|  |
| --- |
| ci6300 – INDIVIDUAL PROJECT |
| PROJECT REPORT |
| Daniel Carnovale |
|  |
| **K1336511** |
| **14/03/2017** |

|  |
| --- |
|  |

Table of Contents

[Introduction and Literature Review 3](#_Toc476410872)

[Methodology and Analysis (can be agile based) 3](#_Toc476410873)

[General: 3](#_Toc476410874)

[Details: 3](#_Toc476410875)

[Amazon 3](#_Toc476410876)

[Amazon EC2 3](#_Toc476410877)

[Console Access 3](#_Toc476410878)

[Database Design Strategy 3](#_Toc476410879)

[Physical Database Implementation 3](#_Toc476410880)

[Design 4](#_Toc476410881)

[Implementation 5](#_Toc476410882)

[Discussing How-To steps (Actual steps in appendix?): 5](#_Toc476410883)

[Testing & Evaluation 5](#_Toc476410884)

[Critical Review 5](#_Toc476410885)

[Appendix A 5](#_Toc476410886)

# Introduction and Literature Review

# Methodology and Analysis (can be agile based)

## General:

The methodology I applied led me to the use of forums, help pages on the web and advice from friends and family members to start investigating available technologies and procedures. Considered technologies include GitHub, MySQL, PuTTY, PHP, Python, Ruby / Perl, HTML5, CSS, Javascript, Bootstrap, Angular, JQuery UI, mySQL, postGreSQL, Ansible and Docker, some of which were utilised in the project. The Amazon services EC2, S3, EBS and Elastic IPs were all implemented into the project as utilisation of the wide range of Amazon technologies was the main direction I wanted to head into for this area. I intended to make full use of their “free tier” procedure / offer, though through easy mistakes regarding instance up-time this was unfortunately not achieved.

This was essentially an “agile” approach in that I chose to investigate tools and make quick assessments on which ones to deploy, without performing a very exhaustive research on the merits of each. I continued use of a tool or resource if it could be quickly and easily deployed to achieve an end, however when this did occur I made notes on how each such tool was deployed. This resulted in the recording of sufficient material to turn these notes into fully elaborated How-To’s.

## Details:

### Chose and Bought a Domain Name!

Describe how we did this. => HOW\_TO get a domainname! I still think dad has this in his email..

### Amazon

I started by creating new gmail account to make use of free tier, with a goal to link this to the Amazon services and proceed with creation of the Linux virtual machine, hence leading to the deliverable: “HOW-TO Amazon “Free Tier” Account Creation”.

Theoretically the creation of an Amazon “Free Tier” account should be straightforward, however sometimes the validity of the new account’s purpose must be evaluated which is reasonable. From my personal experience, I had to wait a few weeks as requested though the services did not arrive, and had to contact Amazon support to resolve the issue. It turned out that my Bank company (Barclays) had some trouble accepting the 1 pound authentication procedure initiated by Amazon and therefore progress was stalled in delivery of these services.

After realisation of this problem, I contacted Barclays and resolved the issue on their end, so finally allowed the process of this service delivery to complete. In a standard situation however, there should be no issues regarding the transition between a newly created Amazon account being linked to these services. Overall, I found Amazon support very efficient and informative, and I was able to resolve my problem with minimal effort after following their analysis of the problem.

### Amazon EC2

It was necessary at the beginning of this process to choose an Amazon Machine Image (AMI), as I wanted to investigate how to manage large amounts of data. This led me to experiment with creating an EBS Volume, hence the write-up of the deliverable: “HOW-TO EBS Volumes”.

### Console Access

I discovered that there is normally no “console” access as expected in stand-alone workstations or in VPlayer-type environments.. I quickly realised that I needed to get a bash prompt onto the newly created instanc. This is outlined in detail in the deliverable: “HOW-TO key-pair, putty” and other related HOW-TOs.

### Database Design Strategy

After already gaining very valuable experience from past Database projects, I was quite familiar with the “StarUML” application, so this was my first go-to when attempting to demonstrate or display the database structure. The end result is always an invaluable piece of information that can be improved upon and ultimately be used to construct and enhance the database. My strategy was to use a top-down approach, in that base classes with no foreign key relationships would be placed at the top of the diagram, and other tables or those with foreign key relationships placed further down in the hierarchy.

### Physical Database Implementation

To implement the database based on the diagram created, I made use of the PHPPgAdmin environment. Initially I encountered a confusion re postgres9,91,92,93, etc. though in the end I decided to go with the latest version (95). Later I found serious compatibility problems with phppgadmin, though through the help of some forum posts I was able to fix the SQL error statements shown when attempting functions such as simply creating a new table. At top of the php code simple edits were required, as indicated in this link (paste link). The database creation itself under this environment was fairly straightforward, in that through links on the page I was able to create the new database, create a new table, add its columns (and specify how it should be structured at the same time) and link the foreign tables together. However, the phpPgAdmin environment quickly hit an issue because it’s difficult to recognise foreign key values as just id’s. I realised that I then needed a more visual interface allowing foreign data to be recognised. => HOW-TO ... “xx”.

Got advice on this as a result of which then used open-source “Tables” facility which allowed data to be visualised.

Dump files “Tables” management environment (imported dump file)

.. ( and so on.. a small paragraph or narrative, each one with a L3 Header, describing what led to each HOW-TO! Note that some paragraphs lead to several HOWtos, some possibly lead to dead-ends hence no how-to.

# Design

I made the following design decisions: (“why I went the way I did”)

* Server hosting In House – I chose Cloud because

I chose the Amazon cloud-based environment due to it being a very well known company; offering great customer service and reliability, a free tier program and having a wide range of virtualisation tools and file storage available to its users. The EC2 and S3 services are the most relevant for this project, and when combined can immensely strengthen a virtual Amazon instance. As an additional benefit, through hosting the virtual server in the cloud, it is both accessible anywhere.

* Choice of Cloud Provider (Amazon, Azura, Rackspace etc) Chose Amazon EC2 because
* Choice of Source Control (GitHub vs BitBucket vs Dropbox) chose GitHub as a repository for all screenshots, source code, version control aspects and documentation (issues etc) needs
* Local access to server (Putty because)
* Database (Technology, Management tool PHPPgAdmin)
* OS (Windows VS Linux)
* Choice of Linux distribution (Amazon Linus)
* Security (Approach, Confidentiality, Privash, SSH Key Pair)
* StarUMLClass Diagram
* Backup techniques (test data, use PhpPgAdmin frequently to export dump files, considered Ansible and Terraform etc)
* Apache / PostGres configuration
* Docker consideration
* Encryption (BoxCryptor for Putty Keys) and linkage with Dropbox
* Domain name purchase
* How should I record information? Word .doc or GitHub .md files. Doc is such a standard, though for technical documentation I believe this should be in .md format.
* How to reliably reproduce runtime environment (research suggests that Docker is a strong solution to this, but as yet I have to learn Docker effectively)
* Choice of web-server (Apache, NginX ?) Apache because..
* Hack attempt prevention, disabling clear-text password login access.
* Similarities between Amazon Linux and Centos 6 (why Centos 6.x instructions should be followed)
* Installation of PostGres
* Regular yum update to enforce preventing of vulnerabilities regarding security issues
* Looked into SQL generation from UML but assumed it was very complex / couldn’t find anything / wouldn’t work for my version
* Benefit of public key usage with GitHub
* Considered enhancement of GitHub credentials
* Consideration of static (elastic) instead of dynamic ip
* Documentation of snapshots to accompany steps document
* Use of TortoiseGit to manage local Git repo

# Implementation

## Discussing How-To steps (Actual steps in appendix?):

Appendix A is the primary deliverable of this project work, and consists of a series of HOW-TO’s. These are designed to be simple, practical and readable instructions that allow a particular technical objective to be achieved, and yet do not assume a high existing level of expertise to understand and execute.

# Testing & Evaluation

* Succeded in getting a web app up and running? (self assessment.. I reckon I did)
* (put the how-tos in front of somebody else! See if they can follow them !)

# Critical Review

Was my approach successful? Will anyone really benefit from my How-Tos?

Which design decisions would I do differently?

# Appendix A

(Append all How-To’s here, so they’re shown in contents section)