Feasibility and research

In this Document I will be going over the frameworks and tools chosen for this project and my reasoning behind my choices. This will include:

* Project Overview
* Integrated Development Environment.
* Source control management
* Programming Languages.
* Libraries and frameworks for the front end and backend.
* Testing
* Deployment methodologies

# Project Overview

For my project I have chosen to build a quiz application for apprenticeship training providers. However, I will be creating it using web technologies, this provides a few benefits over creating native applications for each mobile platform.

By using web technologies each platform that supports a modern web browser will be able to load and run the application allowing easy portability between platforms.

Along with this creating a web application would create a more consistent experience for users as it will use the same user interface on each platform.

By creating a web app, only one code base needs to be developed and maintained, along with only having to use one language and set of standard libraries, this provides a more cohesive codebase.

On the overhand, by using web technologies a couple of downsides will arise. For instance, the application will likely not conform to the platforms UI style which might confuse users.

Along with this, compared to native applications; there could be a performance difference. This is because using native libraries will always be faster than running a web browser, due to there being one less level of abstraction.

# Programming Languages

As this is a web application, I will be using languages that are like JavaScript or get transpiled to JavaScript. Along with this NodeJS runs on JavaScript allowing a unified language for both the frontend and backend

## Typescript

As this will be a web application, I will be using a combination of Typescript and JavaScript.

My reasoning behind this is typescript and its type system will prevent common JavaScript runtime errors caused by incorrect types. Along with this, typescript includes language features utilised by some typescript libraries that make creating applications easier, for instance inbuilt jsx, for react, and decorators.

## Bash

This will be used to create script for installing and setting up development environments. This is useful as most terminal applications will support bash except for Windows.

# Frameworks/Tools

JavaScript comes with a wide variety of frameworks useful for creating applications on both the frontend and backend. by using JavaScript for both the front and backend there are places where I can utilise the same libraries and

## Nextjs

Nextjs is a frontend framework based on react. It simplifies the creation of frontend applications by providing a structure like how normal webpages are stored in a web server.

Each page is a react component. Stored in its own file in a ‘pages’ folder. This allows new pages to be added quickly, whilst having client-side routing between pages setup in the background.

It also has a unique feature where it can ‘render’ pages, generating the required html; on the server and send that over so the user’s device will spend less time running react re-renders and make the application seem faster.

## Nestjs

Nestjs is a backend framework for creating web servers. Its main benefit is including a few libraries that make constructing data models, endpoints and tying them together easy. This is because it utilises some of typescripts extra language features, such as decorator functions; to create and assemble a backend server.

It also has a variety of extra libraries that can be imported for extra functionality that could be used later for some use cases

## Graphql/Relay

Graphql is a query language used for querying structured data from a backend. It is used instead of REST for making API calls. Its benefits include single REST Endpoint for all queries, requests reflect the structure of the returned data, a strict type system. Along with this, Relay is a client for graphql API’s, created by Facebook; that integrates directly with react and conforms to the flow architecture.

## Postgres Database

I will be using Postgres as it is supported on hosting services like Heroku and in NodeJS, while I could use SQLite many hosting services have ephemeral storage, where it can be cleared at any time thus erasing any stored data.

# Integrated Development Environment

## Visual Studio Code

I will be using visual studio code as my development environment. This is because of its inbuilt support for JavaScript/typescript along with its rich array of plugins that can assist with developing applications

# Source Control

For source control I will be using git. This is because it is widely used with a large support base. Along with this I will be using an addon to git called git flow. Git flow is one of many ways of structuring a git repository, I chose t because it defines boundaries between different steps of a software’s development.

It structures a repo into three branches: “master” for a functioning product, “develop” for adding features to, before release, and feature branches for developing single features. There are other branches for different use cases.

Along with git, I will be using a project called NX to manage my project. NX is good for handling monorepos, where all code for a project is kept in one repository. This will allow shared resources between projects. It also comes with project generation system allowing for new projects to be quickly generated in the repo.

An issue with this is an increased complexity where the links between projects could be hidden from a developer as well as the configuration being specific to NX.

# Deployment

For deployment I’m choosing to use docker. This is because using docker compose will allow bundling the frontend, backend, and a database together, for quick deployment to a production server.