Technical manual

Documentation for the Emma Project

Table of Contents

[Web Portal Set Up 2](#_Toc450296569)

[LAMP 2](#_Toc450296570)

[Network Security Groups 2](#_Toc450296571)

[WinSCP 3](#_Toc450296572)

[Web Portal Files 4](#_Toc450296573)

[Emma Root folder 4](#_Toc450296574)

[CSS sub-folder 5](#_Toc450296575)

[JS sub-folder 5](#_Toc450296576)

[User\_query sub-folder 6](#_Toc450296577)

[User\_profile sub-folder 8](#_Toc450296578)

[Documentation sub-folder 8](#_Toc450296579)

[Query Queue Logic 9](#_Toc450296580)

[Query Status Values 9](#_Toc450296581)

[New Queries 9](#_Toc450296582)

[Changing the Query Status 9](#_Toc450296583)

[Query Lifecycle 9](#_Toc450296584)

[Database Schemas 10](#_Toc450296585)

[ER Diagram 10](#_Toc450296586)

[Relational Schema 10](#_Toc450296587)

[SQL Views 11](#_Toc450296588)

[SQL Create Table Statements 11](#_Toc450296589)

[SQL Insert Statements 13](#_Toc450296590)

[Data Flow Diagram 14](#_Toc450296591)

[Use Cases 15](#_Toc450296592)

[Reference Information 16](#_Toc450296593)

[AWS EC2 Instance 16](#_Toc450296594)

[WinSCP Credentials 16](#_Toc450296595)

[Emma Database through phpMyAdmin 16](#_Toc450296596)

[Emma Web Portal 17](#_Toc450296597)

[Database Connection Credentials 17](#_Toc450296598)

# Web Portal Set Up

The Emma Web Portal was hosted using Amazon Web Services on an Amazon Linux AMI EC2 instance. The following AWS tutorials were used to launch a virtual machine, connect to the Linux instance, configure the EC2 instance, install the Apache web server, PHP and MySQL software packages (LAMP stack) onto the instance and start the web server.

<https://aws.amazon.com/getting-started/tutorials/launch-a-virtual-machine/>

<http://docs.aws.amazon.com/gettingstarted/latest/wah-linux/getting-started-deploy-app.html>

## LAMP

The LAMP tutorial below outlined how to properly set the file permissions on the server so that the ec2-user can create and modify the files and directories when developing and maintaining the web portal. phpMyAdmin was also installed on the web server to allow the developers to view and edit the MySQL databases through a web-based management tool. As the development team needed to access phpMyAdmin from various development machines with different IP addresses, the phpMyAdmin installation required configuration allow access from multiple local machines. For phpMyAdmin credentials, see the Reference Information section. For security purposes, please remove the 0.0.0.0 non-routable meta-address and replace with the future developers local IP addresses.

<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/install-LAMP.html>

## Network Security Groups

Traffic to the instance was controlled through the use of Security Groups.

<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/authorizing-access-to-an-instance.html>

The recommended rules for a Web Server Security Group found in the following user guide were used:

<http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Scenario3.html#SecurityGroups-3>

As the development team needed inbound SSH access to the Linux instance from various development machines with different IP addresses, the Web Server Security Group required configuration allow access from multiple local machines. For security purposes, please edit the inbound rules for WebServerSG of type SSH and RDP so that their source IP address matches the local IP addresses of future developers and not the non-routable meta-address 0.0.0.0.

## WinSCP

To access the Web Portal source code files on the AWS instance, WinSCP was used. To set up the GUI-based file manager, the following AWS tutorial was used:

<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html>

For WinSCP credentials, see the Reference Information section.

# Web Portal Files

## Emma Root folder

This folder contains all the files that were used to display the Emma login page and main page to the agent before a user is selected from the left queue pane.

**Settings.php**

Contains the variables for the host name, user name, password and SQL database name.

**Connect.php**

Contains the commands for connecting to the database, selecting the database and starting the session.

**Index.php**

The first page (login screen) that the Agent sees when they load up the Emma web portal. It connects to the database and checks that the Agent’s email address and password match a row in the User table and sets the UserId and UserName session variables.

**Main.html**

The main page that forms the basis onto which the rest of the web portal is built. It contains the div elements for navigation bar and the left queue pane. On first loading, when a user has not yet been selected, a message in the central main display asks the Agent to select a user from the queue.

**Logout.php**

The php script that runs when an Agent logs out of the Emma web portal. If the Agent has any assigned queries that have been unanswered, the script automatically sets these queries to open and unassigned.

**Get\_agent\_name.php**

This php script retrieves the Agent’s name from the database according to the session variable set when the Agent logged into the Emma web portal and provides a visual indication of who is logged into the system.

**Getqueue.php**

This php script retrieves all the open queries from the query table, groups them according to UserId and displays them in the left queue pane under “Open Users”. It also checks the releaserequest table for the UserId of the agent that is logged in to insert the script that alerts the agent that they are holding multiple users.

**Getassigned.php**

This php script retrieves all the unanswered queries from the query table that are assigned to the logged in agent, groups them according to UserId and displays them in the left queue pane under “Assigned Users”. When an agent clicks on a user with “Open” queries, it assigns all of these queries to the agent and marks them as “Assigned”. Refer to section Query Queue Logic.

This script also checks if there are any open or assigned users available for the logged in agent to select. If not, it inserts buttons that the agent can use to alert the agent with the most assigned users or all agents with multiple users.

**Releaserequest.php**

This php script is executed when an agent with no assigned users alerts other agents with multiple users by finding these agents in the multiuserholder view and inserting rows into the releaserequest table.

**Checksession.php**

This script checks for a username session variable and triggers an alert in the main.js file if the main.html page has been loaded without an agent logging into the system and redirects them to the index.php login page.

### CSS sub-folder

**Style.css and Custom.css**

The Cascading Style Sheets used for presentation of the web portal were separated from the content of the web portal to improve content accessibility and allow for greater control and flexibility of design characteristics.

### JS sub-folder

**Main.js**

Contains all the functions used to add functionality the web portal such as rendering the Main Display, retrieving the Agent’s name from the database, loading the unanswered open and assigned queries from the database, displaying user queries in different formats, displaying results from Google Search and Maps, displaying user profile information and alerting agents who have multiple assigned users.

**Jquery-2.2.3.min.js and Jquery-2.2.3.js**

jQuery library features such as selecting DOM elements and handling click events were used in various functions in the Main JavaScript file.

## User\_query sub-folder

This sub-folder contains all the files that were used to render the main central display to the agent after a user is selected from the left queue pane.

**Get\_one\_query.php**

This is the main php script that brings together the different elements of the “User Query” by retrieving from the database a single query for a specific user and including all the other relevant php scripts: check\_assigned.php, navigation.php, query\_content.php and query\_result.php. On initial loading of the “User Query” tab, the query that is retrieved is the oldest unanswered query for the specific user. A different query can be displayed when selected from the “Unanswered Queries” or “Query History” tabs.

**Check\_assigned.php**

When the agent clicks on a user with open queries in the left pane, this php script changes the status of these queries to assigned and assigns them to the logged in agent. Then it checks that the unanswered user queries to be displayed are assigned to the agent that is logged in. This aims to prevent synchronisation errors that may occur when two different agents click on the same user at the same time.

**Navigation.php**

This php fragment renders the “User Query”, “User Profile”, “Unanswered Queries” and “Query History” navigation tabs once a user is selected from the left queue panel.

**Query\_content.php**

This php fragment renders the top half of the “User Query” display containing the content of a single query, the text and (if available) audio and image files. Also included are forms for editing the text content of the query, submitting a text response to the query and a button for marking all of the Assigned queries for the user as Open.

**Query\_result.php**

This php fragment renders the bottom half of the “User Query” display containing the result from when the query is input into Google Search and when the user’s location is input into Google Maps. Clicking on the headings of these elements will open the results in a new browser tab.

**Loading.gif**

The loading image that is displayed before the Google Search results are retrieved.

**Searchgoogle.php**

This php script retrieves the search results from Google. This acts as a workaround to bypass the security mechanism of JavaScript’s Same Origin Policy.

**Query\_action.php**

This php file contains the scripts for manipulating the query table in the database to fix the text content of a query, to submit an answer to a specific query and to mark all assigned queries for a specific user as open.

**Get\_unanswered.php**

This php script displays up to 10 of the user’s unanswered queries in the “Unanswered Queries” tab from oldest to newest. Clicking on the links under the “Query”, “Location” and “Image” columns opens additional information in a new browser tab. Clicking on the “Time” of a query displays this specific query in the “User Query” tab. A form for submitting an answer to a specific query is also provided. The “Unanswered Queries” tab is used to isolate out even very old unanswered queries.

**Get\_history.php**

This php script displays up to 20 of the user’s queries in the “Query History” tab from oldest to newest. It provides all the same functionality as the get\_unanswered.php script but it shows a full history of 20 chronological queries regardless of whether they have been answered or not.

**Processmessage.php**

This php script is included in the query\_action.php script when an answer is posted for a specific query. It contains functions for retrieving the registration ID of a specific user and sending the answer as a JSON response string to the Google Cloud Messaging server which is then sent as a push notification to the user’s device.

## User\_profile sub-folder

This sub-folder contains all the files that were used to render the “User Profile” tab in the main central display after a user is selected from the left queue pane.

**Get\_user\_profile.php**

This php script retrieves all of the available profile information for a specific user from the User table in the database and displays it in a table on the “User Profile” tab. Any of the information fields that can be modified are rendered as form inputs. Each of these are validated using HTML5 validation attributes according to the input type.

**User\_action.php**

This php script submits the form inputs to the database and updates the User table.

## Documentation sub-folder

**Technical Manual.docx**

This document.

**Agent Manual.docx**

The guide for agents using the Emma Web Portal.

**AWS settings.php**

The settings PHP script used to connect to the database on the server when moving development files from the local machine to the server.

**Insert Multi Rows.xlsx**

An Excel spreadsheet used to quickly create multiple SQL insert queries for populating rows of sample data into tables for testing purposes.

**Emma.sql**

A backup of SQL database for easy data migration across web hosts.

**Entity Relationship Diagram.vsd**

The relationships between the entities identified in the Emma project.

**Data Flow Diagram.vsd**

The data flows between the user, the system and the database tables by use case.

**Use Cases.xlsx**

The use cases identified in the Emma project by event.

# Query Queue Logic

## Query Status Values

The status of a query can be one of three integer values:

0 – Open: the query has not been assigned to any agent and is visible in the “Open Users” queue on the left pane of the web portal.

1 – Assigned: the query has been assigned to a specific agent but has not been answered and is visible to this agent in the “Assigned Users” queue on the left pane or in the “Unanswered Queries” and “Query History” tab on the main display.

2 – Closed: the query has been answered and can only be viewed in the “Query History” tab on the main display.

## New Queries

For each new query, the database is checked and if any queries for a particular user has a status of 0, the new query status is set to 0 and assignedto is set to NULL.

If any queries for a particular user has a status of 1, the new query is set to 1 and assignedto is set to agent’s userid.

If all queries for a particular user have a status of 2, the new query is set to 0 and assignedto is set to NULL.

## Changing the Query Status

When an agent selects a user with open queries, the status of all open queries is set to 1 and assignedto is set to the agent’s userid.

If an agent cannot answer all queries (e.g. the agent is finishing work and a different agent needs to take over), the agent can set the status to 0 and assignedto to NULL for all assigned queries of a specific user by clicking on the “Mark all Assigned queries as Open”. This is automatically done for all assigned users when the agent logs out of the web portal.

When an agent answers one query, the status of the query is set to 2 and the answer and answertime is updated in the database.

## Query Lifecycle

Queries can move along the status lifecycle in both directions but only one step at a time e.g. from 0 to 1 and back to 0 or from 0 to 1 to 2.

Once a query is closed, it cannot be changed back to open or assigned.

# Database Schemas

## ER Diagram



## Relational Schema

USER (UserID, Password, RegID, Name, Email, Phone, DOB, Role, DateJoined)

QUERY (QueryID, UserID, Time, Location, Content, Audio, Image, Answer, AssignedTo, AnswerTime, Status) Foreign Key UserID References USER

RELEASEREQUEST (AgentID, RequestSentBy, RequestDate) Foreign Key RequestSentBy References USER

WOPTABLE (UserID, AssignedTo)

MULTIUSERHOLDER (AssignedTo, NoOfAssignedUser)

## SQL Views

In order to provide the functionality of alerting agents that have multiple assigned users to release one or more users, two Views were created in the Emma database. The WOPTABLE View monitored which user was assigned to which agent while the MULTIUSERHOLDER View dynamically tallied the number of users that were assigned to a specific agent.

## SQL Create Table Statements

CREATE TABLE USER (

UserID INT(11) NOT NULL AUTO\_INCREMENT ,

Password VARCHAR(20) ,

RegID VARCHAR(255) ,

Name VARCHAR(100) ,

Email VARCHAR(100) ,

Phone VARCHAR(11) ,

DOB DATE ,

Role INT(1) ,

DateJoined DATE ,

CONSTRAINT PK\_USER PRIMARY KEY (UserID)

);

CREATE TABLE QUERY (

QueryID INT(11) NOT NULL AUTO\_INCREMENT ,

UserID INT(11) NOT NULL ,

Time DATETIME NOT NULL ,

Location VARCHAR(50) NOT NULL ,

Content VARCHAR(250) NOT NULL ,

Audio VARCHAR(250) ,

Image VARCHAR(250) ,

Answer VARCHAR(250) ,

AssignedTo INT(11) ,

AnswerTime DATETIME ,

Status INT(1) NOT NULL ,

CONSTRAINT PK\_USER PRIMARY KEY (QueryID) ,

CONSTRAINT FK\_QUERY\_USER FOREIGN KEY (UserID) REFERENCES USER

);

CREATE TABLE RELEASEREQUEST (

AgentID VARCHAR(7) NOT NULL ,

RequestSentBy VARCHAR(7) NOT NULL ,

RequestDate DATETIME NOT NULL ,

CONSTRAINT PK\_RELEASEREQUEST PRIMARY KEY (AgentID) ,

CONSTRAINT FK\_RELEASEREQUEST\_USER FOREIGN KEY (RequestSentBy)

REFERENCES USER

);

CREATE TABLE MULTIUSERHOLDER (

AssignedTo VARCHAR(50) ,

NoOfAssignedUser BIGINT(21)

);

CREATE TABLE WOPTABLE (

UserID INT(11) ,

AssignedTo VARCHAR(50)

);

CREATE VIEW `multiuserholder` AS select `woptable`.`assignedto` AS `assignedto`,count(`woptable`.`assignedto`) AS `noofassigneduser` from `woptable` group by `woptable`.`assignedto` having (`noofassigneduser` > 1) ;

CREATE VIEW `woptable` AS select distinct `query`.`userid` AS `userid`,`query`.`assignedto` AS `assignedto` from `query` where (`query`.`status` = 1) ;

## SQL Insert Statements

See Insert Multi Rows Excel spreadsheet file.

# Data Flow Diagram



# Use Cases

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| EVENT TABLE | | | | | | |
| Event | Ext/St/Temp | Trigger | Source | Activity/Use Case | Response | Destination |
| User asks Emma a query | External | QueryID, UserID, Time, Location, Content | User | 1. Send query | Query sent confirmation |  |
| User wants to update their details | External | QueryID, UserID, Time, Location, Content | User | 2. Send user details | User details sent confirmation |  |
| Agent sees unassigned user in open queue | State | (Triggered from Use Case 1 or 2) |  | 3. Assign user | UserID, AssignedTo, Status |  |
| User details received from user | State | (Triggered from Use Case 3) |  | 4. Update user details | UserID, Query content |  |
| Query received from user | State | (Triggered from Use Case 3) |  | 5. Look up answer | Query content, Time, Location, Audio, Image |  |
| Answer to query found | State | (Triggered from Use Case 5) |  | 6. Send response | Answer | User |
| Agent has no users to assign to themselves | State | (Triggered from Use Case 3) |  | 7. Request user release | AgentID |  |

# Reference Information

## AWS EC2 Instance

<http://aws.amazon.com/>

E-mail: [100002213@student.swin.edu.au](mailto:100002213@student.swin.edu.au)

Password: emmaproject

**Please note:** this AWS account was set up for testing purposes using a Swinburne University student email address. After November 2016, the EC2 instance will be terminated and access to this account will no longer be available. Please migrate the system to another web host before this time.

## WinSCP Credentials

File protocol: SFTP

Host name: ec2-54-206-55-193.ap-southeast-2.compute.amazonaws.com

Port number:22

User name: ec2-user

In lieu of a password, an AWS private key file was used to authenticate this account. To utilise this file, on the WinSCP Login screen below the User name, click the Edit button and then the Advanced… button. Go to the SSH heading, then to the Authentication subheading and then to the Authentication parameters section. Go to the Private Key file field and locate the emma.ppk file.

## Emma Database through phpMyAdmin

Before accessing phpMyAdmin, please add your local IP address to the phpMyAdmin configuration by following the LAMP tutorial in the Web Portal Set Up section.

<http://54.206.55.193/phpmyadmin/>

Username: root

Password: emmaswin123

## Emma Web Portal

<http://54.206.55.193/emma/index.php>

Email: [tester@tester.com](mailto:tester@tester.com)

Password: tester

## Database Connection Credentials

Host: localhost

User: root

Pwd: emmaswin123

Sql\_db: emma