Candidate Number: 7629

Jonathan Broster

A (web) app that controls a Database of suppliers for a company.

H446

A-Level Computer Science

A-Level Computer Science

Practical Programming Project

Contents

[Analysis 1](#_Toc137033106)

[Summary 1](#_Toc137033107)

[Computation 1](#_Toc137033108)

[Stakeholder 1](#_Toc137033109)

[Research 1](#_Toc137033110)

[Essential features 1](#_Toc137033111)

[ 1](#_Toc137033112)

[Limitations 1](#_Toc137033113)

[Requirements 1](#_Toc137033114)

[Success Criteria 2](#_Toc137033115)

[Design 2](#_Toc137033116)

[Development and testing 2](#_Toc137033117)

[Evaluation 2](#_Toc137033118)

# Analysis

## Summary

I would like to create a (web) app that controls a Database of suppliers for a company.

## Computation

The solution is applicable for computational methods for many reasons. The solution will contain a database which will need to store thousands of entries and be sorted and modified extremely fast. This is simply not feasible in any way other than by computer, or else one would be dealing with a massive stack of paper, which is not very inefficient. In addition to this, it is much easier to get analytics and statistics from the computer database than the filing cabinet.

### Decomposition

The project can be split into three main parts which should be able to work independently of each other:

* Database
* UI
* Web Host
* Statistics Engine

By splitting the project in this way, I can ensure that the project can be adapted to meet the needs of any other potential clients other than the stakeholders, so that the code is versatile and works on many devices. I should also be able to employ abstraction in these three areas, so that when developing the UI, I do not need to code the database directly, I only need to interface with it.

The problem involves some steps:

1. Load the database in a web app.
2. Allow the user to interface with the database.
3. Change the database to fit with what the user changed.
4. Reload the database.

This is an example of how the user will interact with the solution. With the help of SQL, this can be done quickly with no interference with the user.

### Divide and Conquer

Solving the above decomposed problems together seems technically challenging. To be able to write my solution to the problem efficiently and easily, I will need to conquer each of the decomposed problems separately. I will even divide these components into smaller algorithms and subprograms that seem more manageable on their own.

A picture containing text, font, white, algebra

Description automatically generated

## Stakeholders

RAPITA

A close-up of black text

Description automatically generated with low confidence

## Research

A black text on a white background

Description automatically generated with medium confidence

## Essential features

## 

## Limitations

TIME



## Requirements

WEB BROWSER – CHROME ETC.

A picture containing text, font, white

Description automatically generated

## Success Criteria

TODO List



|  |  |  |
| --- | --- | --- |
| Criteria Number | Description | Achieved? |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Design

## Decomposition

A picture containing text, font, white

Description automatically generated

## The Solution



## Algorithms

A picture containing text, font, white

Description automatically generated

## Usability Features



## Validation and Data Structures

A black text on a white background

Description automatically generated with low confidence

## Test Data



## Post-Development Data



# Development and testing

## Process

A white background with black text

Description automatically generated with low confidence

## Prototype Versions



## The Code

A picture containing text, font, screenshot, algebra

Description automatically generated

# Evaluation

## Testing

A picture containing text, font, white

Description automatically generated

## Success Criteria Testing

A close up of black text

Description automatically generated with low confidence

## Future Development



## Usability Features Testing

A picture containing text, font, white

Description automatically generated

## Future Opportunities for Useability Upgrades

Black text on a white background

Description automatically generated with medium confidence

## Maintenance



## Possible Improvements



## A picture containing text, font, white Description automatically generated