UX3 / DIG31

# A3: Assessment Brief

## Full Stack Website Application

### Overview

This assessment concerns the third and final step of your website app, that is to **build and deploy a full stack website application** based on your Project Proposal Document created in assessment 2. This full stack app will consist of a backend data service (built upon assessment one), and a frontend website application built as a single page application (SPA). Both your frontend and backend will be hosted online (at two separate URLs), with your frontend application harnessing your backend data service for user authentication and data services, resulting in a “full stack” application.

### Backend Data Service

In this assessment you will need to take your backend data service built in assessment one, and extend it to fulfil the technical requirements laid out in your Project Proposal Document from assessment two and the minimum requirements outlined in this document. Depending on your app idea, this may comprise of significant extensions to your backend, or just minor additions, regardless, the functionality follows the same structure and concepts covered in assessment one. Thus less weight is allocated to the backend in the marking rubric, leaving the majority of weight allocated to your frontend which is the major assessment component for this assignment.

### Frontend Application (Single Page Application)

To fulfil the frontend requirements for your app as set out in your Project Proposal Document (assessment two), you are tasked in this assessment to build a single-page-application (SPA) that successfully communicates with your backend data service. Your SPA must be built using HTML, SCSS (Sass) and JavaScript, leveraging modern web development practices and techniques. Furthermore you must utilise a web component library (such as Shoelace - <https://shoelace.style> ) to speed up development time, allowing more focus on your aesthetics and user experience.

### Important Note about Frameworks (React, Vue)

The modern web industry utilises a plethora of large frameworks such as React and Vue for speeding up the development process, allowing developers to leverage large libraries of existing code. However such frameworks as React and Vue are not for beginner/intermediate developers, and require considerable knowledge of advanced JavaScript concepts. Thus students are NOT recommended to use these frameworks in their SPA, as they can act as a roadblock to development, causing delays rather than speeding things up. It’s important students utilise the code and concepts taught in the weekly exercises which are based in vanilla JavaScript, and designed specifically for beginners in mind, with minimal layers of abstraction. For advanced students who already have prior experience using frameworks like React or Vue and wish to utilise them in their SPA, they must consult their tutors first before commencing any development. Here are some great articles written on this topic for those students wishing to learn more: <https://bit.ly/2ZUqys5> and <https://bit.ly/3qKTrCx>.

### Sourcing Content

When it comes to populating your app with content, you are welcome to source such content from online, such as stock media, and any other relevant content. It is most important your time and attention is placed on the overall functionality and user experience on the app, and not dedicated to creating content from scratch.

### Hosting

You will need to host both your backend and frontend online, accessible via two separate URLs, these must to be included in the final submission. Your hosting provider for the backend needs to be a NODE JS hosting provider, as standard web hosts (using LAMP servers or similar) will not be sufficient to run the backend service. Students are welcome to use a host of their choice, however it is highly recommended to use the provider **Heroku** (<https://www.heroku.com/>) as they provide a free plan, and are easy to use. For your frontend SPA, given this consists of just HTML,CSS and JavaScript (no special server requirements), this can be hosted with any standard hosting provider of your choice. However it is highly recommended to use the provider **Netlify** (<https://netlify.com>) as they provide an easy to user service with free accounts.

### MongoDB Database Hosting

You will need to host your MongoDB database online which is separate to your NODE JS backend. Students are welcome to use a MongoDB host of their choice, however it is highly recommended to use the service MongoDB Atlas - <https://www.mongodb.com/cloud/atlas> which offers a free cluster and is easiest to use.

### Utilising Public APIs

With a growing list of public APIs available on the web, web developers can easily provide a great range of functionality and services to their users WITHOUT having to build complex backend services themselves. From recipes, to health stats, movies, music, travel, and more, there is an API service for just about everything. You are invited to explore the resources below which list many publicly available APIs:

* <https://medium.com/better-programming/a-curated-list-of-100-cool-and-fun-public-apis-to-inspire-your-next-project-7600ce3e9b3>
* <https://rapidapi.com/collection/list-of-free-apis>
* <https://apilist.fun/>
* <https://www.programmableweb.com/apis/directory>
* <https://any-api.com/>
* <https://rapidapi.com/>

### 1. Essential Functionality

#### Backend Data Service

Your backend data service must adhere to the same functionality requirements set out in the assessment one brief as well as the additional requirements listed below:

* Your user collection must contain a field/property to store a profile/avatar image. This must be stored as the filename for the image, stored as a String data type.
* In addition to above point, your API user endpoint for the collection must also include functionality to process the upload, resizing and storage of the image. To achieve this you must utilise the following NPM packages:
  + **multer** (for handling multipart/form-data)
    - <https://www.npmjs.com/package/multer>
  + **sharp** (for converting large images)
    - <https://www.npmjs.com/package/sharp>

#### Frontend SPA

Your frontend single page application must provide the following minimum functionality:

* Sign in page
  + Sign in form with email and password fields.
* Sign up page
  + Sign up form with user fields as listed in assessment one brief under *Functionality > User Data.*
* Guide/Intro page
  + This page is the first page a NEW user sees after signing in for the first time. All subsequent sign-ins should redirect the user to the home page.
* Home page
  + This is the first page the user sees after signing in – excluding the very first sign in for new users which should redirect users to the guide/intro page.
* Profile page
  + This page contains all user fields, with the functionality to update each field.
* Sign out functionality
  + This is not a page, but a function included somewhere in your SPA allowing users to sign out of their current session.
  + Signing out must delete the local access token and redirect the user to the sign in page.

### 2. Technical Execution

#### JavaScript

* JavaScript Classes must be used at least once
* Fetch() must be used for making API requests.

#### Setup and Frameworks

* Application is to be a Single Page Application (SPA) utilising a single index.html file with a single <div id=”root”> element for all app content to be placed.
* A web component library must be utilised for UI components, such as Shoelace (<https://shoelace.style>) to speed up interface development.

#### Styling (SCSS)

* All styling must be written in SCSS (Sass), and compiled into CSS using a compiler (this is achieved with a bundler such as Parcel or Webpack).
* Layouts must be responsive using CSS @media queries.

#### Build Tools and Package Managers

* NPM must be utilised as the package manager.
* A bundler tool such as Parcel (recommended) or Webpack must be utilised to compile all modern ES6 JavaScript for compatibility with older browsers.

#### Quality Assurance & Hosting

* The final website code must adhere to quality requirements:
  + Compliant W3C Valid HTML
  + Compliant W3C Valid CSS
  + Free of any JavaScript errors
* Both Frontend and backend must be hosted online at two separate URLs.

### 3. User Experience

#### UX Flow

* Your app’s interaction flow is logical, smooth and conducive to project goals.
* Your app content is engaging and relevant.

#### Responsive Development

* App layout looks and functions as expected on desktop and mobile size devices
* CSS media queries are harnessed to achieve a responsive layout, strategically resizing and showing/hiding content to suit the device size and to optimise user experience.
* App appears and functions uniformly across multiple browsers and devices.

#### Animation

Your app must utilise animation to enhance the user experience, this should be accomplished via any one (or a combination) of the following methods:

* + CSS animation/transitions
  + JavaScript animation libraries such as GSAP (recommended) or Anime.js, or
  + SVG animations

### 4. Visual Design

Your app adheres to core design principles such as contrast, proximity, colour theory, and typography design. Final visual design is aesthetically pleasing and of professional quality and doesn’t compromise user experience.

### What, Where and When to Submit

Students must submit their assessment on **Blackboard** by the due date specified in the unit outline.

#### What to submit

Students need to submit the following in a single ZIP file:

* ProjectDetails.txt
  + Plain text file containing live URL for both frontend and backend
  + Login details (email and password) for each user access level, i.e. one user login per access level
* Backend.zip – zip of your backend project folder (contains all working code)
* Frontend.zip – zip of your frontend project folder (contains working code)