The Potential of Time Break Reminder Program in Digital Work Environments

Lely Priska D. Tampubolon[a*] , Gabriel Alvaro[b]

[a] lely.priska@perbanas.id
[b] gabriel.alvaro08@perbanas.id
[a,b] *Faculty of Information Technology, Perbanas Institute, Jl. Perbanas, DKI Jakarta, Indonesia*

Abstract

This study examines the usefulness of break reminder software to reduce Computer Vision Syndrome (CVS), a condition characterized by eyestrain, headaches and visual disturbances caused by prolonged use of digital devices. By conducting a comprehensive review of the current literature, this study examines the potential of using structured screen breaks to reduce symptoms of Computer Vision Syndrome (CVS). The study emphasizes the importance of user compliance and the influence of software architecture on efficacy. The research resulted in a simple program that serves as a break time reminder. Further research is needed to be comprehensive and enhance the benefits of break reminder software for digital health and productivity.

Keywords: Computer Vision Syndrome, Digital Eye Strain, Digital Health, Time Break Reminder Program

1. Introduction

The advent of the digital age, driven by the COVID-19 pandemic, has contributed to an unparalleled surge in computer usage, resulting in a substantial increase in the prevalence of Computer Vision Syndrome (CVS). Computer Vision Syndrome (CVS) refers to a variety of eye-related symptoms that occur due to prolonged computer use, including dry eyes, itching, and blurred vision. CVS has a high prevalence, with variables such as prolonged computer use, lack of breaks, and certain personal habits playing a role in its development (Erdinest and Berkow, 2021), (Derbew et al., 2021). Given the prevalence of Computer Vision Syndrome (CVS), especially among people who use computers for long periods of time, it is imperative to implement effective intervention measures.

The purpose of this study is to build a program as a flexible break time reminder that is expected to reduce the symptoms of Computer Vision Syndrome (CVS). This study created a break time reminder program that integrates the Pomodoro Technique, a method that divides work into intervals with short breaks, to limit

computer usage time and improve eye health. In a previous study, the implementation of such software effectively reduced the symptoms of Computer Vision Syndrome (CVS), which is consistent with research showing that taking regular breaks and practicing limited computer use are essential for reducing CVS (Poudel and Khanal, 2020). Essentially, this research provides valuable information on how technology can be used to help improve health in a society that is increasingly dependent on digital devices.

Following this introduction, the paper is organized into chapters, each of which focuses on a different component of the investigation. This includes an extensive examination of the relevant literature, the methodology used, the findings obtained, and the conclusions drawn from the study. To summarize, this chapter provides an overview of CVS in the digital age and introduces break time reminder software as a viable method to help alleviate its symptoms. The importance of this research is emphasized by the increasing reliance on digital technology and the increasing complaints associated with computer vision syndrome (CVS). The program was built by emphasizing a method that divides work into intervals with short breaks. The user can input the time according to the time required.

2. Literature Review

This chapter provides a comprehensive examination of the literature surrounding Computer Vision Syndrome (CVS), its causes, symptoms, prevalence, and various intervention strategies, as well as the role of technological solutions such as time break reminder software in its management.

2.1 Definition and Symptoms

CVS, commonly known as digital eye strain or visual fatigue, encompasses a spectrum of eye and vision-related problems resulting from prolonged use of digital devices. Symptoms such as eye strain, headaches, blurred vision, dry eyes, and neck and shoulder pain significantly affect individuals' comfort and productivity (Erdinest and Berkow, 2021). These symptoms are further classified into categories related to visual symptoms, digital screen impact, and ocular surface issues.

2.2 Prevalence of CVS

The prevalence of CVS has been escalating, particularly in the context of the COVID-19 pandemic, which has led to increased reliance on digital devices. Studies have reported a high incidence of CVS among computer users, with prevalence rates ranging from 70% to 90% in some groups, thereby highlighting the public health significance of this syndrome (Derbew *et al.*, 2021). The shift to remote work and online

learning models has further heightened the risk of CVS among diverse populations, including students and professionals.

2.3 Risk Factors Contributing to CVS

Various factors contribute to the development of CVS. These include the duration of screen exposure, improper ergonomic setup, poor workplace lighting, and individual visual issues. Notably, a lack of regular breaks during prolonged computer use is a significant risk factor (Poudel and Khanal, 2020). Additionally, the physical setup of the workspace and the user's posture while using digital devices play crucial roles in exacerbating or mitigating CVS symptoms.

2.4 Intervention Strategies for CVS

The Pomodoro technique is a time management methodology created by Francesco Cirillo in the late 1980s. The method is breaking tasks into multiple intervals, often lasting 25 minutes, with brief pauses in between. Each discrete time period is referred to as a pomodoro, and the strategy is designed to mitigate the impact of interruptions on concentration and efficiency. The term "Pomodoro" is derived from the Italian word for tomato, as a reference to the tomato-shaped kitchen timer that Cirillo, a student, utilized. The practice is favored by persons seeking to enhance concentration and efficiency during work or study sessions (Iyengar, Vaishya and Botchu, 2023). In their study on a procrastination-reducing tool for graduate students (Almarzouki *et al.*, 2021), elucidated that this tool draws inspiration from the Pomodoro Technique and involves a concentrated study session of 25 minutes, followed by a five-minute break interval. This method enhances time management by taking into account factors such as examination dates, assignment due dates, the intricacy of the subject matter, and the amount of material to be studied. The application additionally offers suggestions for pertinent resources based on uploaded content to enhance time management, and alerts students if they begin to delay or postpone work.

In medical research, several interventions have been suggested to alleviate the symptoms of CVS. The 20-20-20 rule, which advises users to take a 20-second break to look at an object 20 feet away every 20 minutes, has been shown to be effective in reducing CVS symptoms (M. Zalat, S. Amer, G. Wassif, S. El Tarhouny, 2021). Ergonomic adjustments, such as optimizing screen position, enhancing ambient lighting, and using artificial tears, are recommended strategies to manage CVS (Anggrainy, Lubis and Ashar, 2020). Moreover, regular eye examinations and addressing any uncorrected refractive errors are essential in preventing CVS.

2.5 The Role of Technological Solutions

Given the digital revolution, the emergence of programs like time break reminder software, which push users to take frequent breaks, has shown promise in combating Computer Vision Syndrome (CVS). The purpose of the program is to promote and maintain controlled screen-time habits and prevent the potential dangers that come with excessive and uninterrupted use of digital devices (Anggrainy, Lubis and Ashar, 2020). Integrating technology-driven solutions into daily routines could play a crucial role in treating and preventing Computer Vision Syndrome (CVS) in the digital age.

2.6 Emerging Research and Future Directions

Previous research has prioritized investigating several aspects of CVS, such as its effects on diverse demographics, the efficacy of various intervention approaches, and the contribution of new technologies to its control. There is growing curiosity in understanding the long-term consequences of CVS and formulating a comprehensive approach that includes both behavioral and technological interventions. This literature review offers a comprehensive understanding of CVS, an increasingly worrying disorder in our increasingly digitized society. It highlights the importance of applying efficient techniques, which include ergonomic and technological approaches, to reduce the effects of Computer Vision Syndrome (CVS). This chapter forms the basis for the following chapters, which will examine the efficacy of application such as break time reminder software in reducing Computer Vision Syndrome (CVS).

2.7 Java Programming

Sun Microsystems (now Oracle Corporation) developed Java, a popular high-level programming language with few implementation dependencies. Classes, objects, and minimal implementation requirements make it a general-purpose programming language. It lets application developers write once, run anywhere (WORA) -- written Java code runs on all Java-supported systems without recompilation.

Most Java applications are compiled to bytecode that runs on any JVM, regardless of computer architecture. Java has similar syntax to C and C++ but less low-level features. The Java runtime offers dynamic features like reflection and runtime code change that compiled languages lack. Thus, Java is a "write once, run anywhere" language that runs on any device with a Java Virtual Machine (JVM) (Ghorbani, Garcia, & Malek, 2019).

Java is used to create Android apps, server-side apps, web apps, software tools, games, and more. Many developers favor Java due to its durability, ease of use, cross-platform capabilities, and security.

Java is versatile, from mainframes to cellphones. Java runs on embedded devices, mobile phones, enterprise servers, and supercomputers. Java underpins web development, including server-side applications, and many

networked applications. In Java 9, the Java Platform Module System (JPMS) encapsulates modules to improve Java application and JDK encapsulation, security, and maintainability. Java's architecture lets developers build and maintain modular apps quickly (Ghorbani et al., 2019).

3. Results and Discussion

This chapter describes the results of a literature review of previous research that explains the importance of computer users to rest their eyes in order to reduce the symptoms of Computer Vision Syndrome (CVS). The programming stages built using the Java language will also be discussed in this chapter. By conducting functionality testing conducted by 20 users through different devices, the test results can be seen in Table 1. User selection for testing was done through a purposive sampling approach. The results of the functionality testing will be analyzed to determine whether this program can be used as one of the strategies to help computer or gadget users in managing rest time flexibly and can help reduce symptoms of Computer Vision Syndrome (CVS).

3.1 Synthesis of Research Findings

Multiple studies have demonstrated that regular breaks, structured through reminder software, can significantly reduce the symptoms of CVS. These symptoms include eye strain, headaches, and blurred vision. The effectiveness is primarily attributed to the enforcement of the 20-20-20 rule, which recommends taking a 20-second break every 20 minutes to look at something 20 feet away (Erdinest and Berkow, 2021), (Derbew *et al.*, 2021). Research indicates that the success of time break reminder software depends heavily on user compliance. Features such as customizable break schedules, user-friendly interfaces, and unobtrusive reminders are crucial for ensuring consistent use (Poudel and Khanal, 2020). Studies have also explored the impact of regular screen breaks on overall productivity and well-being. Findings suggest that while there might be a short-term reduction in screen time, the overall productivity and well-being of users improve due to reduced discomfort and fatigue (M. Zalat, S. Amer, G. Wassif, S. El Tarhouny, 2021), (Anggrainy, Lubis and Ashar, 2020).

3.2 Stages of Program Development With Java Programming

An elucidation of the sequential actions involved in a Java program.

Step 1: The application starts by prompting the user to enter the duration of the work session (workTime), the duration of the break session (breakTime), and the number of work/break cycles (numCycles) via a Scanner.

The user inputs the durations and the number of cycles.

The software commences the initial work session by displaying the message "Work session #1 started" and then suspends execution for the designated work duration using the TimeUnit.MINUTES.sleep() method.

Step 2: After finishing the work session, the application will show the message "Time for a break!" and initiate the break session.

The break session is initiated by temporarily suspending execution for the designated period of the break.

Following the break, the application displays the message "Time to work again!" and advances to the subsequent work session in the cycle.

Step 3: The program proceeds with the work and break cycles using a for loop and a try-catch block to handle any InterruptedException that might be thrown during the Thread.sleep() execution.

The program's terminal output demonstrates a pattern of alternating between work and break sessions, which is determined by the user's input of the number of cycles.

Step 4: The program may have a code segment that establishes the text color for the terminal output, although the precise implementation specifics are not apparent from the description. After the completion of all the work/break cycles set by the user, the software will display the message "All work/break cycles completed." This message serves as confirmation that the routine has concluded.

3.3 Discussion

The figure 1 depicts a segment of a time management application implemented in Java. It utilizes essential programming elements such as loops, try-catch blocks for handling exceptions, and user input management. This source code serves as a prime example of a basic interactive application created to schedule concentrated work sessions with intermittent pauses, using ideas similar to the Pomodoro Technique, a time management approach devised by Francesco Cirillo. The application's user-centric design is demonstrated by its command-line interface, which prompts for input parameters to personalize the length of work and rest periods, thus promoting productivity through organized time segments. The application's architecture is notably flexible, allowing for easy expansion or improvement.

The figure 2 depicts a portion of a Java application source code, illustrating a basic time management system. The code is designed to enhance user engagement in the process of allocating work and break intervals dynamically, with the potential to optimize productivity by implementing time-limited work cycles. The program employs fundamental control flow elements, such as loops and exception handling, to enable the implementation of the Pomodoro Technique or other comparable time management systems. The program additionally encompasses a command-line interface output, which showcases the program's runtime

interaction. The application's iterative design, incorporating stop intervals determined by user input, exemplifies the utilization of programming to optimize personal productivity in job management. The code's modular architecture implies the potential for scalability and compatibility with advanced task management frameworks.

Figure 1. Source code of the program

The picture 3 displays a Java application source code specifically created to improve personal productivity by implementing structured work and break sessions. The code utilizes iterative logic to alternate between user-defined intervals of concentrated work and relaxation, demonstrating an application of time management approaches like the Pomodoro Technique. Exception handling is employed to handle any disruptions in sleep times. The terminal output, shown below the code snippet, demonstrates the application's live execution, clearly indicating the commencement and completion of each work and break cycle. This program exemplifies a basic yet useful instance of customized software designed to promote discipline in work habits, potentially leading to a decrease in mental exhaustion and an enhancement in overall task effectiveness.

Figure 2. Illustrating A Basic Time Management System.

The figure 4 illustrates a Java software named 'reminder.java' that is created to organize productivity sessions by incorporating work-break intervals. This source code represents an implementation of time management software. It establishes a loop that switches between work and rest periods based on the user's input. The execution of the code is presented in an integrated development environment, and the output is exhibited in a terminal window. The program's output accurately represents the sequential progression, indicating the start and end of each interval. This application showcases a simple yet efficient approach to improve concentration and effectively allocate time, perhaps serving as a tool for empirical research on work patterns and productivity.

The development lifetime of a Java application has multiple crucial phases. At first, the development team prioritized creating a strong user input capability that allows users to define rest times in minute intervals. The fundamental element necessitated a user interface with the ability to precisely capture and store user-defined data. Subsequently, the team incorporated a reminder mechanism into the application, which was specifically designed to notify users at pre-established intervals, thereby incorporating a crucial timing feature for the program's functionality. Subsequently, efforts were focused on enhancing the user interface to guarantee promptness and simplicity of engagement, with the objective of facilitating a smooth user experience.

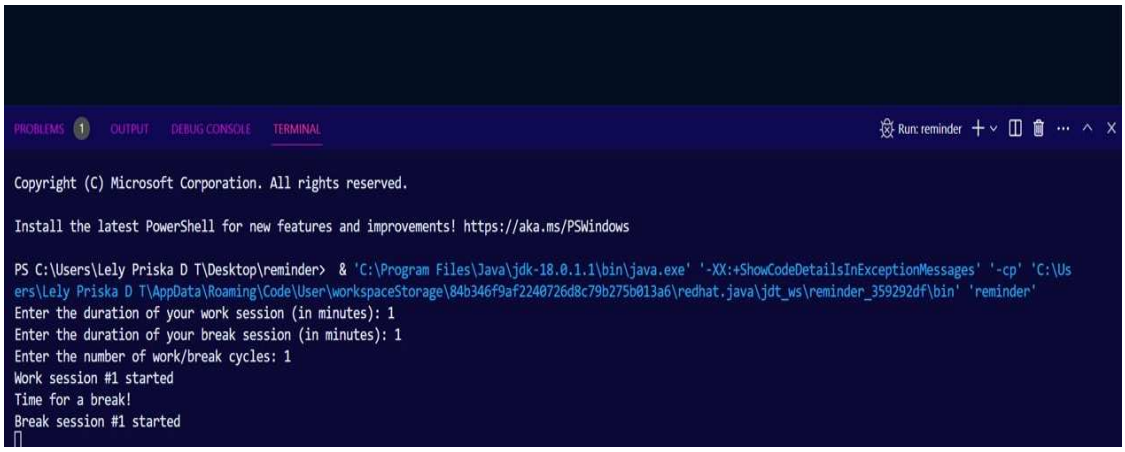

Figure 3. created to improve personal productivity by implementing structured work and break sessions

Finally, the development approach focused on the critical topic of error handling, ensuring that the application could effectively handle erroneous inputs by providing users with concise and understandable feedback. This phase was crucial for preserving the stability of the program and bolstering user confidence. While the testing showed that the functionality and compatibility of these features were well done, it was determined that further iterations were needed to enhance the visual appeal of the user interface and fully implement error handling capabilities. This would complete the program's user-centric design and make it more operationally robust.

Figure 4. Output Display on the Console after all cycles completed

Table 1. Functionality Testing Results

Program Feature	Test Description	Expected Result	Actual Result	Status (Pass/Fail)	Notes for Improvement
User Input Time	Entering rest time in minute format	Program accepts input and stores it	Runs well	Pass	-
Reminder Trigger	Reminder appears at the set time	Reminder appears on time	Runs well	Pass	-
UI Responsiveness	Interaction with the user interface	Interface is responsive and easy to use	Yes	Pass	Display lacks appeal
Error Handling	Entering invalid time input	Program displays a clear error message	Not yet available	Fail	No notes yet
Compatibility	Running the program on various devices	Program runs well on all tested devices	Runs well	Pass	-
Reminder Reliability	Reminder functions consistently	Reminder operates without failure	Runs well	Pass	-

A specific analysis combining the programming stages with the test results:

User Input Time:

Test Description: The test involves entering rest time in minute format.

Expected Result: The program should accept input and store it.

Actual Result: The program runs well.

Status: Pass

Notes for Improvement: No notes indicated, which suggests that this feature meets the expectations and requirements as planned.

Reminder Trigger:

Test Description: The reminder appears at the set time.

Expected Result: The reminder should appear on time.

Actual Result: The reminder runs well.

Status: Pass

Notes for Improvement: No suggestions provided, implying the reminder trigger feature works as intended.

UI Responsiveness:

Test Description: Interaction with the user interface.

Expected Result: Interface is responsive and easy to use.

Actual Result: The interface is responsive.

Status: Pass, but with a note.

Notes for Improvement: "Display lacks appeal" suggests that while the functionality meets the criteria for performance, the aesthetic or user experience aspect could be enhanced.

Error Handling:

Test Description: Entering invalid time input.

Expected Result: Program displays a clear error message.

Actual Result: Not yet available, indicating that the test has either not been conducted or the feature is not implemented.

Status: Fail

Notes for Improvement: No notes yet, which may mean this aspect is still under development or awaiting further analysis.

Compatibility:

Test Description: Running the program on various devices.

Expected Result: Program runs well on all tested devices.

Actual Result: Runs well.

Status: Pass

Notes for Improvement: No issues reported, indicating good cross-platform compatibility.

Reminder Reliability:

Test Description: Reminder functions consistently.

Expected Result: Reminder operates without failure.

Actual Result: Runs well.

Status: Pass

Notes for Improvement: No notes provided, suggesting reliable performance of the reminder feature.

According to the test results, the application performs effectively in most aspects, but there are certain areas that require development. These areas include enhancing the visual attractiveness of the user interface design and addressing the functioning and testing of error handling. The testing approach seems to be thorough, encompassing functional elements such as user input, system behavior (including reminder triggers and reliability), user experience (UI responsiveness), and system integration (compatibility).

To summarize, the study analysis highlights that the program exhibits robust functioning and compatibility. However, it necessitates enhancements in user interface design to enhance user engagement and satisfaction. The error handling component requires attention to guarantee that the software can effectively manage faulty inputs, which is crucial for overall usability and user experience. Refinements were made to the development based on user feedback and testing. In general, the application exhibited operational dependability, suggesting an advanced phase of development for essential functionalities, with continuous enhancements to enhance user engagement and error handling.

4. Conclusions, Limitations, and Future Directions

In this chapter, we summarize the conclusions drawn from the synthesized research on time break reminder software in mitigating Computer Vision Syndrome (CVS), discuss the limitations of the current studies, and suggest directions for future research.

4.1 Conclusions and Limitations of Current Research

The conclusions and limitations of this research are:

1. The time breaker program built functionally can work well. Flexible time settings can be an advantage for users who want to take a short break.
2. There are still functions that need to be improved, namely error handling
3. The program is still running in the console, so the user interface is not attractive

4.2 Future Research Directions

Considering the findings and constraints of the program in this investigation, below are some recommendations for further research:

1. Error Handling Development: Subsequent investigations may concentrate on enhancing the error handling system within the application. This entails the creation of error messages and suggestions that are more instructive for users in cases where they input erroneous data.
2. User Interface Design: In light of the program's current utilization of a less visually appealing user interface on a console, further investigation might be conducted to examine the creation of a more user-friendly and visually appealing graphical interface (GUI). An intelligently crafted graphical user interface (GUI) has the potential to enhance user involvement and streamline the navigation and utilization of the program's functionalities.

3. Integration with Operating Systems and Other Applications: To enhance user engagement and usability, future research should explore the possibility of integrating the program with commonly used operating systems or applications, such as digital calendars or task reminders, in order to offer a more cohesive experience.

References

Almarzouki, N. *et al.* (2021) 'Digital Eye Strain During COVID-19 Lockdown in Jeddah, Saudi Arabia', *Journal of Contemporary Medical Sciences*, 7(1), pp. 40–45. doi: 10.22317/jcms.v7i1.937.

Anggrainy, P., Lubis, R. R. and Ashar, T. (2020) 'The effect of trick intervention 20-20-20 on computer vision syndrome incidence in computer workers', *Oftalmologicheskii Zhurnal*, 1(1), pp. 22–27. doi: 10.31288/oftalmolzh202012227.

Derbew, H. *et al.* (2021) 'Assessment of Computer Vision Syndrome and Personal Risk Factors among Employees of Commercial Bank of Ethiopia in Addis Ababa, Ethiopia', *Journal of Environmental and Public Health*.

Erdinest, N. and Berkow, D. (2021) 'Computer Vision Syndrome', *Europe PMC*, 160(6), pp. 386–392.

Iyengar, K. P., Vaishya, R. and Botchu, R. (2023) 'Can We Apply Pomodoro Technique in Academic Publishing?', *Apollo Medicine* |, October(09), p. 1. doi: 10.4103/am.am.

M. Zalat, S. Amer, G. Wassif, S. El Tarhouny, and T. M. (2021) 'Computer vision syndrome, visual ergonomics and amelioration among staff members in a Saudi medical college', *International Journal of Occupational Safety and Ergonomics*, 28, pp. 1033–1041.

Poudel, S. and Khanal, S. P. (2020) 'Magnitude and Determinants of Computer Vision Syndrome (CVS) among IT Workers in Kathmandu, Nepal', *Nepalese Journal of Ophthalmology*, 12(2), pp. 245–251. doi: 10.3126/nepjoph.v12i2.29387.