tasks
Task 1: Quality of requirement specification (10 points)

The table provides a brief description of the quality attributes of a requirement. Using this checklist, your task is to review each of the requirements to identify any problems. For a found problem, explain why it is a problem, and provide a fixed version of the requirement specification.

Attribute	Definition	Yes	No	N/A
Complete	**Is the requirement fully specified and allows the developer to implement it?**			
Correct	**Does the requirement reflect what the user, client, or their representatives desire?**			
Consistent	**Is the requirement consistent with other requirements of the system?**			
Unambiguous	**Does the requirement not contain ambiguities that lead stakeholders to interpret it differently?**			
Verifiable	**Is the requirement possible to verify after its implementation?**			

Assessment criteria:

1. ***Number of issues detected***
2. ***The rationale given from the issues***

Task 1: Solution

We wrote in *italics* the sections in the corrections that are suggested or examples of how the behavior could be. The idea is to represent what is missing by giving a possible interpretation of the requirement

FR1. Sign up and sign in : A user must be able to sign in to an authorized session and gain rights to log workouts, create exercises, leave comments, and upload files.

Complete (NO); Correct (YES); Consistent (YES); Unambiguous (NO); Verifiable (NO)

a) Not complete: it doesn't have any details of the desired authentication, doesn't define the Sign-up functionality and doesn't mention security measures.
b) Ambiguous: "upload files", "rights", "authorized session" are vague phrases that can be interpreted differently
c) Not verifiable: since it is not very concrete and doesn't state behaviors, it is not easy to verify.

Solution: A user must be able to sign up by creating a *unique* username and a password *without restrictions*, linked to a *single* email account (*each email can be associated with only one account*). To complete the registration, the user must verify the account *via an email* authorization process.
Once verified, the user can sign in using the registered username and password. Upon successful authentication, the user gains permissions to log workouts (FR2), create exercises (FR2), leave comments (FR9) and upload files (FR7).

FR2.Create/Log workout: The athlete should be able to create/log a workout consisting of a date, name, zero or more exercises, and notes. A visibility level (Public, Coach, or Private) must also be set. Each exercise will have one or more sets of some number of units.

Complete (NO); Correct (YES); Consistent (YES); Unambiguous (NO); Verifiable (YES)
a) Not complete: It is almost complete, but it lacks the information necessary for the implementation like character limits, allowed date formats…
b) Ambiguous: The "units" can be considered vague, the "notes" could be considered optional or not and there is no explanation for the behavior of the different visibility levels which could be misleading.

Solution: An athlete must be able to create and log a workout by providing a date (scheduled or completed date of workout), a *unique* name (max 50 characters), zero or more exercises (max 50 exercises) consisting in one or more sets of a specified number of units (*repetitions, duration, weight*) and a visibility level of the following:

- **Public:** Visible to all users.
- **Coach:** Only visible to assigned coaches or designated users.
- **Private:** Only visible to the athlete.

Additionally, the user can *optionally* add text in a field “notes”.

FR3. Upload images and video to workout: The athlete should be able to upload images and/or videos to their workouts.

Complete (NO); Correct (YES); Consistent (YES); Unambiguous (NO); Verifiable (NO)
a) Not complete: there isn’t a specification of when or where can the user do this, which can lead to mistakes in the implementation
b) Ambiguous: it also doesn’t specify what types of images and videos it accepts, how many, if there are any visibility settings…
c) Without size limit it can be hard to test.

Solution: An athlete can upload from one to ten pictures and/or videos with formats *PNG, JPEG or MP4* with a maximum of *100 MB* to a previously created workout, with the *same visibility properties* as the workout which it has been added to.

FR4. Edit workout: The athlete should be able to edit all details of their workouts after they have been created and saved.

Complete (NO); Correct (YES); Consistent (YES); Unambiguous (NO); Verifiable (YES)
a) Not complete: it doesn’t specify if there are any restrictions on editing like if workouts with comments or uploaded media can be modified. It also does not mention if changes are logged or if previous versions can be restored.
b) Ambiguous: it’s all mostly clear, but "all details" could be considered ambiguous as it’s unclear if it includes media (FR3) or visibility settings.

Solution: An athlete can edit all details of a previously created workout, including name, date, exercises, sets, notes, and visibility settings. Uploaded media (FR3) *can also be modified*, with the ability to *add, remove, or replace files*. Changes are *saved and there is no track* of previous modifications or versions.

FR5. View workout: The user should be able to view all of the details, files, and comments on workouts of sufficient visibility.
For athletes, this means that the workout needs to be either their own or public.
For coaches, this means that the workout is at least one of their athletes' non-private workouts OR the workout is public.
For visitors, this means that the workout needs to be public.

Complete (NO); Correct (YES); Consistent (YES); Unambiguous (NO); Verifiable (YES)
a) Not complete: the textual description is not enough to represent what the requirement really means to.
b) Ambiguous: the definition of “visitor” and “sufficient visibility” is unclear.

Solution**:** The user should be able to view all of the details, files, and comments on workouts following the table:

User Type	Own Workouts	Public Workouts	Other Athletes' Workouts
Athlete	Yes	Yes	No
Coach	Yes	Yes	Only for assigned athletes
Visitor	No (they don't have)	Yes	No

A visitor is defined as an *unauthenticated user or a registered user without an athlete/coach* role.

FR6. Create exercise: The user should be able to create a new exercise both inside and outside of workouts. An exercise consists of a name, description, and unit (e.g., seconds, km, reps).

Complete (NO); Correct (YES); Consistent (YES); Unambiguous (NO); Verifiable (NO)
a) Not complete: it doesn't say if exercises created outside of a workout are saved for future use, if they are private or shareable, or if there are constraints on name length, description length, or unit types.
b) Ambiguous: It is unclear whether users can edit or delete exercises after creation and it doesn't specify who can create exercises (athletes, coaches, admins).
c) Not verifiable: without constraints on values and behaviors test cases cannot be fully defined.

Solution: Users can create exercises either inside a workout (linked only to that workout) or outside a workout (*saved for reuse in the user's personal library*). An exercise consists of a name (*max 50 chars*), description (*max 500 chars*) and a unit selected from *predefined options* (seconds, km, reps, kg, etc. *Users can edit or delete exercises they have created, but changes do not affect exercises already logged in past workouts*.

FR7. Upload documents: The coach should be able to upload documents for each of his athletes.
Complete (NO); Correct (YES); Consistent (YES); Unambiguous (NO); Verifiable (NO)
a) Not complete: it doesn't specify where these documents are stored, if they are linked to specific workouts or an athlete's profile, and if there are any restrictions on file types, sizes, or number of uploads.
b) Ambiguous: the visibility level and the definition of "document" is unclear.
c) Not verifiable: Without constraints on file size, format, and visibility rules, it is difficult to create test cases to verify.
Solution: Coaches can upload, *replace or delete* documents for their athletes. Documents (*PDF, DOCX and TXT, maximum 10 MB per file*) are stored in the *athlete's profile* and are only visible *to the athlete and the coach* who uploaded them.

FR8. View coach documents: The athlete should be able to view all of the documents that the coach has uploaded for them.

Complete (NO); Correct (YES); Consistent (YES); Unambiguous (NO); Verifiable (YES)
a) Not complete: It could be considered complete if that was exactly all it does, but it would be better to specify whether athletes can download or filter documents, if they receive a notification when a new document is uploaded…
b) Ambiguous: It is unclear if an athlete can still access documents if their coach is removed from the system or if they are reassigned to a different coach.
Solution: Athletes can view and *download* all documents uploaded by their assigned coach/*coaches*. Documents are listed in a *dedicated section, sorted by date*. If a coach deletes a document, it is *immediately removed from the athlete's view*.

FR9. Leave comment: The athlete should be able to comment on their own workouts. The coach should be able to comment on their athletes' non-private workouts.

Complete (NO); Correct (YES); Consistent (YES); Unambiguous (YES); Verifiable (YES)
a) Not complete: It doesn't say if comments can be edited or deleted by the user who posted them. It also does not mention if multiple coaches can comment or if there are any character limits.
Solution: Athletes can comment on their own workouts and coaches can comment on any non-private workout of their assigned athletes. Comments are limited to 500 characters and users can't edit but they can delete their own comments. If a workout's visibility changes, coach comments are hidden

FR10. View list of workouts: The athlete should be able to view a list of all of their own workouts.The athlete should be able to view the workouts of the coach.

Complete (YES); Correct (YES); Consistent (NO); Unambiguous (NO); Verifiable (YES)
a) Not consistent: This contradicts FR5, which specifies workout visibility rules. Athletes can view their own workouts and public ones, but nothing was mentioned about accessing workouts of a coach
c) Ambiguous: The phrase "view the workouts of the coach" is unclear, it could both mean workouts that the coach created for the athlete or personal workouts that belong to the coach.

Solution: Athletes can view a list of their own workouts. Athletes *can't view private workouts* of their coach, but if a coach *assigns a workout to an athlete*, it will appear in their list.

FR11. View list of athletes' workouts: The coach should be able to view a list of all of their athletes' workouts.

Complete (YES); Correct (YES); Consistent (YES); Unambiguous (YES); Verifiable (YES)
This could be considered not complete or ambiguous if read standalone, but knowing FR5 and FR2, it could be understood correctly by everyone.

FR12. View all public workouts: The user should be able to view a list of all public workouts.

Complete (YES); Correct (YES); Consistent (YES); Unambiguous (NO); Verifiable (YES)
a) Ambiguous: In this case, "user" is vague since it has never been cleared out the definition of visitor. In case it had been defined previously as part of “user” or not, then this wouldn’t be ambiguous.
Solution: All users (*including visitors*, athletes, and coaches) can view a list of public workouts. Public workouts do *not* display private athlete or coach details, only workout-specific information.

FR13. Filter workouts: The user should be able to filter out workouts from each of the aforementioned lists.

Complete (NO); Correct (YES); Consistent (YES); Unambiguous (YES); Verifiable (NO)
a) Not complete: “aforementioned” is not defined and the requirement should specify which filter criteria are available (by date, by athlete, by visibility, by number of exercises…). It also doesn’t define if users can apply multiple filters at the same time, for example.
b) Not verifiable: without the definition of the lists or the filtering methods it can’t be easily tested.
Solution: Users can filter the lists of workouts (own - FR10, athlete’s - FR11 and public -FR12) based on date, visibility (public, private, coach), number of exercises, and athlete (for coaches).

FR14. Sort workouts: The user should be able to sort workouts (e.g., by date or name).

Complete (YES); Correct (YES); Consistent (YES); Unambiguous (YES); Verifiable (YES)

FR15. View athletes/roster: The coach should be able to view a list of all of their athletes.

Complete (NO); Correct (YES); Consistent (YES); Unambiguous (YES); Verifiable (YES)
a) Not complete: the requirement does not specify what information is displayed in the roster (athlete name, email, workouts…). It also doesn’t say if there are filtering or searching options

Solution: The coach can view a list of all assigned athletes including *name and email*, with options of *searching by name*.

FR16. Add athlete to roster: The coach should be able to add another athlete by sending a request to that user.

Complete (NO); Correct (YES); Consistent (YES); Unambiguous (YES); Verifiable (NO)
a) Not complete: The requirement does not specify how the request is sent (email, inside the app notifications) or if the athlete can see details about the coach before accepting.

b) Not verifiable: since it is not specified how it is done, it is hard to develop a proper test for it.

Solution: the coach can send a request to an athlete, which is handled inside the app, and athletes must accept/decline (FR18) before appearing in the coach's roster.

FR17. Remove athlete from roster: The coach should be able to remove athletes.

Complete (YES); Correct (YES); Consistent (YES); Unambiguous (NO); Verifiable (YES)
a) Ambiguous: "Remove" can be considered vague. In the description it isn't specified if the removal is from its own roster, someone else's or even from the whole system.
Solution: The coach can remove an athlete from their roster
FR18. Accept/decline coach's offer: The athlete should be able to accept/decline a coach's offer.
Complete (YES); Correct (YES); Consistent (YES); Unambiguous (YES); Verifiable (YES)

FR19. Delete athletes: The coach shall be able to delete the athletes from the whole system.

Complete (YES); Correct (NO); Consistent (NO); Unambiguous (NO); Verifiable (YES)
a) Not correct: Coaches shouldn't be able to delete entire user accounts, only the user or an admin should have that control.
b) Not consistent: FR17 only mentions removing an athlete from the roster, not deleting them permanently.
c) Ambiguous: "Delete" is quite vague, does it erase all their data and workouts? What happens to workouts they shared publicly?

Solution: Only admins delete athletes' accounts, removing all their data and workouts from the system.

FR20. Set coach: The athlete should be able to specify a coach.

Complete (NO); Correct (YES); Consistent (YES); Unambiguous (NO); Verifiable (YES)
a) Not complete: the requirement doesn't specify how the athlete selects a coach (from a list, by searching usernames…). It also does not clarify whether a coach needs to accept or if they are automatically assigned.
b) Ambiguous: "specify a coach" can be vague. One person could understand that an athlete can only specify one coach, but other could understand that they can have multiple, and it isn't clear how it relates to the previous requirements.
Solution: an athlete can *search by name* and select a coach from a list. The coach *receives a request* and they can accept or decline. Athletes can have *multiple* active coaches at a time.

FR21. Remove coach: The athlete should be able to remove a user as their coach.

Complete (YES); Correct (YES); Consistent (YES); Unambiguous (YES); Verifiable (YES)

FR22. Register new user. The user should be able to register a new account with personal information, a username, email address, and password.

Complete (NO); Correct (YES); Consistent (YES); Unambiguous (YES); Verifiable (YES)
a) Not complete: the requirement doesn't specify the type of user that can be registered nor the password complexity rules, just like in FR1

Solution: A user should be able to register a new account (either athlete or coach) with personal information, a username, email address and password with the same restrictions as in FR1.

FR23. Coach permission: The athlete should be able to specify a coach without coach permission.
Complete (NO); Correct (NO); Consistent (NO); Unambiguous (YES); Verifiable (YES)
a) Not complete: Just like FR20, it doesn't specify how the athlete selects a coach.
b) Not correct: Allowing an athlete to assign a coach without the coach's consent is problematic.
c) Not consistent: Directly contradicts FR16 (coaches send athlete requests) and FR20 (setting a coach with approval).
Solution: remove this requirement, as the correct meaning has already been defined in FR16 and FR20.

Task 2 Non-functional requirements Specification & Measurement (10 points)

The table above does not include non-functional requirements of the SecFit system.

- **With your own knowledge, identify 5 non-functional requirements from ISO25010 categories: that you think are relevant to SecFit. Two of them have to belong to Usability and Performance categories.**
- **Provide measurements for the following software quality attributes based on the given requirements. Use the table below to specify the requirements. See the example:**

Measures for Quality in Use

Example: measure to evaluate Efficiency

Unique Identifier	NFR001
Name	User Productivity When Performing the Function
Description	Percentage of time the user is effectively performing the function
Measurement Function	Prod = Tp / Tt Prod = Productivity Tp = Productive Time (Task Time - time spent searching in help, time dealing with errors) Tt = Total Task Time
Minimum limit	90 < Prod < 100%: excellent productivity 75 < Prod < 89%: average productivity Prod < 74%: low productivity

*To identify non-functional requirements, support can be drawn from the Quality Models defined in the International Standard ISO/IEC 25.010 (Quality in Use Model and Product Quality Model).

** To identify measures for assessing non-functional requirements, support can be drawn from the International Standard ISO/IEC 25.022 (Measurement of Quality in Use) and the International Standard ISO/IEC 25.023 (Measurement of System and Software Product Quality).

Requirement ID	
Name	
Description	
Measurement Function	
Acceptance Criteria	

Assessment criteria:

1. ***Quality of requirements***

Requirement ID	NFR1
Name	Data encryption (under Security Factor)
Description	The encryption of user data (like email, password, uploaded documents, and images and videos of workout) to ensure confidentiality
Measurement Function	SecFit should adhere to AES (Advanced Encryption Standard).
Acceptance Criteria	Minimum: AES-128 Recommeded: AES-265 with GCM Mode

Requirement ID	NFR2
Name	Availability of SecFit (under Reliability Factor)
Description	SecFit should have an uptime (amount of time a system is up and running) of atleast 99.99%.
Measurement Function	A = Total time SecFit is available during the year T = Total duration of a year U(uptime) = A/T
Acceptance Criteria	U>= 99.99%

Requirement ID	NFR3
Name	Accessibility of SecFit (under Usability)
Description	SecFit should be accessible to people with disabilities.
Measurement Function	SecFit should confirm to WCAG (Web Content Accessibility Guidelines)
Acceptance Criteria	SecFit should atleast meet WCAG 2.1 Level AA.

Requirement ID	NFR4
Name	Upload time of documents on SecFit (under Performance Efficiency)
Description	The time it takes for the users to upload PDF documents
Measurement Function	T1 = Time taken by 1 user to upload 1MB files (PDFs of 1-5 pages) T2 = Time taken by 1 user to upload 1-10 MB files (PDFs of 10-50 pages) T3 = Time taken by 1 user to upload 10-100 MB files (PDFs of 50+ pages) T4 = Time taken by 1 user to upload 100MB – 1GB+ files
Acceptance Criteria	T1< 2s T2>= 2s and T2<= 5s T3>= 5s and T3<= 15s T4>=15s and T4<=60s

Requirement ID	NFR5
Name	Learnability of SecFit (under Usability)
Description	Amount of time a new user spends in learning the functionality of the system
Measurement Function	T1 = Time taken by users of age group 5-12 years T2 = Time taken by users of age group 12-50 years T3 = Time taken by users above 50 years of age
Acceptance Criteria	T1>= 5 mins and T1<= 15 mins T2>= 5 mins and T2<= 10 mins T3>= 5 mins and T3<= 40 mins

Task 3: Sustainability requirements (10 points)

- **What does it mean Web Sustainability according to (https://w3c.github.io/sustyweb/)?**
- **Sustainability Requirements are properties or constraints of a software system so that they can be operated with a positive impact on the environment, society, and economic aspects. Based on the Web Sustainability for Guideline User Experience Design, identify and specify five relevant requirements that contribute to web sustainability.**
- **How do these sustainability requirements impact other non-functional requirements?**

Assessment criteria:

1. ***Quality of the five identified requirements***
2. ***Relevance of the identified requirement to sustainability***

Task 3: Solution

Web sustainability is an approach is designing digital products and services that **puts people and the planet first**. It aims for a **clean** (hosted using renewables), **efficient** (using the fewest resources possible), **open** (accessible and user-controlled data), **honest** (avoiding misleading or exploiting visitors), **regenerative** (support people and the planet), and **resilient** (function under any circumstances) service or product.
Internet, with its large range of uses, such as websites and cryptocurrencies, consumes large amounts of electricity in datacenters, telecom networks and end user devices. This usage is so high that if Internet was a country, it would be the 4th largest polluter in the world. Thus, we need to embrace sustainability and create a web that is good for both people and the planet.

The five requirements that are relevant to the application SecFit are:

Requirement 2.15: Taking a more sustainable approach to image assets.

Implementation: Optimizing Images, Lazy Loading and letting the visitor select the display size and giving them the option to deactivate images can make for a much more sustainable product. This is also related to Requirement 2.16: Taking a more sustainable approach to media assets and Requirement 4.3: Compress your Files

Impact on NFR: This helps in improving the Performance of the application, by speeding up HTTP requests and data transfer. By sending lower resolution for less capable devices, the data savings will translate to a performance boost.

Requirement 2.2: Assess and Research Visitor Needs

Implementation: Through data analysis, we can determine to an extent, the kind of workout a user is interested in (Example: strength training, building endurance) and can use it to give relevant recommendations while searching for a coach or browsing through the public workouts. This will save their time, save resources and is also in accordance with
Requirement 2.9: Respect the Visitor's Attention

Impact on NFR: This improves Usability by ensuring that the application aligns with user expectations and preferences. The personalized experience leads to efficient navigation and reduced cognitive load. Tailored suggestions also help users with cognitive disabilities by simplifying decision making and reducing unnecessary interactions. All this contributes to enhancing its usability.

Requirement 4.2: Optimize Browser Caching

Implementation: An athlete might repeatedly want to view the training plan uploaded by her coach and the coach might need to view her athlete's workout video more than once in order to correctly assess the posture/suggest improvements. Optimizing browser caching helps and is hand-in-hand with Requirement 3.24: Run Fewer, Simpler Queries as Possible

Impact on NFR: This not only helps in improving performance efficiency but also enhancing scalability. It reduces server load by reducing frequent requests, allowing the system to handle more users without degrading the performance.

Requirement 5.20: Promote Responsible Data Practices

Implementation: Since there are options such as keeping ones media files private / accessible to coach / available to the public, there should be strict guidelines to demarcate between these categories to ensure data privacy thereby, preventing misuse of data. This also means taking Requirement 5.21: Implement Appropriate Data Management Procedures into consideration and ensuring that expired data of old customers is deleted.

Impact on NFR: Ensuring that files and personal data is protected from unauthorized access enhances security of the app. Also, implementing structured data management procedures makes it easier to update, manage, and clean up old or unnecessary data.

Requirement 2.20: Provide Accessible, Usable, Minimal Web Forms

Implementation: Since, this is a site that requires users to create a profile and input/upload personal information/files, it is best to ask only the required information. This minimalism is efficient and quicker. It can also be implemented vice versa by displaying only necessary information on the screen for the user, which can lead to Requirement 2.6: Create a Lightweight Experience by Default.

Impact on NFR: The non-functional requirement satisfied through this implementation is usability (simplifies user interactions by reducing unnecessary form fields, making it easier and quicker for users to complete actions), accessibility and performance efficiency.

Feedback Report

Task 1:

To approach this, first we read the feedback comments to understand where to put our focus. We read again our solution and paid special attention to the requirements where the feedback affected and decided which needed to be modified. The requirements most affected were:

- FR5: the change of the solution into a table for more completion and readability.
- FR13: the definition of the lists is an important detail in the requirement for it to be complete, in the first delivery we overlooked it.
- FR22: it is important to be consistent in the solutions given. As we corrected FR1, the same corrections should be applied in FR22.

Other requirements affected were also corrected, but the corrections weren't that relevant or even they were already correct in the first place:

- FR1: the password strength is already mentioned in our solution. It was specified as weak and without restrictions as an example of where the desired information should be.
- FR3: the size of the files is important for verifiability, which was specified in other requirements but was overlooked in this one, and now added in order to be correct.
- FR16: the solution given corrected the problems of the original requirement, but we didn't specify that it was not verifiable due to the other problems we found. Now corrected.
- FR23: this was already correct.

Tasks 2 and 3

These tasks weren't given any negative feedback or suggestions for change, so we re-read them just in case we found anything to change. In the end, these two tasks both turned out very nicely from the beginning so no changes were necessary.