This is the revised project proposal report, updated to reflect the simplified scope. We have removed **OAuth (Google Sign-In)** and replaced the complex **PDF report generation** with a simpler **In-Dashboard Comparison Chart**.

The timeline has also been condensed to reflect the ambitious **four-week goal**.

# 📄 Project Proposal Report:

# The IdleInsights (Revised Scope)

**Date:** December 14, 2025 1

**File Name (Example):** Group 19 [125, 138, 153].pdf (Adjust with your group number and IDs) 2

### 1. Title of the Project

**The Procrastination Analyst:** A Data-Driven MERN Application for Root Cause Pattern Identification and Countermeasure Suggestion3.

### 2. Name of Supervisor

[Insert Instructor/Lab Supervisor Name] 4

### 3. List of Team Members (With Detailed Technical Roles)

1. **[Name]** ([ID]) – *Focus: Backend Architecture, Mongoose Middleware & Asynchronous Logic.* Responsible for defining schemas, implementing the PVI algorithm, and managing complex aggregation queries5.
2. **[Name]** ([ID]) – *Focus: Security Implementation,* ***Authentication (JWT)****, and File Handling.* Responsible for user models, token generation, API route protection, Multer integration, and serving benchmarking data6.
3. **[Name]** ([ID]) – *Focus: Frontend Component Design, Routing, State Management & Conditional Rendering.* Responsible for custom UI/UX, React component architecture, data visualization, and implementing the logic for the Countermeasure Modes7.

### 4. Project Overview

The Procrastination Analyst is a highly specialized web application built entirely on the **MERN (MongoDB, Express, React, Node.js) stack**8. It acts as a detailed self-diagnostic tool by moving beyond traditional time-tracking9. The system captures both a user's **Intended Tasks** (with a subjective **Difficulty Score**) and their subsequent **Confessional Log** (detailing the distraction activity)10. This comparative data feeds an **asynchronous algorithm** that calculates the user's **Procrastination Velocity Index (PVI)** and identifies their specific behavioral **Root Cause Pattern**, which is then used to provide interactive, data-driven countermeasures11.

### 5. Motivation Behind the Project

The primary motivation is to showcase a strong mastery of all technical requirements outlined in the course guidelines while adhering to the creative theme of **"Uselessly Useful"**12.

* **Technical Excellence:** The project integrates every required component: multiple schemas (User, Task, Log), full CRUD operations, **JWT**, advanced Mongoose middleware, asynchronous processing for analysis, and server-side file handling13.
* **Creative Relevance:** By turning the boring act of distraction into a hyper-analyzed academic study, the project fulfills the core creative brief14.
* **Design Focus:** The commitment to building all frontend components, charts, and dashboards from scratch ensures strong skills in clean HTML, CSS, and modern React component design15.

### 6. Key Features of the Project (Detailed Use-Case Descriptions)

The application’s functionality is divided into four robust, interconnected systems16:

#### A. Core Infrastructure and Data Capture

|  |  |  |
| --- | --- | --- |
| **Feature** | **Use-Case Description** | **Technical Requirement Highlight** |
| **Secure User Authentication** | Users can register and log in via email/password secured by **JWT**. All API access requires a valid token via Express middleware17. | **Token-based Authentication** |
| **Intent Manager (CRUD)** | Users create documents detailing their work intentions, including the task title and their personal **Difficulty Score (1-5)**. This score is a mandatory input for the PVI algorithm18. | **Multiple Schemas** (IntendedTask) & **Basic CRUD** |
| **Confessional Log (CRUD)** | The primary input where users log a distraction session, linking to the **Avoided Task**. Captures Duration, Actual Activity (e.g., 'Watching Videos'), and Activity Detail (e.g., 'YouTube/Cooking')19. | **Multiple Schemas** (LogEntry) & **Basic CRUD** |

#### B. Advanced Backend Logic & Analysis Engine

|  |  |  |
| --- | --- | --- |
| **Feature** | **Use-Case Description** | **Technical Requirement Highlight** |
| **PVI Algorithm** | Upon saving a new LogEntry, a Mongoose **post-save hook (middleware)** triggers an **asynchronous function** that calculates the log's impact. The overall PVI (Procrastination Velocity Index) is updated based on total duration and difficulty of avoided tasks21. | **Asynchronous Programming & Middleware Usage** |
| **Root Cause Pattern ID** | The asynchronous analysis function analyzes the user's logs for recurring patterns (e.g., 'Social Media' before 'Difficult' tasks). It then assigns a behavioral label (*e.g., Overwhelm Avoidance*) to the user's profile22. | **Middleware Usage** |
| **In-Dashboard Benchmarking Chart** | An endpoint that returns aggregated, anonymized user data, enabling the frontend to display a direct bar chart comparison of the user’s PVI versus the overall system average PVI.  As the file handling part, the user can download this chart as image/pdf whichever we decide | **Middleware** & **Frontend Data Support** |

#### C. The Analyst Dashboard & Intervention System

|  |  |  |
| --- | --- | --- |
| **Feature** | **Use-Case Description** | **Technical Requirement Highlight** |
| **PVI Velocity Gauge** | The central dashboard visualization that dynamically changes color (Green/Yellow/Red) and accompanying text based on the user’s current PVI score23. | **Conditional Rendering & Custom Design** |
| **Activity/Time Charts** | Simple visualization components displaying the user's most time-wasting platform and their most distracted hours of the day24. | **Frontend Data Visualization** |
| **Countermeasure Generator (Modes)** | The core interactive solution. The system uses **Conditional Rendering** to recommend a specific "Mode" based on the identified **Root Cause Pattern** (e.g., suggests "Task Deconstructor Mode" for *Overwhelm Avoidance*)25. | **Conditional Rendering** |
| **Interactive Modes** | Dedicated views accessed via **React Router** where users can use tailored micro-tools: e.g., a simple Pomodoro timer (using **React Hooks**) in the "Frequent Break Mode," or a sub-task lister in the "Task Deconstructor Mode"26. | **Proper Routing & Effective Use of Hooks** |

### 7. Tools and Technologies (Updated)

* **Frontend:** React.js (Hooks, Components), React Router, Custom CSS (or modern framework like Tailwind/Bootstrap used exclusively for styling, not templates), Axios for API interaction27.
* **Backend:** Node.js (Runtime) 28, Express.js (Framework)29.
* **Database:** MongoDB Atlas (Cloud Database) 30, Mongoose ODM (Object Data Modeling)31.
* **Security:** JSON Web Tokens (JWT), Bcrypt.js (Password Hashing).
* **Other Tools:** Git/GitHub (Version Control), Multer (File Handling)32.

### 8. Proposed Timeline

|  |  |  |
| --- | --- | --- |
| **Week** | **Phase** | **Key Milestones and Deliverables** |
| **Week 1** | **Foundation & Security Setup** | Define all Mongoose Schemas (*User, IntendedTask, LogEntry*). Set up Express server and database connection. **Implement secure JWT authentication** (Register/Login routes) and JWT Middleware for route protection. |
| **Week 2** | **Core Data Logic & CRUD** | Complete all **CRUD** operations for IntendedTask and LogEntry. Implement Multer for **Artifact File Handling** (Upload feature). Begin data aggregation methods required for the PVI. |
| **Week 3** | **Advanced Backend Logic** | Implement the **PVI Algorithm (Asynchronous function)**. Integrate **Mongoose Middleware** hooks to trigger analysis on log entry creation. Finalize **Root Cause Pattern ID** logic. |
| **Week 4** | **Frontend Core & Integration** | Set up React component structure and React Router. Build the login/registration views, Intent Manager, and Confessional Log forms. Establish secure API connections across the application. |
| **Week 5** | **Dashboard & Intervention** | Build the Analysis Dashboard. Implement **Conditional Rendering** for the PVI Gauge and Mode suggestions. Create the **Activity/Time Charts** and the **In-Dashboard Comparison Chart**. |
| **Week 6** | **Polish, Testing & Final Deployment** | Develop the **Interactive Modes** components (e.g., timers, listers). Perform comprehensive system-wide testing (unit and integration tests). Finalize custom CSS/styling, documentation, and prepare the project for deployment and presentation. |