TIME FOR A BREAK

Group 22

IT314 Software Engineering
Winter 2020-21

Under the able guidance of **Prof. Saurabh Tiwari**

Table of Contents

Project Description

Time for a break is as simple as a project can be. The project's main requirement is to remind users of taking breaks between long hours of screen time. A typical quartz alarm clock can fulfill the same condition. Then what is the need for this project? For instance, in today's world, nobody likes anything monotonous or inflexible. In other words, one wants something in the form of a liquid and not as solid because it can take the shape that users request. The same is the case with our application. Time for a break is made to fulfill the requirement of a comfortable, customizable and innovative solution for a simple reminder of taking a break that helps users stay healthy and motivated during the long screen times.

Time for a Break is a vital application for those with high screen time along with a desk job. Research suggests that if one person regularly remains seated in a single position staring at a computer screen, it will severely negatively impact their body, especially the eyes and the spinal cord. Taking regular breaks and doing some remedial fitness exercises during break time can help them relax and remain in the right touch with their bodies. To help such customers break the monotony and remind them to take a break at regular intervals, Time for a Break is the go-to app. As a native application, it is ideal for such a purpose.

We hope to achieve some objectives and goals with a successful deployment by developing this application. This user-friendly app provides all the necessary functionalities for those planning to follow a healthy regime comprehensively. We hope to provide prospective customers with something that fulfills all their requirements with this product. These include the app runs in the background, custom audio notifications, schedule, cancel breaks, exercise resources and tips, session reports (include details like the number of breaks taken, breaks skipped, and so on). Since our product ticks off all the necessary boxes, we believe our product will stay true to the users' expectations and help them achieve their goals.

Stakeholders and Users

Stakeholders:

- Desktop/ Laptop users
    - Useful for particularly those with high screen time
    - They are the end-users of the product
    - People with a desk job and a sedentary lifestyle provide a huge market for this product
- Developers
    - The team of students working on this project
    - Use the requirements and feedback obtained from the end-users to keep improving the software with future updates
    - Follow Software Engineering practices like test cases preparation, use case models, etc. for creating the final product
- Mentors
    - Monitoring the progress of the work done and providing their valuable feedback.

Users: Desktop / Laptop/ System users (especially those with a desk job/ high screen time)

Functional Requirements

The following functional requirements were distilled from our requirements elicitation steps. They cover all the major features and requirements required by a user. They are as follows:-

1. Change Break Duration and Frequency

If the user feels that the break duration is more than he/she requires then he can change it. Similarly, if the user feels that the break duration is less he can change it in a way that break duration is more than it was before. This feature is applicable for both mini-breaks and long breaks. Users can also change the frequency of the breaks, i.e, after how long a break is scheduled.

2. Cancel
This feature enables the user to cancel the break. This feature does not affect the minibreak and long break scheduled in the future.

3. Strict Mode
Strict mode prevents the user from skipping either mini breaks or long breaks and is designed to help the user follow the schedule strictly.

4. Postpone
The user can postpone the break if he/ she wishes to continue with the work he is doing. However, there is a limit to the number of postponed breaks.

5. Audio Preferences
There are several choices available like the user can change the Audio tone from his/ her local storage. Also, the user may choose not to have audio playback. Users can add as many audio files as they like, the application will play one of them randomly. By default, one default audio file has been added.

6. Notifications Preferences
The user will be able to change the notifications setting like don't show the minibreak notification beforehand. Similarly for a long break.

7. Report Generation
For each previous session, a report is generated. This feature enables the user to analyze the number of times he has taken minibreaks and long breaks, postponed it, and canceled it. The report will specifically show the number of long and short breaks attended and the number of short and long breaks skipped.

8. Quit the session
This feature will enable the user to stop the sessions in case they do not wish to take breaks (in a meeting, watching a movie, etc.). Quitting the session resets the timer for the start of the next session, and generated a report for the session that just ended.

9. Ideas
This feature will enable the user to add some ideas that will be shown when any break notification window pops up.

Non-Functional Requirements

The following non-functional requirements must be met in order to run the application smoothly:-

- Target Operating System: Windows 10
- The app should be able to work without any active internet connection i.e it will be a native desktop application.
- The system state before and after the break should be the same i.e. no data/state loss should occur due to the application. The app should not clash or interfere with other running apps.
- The application should continue to run in the background whenever the system is active.
- Storage Requirements: Local storage will be used by the application for its functionalities.
- Performance Constraints: Acceptable response time against some specific sort of user activity.
- The application's clock should be properly synchronized with the system's clock, i.e, there should not be any delay in notification.
- If the user has selected the strict mode then he/ she has to take the break.
- For notifications to appear, the user should enable notifications in Windows (operating system settings).
- The audio file uploaded should be of .mp3 and .wav format only.

Requirements Elicitation

Before embarking on our product, we studied the existing systems in place for such an application. We read and talked about how people use manual methods to take periodic breaks from their busy work schedules. We found that, despite the presence of some reminder apps such as ours, their use was not widespread enough to account for a huge share of users with high screen time.

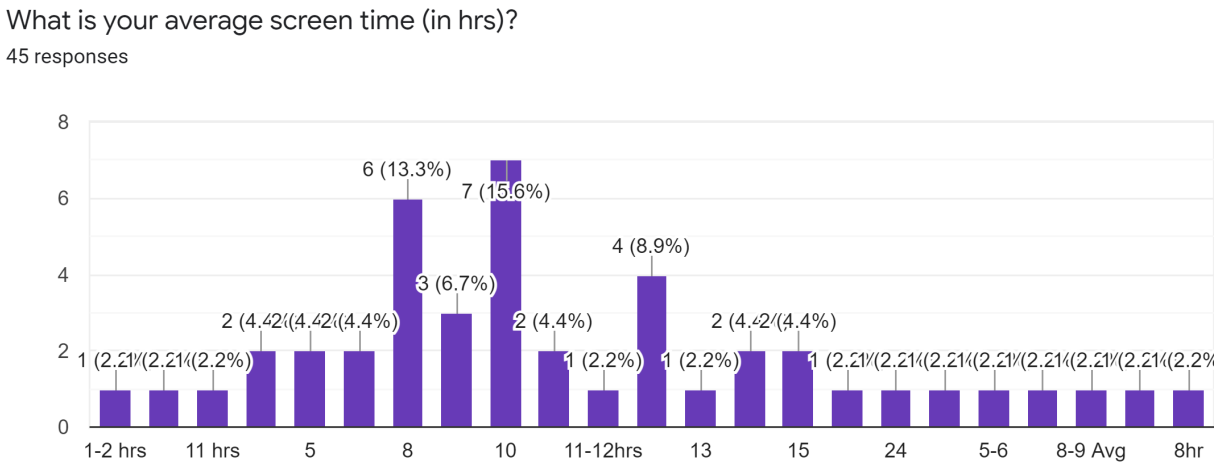
Till now, such applications have not found popular traction because it is difficult to find a single application that fulfills all the needs of a user. While the majority of the current applications offer basic functionality, users are divided over secondary functionalities like notifications, statistics, and so on. Also, platform cross-functionality was also an issue with several apps. With our product, we aim to resolve some of these issues and

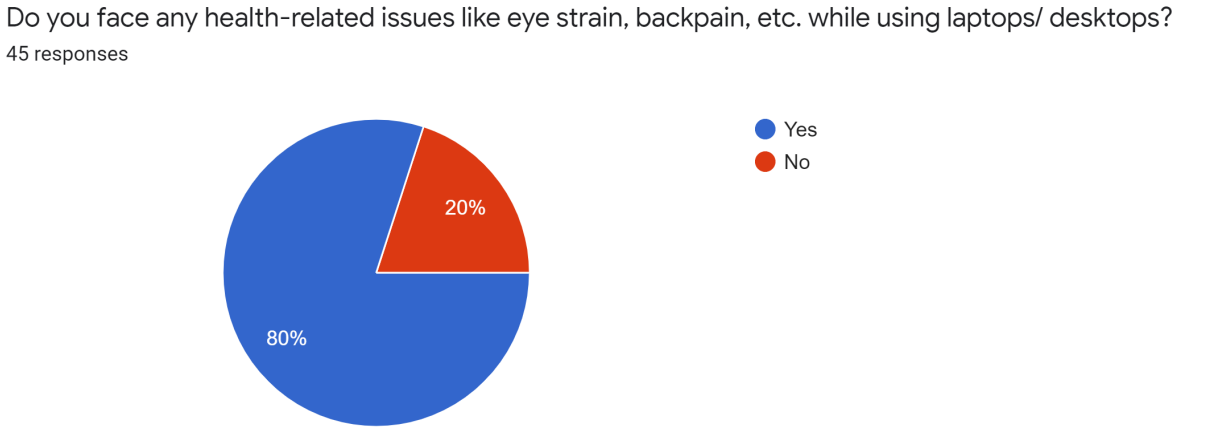
make sure that more and more users who need this can use such an application in a user-friendly manner.

For developing our product, various techniques of requirement gathering were performed. In order to reach out to the maximum number of respondents, we created a variety of platforms and employed different methods to get their desired requirements.

Google Forms: Two Google Forms were created. One of them was for surveying the requirements and desired features. Questions like what features you would expect, what is most important, and so on, were asked. The other Google Form was created for collecting feedback from users of the product.

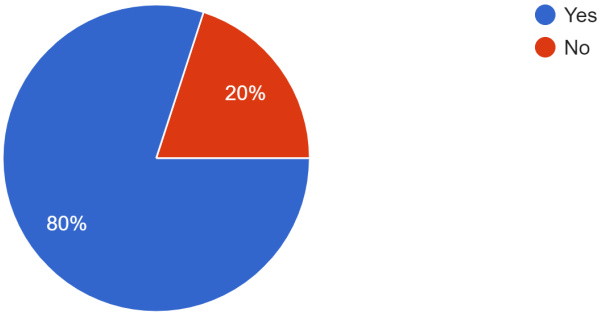
Survey Form responses:





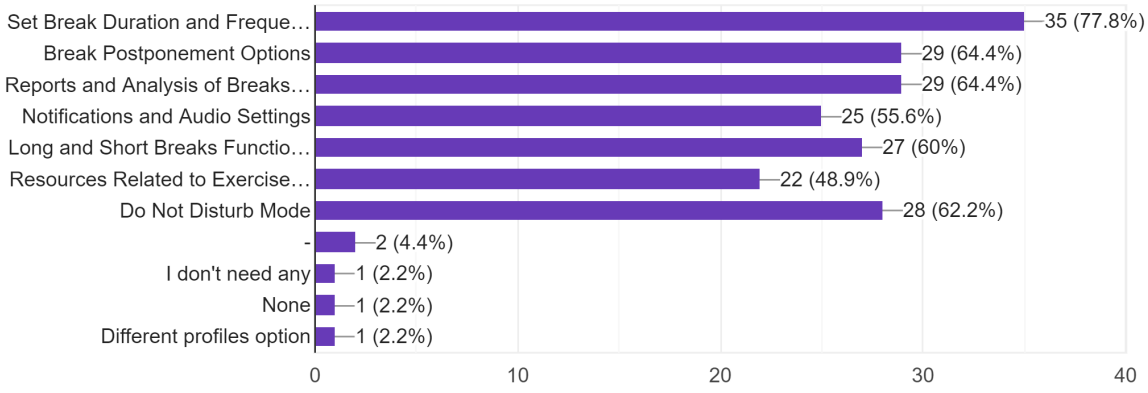
Do you feel the need of an app that reminds you periodically to take a break?

45 responses



Which of the following features would you like such an application to have?

45 responses



Interviews: We interviewed some students/ professionals who had high screen times and took their views on such reminder apps and how necessary they feel to take breaks at short intervals. We discuss with them their working habits and sitting posture. Each interview had a fixed set of questions and some were added upon discussion. From here, we were able to get different perspectives of students and professionals so that we could make our product more accommodating.

Informal Discussions: Informal discussions among ourselves (developers), batchmates, and friends were common in the initial stage when we were planning the product. Such discussions were fruitful because we were able to freely discuss the issues with prospective users, and improve upon our own ideas. People were able to share their opinions, and we also got to know if they used any such existing product.

General Reading: General reading from online websites and articles helped us understand the current landscape of viability of such a product. We learned about the advancements in this domain and what the current apps provide as features.

User Interfaces

- Users are not segregated into different categories i.e there will be only one user per system. Hence, there are no separate interfaces.
- The single interface of our application works offline and has to be installed on the local machine. Once installed, it works as a desktop application.

Software Model Process

After brainstorming different options for the most efficient way to create our model, and understanding the basic nature of our product, we decided that the **Waterfall model** would be the most suitable software model process for our product.

The waterfall model is appropriate here because the problem statement and requirements are fairly clear. Once the functional requirements and features are decided upon, there is little room for any changes in that. Our product is also, in a sense, static in nature, i.e., there are no database dynamics involved, no internet connection is needed. Also, there are existing applications for this task, so with our product, we are basically trying to better the existing systems. With a simple goal and a short development life cycle, this product can deliver its goals efficiently. Keeping the described reasons in mind, we decided to proceed with the Waterfall model.

Use Case Diagram

Use Case Description

Use Case Name	Settings
Description	By going through this use case(Activity), System users can change settings in the app for their system. It consists of Notification preference and Strict Mode.
Precondition	There is no restriction(Condition) unless the user can't access the Settings Page.
PostCondition	If the user has not changed anything then the previous settings should be kept intact else the new ones should be applied and saved in the local storage.
Basic Flow (System Happiness)	The user starts the app during the time other than the break time(Trigger period specified by user) and opens the settings and then he/she may i**)** adjust app notifications i.e whether notification should appear before 5 seconds of any break or not. **ii)** set the strict mode in the app. By enabling strict mode, 'skip the break' button (through which the user can skip any particular long or short break) will not appear in the break window. **iii)** keep the default settings as it is.
Alternative Flow	Notifications are enabled by default.

Use Case Name	App/Break Trigger
Description	App would trigger at the specified period with the specified action and would remain active for some certain time period given by the user. The App can also be terminated by the user before its terminating time.
Precondition	The time at which the app should trigger, the time period range during which the app should be active and the action to be carried out; these quantities should be specified by the user and also during the specified active time, system should be on.
PostCondition	After the completion of the break event, all information and user activity(User activeness, forced termination etc) during that time should be saved in the database and should be

	added in the report being generated after the end of that session.
Basic Flow (System Happiness)	At the certain exact time(given by the user), the app triggers and implements some certain actions(Audio, notifications etc, it would fetch these data from the local database) for some certain time period(given by the user). To render the idea(quotation) or to provide the music, this page will fetch one idea and music file randomly from the existing ideas and music files from the local storage. Now in some cases, when some break is active and rendering some actions, the user can terminate it too using the 'Skip the break' button available on the break window if the strict mode is off.
Alternative Flow	=> If the user hasn't given the data for the actions to be implemented during some break time, then the app will render the default actions. => If the system turns off while the break time is on then the break should terminate abruptly.

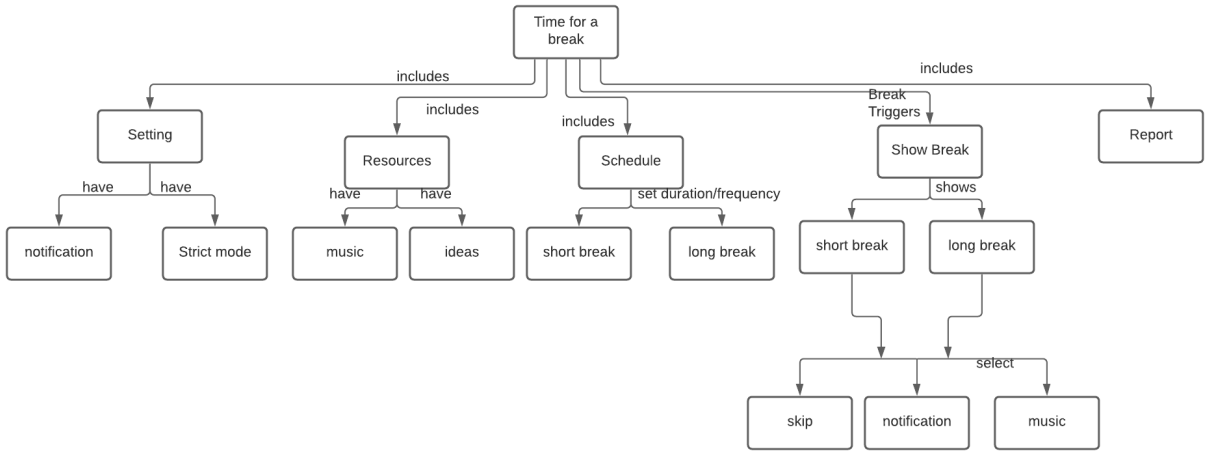
Use Case Name	**Report generation**
Description	Whenever the user goes into the report page, the page will render the report of the previous session.
Precondition	This use case includes the break trigger use case. So if no breaks have occurred in the system after the installation of the app, the report will show empty/zero values.
PostCondition	After the end of each session, all the required information should be stored in the local storage successfully so that it can be used while rendering the report page.
Basic Flow (System Happiness)	The user has ended the current running session. So then all the required information will be stored in the local storage immediately. Then at any point of time when user goes into the report page, it will show following informations about the previous session; ● Total duration of the session ● Number of short breaks skipped ● Number of short breaks attended ● Number of long breaks skipped

	• Number of long breaks attended • Percentage of total breaks attended • On the basis of the percentage(75% is the threshold), greet message will be shown(Congratulation/Try hard)
Alternative Flow	At any time(even when the current session is running), the user goes into the report page, it will show the report of the previous session.

Use Case Name	**Schedule**
Description	By going through this use case(Activity), System users can change certain quantities related with breaks i.e duration and frequency for short and long breaks.
Precondition	There is no restriction(Condition) unless the user can't access the Schedule Page.
PostCondition	If the user has not changed anything then the previous settings should be kept intact else the new ones should be applied and saved in the local storage.
Basic Flow (System Happiness)	The user starts the app during the time other than the break time(Trigger period specified by user) and opens the schedule and then he/she may i**)** change two quantities(duration and frequency) for long and short breaks. **ii)** Duration of a break stands for how long that break should last and frequency stands for after what time that break should trigger. **iii)** keep the default schedule as it is.
Alternative Flow	At the very start, the default value for all four variables(short duration, short frequency, long duration, long frequency) is set as 1min.

Use Case Name	**Resources**
Description	Resources page consists of two major parts i.e ideas and music. They both have separate routes to go and the user can handle(change or keep) them independently.
Precondition	The user should be able to go through the routes of ideas and music. In music, the user can add only .mp3 and .wav music file formats.
PostCondition	If the user has not changed anything then the previous resources should be kept intact else the new ones should be applied and saved in the local storage and also should be rendered on their respective pages in real time.
Basic Flow (System Happiness)	The user starts the app during the time other than the break time(Trigger period specified by user) and opens the resources page and then he/she may i**)** add some new ideas or remove the existing ones in the section of ideas in resources. **ii)** add some new supported music files or remove the existing ones in the section of music in resources. Also the user can play the existing music music files here. **iii)** keep the default resources as it is.
Alternative Flow	Some default 5-6 ideas and one default music file will be added and these default data can not be deleted by the user.

Concept Map



Future Scope

From the perspective of an offline version of such an application, our product is pretty well-developed and self-contained. But, there is a huge scope in further enhancing this product and offering many other functionalities. With all the basic features and functionalities already provided, we can make the product more adaptive to the ever-increasing affinity for online Cloud products.

Our application could be migrated to an online version where users can have login/ logout functionality. They can also set up profiles in case multiple users share a single account (pretty much like Netflix and Spotify). In such a version, users can import their Spotify playlists, online music videos, and much more in order to make the breaks much more interesting. With Cloud support, cross-device support can also be extended so that if a user shifts from one machine to another, the timer continues in sync. Multiple users who use the same device can have separate timers for them when they are using that particular machine.

Another aspect that can be particularly developed is the UI/ UX designing. As good and user-friendly the current application is, there is also a scope for improvement and imbibing the latest trends in application design for appealing to all the age groups. A focus on providing more customizable components like backgrounds and themes can be made for allowing users to set up the application based on their preferences.