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. It should be intuitive and easy to use for a variety of users.

## 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

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, modular 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. Furthermore, it will make the project easier to debug because I will be able to more easily identify in which section of the project the bug is in, rather than having to debug the entire code.

The solution involves an algorithm which has 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.

### Abstraction

The database contains a huge amount of data, which is not all relevant to the user. The solution’s primary purpose is to be able to abstract this data so that inly the useful things are shown at any one time. These useful things do change though, so it is imperative that the database be dynamic and changing to the user’s needs. One way I can do this is by a search/sort system where the user must type in the subject that they are looking for and the database shows them the contents that are related to it.

Furthermore, the user does not need to see the actual processes that involve searching the database. To make the solution more intuitive to a non-programmer, there must be a GUI on top of the database which acts to let the user interact with the database, without the user knowing all the SQL to be able to interface with it directly. Therefore, there should be buttons which are clearly labelled to represent the lines of SQL. For example, the button for “SELECT \* FROM Suppliers.Companies WHERE Product = ‘Electronic’ ORDER BY name;” could be a drop-down box that allows you to select product types to view.

### Database Logic

The database must be efficient because it will store thousands of suppliers, and each must be able to be edited, added removed and changed. This can be done with SQL. For example, I can use statements like “SELECT \* FROM Suppliers.Companies WHERE Product = ‘Electronic’ ORDER BY name;” to find all of the companies who supply electronic products.

### UI and Statistics

I might also add graphs to show the statistics of the database. These require computational algorithms. For example, I will need to find the mean, standard deviation, etc. I will need an algrothim that plots a curve on a graph so that I can give analysis of the database that is visual and easier to understand at a glance.

### Version Controls Alorithms

Another extension possiblity is Verson Control. There could be a graph of how the database has cahnged over time. changed over time is needed, I would need an algorithm to collect the data over the recent time. I could also create an algorithm which logs all the changes so that a manager can look and see who made any changes that might not have been good. There should then also be the option to go back to previous versions of the database, so that if a large mistake is made, it can be rolled back to a different version of the database.

A picture containing text, font, white, algebra

Description automatically generated

## Stakeholder

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

# Development and testing

# Evaluation