FocusQuest Implementation Document

FocusQuest is a productivity-focused web application built with modern web technologies, aiming to gamify focus sessions and offer AI-driven insights for improved performance and engagement.

Frontend

Technology Stack

- **Framework**: React, powered by **Vite** for fast build and development processes.
- **Styling**: TailwindCSS, providing utility-first CSS styling for consistent and efficient UI design.

Structure and Components

- **UI Components**:
  - Located in `client/src/components/ui/`
  - Example Components:
    - `ChartStyle`: Handles chart rendering with dynamic styling based on input props.
    - `ChartTooltip`: Provides tooltips for charts, dynamically styled.
- **Pages**:
  - Located in `client/src/pages/`
  - Example:
    - `auth-page.tsx`: Manages user authentication and profile interaction.

Backend

Technology Stack

- **Runtime**: Node.js
- **Framework**: Express.js for handling API requests and routing.

API Routing

- Routes defined in `server/routes.ts`
- Major Endpoints:
  - Integration with AI services (Gemini API).
  - User session management and data retrieval.

Middleware

- **Authentication**:
  - `isAuthenticated` middleware protects sensitive API endpoints, ensuring only authorized users can access them.

Database

- **Database**: Neon PostgreSQL
- **ORM**: Drizzle ORM for structured and type-safe database interactions.
- **Configuration**: Found in `server/db.ts`.

Time Frame

Figma Prototype: 7-8 hrs
Setting up Replit environment along with Database config: 3-4 hrs
Converting Figma prototype to code: 3-4 hrs
Refactoring code to fit requirements: 10-15 hrs
Testing functionality (Black box): 4-5 hrs

Total Components:

Routes: ~21
React Components: ~50
Database Functions: ~20
Utility Functions: ~25
Pages: 8
Database Tables: 3

User Inputs (10):
Login form
Registration form
Task creation form
Timer duration input
Task search/filter
Task category selection
Focus session task selection
AI message input
Task completion toggle
Profile settings
User Outputs (5):
Task lists/details display
Focus timer display
Stats dashboard
AI assistant responses
User profile information
User Queries (8):
Get tasks
Get upcoming tasks
Get today's tasks
Get task by ID
Get focus sessions by task
Get focus sessions by user
Get user stats
Get AI response
Data Files (30):
Schema definitions (users, tasks, focusSessions)

UI components (27 components in client/src/components/ui)
Database configuration
Authentication handlers
Relational Tables: 3 Key tables

- `users`
- `tasks`
- `focus_sessions`

External Interfaces (4):
Database (Neon PostgreSQL)
AI Service (OpenAI/Gemini)
Authentication System
File System (for static assets)

Functionality Details

Functionality	Details	Simple	Average	Complex	Complexity
User Inputs	- Variable input, dynamic based on UI fields (`ai-page.tsx`). - Worst case: 10 - Supports multiple messages and interactive elements.	3	5	6	50
User Outputs	- Dynamic output generated by AI in response to inputs. - Worst case: 5 - Messages are appended sequentially to the UI.	4	5	7	20
User Queries	- Worst case: 8 primary queries handled: - Additional endpoints for user data and focus sessions.	6	8	9	64
Data Files	Worst case: 30 files	6	7	10	300

Relational Tables	- 3 Key tables: • `users` • `tasks` • `focus_sessions`	3	6	8	24
External Interfaces	- 4	3	6	10	40

GFP: 498

PCA Computation

PC = 0(no influence) 1(incidental) 2(Moderate) 3(Average) 4(Significant) 5 (essential)

(1) Does the system require reliable backup and recovery? 1

(2) Are data communications required? 5

(3) Are there distributed processing functions? 2

(4) Is performance critical? 2

(5) Will the system run in an existing, heavily utilized operational environment? 3

(6) Does the system require online data entry? 5

(7) Does the online data entry require the input transaction to be built over multiple screens or operations? 4

(8) Are the master files updated online? 5

(9) Are the inputs, outputs, files, or inquiries complex? 4

(10) Is the internal processing complex? 4

(11) Is the code designed to be reusable? 3

(12) Are conversion and installation included in the design?3

(13) Is the system designed for multiple installations in different organizations? 0

(14) Is the application designed to facilitate change and ease of use by the user? 5

Processing Complexity Adjustment (PCA): PCA = 0.65 + 0.01 * PC

PC = 46.

PCA = 0.65 + 0.01 *46 = 1.11

FP = GCP * PCA

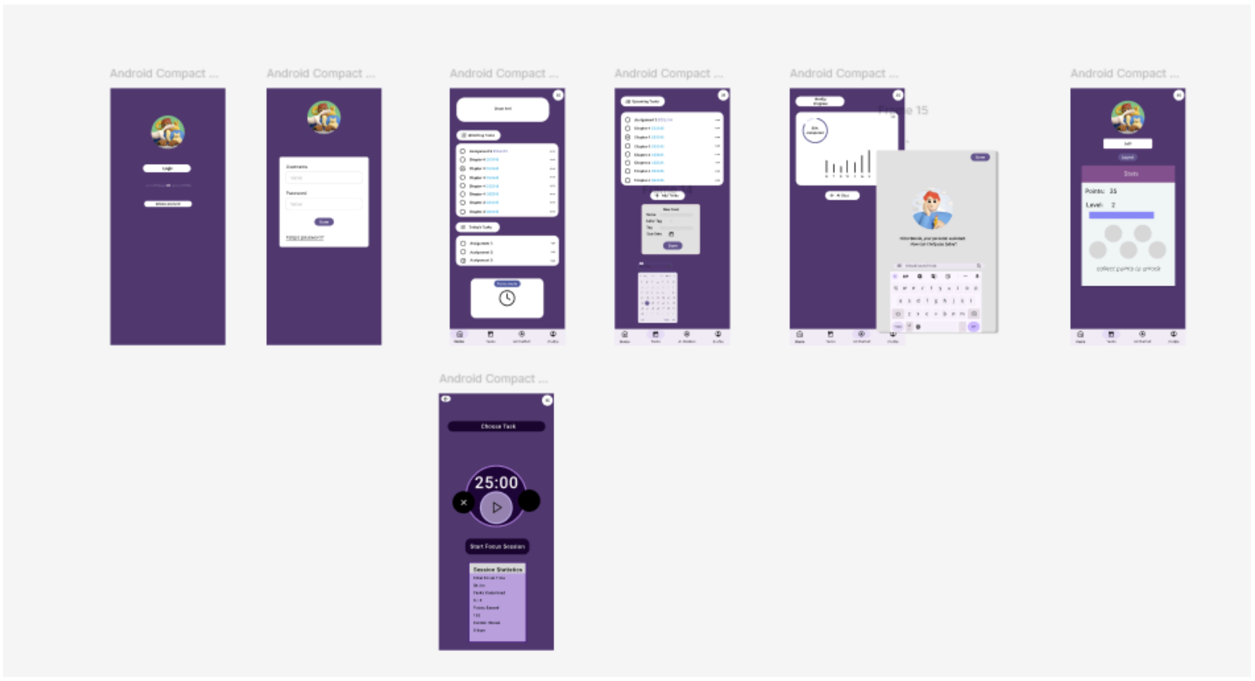
498*1.11 = 552.78

Productivity = FP/ (no of people* no of weeks) = 552.78/ 10 *9 = 6.14 FP/person-week

E = FP/productivity = 552.78/6.14 = 90.03

In conclusion, since our team size = 9, then project duration is: D = E / team size = 90.03/9 = 10.003 (round up to 11)

Prototype



Summary

After building a Figma prototype, I used Replit to convert it to code and refactored it where needed to save me time provided the time constraints.