T086 Technical Challenges

Introduction

A variety of technical challenges have been encountered over the course of our BeeAware project. Specifically, the overarching problem we faced was framework compatibility issues with our client's hosting service. This document aims to identify each technical problem that arose in Phase 2 (Semester 2, IFB399) and how our team solved each of these problems in chronological order.

Technical Challenge 1

Framework Incompatibility (React & .NET Core with CrazyDomains through Plesk)

Our client initially wanted us to work with React as a frontend framework, and .NET Core as a backend framework. However, when trying to host our codebase on our client's hosting service (CrazyDomains through Plesk) our team and two other development teams (T038, T101) faced a problem with framework compatibility. Our backend team member, Ben, contacted CrazyDomains support and found out that although Plesk states that it can support the .NET Core backend framework, CrazyDomains cannot.

Technical Challenge 1 Solution

After discussing with CrazyDomains support and our client about possible solutions, we decided to change our backend framework to .NET Framework 4.8 as this was compatible with CrazyDomains. As an implication, our React frontend framework also needed to change to RazorViews as the backend was no longer compatible.

Technical Challenge 2

Project Teams Revert Codebase (RazorViews & .NET Framework 4.8)

Although initially all project teams were on-board for changing frameworks – one team (T038) changed their deliverable with our client and reverted back to the old codebase (React & .NET Core). Their newly agreed deliverable was to have everything locally hosted instead of changing frameworks to deploy the codebase on CrazyDomains. Upon this change our Client told us to do the same as all project teams should be working cohesively.

Technical Challenge 1 Solution

To our team, we believed that this solution did need meet our client's expectations. Ben researched an alternative hosting service, DigitalOceans, and informed our client that we could use Docker to host our codebase (React & .NET Core). This solution would provide our client with a relatively cheap option that has more flexibility in terms of scalability, frameworks, reliability, availability, and enabling remote access to the application.