# Architectural Blueprint for the Next-Generation Digital CADDI Intervention: Enhancing Clinical Efficacy through Advanced UI Patterns and AI-Augmented Engineering

## Executive Summary: The Clinical and Technical Imperative

The development of the "Spark" application represents a critical intersection between clinical psychology and modern software engineering. By digitizing the Cognitive Behavioral Therapy for ADHD Inattentive Presentation (CADDI) protocol, the application aims to address a pervasive public health challenge: the management of Adult ADHD, specifically the inattentive subtype (ADHD-I), which is characterized by profound difficulties in regulation of attention, activity, and goal-directed behaviors.1 While traditional pharmacotherapy remains a frontline treatment, psychosocial interventions like CADDI are essential for addressing the functional impairments—such as disorganization, procrastination, and time blindness—that medication alone often fails to resolve.2

The current iteration of the Spark app, hosted on a web client, serves as a foundational delivery mechanism. However, to achieve the user's goal of getting the application "right and tight," a rigorous translation of clinical mechanisms into digital user experience (UX) patterns is required. This report provides an exhaustive analysis of the necessary feature set, grounded in the specific deficits associated with ADHD-I. It moves beyond superficial feature listing to explore the neurocognitive rationale behind each interface decision, the specific React ecosystem libraries best suited for implementation, and the precise "prompt engineering" strategies required to leverage Artificial Intelligence (referred to by the user as "Google Antigravity") to accelerate development.

The analysis synthesizes data from 112 research snippets, covering clinical trials of the CADDI protocol, comparative analyses of React component libraries, and guidelines for neuro-inclusive design. Key findings indicate that the efficacy of a digital CADDI intervention hinges on externalizing executive functions. The application must act as a "cognitive prosthetic," offloading the burden of prioritization (via the Eisenhower Matrix) and temporal processing (via Time Blocking) from the user's working memory to the digital interface.4 Furthermore, to combat the high attrition rates observed in CBT trials—21.3% even in controlled settings 1—the application must utilize gamification and low-friction interaction models (like swipe interfaces) to sustain engagement through dopaminergic reinforcement.6

This report is structured to guide the transformation of Spark from a content delivery system into a dynamic therapeutic tool. It addresses the architectural decisions required to build a robust, accessible, and clinically valid application, utilizing the power of Generative AI to bridge the gap between complex clinical requirements and executable code.

## Part I: The Clinical Foundation – Digitizing the CADDI Protocol

### 1.1 The Specificity of ADHD Inattentive Presentation

To build a "tight" application, one must first understand the specific clinical target. The CADDI protocol is not a generic ADHD treatment; it is tailored for the *Inattentive Presentation* (ADHD-I). Unlike the combined type, which presents with overt hyperactivity, ADHD-I is characterized by internal dysregulation: staring out windows, losing track of thoughts, and a "sluggish cognitive tempo" often misdiagnosed as depression or anxiety.1 The clinical data suggests that high levels of inattention imply severe impairment in everyday life, particularly in the organization of activity and the initiation of tasks.1

The implication for the Spark app is that the UI must be aggressive in its simplicity and clarity. A user with ADHD-I does not need more information; they need structure. The app must reduce the "noise" of daily life. Research indicates that the CADDI protocol incorporates skills training in three specific domains: organization, behavioral activation, and mindfulness.1 Therefore, the digital architecture must have distinct modules corresponding to these clinical pillars:

| **CADDI Module** | **Clinical Goal** | **Digital Feature Translation** |
| --- | --- | --- |
| **Organization** | Reduce clutter and prioritize tasks based on importance rather than urgency. | **Eisenhower Matrix** (Drag-and-Drop Interface) |
| **Behavioral Activation** | Overcome inertia and the "Wall of Awful" to initiate tasks. | **Gamified Micro-Steps** & **Swipe-to-Start UI** |
| **Mindfulness** | Increase awareness of attention drift and emotional state. | **Visual Timers** & **Implementation Intention Prompts** |

### 1.2 Addressing Attrition and Engagement

A critical insight from the CADDI clinical trials is the challenge of retention. In the randomized controlled trial comparing CADDI to Dialectical Behavior Therapy (DBT), attrition was 21.3%, and while satisfaction was high, maintaining consistent practice of home assignments was difficult.1 In a digital context, "home assignments" are the app usage itself. If the app is tedious, users will churn.

The "tightness" the user seeks is likely a function of *adherence*. How does the app ensure the user returns? The research suggests that the interface must provide immediate, tangible feedback. In face-to-face therapy, the therapist provides social reinforcement. In the app, this must be replaced by **Gamification Mechanics** (confetti, streaks, progress rings) and **Low-Friction Interactions** (swiping cards). These are not merely aesthetic choices; they are prosthetic motivators designed to bypass the dopamine deficit that makes task initiation painful for the ADHD brain.6

### 1.3 The Role of Digital Implementation Intentions

The CADDI protocol utilizes "behavior analysis" to identify reinforcement contingencies—understanding *why* a behavior happens or fails to happen.8 A key component of this is "Implementation Intentions" or "If-Then" planning (e.g., "If I get distracted by my phone, then I will put it in the drawer").3 The digital implementation of this requires an interface that captures these intentions explicitly. It is not enough to have a to-do list; the app must ask the user, "What will you do if you get stuck?" This prompts the user to simulate the obstacle and the solution, a cognitive process that significantly increases the likelihood of goal attainment.9

The current "Spark" app must evolve to include these meta-cognitive layers. It is not just about *what* to do (the task), but *how* to do it (the plan). The integration of AI ("Google Antigravity") is particularly potent here, as an LLM can generate personalized implementation intentions based on the specific task the user is struggling with, effectively simulating the guidance of a CBT therapist in real-time.

## Part II: The Prioritization Engine – The Eisenhower Matrix

### 2.1 Clinical Rationale: Externalizing Executive Decision Making

One of the most debilitating symptoms of ADHD is "prioritization paralysis"—the inability to distinguish between tasks that are urgent (loud, immediate, anxiety-inducing) and tasks that are significant (long-term value). The research highlights that the mental effort required just to decide what to do first can completely derail the day before it begins.4 This is often referred to as "cognitive noise."

The Eisenhower Matrix offers a structured heuristic to sort this noise into four quadrants:

1. **Do First (Urgent & Important):** Crises, deadlines.
2. **Schedule (Not Urgent & Important):** Strategic planning, exercise, skill building.
3. **Delegate (Urgent & Not Important):** Interruptions, some emails.
4. **Delete (Not Urgent & Not Important):** Time wasters.

For the Spark app, implementing this matrix is critical because it forces a *binary decision methodology*. Instead of staring at a linear list of 20 items (which causes overwhelm), the user processes items one by one, asking "Is this urgent? Is this important?" This externalizes the executive function of prioritization. The app provides the "scaffold" that the ADHD prefrontal cortex struggles to construct.10

### 2.2 Technical Implementation Strategy: React Libraries and State

To make this feature "right and tight," the interaction must be fluid. A clunky interface will break the user's fragile focus. The standard UI pattern for this is **Drag-and-Drop (DnD)**.

#### 2.2.1 Library Selection Analysis

The React ecosystem offers several powerful libraries for drag-and-drop interactions. A comparative analysis based on the research snippets suggests the following:

| **Library** | **Key Features** | **Pros for ADHD App** | **Cons** |
| --- | --- | --- | --- |
| **dnd-kit** 11 | Modern, modular, lightweight, accessible. | Highly flexible; supports grid layouts (2x2 matrix) and collision detection effectively. | Slightly steeper learning curve than older libraries. |
| **React-Beautiful-DND** 12 | Fluid animations, "physical" feel. | Visually satisfying ("juicy") interactions which help engagement. Strong keyboard accessibility. | **Deprecated/Maintenance Mode**. Does not support nested containers well.14 |
| **React DnD** 15 | HTML5 backend, extremely customizable. | Good for very complex interactions. | Overkill for a simple matrix; requires more boilerplate code. |

**Recommendation:** Utilize **dnd-kit** or a fork of **React-Beautiful-DND** (like @hello-pangea/dnd) if the "physicality" of the list movement is a priority. The "tightness" of the app will come from the *animation* feedback—when a task is dropped into a quadrant, it should snap into place with a satisfying micro-interaction.

#### 2.2.2 State Management and Persistence

The implementation requires a robust state management strategy. Since tasks move between quadrants (which are essentially status categories), the state should be normalized.

* **Data Structure:** An array of Task objects, each with a quadrantId property.
* **Persistence:** Changes must be saved immediately to the backend (Firebase/MongoDB) to prevent data loss, which is catastrophic for trust in an ADHD tool.
* **Constraint Logic:** A critical feature for ADHD management is *limiting* the "Do First" quadrant. The app should programmatically restrict this container to 3-5 items.16 If the user tries to drag a 6th item in, the app should reject the drop and trigger a toast notification: "Whoa there! Let's focus on these 5 things first." This acts as a digital boundary, preventing the user from overcommitting.

### 2.3 Asking "Google Antigravity": AI Implementation Prompts

To leverage AI to build this complex component, the user needs to provide a prompt that specifies the library, the layout, and the logic.

**Prompt Strategy for Component Generation:**

"Act as a Senior React Engineer. I need a component called EisenhowerMatrix for an ADHD productivity app.

Requirements:

1. **Library:** Use dnd-kit for drag-and-drop functionality.
2. **Layout:** Use CSS Grid (Tailwind) to create a 2x2 matrix.
   * Top-Left: 'Do First' (Red tint).
   * Top-Right: 'Schedule' (Blue tint).
   * Bottom-Left: 'Delegate' (Green tint).
   * Bottom-Right: 'Delete' (Gray tint).
3. **Sidebar:** A collapsible 'Inbox' sidebar on the left containing uncategorized tasks.
4. **Logic:** Users drag items from the Inbox to a quadrant.
5. **Constraint:** The 'Do First' quadrant must limit items to a maximum of 5. Implement a collision detection strategy that rejects the drop if the limit is reached.
6. Accessibility: Ensure all drop zones are keyboard accessible via Tab and Space/Enter.  
   Provide the full TypeScript code including the DndContext, SortableContext, and the state update logic."

**Prompt Strategy for Feature Insight (The "Why"):**

"Act as a Clinical Psychologist specializing in ADHD. Explain to a user *why* they should use the Eisenhower Matrix feature in the Spark app. Generate a tooltip text that appears when they hover over the 'Do First' quadrant. The tone should be empathetic but firm, explaining the danger of 'urgency addiction'."

## Part III: Temporal Prosthetics – Time Blocking

### 3.1 Clinical Rationale: Correcting Time Blindness

"Time blindness"—the inability to accurately sense the passage of time or estimate how long a task will take—is a hallmark of executive dysfunction.5 A standard to-do list is "time-agnostic"; it presents a task like "Write Report" (4 hours) next to "Email Bob" (5 minutes) as visual equals. This leads to poor planning and the inevitable failure to complete the list, fueling the cycle of shame and avoidance.1

**Time Blocking** (or Time Boxing) addresses this by giving tasks a *spatial* dimension on a calendar. It forces the user to confront the finite nature of the day. Research indicates that when tasks are broken into manageable chunks and placed on a timeline, it builds executive function skills and reduces the anxiety of the "unknown".5 For the Spark app, this feature is not just a calendar; it is a reality-testing machine.

### 3.2 Technical Implementation Strategy: The Digital Timeline

#### 3.2.1 Library Comparison: React-Big-Calendar vs. FullCalendar

Implementing a robust calendar view from scratch is notoriously difficult due to date math and edge cases (time zones, leap years). Existing libraries are essential.

| **Library** | **Characteristics** | **Verdict for Spark App** |
| --- | --- | --- |
| **React-Big-Calendar** 18 | Lightweight, built for React, uses Flexbox. | **Strong Contender.** It is "closer to the metal" of React, allowing for easier custom styling with Tailwind to make it look "friendly" rather than "corporate." |
| **FullCalendar** 18 | The industry standard. Extremely feature-rich (dragging, resizing, resource views). | **Robust Choice.** If the app needs complex recurrence rules (e.g., "Every 3rd Tuesday"), this is safer. However, the free version has limits. |
| **Planby** 21 | Specialized for EPG/Timeline views (horizontal scrolling). | **Niche Choice.** Good if the app visualizes the day as a horizontal stream, which can be intuitive for some ADHD users, but less standard. |

**Recommendation:** **React-Big-Calendar** is likely the best fit for a "tight" custom app. It allows the developer to strip away the corporate look of standard calendars. The "events" can be styled as colorful "blocks" that visually represent the energy required for the task.

#### 3.2.2 Designing for the ADHD Brain

Standard calendars are often cluttered. The Spark app's Time Blocking feature must be **visually quiet**.

* **Current Time Indicator:** A prominent red line moving down the screen helps anchor the user in the "Now".22
* **Visualizing Gaps:** The app should highlight "free time" as positive space, not just empty space.
* **Draggable Durations:** Users with ADHD often underestimate time. The UI must allow dragging the bottom edge of a block to extend it easily. If a task runs over, the user "stretches" the block, and the subsequent blocks should cascade (shift down) automatically—a feature known as "magnetic scheduling" which reduces the friction of rescheduling.17

### 3.3 Leveraging "Google Antigravity": AI-Assisted Planning

The "Blank Page Problem" applies to schedules too. Users often know *what* they need to do but freeze when trying to slot it into a grid. This is the perfect use case for Generative AI.

The "Magic Plan" Feature:

The user enters a dump of tasks: "I need to study, clean the kitchen, and call mom."

The AI processes this and returns a structured schedule.

**Prompt Strategy:**

"Act as an Executive Function Coach.

Input: A list of unstructured tasks: [User Input] + Current Time:.

Task:

1. Estimate the duration for each task (be generous, add 20% buffer for ADHD time blindness).
2. Order them logically (e.g., high-focus tasks like 'studying' first, low-focus 'cleaning' later).
3. Insert 10-minute 'transition breaks' between tasks.
4. Return a JSON array of objects compatible with react-big-calendar: ``.  
   Constraint: Ensure the schedule does not exceed 8 hours total."

This prompts the AI to do the heavy lifting of *sequencing* and *estimating*, two high-load cognitive tasks. The React app then simply renders the JSON as visual blocks.

## Part IV: The Neurochemistry of Engagement – Gamification and Swipe UI

### 4.1 Clinical Rationale: The Dopamine Bridge

ADHD is chemically characterized by a dysregulation of dopamine, the neurotransmitter responsible for reward and motivation. This creates a "motivation deficit" where the intrinsic reward of completing a task is insufficient to initiate it.6 **Gamification** is not about making the app a game; it is about providing *extrinsic* prosthetic rewards to bridge the gap until the intrinsic satisfaction kicks in.

Research suggests that "streaks" and "rewards" (like unlocking badges or visual celebrations) can effectively maintain engagement in digital health interventions.6 However, the design must be careful not to become a "distraction engine." The rewards must be tied strictly to *functional outcomes* defined in the CADDI protocol (e.g., completing a behavioral analysis, finishing a time block).

### 4.2 Technical Implementation: "Juicy" UI Patterns

#### 4.2.1 The Swipe Interface (Backlog Triage)

Managing a long backlog of tasks is overwhelming. Traditional list management (checkboxes) is high-friction. The "Tinder" swipe pattern (Right for "Do Today", Left for "Snooze") utilizes **embodied cognition**. The physical act of swiping "away" a task helps the user mentally discard it, reducing decision fatigue.7

**Library:** **react-tinder-card**.23

* This library provides the physics-based interaction out of the box.
* **Implementation:** The "Deck" component takes the inboxTasks array.
  + onSwipe('right'): Moves task to the Eisenhower "Inbox" for sorting.
  + onSwipe('left'): Sets a snoozeUntil timestamp for +24 hours.
  + onSwipe('up'): Deletes the task.
* **Insight:** This turns the "chore" of planning into a rapid-fire, almost rhythmic activity.

#### 4.2.2 Confetti and Visual Celebration

When a task is completed, the app must provide immediate sensory feedback.

Library: react-canvas-confetti.24

* It is more performant than DOM-based confetti because it uses the Canvas API, preventing jank (stuttering) which breaks immersion.
* **Trigger:** The onDragEnd event in the Eisenhower Matrix (when moving to "Done") or the checkbox in the Time Block should fire the confetti() instance.
* **Variable Reward:** To prevent habituation (boredom), the intensity of the confetti should vary. Small tasks get a small "pop"; big tasks get a "shower." This variable ratio schedule is the most effective reinforcement schedule in psychology.

#### 4.2.3 Visual Timers (The Pomodoro)

Digital clocks (24:59) induce anxiety because they emphasize the loss of time. Visual timers (a circle shrinking) emphasize the structure of time.

Library: react-countdown-circle-timer.25

* This SVG-based component allows for smooth, color-coded transitions (Green -> Yellow -> Red).
* **Feature:** "Focus Mode." When a user clicks a Time Block, this timer fills the screen. It is a "monotasking" interface that blocks out all other UI elements, aiding the CADDI goal of "mindfulness" and attention regulation.

### 4.3 Asking "Google Antigravity": Engagement Logic Prompts

**Prompt Strategy:**

"Act as a Game Designer and React Developer. I want to implement a 'Streak' feature for my ADHD app.

Logic:

1. The user gets a 'streak' point for every day they complete at least 3 tasks.
2. **Forgiveness:** If they miss one day, the streak does not reset immediately; it enters a 'frozen' state. If they miss two days, it resets. (This is crucial for ADHD rejection sensitivity).
3. UI: Generate a React component using framer-motion that displays a flame icon. If the streak is active, the flame pulses. If frozen, it turns blue and static.  
   Provide the useEffect logic to check the lastLoginDate and calculate the streak status."

## Part V: Implementation Intentions and The AI Therapist

### 5.1 Clinical Rationale: If-Then Planning

The CADDI protocol emphasizes "Behavioral Analysis"—understanding the chain of events that leads to procrastination.8 A powerful tool for this is **Implementation Intentions**: pre-deciding how to handle obstacles.

* *Standard Goal:* "I will write the report."
* *Implementation Intention:* "If I feel the urge to check Reddit, then I will stand up and stretch for 1 minute."

Research shows that forming these specific plans automates the response to the obstacle, requiring less willpower in the moment.3

### 5.2 Technical Implementation: The Obstacle Wizard

The Spark app should not just accept a task; it should act as a wizard.

* **UI Flow:** When a user creates a "Big" task, a modal appears: "What's the biggest obstacle to this?" (Options: Boredom, Anxiety, Distraction).
* **Dynamic Response:** Based on the selection, the app suggests an "If-Then" plan.

### 5.3 Leveraging "Google Antigravity": Dynamic Content Generation

Hard-coding these responses is limiting. AI can generate infinite, context-aware strategies.

**Prompt Strategy:**

"Act as a CBT Therapist.

Input: User task: 'Do my taxes'. User Obstacle: 'I feel overwhelmed by the paperwork.'

Task: Generate 3 distinct 'Micro-Strategies' using the 'Implementation Intention' format (If X, then Y).

Tone: Encouraging, practical, non-judgmental.

Format: JSON array.

Example Output: 'If I feel overwhelmed, then I will stack the papers in one pile and only look at the top one.'"

This allows the app to feel "alive" and personalized, addressing the user's specific emotional barriers in real-time.

## Part VI: Neuro-Inclusive Design & Accessibility Architecture

### 6.1 Accessibility Standards for Cognitive Disabilities

To be "right and tight," the Spark app must be accessible not just physically, but cognitively. The W3C's Web Content Accessibility Guidelines (WCAG) 2.2 include specific provisions for cognitive accessibility.27

* **Touch Targets:** Mobile users with ADHD often have co-occurring dyspraxia (clumsiness). Touch targets must be at least **44x44 CSS pixels**.27
* **Error Prevention:** ADHD users are impulsive. Destructive actions (deleting a task) should not happen instantly. However, "Are you sure?" modals are annoying.
  + *Solution:* **Toast with Undo.** When an item is deleted, it disappears immediately (satisfaction), but a "Task Deleted (Undo)" toast remains for 5 seconds.
* **Sensory Overload:** The interface must avoid "shimmering" or auto-playing animations which can be distracting or trigger sensory processing issues.29

### 6.2 Styling with Tailwind CSS for Focus

Using Tailwind CSS allows for rapid implementation of a "Calm Tech" aesthetic.

* **Dark Mode:** Essential. Many ADHD users have photophobia or use apps late at night. Use Tailwind’s dark: variant for all components.
* **Color Coding:** Use color *functionally*, not decoratively.
  + Red = Urgent (Do First).
  + Blue = Structure (Schedule).
  + Green = Delegation.
  + Gray = Low priority.
* **Consistency:** The prompt to the AI should enforce a "Design System."

**Prompt Strategy for Styling:**

"Generate the React code using Tailwind CSS.

Style Guide:

* **Font:** Inter or System Sans-Serif (High readability).
* **Colors:** Slate-900 for text, Slate-50 for backgrounds (Soft contrast).
* **Spacing:** Use relaxed spacing (p-4, gap-4) to reduce visual clutter (White space is cognitive rest).
* **Dark Mode:** Ensure fully supported dark:bg-slate-900 alternatives.
* **Accessibility:** Ensure all text meets WCAG AA contrast ratios."

## Part VII: Engineering with "Google Antigravity" (Generative AI)

The user's request to "ask Google antigravity" implies a desire to use advanced AI tools to build these features. This section synthesizes the prompt engineering best practices found in the research 30 into a cohesive workflow for the Spark app developer.

### 7.1 The "Persona-Context-Constraint" Prompting Framework

To get high-quality code from an LLM, prompts must follow a strict structure:

1. **Persona:** Who is the AI? (e.g., "Senior React Developer", "CBT Therapist").
2. **Context:** What is the app? (ADHD productivity tool, CADDI protocol).
3. **Task:** Specific component or logic needed.
4. **Constraints:** Libraries to use, accessibility standards, style guide.

### 7.2 Integrated Workflow Prompts

**Phase 1: Database Schema Design**

"Act as a Database Architect. I am building a React/Firebase app for ADHD. I need a schema to support:

1. Tasks (with Eisenhower Quadrant, Time Block start/end, Completion Status).
2. Streaks (Last login, Current streak count, Freeze status).
3. User Settings (Dark mode pref, Work hours).  
   Output the JSON structure for the Firestore documents,
4. ensuring efficient querying for 'Today's Tasks'."

**Phase 2: Component Logic (The Hook)**

"Act as a React Expert. Write a custom hook useEisenhowerMatrix.

Input: Array of Tasks.

Logic: Filter tasks into 4 arrays based on the quadrant property.

Functions: moveTask(taskId, targetQuadrant), checkLimit(quadrantId) (return false if 'doFirst' has > 5 items).

Tech: TypeScript, generic state management."

**Phase 3: The "Glue" (AI Content)**

"Act as a Backend Developer. Write a Cloud Function (Node.js) that acts as a proxy to the OpenAI API.

Endpoint: /generate-breakdown.

Input: A large task name (e.g., 'Clean the house').

Prompt to LLM: 'Break this task down into 5 micro-steps that take less than 10 minutes each. Return JSON.'

Output: The JSON array from the LLM.

Ensure error handling if the API times out."

### 7.3 Debugging with AI

When the "tight" app has bugs, the AI can be used for "Rubber Duck Debugging."

"I am using react-beautiful-dnd and my items are snapping back to the original position instead of dropping. Here is my onDragEnd function: [Code]. Analyze why the state isn't updating correctly and provide the fix."

## Conclusion

The transformation of the Spark app into a clinically valid, high-engagement digital therapeutic requires a deliberate fusion of CADDI principles with advanced React engineering. By implementing the **Eisenhower Matrix** to externalize prioritization, **Time Blocking** to visualize time, and **Gamification/Swipe UI** to bridge the dopamine gap, the app directly addresses the neurological deficits of ADHD-I.

Furthermore, the strategic use of "Google Antigravity" (Generative AI) allows for the rapid development of these complex features and the creation of a "smart" content layer that adapts to the user's psychological state. This report provides the architectural blueprint—from the choice of dnd-kit and react-big-calendar to the specific prompt templates—necessary to execute this vision. The result will be an application that is not only "tight" in its code quality but profound in its capacity to scaffold the executive functions of its users.

## Table of Tables

| **Table ID** | **Title** | **Section** |
| --- | --- | --- |
| 1 | CADDI Module to Digital Feature Mapping | 1.1 |
| 2 | React Drag-and-Drop Library Comparison | 2.2.1 |
| 3 | Calendar Library Feature Analysis | 3.2.1 |

*(End of Report)*

#### Works cited

1. (PDF) Cognitive behavioral therapy for ADHD predominantly inattentive presentation: randomized controlled trial of two psychological treatments - ResearchGate, accessed December 16, 2025, <https://www.researchgate.net/publication/390761050_Cognitive_behavioral_therapy_for_ADHD_predominantly_inattentive_presentation_randomized_controlled_trial_of_two_psychological_treatments>
2. "Digital Health Intervention for Children with ADHD to Improve Mental H" by Nancy Herrera, Franceli L. Cibrian et al., accessed December 16, 2025, <https://digitalcommons.chapman.edu/engineering_articles/213/>
3. Group cognitive behavioral therapy for ADHD inattentive presentation : feasibility, patients' perspectives and effectiveness - Karolinska Institutet - Figshare, accessed December 16, 2025, <https://openarchive.ki.se/articles/thesis/Group_cognitive_behavioral_therapy_for_ADHD_inattentive_presentation_feasibility_patients_perspectives_and_effectiveness/27890013>
4. Eisenhower Matrix for ADHD: Prioritize with less overwhelm - Tiimo App, accessed December 16, 2025, <https://www.tiimoapp.com/resource-hub/how-to-prioritize-tasks-eisenhower-matrix>
5. Time Blocking For ADHD: Manage Your Day With Less Stress | Life Skills Advocate, accessed December 16, 2025, <https://lifeskillsadvocate.com/blog/time-blocking-for-adhd/>
6. Gamification in React Native - Commt, accessed December 16, 2025, <https://commt.co/blog/gamification-in-react-native>
7. 3DJakob/react-tinder-card-demo - GitHub, accessed December 16, 2025, <https://github.com/3DJakob/react-tinder-card-demo>
8. “It was very nice to be in a room where everyone had ADD—that's kind of VIP”: Exploring clients' perceptions of group CBT for ADHD inattentive presentation - PMC - PubMed Central, accessed December 16, 2025, <https://pmc.ncbi.nlm.nih.gov/articles/PMC11192351/>
9. Design Principles - React, accessed December 16, 2025, <https://legacy.reactjs.org/docs/design-principles.html>
10. The Best ADHD App for Adults - Priority Matrix, accessed December 16, 2025, <https://appfluence.com/best-adhd-app-for-adults/>
11. Building a Drag-and-Drop Kanban Board with React and dnd-kit - Reddit, accessed December 16, 2025, <https://www.reddit.com/r/react/comments/1g8h39z/building_a_draganddrop_kanban_board_with_react/>
12. erictherobot/eisenhower-matrix - GitHub, accessed December 16, 2025, <https://github.com/erictherobot/eisenhower-matrix>
13. atlassian/react-beautiful-dnd: Beautiful and accessible drag and drop for lists with React, accessed December 16, 2025, <https://github.com/atlassian/react-beautiful-dnd>
14. Nested Drag and Drop with react-beautiful-dnd - Stack Overflow, accessed December 16, 2025, <https://stackoverflow.com/questions/52814011/nested-drag-and-drop-with-react-beautiful-dnd>
15. Top React Libraries You Should Know About in 2024 — 1 | by Alen Sabu | Medium, accessed December 16, 2025, <https://medium.com/@alensabu12xtz/top-react-libraries-you-should-know-about-in-2024-1-3aa7b892eadc>
16. ADHD app recommendations with these features: what works for y'all? - Reddit, accessed December 16, 2025, <https://www.reddit.com/r/productivity/comments/1d7cigo/adhd_app_recommendations_with_these_features_what/>
17. Time Blocking Planner Apps: We tested 20 to find the Best 8 | Saner.AI, accessed December 16, 2025, <https://www.saner.ai/blogs/best-time-blocking-planner-apps>
18. React FullCalendar vs Big Calendar - Bryntum, accessed December 16, 2025, <https://bryntum.com/blog/react-fullcalendar-vs-big-calendar/>
19. Top React calendar components - Choosing the best library for your UI - DronaHQ, accessed December 16, 2025, <https://www.dronahq.com/top-react-calendar-components/>
20. Best React scheduler component libraries - LogRocket Blog, accessed December 16, 2025, <https://blog.logrocket.com/best-react-scheduler-component-libraries/>
21. Planby - React Scheduler Component | Schedule, Timeline & Calendar for React, accessed December 16, 2025, <https://planby.app/>
22. I built a calendar app that displays time differently for people with ADHD - Reddit, accessed December 16, 2025, <https://www.reddit.com/r/SideProject/comments/1krva6u/i_built_a_calendar_app_that_displays_time/>
23. react-tinder-card - NPM, accessed December 16, 2025, <https://www.npmjs.com/package/react-tinder-card>
24. How to make Confetti with React in 5 minutes - Starter Notion blog for Nextjs with Tailwind, accessed December 16, 2025, <https://nextjs-notion-blog-starter.vercel.app/blog/how-to-make-confetti-with-react-in-5-minutes>
25. react-countdown-circle-timer examples - CodeSandbox, accessed December 16, 2025, <https://codesandbox.io/examples/package/react-countdown-circle-timer>
26. show only minutes in react using react-countdown-circle-timer - Stack Overflow, accessed December 16, 2025, <https://stackoverflow.com/questions/65145188/show-only-minutes-in-react-using-react-countdown-circle-timer>
27. Mobile Accessibility | Guidelines and Standards for Mobile - Recite Me, accessed December 16, 2025, <https://reciteme.com/us/news/mobile-accessibility/>
28. Mobile Accessibility at W3C | Web Accessibility Initiative (WAI), accessed December 16, 2025, <https://www.w3.org/WAI/standards-guidelines/mobile/>
29. Inclusive UX/UI for Neurodivergent Users: Best Practices and Challenges - Medium, accessed December 16, 2025, <https://medium.com/design-bootcamp/inclusive-ux-ui-for-neurodivergent-users-best-practices-and-challenges-488677ed2c6e>
30. Prompt engineering | OpenAI API, accessed December 16, 2025, <https://platform.openai.com/docs/guides/prompt-engineering>
31. 11 Prompting Tips for Building UIs That Don't Suck - Builder.io, accessed December 16, 2025, <https://www.builder.io/blog/prompting-tips>