Postmortem Analysis

Wellness App Development

1. Overall Architecture

The system was designed as a React-based single-page application (SPA) with a modular component structure. The architecture followed a component-based design with reusable UI elements and a centralized state management approach using React's useState and useContext. The backend was integrated via Appwrite for authentication and database services. The system was divided into the following major components:

Frontend: React components for pages like Dashboard, Resources, Survey, Blog, and User Profile.

Backend: Appwrite for user authentication, database management, and API integration.

Styling: CSS modules for component-specific styles.

Shared components were placed in a common components directory to prevent unnecessary reuse of code.

4. What Went Wrong

Merge Conflicts: Frequent Git merge conflicts due to simultaneous work on the same files. State Management: Lack of a centralized state management library (e.g., Redux) caused some state inconsistencies. Time Management: Sometimes the occasion arose where the time required for testing and debugging was underestimated. CSS Issues: Styling inconsistencies across components due to lack of a global design system.

6. Time Allocation

Most Time-Consuming: Debugging and resolving merge conflicts.

11. Project Changes from Initial Vision Initial Vision

Focused primarily on wellness tracking and user engagement.

Final Outcome: Expanded to include features like Relaxation games and "Message in a Bottle," which were not part of the initial plan.

Unexpected Changes: Added more interactive features based on customer feedback during development.

12. Lessons Learned Team Development

Start Doing: Use feature branches for isolated development to reduce merge conflicts. Keep Doing: Regular stand-ups and progress tracking. Stop Doing: Avoid

working directly on the main or master branch. Large Project Development: Importance of modular design for scalability. Testing should be integrated into the development process, not left for the end.

13. Areas for Improvement

Testing: Implement automated testing for both frontend and backend.

Documentation: Improve code documentation for easier onboarding of new developers.

Error Handling: Add robust error handling for API calls and user inputs.

Conclusion

The development of the Wellness App was a valuable learning experience. While there were challenges like merge conflicts, state management, along with testing and CSS consistency, the team successfully delivered a functional and engaging product. Moving forward, the team will adopt better state management practices, automate testing, and focus on stronger initial planning to streamline the development process. Overall, the project demonstrated the importance of collaboration, adaptability, and continuous learning in software development