FocusForge
Manvitha P. K. and Hrehaan M.

PROJECT PROBLEM STATEMENT

Many high school students struggle with maintaining focus due to distractions and poor time management habits. Existing productivity and Pomodoro timer apps are often cluttered with unnecessary features or not tailored to student-specific needs. This results in low adoption and limited effectiveness. The opportunity is to create **FocusForge**, a simple, lightweight, offline Pomodoro timer app designed specifically for students, helping them structure their study sessions effectively through customizable timed focus periods and breaks.

CONCISE DESIGN OVERVIEW

FocusForge is designed as a minimalistic Pomodoro timer web application with three main components:

- **Timer Engine:** Manages session timing, phase transitions (focus, short break, long break), and session counting. It runs the countdown logic, handles pause/resume/skip, and plays notification sounds.
- **User Interface:** Presents a clean menu system, timer display, session counters, settings panel, and logs for user interaction and feedback. The UI includes buttons, keyboard shortcuts, and modals for confirmations.
- **Settings & Persistence:** Allows users to customize session lengths and number of sessions before a long break. Settings and daily session counts are saved in browser storage to persist across sessions without requiring internet access.

SCOPE STATEMENT

INCLUDES:

- Develop a functional Pomodoro timer app focused on student use
- Implement adjustable session and break durations
- Include session counter reset and exit options
- Use client-side local storage for persistence (offline capability)
- Design a simple, intuitive UI with logs and modals
- Enable keyboard shortcuts for accessibility
- Test on major browsers

EXCLUDES:

- User accounts or cloud synchronization
- Complex task or project management features
- Social or collaborative functions

- Mobile app native development (this is a web app)
- Automated notifications outside the browser (e.g., push notifications)

SMART Goals:

- The initial goal is to **build the basic Pomodoro timer functionality** by **May 22**. This includes implementing a countdown for focus sessions and breaks using standard Java features.
- By **May 30**, the project should include additional features such as a **session counter** that tracks completed Pomodoros, along with a **menu system** that allows users to start a session, reset the counter, or exit the program.
- The goal for **June 10** is to **fully test and debug** the application. This involves testing all menu options, timer transitions, and handling edge cases like invalid inputs.
- Finally, the completed and tested application will be **presented and demonstrated live** by either **June 17 (for 11th graders)** or **June 3 (for 12th graders)**, depending on the class schedule.

TIMELINE

Date	Task	Milestone
Day 1	Research Pomodoro techniques and student needs	Problem statement & design
Day 2	Define project scope and set goals	Scope finalized
Day 3-4	Develop core Pomodoro timer logic (JS class)	Timer logic prototype
Day 5	Build UI layout (HTML/CSS) and basic menu system	UI prototype
Day 6	Integrate timer logic with UI controls	Interactive timer
Day 7	Add settings panel and persistence (localStorage)	Settings functionality
Day 8	Implement session counting and logs	Session tracking ready
Day 9	Add modals and keyboard shortcuts	Accessibility improvements
Day 10	Testing (unit and manual) and bug fixes	Test plan execution

Day 11	Final UI polishing and responsiveness improvements	Final prototype ready
Day 12	Documentation and project submission	Project delivery

TEST PLAN

Whole Solution Testing:

- Manual testing of the entire app workflow:
    - Start/pause/resume/skip/reset sessions
    - Changing settings and verifying they persist
    - Session counter increments correctly across multiple session
    - Modal dialogs prompt correctly and prevent accidental actions
    - Keyboard shortcuts trigger appropriate functions
    - UI adapts correctly to different screen sizes

Component Testing:

- **Timer Logic:**
    - Verify countdown accuracy over various durations
    - Test phase transitions after countdown completion
    - Confirm pause/resume behavior maintains correct timing
- **Settings:**
    - Validate input restrictions (e.g., min/max values)
    - Confirm settings save and load correctly from localStorage
- **Session Counter:**
    - Confirm counts increment only after focus sessions complete
    - Verify counter resets correctly and daily reset works
- **UI Components:**
    - Test buttons and modals open/close as expected
    - Keyboard shortcut responses
    - Log messages reflect correct user actions and timer events

Testing Type:

- Testing will be **manual**, using positive test cases (valid inputs, expected behavior) and negative test cases (invalid inputs, cancellation of modal prompts).
- Automated unit testing is not included but could be added in future iterations.

RISKS AND CONTINGENCY PLANS

Risk	Probability	Impact	Mitigation Plan
Browser compatibility issues	Medium	Medium	Test early on multiple browsers; use standard APIs.
LocalStorage not supported or cleared	Low	High	Warn user about data loss; allow export/import settings.
User confusion with UI or settings	Medium	Medium	Provide clear labels, tooltips, and logs for feedback.
Timer inaccuracy due to tab inactivity or sleep	Medium	Medium	Use Date timestamps instead of intervals if needed.
Feature creep leading to complexity	Medium	High	Stick strictly to scope; defer extra features to future.
Lost data if user clears browser cache	Low	Medium	Remind users to save settings externally if needed.
Time constraints during development	Medium	High	Prioritize core features; test early to reduce bugs.