Podczas projektowania pracy nad projektem zespołowym wykonaliśmy plik ze szczegółowym harmonogramem pracy. Całą dokumentacją techniczną a także podziałem zadań na poszczególnych członków zespołu. Niektóre funkcjonalności sytemu były za ambitne na nasze umiejętności. Jednak największym błędem nie było wyznaczenie jednej osoby odpowiedzialnej za spinanie wszystkich wykonanych zadań w jedną projektową całość. Nie wyznaczyliśmy także czasu oddania poszczególnych zadań co przysporzyło nam bardzo dużo problemów przy terminowej realizacji projektu. Brak wyznaczonego kierownika spowodował również wielki bałagan na repozytorium. Dużo czasu zmarnowaliśmy czekając sami na siebie, zadania wykonane na backendzie pozostały długo bez testów z powodu nie skończonych prac na frontendzie. Osoba odpowiedzialna za frontend zaczęła wykonywać zadania w kolejności nie sprzyjającej backendowi. Testowanie projektu zostało na ostatnie dni przed finalnym zakończeniem, większość czasu została zmarnowana przez poszczególne zespoły na czekanie aż drugi zespół wykona rzecz zdatną do podpięcia aby działała komunikacja.

Podczas realizacji projektu zespołowego w ramach projektu zespołowego, nasz zespół składający się z kilku osób przystąpił do tworzenia złożonego systemu informatycznego. Na początku, z dużym entuzjazmem, przygotowaliśmy szczegółowy harmonogram pracy, dokumentację techniczną oraz wstępny podział zadań na poszczególnych członków zespołu. Projekt został podzielony na dwie główne sekcje: backend (Java/Spring Boot) oraz frontend (Angular/TypeScript). W fazie planowania, ambitnie zaplanowaliśmy szereg funkcjonalności, z których część wyraźnie przekraczała nasze dotychczasowe umiejętności techniczne. Uznaliśmy jednak, że w trakcie realizacji nauczymy się niezbędnych technologii. Podział zadań został wykonany według deklarowanych umiejętności członków zespołu - osoby z większym doświadczeniem w Javie zajęły się backendem, a osoby znające TypeScript - frontendem. Kluczowym błędem okazał się brak wyznaczenia jednej osoby pełniącej rolę kierownika projektu lub koordynatora, który spinałby wszystkie wykonane zadania w spójną całość. Nie ustaliliśmy także sztywnych terminów (deadline'ów) dla poszczególnych zadań. Każdy członek zespołu pracował we własnym tempie, często bez wiedzy o postępach pozostałych osób.
W praktyce doprowadziło to do szeregu problemów. Repozytorium Git szybko stało się chaotyczne - brak ustalonej strategii branching i review powodował konflikty w kodzie, zduplikowane funkcjonalności oraz niespójny styl pisania kodu. Zespół backendowy, pracując sprawnie, zaimplementował większość endpointów API wraz z logiką biznesową. Jednak przez długi czas pozostawały one nietestowane w pełnym środowisku, ponieważ frontend nie był gotowy do komunikacji z backendem. Osoba odpowiedzialna za frontend, nie znając priorytetów backendowego zespołu, rozpoczęła implementację od wizualnie atrakcyjnych, ale mniej krytycznych komponentów interfejsu użytkownika. Tymczasem kluczowe elementy, takie jak moduł logowania, formularze do obsługi głównych funkcjonalności systemu czy mechanizmy

komunikacji z API, zostały odłożone na później. W rezultacie zespół backendowy spędził tygodnie czekając na możliwość przetestowania swojej pracy w rzeczywistym środowisku. Podobnie, zespół frontendowy napotkał własne problemy. Gdy wreszcie przystąpili do implementacji komunikacji z API, okazało się, że niektóre endpointy zostały zaprojektowane w sposób, który nie był optymalny z perspektywy interfejsu użytkownika. Brakowało dokumentacji API, a struktura przesyłanych danych nie zawsze odpowiadała potrzebom frontendu. Wymusiło to wielokrotne poprawki w backendzie. Testowanie integracyjne całego systemu zostało zaplanowane na ostatnie dni przed finalnym terminem oddania projektu. Gdy wreszcie udało się połączyć obie części, odkryliśmy szereg problemów z CORS, uwierzytelnianiem, formatami danych oraz logiką biznesową, która nie działała tak jak zakładano. Większość czasu w ostatnim tygodniu została poświęcona na gaszenie pożarów i naprawianie podstawowych błędów, zamiast na dopracowanie funkcjonalności i optymalizację. Ostatecznie projekt został oddany w terminie, z brakiem jednej funkcjonalności, a także jedną działającą prowizorycznie.

Wnioski:

1. **Brak struktury zarządzania projektem:** Projekt nie posiadał jasno określonego kierownika (project managera) odpowiedzialnego za koordynację pracy zespołu. W rezultacie nikt nie monitorował postępów, nie identyfikował blokad ani nie egzekwował terminów. W projektach zespołowych kluczowe jest wyznaczenie osoby odpowiedzialnej za nadzór i integrację prac.
2. **Niewłaściwe planowanie kamieni milowych:** Brak konkretnych terminów dla poszczególnych zadań i punktów kontrolnych uniemożliwił wczesne wykrycie opóźnień. Harmonogram powinien zawierać nie tylko listę zadań, ale także sztywne daty ich realizacji oraz mechanizmy weryfikacji postępów.
3. **Brak priorytetyzacji zadań:** Zadania powinny być realizowane w kolejności zapewniającej ciągłość pracy obu zespołów. Kluczowe jest wypracowanie przez zespoły minimum funkcjonalności (MVP - Minimum Viable Product) umożliwiającego wczesne testowanie integracji, zamiast równoległej pracy nad wszystkimi funkcjonalnościami.
4. **Niewystarczająca komunikacja między zespołami:** Backend i frontend pracowały w izolacji, bez regularnych spotkań synchronizacyjnych. Należało ustanowić codzienne lub cotygodniowe stand-upy oraz wspólnie opracować kontrakt API przed rozpoczęciem implementacji.
5. **Odłożenie testowania na koniec projektu:** Testowanie integracyjne powinno rozpocząć się znacznie wcześniej, nawet przy niekompletnej funkcjonalności.

Wczesne testy ujawniają problemy architektoniczne i komunikacyjne, gdy ich naprawa jest jeszcze stosunkowo prosta.

6. **Niedoszacowanie złożoności zadań:** Ambitne funkcjonalności przekraczające umiejętności zespołu powinny być albo uproszczone, albo zaplanowane z większym buforem czasowym na naukę. Realistyczna ocena kompetencji zespołu jest kluczowa w fazie planowania.
7. **Brak standardów i procedur zespołowych:** Nie ustalono zasad pracy z repozytorium (git flow, code review, strategie merge), standardów kodowania czy dokumentacji. Te elementy powinny być zdefiniowane i uzgodnione przed rozpoczęciem implementacji.
8. **Nieefektywne wykorzystanie czasu zespołu:** Długie okresy bezczynności spowodowane czekaniem na drugą stronę świadczą o błędach w planowaniu zależności. Należy identyfikować blokady z wyprzedzeniem i organizować pracę tak, aby każdy członek zespołu miał dostęp do zadań niezależnych w przypadku przestojów.