|  |
| --- |
| Name: J.A. Mujeeb |
| Student Reference Number: 10707284 |



|  |  |  |
| --- | --- | --- |
| Module Code: CNET343SL | Module Name: Distributed Systems | |
| Coursework Title: Weather Reporting System Proposal | | |
| Deadline Date: 03/12/2021 | | Member of staff responsible for coursework: Mr. Pramudya Thilakaratne. |
| Programme: BSc (Hons) Plymouth Software Engineering | | |
| Please note that University Academic Regulations are available under Rules and Regulations on the University website [www.plymouth.ac.uk/studenthandbook](http://www.plymouth.ac.uk/studenthandbook). | | |
| Group work: please list all names of all participants formally associated with this work and state whether the work was undertaken alone or as part of a team. Please note you may be required to identify individual responsibility for component parts.  J.A. Mujeeb – 10707284  G.M.D.D. Ratnayake – 10707351  S.O. Perera – 10707315  N. S. De Alwis – 10707160  M. D. A. Medhavi – 10707278  P. P. L. Dilhani – 10709402  ***We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.***  Signed on behalf of the group: J.A. Mujeeb | | |
| Individual assignment: ***I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.***    Signed: | | |
| Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence.  I \*have used/not used translation software.  If used, please state name of software………………………………………………………………… | | |
| **Overall mark \_\_\_\_\_% Assessors Initials \_\_\_\_\_\_ Date\_\_\_\_\_\_\_\_\_** | | |

* \*Please delete as appropriateSci/ps/d:/students/cwkfrontcover/2013/14

Contents

[2. Acknowledgement 5](#_Toc102987585)

[3. Introduction to the Distributed System 5](#_Toc102987586)

[4. Introduction to Quick Weather – 5](#_Toc102987587)

[5. Usefulness of the application to the real-world 6](#_Toc102987588)

[6. Scope of Project 6](#_Toc102987589)

[6.1 Diagrams 6](#_Toc102987590)

[6.1.1 System Diagram 7](#_Toc102987591)

[6.1.2 Context Diagram 7](#_Toc102987592)

[6.1.3 Use Case Diagram 8](#_Toc102987593)

[7. Mobile Application as Client 8](#_Toc102987594)

[8. Methodology 8](#_Toc102987595)

[9. Database: PhpMyAdmin MySQL 8](#_Toc102987596)

[10. Gantt Chart 9](#_Toc102987597)

[11. Selection of Middleware 9](#_Toc102987598)

[12. Technical Diagram 9](#_Toc102987599)

[12.1 Overview of Technical Diagram 9](#_Toc102987600)

[12.2 Technical Architectural Diagram 10](#_Toc102987601)

[12.2.1 Overview of Technical Diagram 10](#_Toc102987602)

[12.2.2 Middleware of Technical Diagram 10](#_Toc102987603)

[12.2.3 Technical Architectural Diagram 10](#_Toc102987604)

[13. Load Balancing 11](#_Toc102987605)

[13.1 With Load Balancing 11](#_Toc102987606)

[13.2 Without Load Balancing 11](#_Toc102987607)

[14. Tolerance to Network Failure 12](#_Toc102987608)

[15. System architectural Diagram 12](#_Toc102987609)

[15.1 User Requirements 12](#_Toc102987610)

[15.2 Functional Requirements – 12](#_Toc102987611)

[16. API / Middleware Justification (Nilhan) 12](#_Toc102987612)

[17. Software components used 12](#_Toc102987613)

[18. Distributing the system 13](#_Toc102987614)

[19. Issues faced and actions taken 13](#_Toc102987615)

[20. Development phases 13](#_Toc102987616)

[21. Tasks undertaken 14](#_Toc102987617)

[21.1 Postman API requests and testing 14](#_Toc102987618)

[21.1.1 Add Post Request 14](#_Toc102987619)

[21.1.2 Login Request 15](#_Toc102987620)

[21.2 Screenshots of the website 15](#_Toc102987621)

[21.3 Screenshots of the Mobile Application 16](#_Toc102987622)

[21.4 Screenshots of the Admin panel 17](#_Toc102987623)

[21.5 Main codes of the website 17](#_Toc102987624)

[21.6 Main codes of the mobile application 17](#_Toc102987625)

[21.7 Main codes of the API 17](#_Toc102987626)

[21.7.1 Controller 18](#_Toc102987627)

[21.7.2 APIs 20](#_Toc102987628)

[22. Future enhancement of the system 27](#_Toc102987629)

[23. Risk assessment – 28](#_Toc102987630)

[24. Quality plan 28](#_Toc102987631)

[25. Summary - 28](#_Toc102987632)

[26. Individual contribution – 28](#_Toc102987633)

[Figure 1: System Diagram 7](#_Toc102987553)

[Figure 2: Context Diagram 7](#_Toc102987554)

[Figure 3: Use Case Diagram 8](#_Toc102987555)

[Figure 4: Gantt Chart (Part 1) 9](#_Toc102987556)

[Figure 5: Gantt Chart (Part 2) 9](#_Toc102987557)

[Figure 6: Overview of Technical Diagram 9](#_Toc102987558)

[Figure 7: Overview of Technical Diagram 10](#_Toc102987559)

[Figure 8: Middleware of Technical Diagram 10](#_Toc102987560)

[Figure 9: Technical Architectural Diagram 10](#_Toc102987561)

[Figure 10: With Load Balancing 11](#_Toc102987562)

[Figure 11: Without Load Balance 11](#_Toc102987563)

[Figure 12: System Architectural Diagram 12](#_Toc102987564)

[Figure 13: Controller.php 18](#_Toc102987565)

[Figure 14: PostHasImagesController.php 18](#_Toc102987566)

[Figure 15: UserController.php part 1 19](#_Toc102987567)

[Figure 16: UserController.php part 2 19](#_Toc102987568)

[Figure 17: UserController.php part 3 20](#_Toc102987569)

[Figure 18: Api.php part 1 20](#_Toc102987570)

[Figure 19: Api.php part 2 21](#_Toc102987571)

[Figure 20: Channels.php 21](#_Toc102987572)

[Figure 21: Console.php 22](#_Toc102987573)

[Figure 22: Web.php 22](#_Toc102987574)

[Figure 23: Authenticate.php 23](#_Toc102987575)

[Figure 24: EncryptCookies.php 23](#_Toc102987576)

[Figure 25: Preventrequestduringmaintenance.php 24](#_Toc102987577)

[Figure 26: Redirectifauthenticated.php part 1 24](#_Toc102987578)

[Figure 27: Redirectifauthenticated.php part 2 25](#_Toc102987579)

[Figure 28: TrimStrings.php 25](#_Toc102987580)

[Figure 29: TrustsHosts.php 26](#_Toc102987581)

[Figure 30: Trustproxies.php part 1 26](#_Toc102987582)

[Figure 31: trustproxies.php part 2 27](#_Toc102987583)

[Figure 32: Verifycsrftoken.php 27](#_Toc102987584)

[Table 1: Methodology 8](#_Toc102982065)

[Table 2: Issues faced, and actions taken 13](#_Toc102982066)

[Table 3: Development Phases 14](#_Toc102982067)

[Table 4: Risk Assessment 18](#_Toc102982068)

[Table 5: Quality Plan 18](#_Toc102982069)

# Acknowledgement

First and foremost, we’d like to extend our sincere gratitude towards Mr. Pramudya Thilakaratne, our module lecturer. We are extremely humbled and grateful to have been able to receive his mentorship, guidance, and support.

The overall accomplishment of this project demanded a significant amount of guidance from many individuals. As a team, we are extremely fortunate to have had this from start to finish.

Finally, we wouldn’t have been able to successfully complete this assignment without the hard work and assistance of all the team colleagues itself. We all enjoyed working with each other.

# Introduction to the Distributed System

The use of weather reporting in day-to-day life is very important. Its utilization could greatly influence the outcome of a scenario. It could be something as simple as deciding whether you should take your umbrella on your way out, or even as major as handling cultural operations, agriculture and farming or livestock protection implementations.

With Sri Lanka’s weather rapidly varying, weather can transition from largely homogeneous temperatures to torrential rain in a heartbeat. With the adaptation of unexpectedly frequent changes, it is crucial to make sure to be prepared.

Sri Lanka is liable to prevailing and predictable effects of climate change. Preceding natural disasters (such as tsunamis, floods, landslides, droughts, and cyclones) can greatly substantiate this. Profoundly weather-sensitive sectors in Sri Lanka include transport, agriculture, construction, energy, and disaster risk management.

It is important to note that even with weather stations, live broadcasts, and television/radio stations existent, not all are aware of the continuously differentiating weather conditions.

The impacts and effects of climate change could invite climate change-induced hazards and disasters. The unawareness of weather reports may affect the lives of many. Thousands of citizens are prone to being affected by the threat of climate change. They may find themselves in life threatening situations. Weather prediction is essential in order to provide citizens with pragmatic information. Furthermore, this also aids in the reduction of weather-related losses, personal safety and health, enhancement of societal benefits and in supporting economic prosperity.

It is abundantly clear that weather forecasting reports are essential to mitigate the effects civilians getting caught in a severe crisis.

# Introduction to Quick Weather –

**What is distributed system?**

* A distributed system is a collection of independent computers that appears to its users as a single coherent system. (Tanenbaum)
* We have a system or application that could be a software. In that particular system we have a number of independent computers (individual computers inside the system). These individual computers are doing their own task in the system, but for the end user this looks like one single system. Those computers are doing their own tasks but for the user, they can see everything as one system.

A distributed system is contrived to assist the development process of services and applications. These services and applications are able to manoeuvre a physical architecture that would contain multiple independent processing elements. The processing elements do not share primary memory but do partake in complying with dispatching asynchronous messages via communication network.

Furthermore, Distributed Systems also utilize independent hardware in furtherance of creating a software. It would appear as a complete system all working as a single unit, but, it makes use of multiple computers in contemplation of making a distributed system function.

The primary memory of other hardware elements is not employed by Distributed Systems. Instead, multiple independent processing elements within the hardware are utilized. Using asynchronous messages, they can communicate over a networked communication.

In the following report, our team would like to highlight the main outcome of our system.

To make the reader of this report get a clear idea, we have simply built a web application and a mobile application that will be running as a client application.

People often need to know what the weather around them would be like, but since Sri Lanka is still a developing country, it is only specified in either the news or rather in the radio stations. A weather would mainly focus on the current location of the user, but with Quick Weather, any user can check the weather around the country without having to go through a hassle of signing into the system.

Our main intention is to build a system that will show our users the needed weather for any planned activities in their personal lives.

# Usefulness of the application to the real-world

Quick weather can perform 02 main functions

* Users can check weather of any locations and users can do so without having to go through a huge hassle, as it does not require users to register into the system, users can access it without delays. The user simply has to enter the city name where they want to search the weather in and type it in to access weather details.
* Users can view posts about weather from the web application (ask about desktop), and these updates are done by other people as well.

# Scope of Project

The following is a list of the application's scope:

* This program can predict the state of the atmosphere for a specific place. Rain, cloudiness, wind speed, and humidity are all factors.
* way to get weather predictions, warnings, and other useful information on our phones in real time
* By checking at the system's post, it was possible to take precautionary steps against catastrophic rains, winds, extreme high or low temperatures, diseases, and pests.

## Diagrams

### System Diagram

****

Figure 1: System Diagram

### Context Diagram

****

Figure 2: Context Diagram

### Use Case Diagram

****

Figure 3: Use Case Diagram

# Mobile Application as Client

Mobile application is created using Android Studio, the language that Android Studio uses is java. Since Android Studio has SDKs that run emulators, it is rather quick and easy to use. The basic uses that the client can do is:

The client can login/signup and post any type of image related to Technologies used:

* Laravel
* Java in web and mobile application both
* MySQL for the database

# Methodology

|  |  |  |
| --- | --- | --- |
| keyword | category | description |
| Java NetBeans | IDE |  |
| Laravel | Framework |  |
| Android Studio | IDE |  |
| MySQL | IDE |  |

Table 1: Methodology

# Database: PhpMyAdmin MySQL

As PHP is utilized to code the Web Application, it was more coherent to utilize PhpMyAdmin for MySQL database management. PhpMyAdmin is an open-source third-party software tool, written in PHP. The primary objective of utilizing phpMyAdmin is to superintend and operate the administration of MySQL over the web. It is also feasible to run CRUD operations like databases, copy, tables, rename, databases, tables, columns, etc. phpMyAdmin can run on any server or any OS as it has a web browser. With the utilization of phpMyAdmin, it is possible to edit, create or delete the database without much difficulty. In collation to the MySQL command-line editor, it is easier to manage elements with the utilization of the phpMyAdmin graphical interface. Several servers can also be operated simultaneously. Data can also be exported into various formats like Word, PDF, SQL, XML, Spreadsheet, etc.

# Gantt Chart

Timeline

Description automatically generated with medium confidence

Figure 4: Gantt Chart (Part 1)

Chart

Description automatically generated

Figure 5: Gantt Chart (Part 2)

# Selection of Middleware

# Technical Diagram

## Overview of Technical Diagram

****

Figure 6: Overview of Technical Diagram

## Technical Architectural Diagram

### Overview of Technical Diagram

****

Figure 7: Overview of Technical Diagram

### Middleware of Technical Diagram

Diagram

Description automatically generated

Figure 8: Middleware of Technical Diagram

### Technical Architectural Diagram

****

Figure 9: Technical Architectural Diagram

Our system contains a mobile app and a web app, and their total processing is made more efficient by a load balancer, which divides incoming network traffic across a cluster of three dispersed servers.

# Load Balancing

A load balancer sits in front of your servers, acting as a "traffic cop," directing client requests across all servers capable of satisfying those requests in a way that maximizes performance and reformulation testing while ensuring that no single server is overworked, potentially degrading performance. The load balancer transfers traffic to the remaining online servers if a single server goes down. When a new server is added to a server group, the load balancer begins sending requests to it automatically.

A load balancer accomplishes the following tasks in this manner:

* Client requests or network load are efficiently distributed among numerous servers.
* Sends requests exclusively to online servers, ensuring high availability and reliability.
* Provides the flexibility to add or remove servers as need dictates.

## With Load Balancing

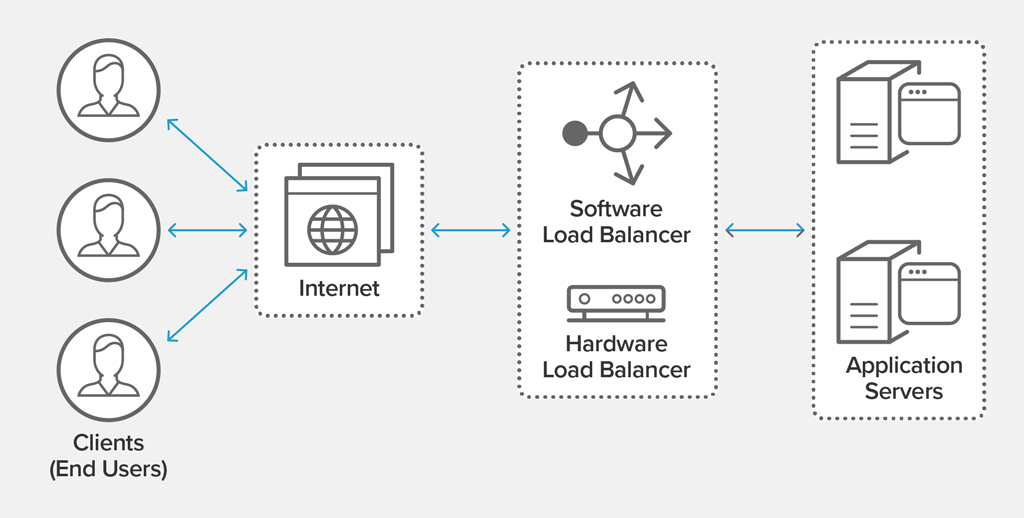


Figure 10: With Load Balancing

## Without Load Balancing

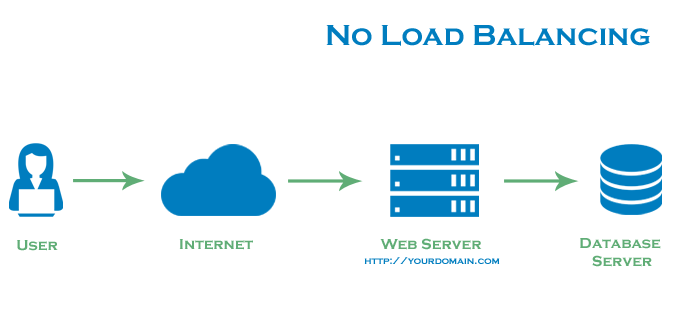


Figure 11: Without Load Balance

# Tolerance to Network Failure

As our system run in 3 servers beside the load balancer, just in case of a network failure it automatically masses to the backup server, thus no network failures can influence the system function.

# System architectural Diagram

****

Figure 12: System Architectural Diagram

## User Requirements

The Graphical User Interface system is designed in a way where the user can:

* Users can check weather without having to sign up/sign into the system.
* The users just must enter the city name of the preferred place where they want to know the weather of.
* The weather will be displayed in an instance for the user to view.
* Users also can check posts that are posted by other members of society.

## Functional Requirements –

* We tend to run multiple copies of each element on totally different machines.
* Replicate data so single machine failure doesn't cause loss of data. Continues to figure.
* The system runs elements on separate machines.
* Has redundancy at all layers.

# API / Middleware Justification (Nilhan)

# Software components used

* Laravel – Laravel was used to create the API for the system, this is a free, open-source web framework that is used for web application development. It follows a model-view-controller design pattern.
* MySQL – Open-source relational database management system, it uses Structured Query Language, one of the most famous languages for managing content in a database.
* Android Studio – Android Studio is a mobile application development platform; Android Studio has their own emulators which makes it easy for developers to use and maintain. Android studio has fast execution, which provides time effectives for developers.
* NetBeans – integrated development environment for Java. NetBeans was used to create our stand-alone application.

# Distributing the system

* Distributed File System

# Issues faced and actions taken

|  |  |
| --- | --- |
| **Issues Faced** | **Actions Taken** |
|  |  |

Table 2: Issues faced, and actions taken

# Development phases

|  |  |
| --- | --- |
| Development phases | Description |
| 1. Planning | Defining the project's boundaries in terms of how large the scope will be. Creating a project plan, calculating how long it will take to finish the work, and breaking it down into pieces, or sequencing the work, and creating an initial project schedule |
| 2. Analysis | Analysing the system's critical features and requirements to determine whether it is worthwhile to develop. |
| 3. Designing | Designing aids in determining the shape of what should be constructed. It functions as a template for how the implementation should proceed. Includes thorough software architecture, database, and frontend designs**.** |
| 4. Coding and debugging | This includes database development, website creation, and the implementation and development of backend logic, as well as frontend development. In the same way, a mobile application with implemented logic and a frontend is generated. In addition, an API to interface with the MySQL database has been developed. |
| 5. Testing | Functional testing is done to determine if we are obtaining the expected outcomes. Every unit of code in our project was tested as part of the unit testing process. Integration Testing was conducted after the system's modules were integrated to determine if they worked together. The complete system was then put through End-to-End testing. User Interface Testing, which entails testing the application's user interface, was carried out to see if the UIs were designed in accordance with the system's specifications. Load Testing was conducted as part of non-functional testing to enhance the system's quality. It checked the system's capacity to take load. |
| 6. Deliver the product | Deliver the website, mobile application and report**.** |

Table 3: Development Phases

# Tasks undertaken

## Postman API requests and testing

Graphical user interface, text, application

Description automatically generated

### Add Post Request

Graphical user interface, text

Description automatically generated

### Login Request

A screenshot of a computer

Description automatically generated with medium confidence

## Screenshots of the website

The web application as shown below is the interface the user is greeted with when

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

## Screenshots of the Mobile Application

This is the interface the client is greeted with when he/ she is in need of creating an account to gain privileges of posting updates on the weather post page on the web page of this distributed application.

A screenshot of a phone

Description automatically generated with low confidence

After the user creates an account, he or she can login through the app to see the weather-related posts that have been submitted as shown below

Graphical user interface, text

Description automatically generated with medium confidence

This is the user login page where he or she enters their email and password to login to the app to view the posts.

A screenshot of a phone

Description automatically generated with low confidence

## Screenshots of the Admin panel

## Main codes of the website

## Main codes of the mobile application

## Main codes of the API

### Controller

A screenshot of a computer

Description automatically generated

Figure 13: Controller.php

A screenshot of a computer

Description automatically generated

Figure 14: PostHasImagesController.php

A screenshot of a computer

Description automatically generated

Figure 15: UserController.php part 1

A screenshot of a computer

Description automatically generated

Figure 16: UserController.php part 2

A screenshot of a computer

Description automatically generated

Figure 17: UserController.php part 3

### APIs

A screenshot of a computer

Description automatically generated

Figure 18: Api.php part 1

A screenshot of a computer

Description automatically generated

Figure 19: Api.php part 2

A screenshot of a computer

Description automatically generated

Figure 20: Channels.php

A screenshot of a computer

Description automatically generated

Figure 21: Console.php

A screenshot of a computer

Description automatically generated

Figure 22: Web.php

A screenshot of a computer

Description automatically generated

Figure 23: Authenticate.php

A screenshot of a computer

Description automatically generated

Figure 24: EncryptCookies.php

A screenshot of a computer

Description automatically generated

Figure 25: Preventrequestduringmaintenance.php

A screenshot of a computer

Description automatically generated

Figure 26: Redirectifauthenticated.php part 1

A screenshot of a computer

Description automatically generated

Figure 27: Redirectifauthenticated.php part 2

A screenshot of a computer

Description automatically generated

Figure 28: TrimStrings.php

A screenshot of a computer

Description automatically generated

Figure 29: TrustsHosts.php

A screenshot of a computer

Description automatically generated

Figure 30: Trustproxies.php part 1

A screenshot of a computer

Description automatically generated

Figure 31: trustproxies.php part 2

A screenshot of a computer

Description automatically generated

Figure 32: Verifycsrftoken.php

# Future enhancement of the system

We as a group would love to upgrade our project if we are to continue this project, this project is to be used by any person who has access to the internet. With this system people can focus and entail in their safety.

We would like to add features were the user can live forecasts and, we would love to add reports from weather stations into the systems, were people can be updated.

We would also like to enhance the project more with checking for weather patterns and collecting data for the betterment of human life.

# Risk assessment –

|  |  |
| --- | --- |
| Risk | Management Strategy |
| Data loss/technical failure | Backup and copy on a storage device. |
| Overrun on the schedule | The project plan now includes a contingency plan. Highlights and supervisor meetings will be used to track and monitor progress. More days have been added to the estimated completion dates for each step. |
| Complexity of the project | To adhere to a set of rules that would allow the basic functionality to be implemented and a working product to be produced. |
| Problems with the technology needed to complete the project | A basic prototype will be constructed for each phase to examine and validate validity. Maintaining a record of each stage. |
| Other requirements | There will be other projects operating concurrently, and this will receive full attention. A time management strategy would assist me in staying on track. |
| Lack of information on this topic | * Before starting, do a feasibility study on the topic. * Read books, essays, etc. on the topic. * Research on the internet. |
| Bad Timing | * Plan, then stick to it. * You must determine which project stages or components are most important to you, as well as the short, medium, and long-term impact of each stage/component. |
| Poor Code Quality | * Review of the code * Coding rules and guidelines that are easy to understand * All code is tested. * The Method of Work. |

Table 4: Risk Assessment

# Quality plan

|  |  |
| --- | --- |
| Unit Testing | Separate platforms and code blocks were tested in debug mode, bug fixing, and problems were resolved |
| Integrated Testing | In unit testing as shown with the screen shots of the working application the issues faced with the collaborative workload of the application were fixed and resolved |

Table 5: Quality Plan

# Summary -

# Individual contribution –

* J.A. Mujeeb (10707284) -
* G.M.D.D. Ratnayake (10707351) -
* S.O. Perera (10707315) -
* N. S. De Alwis (10707160) -
* M. D. A. Medhavi (10707278) -
* P. P. L. Dilhani (10709402)