Finished Date 07/03/2017

Author: David Cook

DSA Assignment Design Documentation

Table of Contents

[1 Introduction 4](#_Toc476862905)

[1.1 Purpose of the document 4](#_Toc476862906)

[1.2 Scope of the System 4](#_Toc476862907)

[2 SYSTEM OVERVIEW 5](#_Toc476862908)

[3 Database Design 6](#_Toc476862909)

[2.1 SYSTEM CHARACTERISTICS 6](#_Toc476862910)

[4 SYSTEM DESIGN 7](#_Toc476862911)

[4.3 Documentation Standards 7](#_Toc476862912)

[4.1 Naming conventions 7](#_Toc476862913)

[4.2 Programming Standards 7](#_Toc476862914)

[4.3 Software development tools 8](#_Toc476862915)

[4.4 Outstanding Issues 8](#_Toc476862916)

[4.5 Decomposition Description 8](#_Toc476862917)

[5 .Component Description 10](#_Toc476862918)

[Component Identifier 11](#_Toc476862919)

[Type 11](#_Toc476862920)

[Purpose 11](#_Toc476862921)

[Function 11](#_Toc476862922)

[Subordinates 11](#_Toc476862923)

[Dependencies 11](#_Toc476862924)

[Document Control 13](#_Toc476862925)

# Introduction

## Purpose of the document

The purpose of this document is to provide technical guidance with regards to how the systems functions as whole and how individual components interact with each other. The intended readership of this document is one who has a technical background but shouldn’t be too elusive for the casual reader.

## Scope of the System

The Scope of this system is outlined in DSA Assignment Specification 2016-17.doc which can found on blackboard.

The products that the system should produce are as follows;

* A Database Conceptual ER model
* A Database Logical ER model
* UWE/CEMS MySQL server
* Wine/ region tables
* An Html page displaying two google maps (Google API) each with their own markers.
* Weather API to return weather data to the location shown by the markers.
* XML Schema (XSD) file to validate the configuration file.
* An XML file holding application-wide data such as paths to API's used, API keys (if used), database and other configuration data.
* Group Metadata file (Both individual and group files)

The task of the system is to as outlined in DSA Assignment Specification 2016-17.doc ***“is to build a data driven web application that makes use of external API's and local data storage to encourage interest in Wines, their Types and the Regions that produce them and to display the local weather information for the vineyard***. It should be noted that the RSS as mentioned in section 4 is not included due to time constants.

The system, will be using a MySQL server hosted through the University’s UWE/CEMS server. As such access to the resources used to produces the system is limited to the authors listed at the end of this document. The file *db\_connection.php* is as the name suggests is the file which has the login credentials to connect the database to the main page the values themselves could be switched to easier display data from another dataset/database. Its functions are described about in more depth below

# 2 SYSTEM OVERVIEW

The project files were shared and managed via GIthub the project can be accessed with the following link: <https://github.com/wilkinson001> the owner of the account is team member Oliver Wilkinson who is also hosting the system. The files on Github include both system essential files, support documentation and examples which were used to support creation of the system.

The essential system file i.e. the files present at run time are:

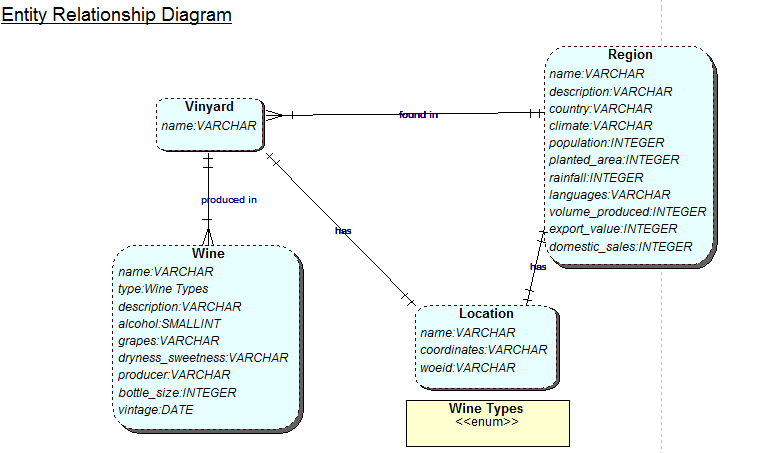
* Index2.2php
  + This is the main HTML file. Here we have the two regions map displaying vineyard markers with location and wine data. It also includes a description of our task, CSS displaying wines and member credits.
* Info2.php
  + This our second page connecting to the main HTML file. It displays a table which pulls data unique to whatever maker was clicked. It also shows the current weather (location based) and forecast for the next five days. Like the main page it also has pictures displayed via CSS.
* db\_connection.php
  + Contains the login credentials for the MYSQL server
* sql.php
  + Function which calls *db\_connection* and get query results from the database.
* setup\_maps.php
  + Another query to setup that maps
* database\_02.sql
  + Creates tables and fills them with the data

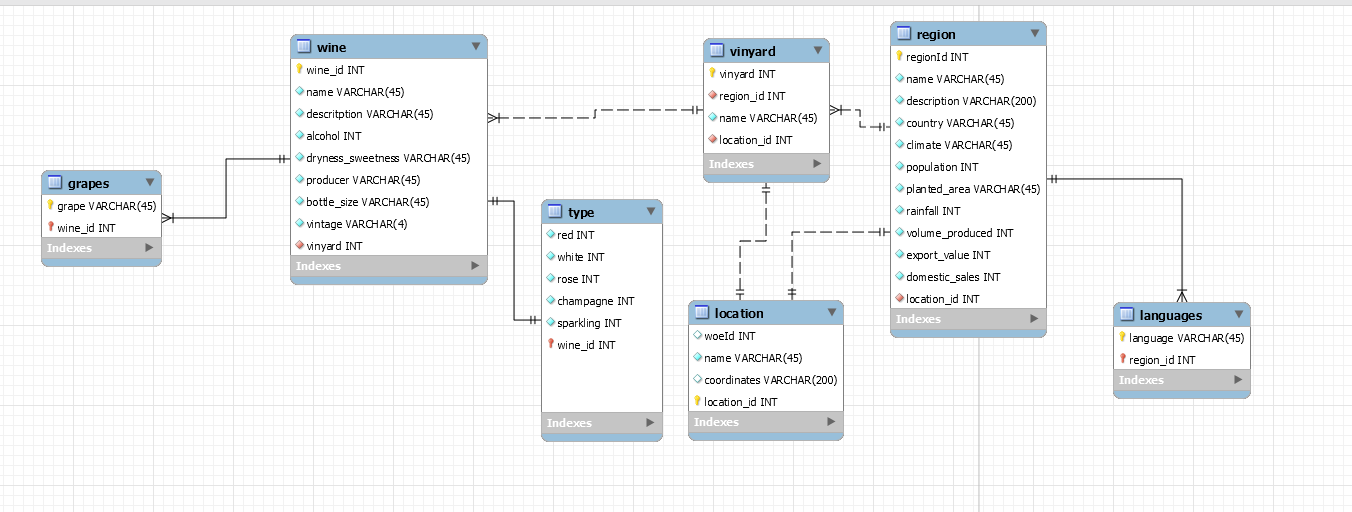
Less critical runtime files include

* logical\_model\_2.mwb
* Assignment-Conceptual ER.qsee
* style.css
* regions.tab
* wines.tab

# 3 Database Design

The following are the two ER diagrams we put together to model how the database will work how elements interact which each other the relationships they have (Many to many)(one to one etc…)

The first model is a simple ER model produced In QSEE showing the abstract view of our main values which we draw from the database. At this stage only simple relationships and variables/data types are given.

The next ER is our logical model displayed in 3rd normal form, it which goes into much more detail with regards to relationships, data types and Keys. This is the graph used for the basis of the MYSQL database.

## 2.1 SYSTEM CHARACTERISTICS

#2 The description of the system should be given in terms of the Architecture of the solution that is being implemented with high level data flows described to set the context of the system, i.e. to look at its external interfaces. This section should also set out to ‘characterise’ the system describing aspects of its operation that indicate if the system has, inter alia:

 to operate in real-time or in bursts, linked to month-end reporting, for example

 the nature of the interface to the users of the system

 many concurrent users

 to be highly resilient or fault tolerant

 to provide security features to protect data

 to be scaleable and easily maintainable in the future

 to have any special back-up facilities to protect important data.

code, functions, configuration, database, administration

# SYSTEM DESIGN

## 4.3 Documentation Standards

1. For a software implementation, this section should contain the standard module header (if necessary) and contain instructions for its completion. In addition this section should define or reference guidelines on the ration of lines of code to comment statements. It may be that these rules highlight specific areas of code where the commentary should be literally line-by-line, as this is a particularly difficult area. Other areas, which may be less difficult, could be commented on a ratio of five lines of code to one line of commentary.

## Naming conventions

1. This section should explain all naming conventions used, and draw attention to any points a maintenance programmer would not expect. A table of the filetypes and the permitted names or extensions for each is recommended for quick reference.
2. Conventions for naming files**,** programs, modules, and possibly other structures such as variables and messages, should all be documented here.

## Programming Standards

1. This section should define the project programming standards. Whatever languages or standards are chosen, the aim should be to create a convenient and easily usable method for writing good-quality software. If an application development tool is used there may be other conventions that need to be defined, e.g. colour schemes.
2. When programming in any new language, a standard for its use should be written to provide guidance for programmers. This standard may be referenced or included here.
3. Where there are external interfaces, the programming standards for the interfaces required should be referenced.
4. In general, the programming standard should define a consistent and uniform programming style. Specific points to cover are:

modularity and structuring;

headers and commenting;

indenting and layout;

library routines to be used;

language constructs to use;

language constructs to avoid.

## Software development tools

1. This section should list the tools chosen to assist software development, including testing. The actual software chosen will be heavily dependent upon the language in which the system will be implemented.
2. The list may include:

an application development too;

a configuration manager / builder;

HTML authoring tools;

a word processor for documentation;

a tool for drawing diagrams;

automated testing tools.

1. Prototyping projects might make use of an interpretative tool, such as an incremental compiler/interpreter/debugger.
2. External interfaces may require some of the modules to be pre-compiled.

## Outstanding Issues

1. Provide details of any design issues that remain unresolved at the date of issue of this document. Explain options, pros and cons, and give an estimate of which option is most likely. Outline impact of each option on the rest of the design.

## Decomposition Description

1. The software components should be summarised. This should be presented as structure charts or object diagrams showing the hierarchy, control flow and data flow between the components.
2. If the UML paradigm is used, then the decomposition description should make extensive use of the nine ‘UML Diagrams’ that in effect define the operation of the system.

# 5 .Component Description

1. For a software implementation, this and the previous section should provide sufficient information for a programmer to produce the software, and for a maintainer, who may not be the developer, to make subsequent changes. The detailed content will depend upon the software tool to be used. The software may be produced using coding statements written by an application programmer. In contrast, it may be automatically generated by an application development tool, or indeed a mixture of both.
2. The descriptions of the components should be laid out hierarchically. There should be subsections dealing with the following aspects of each component:

* 5.n Component identifier
* 5.n.1 Type
* 5.n.2 Purpose
* 5.n.3 Function
* 5.n.4 Subordinates
* 5.n.5 Dependencies
* 5.n.6 Interfaces
* 5.n.7 Resources
* 5.n.8 References
* 5.n.9 Processing
* 5.n.10 Data
* Stuff hetdjei djyeojd fheidj

1. The number 'n' should relate to the place of the component in the hierarchy.

## Component Identifier

1. Each component should have a unique identifier. The identifiers to be used for components should be defined by the project and described elsewhere.

### Type

1. This section should describe the type of component, e.g. task, subroutine, subprogram, package, file.
2. The contents of some component description sections depend on the component type. For the purpose of this template the categories: executable, i.e. contains computer instructions, or non-executable, i.e. contains only data, are used.

### Purpose

1. The purpose of a component should be defined by tracing it to the software requirements that it implements.
2. Backwards traceability depends upon each component description explicitly referencing the requirements that justify its existence.

### Function

1. The function of a component must be defined in this document. This should a short description of what the component does and will depend upon the component type e.g. it may be a description of the process or of the data to be stored or transmitted.
2. More detail will be provided in Processing (see below).

### Subordinates

1. This section should list the modules that are ‘called by’ this component. The subordinates of a database could be the files that ‘compose’ it. The subordinates of an object are the objects that are ‘used by’ it.

### Dependencies

1. The dependencies of a component should be defined by listing the constraints placed upon its use by other components. For example:

* what operations have to have taken place before this component is called?
* what operations are excluded when this operation is taking place?

Document Control

|  |  |
| --- | --- |
| **Title:** | Technical Design Document |
| **Issue:** | Issue 1 |
| **Date:** | 17 January 2001 |
| **Author:** | Dr Dave Sloggett |
| **Distribution:** | EC DG Enterprise – Gavino Murgia Project Team |
| **Reference:** | IDA-MS-TD |
| **Filename:** | IDA-MS-TD-i1 |
| **Control:** | Reissue as complete document only |

Document Signoff

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nature of Signoff** | **Person** | **Signature** | **Date** | **Role** |
| Authors | Dr Dave Sloggett |  |  | Project Member |
| Reviewers | Mark Pillatt |  |  | Consultant |