|  |  |
| --- | --- |
| **SCHOOL** | School Of Computing |
| **COURSE** | Diploma in Information Technology |
| **MODULE CODE** | ST1503 |
| **MODULE NAME** | Fullstack Web Development Project |
| **ACADEMIC YEAR** | 2020/2021 |
| **SEMESTER** | 1 |
| **Team Size** | 2 |

Overview

You are an intern at a company named JiBaBoom. Depending on your choice of a problem statement, JiBaBoom will be a different company providing a different solution. But in all cases, JiBaBoom will require you to build an independent prototype to help extend their application.

[**Questions/Issues?**](#_iq5k76pfpg9d) **2**

[**Problems**](#_jw99evxtlzy5) **2**

[**Schedule**](#_30j0zll) **2**

[**Submission Instructions**](#_1fob9te) **3**

[**Technical, Non Functional Requirements**](#_3znysh7) **3**

[**Basic Functionalities**](#_2et92p0) **3**

[Basic Data Viewer (Frontend)](#_58mt4aj1bsaf) 3

[Basic Insert API](#_3dy6vkm) 4

[Basic Result Viewer (Frontend)](#_1t3h5sf) 4

[Basic Result API](#_2s8eyo1) 4

[**Advanced Functionalities**](#_27bw2gxdj167) **6**

[Advance Data Viewer (Frontend)](#_9ypyzk4xp8i9) 6

[Advance Insert API](#_3rdcrjn) 6

[Advance Result Viewer (Frontend)](#_26in1rg) 7

[Advance Result API](#_lnxbz9) 7

[**Documentation**](#_35nkun2) **7**

[Project Planning](#_dxf7fwbnks1a) 7

[Project Worksheet](#_44sinio) 7

[Problem Worksheet](#_2jxsxqh) 8

[Wireframes](#_z337ya) 8

[Database Schema](#_1y810tw) 8

[Code Style/Quality](#_4i7ojhp) 8

[Weekly Report](#_2xcytpi) 8

[API Documentation](#_1ci93xb) 8

[**Hosting**](#_3whwml4) **8**

[**Technologies Required/Recommended**](#_2bn6wsx) **9**

# Questions/Issues?

Please post on teams or leave a comment in the google document if you spot any inconsistencies or if you have any queries about the documentation.

# Problems

There will be a total of 5 problems to choose from

|  |  |
| --- | --- |
| **Problem** | **Description** |
| Advertisement | Different companies provide different advertising options to reach a different range of audiences, with a given budget, compute the best set of advertisement options to engage to reach the most audience. |
| Lectures, Halls and Technicians | To hold lectures, we need halls and technicians on standby, given the schedule of lectures and technicians, compute the best way to allocate lectures to halls such that the number of halls used is the minimum and compute how many surpluses/lack of technicians there will be. |
| Meeting Time Voter | Help several participants choose a time for a meeting. It can be based on their availability or unavailability. |
| Music Festival | Help visitors to a music festival choose the set of performances to watch for them to make the best of their money, either based on the number of performances watched or the overall popularity. |
| Task Allocator | A project like this has several tasks and some of the tasks have deadlines, schedule the tasks to minimize the amount of lateness and split the task equally among 2 team members. |

# Schedule

The project will be separated into 2 phases, each phase is further divided into 2 CA which focuses on different aspects of the project. Below are the deadlines for each phase.

|  |  |  |  |
| --- | --- | --- | --- |
| **Phase** | **Deadline** | **CA** | **Weightage** |
| Phase 1 | 2020/05/25 0000hrs | 1 | 30% |
| 2 | 10% |
| Phase 2 | 2020/08/17 0000hrs | 3 | 40% |
| 4 | 20% |

|  |  |  |
| --- | --- | --- |
| **CA** | **Objective** | **Means of Assessment** |
| 1 | To assess your ability to understand the question and come up with plans to work on the project. | Submission of different documentation and worksheets. |
| 2 | To access your ability to develop a small part of the project based on your documentation. | Code submission and demo. |
| 3 | To assess your ability to develop the rest of the project based on your documentation. It would also assess your ability to react to changes should you realize that your initial plan had errors. | Code submission and demo |
| 4 | To assess several aspects:   1. Your ability to document any changes/additional information 2. Your ability to present on your project and how the development and testing were carried out. 3. Your ability to host your application on a cloud hosting service. | Submission of different documentation and presentation. |

# Submission Instructions

All submissions, which include documentation, worksheets, and code, will be taken from the last commit of your master branch on Github before the deadline. Thus, please ensure that you commit your latest version before the deadline.

# Technical, Non Functional Requirements

1. Frontend
   1. The installation process must only include cloning the repository, there **should be NO** additional installation.
   2. Must be hosted on localhost on port **8080** by running **npx http-server** in the **./frontend** directory
   3. Must have **index.html** as the entry point.
   4. Must not require server side rendering.
2. Backend
   1. The installation process must only include cloning the repository and running **npm install** to install any dependencies.
   2. Must be a **nodejs** application
   3. Must define the **npm start** script to start running the backend
   4. Must be connected on port **3000**
   5. Must not be serving any static content
   6. Must serve JSON responses only
3. Database
   1. Must connect to a free-tier database cloud service provider (e.g. elephantsql)
   2. Must require no additional installations.

# Basic Functionalities

Below is the list of basic functionality that will be graded. Additional API will be required from the backend to support the functionality of the frontend.

|  |  |
| --- | --- |
| **Features** | **Details** |
| Basic Data Viewer (Frontend) The data viewer should allow admins to view the data available in the database, it should support the following functionality:   1. Displays data 2. Filtering by 2 attributes 3. Pagination    1. First Page    2. Previous Page    3. Next Page    4. Page Size   You will need to:   1. Create a wireframe 2. Justify your design by showing at least 3 other existing designs and what you liked or disliked about them. 3. Demo the functionality   This will be graded based on the demo  *The frontend should not communicate directly with the database, but instead, the frontend should send an HTTP request to the backend, then the backend should fetch the requested data from the backend and return it to the frontend. Thus, you’ll probably need another API here and it should be documented* | **Refer to individual problem statements for the required filters.**    *Sample Wireframe* |
| Basic Insert API The Backend should have an API for JiBaBoom to insert data, which will then be viewed in the basic data viewer and used for computation.  It will need to   1. Follow the given schema 2. Allow insertion of multiple data with a single HTTP request 3. Return appropriate error message when insertion failed due to either    1. Duplicate entries    2. Invalid Input    3. Database error    4. Any other errors   This will be tested by an automated script. | **Refer to the individual problem statement for the API Schema.** |
| Basic Result Viewer (Frontend) The result viewer should allow the admin to verify the computation result visually. It should have the following functionality:   1. Have sufficient fields for admin to enter the input for computation 2. Should display the computation result on the webpage   You will need to   1. Create a wireframe 2. Justify your design by showing at least 3 other existing designs and what you liked or disliked about them. 3. Demo the functionality   This will be graded based on the demo. | **Refer to individual problem statements for the required fields.**    *Sample Wireframe* |
| Basic Result API The Backend should have an API for JiBaBoom’s backend to access the computation result.  It will need to   1. Follow the given schema   This will be tested with an automated script | **Refer to the individual problem statement for the API Schema.** |

# Advanced Functionalities

Below is the list of advanced functionality that will be graded. Additional API will be required from the backend to support the functionality of the frontend.

|  |  |
| --- | --- |
| **Features** | **Details** |
| Advance Data Viewer (Frontend) The advanced data viewer can be either an entirely new page or a modification of the basic viewer. In either case, it should support the following features:   1. Displays data 2. Filtering based on the problem description 3. Pagination    1. First page    2. Previous page    3. Next page    4. Page size   You will need to:   1. Create a wireframe 2. Justify your design by showing at least 3 other existing designs and what you liked or disliked about them. 3. Demo the functionality   This will be graded based on the demo.  *The frontend should not communicate directly with the database, but instead, the frontend should send an HTTP request to the backend, then the backend should fetch the requested data from the backend and return it to the frontend. Thus, you’ll probably need another API here and it should be documented* | **Refer to individual problem statements for the required filters.**    *Sample Wireframe* |
| Advance Insert API The Backend should have a new for JiBaBoom to insert the advanced data.  It will need to   1. Follow the given schema 2. Allow insertion of multiple data with a single HTTP request 3. Return appropriate error message when insertion failed due to either    1. Duplicate entries    2. Invalid Input    3. Database error    4. Any other errors   This will be tested by an automated script. | **Refer to the individual problem statement for the API Schema.** |
| Advance Result Viewer (Frontend) The advanced result viewer can either be an extension to the basic result viewer or an entirely new page, in either case, it should allow admin to verify the computation result visually. It should have the following functionality:   1. Have sufficient fields for admin to enter the input for computation 2. Should display the computation result on the webpage   You will need to   1. Create a wireframe 2. Justify your design by showing at least 3 other existing designs and what you liked or disliked about them.   Demo the functionality | **Refer to individual problem statements for the required fields.**    *Sample Wireframe* |
| Advance Result API The Backend should have a new API for JiBaBoom’s backend to access the computation result.  It will need to   1. Follow the given schema   This will be tested with an automated script | **Refer to the individual problem statement for the API Schema.** |

# Documentation

To ensure you understand the assignment, and the process needed to integrate the different components together, there are some documents and worksheets that you need to fill up.

|  |  |
| --- | --- |
| **Documents** | **Tasks** |
| Project Planning This document helps you identify the smaller tasks and helps you create a rough schedule for you and your partner to follow  Filename: **project\_planning.md** | 1. Breakdown the project into many smaller parts 2. Create Github Issues to match the list of tasks 3. Create Tags for issues 4. Create Milestones for the project 5. Assign tasks to members 6. Assign milestones to tasks 7. Assign tags to tasks |
| Project Worksheet This document helps you identify how the different components interact with each other.  Filename: **project\_worksheet.md** | 1. Identify the components that each component communicates with. 2. Identify the messages between each communication. 3. Identify the sequence of each communication. |
| Problem Worksheet This document helps you understand how to compute the result for your chosen problem.  Filename: **problems/<PROBLEM\_NAME>.md** | 1. Go through to make sure that you understand the steps taken to solve your chosen problem. |
| Wireframes As seen in the functionalities you are required to create wireframes and justify your design by finding 3 other existing designs and commenting on the pros/cons of the existing designs.  Filename: **basic\_frontend\_data\_viewer.md** | 1. Add the wireframe 2. Add the 3 existing designs to be used for justification 3. Add what you liked about the existing designs 4. Add what you dislike about the existing designs |
| Database Schema Modify the fileto document how your database looks like.  Filename: **schema.md** | 1. Create statements for every table. 2. The creation statements should be ordered such that dependencies are created first. |
| Code Style/Quality This document helps to guide the look and feel of the code so that even when there are multiple developers, the style remains consistent.  Filename: **code\_style.md** | 1. Identify naming convention 2. Identify common agreement for syntaxes such as brackets and commas |
| Weekly Report This document helps keep track of what happened weekly. Everything can go in here, change in plans, development process, thoughts, etc…  Filename: **weekly\_reports/weekly\_report\_X.md** | 1. What did the team do this week? 2. What did the team plan to do next week? 3. Any additional thoughts? |
| API Documentation This document allows you to define your API schema.  Filename: **api.md** | For each additional API you implement, it should include:   1. HTTP Method 2. Endpoint 3. Request Body Format 4. Response Body Format 5. Error Body Format 6. Request Body Sample 7. Response Body Sample 8. Error Body Sample |

# Hosting

At the end of the assignment, you’ll also need to host the application on some free hosting platform

|  |  |
| --- | --- |
| Component | Hosted On |
| Frontend | **Github Pages** |
| Backend | **Heroku** |

# Technologies Required/Recommended

|  |  |  |
| --- | --- | --- |
| Technology | Purpose | Optional? |
| HTML/CSS/JavaScript | Frontend Development | Compulsory |
| NodeJs | Backend/Mobile Development | Compulsory |
| ExpressJs | Backend Development | Optional |
| Markdown | Documentation | Compulsory |
| Git | Version Control | Compulsory |
| GitHub | Git Online Service | Compulsory |
| VSCode | Code Editor | Optional |
| PlantUML | Text to Diagram Generator | Compulsory |
| Github Page | Frontend Hosting | Compulsory |
| Heroku | Backend Hosting | Compulsory |
| pg | PostgreSQL client for Node.js | Optional |
| PostgreSQL | Database | Optional |
| ESLint [VSCode Extension] | JavaScript lint | Optional |
| Jest [VSCode Extension] | NodeJs Test runner | Optional |
| Prettier [VSCode Extension] | Code Formatter | Optional |
| REST Client [VSCode Extension] | Send HTTP request | Optional |
| Live Share [VSCode Extension] | Real-time collaboration [HBL] | Optional |
| Lucidchart | Wireframing | Optional |
| Postman | Send HTTP request | Optional |
| ReactJs | Frontend Library | Optional |
| Bootstrap | CSS Framework | Optional |
| JQuery | Frontend Library | Optional |