Sashin Tulsiram

sashintulsiram@gmail.com

sashin tulsiram – full stack developer challenge

Statement of Work

Table of Contents

[Statement of Work for Software Development Project 2](#_Toc145426473)

[Project Requirements 2](#_Toc145426474)

[Project Deliverables 2](#_Toc145426475)

[Project Plan 3](#_Toc145426476)

[Technical Approach 4](#_Toc145426477)

[Standards and Testing 4](#_Toc145426478)

[Standards 5](#_Toc145426479)

[Tests results 7](#_Toc145426480)

[Acceptance Criteria 11](#_Toc145426481)

[Closure 11](#_Toc145426482)

[Documentation Required 11](#_Toc145426483)

[Project Summary 12](#_Toc145426484)

# Statement of Work for Software Development Project

## Project Requirements

1. Front-end – The front-end development of our application will be performed using ReactJS, HTML, and CSS.
   * Single page.
   * Responsive layout.
   * Use any preferred compiler to build & run your site.
   * Case studies should be implemented as a slider.
   * Brands retrieved via API.
   * UI component to sort & filter brands.
2. Back-end – The back-end development of our application will be performed using Python, the relevant Python packages for the application, NodeJS and MySQL for the database.
   * API to retrieve data for Trusted by leading brands section.
   * Ability to filter and sort brands. - E.g. Sort alphabetically, filter by published.
   * Data should be stored in a database of your choice.
3. Assessment criteria – Each of the below assessment criteria will be detailed under the “Technology and Architecture” and a detailed explanation will be provided for each section.
   * Technical approach
   * Coding standards
   * Data structure
   * HTML & SASS or CSS structure

## Project Deliverables

* **A Full Stack Web Application**:
  + - **Database**: A MySQL database will be created with the required tables for the project. The data for the project is in the *“kjd-assets”* folder and will be inserted into the table.
    - **Back-end**: An API will be developed using Python – it will have capabilities allowing it to create, read, update or delete database records. The REST API operations for creating, updating or deleting database records will be tested but commented out in a final production source code as it is not required. All REST API operations will be tested using Postman. The REST API will return data in JSON. Flask is a micro web framework - it will be used to connect to the database, run our frontend framework, manage web connections and interact with our REST API.
    - **Frontend**: Single page with a responsive layout created according to Figma design, using provided assets, provided font, has a filter and sort UI component, and the ability to retrieve brand data via API from the frontend of the application. React will be used as the frontend framework of the application as it meets the requirements highlighted for the frontend.
* **GitHub Software Repository**: Create a repo with source code. Create a detailed ReadMe file on how to get the project running. Create a Requirements.txt file. Test ReadMe file on Windows 11 Enterprise and Ubuntu LTS 22.04.2.
* **Documentation**: Statement of Work.
* **~~Documentation~~**~~: Technology and Architecture.~~

## Project Plan

|  |  |  |  |
| --- | --- | --- | --- |
| Task No. | Detail of Task | Due Date | Status |
| 1 | Select application technology, development tools and software development methodology as per requirements:   * 1. Application technology – Python, MySQL, Flask, ReactJS, NodeJS and Python packages (pip, .   2. Development tools – VS Code, Postman, NodeJS, and MySQL.   3. Software development methodology – Waterfall. | 23/08/23 | Completed |
| 2 | Create draft documents:   * Statement of Work * Technology and Architecture | 23/08/23 | Completed |
| 3 | Development (Development Environment) – Phase 1 –   1. Download / install required development tools and application technology. 2. Set up GitHub repo. 3. Install Python. 4. Install NodeJS. 5. MySQL database. 6. Set up virtual environment. | 23/08/23 to  24/08/23 | Completed |
| 4 | Development (Development Environment) – Phase 2 –   1. Set up and configure MySQL database with required table and data. 2. Write source code using Python for Flask web server. Check for errors (syntax errors, runtime errors, logical errors, name errors, type errors, index errors, and attribute errors). Run and connect to web server. Commit code to repository. 3. Write source code using Python for REST API and connect to Flask web server. Check for errors (syntax errors, runtime errors, logical errors, name errors, type errors, index errors, and attribute errors). Test REST API using Postman. Commit code to repository. 4. Use yarn and npm to install the required packages for the React app. 5. Create a single web page with a responsive layout. 6. Error and exception handling 7. Use npm run eject to build config pack. 8. Test web application. 9. Commit code to repository. 10. Run the web application and test user functionality. | 24/08/23 to 27/08/23 | Completed |
| 5 | Development (Test Environment) – Phase 3 –   1. Unit Testing of web application. 2. Generate Requirements.txt for Test Environment. 3. Create ReadMe documentation for Test Environment. 4. Commit master source code to repository. 5. Test ReadMe document on Windows 11 Enterprise and Ubuntu 22.04.2. Test user functionality. | 28/08/23 | Completed |
| 6 | Create final documents:   * Statement of Work * ReadMe | 28/08/23 | Completed |
| 7 | Send a mail with .zip file containing all documentation and share GitHub software repository link. | 28/08/23 | Completed |

**Project Completion (%): 100 %**

## Technical Approach

The project requirements had to be listed and a plan drafted immediately to meet the deadline for the project. The development of the project was broken down into phases and then more manageable tasks.

The main challenges of the project were reviewed such as the time constraints, storing of SVG data, the rendering of SVG data, and frontend customization such as the filter and sort components. These challenges were researched first to implement the correct technologies. When selecting technologies to use, I selected technologies that I was most familiar with and had the capability to meet the project requirements.

## Standards and Testing

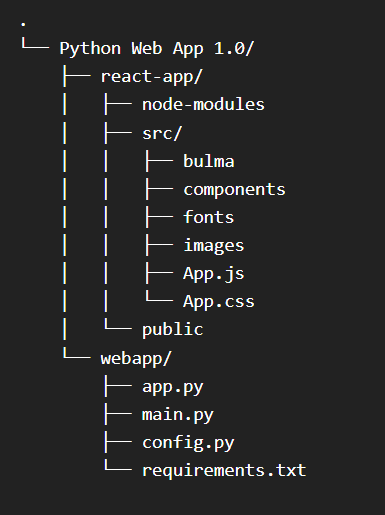
**There are no stated standards, however, to standardise our application – we will implement the below approach:**

1. Implement Standard naming convention – GitHub, React, Database, Python.
2. Keep code maintainable.
3. Keep code transparent, sane, and readable.
4. Keep code scalable.

**The application will be tested using the following techniques:**

1. Unit testing
2. Application security testing
3. System testing
4. Acceptance testing

**Folder structure of application:**



### Standards

#### GitHub naming convention

|  |  |  |
| --- | --- | --- |
| Artifact | Standard Naming Convention | Name of Artifact |
| Software repository name | [Artifact]-[Project Name] | Software\_Repository-\_Full\_Stack\_Development\_Challenge-Python\_Web\_Application |
| Branch(s) name and structure | [Branch Number]-[Software Developed\_Version] |  |
| Project | [Artifact]-[Project Name] | Project-\_Full\_Stack\_Development\_Challenge |
|  |  |  |

#### React naming convention (frontend)

|  |  |  |
| --- | --- | --- |
| Artifact | Standard Naming Convention | Name of Artifact |
| Functions | Pascal case | PascalCase |
| Variables | Pascal case | PascalCase |
| CSS Variables | Pascal case | PascalCase |
| Classes | Pascal Case | PascalCase |
|  |  |  |

#### Python naming convention (backend)

|  |  |  |
| --- | --- | --- |
| Artifact | Standard Naming Convention | Name of Artifact |
| Functions | Snake Case | Snake\_Case  Snake\_Case  Snake\_Case |
| Variables | Snake Case | Snake\_Case  Snake\_Case  Snake\_Case |
| Classes | Snake Case | Snake\_Case  Snake\_Case  Snake\_Case |
| Lists | Snake Case | Snake\_Case  Snake\_Case  Snake\_Case |
|  |  |  |

#### Database naming convention

|  |  |  |
| --- | --- | --- |
| Artifact | Standard Naming Convention | Name of Artifact |
| Databases | [Description]\_[ Description]\_db01 | st\_python\_webapp01\_db01 |
| Tables | [Description]\_[ Description]\_tb01 | trusted\_leading\_brands\_tb01 |
| Columns | [Description]\_[ Description]\_tb01 | id\_tb01  company\_name\_tb01  economic\_sectore\_tb01  sector\_tb01  industry\_tb01  brand\_image\_tb01 |

### Tests results

#### Unit testing

Unit testing is a software testing process for testing specific units, components, or software elements. This is the most basic type of testing, and the goal for this level of testing is to validate that each unit of code performs how it should and is free of bugs, errors, and glitches.

#### Application security testing

Application security testing inspects the software packages used in the application for security vulnerabilities or threats.

#### System testing

System testing inspects components like performance, load, reliability, and security with the goal of evaluating the end-to-end system specifications.

#### Acceptance testing

Acceptance testing is the final phase, it needs to validate the end-to-end business flow and check for things like cosmetic errors, spelling mistakes, and usability.

**Testing scope:**

1. Responsive layout
2. Case studies in a slider
3. Brands Retrieved via API
4. Sort and filter component

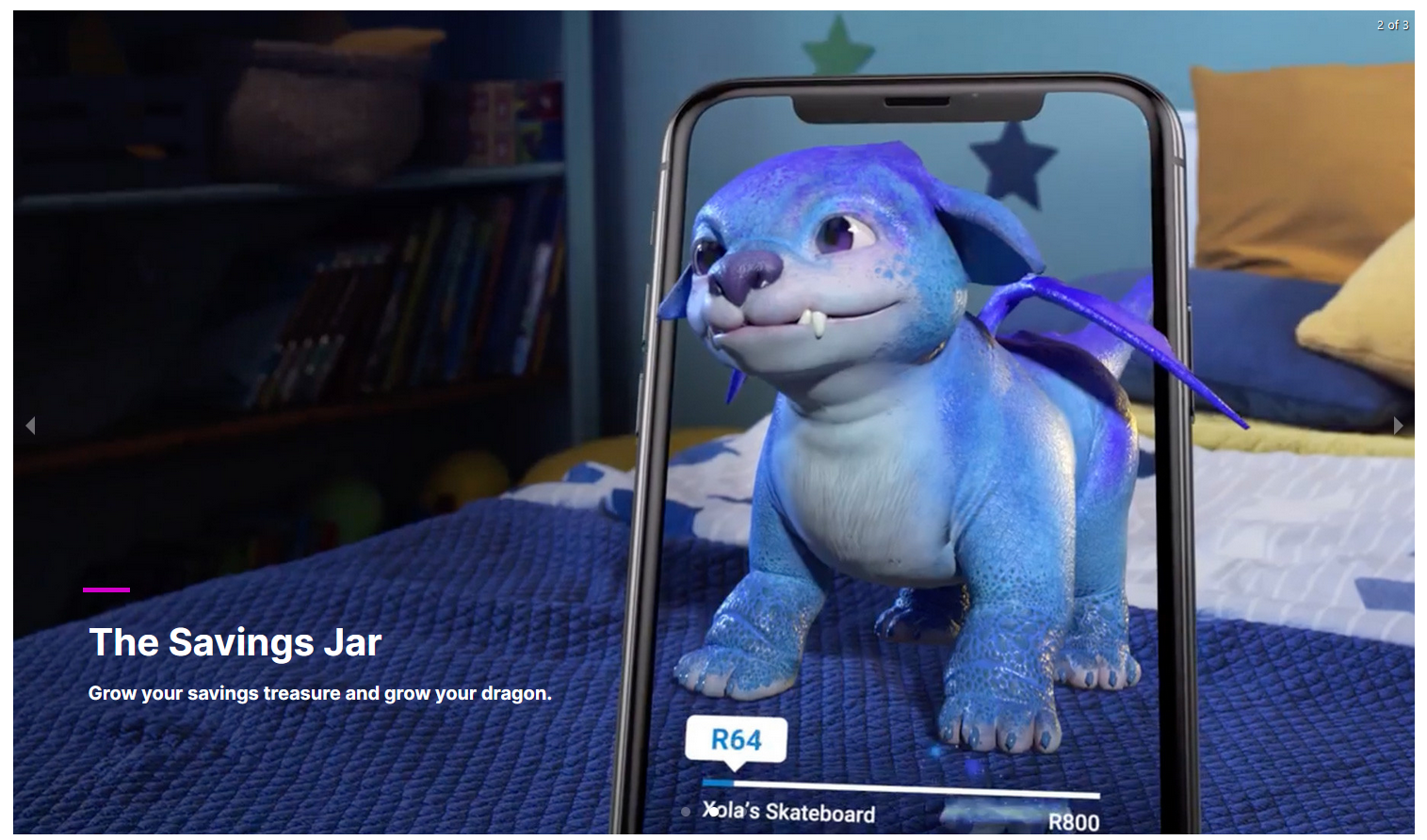
##### Responsive layout

Yes, elements adjust when adjusting the window from the browser.

##### Case studies in a slider

A screenshot of a video

Description automatically generated



A person looking surprised with his hands on his chest

Description automatically generated

##### Brands Retrieved via API

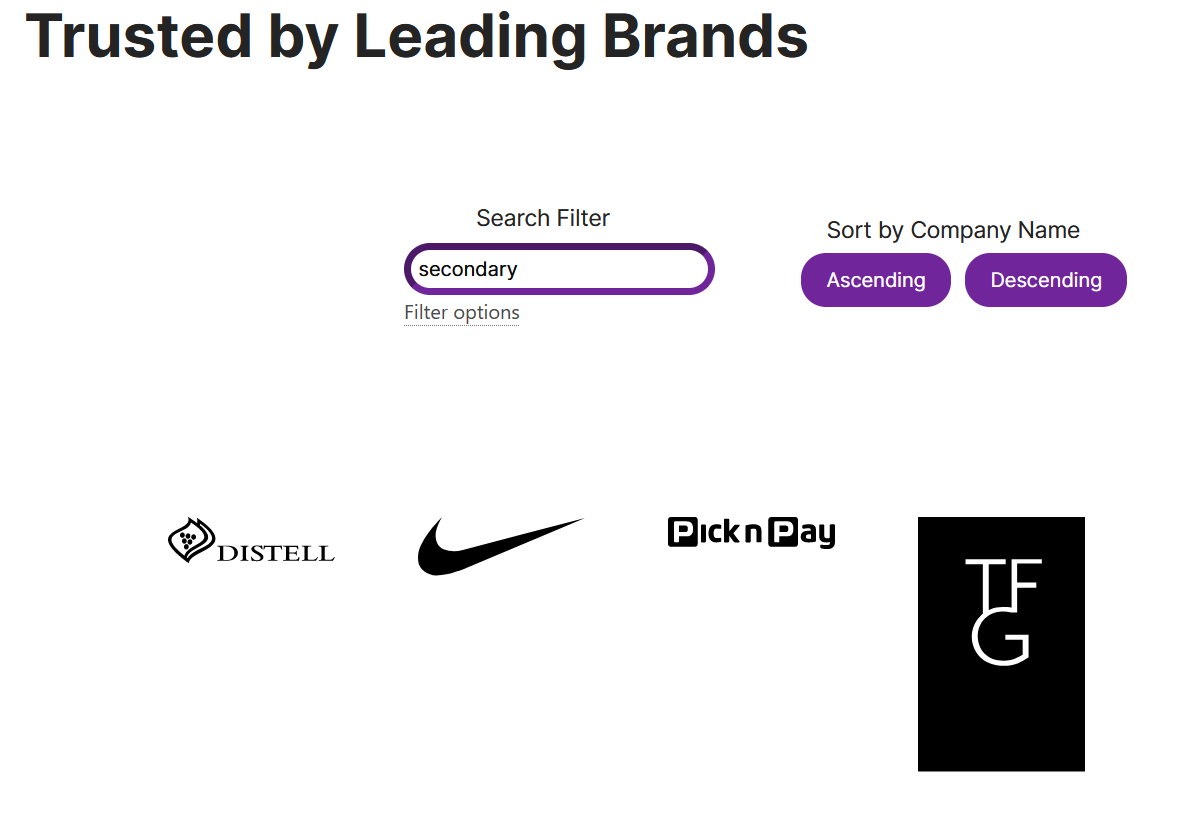
A group of logos with text

Description automatically generated

##### Sort and filter component

A screenshot of a computer

Description automatically generated



A screenshot of a website

Description automatically generated

## Acceptance Criteria

The following criteria needs to be met for us to determine if the project requirements were met:

1. Frontend – A responsive single web page designed according to the supplied Figma design, contains a slider, and contains a filter/sort component.
2. Backend – A REST API which is connected to the database. A web application which can serve the React frontend and handle web requests.
3. Database – A MySQL database with a table and columns to manage data for Trusted Brands.

## Closure

The project will be delivered after we have completed all our final testing and compiled the final documentation for the project. The source code for the project will be shared via GitHub, the database will export and stored on GitHub, and a mail with a link to the GitHub repo and documentation will be sent to the HR team.

## Documentation Required

1. Statement of Work – Details project information, requirements, deliverables, etc.
2. Document detailing how to set up test environment or ReadMe –
   1. Windows (Windows 11 Enterprise)
   2. Ubuntu 22.04 LTS
3. GitHub repo with source code, Requirements.txt, and ReadMe

# Project Summary

We have created a full stack development which meets the criteria listed for the project. The full stack development consists of:

* Backend web server – REST API using Python. Can be used to further improve the application by compiling the frontend web server into the backend web server.
* Frontend web server – Web server for client-side connections. Hosts our single web page and interacts with backend web server which runs on port 5000 on the local host. It was developed using ReactJS and bulma CSS.
* Database – The database consists of 1 database, 1 table and 7 rows. The columns are id, company name, sector, industry, economic sector, brand image and company description. The datatypes for the columns are as follows: INT (id), VARCHAR (company name, sector, industry, economic sector, company description) and LONGTEXT (brand image). The brand image was in a SVG format and by storing it as a string, we only needed to render it onto the frontend of our application. This also meant that we did not need to encode or decode this information, however, it can be a future improvement.
* API – The REST API was written in Python using the requests package which allows you to submit API methods and pymysql packages which allows you to connect to the database and submit queries.