

Focus Mate: An AI-Assisted Mobile Assistant Unifying Task Planning, Contextual Distraction Blocking, and Behavioral Feedback

Cosmin-Teodor Marciuc

Universitatea Alexandru Ioan
Cuza
Iași, Romania
teomarciuc13@gmail.com

Ana Maria Bores

Universitatea Alexandru Ioan
Cuza
Iași, Romania
anamaria.bores@usm.ro

Adrian Iftene

Universitatea Alexandru Ioan
Cuza
Iași, Romania
adiftene@gmail.com

ABSTRACT

The constant use of mobile applications and social networks has created an “attention economy” in which platforms are deliberately engineered to keep users engaged for extended periods, fragmenting concentration and directly affecting individual productivity. Although numerous productivity tools exist, they suffer from a major lack of integration: applications are good either at organizing tasks or at restricting digital access, but they rarely communicate within a unified system. This paper presents Focus Mate, an Android digital assistant that couples planning directly with digital discipline: every calendar task can be linked to a blocking profile that is enforced automatically, at operating-system level, for the duration of the task. The system further integrates AI-based timetable import from a photograph, spatio-temporal validation of the daily agenda with transit warnings, a hybrid group-meeting scheduler that combines a deterministic algorithm with a large language model, and AI-generated narrative productivity reports. The solution was evaluated with ten users through a quantitative usability protocol; mean scores were consistently above 4.0 on a 5-point scale (interface intuitiveness 4.30, perceived blocking effectiveness 4.56, perceived timetable-import accuracy 4.30, meeting-suggestion quality 4.20), with a Net Promoter Score of +40 (mean recommendation likelihood 8.4/10). The results suggest that a unified assistant can reduce the perceived effort of protecting one’s attention.

AUTHOR KEYWORDS

attention management, distraction blocking, mobile productivity, task planning, large language models, usability testing, Android

DOI: 10.37789/icusi.2026.12

Introduction

The present-day digital environment has turned user attention into an economic resource. Davenport and Beck [1] formulated the concept of the *attention economy*, arguing that in an informationally saturated medium the user’s attention becomes the scarcest, and therefore the most valuable, asset. This perspective explains why social media platforms, entertainment applications and streaming services are designed with carefully optimized retention mechanisms: push notifications, infinite scrolling, autoplay and algorithmic personalization are not mere design options, but instruments for monetizing attention. A relevant consequence is that individual self-discipline can no longer sustainably compensate for the pressure exerted by these systems.

The cognitive cost of this pressure is well documented. Longitudinal studies by Gloria Mark [5] indicate a sharp reduction of the average attention span on a single screen, from approximately 2.5 minutes in 2004 to 47 seconds in 2023, and show that after a context switch the brain needs on average about 23 minutes to regain the previous level of concentration. The real impact of a digital distraction therefore greatly exceeds its duration: an effective productivity tool must *prevent* distractions before they occur, not merely account for them afterwards. Cal Newport [6] argues that the capacity for uninterrupted, cognitively intense *deep work* is becoming a defining competence of the contemporary economy, while Nir Eyal [2] proposes *timeboxing*—assigning each task a precise calendar interval—as an actionable alternative to classic to-do lists. Timeboxing is particularly relevant here because a task with a precise interval offers a natural frame for automatically applying contextual restrictions during that interval.

This convergence is echoed in the empirical literature on digital self-control tools. Lyngs et al. [4] survey the design space of such tools and note that most rely on the user’s own willpower at the moment of temptation rather than removing the temptation beforehand, which is precisely the gap Focus Mate targets by binding restrictions to the calendar instead of to an ad-hoc session. In the same vein, Kim et al. [3] show, in a controlled study of a lockout-style intervention, that the effort required to bypass a restriction is directly tied to how much it reduces app visits (a ten-digit typing task cut visits by about a quarter, a thirty-digit one by nearly half), which is consistent with the graded friction Focus Mate implements across its Light/Hard/Lockdown modes.

Despite this theoretical convergence, the tools available on the market remain fragmented: they are very good either at organizing tasks (to-do lists, calendars) or at forbidding digital access (blocker applications), but they rarely communicate with each other inside a unified system. Faced with dozens of daily notifications, staying focused on the current task and keeping to a schedule becomes tiresome work in itself. This raises the research question addressed by this paper: *can a single application combine intelligent planning, the blocking of digital temptations directly at operating-system level, and AI-based behavior analysis?* From this practical need—managing crowded academic schedules, planning meetings with colleagues, and controlling excessive device usage—Focus Mate was born: a digital assistant created not only to organize time, but to actively protect it.

The general objective of the work is the design and implementation of a mobile assistant that unifies, in one coherent ecosystem, task planning, contextual distraction blocking and

AI-assisted behavioral analysis. The specific objectives are: (O1) removing the decoupling between planning and execution, by directly linking every calendar task to a set of digital restrictions activated automatically at its start time; (O2) eliminating manual data-entry effort through AI-based timetable import from a photograph, validated by a strict server-side JSON schema; (O3) extending temporal planning with a spatial dimension, by dynamically computing transit times between consecutive tasks and issuing preventive warnings; (O4) treating group-meeting scheduling as a spatio-temporal problem, comparing two parallel engines (a deterministic mathematical one and an LLM-based one) through user preference; (O5) transforming raw usage data into actionable feedback through personalized narrative reports; and (O6) evaluating the solution with real users through a quantitative usability protocol.

The main contributions are: (1) a task-linked intelligent blocking system that enforces restrictions strictly while a task is in progress, including specific web addresses and keywords read directly from the browser; (2) automatic timetable import from a photograph using a multimodal generative model; (3) a transit-warning system that validates the daily agenda both temporally and spatially; (4) a Smart Meeting module with two comparatively evaluated engines, combined with map-based location optimization; (5) an AI-written productivity report contextualized by the user’s planned tasks; and (6) a usability evaluation protocol built on quantitative, self-reported metrics, applied to ten users.

The remainder of the paper is organized as follows: the next section analyzes existing applications; *Proposed Solution* presents the architecture and the main components of the system; *User Evaluation* reports the user evaluation; and the paper concludes with a summary and future work.

Existing Applications

Rather than a cold product-versus-competition comparison, the analysis follows an *aggregation* logic: each studied application contributes a valuable piece of a larger puzzle, and the contribution of Focus Mate consists precisely in reuniting these pieces under a single interface, connected by a common task-based planning layer. Five commercial applications with the most direct impact on the design decisions are discussed, complementing the research-literature perspective already introduced in the Introduction.

AppBlock¹ is one of the most mature Android blocking solutions. Its model is built around the duality between a *blacklist* (explicitly forbidden applications) and a *whitelist* (the only allowed applications, everything else being implicitly blocked), complemented by web filtering that intercepts URLs and keywords in the browser. Focus Mate adopts all these concepts in its blocking-template entity, with one essential difference: AppBlock operates on rigid time schedules, whereas Focus Mate synchronizes the blockade automatically with the interval of the active task. A custom variation was added on top of web filtering: instead of only showing a blocking screen, the browser is redirected to a Google search on a motivational theme (e.g., “productivity tips for students”), chosen randomly from a predefined list.

Feature	AB	SF	DX	HN	HI	FM
Whitelist / blacklist blocking	yes	part.	no	no	no	yes
Web filtering (URL / keywords)	yes	yes	no	no	no	yes
In-app blocking (Shorts, Reels)	no	yes	no	no	no	yes
Full lockdown with call exception	no	no	yes	no	no	yes
Reminder vs. exact alarm	no	no	no	yes	part.	yes
Spatio-temporal planning	no	no	no	no	no	yes

Table 1: Comparative synthesis of the five reference applications and Focus Mate, based on the public Play Store / vendor listings accessed in May 2026 (AB = AppBlock, SF = StayFree, DX = Detox, HN = HabitNow, HI = Habit-It, FM = Focus Mate).

StayFree² relies on informative warnings rather than absolute interdiction, in line with the behavioral-regulation literature in which *friction* inserted into an automatic habit has a notable discouraging effect. Its most innovative contribution is *in-app blocking*: disabling specific components of an application (Shorts, Reels, Stories, Explore) instead of the whole application, acknowledging that a modern application is an aggregate of functionalities with different behavioral impact. Both ideas were transposed into Focus Mate (the tolerant “Light” mode and the `inAppBlocks` field, described below in *Contextual Distraction Blocking*).

Detox³ goes in the opposite direction: a total device lockdown for a user-chosen session, with an opaque overlay and the phone application as the only tolerated exception. This philosophy became the Lockdown mode of Focus Mate, with the difference that lockdown is not an ad-hoc manually started session, but is attached to a planned task and activated automatically.

HabitNow⁴ offers a mature answer to an often overlooked detail: the distinction between a *reminder* (an ignorable informative notification) and a *scheduled alarm* (an insistent audio interception). The distinction is technical, not merely aesthetic: on Android the former uses standard notifications while the latter requires `AlarmManager` with the `SCHEDULE_EXACT_ALARM` permission. Focus Mate adopts both, together with HabitNow’s clear separation between repeated and one-off activities.

Habit-It⁵ approaches motivation through light gamification: streaks, weekly statistics, and cumulative indicators. Focus Mate adapts this philosophy to its own goal (keeping an academic schedule rather than forming an abstract habit): streaks are tracked per individual task, and a 30-day heatmap marks the “perfect” days, so that missing one task does not cancel the progress of the others.

As Table 1 shows, none of the five applications provides, in isolation, a complete experience: AppBlock covers blocking but does not tie it to structured planning; StayFree introduces in-app granularity but imposes no real constraint; Detox applies a firm blockade but without calendar or context; HabitNow offers mature notifications but cannot intervene on phone usage; Habit-It visualizes data but does not interpret behavior. Focus Mate is, by aggregation, the first solution that simultaneously covers the whole spectrum, with spatio-temporal planning acting as the connecting layer.

¹<https://appblock.app/>

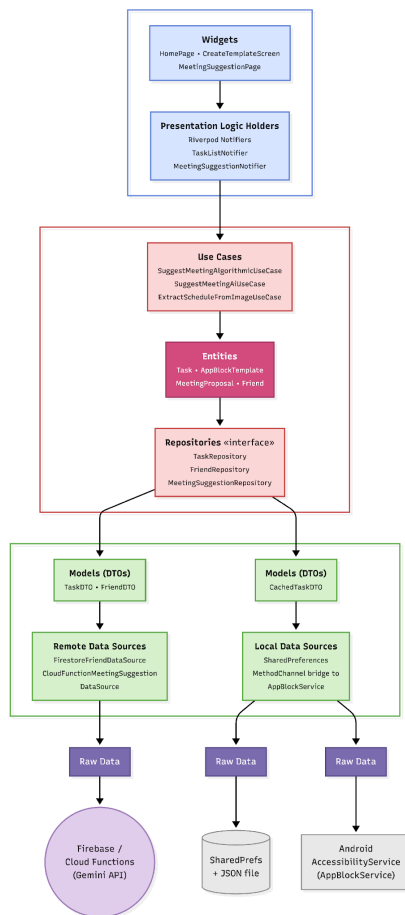


Figure 1: Clean Architecture layering in Focus Mate: dependencies point from Presentation and Data towards the Domain; native and cloud resources sit below the Data layer.

Proposed Solution

Architecture

The application is developed with Flutter⁶ and follows the Clean Architecture model, which strictly separates the graphical interface from business logic and data sources (Figure 1). The *Presentation* layer contains only visual elements and state management and never calls the database directly; the *Domain* layer contains the core entities (*Task*, *AppBlockTemplate*, *MeetingProposal*) and the business rules (use cases) and depends on no external package; the *Data* layer hosts the repositories and the external sources (Firebase, local storage, device sensors).

State management relies on Riverpod⁷, chosen over Provider, BLoC or GetX for three decisive properties: compile-time safety; independence from *BuildContext*, which allows state to be read outside the widget tree—essential for background services; and elegant composition of asynchronous sources. The transit-warning system, for instance, simultaneously needs the task list and the user’s reference locations: a single composed provider reacts automatically to changes in

either source and re-triggers the computation, avoiding deeply nested callbacks.

Since Focus Mate needs capabilities not exposed by Flutter—reading the list of installed applications, scheduling hardware-level exact alarms, monitoring the foreground application—the Dart code talks to the native Kotlin side through two channels: *MethodChannel*, for synchronous bidirectional calls, and *EventChannel*, for continuous native-to-Dart event streams. A more delicate case is the communication with the *AccessibilityService* that performs the actual blocking: this service runs in an isolated context, outside the process hosting the Flutter engine, which makes both channels unusable. The adopted solution is symmetrically shared state: *SharedPreferences* for small, frequently updated data (the prevented-distraction counters), and an atomically written JSON file for the structured blocking schedule, whose timestamp the service checks at every window-state event, reloading the content only when it has changed. This combination empirically proved more stable than *ContentProvider* or *Messenger*-based alternatives.

Finally, all AI features (timetable reading, meeting suggestion, report generation) are deliberately moved server-side, into *Firebase Cloud Functions*⁸ (Node.js 20). Centralizing the calls hides the API keys completely, allows strict schema validation of both input and output, and permits swapping the underlying model without redeploying the mobile application, at the acknowledged cost of one to two seconds of extra latency per call.

Planning, AI Timetable Import and Transit Warnings

The planning module supports detailed one-off and recurring tasks, organized in a daily agenda; the *Task* entity is not only displayed in the calendar, but also dictates to the Android system when the blocking of digital temptations must begin. To guarantee that the user is alerted on time despite the aggressive battery optimizations of modern Android (Doze mode), the application distinguishes—following *HabitNow*—between ignorable reminders, delivered as standard notifications, and exact alarms scheduled at hardware level through the *Exact Alarms API*, which reliably wake the application even from deep sleep. The notification serves as a last warning before the blocking system becomes active.

To eliminate the effort of manually entering an entire academic timetable, the application includes an AI import flow. A photograph of the timetable is securely sent to the server, where the Gemini 2.5 Flash multimodal model extracts, for every class, its name, week day and time interval. The location is deliberately *not* inferred from the image: it is taken from the *Work/Faculty* reference address defined at onboarding and attached to all imported classes, and the number of additional individual-study hours is likewise a configurable default rather than an AI guess. The server forces the model to answer in a strict JSON format, carefully validated before being returned, and the extracted timetable is presented in a review interface where the user can adjust every entry before saving.

A classic digital agenda treats time in isolation: a schedule ending at the faculty at 14:00 followed by a task at home at 14:15 would be considered perfectly valid. Focus Mate

²<https://stayfreeapps.com/>

³<https://getdetoxapp.com/>

⁴<https://play.google.com/store/apps/details?id=com.habitnow>

⁵<https://play.google.com/store/apps/details?id=ro.thinkoutside.habitit>

⁶<https://flutter.dev/>

⁷<https://riverpod.dev/>

⁸<https://firebase.google.com/products/functions>

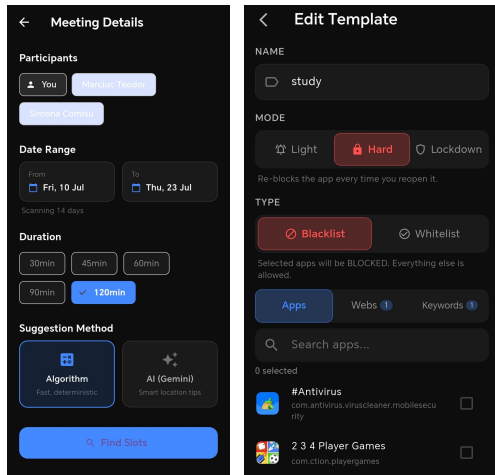


Figure 2: Focus Mate UI: (a) the Smart Meeting setup screen: participants, date range, duration and the explicit choice between the two engines; (b) the blocking-template editor: the three modes, the blacklist/whitelist duality, the Apps/Webs/Keywords tabs.

addresses this through *spatio-temporal validation*. Configuring two reference addresses (“Home” and “Work/Faculty”, resolved through the Places autocomplete) is a mandatory onboarding step—an architectural guarantee that downstream algorithms can always assume complete coordinates instead of permanently handling the null case. When the day’s schedule is loaded, the system extracts the GPS coordinates of every task and compares the available gap between consecutive tasks with dynamically estimated travel times for walking (only within a reasonable distance), public transport and driving. If the available time is shorter than the fastest valid option, a visual warning is displayed. The warning is purely advisory and dismissible, but it adds an essential layer of awareness, preventing physical delays before they happen.

Contextual Distraction Blocking

Classic blockers rely on fixed time intervals, while the real schedule of a student is variable. In Focus Mate, restrictions are activated automatically at the start time of a calendar task and deactivated at its end, with no manual start/stop effort. The core is a native `AccessibilityService` that detects every foreground-application change and checks the package name against the active template. Beyond package-level blocking, the service reads URLs typed in the browser address bar and keywords visible in the interface, enabling granular web filtering.

When a temptation is detected, the application acts on two fronts: the user is immediately redirected to the home screen through the global `GLOBAL_ACTION_HOME` action (a decision that solved a real detection race with certain launchers), and a full-screen blocking overlay (`ACCESSIBILITY_OVERLAY`) is displayed, so the interruption cannot be ignored or bypassed.

A single blocking rule cannot cover all usage contexts, so every template carries one of three modes, selectable in the template editor (Figure 2b). *Light* is the most permissive: on the first access of a restricted application an informative overlay is shown for about four seconds, after which the application remains accessible for a thirty-second tolerance

window—a low-cost interruption that creates awareness without frustration. *Hard* (the default) applies the overlay-plus-redirect sequence at every detection, with a five-second global cooldown; repeated attempts quickly become frustrating and are discouraged. *Lockdown*, inspired by Detox but bound to a planned task, is time-triggered: an opaque overlay covers the entire screen (including the launcher), showing the task name and a countdown; the only permitted exit is the phone dialer, launched via `ACTION_DIAL`, whose package is obtained from `TelecomManager`; the service detects in real time when the user leaves the dialer and automatically restores the overlay.

Blocking an entire application is often disproportionate: a student using YouTube for tutorials does not need to lose video search; the real problem is typically the Shorts feed. A template therefore carries an `inAppBlocks` field, mapping package names to lists of forbidden in-app features (shorts, reels, stories, explore, ...). Because third-party applications intentionally obfuscate their internals, detection uses a cascade of three signal levels, applied in decreasing order of stability: stable resource view identifiers (e.g., `reel_recycler` for YouTube Shorts, `clips_viewer_view_pager` for Instagram Reels); conditioned identifiers, where a search-field `EditText` triggers only in the `isFocused/isSelected` state, avoiding false positives from the mere presence of a search bar; and long, specific content descriptions, with an additional `isSelected` rule for bottom navigation tabs. On detection, an informative overlay (“Shorts is blocked”) is shown, then the forbidden surface is closed through a `GLOBAL_ACTION_BACK` event, with a twenty-second cooldown per (package, feature) pair. The application thus remains usable for its legitimate purpose while the identified sources of impulsive consumption are strictly forbidden.

Smart Meeting: Finding Optimal Group Meetings

Classic group planners (Google Calendar, Doodle) search for a common free time window while completely ignoring the spatial dimension, although a real meeting requires both a time and a place convenient for everyone. The problem is formulated as follows: given the schedules of N participants (task lists with start/end times and optional GPS coordinates), a desired duration and an acceptable day window, find at most K proposals (start, end, location) such that no participant has an overlapping task and the location is reasonable in terms of travel time for every participant. The module runs over a default 14-day horizon, generates up to eight raw proposals per day and presents only the top five after scoring. Two independent engines coexist, the user explicitly choosing one per query in the setup screen (Figure 2a); both converge into a common proposal pipeline (Figure 3).

The *algorithmic engine* has no external dependency and treats the problem strictly temporally, in four steps: collect all valid busy intervals of all participants for the target date (both weekly recurring and one-off tasks, normalizing intervals that cross midnight); sort and merge overlapping intervals into the group’s collective busy time; sweep the gaps within the acceptable window (default 08:00–22:00) and extract consecutive slots of the required duration; and package each slot as a proposal with the location deliberately marked “to be

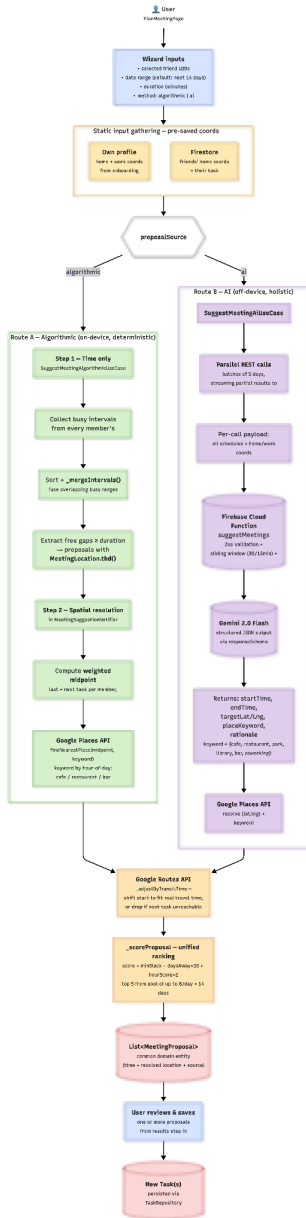


Figure 3: The Smart Meeting flow: inputs are routed to the algorithmic or the AI engine; both converge into a common list of proposals, adjusted with real transit times and ranked by a unified score.

determined”—the engine decides *when*, not *where*. The total complexity is $O(N \log N)$, dominated by sorting, and the response time stays under 50 ms.

The AI engine delegates to a Cloud Function that receives the full context: schedules, reference locations of every participant, target date, duration and proposal limit, everything validated server-side with a Zod schema (maximum lengths, numeric ranges, ISO time formats). The function builds the prompt server-side and calls the `gemini-2.0-flash` model with temperature 0.3 and *structured output*: the response schema is declared explicitly, so the model can only return a JSON object whose proposals contain the six mandatory fields (start, end, target coordinates, place keyword, rationale). The system prompt imposes a 09:00–22:00

window; explicit ranking criteria (maximize the free “slack” around the meeting for all members, prefer times close to 14:00); a closed list of six place categories (cafe, restaurant, park, library, bar, coworking) with the explicit interdiction to invent specific venue names; an anti-injection instruction to ignore any instructions embedded in task titles; and no artificial transit buffers, since real travel is validated later by the client. The answer is re-validated through Zod before reaching the client.

For algorithmic proposals, the location is resolved in a post-processing stage. A *weighted midpoint* is computed: for every member, the departure point (the last located task ending before the slot, with fallback on the Home address) and the next destination (the first located task after the slot) define a personal midpoint; the collective midpoint is the weighted mean of personal midpoints, each member weighted by $1/\text{clamp}(\text{slack}, 15, 480)$, so members with tight agendas “pull” the meeting location towards themselves. The place category follows a fixed time-of-day heuristic (cafe / restaurant / bar). For AI proposals, the coordinates and keyword come directly from the model. In both cases the pair (coordinates, category) is turned into a real venue through Places Nearby Search, with per-request caching that reuses results within roughly 200 m for the same keyword. A final adjustment stage, identical for both engines, computes real travel durations through the Routes API (with a Haversine fallback and a locally persisted cache): if a member cannot arrive in time, the proposal is shifted forward to the latest arrival while preserving the duration, and if after shifting a member can no longer reach their next fixed task, the proposal is dropped entirely. All surviving proposals pass through the same scoring function, $\text{score} = \text{minSlack} - 10 \cdot \text{daysAway} + 2 \cdot \text{hourScore}$ with $\text{hourScore} = -|\text{hour} - 14|$, and the top five reach the interface. The three weighting constants (10, 2, and the implicit weight of 1 on *minSlack*) were set by hand during development rather than tuned or varied systematically; a sensitivity analysis over these constants, and over the 08:00–22:00 and 09:00–22:00 window bounds, is left for future work.

The two engines are complementary rather than competing: the algorithmic one is deterministic, free and nearly instantaneous, making it the natural fallback; the AI one takes 2–10 seconds and incurs per-request cost, but integrates contextual reasoning (a short Friday-evening meeting receives *bar*, a long working-day one *coworking*) and justifies each proposal through its rationale field. Keeping both as parallel options was directly supported by the user preferences reported in *User Evaluation*.

Monitoring and the AI Report

The monitoring module extracts application-usage data directly from the operating system (`UsageStatsManager`) and presents clear statistics: most used applications, hourly distributions, per-task streaks, a 30-day heatmap of “perfect” days, and—connecting the blocking module to the analytics—the number of interventions of the blocking system. A distinctive element is the contextualization of screen time with respect to the running task. A simple switch on every task, “Offline session” (`isOfflineFocus`), lets the user declare the intention not to use the phone at all during that task (reading, screen-free break); in that case any usage during the interval is automatically counted as distraction. For regular

(online) tasks, time is categorized through the contextual classification of applications as productive, distracting or neutral. The design deliberately transfers the declaration of intent to the user at planning time, instead of guessing it heuristically—Chrome can be both a study tool and a distraction source—so the application measures conformance to an explicit self-commitment rather than an external judgment.

At the end of each week, the aggregated statistics are serialized into a textual summary and sent to the server-side model, which writes a personalized, encouraging narrative report explaining where the user excelled and what can be improved, with concrete advice. The report is displayed directly in the application after a few seconds of generation.

Data Handling and Consent

Focus Mate is, by design, a data-intensive application: to deliver its features it reads the foreground application and on-screen text (for blocking), typed URLs and keywords (for web filtering), device GPS coordinates and two reference addresses (for spatio-temporal validation and Smart Meeting), application-usage statistics (for monitoring), and it transmits photographs of timetables, weekly usage summaries, and—for Smart Meeting—the schedules and reference locations of every invited participant to server-side AI models. All of this processing is local-first where possible (blocking decisions, usage statistics) and is sent to the server only for the three AI features described above (timetable import, meeting suggestion, report generation), where it is used solely to produce the requested output and is not persisted beyond what is required to serve the request.

Because Smart Meeting shares one participant's schedule and reference locations with the other invited members and with the AI engine, the feature also touches the data of people who are not the app's primary user and who did not go through the onboarding flow themselves; in the current implementation, consent for this sharing is assumed to be given implicitly when a user adds someone as a friend and includes them in a meeting query, and is not separately requested from the invited participant. Making this consent explicit—for instance by requiring invited participants to confirm sharing before their schedule is used in a proposal—is identified here as a required change for any deployment beyond controlled usability testing.

The usability evaluation reported in *User Evaluation* did not go through a formal institutional ethics review, as is common for a small, non-clinical, university project. Participants were informed verbally, before installing the application, about what the app does and what data it collects, and their participation was voluntary. We flag this explicitly as a limitation: a formal informed-consent procedure, with a written form and, for participants below the age of majority, guardian consent, is necessary for any future, larger-scale or published evaluation, and is planned for the extended study described in *Conclusions and Future Work*.

User Evaluation

Protocol and Metrics

The application was evaluated through usability testing in May–June 2026, on a group of ten participants recruited from the author's personal circle, mainly from the academic environment: eight students, one employee and one with another occupation, aged between 13 and 25 (median 21). Four

Code	Metric	Mean	Scale
SML	Perceived ease of use of the interface	4.30	1–5
RFS	Perceived effectiveness of blocking	4.56	1–5
AIO	Perceived accuracy of timetable import	4.30	1–5
CSI	Perceived quality of meeting suggestions	4.20	1–5
UR	Usefulness of the AI report (auxiliary)	3.90	1–5
ACS	Perceived accuracy of contextual categorization (auxiliary)	4.40	1–5
—	Mean recommendation likelihood	8.40	1–10
NPS	Net Promoter Score	+40	−100–+100

Table 2: Mean scores over the ten participants. NPS is computed as %Promoters (score 9–10) minus %Detractors (score 0–6) from the individual recommendation-likelihood responses; scores of 7–8 count as Passives and are excluded from the calculation.

respondents had never used similar applications, three use them rarely and three daily; six estimated their daily digital-distraction time between 2 and 4 hours, three above 4 hours and one between 1 and 2 hours. The group size follows the recommendation of Nielsen [7] that five users are sufficient to uncover the majority of usability problems, and doubles it. Participants answered a profiling pre-questionnaire, installed the application, used it for several days and completed a final evaluation questionnaire; see *Data Handling and Consent* above for how participants were informed prior to testing.

Five concrete scenarios had to be covered by every participant, in a freely chosen order (the goal being independent verification of every functionality, not the measurement of a specific flow): (1) manually adding tasks to the calendar; (2) photographing a timetable and importing it automatically with the AI; (3) configuring a blocking profile, associating it with a task and observing whether the system enforces the restrictions automatically; (4) planning a group meeting with Smart Meeting; and (5) requesting the AI weekly report.

Four primary quantitative metrics were defined, all measured through 5-point Likert items in the final questionnaire and hence reflecting participants' *perceptions* rather than instrumented system counters: SML, the perceived ease of use of the interface; RFS, participants' perceived effectiveness of the automatic blocking module when they actively tried to bypass it; AIO, participants' perceived accuracy of the AI timetable import; and CSI, the perceived quality of the Smart Meeting suggestions. Two auxiliary metrics (UR, the usefulness of the AI report; ACS, the accuracy of contextual categorization) and a 1–10 recommendation-likelihood item, from which the Net Promoter Score below is derived, complete the picture. Because the application does log blocking interventions and import corrections internally (*Contextual Distraction Blocking and Planning, AI Timetable Import and Transit Warnings* above), instrumented, objective counterparts to RFS and AIO are a natural extension of the protocol and are discussed in *Conclusions and Future Work*.

Quantitative Results

As Table 2 shows, all four primary perceived-quality scores are consistently above the 4.0 threshold on the 5-point scale, which in the usual interpretation of Likert questionnaires places the application in the “above expectations” zone. The SML score of 4.30 indicates an interface perceived as intuitive by the large majority of respondents, and the RFS score of 4.56—the highest of the whole set—shows that participants perceived the automatic blocking module as working almost flawlessly when they actively tried to bypass it. AIO (4.30) and CSI (4.20), which target the two functionalities with the largest generative-AI component, are also consistently

above the “good” threshold, suggesting that the integration of generative models is perceived by end users as bringing real value. UR (3.90), although slightly lower, remains positive; participant comments suggest it reflects varied expectations about the type of narrative feedback rather than a technical problem.

Of the ten individual recommendation-likelihood scores (8, 9, 8, 9, 8, 7, 10, 8, 9, 8), four are Promoters (9–10, 40%), six are Passives (7–8, 60%) and none is a Detractor (0–6, 0%), giving $NPS = 40\% - 0\% = +40$, a positive score on the -100 to +100 scale, alongside a mean recommendation likelihood of 8.4/10. The absence of any Detractor is consistent with the generally favorable qualitative feedback discussed next, though the majority-Passive composition indicates room to convert satisfied users into active advocates.

Qualitative Observations

The open question about the two Smart Meeting engines produced a nuanced distribution. Four participants clearly preferred the AI variant, motivating that it “feels more human”, “considers more possibilities, the way I would” and gives “contextually better results”. Four considered the two methods of similar usefulness, observing that the AI variant is richer but less consistent between runs, while the algorithmic one is faster and predictable. Two explicitly preferred the algorithmic variant, invoking response time and predictability (“I don’t have the patience to wait for the AI variant”). This distribution supports the architectural decision to keep both engines as parallel options instead of choosing a single one: the preference is real but not unanimous, and different users privilege different criteria—speed and determinism versus contextuality and variety. A direct, same-input comparison of the two engines (number of proposals generated, proposals surviving the transit-feasibility stage, and latency) would complement this preference-based evidence and is planned as follow-up work.

Eight out of ten respondents found the configuration of a blocking profile intuitive. Two observations deserve attention: one participant found it unintuitive that the blocking page is named “Focus”, and another reported that the profile configuration was hard to find on first use. The most relevant improvement suggestions were blocking whole categories of applications (“games”, “social media”) without selecting each application individually—raised independently by two participants—and clarifying how whitelists interact with keyword filters.

The favorite-feature question produced three dominant directions: the photo timetable import (four participants, for the time saved), the Smart Meeting module (three, for practical usefulness) and the statistics page with the visual report (three; one participant explicitly remarked that the application “made me realize how much time I waste”). Seven out of ten participants reported no technical problems. Three concrete observations were recorded: a language inconsistency (a few words left in Romanian on the focus screen); a real functional bug in the whitelist-keyword interaction, where keyword-based web filtering fired even when the active template was set to *whitelist* mode, i.e., the keyword check was not gated by the template’s blacklist/whitelist type before being applied (subsequently reproduced and documented); and small discrepancies between the real timetable and the automatically

extracted one, mitigated by the possibility to edit the result after import. No observation indicated a crash or data loss.

Limitations of the Validation

Although the sample of ten participants exceeds the minimum threshold recommended in the usability-testing literature, it remains small for statistically strong conclusions. Recruitment from the author’s personal circle, predominantly academic, introduces a risk of demographic homogeneity and of favorable bias; employees from diverse fields, users over 30 and users unfamiliar with productivity applications are underrepresented. The testing duration of a few days captures first impressions but not long-term usage patterns, and satisfaction scores may include a novelty effect. All primary and auxiliary metrics reported above are self-reported perceptions collected through a single post-use questionnaire, not instrumented measurements, which further limits how strongly they can be generalized. An extended evaluation—larger and demographically diversified sample, recruited outside the personal circle, over at least two continuous weeks, combining self-report with the instrumented counters already logged by the application, and conducted under a formal informed-consent procedure (*Data Handling and Consent* above)—is a clear direction for future work.

Conclusions and Future Work

Focus Mate validated the central hypothesis of the research by directly coupling planning with digital discipline. Through its contributions—contextual OS-level blocking bound to the active task, AI timetable extraction from a photograph, the hybrid Smart Meeting engine and narrative AI reporting—the system was consistently rated above expectations by the ten evaluation participants on all four primary perceived-quality metrics, suggesting that a unified digital assistant can help reduce the perceived effort of protecting one’s attention; confirming an objective, measured reduction in cognitive load or distraction time would require the instrumented, longitudinal follow-up outlined below.

The current implementation has three acknowledged limitations: the blocking mechanism is exclusive to the Android ecosystem (an iOS port would require rewriting the logic on top of the Screen Time API); the probabilistic nature of generative models can introduce tone variations in the reports even under a strictly enforced JSON structure; and URL extraction from accessibility nodes, effective on major browsers, can be partially evaded by niche browsers.

Several directions extend the solution beyond its current frame. Replacing or complementing the self-reported RFS and AIO metrics with the instrumented counters the application already logs (blocking interventions attempted/succeeded, timetable fields extracted/corrected) would turn *User Evaluation* into a mixed objective-subjective protocol, directly addressing the main methodological limitation identified above. A head-to-head, same-input comparison of the two Smart Meeting engines, and a small sensitivity analysis of the proposal-scoring constants (*Smart Meeting: Finding Optimal Group Meetings* above), would strengthen the evidence for objective O4. Full cloud synchronization of the locally stored statistics would enable multi-device productivity analytics. An “Accountability” mode would let a group of friends share blocking templates and commit together to a common session, exploiting the existing friends infrastructure of

Smart Meeting, provided the explicit per-participant consent flow discussed in *Data Handling and Consent* is in place. Extending the deep-work timer to wearables through haptic feedback would alert the user without the phone leaving the pocket. An optional, per-category gamification layer would integrate naturally with the narrative AI report. A desktop agent for Linux, macOS and Windows, combined with a cross-platform browser extension and centrally aggregated statistics, would close the circle of digital discipline beyond a single screen. Finally, the empirical validation should be extended longitudinally and comparatively, under a formal informed-consent procedure, including head-to-head testing against the applications analyzed earlier in *Existing Applications*.

Attention has become the currency of the modern economy, and information technology is simultaneously a cause of today's cognitive fragmentation and our best chance to reverse it. Focus Mate is an experiment in this direction: a demonstration that algorithmic innovation and agent-based systems can be used not to capture and sell attention, but to protect it.

References

- [1] Thomas H. Davenport and John C. Beck. 2001. *The Attention Economy: Understanding the New Currency of Business*. Harvard Business School Press, Boston, MA, USA.
- [2] Nir Eyal. 2019. *Indistractable: How to Control Your Attention and Choose Your Life*. BenBella Books, Dallas, TX, USA.
- [3] Jaejeung Kim, Joonyoung Park, Hyunsoo Lee, Minsam Ko, and Uichin Lee. 2019. LocknType: Lockout Task Intervention for Discouraging Smartphone App Use. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–12. doi:10.1145/3290605.3300927
- [4] Ulrik Lyngs, Kai Lukoff, Petr Slovak, Reuben Binns, Adam Slack, Michael Inzlicht, Max Van Kleek, and Nigel Shadbolt. 2019. Self-Control in Cyberspace: Applying Dual Systems Theory to a Review of Digital Self-Control Tools. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–18. doi:10.1145/3290605.3300361
- [5] Gloria Mark. 2023. *Attention Span: A Groundbreaking Way to Restore Balance, Happiness and Productivity*. Hanover Square Press, Toronto, Canada.
- [6] Cal Newport. 2016. *Deep Work: Rules for Focused Success in a Distracted World*. Grand Central Publishing, New York, NY, USA.
- [7] Jakob Nielsen. 2000. Why You Only Need to Test with 5 Users. Nielsen Norman Group. <https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/>.