WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

Eyerest: An Intelligent Eye Care Break Reminder and Digital Eye Wellness Monitoring System

Kanishka D S, Mahalakshmi M, Priyanevatha A, Mrs. N. G. Dharaniya*

Student, *Assistant Professor
Department of Information Technology, Sri Shakthi Institute of Engineering and Technology, Coimbatore-641062, Tamil Nadu, India.
Email ID: kanishkads10@gmail.com

ABSTRACT

People who spend long hours working with computers, laptops and other digital devices often forget to take regular breaks, and continuous screen use is commonly associated with eye discomfort, dryness and tiredness. Conventional timers provide basic notifications but seldom combine guided eye exercises, progress tracking, motivational rewards and social challenges. This paper presents EyeRest, a web-based digital eye wellness application that brings these features together in a single interface. The application provides user registration and login, configurable focus and break timers with start, pause, resume and reset controls, break reminders, guided eye exercises, a daily dashboard, points, badges, streaks and goals, in-application activity tracking, a friends' challenge with a leaderboard, history, settings and eye-care tips. The interface is developed with React, HTML, CSS and JavaScript using the Vite build tool. When a focus session ends, EyeRest notifies the user and guides them into a scheduled break with an eye exercise; completed sessions and breaks are recorded where supported and used to update the dashboard and rewards. EyeRest offers general wellness guidance only: it does not diagnose, treat or prevent any eye condition, and it records only activity inside the application, not total device screen time. Functional testing is planned through twelve test cases, of which one is currently supported by recorded evidence. No user-study findings or effectiveness figures are reported, because none were recorded, and further evaluation with real users is required.

Keywords: Digital Eye Strain, Break Reminder, Focus Timer, Guided Eye Exercises, Gamification, Digital Wellness, React, Behaviour Change.

1. INTRODUCTION

Digital screens are central to study, software development, office work and leisure. Many people work at a computer for hours without a pause, because the task absorbs attention and nothing in the environment signals that a break is due. Prolonged, uninterrupted screen use is commonly associated with eye discomfort, dryness, tiredness and difficulty keeping to healthy work routines [1], [2]. General guidance, such as taking short, regular breaks and looking at a distant object, is widely published [3], [4], but it is easy to forget in the middle of a task.

Modern web technologies make it practical to combine reminders, guidance and feedback in one application that runs in an ordinary browser. Timers can structure work into focus sessions, notifications can prompt a break, short exercises can give the break a purpose, and records and rewards can encourage the user to repeat the habit from one day to the next. EyeRest is built on this idea. The word "intelligent" in its title refers to the automated scheduling, reminding and guidance that the application provides; EyeRest does not use machine learning.

1.1 Existing challenges in screen-break management

Existing approaches include manual self-discipline based on published guidance, basic timers and time-management methods, stand-alone break-reminder utilities, gamified wellness applications and the screen-time dashboards of operating systems. Although each is useful, they are typically single-purpose: a timer may not guide the break, an exercise application may not track focus sessions, and a statistics tool may not provide rewards. Users who want all of these functions must combine several tools, which adds friction.

The Pomodoro technique [5] divides work into timed intervals separated by short breaks and is widely adopted, but it concentrates on productivity and does not prescribe what the user should do during the break. A further limit concerns tracking: a web application can observe only what happens inside itself, because a browser page cannot normally see the activity of other applications. Any statistics it shows must therefore be labelled as application-recorded activity and not as total device screen time.

1.2 Role of guided exercises, gamification and behaviour design

Eye-strain reviews describe behavioural measures such as breaks, varied viewing distance and blinking alongside optical and environmental ones [1], [3]. The strength of evidence for specific exercises varies, so EyeRest presents its exercises as general wellness activities that give the user something simple and concrete to do during a break.

Gamification, the use of game design elements in non-game contexts [6], is common in wellness applications. A review of empirical studies found that its effects are generally positive but depend on the context and on the users [7]. Fogg's behaviour model proposes that a behaviour occurs when motivation, ability and a prompt coincide [8]; in EyeRest the break reminder is the prompt, the short exercise makes the behaviour easy to perform, and points, badges and streaks supply motivation. Usability research adds that tools intended for frequent use must be simple and not intrusive [9], which motivates the configurable alerts, durations and sounds.

1.3 Problem statement and objectives

People who spend long hours working with computers and other digital devices may forget to take regular breaks. Continuous screen use can contribute to eye discomfort, dryness, tiredness and difficulty maintaining healthy work routines. Traditional timers provide basic notifications but may not offer an integrated experience combining guided eye exercises, progress tracking, motivational rewards and social challenges. There is therefore a need for an accessible application that brings these features together in a single digital wellness platform.

The main objective of this work is to develop a web-based digital eye wellness application that encourages healthy screen use through configurable timers, break reminders, guided exercises and progress monitoring, while stating candidly what the application does and does not do. The specific objectives are:

i. To provide user registration and login with a personalised greeting.

ii. To provide configurable focus and break timers with start, pause, resume and reset controls, and to notify the user when a session ends.

iii. To present guided eye exercises and relaxation activities during breaks, with completion feedback.

iv. To present a daily dashboard, history and daily goals based on activity recorded inside the application.

v. To motivate consistent use through points, badges, streaks, a friends' challenge and a leaderboard.

vi. To allow personalisation through settings and to present eye-care tips as general wellness guidance.

vii. To evaluate the application through functional, integration and usability testing, and to avoid medical claims.

2. MATERIALS AND METHODS

2.1 Development approach

The project follows an iterative, incremental approach suited to a small team. A minimal path consisting of the interface and a working focus timer was established first. Further modules, such as break guidance, the dashboard, gamification, history, the friends' challenge and settings, were added in successive iterations, and each iteration was checked manually before the next was begun. This approach suits an application whose user experience can only be judged by using it.

Requirement analysis identified the intended users, namely students, software developers, office workers and other screen users, and the tasks they need to perform. Table 1 summarises the ten modules that result. All of them are reached from a single web interface, and the data produced by one module, such as a completed session, is used by others, such as the dashboard, history and rewards. This connection distinguishes EyeRest from a stand-alone timer.

No.	Module	Function
1	User authentication	Login and registration; personalised greeting using the user's name or an e-mail-based fallback
2	Focus timer and break reminder	Configurable focus and break durations; start, pause, resume and reset; notification into a scheduled break
3	Guided eye exercises	Short activities during breaks with instructions, illustrations or animations and completion feedback

No.	Module	Function
4	Daily dashboard	Focus sessions, completed breaks, focus time and daily-goal progress
5	Gamification and rewards	Points, badges, streaks and progress indicators
6	Screen-time activity tracking	Tracking of activity inside EyeRest, distinct from total device screen time
7	Friends challenge	Eye-care challenges, comparison of progress and a leaderboard
8	History and progress	Records of sessions, breaks and exercise activities
9	Settings and personalisation	Timer durations, reminders, sounds and break preferences
10	Eye-care tips	Tips on breaks, comfortable screen habits, blinking, posture and relaxation

Table 1. Modules of the EyeRest application.

2.2 System architecture

The system follows a layered architecture, shown in Fig. 1. The presentation layer runs in the user's browser and provides the screens through which users register, log in, run timers, follow exercises and view their progress. The application-logic layer, also in the browser, holds the focus and break timer logic, the recording of sessions and breaks, the points, badge, streak and goal rules, and the reminder and alert dispatch. The alert layer uses browser notifications where permitted, in-application alerts and sound. The data layer retains settings, records and progress between sessions.

The architecture is intended to keep the modules logically separate, so that individual components, such as the timer logic or the gamification rules, can be tested, maintained and improved without redesigning the complete application.

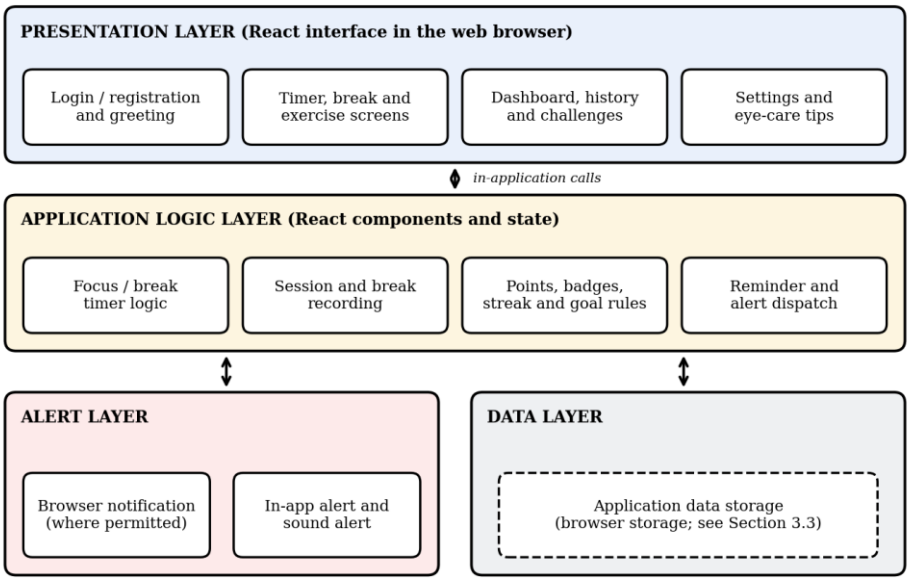


Fig. 1. System architecture of the proposed EyeRest application.

2.3 Technology stack and development environment

The interface is built with React, a JavaScript library for composing user interfaces from components [10], together with HTML, CSS and JavaScript. Vite provides the development server and production builds [11]. Table 2 lists the technologies and their responsibilities. The application runs in any current browser that supports these technologies and needs no specialised hardware; development uses Node.js with the npm package manager and a code editor such as Visual Studio Code.

Layer	Technology	Responsibility
Frontend	React, HTML, CSS, JavaScript [10]	Authentication, timer, exercises, dashboard, history, challenges, settings and tips
Build tool	Vite [11]	Development server and production builds
Alerts	Browser notifications [12], in-app alerts, sound	Inform the user when a focus session or break ends
Data storage	Browser storage [13]	Retains settings, records and progress between sessions
Backend (optional)	Node.js, Express.js, MongoDB	Not part of the verified implementation; proposed as future work

Table 2. Technology stack and responsibilities.

2.4 Data collection and processing

The system processes only the information needed for its features: account details entered at registration, the focus and break durations chosen by the user, records of completed sessions, breaks and exercises, and points, badges, streaks and challenge data. No external dataset is used and no machine-learning model is trained. The login screen states that health data stays in the browser, which indicates that browser storage is used [13]. Such storage is simple but local to one browser; data are not synchronised across devices and may be lost if the user clears the browser data.

Personal information should be collected only when necessary, passwords should never be stored in plain text, and form values and durations should be validated for type and range [14]. Validation is performed before a timer is started or a setting is saved, and failures of the environment, such as denied notification permission, blocked audio or unavailable storage, should produce an understandable message without stopping the timer. Participation in challenges and leaderboards should be optional. Because EyeRest tracks only its own activity, it avoids the privacy concerns of monitoring other applications.

2.5 Data storage design

The data that the application must retain between sessions fall into four categories: account and session information; focus and break records (type, start time, duration and completion status); settings and preferences (durations, reminder and sound options); and rewards and challenge data (points, badges, streak, invite codes and leaderboard entries). Dashboard totals, such as the number of sessions and the total time in focus sessions, are derived from the stored records by counting and summing. A database such as MongoDB would allow data to be shared across devices but requires a backend, and is therefore listed as future work.

2.6 Timer logic and break guidance

The timer can be described as a small state machine with the states idle, running, paused and finished. Start moves the timer from idle to running; pause and resume alternate between running and paused; reset returns it to idle; and when the remaining time reaches zero the timer moves to finished, triggers the alert and starts the break, which uses the same logic with its own duration. Invalid durations, such as zero or negative values, should be rejected before the timer starts. Because browsers may throttle timers in background tabs, a robust design derives the remaining time from recorded start and end timestamps instead of counting ticks; whether the implementation uses this technique has not been verified.

When a session or break ends, the dispatcher checks the user's settings and uses the enabled channels: an in-application alert, a sound alert or a browser notification, which requires the user's permission [12]. Not every channel is always available, so the in-application alert serves as the fallback. During the break the application presents a guided exercise that is brief, can be understood at a glance, asks the user to look away from the display where possible, and ends with a completion message.

2.7 Gamification and reward design

Points are awarded for completed sessions and breaks, and badges, streaks and daily-goal progress are derived from the records. Two principles guide the design. First, rewards are given for completing breaks as well as sessions, so that users are not encouraged to work longer in order to earn more. Second, the rules should be transparent, so that rewards do not appear arbitrary. Because gamification effects vary with context and user [7], the application treats rewards as support for the habit, not as its purpose.

2.8 Analysis workflow

The workflow of one focus–break cycle is shown in Fig. 2. The user logs in and reaches a personalised dashboard, configures and starts a focus session, and receives a break reminder when the configured time elapses. The user follows a guided eye exercise, after which the application records the completed session or break where supported and updates points, streaks and goals. The results appear on the dashboard and in the history, can be compared with friends in a challenge, and the user may adjust the settings for future sessions.

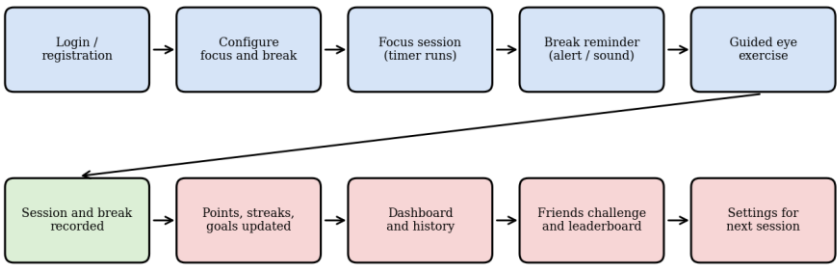


Fig. 2. Workflow of the proposed EyeRest application.

2.9 Testing and evaluation

Testing combines functional checks of each module with a review of usability and of behaviour in unusual conditions. Functional testing checks that each interface function works as specified, integration testing checks that interface, timer logic, alerts and storage operate together (for example, that a completed session appears in the history), and validation testing checks the handling of invalid input. Environment testing checks behaviour when notification permission is denied, audio is blocked or the tab is in the background, and a usability review observes real users completing a focus–break cycle. Important cases should be repeated in more than one browser. Quantitative claims about timer accuracy, performance or benefit should be reported only when supported by actual measurements.

3. RESULTS AND DISCUSSION

3.1 Implemented application

The development of EyeRest produced a working web interface that integrates the modules described above. The authentication screen is presented as EyeRest, “Smart Eye Care Companion”, with Sign In and Create Account tabs. Registration asks for full name, e-mail address, password (at least six characters) and password confirmation, and a one-click demo account allows the application to be explored without registration.

The home dashboard (Fig. 3) greets the signed-in user by name and shows status labels for the streak and points earned, with shortcuts to Focus Mode, Screen Time, Exercises, Rewards & Badges and History. It contains the Eye Care Focus Timer, which displays a 20:00 countdown in the Ready state with selectable work durations of 10, 20, 30 and 45 min and break durations of 20 sec, 30 sec, 1 min and 5 min, Start Session and Reset buttons, and toggles for Sound, Alerts and Auto-Switch. A Today’s Goals panel shows eye breaks, healthy focus time and eye exercises against daily targets, and a Daily Eye Care Tip is displayed alongside. A side navigation lists Home, Focus Timer, Eye Exercises, Screen Time, Achievements, Friends Challenge, Activity History, Daily Goals, Eye Care Tips, Relaxation Corner, Settings and Profile. The values shown describe activity recorded inside EyeRest, not total device screen time.

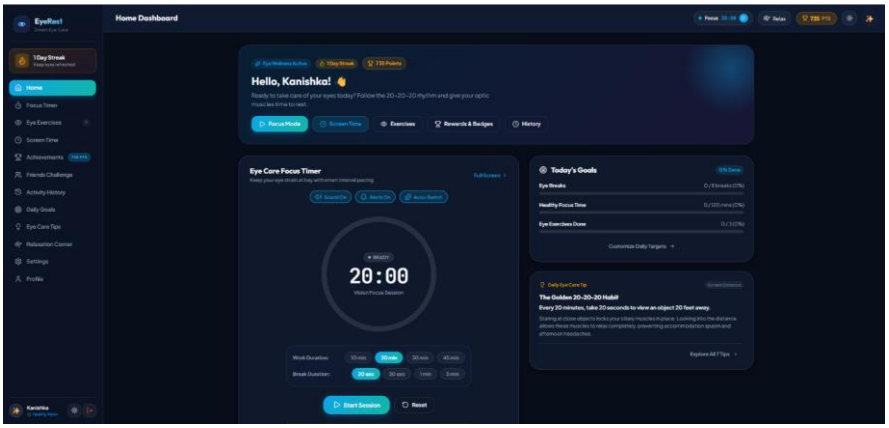


Fig. 3. Daily dashboard of the EyeRest application.

The Interactive Eye Exercises screen (Fig. 4) offers a Start Guided Routine button for all seven exercises, a counter of exercises completed today and a Gentle Guidance notice advising the user to keep movements soft and to rest if dizziness or discomfort occurs. Six exercise cards are visible: 20-20-20 Rule, Blinking Exercise, Eye Movement (4 Directions), Near and Far Focus, Figure Eight (Infinity Loop) and Palming Relaxation, each with an illustration, a duration, a key technique and a Start Exercise button.

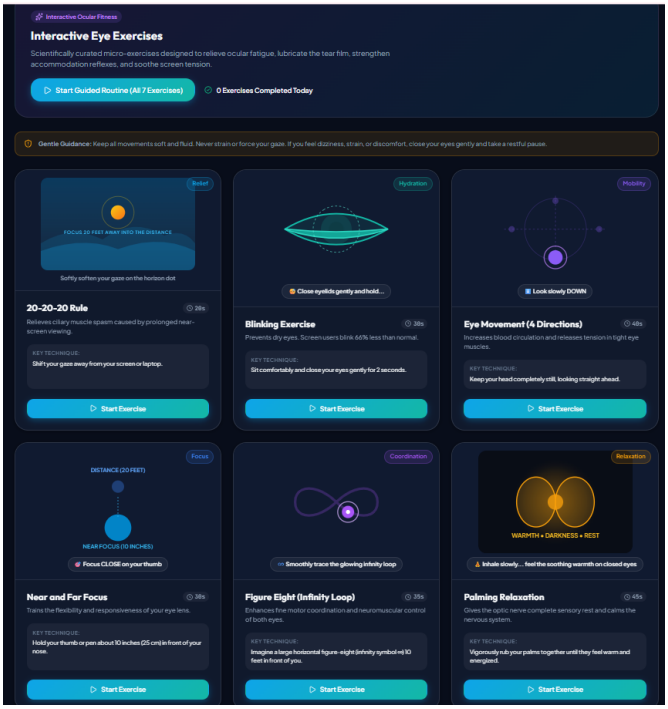


Fig. 4. Interactive guided eye-exercises screen of the EyeRest application.

The screenshots are consistent in layout and terminology: the same dark theme, rounded cards and teal accent are used throughout, and the navigation labels agree with the module names in Table 1. The break overlay, the countdown behaviour and the end-of-session alert, as well as the history, rewards, friends' challenge, settings and tips screens, are not visible in the available screenshots and are therefore not reported.

3.2 Functional evaluation

Twelve test cases were defined for the project (Table 3). Only T1, the display of the login and registration screen, is supported by recorded evidence in the form of a screenshot. The remaining eleven are pending, and no test is recorded as passed without evidence. Until they are executed, no statement is made about the correctness, reliability or cross-browser behaviour of the timer, and no percentage figure is given.

Test ID	Test scenario	Expected behaviour	Status
T1	Open the application	Sign In and Create Account options shown	Verified (screenshot)
T2	Log in with a named account and an e-mail-only account	Name shown, or e-mail-based fallback	Pending
T3	Configure and start a focus timer	Countdown starts	Pending
T4	Pause, resume and reset during a session	Timer responds correctly	Pending
T5	Short focus duration elapses	Alert shown; break begins	Pending
T6	Complete a break	Exercise shown with completion feedback	Pending
T7	Complete one session	Counters and focus time updated	Pending
T8	Complete sessions and breaks	Points, badges, streak follow the rules	Pending

Test ID	Test scenario	Expected behaviour	Status
T9	Open history after sessions	Previous sessions and breaks listed	Pending
T10	Change a duration or sound option	New value used and kept after reload	Pending
T11	Open the challenge view	Challenge and leaderboard shown	Pending
T12	Enter a zero or negative duration	Validation message; timer does not start	Pending

Table 3. Functional test scenarios and their status.

3.3 Comparison with conventional approaches

Table 4 compares the approaches discussed in Section 1.1. The comparison is qualitative and based on the general characteristics of each category, not on a measured evaluation.

Approach	Examples	Strengths	Limitations
Manual self-discipline	General guidance such as the 20-20-20 rule [4]	No tools needed; easy to understand	Relies on memory; no record or feedback
Basic timers	Pomodoro technique [5]; device timers	Simple; widely understood	Do not guide the break or record eye-care activity
Stand-alone reminders	Browser and desktop utilities	Prompt the user at intervals	Single-purpose; limited guidance and tracking
Gamified wellness apps	Points and badges [6], [7]	Motivating	Effect depends on context; may lack eye exercises
Device screen-time tools	Operating-system dashboards	Report device-level usage	Report usage rather than guide breaks
EyeRest	React web application	Combines timer, exercises, tracking, rewards, history and challenge	Records only in-app activity; effect not evaluated

Table 4. Comparison of approaches to screen-break management.

EyeRest should not be considered more effective than the other approaches without comparative testing. Its contribution is integration: it assembles established ideas and widely used web technologies into one tool, and it does not claim to advance eye-care science.

3.4 Discussion

The proposed design forms a closed loop in which each completed session and break feeds the progress that the user sees, which in turn motivates the next session. A single interface reduces the effort of moving between separate tools, and modular development can simplify maintenance and the addition of new exercises or features. The usefulness of the application currently rests on the design rationale drawn from the literature [3], [8] and not on measured outcomes.

The system also faces practical challenges. EyeRest provides general wellness guidance only and does not diagnose, treat or prevent eye disease; users with persistent symptoms should consult a qualified eye-care professional. Only activity inside the application is recorded. Browser notifications depend on permission and browser support, and timers in background tabs may be delayed. The application can prompt a break but cannot ensure that the user takes it, and data held only in the browser are not synchronised across devices. The influence of points, badges and challenges on long-term habits has not been measured, and rewards may encourage some users to pursue points instead of rest. The status of the friends' challenge as live or demonstration data, the password-handling mechanism and a camera-based feature added late in development have not been verified or evaluated for this paper.

4. CONCLUSION

EyeRest presents a technology-based approach to encouraging regular breaks during screen work. A React interface combines configurable focus and break timers, reminders, guided eye exercises, in-application activity tracking, points, badges, streaks, goals, a friends' challenge, history, settings and eye-care tips in a single application. It demonstrates that a conventional web stack can be combined into an integrated tool that prompts breaks, guides simple exercises and records progress, while stating clearly that it offers wellness guidance only.

The modular architecture provides a foundation for further development. However, the practical value of the application depends on completing the pending tests, on the reliability of timers and alerts across browsers, and on evidence from real users.

Future development can focus on executing and recording all test cases; a usability study with real users; a verified backend with cloud synchronisation; live multiplayer challenges; optional device-level screen-time tracking with explicit consent and careful privacy design; adaptive break suggestions that make no medical claims; a larger, reviewed library of exercises and animations; richer analytics and exportable reports; an installable or mobile version that can deliver reminders when the browser is closed; accessibility and multilingual support following WCAG [15]; and stronger authentication and privacy controls.

5. ACKNOWLEDGMENTS

The authors would like to thank Sri Shakthi Institute of Engineering and Technology for providing the academic environment and facilities required for this project. The authors also express their sincere gratitude to Dr. S. Prakash, Head of the Department of Information Technology, the project guide Mrs. N. G. Dharaniya, the project co-ordinator Dr. T. Esther, and the faculty members for their guidance, encouragement and support throughout the development of the project.

6. REFERENCES

1. A. L. Sheppard and J. S. Wolffsohn, "Digital eye strain: prevalence, measurement and amelioration," BMJ Open Ophthalmology, vol. 3, no. 1, Art. e000146, 2018.

2. M. Rosenfield, "Computer vision syndrome: a review of ocular causes and potential treatments," Ophthalmic and Physiological Optics, vol. 31, no. 5, pp. 502–515, 2011.

3. C. Coles-Brennan, A. Sulley, and G. Young, "Management of digital eye strain," Clinical and Experimental Optometry, vol. 102, no. 1, pp. 18–29, 2019.

4. American Optometric Association, "Computer vision syndrome." Available at: https://www.aoa.org/healthy-eyes/eye-and-vision-conditions/computer-vision-syndrome

5. F. Cirillo, The Pomodoro Technique. London, U.K.: Virgin Books, 2018.

6. S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: Defining 'gamification'," in Proc. 15th Int. Academic MindTrek Conf., 2011, pp. 9–15.

7. J. Hamari, J. Koivisto, and H. Sarsa, "Does gamification work? A literature review of empirical studies on gamification," in Proc. 47th Hawaii Int. Conf. System Sciences (HICSS), 2014, pp. 3025–3034.

8. B. J. Fogg, "A behavior model for persuasive design," in Proc. 4th Int. Conf. Persuasive Technology, 2009, Art. 40.

9. J. Nielsen, Usability Engineering. San Diego, CA, USA: Academic Press, 1993.

10. Meta Platforms. React Documentation. Available at: https://react.dev/

11. Vite. Vite Documentation. Available at: https://vite.dev/

12. Mozilla. Notifications API, MDN Web Docs. Available at: https://developer.mozilla.org/en-US/docs/Web/API/Notifications_API

13. Mozilla. Web Storage API, MDN Web Docs. Available at: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API

14. OWASP Foundation. OWASP Application Security Guidance. Available at: https://owasp.org/

15. W3C. Web Content Accessibility Guidelines (WCAG) 2.1, W3C Recommendation, 2018. Available at: https://www.w3.org/TR/WCAG21/