Workboard Project Proposal

Date: 03/30/2023

Version 1

Table of Contents

[Purpose of this document 3](#_Toc134537439)

[Why Build This Project? 3](#_Toc134537440)

[Summary of solution 3](#_Toc134537441)

[Goals of solution 4](#_Toc134537442)

[Scope of solution 4](#_Toc134537443)

[Sitemap at completion of project: 4](#_Toc134537444)

[Requirements (written as user stories) 5](#_Toc134537445)

[Normal User Stories 5](#_Toc134537446)

[Admin User Stories 5](#_Toc134537447)

[Methodology for project 5](#_Toc134537448)

[Projected Project timeline and Deliverables 5](#_Toc134537449)

[Sprint 1 5](#_Toc134537450)

[Sprint 2 6](#_Toc134537451)

[Sprint 3 6](#_Toc134537452)

[Sprint 4 6](#_Toc134537453)

[Required Resources 6](#_Toc134537454)

[Tech Stack 6](#_Toc134537455)

[Libraries 7](#_Toc134537456)

[Cloud Services 7](#_Toc134537457)

[Deployment strategy 7](#_Toc134537458)

[Testing strategy 7](#_Toc134537459)

# Purpose of this document

This document outlines everything about Workboard, a new project I am building for my portfolio.

# Why Build This Project?

This project will be an incredibly valuable portfolio project as it will show that I can build a complex software system from start to finish. It will also help me learn about features that I will likely use in my day-to-day when building complex software systems. In particular, the requirements I have specified will require me to build a real-time API, which will likely use WebSockets which I have not used before.

I do not expect to complete this project before I start applying to jobs. On the contrary, I expect to be working on this project throughout my internship. But when it is complete, it will be a thing of beauty.

# Problem Statement

Teams need to be able to manage projects across a portfolio or program, manage roles within each project, manage tickets within each project, and track KPIs. They need one consolidated place to keep track of all work. This is where this solution comes into the picture.

# Summary of solution

WorkBoard is an easy, simple way to manage your projects using Scrum & Kanban methodologies. It is very similar to Trello, but not a direct clone. The primary unit in WorkBoard is the **board**, which is a place for all the work for a particular project to go. You can create cards for tasks on the board, and these cards will be organized into user-defined columns. You can assign team members to help you manage these tasks, and you can upload files such as project briefs, SRS documents, and much more, to help describe the board as a whole. Finally there will be the ability to run different types of reports on the board to help project managers see where the project is at and how it is progressing.

By default, WorkBoard is methodology-agnostic, meaning it does not enforce a specific project management methodology such as Scrum. However, there will be different generic actions and reporting mechanisms that will aid in supporting a particular methodology (e.g. adding a burndown chart to a particular column).

# Goals of solution

* Provide a simple, easy, and intuitive way for teams to manage their work in one place
* Notify team members of important changes to the board via in-app notifications and email.
* Enable users to give plenty of details to a task so that any team member on the board can pick up the task and begin working on it.
* Help me gain valuable skills and add an amazing full-stack project to my portfolio

# Scope of solution

* Web application that runs everything

# Sitemap at completion of project:

Screens:

* Login/signup
* User dashboard (shows all boards, notifications, and gives access to user settings)
* User settings
* Notifications
* Board View (shows a board and all of its columns, basically like the board views on Kanbanflow, Asana, or Trello)
* Edit Board (a dedicated screen to upload files for a board, update the board description, add team members, etc)
* Reports (shows reports for a particular board and allows the user to export those reports to something like an Excel file or similar)

Modals:

* Create/edit card
* Create/edit column
* Back up/restore board

# Requirements (written as user stories)

See [this project brief/SRS](https://app.asana.com/0/1201767691363844/1201767657940206).

# Methodology for project

I chose a lightweight implementation of Scrum for this project because it will help me stay organized and develop the project incrementally.

# Projected Project timeline and Deliverables

Note: This project timeline is a very rough estimate and the actual timeline may vary greatly from this. This is just to get me started.

## Sprint 1

Start Date: 05/08/2023

End Date: 05/14/2023

Sprint Goal: Create initial board view screen with columns and cards (no authentication or authorization yet)

## Sprint 2

Start Date: 05/15/2023

End Date: 05/21/2023

Sprint Goal: Create rest of screens in the application (still no authentication or authorization yet).

## Sprint 3

Start Date: 05/22/2023

End Date: 05/28/2023

Sprint Goal: Create login flow and add basic authentication checks

## Sprint 4

Start Date: 05/29/2023

End Date: 06/04/2023

Sprint Goal: Add team member invites and notification features

## Sprint 5

Start Date: 06/05/2023

End Date: 06/11/2023

Sprint Goal: Deploy application (definitely subject to massive change depending on how the application development goes)

# Required Resources

## Tech Stack

* ASP .NET with SignalR for the realtime API
* React with Typescript for the front end, also using a SignalR library.
* SQL Server for the data store
* Possibly Docker for packaging the application depending on whether the bug for building for linux/amd64 is fixed.

## Libraries

* MSTest and FakeItEasy for unit testing
* Entity Framework Core for the ORM
* Serilog for logging
* FluentValidation for validating models

## Cloud Services

* Render for deploying the back end
* Netlify for deploying the front end

# Deployment strategy

If the linux/amd64 bug is fixed, the backend will be packaged with Docker and deployed to Render, while the frontend will be deployed directly to Netlify. Environment variables will be favored over other types of configurations for the most part. Finally, to ensure that the backend and front end are released together, we will create a release script that releases both of them at the same time.

# Testing strategy

The application will be built using a test-first approach. For each new feature, tests will be written first, then the application code will be written so that all tests pass.