Evaluation of the final solution

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Definition/details about criteria** | **Strategy for evaluation** |
| Completeness | How well does it meet the requirements specified in the SRS?  Does the solution do everything that it is expected to do?  Are all the solution’s features well thought out and developed? Are they the most efficient and effective as they could be? | 1. Get a current user or a test participant and ask them for suggestions to improve on the current solution 2. If they suggest that the solution is missing key features, determine the importance of the suggested features, the difficulty of implementing it and decide whether a software update is required |
| Readability and clarity | How legible (clear) is the solution’s layout and design?  Are all elements and text easy to read? | 1. Get a current user or a test participant and ask them about the current solution, specifically about whether they can see all the elements in the solution easily 2. If they are unable to see elements of the solution, determine whether changing the element for readability/clarity (by increasing the size/changing the appearance of the element) would adversely affect the solution’s attractiveness and decide whether a software update is required |
| Attractiveness | How visually appealing is the solution’s design? | 1. Get a current user or a test participant and ask them about the current solution, specifically about the appearance of the solution 2. If they are put off by the solution’s appearance, determine if it is a popular opinion or just an opinion they personally have. This can be done by asking other users or test participants. 3. If it seems like a popular opinion, ask for suggestions on how to improve the solution’s appearance and decide whether a software update is required |
| Accuracy | Is the information produced always correct?  Are all user inputs validated and/or manipulated appropriately? | 1. Get a current user or a test participant and ask them about the current solution, specifically about the timer feature 2. If they decide that the timer isn’t accurate/doesn’t work or if they find that invalid inputs can be entered, decide whether their claims are true by manually recording the time or by inputting the invalid inputs. 3. If their claims are true, decide whether a software update is required. |
| Accessibility | Are the functions of the user interface easy to find and use? Does the solution cater to those with special needs (disabilities, impairments etc.)? | 1. Get a new user or a test participant and get them to try out the software solution for the first time. If new users/test participants cannot be found, ask current or previous participants about their experience with the software solution. 2. If the users or test participants have found difficulty in using or finding the solution’s functions, decide whether a software update is required. |
| Timeliness | Is the information produced (the statistics/reports/graphs) relevant and useful for the user’s needs? | 1. Get a current user or a test participant and ask them about the current solution, specifically about the report feature 2. If the user or test participant suggests to add more features to the report page, determine whether the it would be useful for other users and the difficulty of implementing that feature 3. If it would benefit other users and is able to be implemented, decide whether a software update is required |
| Communication of message | Do users know what the software solution is trying to achieve?  How clearly is the solution presenting the produced information? | 1. Get a new user or a test participant and get them to try out the software solution for the first time. If new users/test participants cannot be found, ask current or previous participants about their experience with the software solution. 2. Ask the user or test participant about what they think the solution is trying to achieve. 3. If they guess incorrectly, ask how the solution’s message could be communicated more clearly and decide if a software update is required |
| Relevance | Does the solution contain all the necessary details or elements to meet the user’s needs? | 1. Get a current user or a test participant and ask them about the performance of software solution 2. If they suggest that the solution is missing key details or elements to meet the user’s needs, determine the importance of the missing details or elements, the difficulty of implementing it and decide whether a software update is required |
| Ease of use | Is the user interface and the functions of the solution intuitive (in a logical order, makes sense), easy to learn and easy to use? | 1. Get a new user or a test participant and get them to try out the software solution for the first time. If new users/test participants cannot be found, ask current or previous participants about their experience with the software solution. 2. If the users or test participants have found that the solution’s functions are not intuitive, or if they have had difficulty in learning or using the solution’s functions, decide whether a software update is required. |
| Intuitiveness | How easy is it for users to understand and use the functions of the software?  Does the solution follow standard conventions? | 1. Get a new user or a test participant and get them to try out the software solution for the first time. If new users/test participants cannot be found, ask current or previous participants about their experience with the software solution. 2. If the users or test participants have had difficulty in understanding or using the solution’s functions, decide whether a software update is required. |
| Consistency | Are the necessary elements of the user interface (such as the navigation bar) in the same positions and have the same functions across the product? | 1. Test all elements that are seen multiple times throughout the solution (such as the navigation bar) 2. If the element performs different functions across the solution, decide whether a software update is required |
| Robustness and tolerance | How well does the solution respond to poor usage? | 1. Test the solution by putting in random and invalid inputs, going to previous pages, resizing the solution – just using the solution badly. 2. After using the solution poorly, observe how the solution responds 3. If the solution responds poorly (by accepting poor inputs, crashing, displaying elements incorrectly) decide whether a software update is required to ensure that the solution responds better |
| Affordance | How forgiving is the solution? (Are users warned of dangerous/drastic decisions, are users able to undo actions etc) | 1. Perform a dangerous/drastic action such as deleting a subject/task or deleting an account. 2. During and after performing a dangerous/drastic decision observe how the solution responds 3. If the solution is not forgiving, determine whether a software update is required |
| Maintainability | Is the solution easy to maintain? | 1. When fixing or modifying features or functions of the solution note the difficulty of modifying the code 2. If the code written is hard to understand, disallowing for easy maintenance or modification, determine whether a software update is required to improve the clarity of the code |
| Scalability/flexibility | Can the solution allow for more functions or features without adversely affecting the layout and functioning of the solution? | 1. When implementing new features note the difficulty of implementing them 2. If the code written is hard to understand or if the layout of the solution disallows for easy implementation of new features, determine whether a software update is required to improve the clarity of the code or to modify the layout of the solution |
| Portability | Can the solution be moved to other operating environments without much effort? | 1. Run the software solution on different browsers and operating systems 2. Observe how the solution performs in the different environments 3. If the solution doesn’t perform well in a certain environment, determine the likelihood of a user using that environment and decide whether a software update is required to cater to those users who do use that environment |
| Responsiveness | Is the user interface fast?  Does it give feedback to the users? | 1. Get a current user or a test participant and ask them about the current solution, specifically about the responsiveness and the feedback the solution gives to the user when they perform a task 2. If the user or test participant decides that the solution takes too long to perform or doesn’t give enough or the appropriate feedback, decide whether a software update is required |
| Speed of processing | Is data able to be entered quickly?  Is the output generated quickly?  Can the user quickly switch between the features of the solution? | 1. Get a current user or a test participant and ask them about the current solution, specifically about the efficiency of the solution 2. If the user or test participant decides that it takes too long to perform a task (entering or processing data takes too long or switching between the features of the solution to perform a task takes too long), decide whether a software update is required |
| Suitability | Is the solution appropriate for its operating environment? | 1. Get a current user or a test participant and ask them about the current solution, specifically about the performance of the solution in their preferred operating environment 2. If the user or test participant decides that the solution doesn’t perform well under their preferred operating environment compared to other operating environments, decide whether a software update is required to move the solution to a more appropriate operating environment or to ensure that it’s compatible with the user’s operating environment |

### Functional requirements

* User register – REQ1

This requirement is met if the user can register with a new account for the software solution. The user register page can be accessed from the user login page through a small link and from there new users are able to register and create a new account for the software solution. The solution is efficient as it only requires two data inputs from the user – a username and a password. Any other details required from the user such as a display name, subjects/tasks, and timer preferences can be entered later, from the solution’s settings. Both inputs are validated to ensure that the length doesn’t exceed 15 characters, that the username is unique and that the password is greater than 5 characters.

* User login – REQ2

This requirement is met if the user can login with their account details to access the software solution. If the user is not already logged in, the login page will be displayed to the user. If they are already logged in, they are redirected to the main page of the solution making the solution efficient as it doesn’t require users to log in unnecessarily. When logging in, users are required to input in their username and password. Depending on what they input, several different error messages can appear – one for missing data inputs, incorrect usernames and one for incorrect username or password inputs. The solution is effective because if users enter in the correct user details, they are allowed access to the solution.

* User logout – REQ3

This requirement is met if the user can log out from the solution and are not allowed access back in without logging in again. From any page of the solution (except for the login/register pages where the user is already logged out), the user can access the log out link from the navigation bar, making the feature easily accessible and consistent which makes the solution efficient. The solution is effective because after the user clicks on the log out link, they are redirected out back to the solution’s login page and are not allowed access again without logging back in.

* Functional timer – REQ4

This requirement is met if the solution has an adjustable, live and accurate timer implemented to allow users to time the length of time spent on a subject/task or to see how long they have until a break. This timer is implemented in the solution’s timer/work page which can be accessed by clicking on a subject/task on the main page. This timer is fully adjustable. Users can choose the timer type, either a count up or countdown timer. If they choose to use the countdown timer, relevant settings such as the timer length, timer sounds and notifications are made available to the user. If they choose to use the count up timer, these settings for the countdown timer are hidden. By only showing the relevant timer settings, this improves the solution’s accuracy (invalid inputs are not allowed), ease of use and relevance, improving the solution’s efficiency. The timer can be adjusted both in the settings and on the timer page itself, also improving the solution’s ease of use. The solution is effective because the timer is adjustable, interactive (it’s able to be started, paused and reset) and is accurate, even if the user uses it poorly (by constantly starting and stopping the timer).

* Timer and break notifications – REQ5

This requirement is met if the solution features sounds that play and/or notifications that show when the countdown timer runs out. In the solution, the timer sounds and notifications can be turned on/off. If the timer sounds are turned on, when the timer runs out a looping sound will play to indicate that the timer has run out and will continue playing until the user clicks on any button on the solution or if they leave the timer page. If the timer notifications are turned on, a push notification will display for a few seconds with a short message to indicate that the timer has run out before disappearing. This feature is efficient and effective as it does what it is expected to do and requires minimal input from the user to stop the sounds or notification.

* Adding subjects/tasks – REQ6

This requirement is met if users can input their own unique set of subjects/tasks. On the main page this can be done through the text input box. Each manual input is validated to ensure that it doesn’t have the same name as another subject/task the user currently has and that it is not too long (greater than 25 characters). If the user has entered any invalid inputs, the solution gives a message to the user, improving the solution’s responsiveness and efficiency. This feature is effective as it does what it is expected to do, allowing users to have a unique set of subjects/tasks that they can work on.

* Deleting subjects/tasks – REQ7

This requirement is met if users can delete subjects/tasks. This can be done by clicking on the red cross – the delete button, displayed next to the subject/task on the main page or in the settings (account) page. When users click on the delete button, a prompt is displayed to the users to confirm their actions. Users can either cancel or confirm their actions. This feature is efficient as the confirmation prompt promotes the solution’s affordance and it makes the changes immediately, making the solution responsive. The feature is effective as confirming the action deletes the subject/task and all of its relevant information.

* Recording length of time – REQ8

This requirement is met if users can record the time they have spent on a subject/task. This feature is implemented in the solution through the implemented live timer that records the time spent on the subject/task (REQ4) and through the manual time input located on the timer done page. The value recorded by the timer is shown to the user on the timer done page and can be used by confirming the length of time through the press of a button. The manual time input is used if the time recorded is inaccurate or if the user wants to input in a different value than the one suggested by the solution. The manual time inputs are validated to ensure that it’s within a reasonable range. This promotes the solution’s relevance, accuracy and responsiveness. These features are effective because by confirming the length of time spent, either the time recorded, or the time manually inputted by the user, the user is immediately redirected back to the main page and the timer values are added to the subject/task worked on.

* Report of time spent on subjects/tasks – REQ9

This requirement is met if users can view the length of time spent on a subject/task for a day. Users can quickly view the length of time worked on a subject/task on the main page and can view a more detailed report in the view statistics page, where the total time and a graph of how time was distributed is also shown. This promotes the solution’s efficiency and effectiveness through improving its readability, accuracy, accessibility, timeliness, relevance and affordance.

* Tutorial – REQ10

This requirement is met if users can view a guide to help them use the solution effectively. The help page, which contains relevant information regarding the solution and how to use its features can be accessed from any part of the solution (expect from the login/register page) from the navigation bar. This promotes the solution’s accessibility, consistency and ease of use. This feature is effective as it contains all the information a user might need to use the solution as well as screenshots of the solution to help them better understand the presented information.

* Settings – REQ11

This requirement is met if users can change various details associated with their account such as timer preferences, account details and subject/task lists. The link to the settings page can be accessed from any part of the solution (expect from the login/register page) from the navigation bar. This promotes the solution’s accessibility, consistency and ease of use. As users are expected to change the timer preferences more frequently than their account details and subject/task lists, the timer preferences are displayed first with the links to the other two setting pages. This promotes the solution’s ease of use and efficiency. On the timer preferences page, the timer type, length, ring and notifications are adjustable. On the settings (account) page, the user’s username, password and display name are able to be changed. For ease of use, the user’s username and display name are autocompleted in the form. For security, the password is not shown and in order for users to change their password, they must enter in their current password and then the password they want to change it to twice. If the form is filled out incorrectly, the user is given a message, promoting the solution’s responsiveness. On this page, users can also delete their account. Since this is a risky action, a confirmation prompt is shown and users must enter in their current password to confirm their actions, promoting the solution’s affordance and security. On the settings (subject/task list), users can change the name of a subject/task and can also delete it. When changing the name of a subject/task, the inputs are validated to ensure that the new name is unique and isn’t too long, promoting the solution’s accuracy. When deleting a subject/task a confirmation prompt is shown, promoting the solution’s affordance. These features are effective as users are able to make changes to their details which immediately take place.

### Non-functional requirements

* Reliability and responsiveness – NREQ1

This requirement is met if the solution meets all of the user’s expectations and processes/performs all of the requirements quickly. From the usability testing where the majority of users agreed that the solution did as it was expected and where it was observed that the functions of the solution were performed quickly, it can be concluded that the solution meets this requirement. The software solution was shown not to crash even when used poorly and gave appropriate feedback to the users when they made changes in the settings or when they entered an incorrect or invalid input.

* Safety – NREQ2

This requirement is met if there is no possible loss, damage or harm that could come from the usage of the solution. From the usability tests and through the way the software solution was developed, the software solution is incapable of causing harm to the user. It does not affect data or other software stored outside of its servers and does not damage the device that it is run on or any of its internal components.

* Security – NREQ3

This requirement is met if unauthorised users are unable to access other user’s accounts and view/edit their information. The solution uses usernames and passwords for user identification. If the user has inputted the incorrect username and/or password, they are not allowed access to the solution. Risky actions such as changing the password or deleting an account requires the user’s input of the current password the user in order to make changes.

* Legibility – NREQ4

This requirement is met if all elements of the solution are able to be seen and read clearly. Throughout the design and the development process of the solution, the contrast of colours, the sizes of elements and text fonts were kept in mind. The results of the usability testing show that the majority of users were able to see all elements the solution presented. The text used in the solution was kept concise to not overwhelm the user interface and to cater to users who were vision impaired and for those who have trouble reading.

* Intuitiveness and consistency – NREQ5

This requirement is met if all features and elements used in the solution are self-explanatory, understood, are kept consistent and adhere to all standard conventions. Through the thorough testings and the usability testing, this requirement was shown to be met as the features of the solution were observed to be consistent across the solution and the users were observed to not have difficulty in using and understanding the features of the solution.

* Accessibility and usability – NREQ6

This requirement is met if all features of the solution are as accessible and easy to use for users as possible. This is shown to be met through the efficiency of the solution as minimal inputs are required to perform actions. For example, when entering data into input boxes, the space bar can be pressed to submit the form instead of pressing confirmation buttons. Any buttons and interactive elements that are used in the solution are made with a relatively large clickable area, so users can interact with elements easily. To assist with usability, a help page has been provided for users who are unsure about how to use the solution.

* Attractiveness – NREQ7

This requirement is met if the solution has a pleasant appearance. The solution has a simple colour scheme consisting of a few colours and utilises some soft rounded shapes to give off a pleasant and calming appearance.

* Affordance – NREQ8

This requirement is met if the solution is forgiving. This requirement is shown to be met because all risky decisions the user could perform while using the solution such as deleting subject/tasks and accounts must be confirmed twice by the user. The first confirmation comes from the action of clicking on the delete button or link. For the deletion of subjects/tasks the second confirmation is given when the user clicks on the confirm option and for the deletion of the user’s account, the second confirmation is given when the user enters in their correct current password. If users decide to cancel their action, the solution will not do anything.

* Tolerance and robustness – NREQ9

This requirement is met if the solution does not crash when used poorly. During the formal testing where the solution was used badly, the solution was shown not to crash or perform poorly.

* Maintainability and portability – NREQ10

This requirement is met if the solution’s code is written in a way that allows for easy maintenance, further development and implementation of new features and if the solution is compatible with various operating systems and web browsers. Throughout the development of the solution, the maintainability of the code was kept in consideration so versioning, commenting, indention and other standard code practices were used. This allows for maintenance and implementation of the solution’s features. The compatibility of the solution was also kept in mind throughout the development of the solution so CSS reset code was used to keep the appearance of the solution relatively similar across web browsers and the solution was tested on different operating environments.