CPD Application Investigation

Document

**PROJECT NAME:** CONTINUING PROFESSIONAL DEVELOPMENT SYSTEM

**VERSION:** 1.0

**DATE:** /05/2020

**PREPARED** BY NNAMDI OSUAGWU

**PROJECT SUPERVISOR:** MARK ZARB

**PROPOSER:** PROFESSOR HEATHER FULFORD AND DR MOIRA BAILEY

**VERSIONING**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **VERSION** | **IMPLEMENTED**  **BY** | **DATE** | **SUBMISSION DATE** | **TO BE**  **APPROVED BY** |
| **1.0** | NNAMDI  OSUAGWU | 10/02/2020 | 12/02/2020 | MARK ZARB |
|  |  |  |  |  |
|  |  |  |  |  |

# Project Oversight

**1.1 PROJECT DESCRIPTION:**

This project is a Continuing Professional Development App. It is a design and development platform that enables users/students to plan, record, and reflect their learning (formal and informal) based on 4 trigger types namely:

* Events
* Enactment of the entrepreneurial process
* Encounters
* Experiences

**Events** refer to reflection by participation in training sessions like classroom-based learning, professional development events, and personal/private study.

**Enactment of the entrepreneurial process** refers to reflection triggered by performing tasks during an enterprise or entrepreneurship project such as setting up a business as a student, also performing market research, financial upskill, and management practice.

**Encounters** refer to reflection triggered by networking and team discussions whether organized, Adhoc meetings, or scheduled professional groups.

**Experiences** refer to reflections that occur during incidents whether positive or negative of any kind.

Also, challenges and problems encountered during learning like from industry experience or in the field.

* 1. **AIM AND OBJECTIVE OF PROJECT:**

This project aims to increase the fluency in reflection during students learning by documenting and archiving the process of the learning as a habit to encourage the value of what is called “chewing over”, but this time using digital tools on an application. This can enable the student to review their learning as they grow in career and life. Also, many professional bodies require students to attach a CPD portfolio as part of the membership and registration process. To effectively communicate and achieve this, students will do two things. One, record their learning which is descriptive learning based on the 4 trigger types mentioned in section 1.1, and then reflect them which is analytical learning.

The existing system in place is designed as a “postcard” see picture below, where one side of the postcard depicts the 4E’s and the flipside is a short brief of the reflections from those triggers. A calendar of 12months will be used as a baseline period for students to demonstrate their learning through reflections after they have recorded the process.

* 1. **SCOPE AND PROCESS OF PROJECT:**

This is a process breakdown of the application in terms of the user interaction flow:

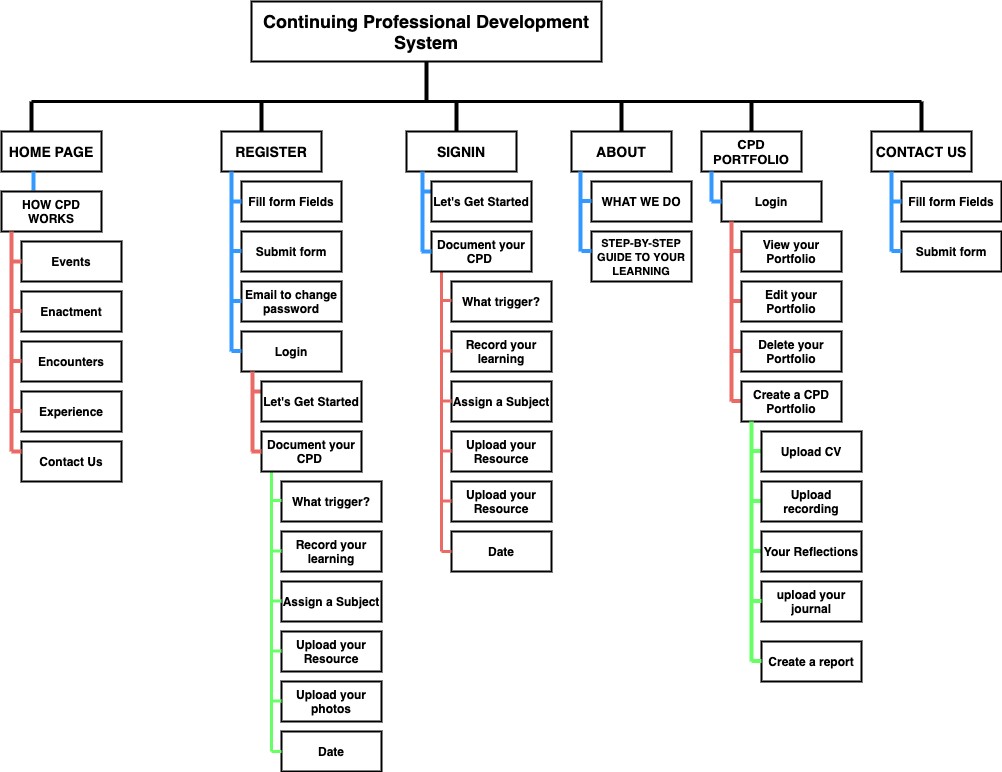


Figure 1.1 User-interaction flow

* 1. **METHODOLOGY:**

In this project, we will be using the Agile Methodology which makes use of the SCRUM framework. SCRUM makes it possible for the project to be broken down into items based on priority (hierarchy of functional features starting from what is most important). These items are called Product Backlog Items (PBI’s or PBL). So, we will have to choose the items (features) to build and deliver as an MVP (Most Viable product) in a sprint (first cycle). Each sprint will have its items to be shipped and deployed called Sprint Backlog (SBL). The SBL is a subset of the BPL based on what can be developed in a sprint.

* 1. **TECHNOLOGY AND SYSTEM ARCHITECTURE:**

In this project, we will employ the Monolithic architecture pattern. This will host a single code base with multiple modules. Modules are files with separate functions that implement certain technical and business features. The whole system in each sprint must be ready before it can be deployed.



Figure 1.2 Monolithic Architecture

U

SER

I

NTERFACE

BUSINESS

LOGIC

DATA

LAYER

DATAB

ASE

In this project, the technology we will employ includes the web stack to be used for the development of the application. MERN stack will be used which makes JavaScript the programming language. The frontend includes ReactJs, Back-end is NodeJS and the database is NoSQL which is MongoDB. My choice of this web stack stems out from the fact that NodeJS has a cross-platform run-time environment that makes mobile, web, and desktop able to consume the same endpoint (data) through REST API (Representational State Transfer).

ReactJs is also most suitable for the client-side due to its inbuilt style of modularity, supports single-page applications, makes page routing easy, and flexibly handles state(data). The database chosen is MongoDB, a NoSQL database system with its ability to handle large unstructured data in a flexible way. Queries are faster and have great performance.

# Project Constraints

**2.1 CONSTRAINTS**

This is a list of the major constraints identified with the project.

|  |  |
| --- | --- |
| **CONSTRAINT** | **ISSUE/SOLUTION** |
| Server Hosting | CPD is designed to meet user(students) needs, yet to be effective and operational, the client-side and server-side script has to be deployed and hosted on a Server (remote cloud). For this project, the serving Host will be initially Heroku which is preferred due to it is flexible and easy to use for this project. then later we will consider AWS, the tools used for Hosting are Git installations and Heroku CLI. Heroku will manage this application with Git, the popular version control system. The application is first deployed or pushed to the Git repository, then the Heroku command Line interface tracks the Git repository. Heroku provides Pipelines, which makes the CPD easy to maintain in separate staging and production environments. |
| GitHub | GitHub repository will be employed. GitHub is a hosting platform for version control and collaboration. A private repository will be created for this project on GitHub with a default branch called Master. Subsequently, other branches will be created based on the different features. For instance, working on the Signup page, the branch name will be called feature\_signup. The project supervisor will be made a collaborator. GitHub master branch is the base branch that will merge all feature branches through a pull request, then the source code can be deployed on a cloud server (Heroku) from the master branch, and changes can be tracked for both GitHub and Heroku. |
| Client-Side (REACT) | Users of this application (CPD) must be able to interact with an interface. They must be able to see all the pages designed in Fig 1.1. HTML for the structure of the webpage (text and content), CSS for styling the webpage (putting aesthetics to the page) will be embedded in the REACT.  The front-end will have an interface that students can interact with. |
| Sever-Side (Node and MongoDB) | This application requires a server to receive the request, and also should respond to the request. It understands the language of the browser and can respond to the request made by a clientside (web interface) over HTTP (Hypertext Transfer Protocol). The server also has a controller function that can communicate with the database and render the data to the front-end. This application will make use of the MongoDB database for storing, querying, retrievals made by several users (students that are registered. |
| Features  • Login system | The system in place for users to log in will take series of steps.  There will be a form that takes in username and password. This credential will be validated from the front-end and also the back-end |
| • Roles and Permission | Two users exist in the CPD app. Logged-in users and general users. Logged-in users can access a CPD portfolio and, a general user cannot. |
| • File Upload | The logged-in user will be permitted to upload resources as part of his CPD representing the materials of his learning. The file upload input will be displayed after he has logged in using a login credential (username and password), and the database will store a reference to this material. |

# Project Requirements

**3.1 FUNCTIONAL REQUIREMENTS**

This section details how the system should behave. In terms of priority, we will use the MOSCOW Method. “M” stands for must, “S” for Should, “C” for could, “W” for Won’t.

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Component** | **Method**  **(MOSCOW)** | **Description** |
| User  Management | Register | S | A user/student can register by entering his name, email, and password. The user waits for approval by the application’s backend. |
| Login | M | A user will enter his email and password to access the admin dashboard. |
| Logout | M | A user can log out through the navigation bar and he won’t be able to access the CPD portfolio except he logs back into the system. |
| Role | Logged-in User | M | This user has access to the CPD portfolio with Admin rights and can create a CPD portfolio, view his portfolio, report his reflections, and more. |
| General User | M | This user can only visit the home page, see how the CPD works. He can also visit the about page and view step by step process of learning. He can also contact us through the contact us form for inquiry. |
| Authentication | Login | M | The system will check the username and password of the user exist in the database as approved by the system. If not, he is redirected to the homepage to register. |
| Authorization | User role check with session or cookies | C | The system check for the user role after logging if the user is a general user or logged-in user through his session. |
| User  Interaction | Search | C | A logged-in user can search for a CPD based on the date he created one. From the CPD search bar, the user will input a keyword and the database will dynamically recall the last date of the document by filtering the search. |
| Resource upload | M | Resources include e-books, images, and files for students to upload while documenting their learning. It is a feature that can read in an image file, e-book and others. |
| Post | M | Logged-in users can access the features of publishing a post and upload an image. The feature of deleting a post and delete comments made by general users and approve comments is going to be nice to have in the future. |
| Profile | C | A logged-in user shall be able to create a profile by filling a form in the CPD Portfolio area after log-in. |
|  | Hyperlinks | M | Links will be created on every page for easy navigation. |
| Navigation bar | M | The Homepage, register page, Login page, about us page, CPD portfolio page, and contact us page will have icons on the navigation bar that routes to its direct page |
| Database | CRUD | M | The database shall be able to perform a create, read/write, update and delete the user and a portfolio |

**3.2 NON-FUNCTIONAL REQUIREMENTS**

These are quality attributes the product should have to give the user a good experience.

|  |  |  |
| --- | --- | --- |
| **DOCUMENT NAME** |  | **DESCRIPTION** |
| Software Requirements | • | This is a web-based application for a start, so an internet connection must be established. |
| Hardware Requirements | • | CPD is intended to work on any computer with a minimum capacity of 4GB of hard drive, Memory of 128MB and above, operation system of WIN XP/VISTA/MAC/LINUX |
| Security | • | Password shall never be viewable at the point of input. |
|  | • | There must password encryption for users in the database during storage. |
|  | • | Users will in the future be forced to change their password the next time they log in and there will be a password expiration time. |
|  | • | HTTPS encryption must be used to protect user credentials over the web when the begins to scale. |
|  | • | The system will approve admin rights to users that have logged in. |
| Performance | • | There should be no limits in the number of users that can be added or created in the database. |
|  | • | The programming language should be up to date in a version that improves performance. |
|  | • | The version of Node should give massive performance improvements and thereby, server response time optimized. |
|  | • | The server’s location must be in proximity to the client, thereby reducing network latency. |
| Reliability | • | The application must ensure a great user experience across all pages. It automatically sorts the fresh content to the user during search. |
|  | • | Links to automatically and dynamically loads page and resource from the database and renders it to the user on his request page. |
|  | • | There is no data loss due to the database type and data type are accurately chosen for all input. |
|  | • | The URL is friendly and file uploads are received seamlessly by the system. |
| Scalability | • | When the app starts to receive more requests per second than the server limit, horizontal scaling is recommended due to its nature of purchasing CPU, memory space, and Hard disk, servers, database per scaling. With this type of scaling, we will be able to keep track of what is been used. |
|  | • | In the long run, an MVC pattern of design for the code will help scale the application due to its modularity. Different developers can work on different areas when it scales. |
| Technology Architecture | • | The client-server model and architecture are built by using Front-end stacks for hosting the page, backend for the server and database to deliver the data, which are NODE and Express, then the database is MongoDB. |
| Ease of Use | • | Users with no technical background would visit how the page works in the home page in order to manage the website, create a CPD, edit and manage their learning process and content online. |
| Availability | • | The application shall be available for use 24/7 |
|  | • | The system shall achieve more than 80% uptime. |
|  | • | The cloud service must keep track of changes with a system alert, also concerning the upgrade of configurations/ library stacks and new deployments. |
| Testing | • | The input for username must be characters of length not greater than 30 |
|  | • | The email for signup must be an input type of email for HTML validation |
|  | • | The password from the input to be stored in the database must be added to a randomly generated special character by the server and hashed using hashing algorithms. |
|  | • | There must be cross-platform testing on several browsers. |

# Appendixes

**KEY TERMS:**

**MOSCOW METHOD**

1. **M - Must have these.**
2. **S - Should have these if at all possible important but not critical to success.**
3. **C - Could have this if it does not interfere with 1 and 2.**
4. **W - Won’t have these but would like in the future (alternatively want) the lowest payback items and not included in the schedule.**