Abstract

This report will detail the process of researching, designing and implementing a computerized system that administers, records, prompts and allows historical analysis of the process of locomotive sandbox refilling.

Train sandboxes are a vital part of safety equipment that a modern locomotive and its train carry. They are housed on the undercarriage of a locomotive to provide the wheels with extra motive traction by dropping sand onto the tracks rails during poor weather or windy conditions.

Due to the safety critical nature of the train carrying a serviceable quality and quantity of sand, and a series of accidents caused by a lack of these - the management of various train depots across the country will be able to investigate useful trends and information, regarding their staff and the integrity of the sandbox equipment.

The implementation is composed of a mobile application, administrative website, both built with PHP and MySQL. The mobile app makes heavy use of JQuery Mobile and is packaged as an Android application using the WebView class.

Acknowledgements

Dr Ian Bayley - Thanks to Ian for supervising my project and giving me guidance and feedback on various areas as I progressed through the project.

John Wyatt - John gave me the initial idea of the project, and asked me to potentially design and implement such a system. I did not end up designing it for him, but for my project instead - however he helped along the way and shared some industry contacts with me.

Dan Ward - Dan is a Fleet Systems Engineer at Southern rail. He aided my research by sharing how Southern rail manage the sandboxes in their large fleet. This proved to be invaluable in formulating the requirements analysis section of the report

Contents

Table of Figures

1. Introduction

It is important that rail travel is safe, and that all preventative measures against accidents are taken. As a result of, train operating companies must ensure the maintenance and upkeep of their trains is carried out on time and in the correct manner.

In the case of this project, the sandbox is our area of concern. This device is located in multiple locations on a locomotive. During weather conditions or track factors where wheel traction may be an issue, sand is dumped from these to aid the wheels rail adhesion. These sandboxes need to be refilled at regular intervals.

Whilst it may seem simple for a train depot employee to remember to fill each sandbox of each train that he or she is responsible for - mistakes happen (described on the next page), and will continue to happen.

In order for these mistakes or carelessness to pose less of a problem, I will be developing a system that tracks and records the sandbox replenishment process. This is made up of a logging application that is used by employees of train operators as they fill sandboxes, together with an administrative website that allows the data collected to be processed into useful information.

The working title for this system is the **SMS - Sandbox Management System.**

Figure 1 – Locomotive Sandbox

1.1 Justification for System

There have been several accidents involving a train trying to slow down, yet being unable to due to its empty sandboxes In November 2010 in Sussex, a SouthEastern passenger train attempted to stop at Hastings station. Instead of stopping, it slid along the rails and overshot the station by nearly three miles before stopping. (Sussex, BBC News, 2011)

The RAIB launched an investigation to identify the reasons for the train's inability to stop normally. This investigation concluded that the train failed stop as the sandboxes were empty. There were errors and inefficiencies in the method used by the train maintenance teams to make sure the sandboxes were full and had the right consistency. (Rail Accident Investigation Branch, 2011)

SouthEastern was then fined £65,000 and ordered to pay £22,589 in costs. (The Independant, 2012). A similar incident occurred again in Nov 2012, where the train slid 120 meters beyond the platform, resulting in passengers having to get off at the next station. (This is Sussex, 2012)

Train operators need to minimize disruption to services, and fines relating to missing stations or possible litigations against them like in the above. There have been many non-fatal and fatal accidents that could have been much worse had there been no sand, therefore – to ensure there is sand, a system that monitors the replenishment of the sandboxes is necessary.

1.2 Literature review

I conducted my search for literature and papers applicable to my project by looking for articles and journals about data logging. As well as this I searched for research into the process of turning data into useful information for analysis.

"Often, the use of computers allows safety enhancing features, such as more sophisticated monitoring." (Commission, Health & Safety, 1998). It is important to turn the potentially meaningless safety data into useful information that humans can understand. Using the data analysis and manipulation functionality of modern programming languages, graphs and charts can be produced to aid analysis in decision making.

1.3 Contact with the industry

I managed to get in contact with the Fleet Systems Engineer for Southern Rail - a large train operator running routes from London to the south coast. The details I learnt with my correspondence here are used in the analysis of the current system / problem, in the planning section.

1.4 Aims and Objectives

The main aims and objectives of the final SMS are:

1 To provide a method of recording reliable data from Train sandboxes that are under the care of the systems users. This would generally be a train storage depot, where trains are stored overnight and maintenance and repairs are carried out.

2 With the recording of the levels of sand in each sandbox of a train and any faults that may occur to these, the system can ensure that they never run out of sand (unless a fatal malfunction of equipment occurs) and possibly cause an accident due to loss of traction. This data can be analysed to produce useful reports for the management of the train depots that the system is running in.

3 The ultimate aim is to ensure that no accidents are caused by deficiencies in the way the sandboxes are filled or the process through which this data is logged and to provide accountability that the sandboxes were not the cause of an accident.

2.1. Requirements Analysis

In my research of the problem, the stakeholders that have been identified are railway operating companies. These could organisations specializing in either passenger, or cargo rail operation. As well as these, there are also businesses independent of railway operators who run railway vehicle storage and maintenance depots. The governments of nations that operate railways can also be considered to have a stake in such a system, due to their responsibility as rail safety regulators. These stakeholders have a legal and moral requirement to operate, or maintain safe rail based services and therefore should be interested in the proposed system.

As sandbox replenishment and maintenance is vital to a correctly function train - the need for this monitoring and data recording system will be of a great interest to these stakeholders - particularly in their operational departments

2.1.1 Current System

In my talks with Dan Ward - the Fleet Systems Engineer at Southern rail, he shared with me the process through which their system operates to deal with the problem of managing and recording sandbox data.

1. During leaf fall season, a program called XY uses an algorithm to produce a list of sandboxes on units that are likely to need refilling.

2. During maintenance, depot staff whilst they are carrying out toilet de-tanking, if possible they will fill the sandbox.

3. When those staff return to the office they clear the task on the program indicating they have filled the sandbox.

There are limitations associated with how this process is carried out.

- The algorithm that is used is only an estimation as to when a sandbox may be running low on sand, and so it still leaves the possibility that one or more sandboxes on the train could have run out, or contain critically low quantities.

- It doesn't take into account any faults that may have occurred to the sandbox.

- The quantities of sand are not logged on a scheduled basis.

- When the sandboxes are filled - the system is not immediately updated, there is a delay between the staff carrying out the filling and returning to the office and updating the system. This means that the staff could forget that they carried out these tasks, or become unsure as to which train or sandbox they actually filled.

All of these lead to a lack of recorded data during this process, and therefore the lack of proof that the sandbox system is always operating in the correct manner. In the event of an accident, the sandboxes cann**ot** be ruled out as to the cause of the accident.

Whilst I have not had contact with any other major rail operators in the UK, it is a fair assumption to say that many smaller rail equipment operators will not have had a custom monitoring system built for them (or do, but one that does not fulfil the sandbox monitoring functionality) and as a result will currently use a paper based logging system.

There are many issues associated with a paper based safety logging system.

- There are no reminders telling the staff that they have missed something that needs checking.

- Errors can take place writing details to paper, with mismatches between different units occurring.

- There is no way to analyse the data in a useful way.

To correct these issues, the proposed system and its requirements are detailed on the next few pages.

2.1.2 Proposed System

The proposed system is of type that shares several different characteristics of the two of the three main class of system - Transactional Processing (TPS) and Analytical Processing systems (APS). (Maciaszek, 2004)

Transactions are defined as logical unit of work that accomplishes a particular business task, whilst guaranteeing the integrity of the data after the task is complete. (Maciaszek, 2004)The relevance of this to this system is in logging the transactions that occur when a sandbox is checked for faults and filled. This provides data for the analytical processing of data that occurs alongside the transactions.

The main class that this system is focused on however, is analytical processing. Compared to transaction processing that typically makes changes to data, analysis of data is the main function of APS. This analysis can be used to aid decision making by the organizations with a stake in railway safety - by analysing the data to see trends in how safety can be improved, and by measuring the accuracy and completion of the various tasks required to keep sandbox equipment functioning. In the event of a rail accident - this system can prove the accountability for an accident does not fall on a malfunctioning sandbox.

This class of system is structured around a database, which allows a central storage point of information that can be accessed by many users at once.

2.1.3 Assumptions

- Due to the safety risks associated with the train leaving the depot with incorrectly functioning sandboxes, and as proposed by the RAIB (Rail Accident Investigation Branch, 2011) – no trains can leave until they are checked.

2.1.4 Functional Requirements

The functional requirements define the system functions that must be met in the design and implementation. This behaviour will be broken down into the individual functions that the system carries out, in terms of the user input, the behaviour, and the output.

Figure 2 – Functional Requirements.

1	Each sandbox of each train (unit) visiting the systems location will be checked for sand levels and any defects. This function will be performed from a mobile device. Data resulting from these checks will be validated and then saved to a database.
2	Database data can be broken down into various categories and viewed in a table format.
3	Database data can be analysed and informative reports produced.
4	Any necessary users (depot staff) will have a unique login access to the system.
5	The system will have three defined levels of access: User, Admin and Engineer. Access to certain functionality will be restricted based on the type of user account.
6	An admin user account can add new user accounts and modify and delete existing accounts.
7	The system can receive user input, to setup the database necessary for the system to operate. This data can be edited and deleted.
8	The scheduling (entering and exiting dates and times) of trains (units) will be input into the system. This will be used to determine which units and any time based warnings the system will be displayed to the user checking the unit's sandboxes. More than one schedule can be defined per train.
9	Faults with sandboxes will be recorded during the sandbox checks. An email will be sent to an engineer with the details of the fault. When the fault is fixed, the engineer will update the system to reflect the repair and its details.

10	A system settings menu will be present. The administrator can set rules that the system functions must follow. One setting would be the maximum number of sandboxes a train can have.

2.1.6 Non-functional Requirements

Figure 3 - Non-functional requirements

1	Performance - The system must maintain a high level of performance – specifically the mobile application. Response times after the user requests a page should be less than 1 second.
2	Security – Access to certain system functionality must be restricted by user type. All system functionality should not be possible to access unless a valid user is logged in.
3	Modularity - The web application must be modular, so new features can be added easily.
4	The browser used to view the admin web application must be JavaScript enabled.

2.2 Research into potential tech

In order to find the most appropriate solutions to the practical requirements, I needed to decide on the correct frameworks and languages I could use to implement the App and website. As there are many differing technologies that can be used to implement such a system, I conducted research into these.

Over the following section, I have concluded the strengths and weaknesses of the competing technologies that could be used to fulfil the necessary functionality that will be needed to meet the aims and objectives.

2.2.1 Server and Database

A web specific software stack carries a vital role in this project, as it is needed to allow access, to store data and allow connections from remote machines to the server. There are a few solutions that are widely used and suitable to this application, which are based on either Linux, or Windows. The two most widely used commercially are the LAMP stack (Linux/Apache/MySQL/PHP) and the WISA stack (Windows\IIS\SQL Server\ASP.NET).

I will be using the LAMP stack in this project, as I have studied PHP and MySQL in a previous module and as a result have some experience in the syntax of these technologies. The use of Linux as the server operating system allows for some useful possibilities.

If in the future the system is to be deployed on a commercial basis, LAMP has some distinct advantages over WISA - mainly the fact that the licences for the windows technologies in WISA are relatively expensive, when compared to the free open source nature of LAMP. After browsing for the price of Microsoft SQL Server 2012, I found the average price to be around £2,750. (Google, Microsoft SQL Server 2012 Standard, 2013)

During the process of development and testing, I will make use of a stack named WAMP - which is a package of PHP, mySQL and apache that runs on a local windows machine. The main benefit for using this over LAMP on a remote server is that as soon as any change made to code is saved, it is immediately reflected on the server, rather than having to transfer the new code to the remote server, which would take far too long. For the purpose of deployment and demonstration, the system will be hosted on the Brookes University server, named SOTS, which is using LAMP.

As a result of the choice of LAMP, the project will be using MySQL as the database. Whilst it is possible to use other database languages such as SQLite or embedded databases like CloudScape by IBM, these are not as well suited to multiple application access.

2.2.3 Data Administration Website (DAW)

Backend

The main role of the DAW is to allow the user to enter information which is then validated, and checked against the database. It also produces information based on user input and data that is stored, producing a specific set of information tailored to what the user wants.

As a result of this, a server side scripting language needs to be used. Four of the most commonly used languages are PHP, Python, Ruby and ASP.NET, all of which are free to use and all open source, apart from ASP.NET. Each of these languages has a similar feature set and allow complex scripting and access to a database.

The most logical scripting language for me to use, is **PHP** partly due to my previous experience using it, and my complete lack of experience with the others. ASP.NET is out of the question, due to my selection of Linux as the operating system. Ruby is a relatively new language which would be interesting to learn, however it is best I develop in a well-documented language that I am already comfortable with - seeing as they can all accomplish the same tasks.

Mark-up and Styling

The mark-up of a website is generally written in HTML. In this case, I will be using HTML5. My reasons for this are that; it is the future,

CSS3 will be used for styling the various pages. This is a new specification of CSS which has been broken down into modules, each of which extend the capabilities of the language. Some of these new features are quite useful to this system. Animations are a new feature of CSS3, these can improve the feel of usability to the user - these can be used to implement transitions and transformations to elements of the page. (css3.info)

Seeing as JQuery Mobile makes use of these features heavily, it would be suitable to implement a similar high quality feel to the DAW.

Ajax

In order to have a responsive user friendly design, the DAW must make use of asynchronous communication with the server, instead of relying on the traditional method of the user input, submit, response series of communication.

Therefore, AJAX - Asynchronous JavaScript and XML will be used. This allows many improvements to the traditional use of just PHP on a web application. When a form needs to be filled in, the traditional method is for the user to input the data, and then click submit which takes them to another page to display the results of the form input (such as validation or confirmation messages). With Ajax, when the submit button is pressed, any errors or messages can be instantly displayed on the same page as the form, even though it has been processed by a server side script. This allows more flexibility with the UI design and better navigation and interactivity. The disadvantage is that it requires JavaScript to be functional.

2.2.4 Mobile Application

Backend

The backend of the web app is written in PHP5 and MySQL - as per the decision to use the LAMP stack.

User Interface and Frameworks

In order to have a user friendly and accessible web app, the website needs to be optimized so that it takes advantage of the small screens and low resolution typical of a mobile touch enabled device. In the context of this system - the web app will be mainly used on a smartphone, which means displaying only a relatively small amount of information or text on the screen at one time.

There are a variety of competing solutions to such a problem. Some of the frameworks I looked into were Sencha Mobile, JQuery Mobile and Mobilize.js. The benefits of using these frameworks are that they will scale the content of the site to fit any differing screen resolutions whilst making full use of the resolution available, and they are all designed to be used primarily by touch devices.

In the end, I decided to go with JQuery Mobile. As a result of this, it will scale to any device that is using it even if the screen resolutions of these devices differ whilst making full use of the resolution available. (JQueryTeam, 2013)

Despite much of the functionality of the Administration application using AJAX for immediate response to user input, JQuery Mobile has support for this built in. Whenever the a link or button is selected, the targeted page is loaded via an Ajax call, which allows a loading animation to appear in the middle of the display whilst the framework fetches that page from the server. In the event that the mobile connection or Wi-Fi is slow, this allows some reassurance to the user that the application is still working.

Deployment

Whilst I had initially decided to write a native Android application for the mobile app - I decided against this in the end. This was due to the time it would take to integrate a native app with the database and website, and my lack of experience developing in Java. There would be a very large amount of parsing different formats of data, which would take a while to learn the syntax of.

Whilst the app will still be delivered as an android application in the format of an .apk, the content of the app is not written natively in Java using client side processing design, but is written in the form of a traditional website, using HTML.

In order for the android system to understand this, I can compile an android application written using a “view” called “WebView”. This is a way of viewing a website hosted on a server, whilst removing all of the traditional elements of a web browser - such as the address bar and back buttons. It also prevents the user from navigating away from the website by disabling any external links.

As the app is written as a website, it allows the ability to deploy the app to many different mobile operating systems - such as iOS, Windows Phone as well as Android. If the clients using the system wish to deploy the app to these other operation systems, then it will be fast and easy to do so, with no apparent changes made to the interface.

Another major advantage of deploying a web app vs. a native app is the fact that whenever an update is made to the server hosted website, it is immediately reflected on all devices running the web app without the need to update the apps for each operating system.

2.3.1 Software Development Model

A functional structured approach to the development will be used rather than the object oriented approach.

Despite my initial decision to go with the Rapid Application Development method, it became apparent that due to my lack of contact with potential clients, the major focus on passing each new prototype to the client for evaluation and modification back and forth would not be suitable.

Eventually I decided to go with Iterative design approach. This is where each component of a system is developed in an overlapping fashion. Once all of the components are completed, they are then integrated together, followed by the testing phase.

As the development of components is non-sequential, if problems arise in one, the others can continue to be worked on - resulting in continuous development process rather than the entire process halting. The model makes use of the “divide and conquer” methodology - where the large complex problem is decomposed into many modules - however, this can lead to some difficulty - as some modules may have to be worked on sequentially - these cannot be independent of one another. (Maciaszek, 2004)

Use Cases

Some aspects of the Unified Modelling Language (UML) will be used in this project, despite the system design not using the object oriented programming approach. The actors that are in these UML diagrams are the end users noted above, who are represented as stick figures.

Use cases are the main focus of the UML representation of system behaviour modelling. These represent the various transactions, operations and algorithms performed by the system on data, when requested to do so by the user. These are the logical representations of the requirements for the system and will be used as a reference for the majority of the later stages of development.

2.4.1 Requirements Modelling

End Users / Actors

There will be three different types of user of SMS. They are all employees of a railway/train depot. These are; the standard **users,** the **administrators** and the **engineers.**

These users carry out different roles in the organization of a rail depot, and therefore they perform different actions on the system. In the context of the Sandbox Management System their roles are detailed below.

User (Sandbox checker) - This is the most common type of user. Their main role is to use the smartphone app to indicate to them which trains they are responsible for checking sandboxes. They also have the ability to add trains into the database, and edit them (if they are given permission by an **administrator**).

Administrator - These users are the senior figures involved in the running of the rail depot. As a result they are given higher privileges than a normal **user**. They have the ability to modify the user accounts (creating, deleting, modifying) and to change the rules and parameters of the system. They can modify the database to a greater extent than the other types of user - such as deleting trains and setting up the scheduling for each train.

Engineer - The engineer is another type of user of the system. Their role is essentially to fix any mechanical problems related to the sandboxes. As this job will be a small part of their day to day role as an engineer in a train depot, they do not really need access to the database or the sandbox checking app. They will receive emails and text messages when a problem with a sandbox is reported by a user. This email would contain the exact details of which specific sandbox and train is broken, along with the comments left by the user.

4. Design

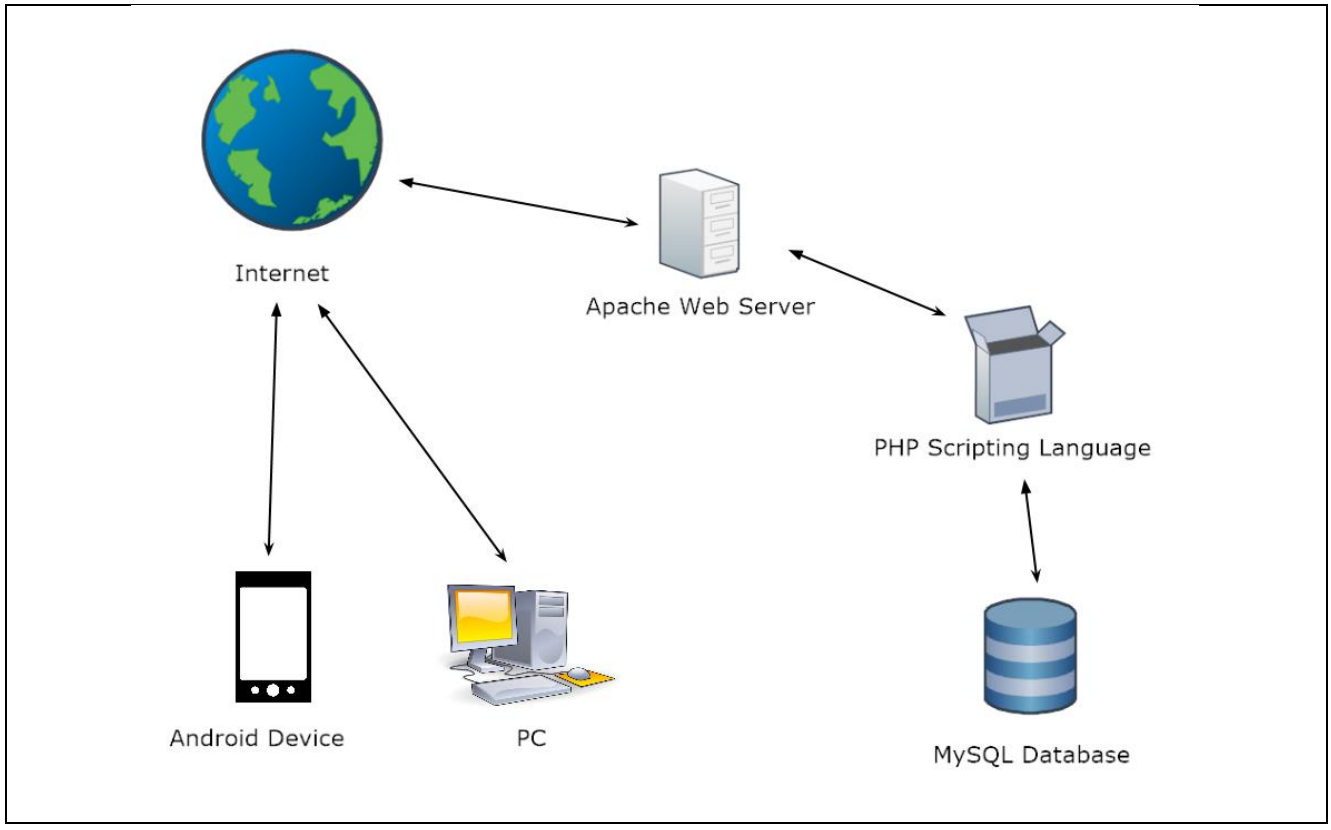
System Architecture

Essentially, the system is split into three parts. These are the server, and a PC or laptop.

- **Linux sever**, with Apache, PHP and MySQL installed (LAMP).
- **Two web applications** - The Data Administration website, and the Mobile Application website. These both access the database to communicate with one another.

Train, sandbox, user and setting information are entered into the database using the administration website from a PC. Data generated by the app can be analysed and generated into visual information or lists. The mobile app then accesses this information and generates the list of Trains for the employee carrying out sandbox checks to complete. The data from those checks is then sent to the database.

Figure 4 - System Architecture Diagram



4.2 Database Design and Implementation

The database has been designed in MySQL, using phpMyAdmin to administer and create the various tables and define the various fields. It is important that the database design meets the requirements specified in the Planning section. The tables and fields need to be normalised to ensure that there is no redundant or duplicated data, as this may lead to a duplicate of some data being updated, whilst the other not - leading to corruption and system errors.

Normalisation

The data is normalized to third normal form.

The benefits of normalization are that changes to a certain record need to be only made once, as that data is only located in one place in the database. However there are some disadvantages in that certain queries may take longer than if the database was not normalized, or normalized to a lower normal form. This is because multiple lookups and too many tables mean searching through the many tables may lower the performance, and make programming the access to the database overly complicated. However in this case, the database will not be huge, and so normalization to the 3rd NF will be the best route forwards.

Figure 5 - Database Table Descriptions

user	The user table contains data related to the users. E.g. name, username/pw.
train	Stores the train name, scheduling, # of sandboxes, type. See below.
sandbox	Stores sandbox check data.
traintypes	Serves as the image location table for different train types.
news	Stores news content for mobile and web application.
settings	Stores saved settings.
repair	Stores repair information.

Figure 6 - Entity-Relationship Model

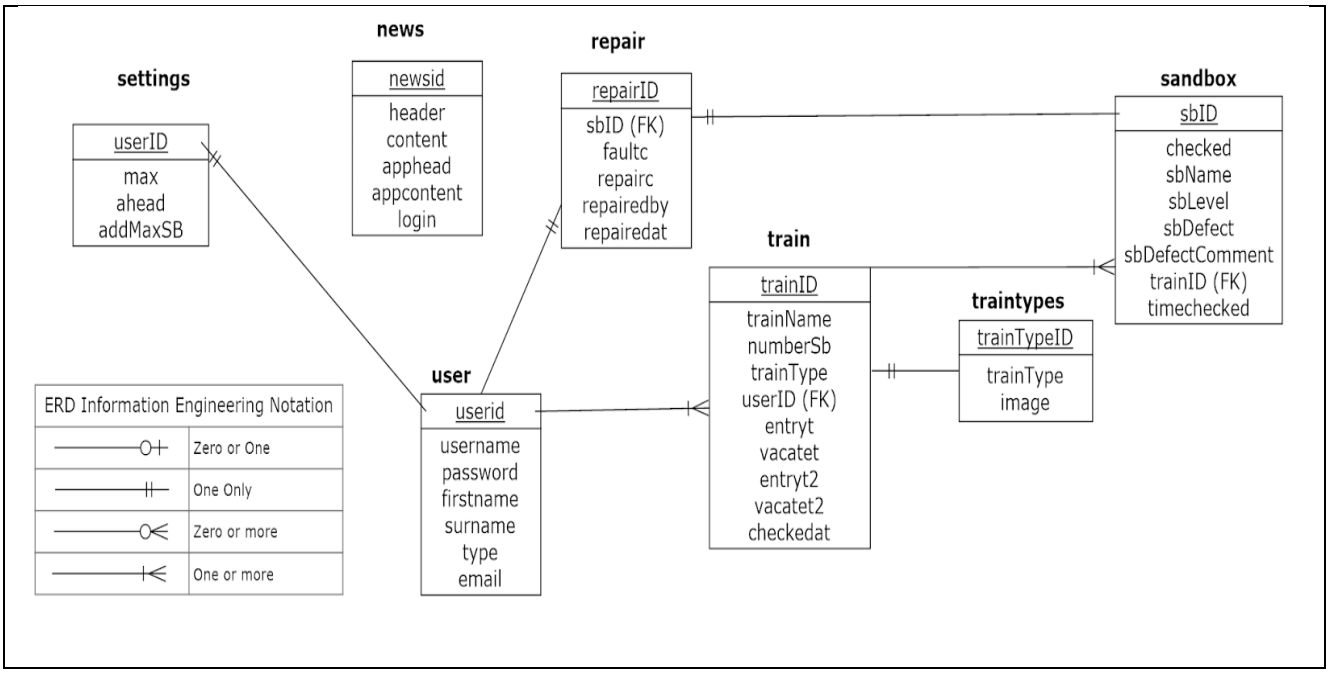
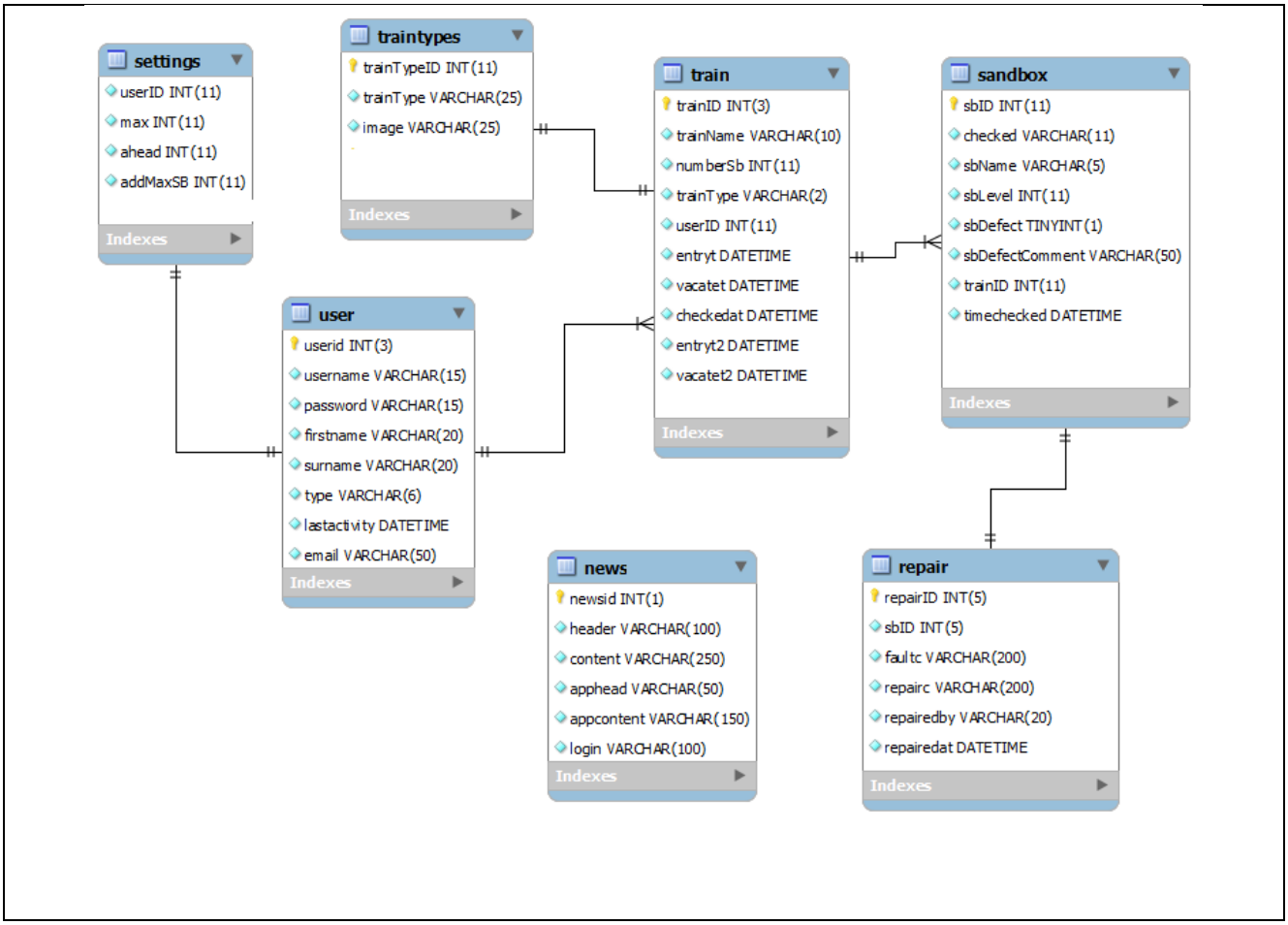


Figure 7 - Physical Implementation from MySQL workbench



4.3 User Interface Design

One of the most important aspects to follow during the UI design is to design a **usable** UI. That means a user should be able to carry out system tasks quickly, in an effective and enjoyable manner, requiring as little training as possible and allowing the user to feel in control.

There are some key usability factors that the UI for both applications should meet, despite the differences in designing for a desktop or laptop computer vs. a 4/5 inch smartphone screen. These are described below:

- Efficiency - how quickly can the user perform the required functions? Is there repetition involved in performing common tasks, or is the completion of tasks intuitive, whilst removing the need for the user to keep repeating themselves.

- Simplicity - is the system logical? Would an untrained user be able to quickly grasp how to perform a task? Is the information displayed on the screen only relevant to the particular tasks the user needs to carry out?

- Memorability – It is important the users can quickly learn the functions of the applications as they will be using them many times a day

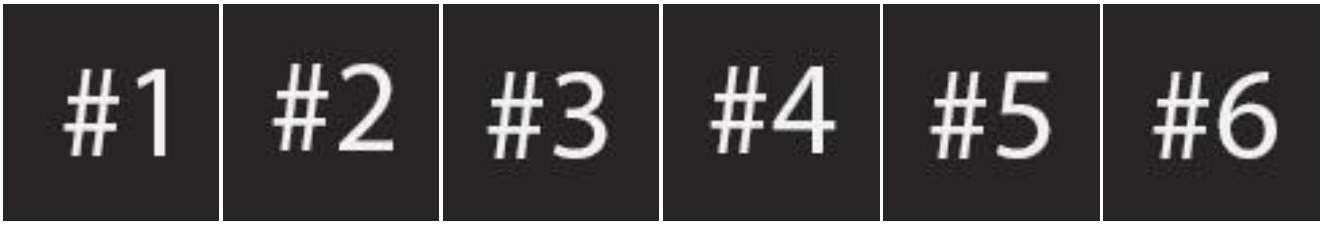
A series of iterative prototypes will be developed using a GUI prototyping tool called Pencil. Once the prototypes meet the requirements, the implementation for the UI begins alongside the programming implementation. A GUI prototyping tool called Pencil is used to sketch these. When the design prototype is finalized, an appropriate colour scheme can be chosen that will be implemented in both applications.

Images and Banner

Most of the graphics used in both applications have been designed by me in Photoshop. They are primarily designed to be clear, simple and informative to the user. As the system is not attempting to convey a brand or attract customers, the design of these is simply to fulfil the functions of the system.

Figure 8 - SMS Banner

Figure 9 - Images indicating the sandbox number being displayed for a train;



There are a few graphics that have been taken for the internet. There are four images of different rail operator's trains and brand, which represent the 4 train types a train can be. These were located on the operator's websites. An image of a green tick was also found on the internet.

Figure 10 - Train Type images

Chiltern

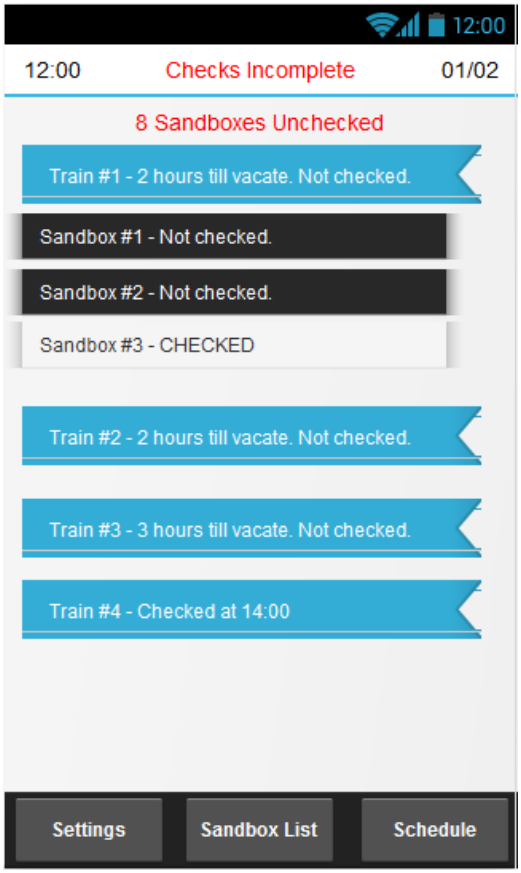
Cross Country

First Great Western

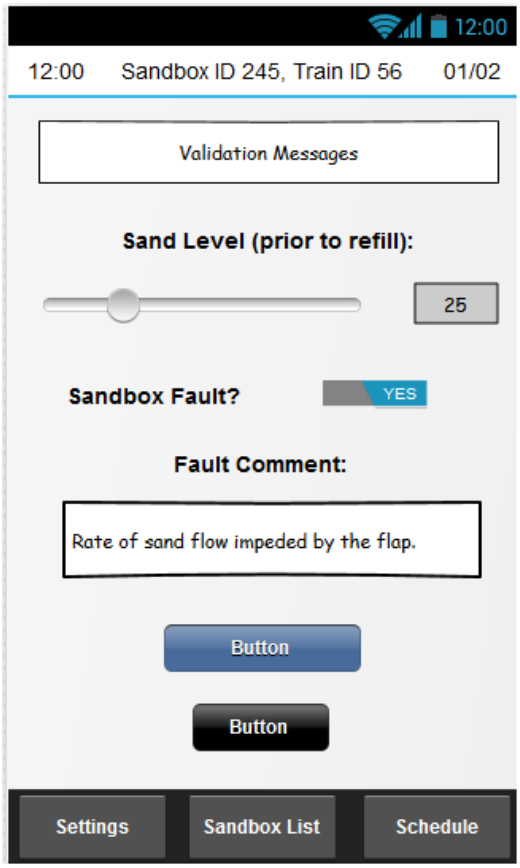
Southwest

Figure 11 - Prototype Mobile App UI

Train and Sandbox listing:



Sandbox Check form:



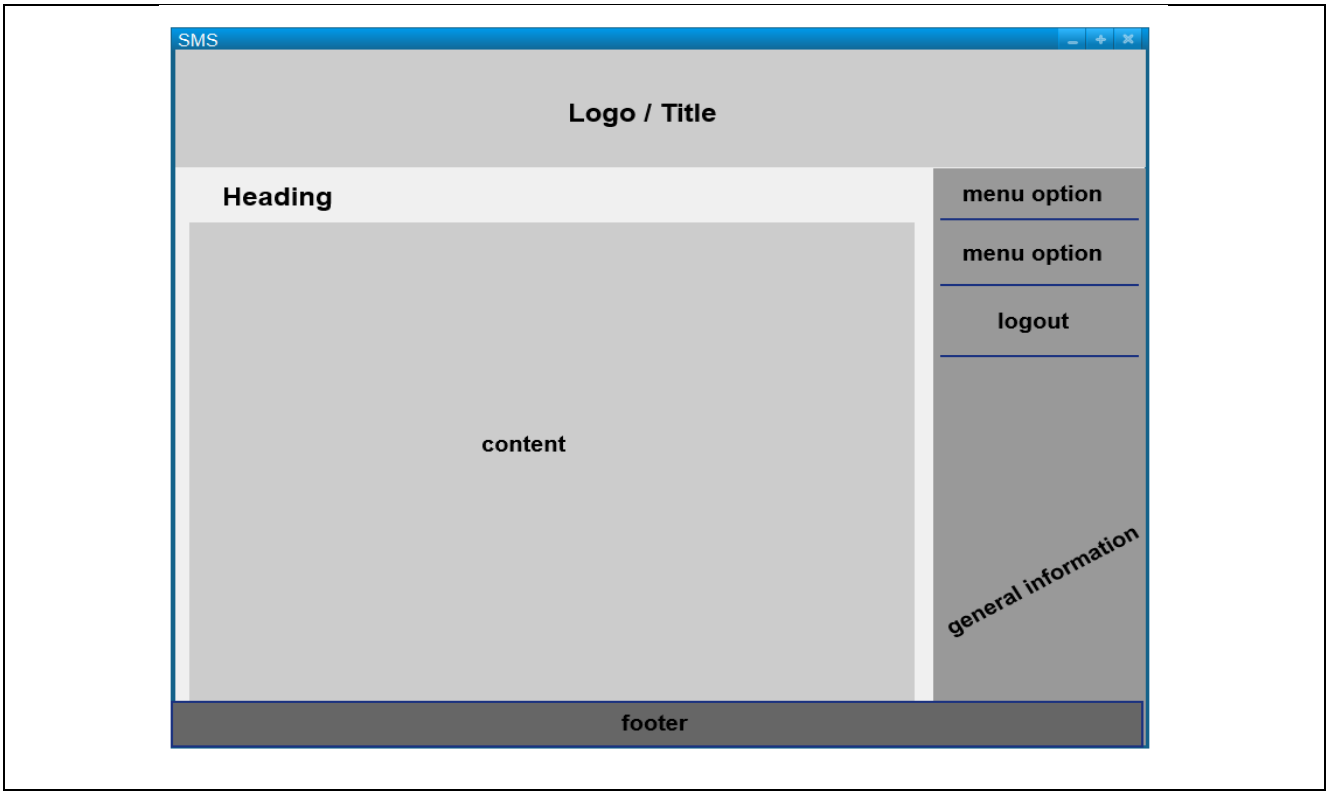
When the user comes to perform the sandbox checking, they are presented with a list of trains and their respective sandboxes. The sandboxes are hidden until the train tab is selected, which then drops down a list of the sandboxes. Black indicates they are unchecked, white means they are checked.

The form is displayed when an individual sandbox is selected. The user must then either move the slider to indicate the % amount of sand as indicated in the actual physical sandbox, or tap the number next to the slider and enter the number using a numeric keyboard.

It may be possible to get rid of the slider and the keyboard, and have a numeric keypad built into the form as buttons to be pressed, however this may need to be implemented at a future date.

Web Application UI

Figure 12 - Prototype Web Application UI

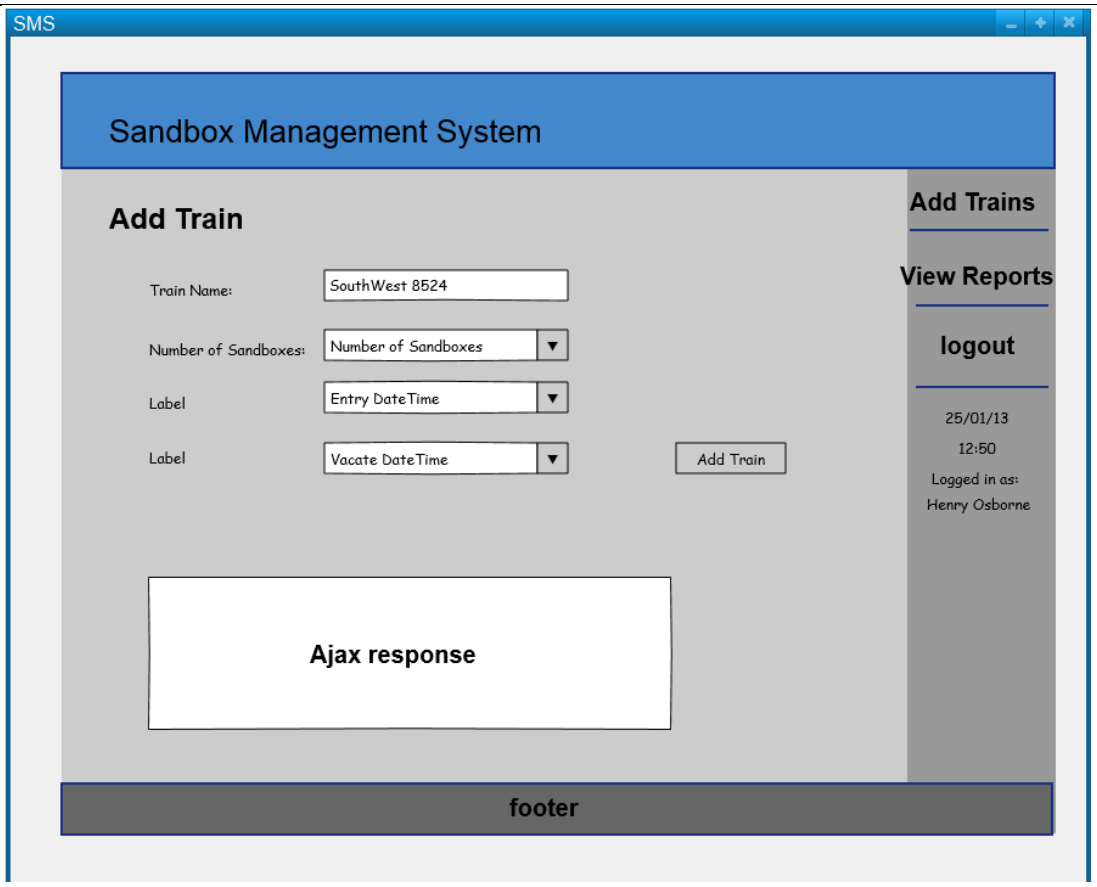


An initial prototype for the layout. A navigation area is located on the right side that maintains its position throughout all pages in the web application.

- The menu options vary depending on what type of user account is logged in.
- The general information area will indicate to the user, the time and date and their name.

As I need to set a maximum width for the website, I conducted some findings into the most common screen resolutions, in offices (the clients hardware would probably match these statistics) and in general. W3C schools found that in January 2013, 90% of the visitors to their website had a screen resolution of 1024*768 or higher. (w3c schools, 2013) The next resolution up from this is 1366*768, so the interface is designed to be usable in this resolution and upwards.

Figure 13 - Second Web Application Prototype UI



- The Ajax response area will confirm user actions.
- Green text will be confirmation of success.
- Red text will be validation failure or error messages.

4.4 Mobile Application Design

Overview

The mobile application is what runs on an android smartphone and is used to the depot employees when they are due to check sandboxes on trains residing in the depot. The main function is to log the replenishment activities for each train and its respective sandboxes.

It will use PHP and MySQL to script and manipulate data; the output of these scripts is in the format of HTML and JQuery Mobile. This creates a dynamic interface that is specifically designed for the smartphones touch interface. The application will be packaged as an Android application, using the WebView.

Main Functions

Check a Sandbox	When the user has selected the sandbox they wish to check, a form is displayed. This receives three sets of input. 1 The current level of sand, as indicated by markings in the sandbox on a 1-100% scale. The user has the option of entering numeric text input with the virtual keyboard, or dragging their finger over a slider that represents the level based on the position of the slider. 2 A Boolean option for whether or not the user believes that there is a fault or malfunction with the sandbox. In this case, the user would represent this as a switch they can press. This would be saved as either: **1 -** yes there is a fault. **0** - the sandbox functions correctly. 3 If the user **does** believe a fault exists, they can enter a text comment using the virtual keyboard, with a brief description as to what the problem is.
Display Trains (and their sandboxes)	Displayed upon successful login. This will display of all the relevant trains, and their respective sandboxes specific to the user that they have been assigned to - the one who is logged in. Only trains due to leave on the same day will be displayed. The trains themselves are displayed in a list format. When they are selected, the list of its sandboxes drops down from the train item.

View the Schedule	Lists all trains and information related to those trains that are registered to the user.
Login / Logout / Main Menu	When the user enters the username and password, check these against the user db table. If there is a match between them, then set a PHP session with the type and id of the user where the match was found.

All types of user can perform the same actions on the mobile application. During development, there will be a settings menu, where the sandbox data can be reset – to reduce the amount of time the testing and bug fixing takes. However, this will be removed when the implementation is complete.

Scripts needed –

index.php – Will display the login screen to the user, and perform the logout script.

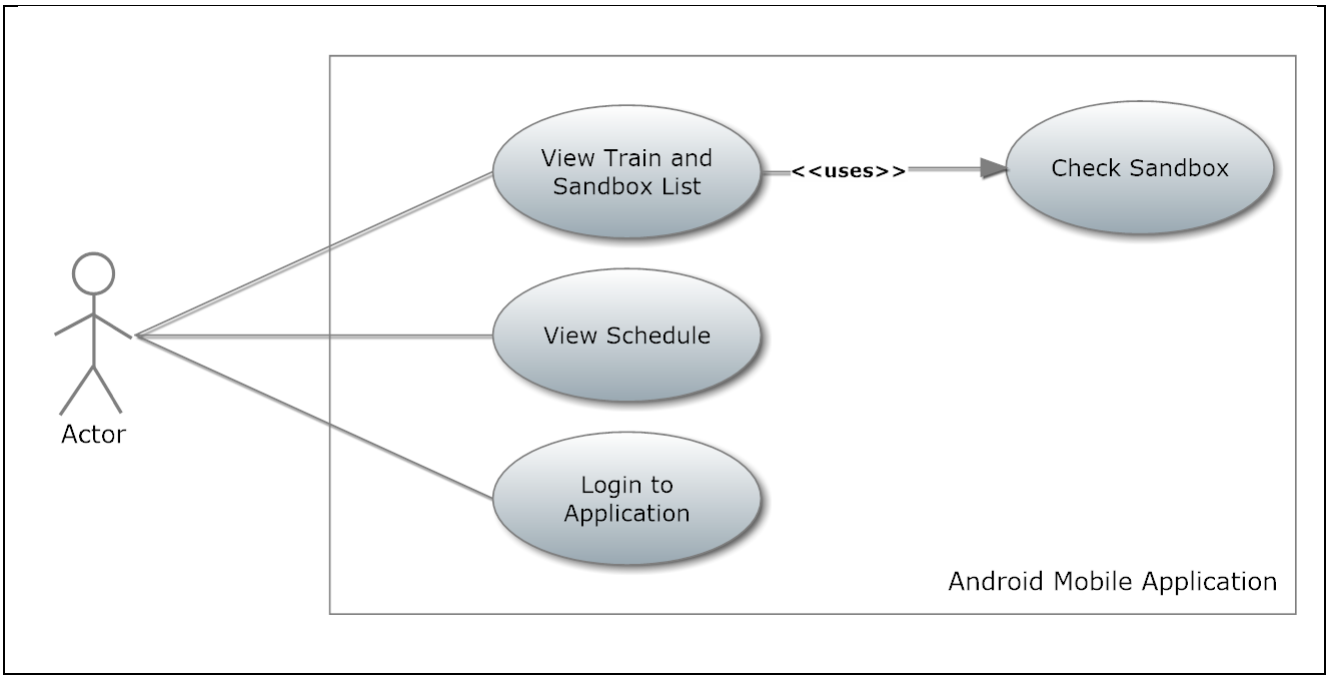
login.php – Recieves login values and checks them against db. If successful will redirect application to the train and sandbox listings. (train.php)

train.php – Queries the db for valid trains and sandboxes for the user. Displays them using Jquery collapsible lists. Creates the links for each sandbox to the sandbox check form.

sbcheckform.php – Produces the form that is filled in when the user inputs the sandbox check data.

schedule.php – Produces in table format, the details of all the trains registered to the user, rather than just the trains leaving on the day as in train.php.

Figure 14 - Use Case – User (User, Admin, Engineer)



Please see Appendice A for use case descriptions.

4.5 Web Application Site Design

Overview

The web application will carry out many functions related to data entry and the viewing of data. In order to satisfy these functions, the form design, validation and confirmation of user input needs to be a priority. As discussed earlier in the research section, it will make heavy use of AJAX, in order to provide instant notification to the user of any action they perform, whilst remaining on the same page and not breaking the flow of data input.

Main Functions

In order to fulfil the functional requirements, the functions that the DAS will perform are listed below by being broken down into the main topic or area where those functions should be met:

Login / Home	Displayed to the user upon typing in the URL into the browser. A login form is located here, for the username and password, followed by a button to click to login. If the user attempts to login with invalid credentials, a message is displayed telling them that these details are incorrect. Once the user has logged in and been authenticated the main part of the screen will change to a news listing. This news can be set by an administrator type account in the settings menu. The navigation menu on the right side of the screen will change from the login box to a menu, with links to navigate to the rest of the DAS functions. These links will vary depending on the account type the user has logged in with.
View Reports and Data	This area of the site is used to view **reports** and **listings** of various types of data in the database. The **reports** are based on a selection of predefined reports, such as missed sandboxes, sandbox fault occurrence, trains leaving late. These are presented to the user in the form of charts, which will be created dynamically based on data in the DB and which report the user selects to view. In order to convert this data into a chart, an algorithm is performed that gathers and sorts the required data, before it is sent to the Google Chart API. This then generates a Pie or Bar chart/graph and sends it

	back as an image, which is then displayed. The **listings** are where data is displayed in a table in ascending or descending order, based on the category of data the user selects. This will include categories such as: Trains, Sandboxes, Users, Train Types, Faults, Repaired Faults, and Schedules.
Database Setup	This part of the site is where we can setup the database, by entering trains into the scheduling system. This has four functions: Add Train, Change Train-User, Setup Schedule, Edit/Delete Train. There will be heavy use of Ajax here, so that the user does not need to navigate away from each section to get confirmation of changes made to the database. As there is a need to have a validated method of date and time input for correct scheduling, I will use a JS plugin for this functionality. The DatePicker and TimePicker classes will be used. (TimePicker).
Engineers Panel	Here the engineer(s) can view reported faults. When the faults are fixed the engineer leaves a comment about the repair. Admin and Engineers can access this.
Users	An Admin can setup, modify and delete user accounts.
Settings	News and other settings can be set here.

Figure 15 - Use Case – Admin

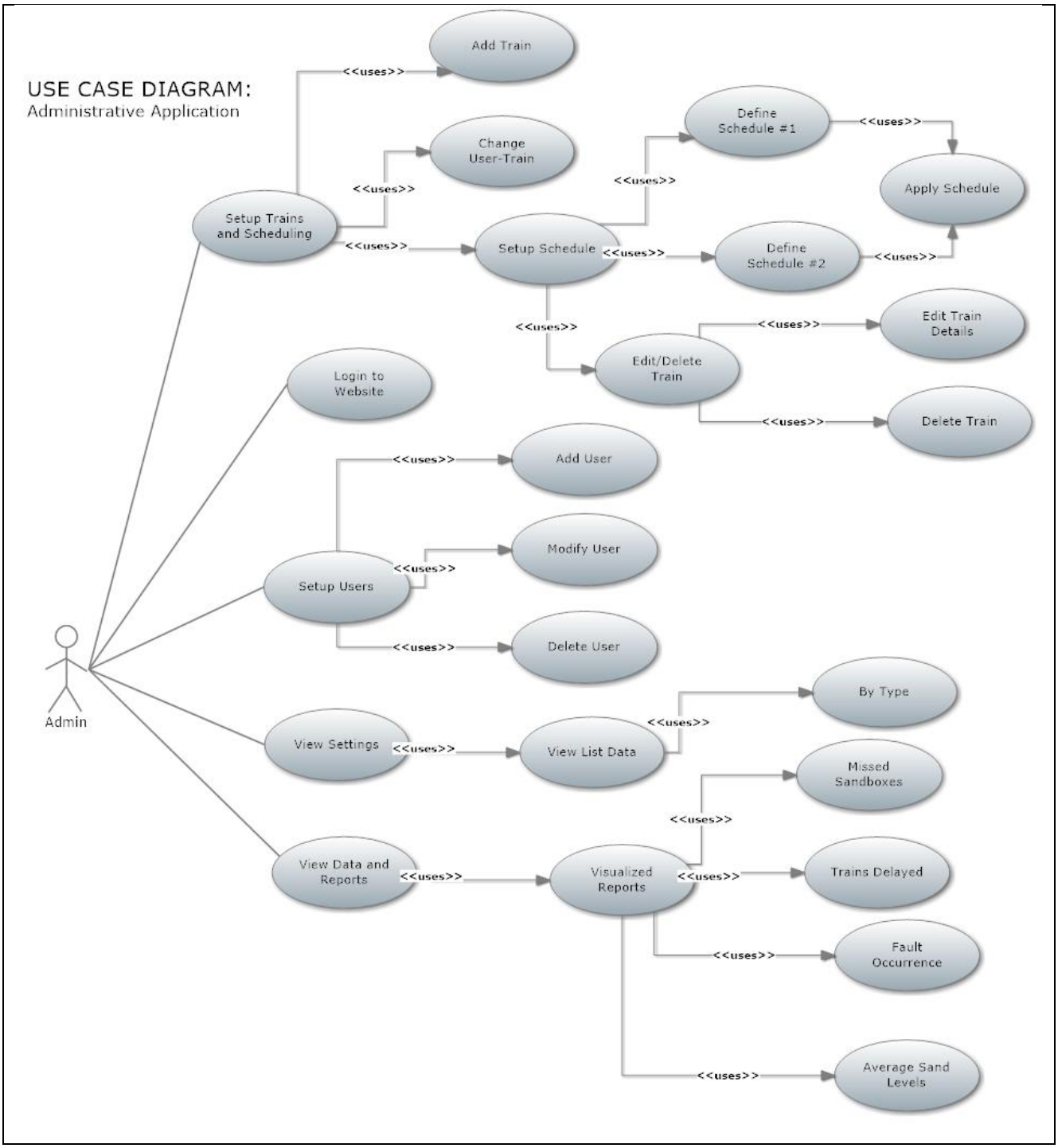


Figure 16 - Use Case – User

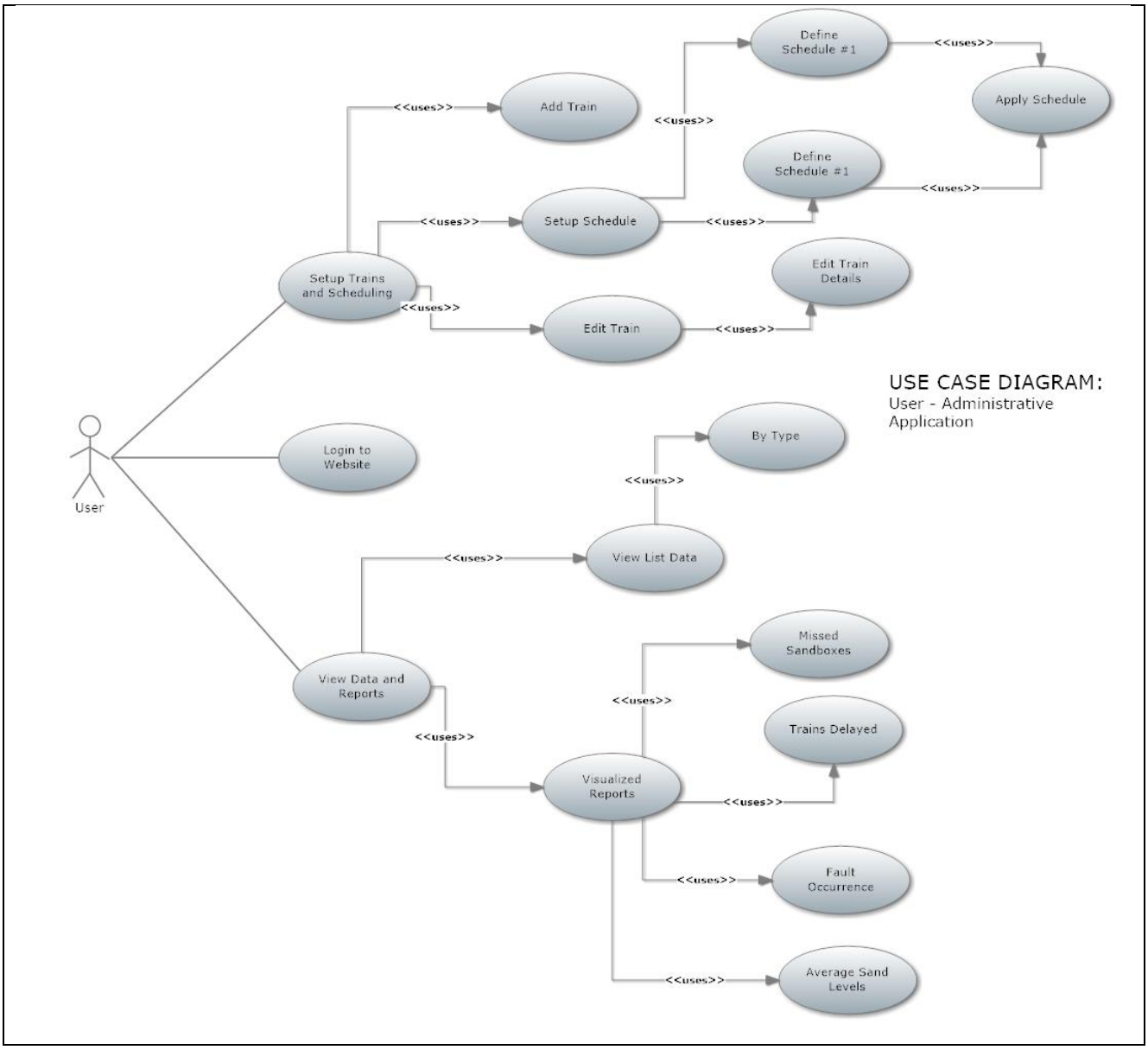
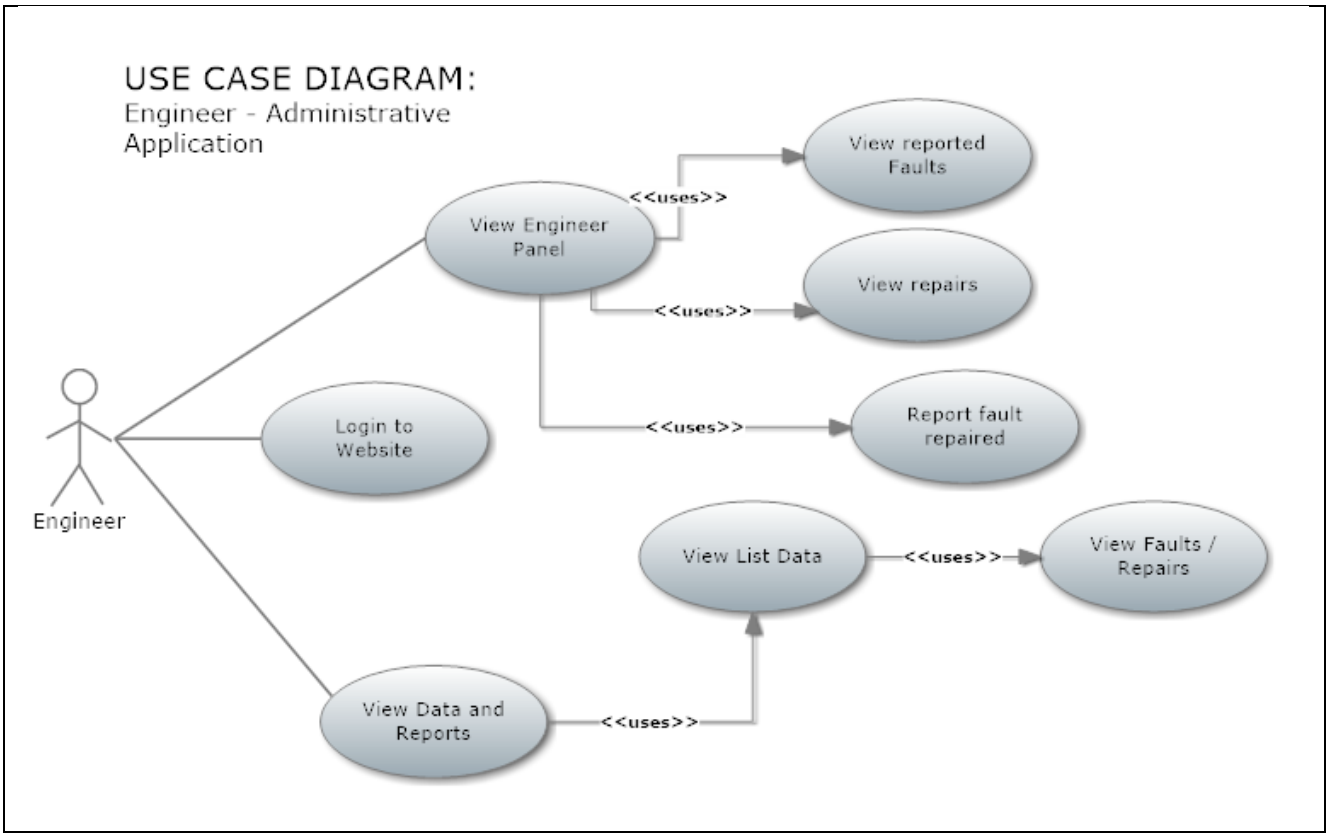


Figure 17 - Use Case – Engineer



Please see Appendix A for use case descriptions.

5. Implementation

5.1. Mobile Web Application

The implementation of the mobile application begun as soon as the basics of the database and administrative site were complete. The lengthy process of scripting the train and sandbox page was then begun. New features were added incrementally, most of these were functions related to the scheduling.

Implementing the website as an Android Application

Packaging the web application as a native Android application in Java was not too difficult, thanks to the documentation provided by Google. (Google, Building Web Apps in WebView, 2013) There are three main steps to this, which involves three files in the application structure. These are the MainActivity.java, AndroidManifest.xml and activity_main.xml

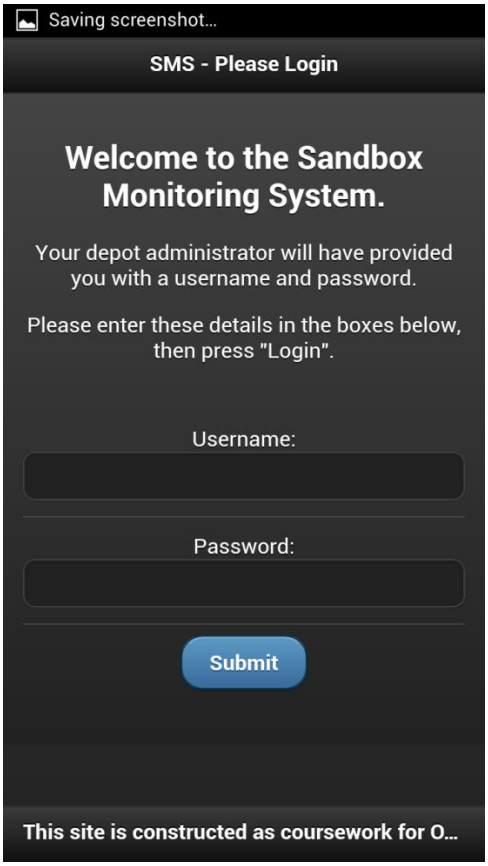
Firstly an activity layout of the type WebView was setup in activity_main. The height and width of the layout are set to fill the parent, so that the screen is filled (apart from the status bar – but this is needed to view network status).

In the main activity java file, the contentView is set to the activity above. JavaScript is enabled, the scrollbars are set to be hidden, and a class loaded that stops the focus of the activity moving onto a new URL when it found – this prevents an external URL being navigated to, rather than internal URLS. The loadUrl is then set to the sots URL (Osborne, 2013).

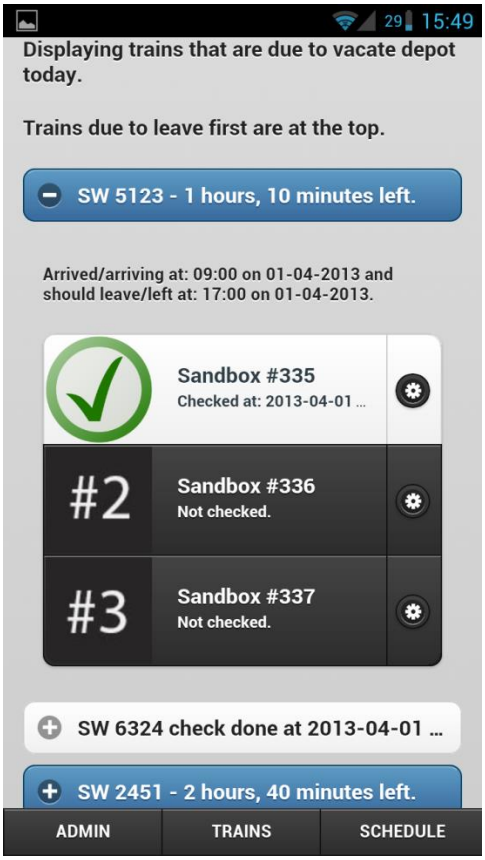
Lastly the android manifest needs to be configured to grant the application internet access, a theme set to remove the activities title bar (which would get in the way), and lastly a very important configuration – enable hardware acceleration. Due to the JavaScript heavy nature of JQuery mobile without hardware acceleration enabled, the user experience is poor due to low level of performance with it disabled.

Figure 18 - Login, Train and Sandbox List

Login Screen

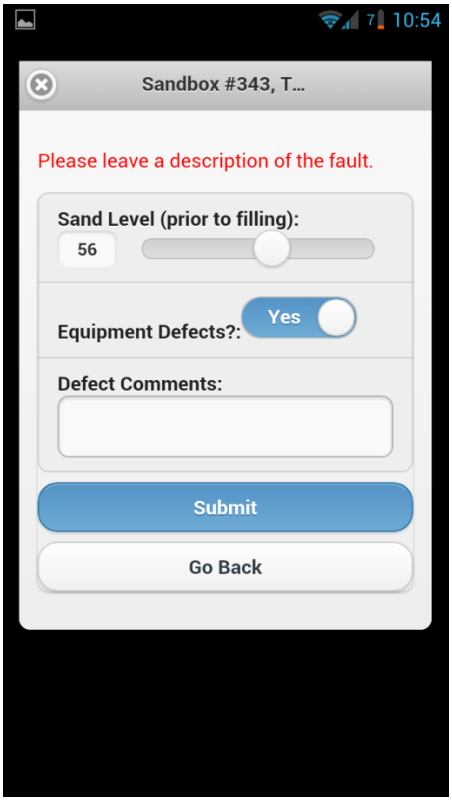


Train and Sandbox List



In the screenshot on the right we have the train and sandbox listing. As was planned in the design, the use of colour to notify the user which sandboxes and trains have had their checks complete has been a priority. The contrast between the white and black with the green tick sign is a clear indication of when that specific task has been completed. To prevent the checked trains clogging up the interface, they are removed after a period of 30 minutes. This gives an appropriate amount of time for the user to be clear of this fact.

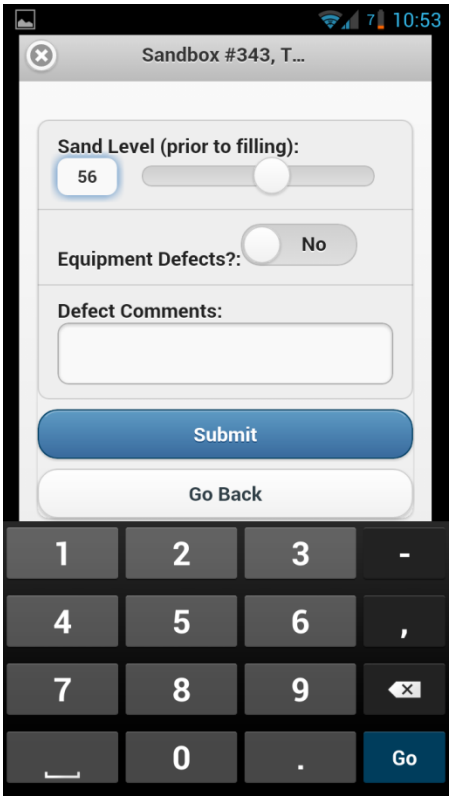
Figure 19 - Sandbox Check Form



Here we have an example of the implemented sandbox check form in action. This is displaying an error message, due to the user not leaving a comment about the fault they have indicated.

By default, the sand level is set to 0. If the user submits when the level is still 0, then they will asked to confirm that this is correct.

If the user has made a mistake, it is possible to go back to the form for an already checked sandbox. In this scenario the values that were entered previously will be displayed, with a message indicating the time that the last check was made at.



To the left is an example of the alternative option for entering the sand level. When the user taps the number box next to the slider, the numeric keyboard is bought up. Even though the slider is very accurate and easy to manipulate, some users may prefer to enter the digits instead.

Fault Report Emails (validate.php, class.phpmailer.php)

Each engineer will get mail about any fault reported with a sandbox. This uses the open source class PhpMailer. (Worx International, 2013). The PHP built in mail class was not used because it doesn't support emailing from a gmail account.

Figure 20 - Sandbox Check Form

index.php – Sets up the initial page that is displayed to the user. This contains the login form, along with the submit button so that the user can validate themselves. When the username and password are entered, the form submits to the same page. It is then checked against the user table for a valid match. In the event that it is, various $_SESSION vars are set that are need to be globally accessed in other parts of the app.

train.php - This script is response for querying the database for the train and sandbox data specific to the logged in user. It calls **update_schedule** initially – which looks for "checked" trains that were checked more than 2 hours ago and have their #2 schedule defined. If there are results for this, the #2 schedule is inserted into the #1 schedule and #2 deleted.

Next, checks are made to determine if either there are no trains registered to that user, or no trains registered that are due to vacate on that day. The respective message is displayed to the user. Otherwise the script continues. The number of sandboxes left are then calculated so that this can

be displayed in the Title of the page, along with the counter indicating how many are left or if the checks are completed.

A while loop then begins which opens the collapsible boxes for each train. It determines if these trains are checked or not and sets the colour of the box to blue (checked) or white (unchecked). If the train has been checked more than 30 minutes ago, its html tags are not echoed. The loop determines if checks are late (the train should have departed) or the time left till it departs. Inside the train while loop, the sandbox loop generates a list item for all sandboxes for each train. Links are placed on these list items with **sbcheckform.php?sb=$id** as the location to take the user to the correct check form for each sandbox. Any loops still open are then ended, and the footer function called to place the navigation links.

sbcheckform.php – Receives the sandbox id from **train.php**. Generates the form that the user fills in. An Ajax function is present that validates the form values with the use of **validate.php**. If the Ajax response contains validation errors, these are displayed in red text on the form. Otherwise a JavaScript is sent back which redirects the user back to the **train.php**

validate.php – Receives the form values from the sandbox form via Ajax POST. It validates these and sends back any validation messages requiring the user to change the values. Assuming the validation stage is passed, if any faults are recorded, these are added to a the fault_occ mySQL table, along with an email sent to the engineers with the train/sb ID’s and comment left. Now the sand level is added to the average, as well as an increment to the level taken – both located in the fault_occ table. Finally the data is added to the sandbox table in mySQL. A calculation is then performed to determine if all the sandboxes for a train are checked. if they are then the time at which the most recent box was checked is added to the train table as the “checkedat” time.

5.2. Web Application

The administrative website was the first part of the system that I started to implement.

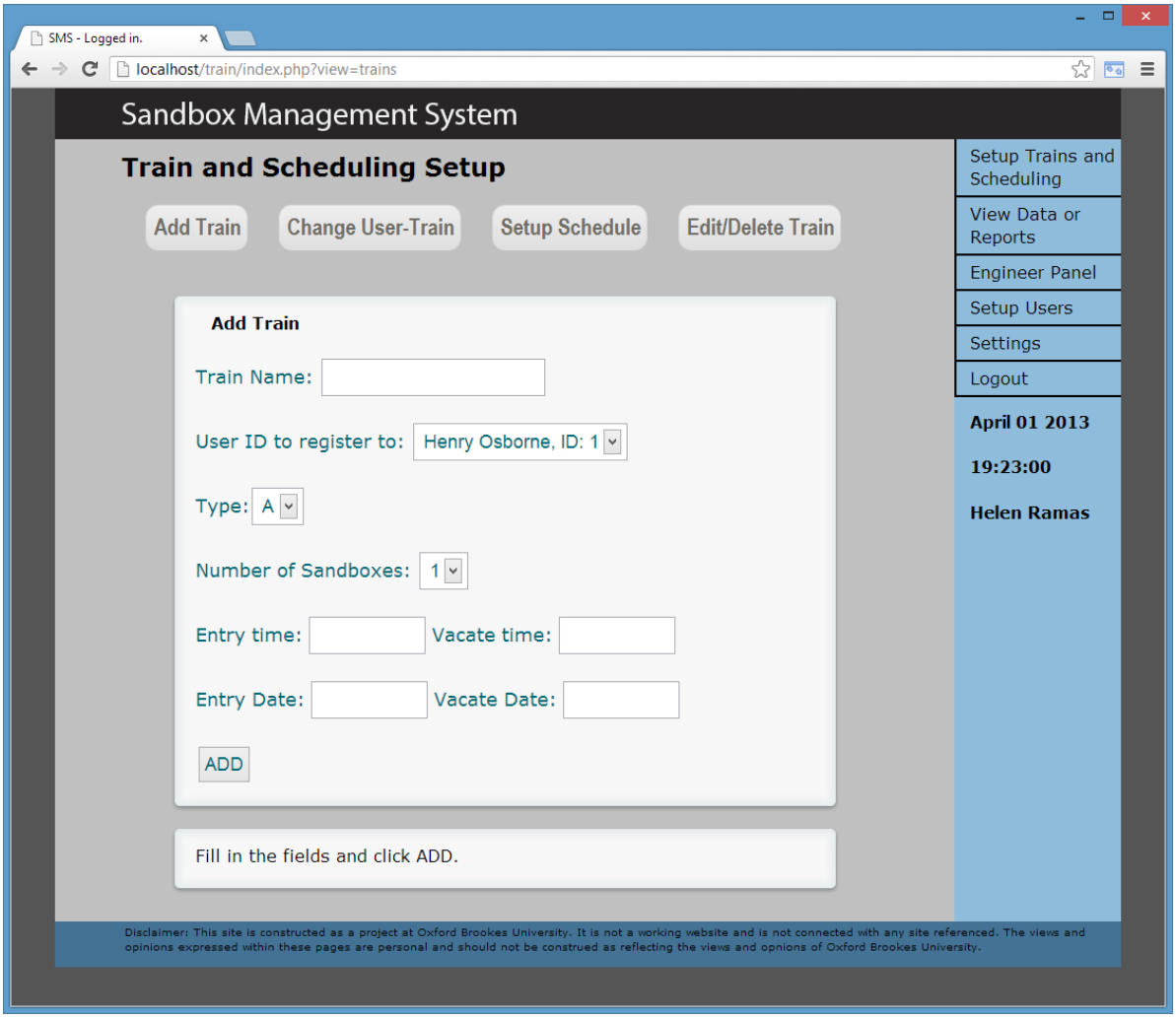
The HTML layout was implemented initially with the aid of Dreamweaver. Once a basic layout for the menu and structure was formed, it was then modified manually. The CSS that was created by Dreamweaver is noted in comments in the appendices.

The next features that were implemented were the train and scheduling forms. The user and engineer panel were next to be implemented, whilst the last features were the reports and styling.

Interface

The final interface ended up fairly similar to the prototype design (please see appendices for detailed screenshots of the rest of the site.):

Figure 21 - Web Interface

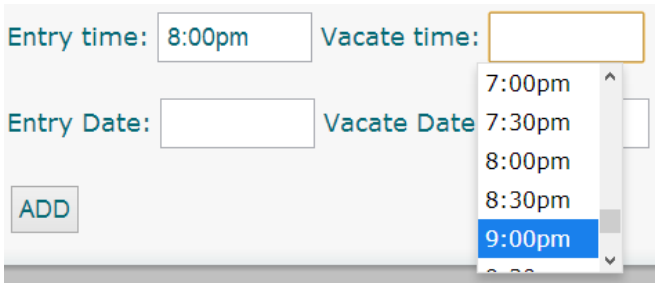


Validation

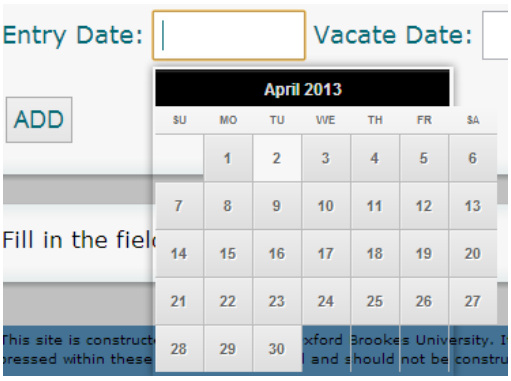
Due to the many fields of data required to be input by the user, as many of these fields as possible were implemented so that the input was predefined – mainly be using combo boxes. This is seen in the previous page, where the user ID to register the train, the type and the number of sandboxes are predefined so that no incorrect data can be added.

Figure 22 - Schedule Validation

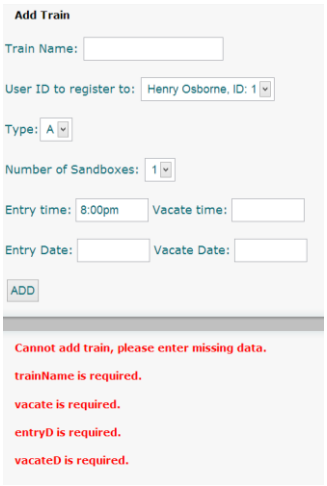
When it came to validation of time and dates, this was made simpler by the use of the JQuery classes DatePicker and TimePicker. Whilst the DatePicker class prevented any user input in the field, the TimePicker does not. Therefore there is some PHP validation making sure that the fields are present, and error messages will be displayed if the time is in the wrong format, or the entry time is in the future compared to the vacate time.

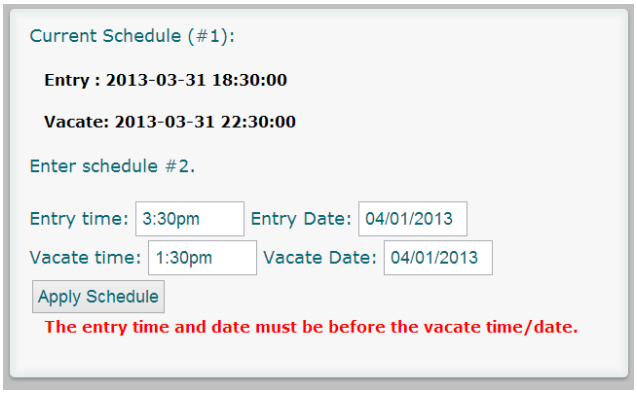


TimePicker – when form is submitted, the Ajax response will indicate if the logic behind the scheduling is incorrect.



DatePicker – the class will prevent the user entering a date in the past. The Ajax response will validate the logic of the dates.





Implementation of Functionality

Some of the interesting implementation is described here. The rest of it is explained as comments in the code itself, along with some in the appendices.

Ajax – the Ajax JavaScript functions are located in each PHP file that requires it. These functions are specific to each page, due to the way the pages element ids are appended into a string, which are then sent to a PHP script. The script then receives the string as individual $_POST variables. After processing these, the response is output as an echo – and placed in the element defined by the Ajax function. Much of the website uses this functionality so that the user can perform many actions without leaving the page.

Figure 23 - Ajax implementation

```
22  function ajax(url) {
23  // gets the element with the ID of 'report'.
24  var report = document.getElementById('report').value
25  // builds the variable sent via $_POST
26    vars = 'rep=' + escape(report);
27  // allows the updating of the page asynchronously
28  if (window.XMLHttpRequest) {
29  // defines the request as xmlHttp
30    xmlHttp = new XMLHttpRequest();
31    }
32  // sets the request up as POST, the url passed to this function
33  // is the destination php script that recieves the post data.
34  xmlHttp.open('POST', url, true);
35  xmlHttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
36  xmlHttp.onreadystatechange = function() {
37  // when the readystate is reached,
38  // this means the script has sent back its response
39  if (xmlHttp.readyState == 4) {
40    // this places the responsetext into the div called visualchart.
41          document.getElementById("visualchart").innerHTML=xmlHttp.responseText
42      }
43
44    }
45  // sends the built variable containing the POST data.
46    xmlHttp.send(vars);
47  }
```

index.php – Contains the login, logout and timeout functionality, as the index is run each time the user navigations the website. A switch statement is present that receives the keywords login, logout which then runs the respective functions located in **function.php**. Every form of navigation between pages is defined as: "**?view=something"**. The index uses the function $_GET to receive this variable and looks for a file of the same name in the views folder. It then appends this to a layout which differs based on if the user is logged in or not. **layout.php/loggedin.php** contains the html doctype, head, header, the menu, the footer and the closing tags. The view file only contains

what is placed within the content div. This allows for an efficient reuse of code as there is no need to repeat the menu and all of the divs in each page.
function.php - This file contains any commonly used functions such as **user_activity, login, logout, db_connect, scripts_css, jquery_scripts.**

gchart.php - This is an open source object oriented PHP wrapper designed to allow easier access to the Google Charts API. (Google, Google Chart, 2013) It helps automate the process of sending the variables in the correct syntax to the API in order to build different charts. (gChartPhp, 2013)

chart.php – This works in combination with gchart.php. It uses the classes of gchart to send the script results to Google, and receives back an image which it then sends via Ajax to visual.php which then displays the image. The reports produced by this file include, **late trains, missed sandboxes, sandbox fault regularity (by sandbox and by train) and average sand levels (by sandbox and by train).** This was the most challenging area of implementation for the web application – but was eventually overcome by learning the mySQL aggregate functions.

Style.css – contains all of the styling of the website, excluding the JQuery SS. Some of this was generated by Dreamweaver, and then modified to suit the design. The parts generated by Dreamweaver are commented in the appendices. The button CSS was initially created with the aid of a website: cssbuttongenerator (FileCart, 2013) and then modified to appropriately fit with the styling of the site.

Changes from Design – Instead of using separate date and time columns for the scheduling, Datetime was used instead. Certain SQL queries did not work correctly unless Datetime was used.

5.3. Implementation Factors for both applications

Browser Compatibility

As the system is not used by the public, on a public website - I can narrow down the browsers that the sites are compatible with. Whilst it would be a benefit for the data administration site to be compatible with all browsers, this is just not possible, due to the variations in how different and especially older browsers render the mark-up.

A vitally important factor with the browser that is being used is it **must** be JavaScript enabled - with the JavaScript turned on. If there is no JavaScript, the system will not function correctly. It is not an unreasonable request for the end users to have JavaScript turned on, since in this modern age, almost every website on the internet makes use of it.

The primary browser that the DAS should be used with is Google Chrome (a version up to date to at least the start of 2013). It should be feasible to expect a company using this system to have a modern browser installed on their network, and as a result this is a requirement for use. Older versions of Internet Explorer are not supported, as I have not implemented compatible Ajax that these browses can use.

Browser compatibility is less of an issue when it comes to the mobile app. The Android mobile browsers are based on the WebKit rendering engine, and so these should all appear very similar. When the WebView is used by an app, the rendering is carried out by the system browser – which is WebKit.

5.4 Scope for future versions

- Given a longer timescale and a budget, it would be possible to implement a more accurate system for measuring the sandbox sand levels. This would involve using an external Bluetooth barcode reader which would interface with a native android app similar to the current web app. This implementation could be slightly more accurate than the current method - but would add complexity to the process due to more equipment to carry for the worker, and expense to the depot.

- Completely automate the sandbox filling and measuring process – although this is of far greater scope than what is currently implemented.

- Implement a setting to set specific times that a train can arrive or leave at.

- Implement a setting to backup and restore database.

- Implement a setting to define train types and add images for those train types.

6. Testing

6.1 Testing Process

To ensure that the system meets the function and non-functional requirements, a series of tests have been performed.

As both the applications that make up the system are largely made up of data input, this will be one of the primary factors or functions that need to be tested. Once the system has received the correct data from the user, then the chance of errors or bugs occurring in the internal logic will occur far less than in the process of validation for the users input.

Below is the format that was used for the test cases.

Test ID	Test Description	Action	Expected Outcome	Actual Outcome	Pass
1	Verify login functionality works	Enter a correct username and password	Application displays login success message	Login success message displayed.	Yes

See Appendix A for test cases and outcomes.

All requirements were met.

6.2 Uncorrected Issues

- Email validation does not work so it has been disabled
- A sandbox cannot be set as empty.
- A user who is logged in on one browser can be deleted by an admin in another browser.

7. Project Management

7.1 Backups

In order to preserve the various states of development of the two applications as their implementation changed and improved, every few days of progress that was made to them would result in myself compressing the respective site into a .RAR file, and naming this file with the date the save was made. I would then store these compressed backups on my development machine, as well as emailing them to my Brookes email and personal Gmail account. The reason for emailing them is that this is probably the most secure method of storing small amounts of data, as Google will have a very secure method of data backup for its customers.

7.2 Changes from initial spec

There were a few major changes I made to the system development, between when I submitted the Interim Report in late November, until now in March.

- The mobile application was originally going to be a fully native Java android application, however due to my inexperience with Java I decided to implement the application as a WebView accessing the site built using PHP and JQuery Mobile. To the user however, it appears to be a native application.

7.3 Record of problems encountered

I made a log of various coding problems encountered during the process. Many small or obvious bugs have not been recorded however - as they would not make much sense when looked at in the future. Many serious problems involved me spending hours and hours trying to solve - eventually resulting in taking a different approach to the original flawed design.

1) Attempting to get a JavaScript function that has been echoed by Ajax to execute.

- This script also happens to be the Ajax setup code for another call.
- Using the eval() function - which executes a string as JavaScript. I could not get this to work.
- FIX: In the end I used a different approach to the Ajax calls that did not involve echoing from Ajax JavaScript's. This specifically was the scheduling area.

- FIX #2: At a later stage, for a different problem, eval() has to be used. However at this point I worked out how to implement it.

2) Producing the reports - specifically counting the averages of values for a certain train ID.

- Trying to separate SQL array results for sandboxes fault occurrence and average sand levels, into separate arrays which only contain the sandbox ID's for each Train ID. This caused many issues

- FIXED: In the end I took a different approach, and learnt about SQL joins between tables to make a much simpler query that gave me the necessary data.

8. Conclusion

In conclusion, I feel the project went fairly well. The system accomplishes what it was planned to do in a user friendly and functional manner, with a clear and logical interface. The sandbox replenishment process is reliably recorded, and a relatively clear set of reports are generated from the data collected that specify where problems with the process lie in terms of individual users, sandboxes and trains.

Some elements of the system could be more robust and feature complete. The main feature of which could be improved would be writing the mobile application as a native android app, rather than as a website designed primarily for a touch interface. This would have been a more fluid implementation in use– however; the objectives have still been met. The use of external equipment such as a Bluetooth barcode reader may be a more reliable method of getting a reliable reading of the sand levels. This would be an area that with more time could be implemented, especially if implemented along with a native android app.

Another area that could have more detail would be the reports produced by the administrative application. Currently there are only four types of report, which could be expanded upon, providing a more detailed account of how the sandbox replenishment process is operating.

The benefit of the system to the industry is not particularly clear however. Mainly due to the integration that would be needed for it work to most effectively with the custom systems that different train operators already use. However, for a small train company, this system would be well suited once it has been setup.

8.1 Deliverables

The final implementation meets all of the requirements set out towards the beginning of this report.

The actual deliverables are: **the mobile application, administrative website, MySQL database.**

Whilst the requirements have been met, given more time, these two requirements would be expanded and refined.

- Database data can be analysed and informative reports produced.
- The scheduling (entering and exiting dates and times) of trains (units) will be input into the system. This will be used to determine which units and any time based warnings the system will be displayed to the user checking the unit's sandboxes. More than one schedule can be defined per train.

8.2 Personal Development

I have learnt a great deal during the process of implementing the system, as well as the theoretical design. In terms of programming, I have learnt a great deal. Prior to beginning the project I had very little programming experience, but now feel I could learn other languages, and implement other software with the experience gained over the last 6 months. Previously I had a small amount of experience with PHP and MySQL, but now feel far more comfortable with the two languages.

I have greatly underestimated the amount of work involved, but have just about managed to complete the requirements, albeit with a few bugs here and there. The SOTS server transfer process is incredibly buggy, and was very frustrating at times – perhaps I should have purchased server hosting from a commercial company.

Time management and fair allocation of the available time to the different aspects of the project has been a key issue throughout the project. I feel that more time could have been spent on the report writing and theoretical development, rather than the practical implementation. There were a number of occasions where the limited time available was used trying to implement a specific feature in a certain manner that was beyond my ability, only for me to come the next day with a rather different approach that was far simpler but still carried out what I wanted to do.

Bibliography

Commission, Health & Safety. (1998). *The use of computers in safety critical applications*. Retrieved from Health and Safety Executive: http://www.hse.gov.uk/nuclear/computers.pdf

css3.info. (n.d.). *CSS3 Everything you need to know*. Retrieved from css3.info: http://www.css3.info/preview/

FileCart. (2013). *CSS Button Generator*. Retrieved from http://www.cssbuttongenerator.com/

gChartPhp. (2013). *gChartPhp*. Retrieved 03 2013, from Github: https://github.com/pacbard/gChartPhp

Google. (2013). *Building Web Apps in WebView*. Retrieved from Developer.Android: http://developer.android.com/guide/webapps/webview.html

Google. (2013). *Google Chart*. Retrieved from https://developers.google.com/chart/

Google. (2013, 02 25). *Microsoft SQL Server 2012 Standard*. Retrieved 02 25, 2013, from Google Shopping: https://www.google.co.uk/shopping/product/9893737594945685817?hl=en&gs_rn=8&gs_ri=psy-ab&tok=gJKqORHFGNW8F4td_hHDwQ&pq=cost%20of%20wisa%20licenceing&cp=21&gs_id=3a&xhr=t&q=cost%20of%20microsoft%20sql%20server&rlz=1C1CHMO_enGB523GB523&um=1&ie=UTF-8&authuse

JQueryTeam. (2013). *JQuery Mobile Reference*. Retrieved from http://jquerymobile.com/

Maciaszek, L. A. (2004). *Requirements Analysis and System Design.*

Osborne, H. (2013, March). Retrieved from SMS Mobile Application: http://sots.brookes.ac.uk/~09034276/mobile/

Rail Accident Investigation Branch. (2011, November). *Rail Accident Report.* Retrieved 01 05, 2013, from RAIB: http://raib.gov.uk/cms_resources.cfm?file=/111117_R182011_Stonegate.pdf

Sussex, BBC News. (2011, November 17). Sussex train overshot station due to poor maintenance.

The Independant. (2012, July 06). Rail firm fined for commuter train 2.5 mile overshoot. London.

This is Sussex. (2012, November 22). Train overshoots Lingfield station platform by six carriages. East Grinstead.

TimePicker. (n.d.). *TimePicker*. Retrieved from http://jonthornton.github.com/jquery-timepicker/

w3c schools. (2013, January). Browser Display Statistics.

Worx International. (2013). *PHPMailer*. Retrieved from http://phpmailer.worxware.com/

Appendix A – Use case descriptions

Mobile App - Use Case Scenarios

Use Case	**1) Login to Application.**
Description	Authenticates the actor who wishes to login, and grants access to the inner-application functions.
Actors	Admin, User, Engineer
Precondition	Actor has a valid username and password associated with the system.
Trigger	Actor types in username and password and taps login.
Main flow	1. The application presents two input boxes - username and password. 2. Actor enters using the android keyboard the username and password. 3. Actor taps the login button. 4. The application validates these credentials and if correct and allows actor access to the system.
Alternative flows	If the wrong username or passwords are entered, an error message is displayed notifying the actor of this.
Post conditions	If the use case was successful the actor is now logged in. Else the system state is that of an error message.

Use Case	2) View Schedule.
Description	Displays all the trains that are registered to the actor. This does not take into account the trains scheduling. It gives an overview of the coming day's tasks so the actor is prepared.
Actors	Admin, User, Engineer.
Precondition	Actor is logged in to the application.
Trigger	Actor taps the Schedule button on the main menu, or the footer navigation area.
Main flow	1. The list of all the trains and their useful details is displayed to the actor.
Alternative flows	None
Post conditions	None

Use Case	3) View Train and Sandbox List
Description	Displays
Actors	Admin, User, Engineer
Precondition	Actor is logged in to the application.
Trigger	Actor taps the Check Sandboxes button on the main menu, or the footer navigation area.
Main flow	1. All trains that are registered to the actor are displayed in Ascending order based on when they are set to leave. 2. All the sandboxes for these trains are placed in a nested list which is activated when the train is tapped.

	3. When one of these sandboxes is tapped. Check sandbox form is displayed (**use case 4**). 4. The current time and date, the number of sandboxes left to check is displayed. 5. Blue train items mean an unchecked train, white means checked. The time left until the train should depart is displayed on the train item. 6. Grey sandbox items mean an unchecked sandbox, white means checked.
Alternative flows	1. Message displayed explaining there are no trains to check on that day. 2. Message displayed explaining actor has no registered trains.
Post conditions	Once a train has been checked, it will stay in the list for 30 minutes, after which it is no longer displayed.

Use Case	**4) Check Sandbox**
Description	Allows the actor to input the level of sand in the sandbox that was tapped (**in use case 3**). Also allows a defect to be noted, along with a description.
Actors	Admin, User, Engineer
Precondition	Actor is logged in to the application.
Trigger	A sandbox has been tapped from the Train and sandbox list.
Main flow	1. Actor notes what the level of sand is/was in the sandbox prior to refilling it. This is expressed on a scale of 0-100. It can be entered using the slider, or using the numeric keyboard. 2. Actor can notify the system that there is a fault with the sandbox. If they do this a comment must be left explaining the fault to some extent. 3. Actor taps submit or go back. Submit validates and saves the data. Go back will send the user back to the train and sandbox list.
Alternative	1. A validation message is displayed telling the user to input the level of

flows	sand. 2. If a defect is noted but no comment is left, a validation message tells the actor to do so, otherwise the data will not save.
Post conditions	Database has been updated. Or no changes have been made to database.

Web Application Use cases

Use Case	**1) Login to Website.**
Description	Authenticates the actor who wishes to login, and grants access to the inner-application functions.
Actors	Admin, User, Engineer
Precondition	Actor has a valid username and password associated with the system.
Trigger	Actor types in username and password and taps login.
Main flow	1. The application presents two input boxes - username and password. 2. Actor enters using the android keyboard the username and password. 3. Actor taps the login button. 4. The application validates these credentials and if correct and allows actor access to the system.
Alternative flows	If the wrong username or passwords are entered, an error message is displayed notifying the actor of this.
Post conditions	If the use case was successful the actor is now logged in. Else the system state is that of an error message.

Use Case	2) View Data and Reports
Description	Gives access to search data, list data,
Actors	Admin, User, Engineer
Precondition	Actor is logged in to the application.
Trigger	The menu option for “View Data and Reports” is clicked from any page.
Main flow	1. The View Data and Reports screen is displayed in the content area. 2. Buttons for View Data in List, Visualized Data are displayed.
Alternative flows	None.

Use Case	3) View List Data
Description	Allows actor to view the different types of data. **Users, train types, trains, sandboxes, repairs**
Actors	Admin, User, Engineer
Precondition	Actor is logged in to the application. Actor is on the View Data and Reports screen.
Trigger	The button for “View Data in List” is clicked from the “View data and reports” section.
Main flow	1. Actor selects which list of data they wish to display from combo box. 2. Actor clicks view. 3. Screen fetches via Ajax the list of data specified from the database and displays it to the user.
Alternative flows	1. Actor selects which list of data they wish to display from combo box. 2. Actor clicks view. 3. Systems displays message indicating there is no data for requested list.

Post conditions	None

Use Case	**4) Visualized Reports**
Description	Allows actor to view different reports.
Actors	Admin, User
Precondition	Actor is logged in to the application. Actor is on the View Data and Reports screen.
Trigger	The button for “Visualized Reports” is clicked from the “View data and reports” section.
Main flow	1. Actor selects which type of report they wish to view. 2. Actor clicks get charts. 3. Database information is requested and processed, and then sent to to Google Charts to generate chart image. 4. Screen fetches via Ajax the chart image and displays it to the user.
Alternative flows	1. Actor clicks get charts for a specific type of report. 2. Message notifies actor that there is not enough data to generate report.
Post conditions	None

Use Case	**5) Setup Users**
Description	Allows the actor to add users to the system, modify them or delete them.
Actors	Admin

Precondition	Actor is logged in to the application.
Trigger	Setup Users button is clicked from the navigation menu.
Main flow	**Add user** 1. Actor clicks Add New User. 2. Form is displayed requesting user information. 3. Actor fills in form and clicks ADD. 4. If any fields are missing or invalid, actor is notified. 5. System adds data to database. 6. System confirms the user is added. **Modify user** 1. Actor selects a surname of a user from the combo box 2. Modify selected user button is clicked 3. Form is displayed requesting modified user information. 4. Actor fills in form and clicks Modify. 5. If any fields are missing or invalid, actor is notified. 6. System updates data in database. 7. System confirms user data modification. **Delete user** 1. Actor selects a surname of a user from the combo. 2. Delete selected user button is clicked 3. If the actor has selected themselves, an error is displayed. 3. Actor confirms the user should be deleted. 4. System deletes data in database. 5. System confirms user data has been deleted.
Alternative flows	**Delete user** 1. Actor selects themselves from the combo box . 2. Delete selected user button is clicked 3. System displays error, cannot delete logged in user.
Post	User data has been added, updated or deleted. OR no changes have been made.

conditions	

Use Case	**6) Engineers Panel**
Description	Allows Actor to view faults, view repaired faults and confirm fault repaired.
Actors	Admin, Engineer
Precondition	Actor is logged in to the application.
Trigger	Engineer panel button is clicked from the navigation menu.
Main flow	1. Sandbox faults are displayed in a list. 2. Actor selects from combo box a repair ID. 3. Actor clicks report a fault fixed button. 4. Actor leaves repair comment. 5. Actor clicks confirm repair complete button. 6. System confirms repair complete.
Alternative flows	1. Actor clicks view fixed faults 2. Fixed faults are displayed in list format. 1. System notifies actor there are no faults logged. 1 .System notifies actor there are no repaired faults logged.
Post conditions	Fault has been reported repaired. OR No changes made.

Use Case	**7) Setup Trains and Scheduling**
Description	Allows Actor to Add Train, Change User-Train Responsibility, Setup Scheduling, Edit or Delete a train.
Actors	Admin, User

Precondition	Actor is logged in to the application.
Trigger	Setup Trains and Scheduling button clicked in navigation menu.
Main flow	**Add Train** 1. Actor clicks Add Train. 2. Form is displayed requesting train information. 3. Actor fills in form and clicks ADD. 4. If any fields are missing or invalid, actor is notified. 5. System adds train to database. 6. System confirms the train is added. **Change user-train responsibility** 1. Actor selects a Train ID / Name from the upper combo box. 2. User that train is currently registered to is displayed. 3. Actor selects the User Id to register to train to from the lower combo box 4. Actor clicks Make Changes. 5. System updates data in database. 6. System responds with success confirmation. **Setup Scheduling** 1. Actor selects a Train ID / Name from the combo box. 2. Actor clicks Lookup. 3. Schedule #1 for that train is displayed along with message indicating if the train is currently in the depot or not. 4. Actor clicks Yes in response to confirmation to reschedule train. 5. Actor enters new entry time, date, vacate time, date. 6. Actor clicks apply schedule. 7. Schedule updated in database. 8. System confirms schedule applied and displays it. 8. Actor clicks Enter schedule #2. 9. Actor enters new entry time, date, vacate time, date. 10. Actor clicks apply schedule. 11. Schedule updated in database. 12. System confirms schedule applied and displays it.

	Edit/Delete train 1. Actor selects a Train ID / Name from the left combo box. 1. Actor selects Edit from the right combo box. 3. Edit form is displayed. 4. Actor fills in form and clicks confirm. 5. Database updated with form values. 6. Confirmation message displayed. 7. Actor selects a Train ID / Name from the left combo box. 8. Actor selects delete from the right combo box. 9. Yes or No Confirm to delete displayed. 10. Actor clicks yes. 11. Train deleted from database. 12. Confirmation message displayed.
Alternative flows	**Setup Scheduling** 1. Actor enters and applies a schedule with the entry dates after the vacate dates. 2. Error displayed. 1. Actor enters and applies a schedule#2 with the entry dates after the vacate dates prior to schedule #1. 2. Error displayed.
Post conditions	Train data has been added, updated or deleted. OR No changes made.

Appendix B - Testing

Mobile App

Test ID	Test Description	Action	Expected Outcome	Actual Outcome	Pass
1	Verify login functionality works	Enter a correct username and password	Application displays login success message	Login success message displayed.	Yes
2	Verify the correct ID and account type is displayed on successful login.	Login successfully as user with ID of 5 and type 'admin'	Your ID is 5. Your account level is admin.	Your ID is 5. Your account level is admin.	Yes
3	Verify "Sandboxes UNCHECKED"" is accurate.	Leave 6 sandboxes unchecked.	Sandboxes UNCHECKED = 6	Sandboxes UNCHECKED = 6	Yes
4	Verify displayed time and date is accurate.	Login successfully, at 18:06, 27/03/13	18:06:30, Wednesday 27th March 2013	18:06:30, Wednesday 27th March 2013	Yes
5	Ensure Checks Incomplete message is accurate	Leave sandbox(s) unchecked.	"Checks INCOMPLETE"	"Checks INCOMPLETE"	Yes
6	Ensure Checks COMPLETE message is accurate.	Check all sandboxes.	"Checks COMPLETE"	"Checks COMPLETE"	Yes
7	Verify the logout button works.	Whilst logged in, tap logout.	Returned to login screen.	Returned to login screen.	Yes
8	Verify the correct sandbox check form is displayed.	Tap Sandbox #1 (id: 363) of Train SW5421 (id: 220)	Sandbox #363, Train #220 at top of form.	Sandbox #363, Train #220 at top of form.	Yes
9	Verify sandbox check form validation functions correctly.	In a previously unchecked sandbox form, leave all fields at	Are you sure the sandbox is/was empty?	Are you sure the sandbox is/was empty?	Yes

	#1	default and tap submit.			
10	Verify sandbox check form validation functions correctly. #2	In a previously unchecked sandbox form, set the sand level, and set fault to yes. Tap submit.	Please leave a description of the fault.	Please leave a description of the fault.	Yes
11	Verify sandbox check form validation functions correctly. #3	In a previously unchecked sandbox form, set the sand level, and leave a fault comment. Tap submit.	You cannot leave a comment if there is no fault.	You cannot leave a comment if there is no fault.	Yes
12	Verify sandbox check form validation functions correctly. #4	In a previously unchecked sandbox form, set the sand level.	Returned to train list.	Returned to train list.	Yes
13	Verify sandbox check form validation functions correctly. #5	In a previously unchecked sandbox form, set the sand level, fault? to yes, and leave a fault comment.	Returned to train list.	Returned to train list.	Yes
14	Verify sandbox check form validation functions correctly. #6	In a previously unchecked sandbox form, set the sand level to 0.	Returned to train list.	Are you sure the sandbox is/was empty?	No
15	Verify previously checked sandbox has form data saved.	Tap a checked sandbox that was set to 54 on the slider.	Sandlevel: 54	Sandlevel: 54	Yes
16	Verify sandbox list item turns white once checked. (from grey)	Check a sandbox.	The sandbox that was checked appears white in the tran/sandbox list.	The sandbox is white.	Yes

17	Verify train list item turns white once checked. (from blue)	Check all sandboxes for a train.	The train list item becomes white.	The train list item becomes white.	Yes
18	Verify Schedule #2 is applied after Schedule #1 has passed.	Enter a Schedule #2 and wait for the Schedule #2 to begin	Appears in train list.	Appears in train list.	Yes
19	Verify after 30 minutes, a checked train is removed from the train list.	Check a train, wait 30 minutes.	Train no longer appears.	Train no longer appears.	Yes
20	Verify changing train name in web application takes effect on train list.	Change train name in admin site.	Train name changes in train list.	Train name changes in train list.	Yes
21	Verify changing train type in web application takes effect on train list.	Change train type in admin site.	Train type changes in train list.	Train type changes in train list.	Yes

Web Application

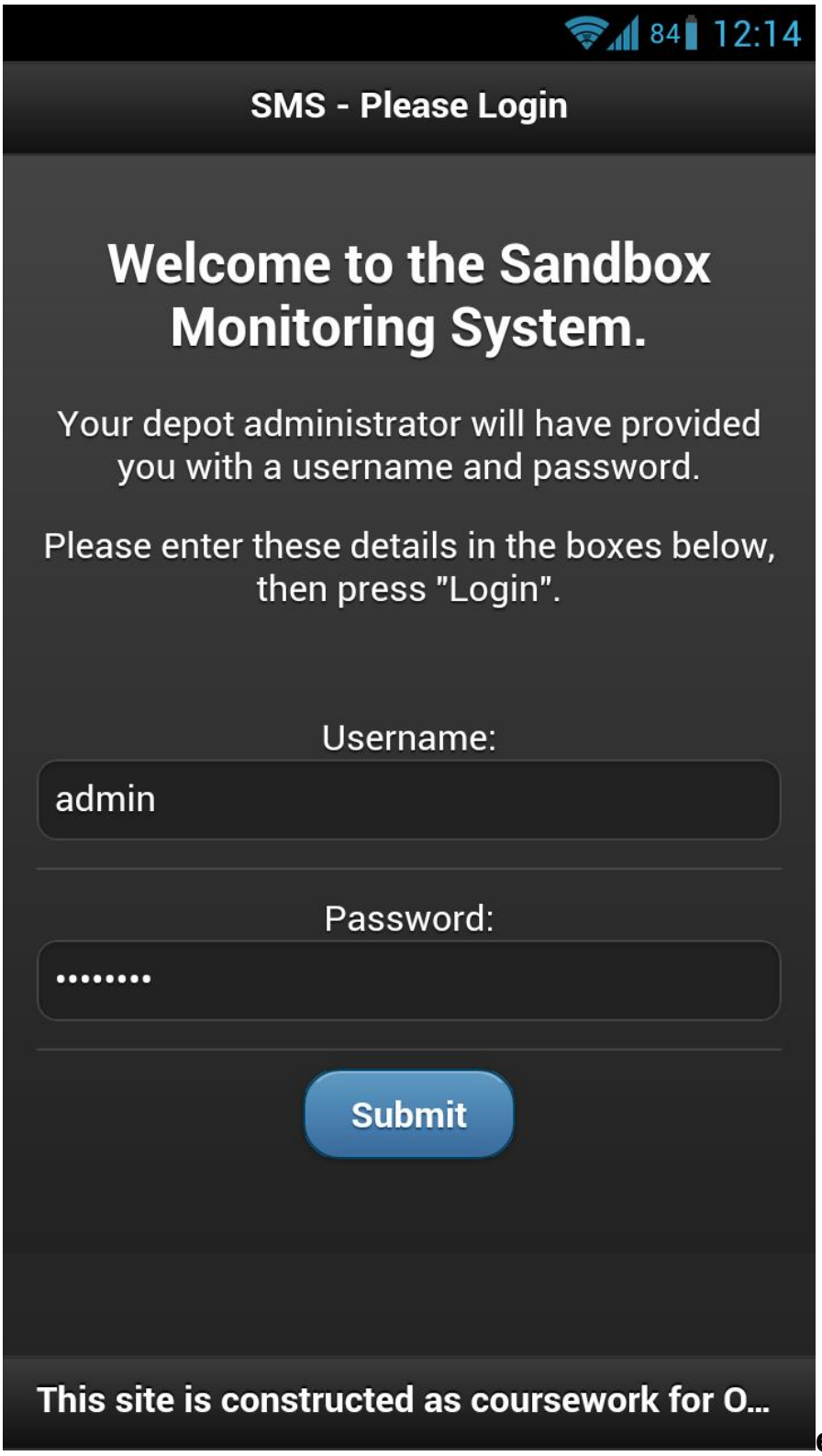
Test ID	Test Description	Action	Expected Outcome	Actual Outcome	Pass
1	Verify access to any system function cannot be accessed when logged out and modifying the URL to the address.	Enter http://sots.brookes.ac.uk/~09034276/train/index.php?view=edit (for each where view = whats in the layout folder. WHEN logged out.	Access Denied.	Access Denied.	Yes.
2	Add a train, don't enter train name	Add a train, fill in all fields except train name.	Train name is required.	Train name is required.	Yes
3	Add a train enter an already used train name.	Add a train, fill in all fields including train name that has been added previously.	Please use a different Train Name	Please use a different Train Name	Yes

4	Add a train, enter no fields.	Add a train, enter no fields.	Train name is required. Entry is required. vacate is required. entry d, vacate d is required.	Train name is required. Entry is required. vacate is required. entry d, vacate d is required.	Yes
5	Verify change user-train responsibility works.	Change a trains user to a different user. Check view train list.	That train has the user changed.	That train has the user changed.	Yes
6	Verify setup schedule – schedule #1 apply works.	Change schedule #1. Schedule #1 should change to new schedule.	Schedule applied correctly.	Schedule applied correctly.	Yes
7	Verify setup schedule – schedule #2 apply works.	Change schedule #2. Schedule #2 should change to new schedule.	Schedule applied correctly.	Schedule applied correctly.	Yes
8	Verify edit train works	Edit the train. Reselect the train.	Edit has taken affect.	Edit has taken affect.	Yes
9	Verify delete train works	Delete a train. Reselect the train.	Train already deleted.	Train already deleted.	Yes
10	View list data.	View each type of list with no data in them.	No data to list.	No data to list.	Yes
11	View reports	View each report type with no data in database. Except data for user that is logged in.	Not enough data to produce report.	Not enough data to produce report.	Yes
12	Verify settings work.	Change each setting.	Settings have taken affect.	Settings have taken affect.	Yes
13	Engineer – view with no data.	Click engineer panel with no data in repair table.	No unfixed faults.	No unfixed faults.	Yes
14	Engineer – view fixed faults with no	Click engineer panel, view fixed faults with no data in	No fixed faults.	No fixed faults.	Yes

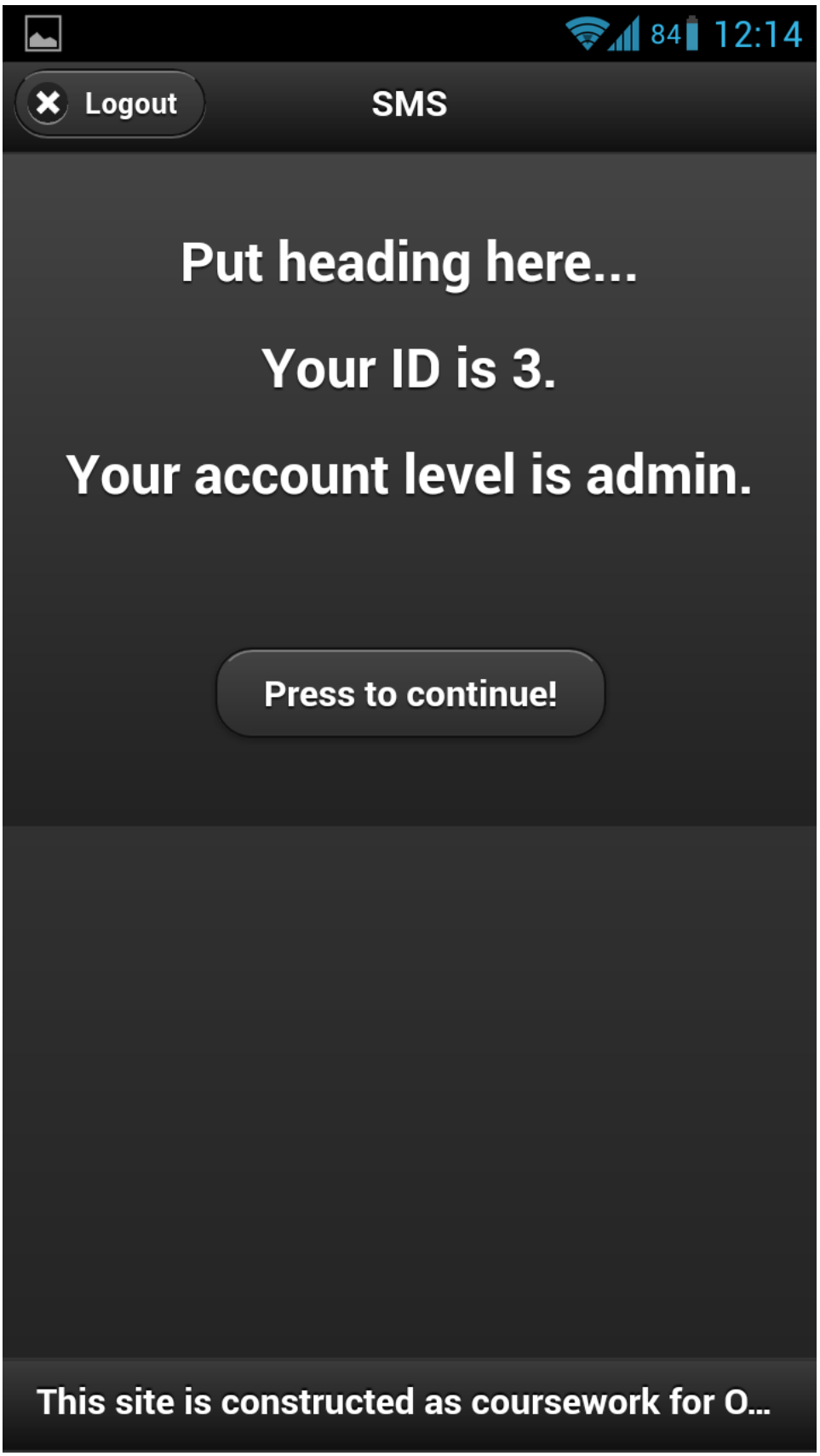
	data.	repair table.			
15	Engineer – report a fault fixed – very affect.	Reports a fault fixed and leave a comment.	Repair taken out of unfixed faults. Placed in fixed faults.	Repair taken out of unfixed faults. Placed in fixed faults.	Yes
16	Add user.	Add a new user with all fields correct.	User added.	User added.	Yes
17	Add User – verify validation	Add a user with all different fields either too long >15 or not present.	Correct validation message displayed.	Correct validation message displayed. Except for email.	No (email validation failed)
18	Modify user	Modify a user with all fields correct.	User modified	User modified.	Yes
19	Modify User – verify validation	Modify a user with all different fields either too long >15 or not present.	Correct validation message displayed.	Correct validation message displayed. Except for email.	No (email validation failed)
20	Delete a user	Delete a user	User deleted	User deleted	Yes
21	Delete user that you are logged in as.	Delete user that you are logged in as.	Cannot delete the user you are logged in as.	Cannot delete the user you are logged in as.	Yes
22	Delete other user that is logged in on another browser.	Delete other user that is logged in on another browser.	User deleted.	User deleted.	Yes
23	Verify logout works	Click logout.	Returned to home screen.	Returned to home screen.	Yes
24	Verify login functionality works	Enter a correct username and password	Home screen displayed. Menu changes to logged in.	Home screen displayed. Menu changes to logged in.	Yes

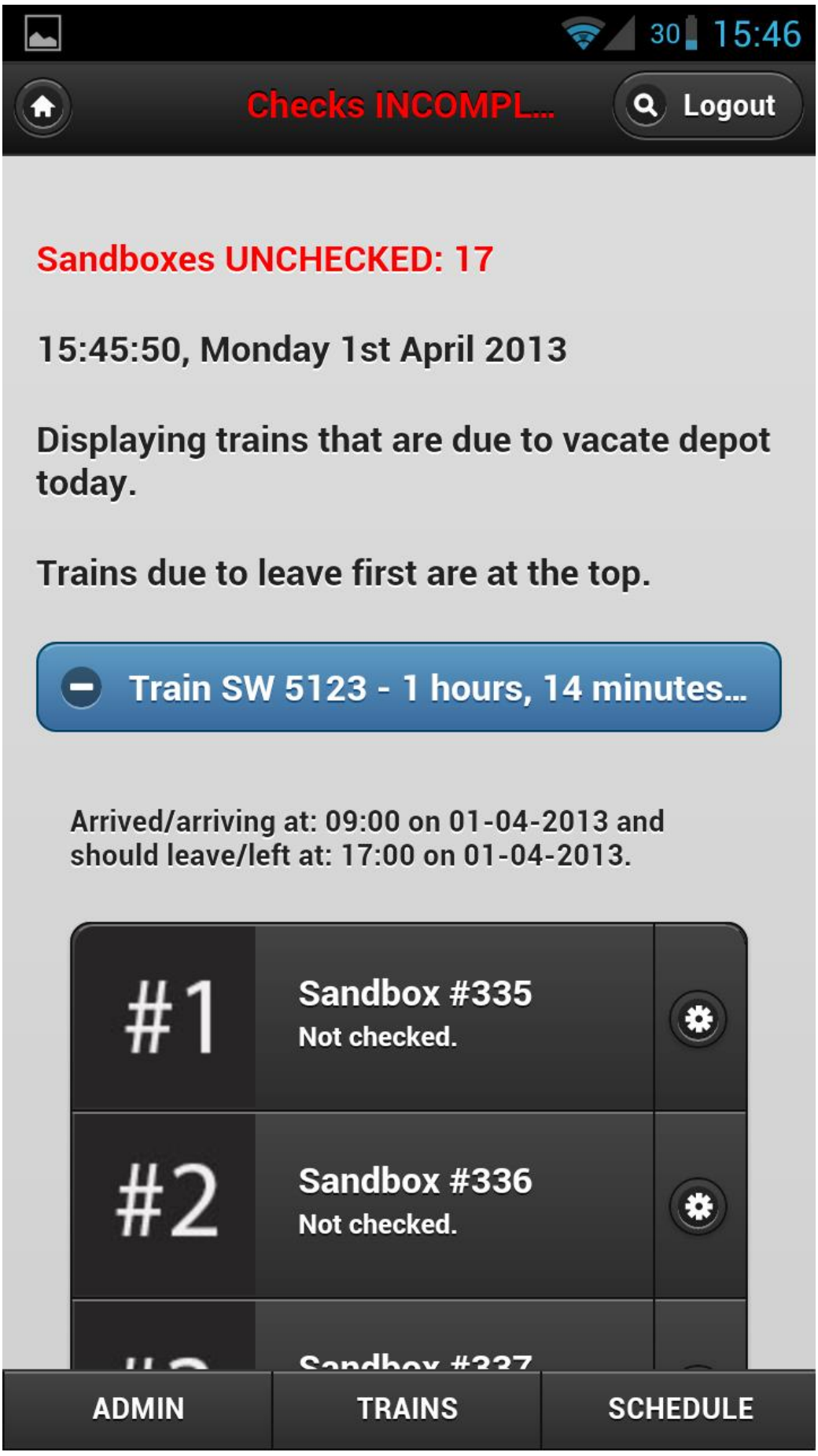
Appendix C – Screenshots

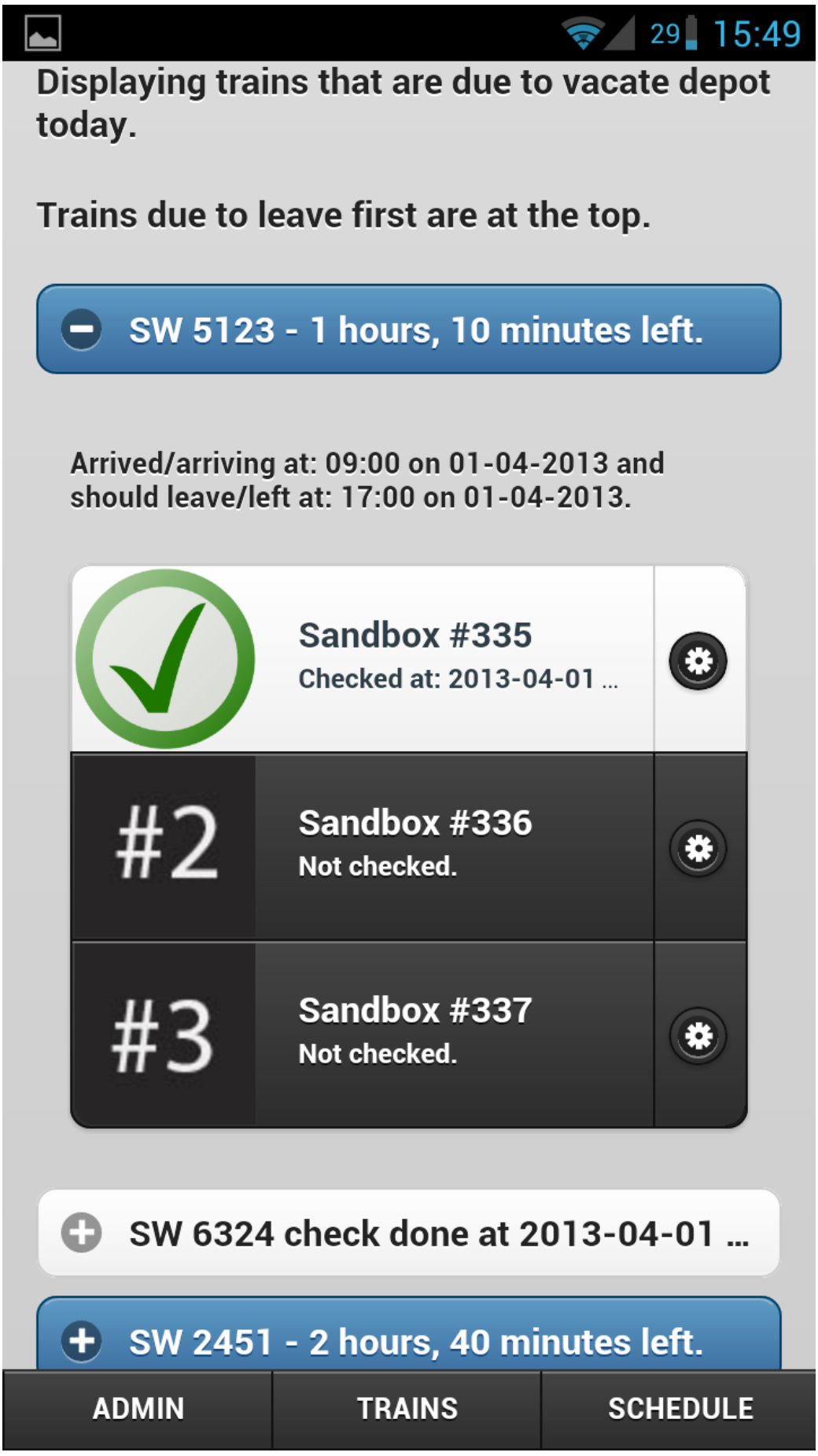
Mobile Application Scre

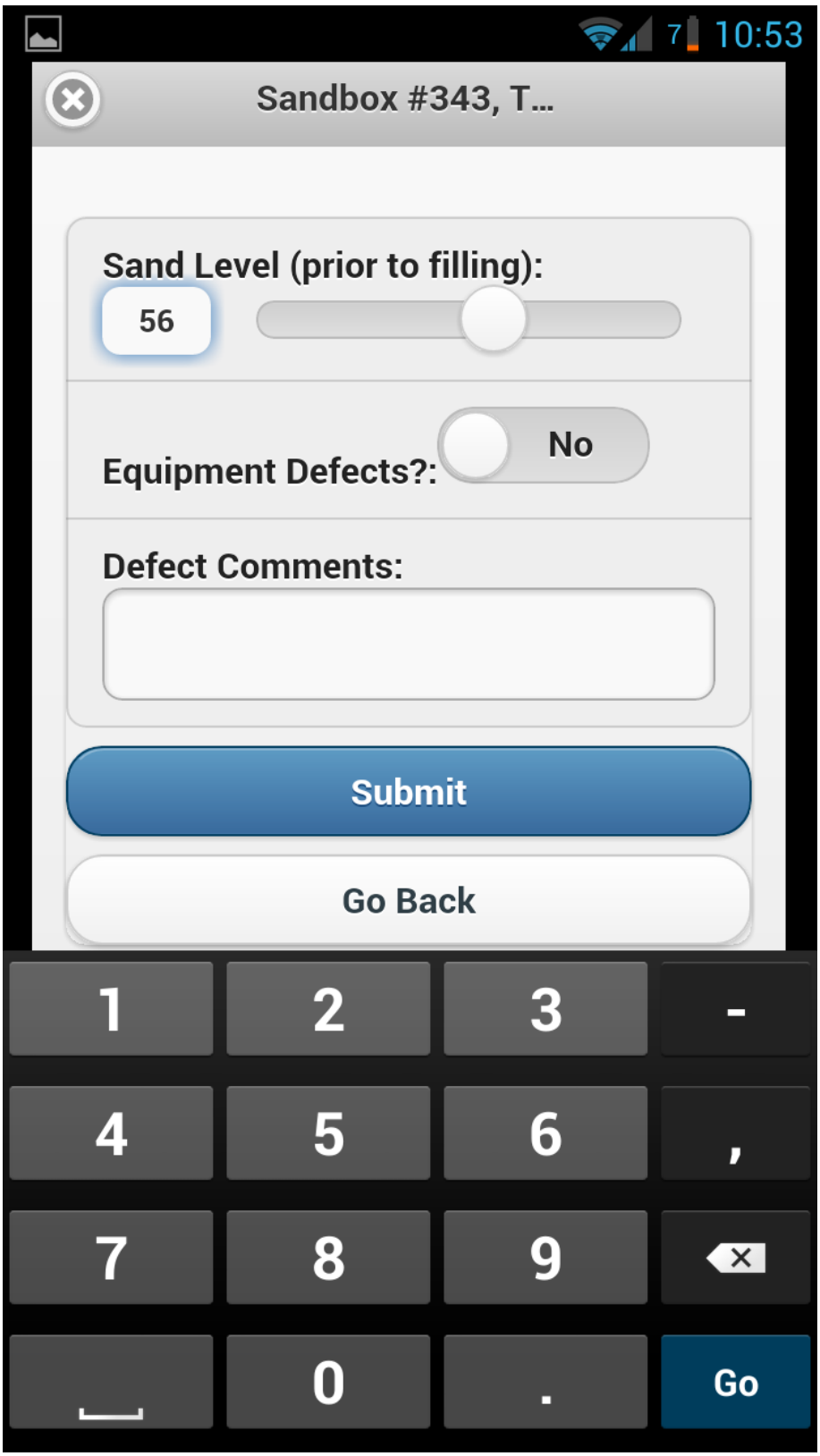


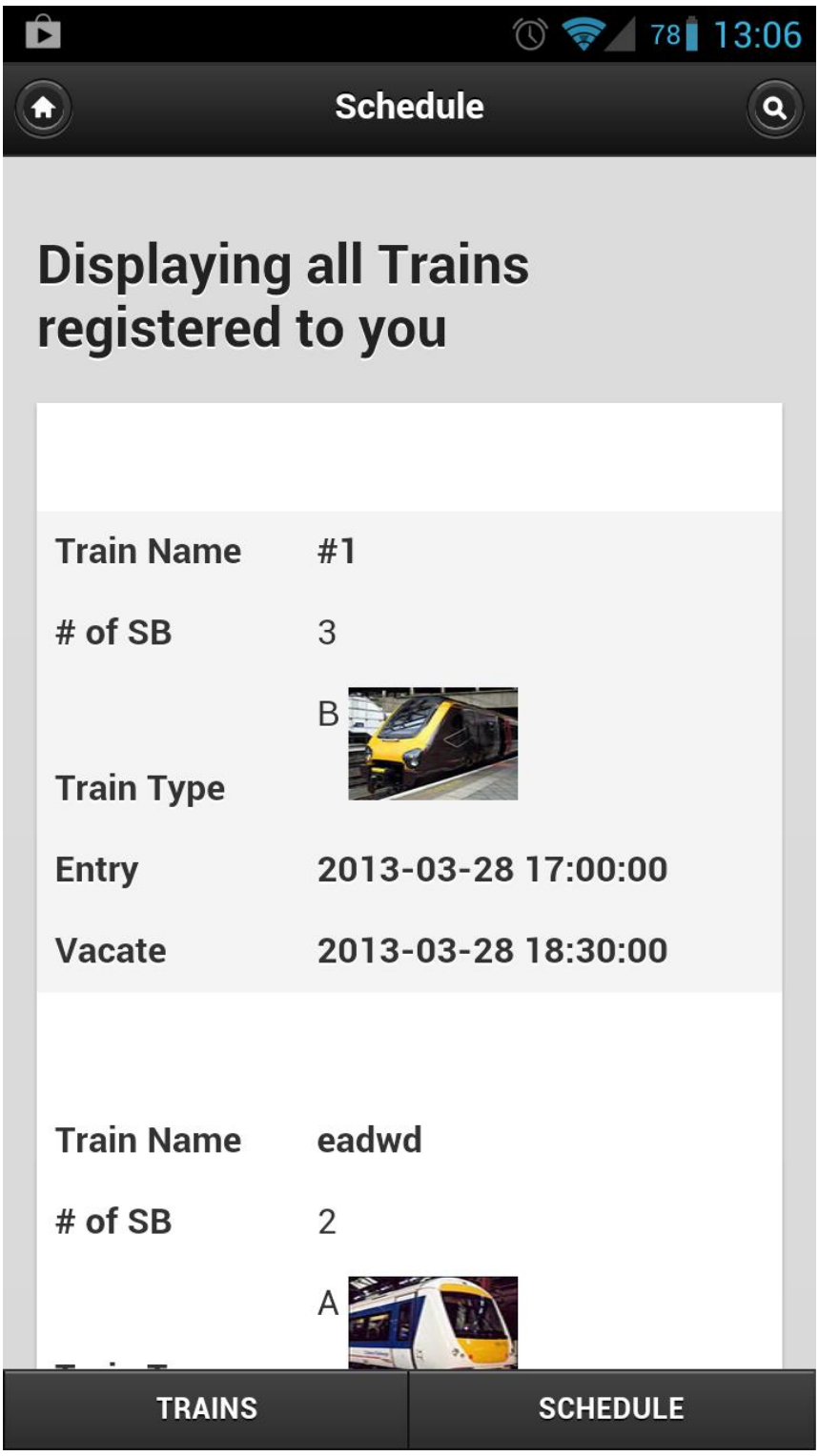
enshots

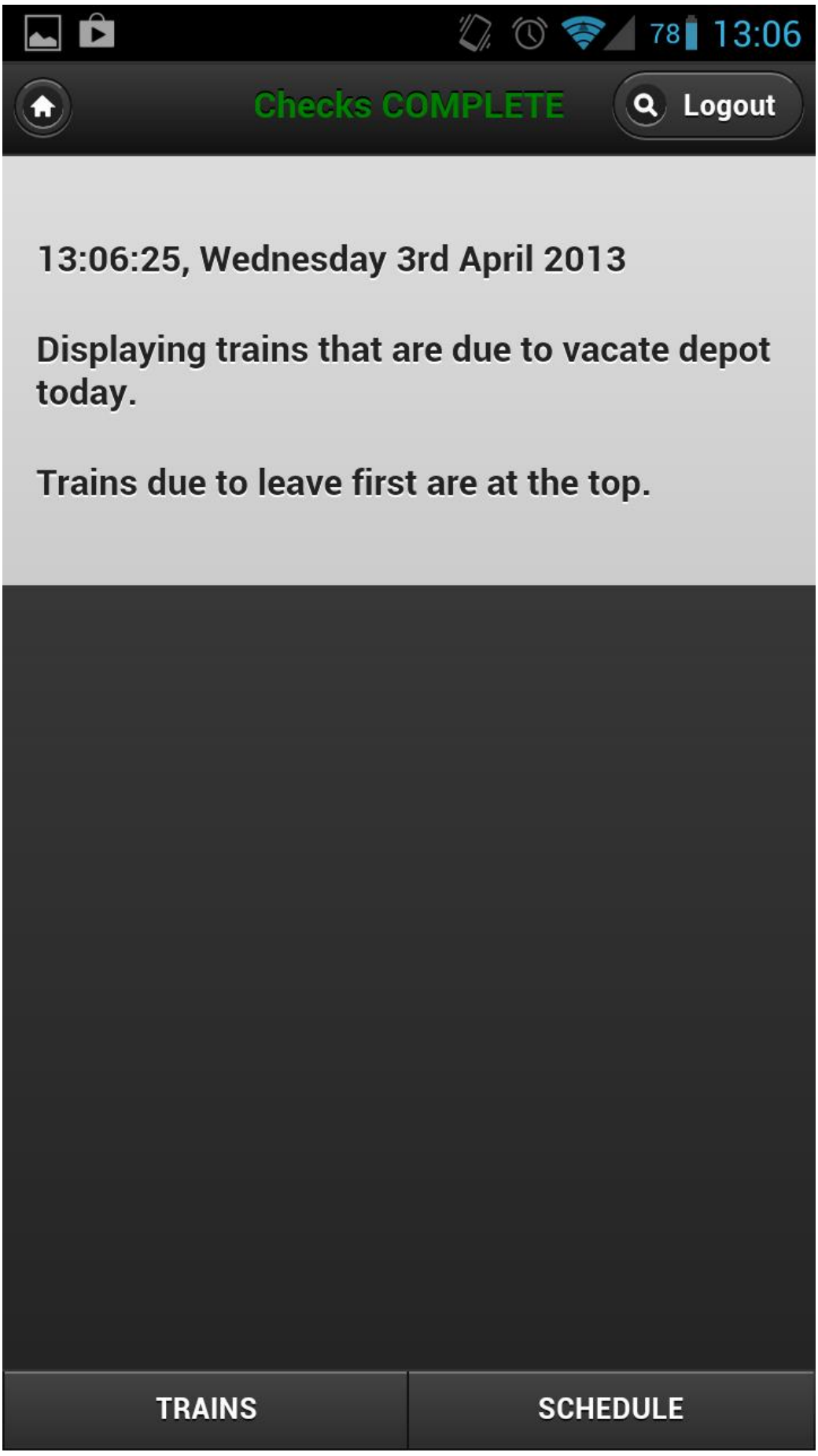

84 12:14
Logout
SMS
Put heading here...
Your ID is 3.
Your account level is admin.
Press to continue!
This site is constructed as coursework for O…


30 15:46
Checks INCOMPL...
Logout
Sandboxes UNCHECKED: 17
15:45:50, Monday 1st April 2013
Displaying trains that are due to vacate depot today.
Trains due to leave first are at the top.
Train SW 5123 - 1 hours, 14 minutes...
Arrived/arriving at: 09:00 on 01-04-2013 and should leave/left at: 17:00 on 01-04-2013.
#1
Sandbox #335
Not checked.
#2
Sandbox #336
Not checked.
ADMIN
TRAINS
SCHEDULE

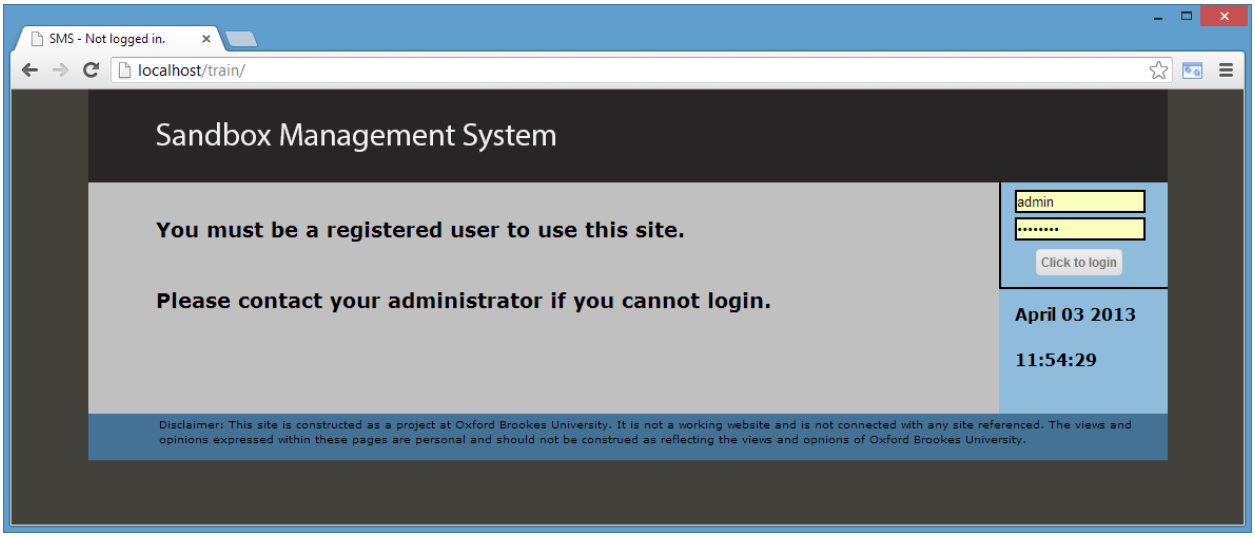

29 15:49
Displaying trains that are due to vacate depot today.
Trains due to leave first are at the top.
SW 5123 - 1 hours, 10 minutes left.
Arrived/arriving at: 09:00 on 01-04-2013 and should leave/left at: 17:00 on 01-04-2013.
Sandbox #335
Checked at: 2013-04-01 ...
#2
Sandbox #336
Not checked.
#3
Sandbox #337
Not checked.
SW 6324 check done at 2013-04-01 ...
SW 2451 - 2 hours, 40 minutes left.
ADMIN
TRAINS
SCHEDULE

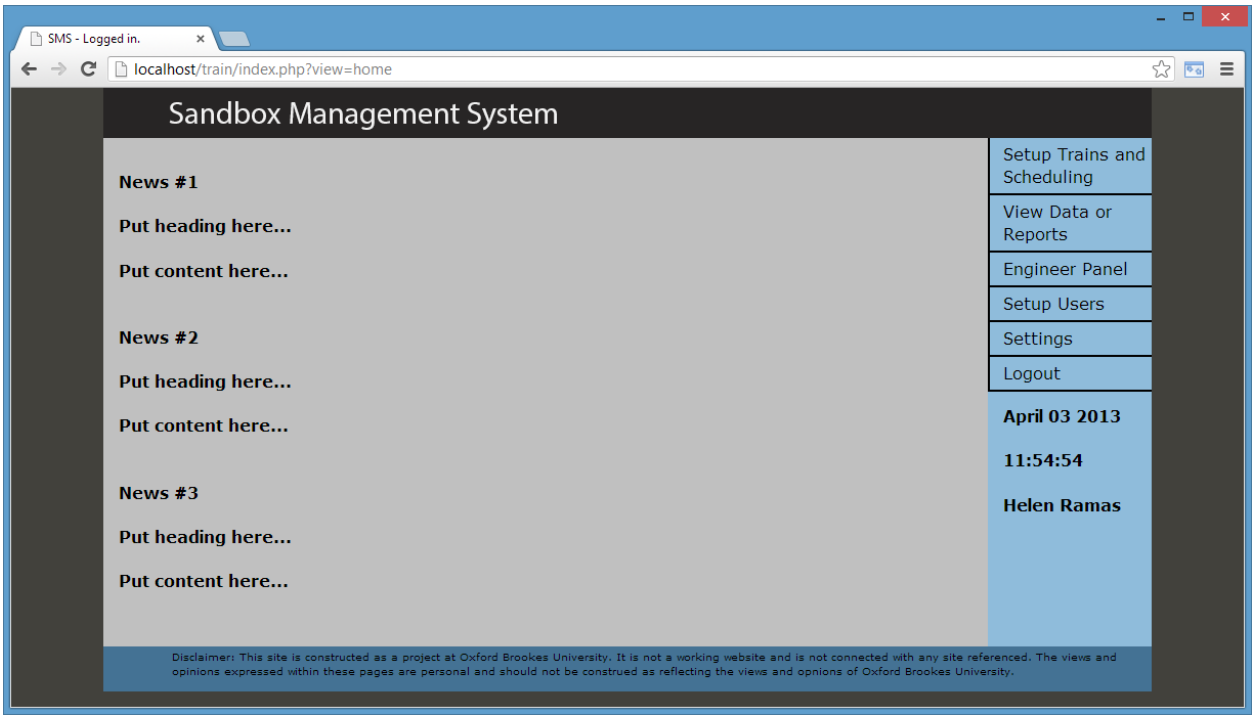

10:53
Sandbox #343, T...
Sand Level (prior to filling):
56
Equipment Defects?:
No
Defect Comments:
Submit
Go Back
1
2
3
-
4
5
6
,
7
8
9
0
.
Go

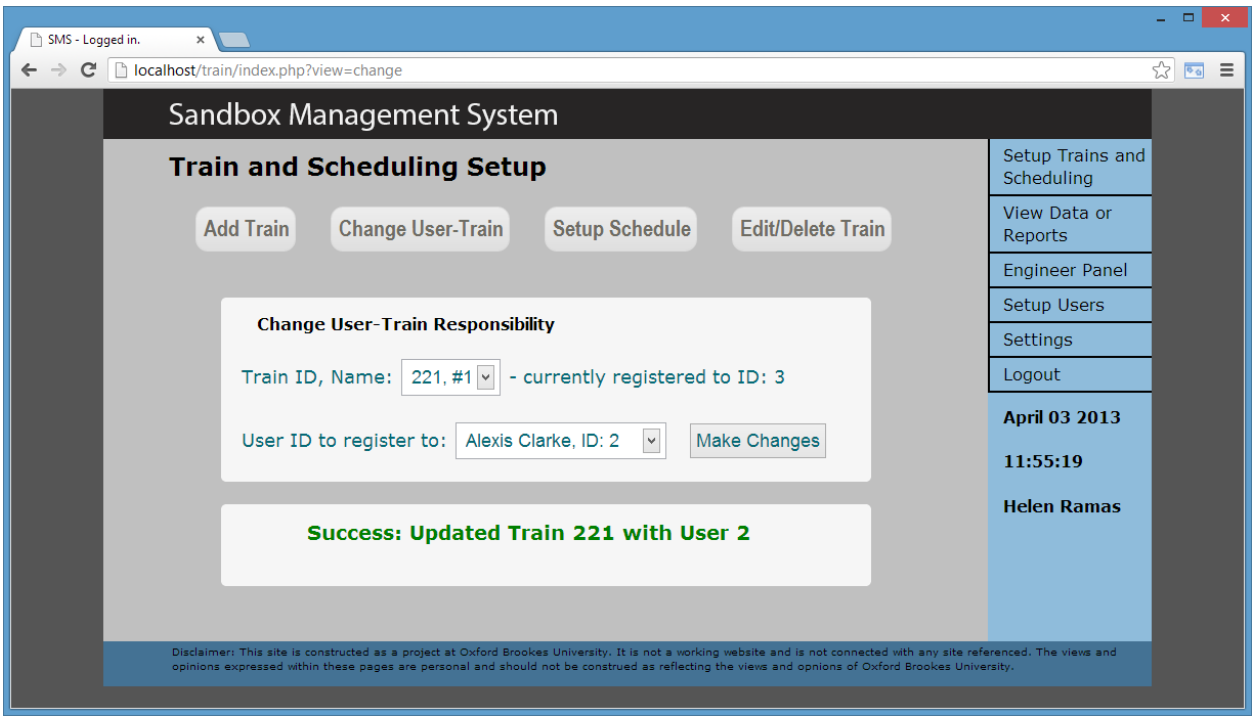

78 13:06
Schedule
Displaying all Trains registered to you
Train Name #1
# of SB 3
Train Type B
Entry 2013-03-28 17:00:00
Vacate 2013-03-28 18:30:00
Train Name eadwd
# of SB 2
A
TRAINS
SCHEDULE

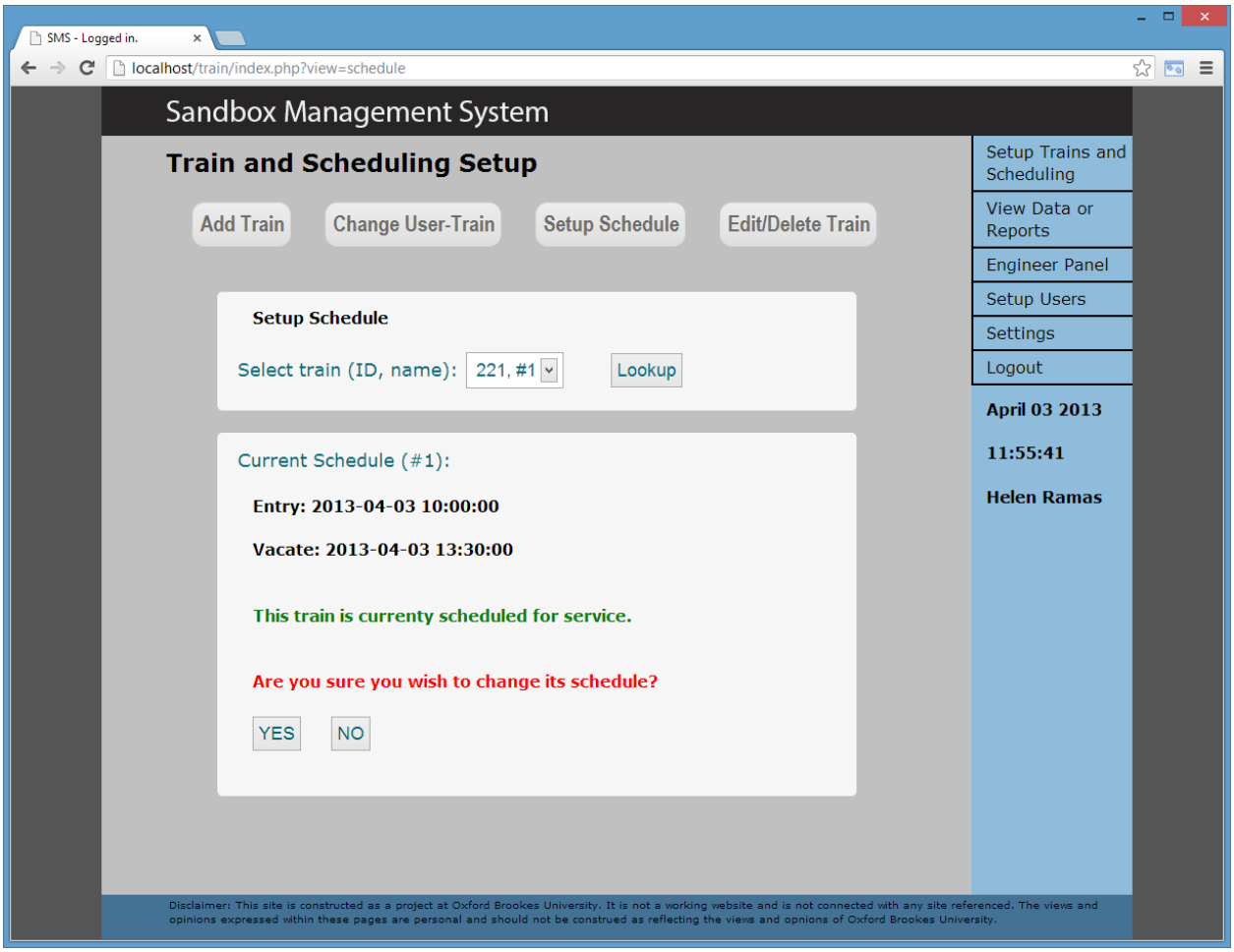

78 13:06
Checks COMPLETE
Logout
13:06:25, Wednesday 3rd April 2013
Displaying trains that are due to vacate depot today.
Trains due to leave first are at the top.
TRAINS
SCHEDULE

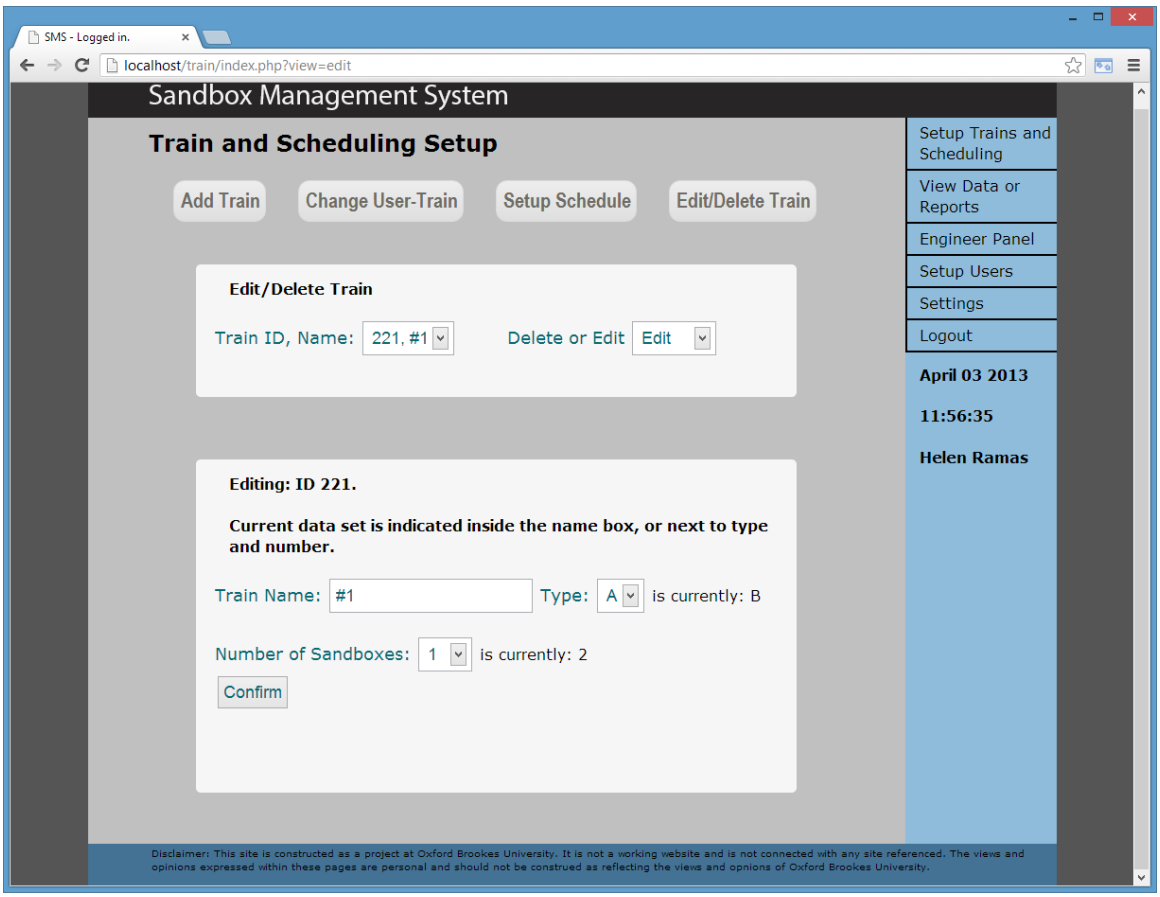
Web Application Screenshots

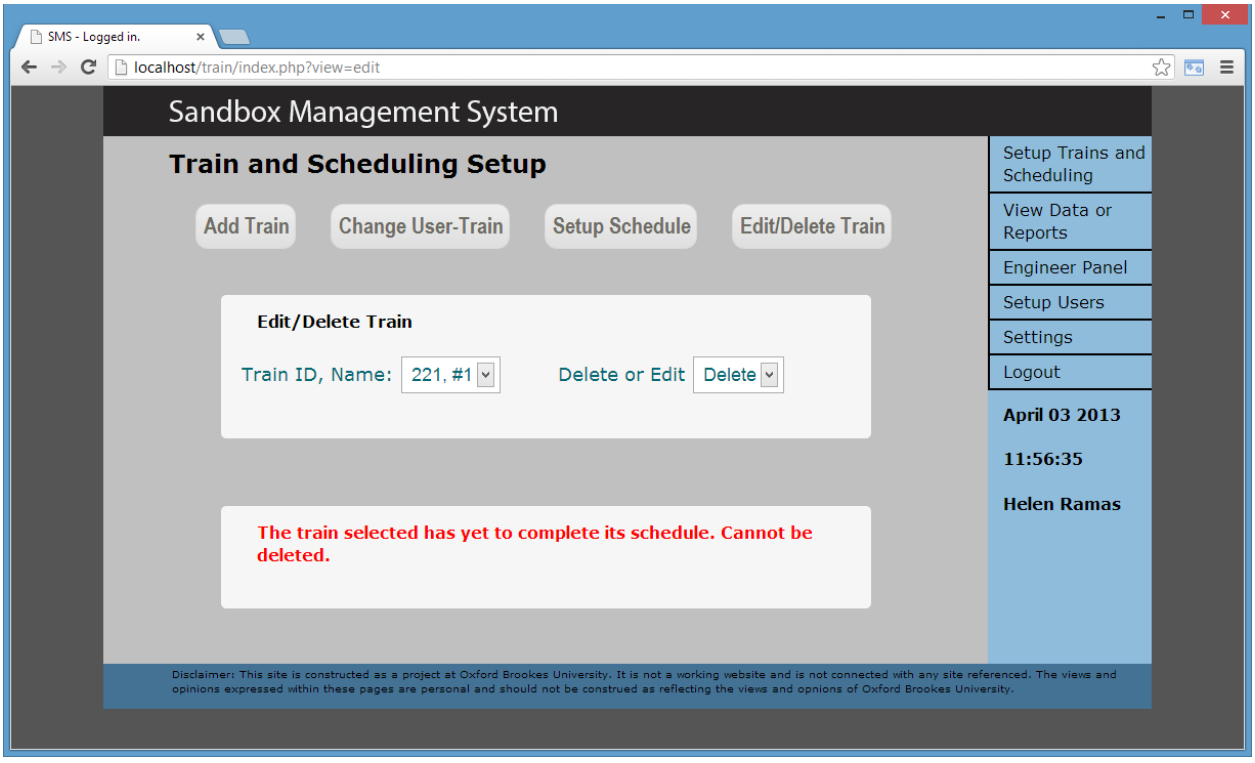


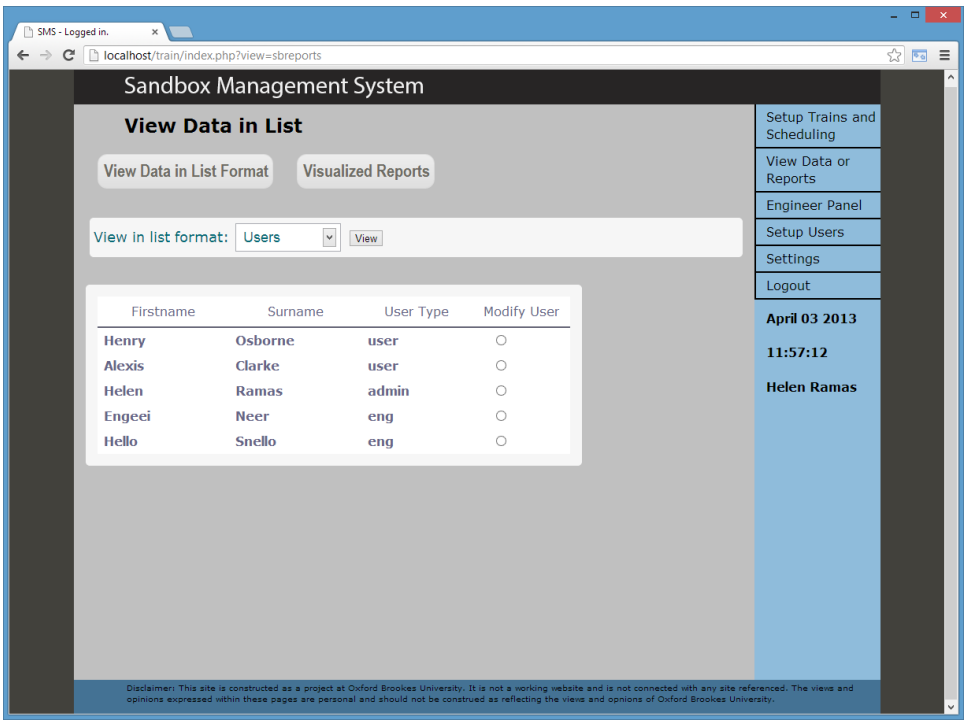


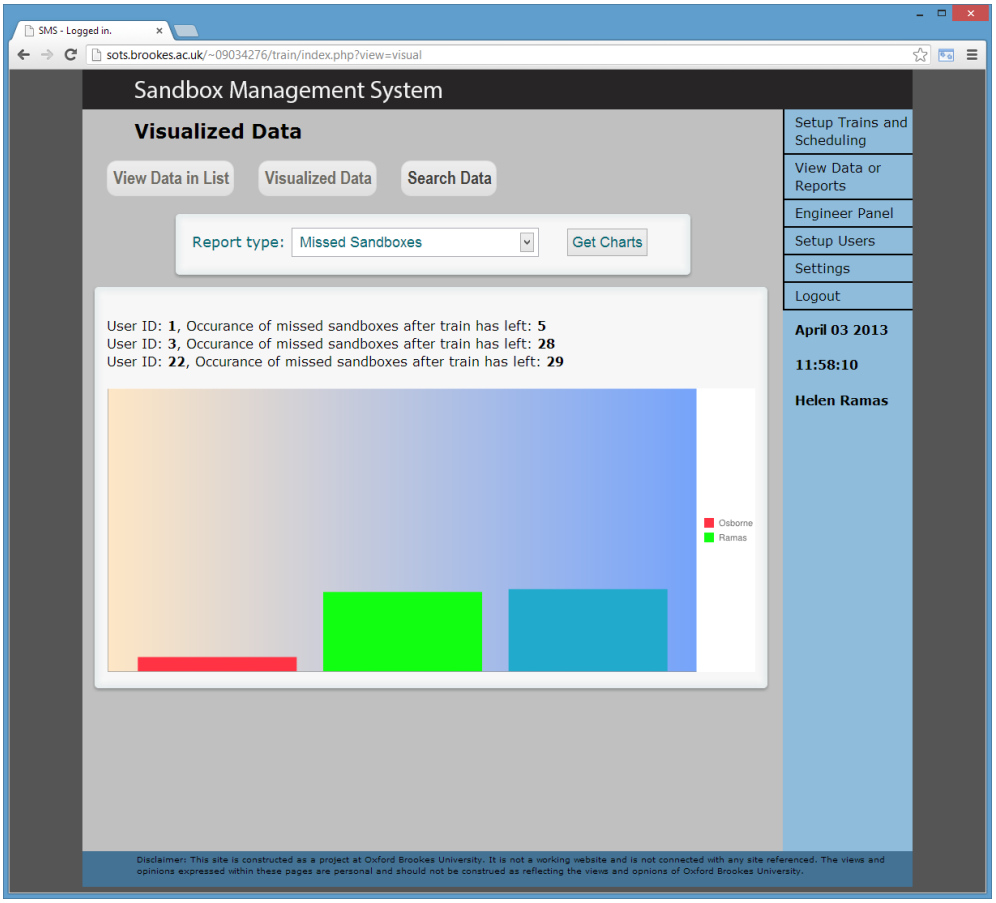

SMS - Logged in.
localhost/train/index.php?view=change
Sandbox Management System
Train and Scheduling Setup
Add Train
Change User-Train
Setup Schedule
Edit/Delete Train
Change User-Train Responsibility
Train ID, Name: 221, #1 - currently registered to ID: 3
User ID to register to: Alexis Clarke, ID: 2
Make Changes
Success: Updated Train 221 with User 2
Setup Trains and Scheduling
View Data or Reports
Engineer Panel
Setup Users
Settings
Logout
April 03 2013
11:55:19
Helen Ramas
Disclaimer: This site is constructed as a project at Oxford Brookes University. It is not a working website and is not connected with any site referenced. The views and opinions expressed within these pages are personal and should not be construed as reflecting the views and opnions of Oxford Brookes University.


SMS - Logged in.
localhost/train/index.php?view=schedule
Sandbox Management System
Train and Scheduling Setup
Add Train
Change User-Train
Setup Schedule
Edit/Delete Train
Setup Schedule
Select train (ID, name): 221, #1
Lookup
Current Schedule (#1):
Entry: 2013-04-03 10:00:00
Vacate: 2013-04-03 13:30:00
This train is currenty scheduled for service.
Are you sure you wish to change its schedule?
YES
NO
Setup Trains and Scheduling
View Data or Reports
Engineer Panel
Setup Users
Settings
Logout
April 03 2013
11:55:41
Helen Ramas
Disclaimer: This site is constructed as a project at Oxford Brookes University. It is not a working website and is not connected with any site referenced. The views and opinions expressed within these pages are personal and should not be construed as reflecting the views and opnions of Oxford Brookes University.

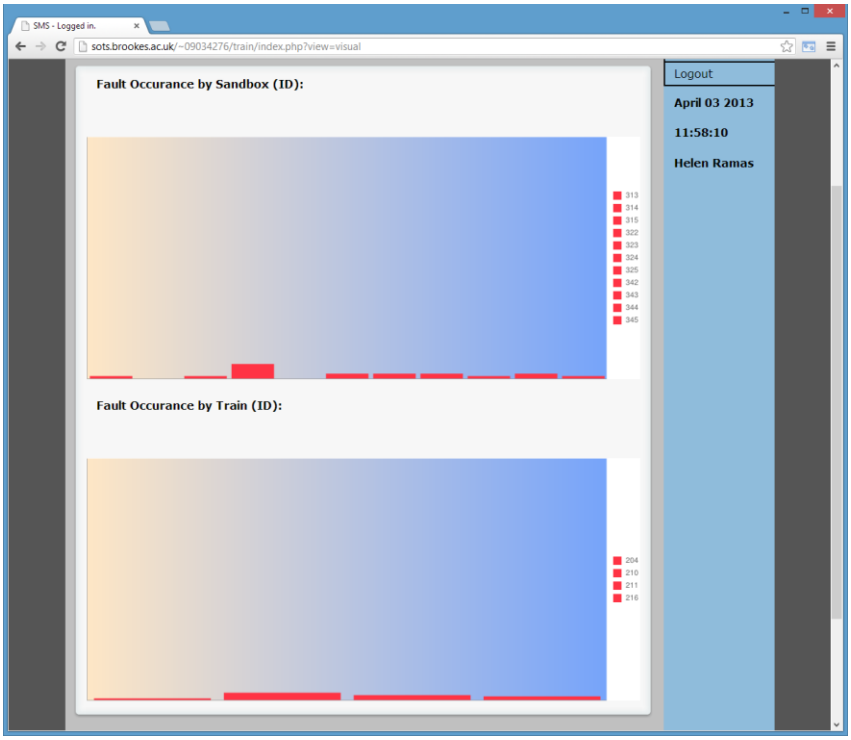

SMS - Logged in.
localhost/train/index.php?view=edit
Sandbox Management System
Train and Scheduling Setup
Add Train
Change User-Train
Setup Schedule
Edit/Delete Train
Edit/Delete Train
Train ID, Name: 221, #1
Delete or Edit Edit
Editing: ID 221.
Current data set is indicated inside the name box, or next to type and number.
Train Name: #1
Type: A
is currently: B
Number of Sandboxes: 1
is currently: 2
Confirm
Setup Trains and Scheduling
View Data or Reports
Engineer Panel
Setup Users
Settings
Logout
April 03 2013
11:56:35
Helen Ramas
Disclaimer: This site is constructed as a project at Oxford Brookes University. It is not a working website and is not connected with any site referenced. The views and opinions expressed within these pages are personal and should not be construed as reflecting the views and opnions of Oxford Brookes University.

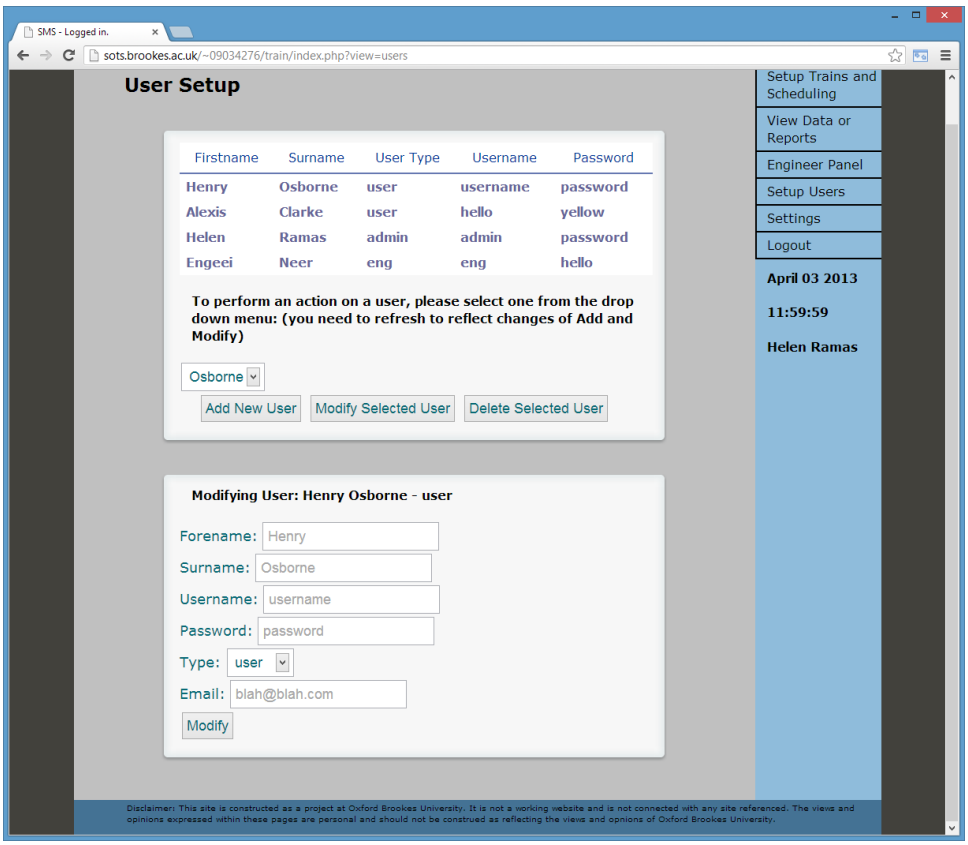

SMS - Logged in.
localhost/train/index.php?view=edit
Sandbox Management System
Train and Scheduling Setup
Add Train
Change User-Train
Setup Schedule
Edit/Delete Train
Edit/Delete Train
Train ID, Name: 221, #1
Delete or Edit Delete
The train selected has yet to complete its schedule. Cannot be deleted.
Setup Trains and Scheduling
View Data or Reports
Engineer Panel
Setup Users
Settings
Logout
April 03 2013
11:56:35
Helen Ramas
Disclaimer: This site is constructed as a project at Oxford Brookes University. It is not a working website and is not connected with any site referenced. The views and opinions expressed within these pages are personal and should not be construed as reflecting the views and opnions of Oxford Brookes University.

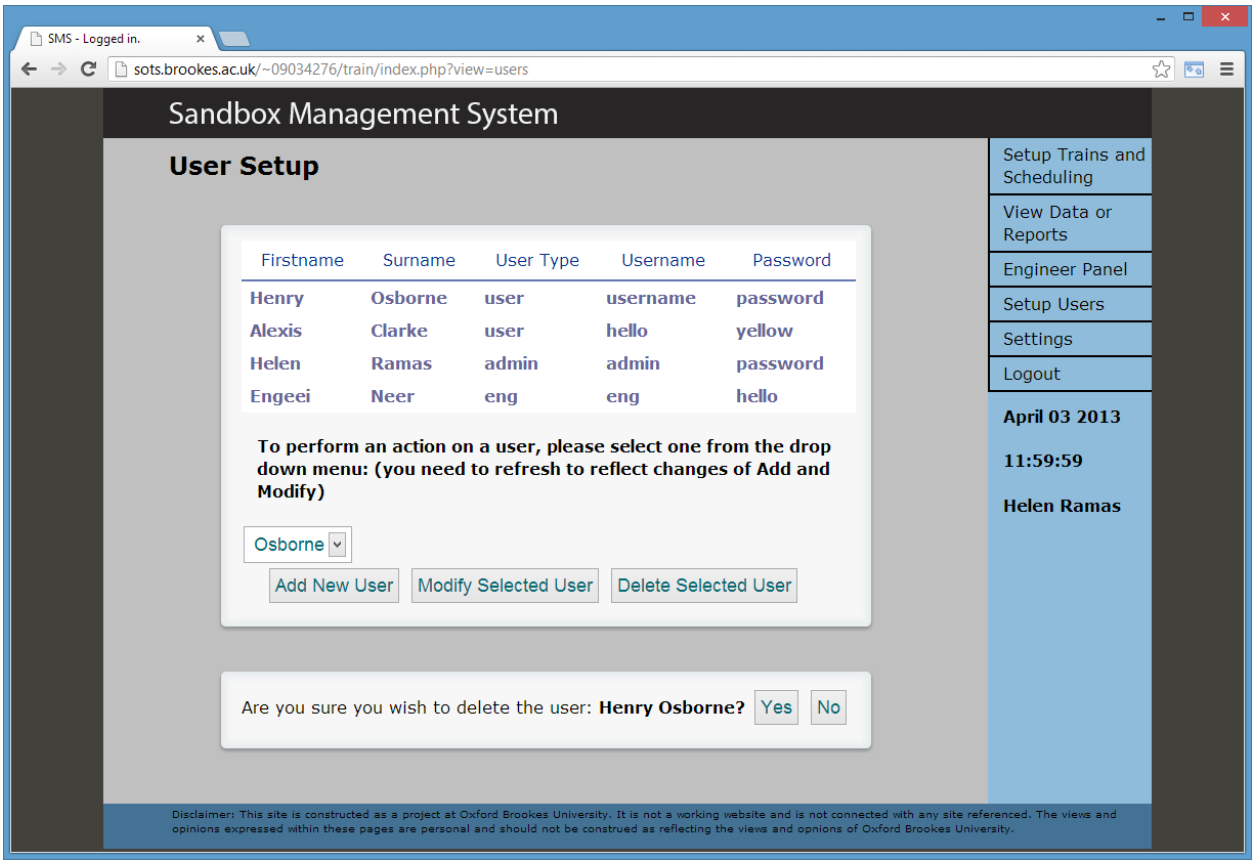

SMS - Logged in.
localhost/train/index.php?view=sbreports
Sandbox Management System
View Data in List
View Data in List Format
Visualized Reports
View in list format: Users View
Firstname Surname User Type Modify User
Henry Osborne user
Alexis Clarke user
Helen Ramas admin
Engeei Neer eng
Hello Snello eng
Setup Trains and Scheduling
View Data or Reports
Engineer Panel
Setup Users
Settings
Logout
April 03 2013
11:57:12
Helen Ramas
Disclaimer: This site is constructed as a project at Oxford Brookes University. It is not a working website and is not connected with any site referenced. The views and opinions expressed within these pages are personal and should not be construed as reflecting the views and opnions of Oxford Brookes University.

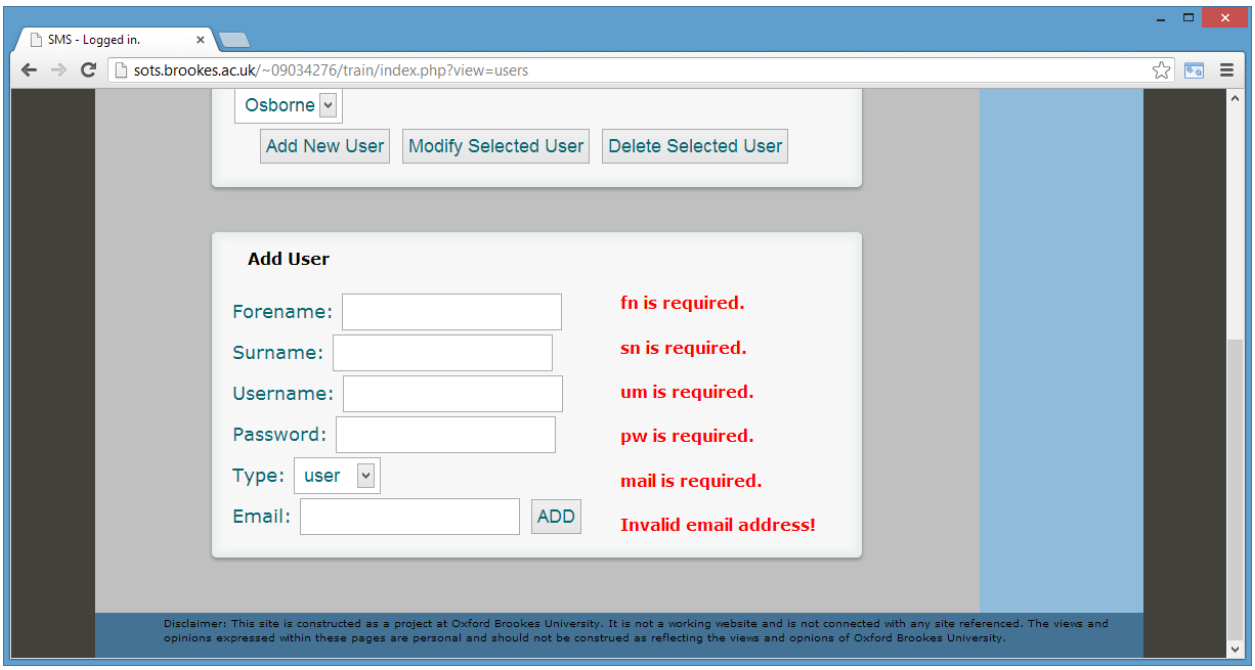

SMS - Logged in.
sots.brookes.ac.uk/~09034276/train/index.php?view=visual
Sandbox Management System
Visualized Data
View Data in List
Visualized Data
Search Data
Report type: Missed Sandboxes Get Charts
User ID: 1, Occurance of missed sandboxes after train has left: 5
User ID: 3, Occurance of missed sandboxes after train has left: 28
User ID: 22, Occurance of missed sandboxes after train has left: 29
Osborne
Ramas
Setup Trains and Scheduling
View Data or Reports
Engineer Panel
Setup Users
Settings
Logout
April 03 2013
11:58:10
Helen Ramas
Disclaimer: This site is constructed as a project at Oxford Brookes University. It is not a working website and is not connected with any site referenced. The views and opinions expressed within these pages are personal and should not be construed as reflecting the views and opnions of Oxford Brookes University.

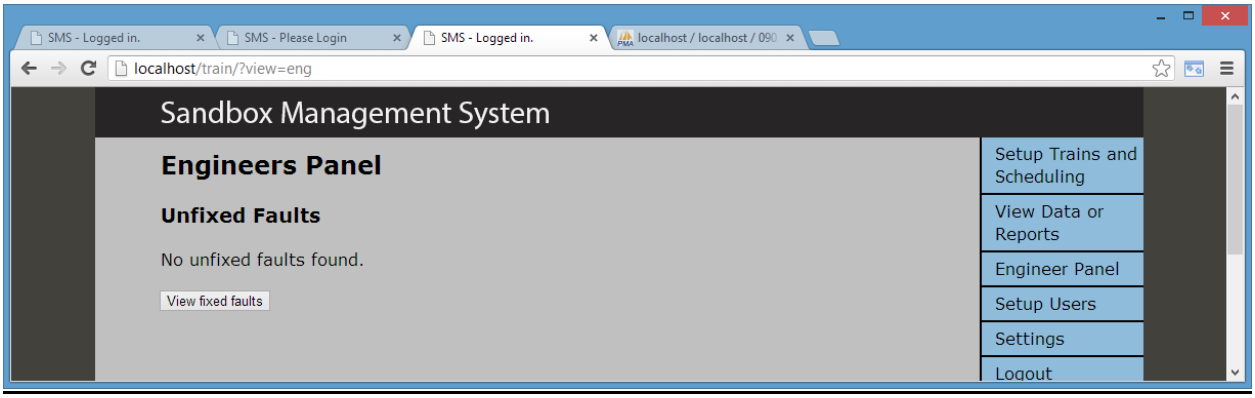

SMS - Logged in.
sots.brookes.ac.uk/~09034276/train/index.php?view=visual
Report type:
Average sand levels prior to filling
Get Charts
Setup Users
Settings
Logout
April 03 2013
11:58:10
Helen Ramas
Average sandlevels by Sandbox (ID) (scale 1-100):
All Tickets by Issue Type
313
314
315
322
323
324
325
342
343
344
345
Average sandlevels by Train (ID) (scale 1-100):
204
210
211
216

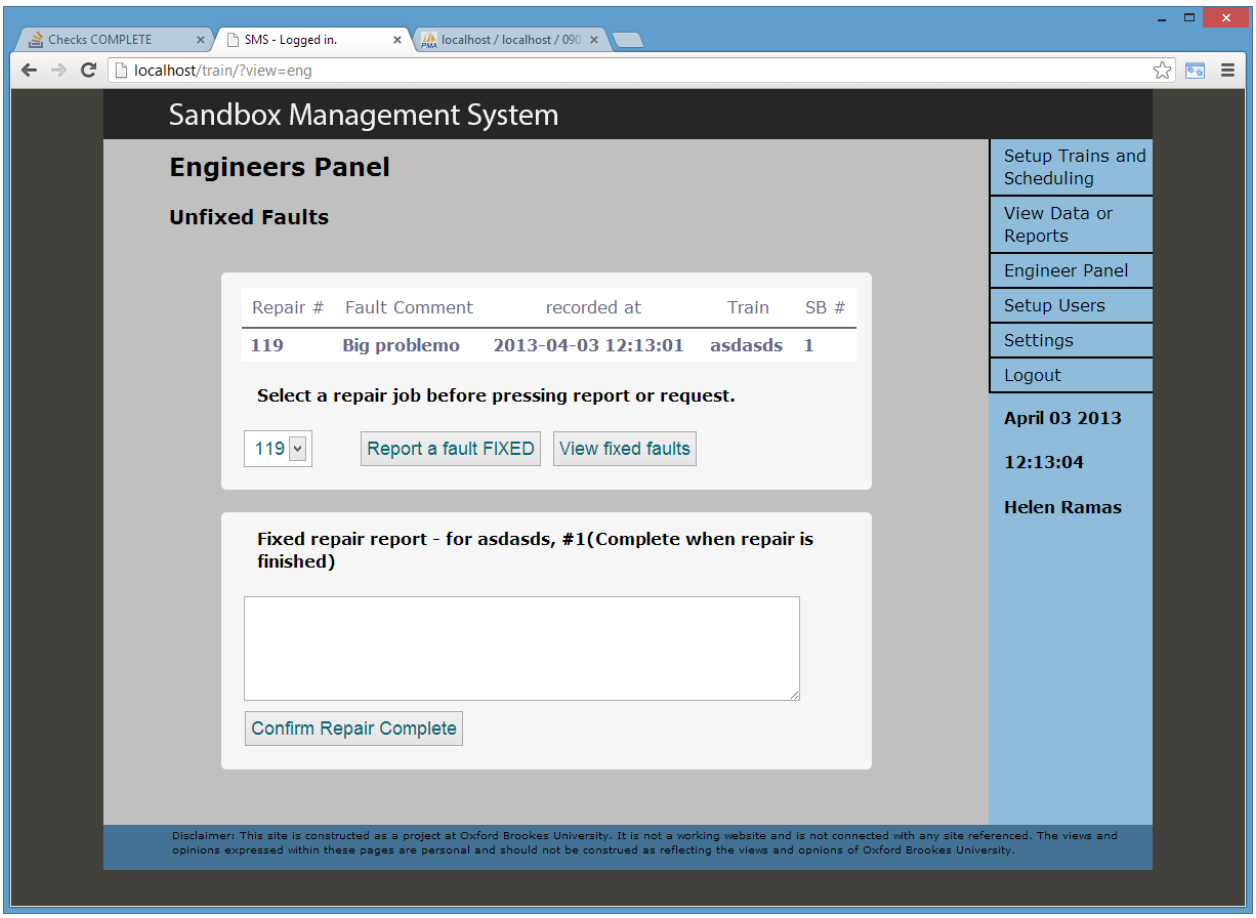

SMS - Logged in.
sots.brookes.ac.uk/~09034276/train/index.php?view=visual
Fault Occurance by Sandbox (ID):
Logout
April 03 2013
11:58:10
Helen Ramas
313
314
315
322
323
324
325
342
343
344
345
Fault Occurance by Train (ID):
204
210
211
216

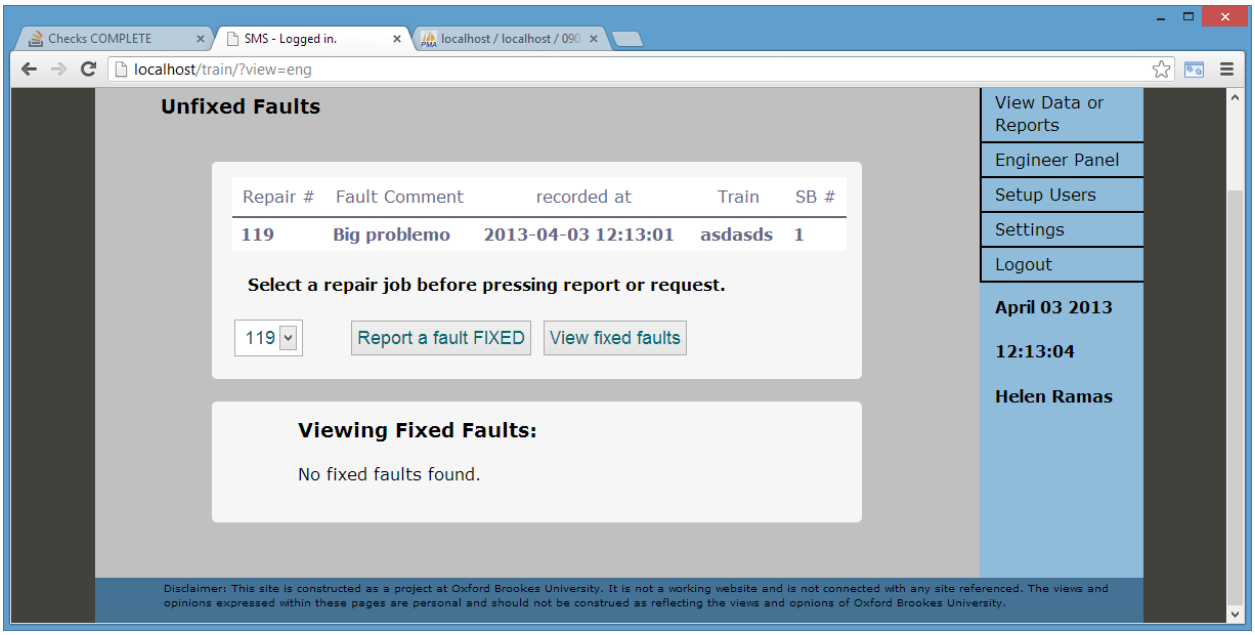

SMS - Logged in.
sots.brookes.ac.uk/~09034276/train/index.php?view=users
User Setup
Setup Trains and Scheduling
View Data or Reports
Engineer Panel
Setup Users
Settings
Logout
April 03 2013
11:59:59
Helen Ramas
Firstname Surname User Type Username Password
Henry Osborne user username password
Alexis Clarke user hello yellow
Helen Ramas admin admin password
Engeei Neer eng eng hello
To perform an action on a user, please select one from the drop down menu: (you need to refresh to reflect changes of Add and Modify)
Osborne
Add New User
Modify Selected User
Delete Selected User
Modifying User: Henry Osborne - user
Forename: Henry
Surname: Osborne
Username: username
Password: password
Type: user
Email: blah@blah.com
Modify
Disclaimer: This site is constructed as a project at Oxford Brookes University. It is not a working website and is not connected with any site referenced. The views and opinions expressed within these pages are personal and should not be construed as reflecting the views and opnions of Oxford Brookes University.

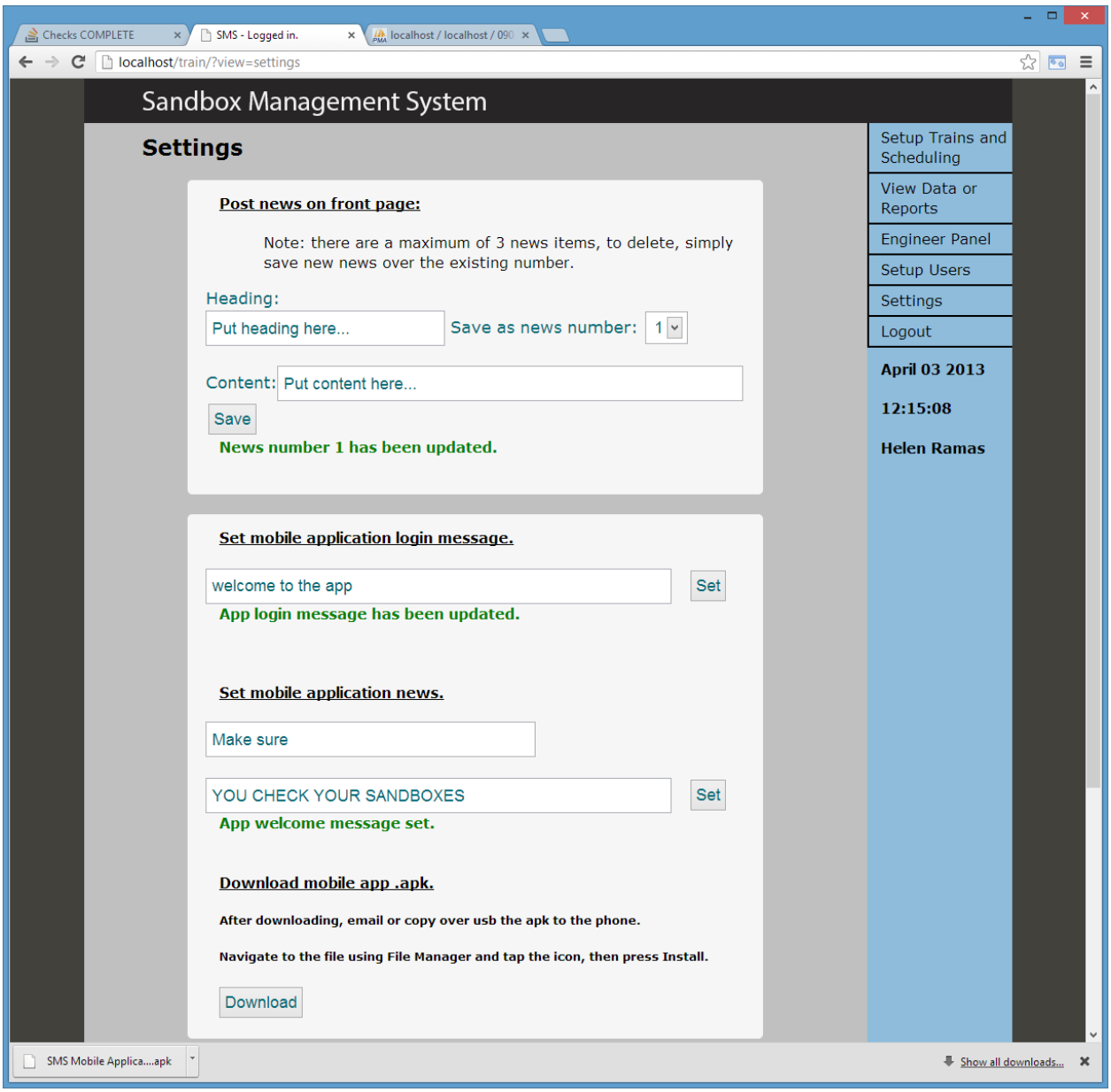

SMS - Logged in.
sots.brookes.ac.uk/~09034276/train/index.php?view=users
Sandbox Management System
User Setup
Setup Trains and Scheduling
View Data or Reports
Engineer Panel
Setup Users
Settings
Logout
April 03 2013
11:59:59
Helen Ramas
Firstname Surname User Type Username Password
Henry Osborne user username password
Alexis Clarke user hello yellow
Helen Ramas admin admin password
Engeei Neer eng eng hello
To perform an action on a user, please select one from the drop down menu: (you need to refresh to reflect changes of Add and Modify)
Osborne
Add New User
Modify Selected User
Delete Selected User
Are you sure you wish to delete the user: Henry Osborne? Yes No
Disclaimer: This site is constructed as a project at Oxford Brookes University. It is not a working website and is not connected with any site referenced. The views and opinions expressed within these pages are personal and should not be construed as reflecting the views and opnions of Oxford Brookes University.


SMS - Logged in.
sots.brookes.ac.uk/~09034276/train/index.php?view=users
Osborne
Add New User
Modify Selected User
Delete Selected User
Add User
Forename:
Surname:
Username:
Password:
Type:
user
Email:
ADD
fn is required.
sn is required.
um is required.
pw is required.
mail is required.
Invalid email address!
Disclaimer: This site is constructed as a project at Oxford Brookes University. It is not a working website and is not connected with any site referenced. The views and opinions expressed within these pages are personal and should not be construed as reflecting the views and opnions of Oxford Brookes University.


SMS - Logged in.
SMS - Please Login
SMS - Logged in.
localhost / localhost / 090
localhost/train/?view=eng
Sandbox Management System
Engineers Panel
Unfixed Faults
No unfixed faults found.
View fixed faults
Setup Trains and Scheduling
View Data or Reports
Engineer Panel
Setup Users
Settings
Logout


Checks COMPLETE
SMS - Logged in.
localhost / localhost / 090
localhost/train/?view=eng
Sandbox Management System
Engineers Panel
Unfixed Faults
Repair # Fault Comment recorded at Train SB #
119 Big problemo 2013-04-03 12:13:01 asdasds 1
Select a repair job before pressing report or request.
119
Report a fault FIXED
View fixed faults
Fixed repair report - for asdasds, #1(Complete when repair is finished)
Confirm Repair Complete
Setup Trains and Scheduling
View Data or Reports
Engineer Panel
Setup Users
Settings
Logout
April 03 2013
12:13:04
Helen Ramas
Disclaimer: This site is constructed as a project at Oxford Brookes University. It is not a working website and is not connected with any site referenced. The views and opinions expressed within these pages are personal and should not be construed as reflecting the views and opnions of Oxford Brookes University.


Checks COMPLETE
SMS - Logged in.
localhost / localhost / 090
localhost/train/?view=eng
Unfixed Faults
Repair # Fault Comment recorded at Train SB #
119 Big problemo 2013-04-03 12:13:01 asdasds 1
Select a repair job before pressing report or request.
119
Report a fault FIXED
View fixed faults
Viewing Fixed Faults:
No fixed faults found.
View Data or Reports
Engineer Panel
Setup Users
Settings
Logout
April 03 2013
12:13:04
Helen Ramas
Disclaimer: This site is constructed as a project at Oxford Brookes University. It is not a working website and is not connected with any site referenced. The views and opinions expressed within these pages are personal and should not be construed as reflecting the views and opnions of Oxford Brookes University.


Checks COMPLETE
SMS - Logged in.
localhost / localhost / 090
localhost/train/?view=settings
Sandbox Management System
Settings
Setup Trains and Scheduling
View Data or Reports
Engineer Panel
Setup Users
Settings
Logout
April 03 2013
12:15:08
Helen Ramas
Post news on front page:
Note: there are a maximum of 3 news items, to delete, simply save new news over the existing number.
Heading:
Put heading here...
Save as news number:
1
Content:
Put content here...
Save
News number 1 has been updated.
Set mobile application login message.
welcome to the app
Set
App login message has been updated.
Set mobile application news.
Make sure
YOU CHECK YOUR SANDBOXES
Set
App welcome message set.
Download mobile app .apk.
After downloading, email or copy over usb the apk to the phone.
Navigate to the file using File Manager and tap the icon, then press Install.
Download
SMS Mobile Applica....apk
Show all downloads...

Appendix D – Source Code – Also on CD (around 75% of the code is printed here), database please see CD.

Mobile Application

index.php

```
<!DOCTYPE html> <html> <head>
<title>SMS Mobile - Login</title>
<?php
        include('function.php'); // includes all functions.
        session_name('mwa'); // gives the session a name, stops conflict between mobile app and web app
        session_start();
        $do = empty($_GET['do']) ? 'index' : $_GET['do']; // sets $do.

            switch($do) {

                case "logout": // when ?do=logout is passed in the URL,
                logout(); //u ser logged out

                break; }
        RefOpeningTags(); // includes the rest of <head>, opens page + header
?>

    <h1>SMS - Please Login</h1>
    </div><!-- /header -->

    <div data-role="content" class= "center" data-theme="a" >
        <h2>Welcome to the Sandbox Monitoring System.</h2>
        <p>Your depot administrator will have provided you with a username and password.</p>
        <p>Please enter these details in the boxes below, then press "Login".</p>
        <br>
        <form action="login.php" method="post" data-ajax="false" class="center">
                <fieldset>
                    <div data-role="fieldcontain">
                        <label for="name">Username: </label>
                        <input type="text" name="username" id="" value="" />
                    </div>


                <div data-role="fieldcontain">
                        <label for="password">Password: </label>
                        <input type="password" name="password" id="" value="" />
                </div>
                <button type="submit" data-theme="b"  data-inline='true'
                name="login" id="login" value="Ok">Submit</button>
            </fieldset>
        </form>
        <br>

    </div><!-- /content -->

    <div data-role="footer" data-position="fixed" data-theme="a">
        <h6>This site is constructed as coursework for Oxford Brookes University</h6>
    </div><!-- /footer -->
</div><!-- /page one -->
</body>
</html>
```

Function.php

```php
<?php

function db_connect()
//connects to sots and selects database
{
    $connection = mysql_pconnect('localhost', '09034276', 'Ticket1');
    if(!$connection) {
        return false;
    }
    if(!mysql_select_db('09034276')) {
          return false;
    }
    return $connection;
}

function update_schedule()
{

    // If a train has a schedule #2 and now (+ 2hours) > schedule #1 then #2 is made #1
    $userID = $_SESSION['userid'];

    $s_check = "SELECT trainID, entryt, vacatet, entryt2, vacatet2 FROM train WHERE userID = $userID";
    $s_check_res=mysql_query($s_check);

        while($s_check_arr = mysql_fetch_assoc($s_check_res)) {

            // set a var to 2 hours ahead of now
            $now = new DateTime();
            $schedulechange = $now->modify('+2 hours');
            $vacate = new DateTime($s_check_arr['vacatet']);

                if (($schedulechange > $vacate) && ($s_check_arr['vacatet2'] > '0000-00-00 00:00:00')) {

                    $entryt = $s_check_arr['entryt2'];
                    $vacatet = $s_check_arr['vacatet2'];
                    $trainID = $s_check_arr['trainID'];

                    echo $trainID;
                    echo "has has its schedule 2 set.";

                    $update = "UPDATE train SET `entryt`= '$entryt',
                    `vacatet`='$vacatet' WHERE trainID = '$trainID' ";

                        mysql_query($update);

                            $remove = "UPDATE train SET `entryt2`= '0000-00-00 00:00:00',
                            `vacatet2`='0000-00-00 00:00:00' WHERE trainID = '$trainID' ";

                            mysql_query($remove);

                            // pass id to function that resets the sandbox and checkedat fields.

                            reset_sandbox_data($trainID);
            }
        }
}

function reset_sandbox_data($trainID)
{
    $insert=sprintf("UPDATE sandbox SET checked='%s', sbLevel='%s',
        sbDefect='%s', sbDefectComment='%s' WHERE trainID = '$trainID' ",
    mysql_real_escape_string($checked),
    mysql_real_escape_string($sbLevel),
    mysql_real_escape_string($sbDefect),
    mysql_real_escape_string($sbDefectComment),
    mysql_real_escape_string($sbID));

    $checkedat = '0000-00-00 00:00:00';

        mysql_query($insert) or die('There has been a problem connecting to the DB. Your changes are not saved.');

            $insert=sprintf("UPDATE train SET checkedat='%s' WHERE trainID = '$trainID'",
                        mysql_real_escape_string($checkedat));

                mysql_query($insert) or die('There has been a problem connecting to the DB. Your changes are not saved.');

    |
}

```

Login.php

```
<!DOCTYPE html> <html> <head> <title>SMS Mobile</title>
        <?php
            session_name('mwa');
            session_start(); //must call session_start before using any $_SESSION variables
            error_reporting(E_ALL ^ E_NOTICE); // ignores error reports.
            include('function.php'); // includes all functions.
            RefOpeningTags(); // includes the rest of <head>, opens page + header
        ?>

    <h1>SMS</h1>
    <a href="index.php?do=logout" data-role="button" data-icon="delete"
    data-iconpos="left" data-mini="true" data-inline="true">Logout</a>

    </div><!-- /header -->

    <div data-role="content" style="margin:0 auto; margin-left:auto;
    margin-right:auto; align:center; text-align:center;" data-theme="a" >

<?php
    // CONVERT FORM VALUES TO PHP VARIABLE
if(isset($_POST['username'])){ $user = $_POST['username']; }
if(isset($_POST['password'])){ $pw = $_POST['password']; }

    db_connect();
    $qry = "SELECT userid FROM user WHERE username='$user' AND password='$pw'";
    $result = mysql_query($qry);
    $row = mysql_fetch_array($result);
    $_SESSION['userid'] = $row['userid'];

        if(mysql_num_rows($result) < 1) //no such user exists
            {
                unset($_SESSION['userid']);
                // LOGIN FAIL
                echo "<h1>Login Failed</h1>";
                echo"<br>";
                echo "<h2>Incorrect login details.</h2>";
                echo"<a href='index.php' data-role='button' data-theme='b' data-inline='true'>Go back to login</a>";
        echo"</div><!-- /content -->";
        echo"<div data-role='footer' data-position='fixed' data-theme='a'>";
        echo"<h6>This site is constructed as coursework for Oxford Brookes University</h6>";
echo"</div></div></body></html>";

            }
        else
                // LOGIN SUCCESS
                // the main menu is loaded via ajax - as a result it must all be echod for security reasons
                // without the echos, if the user failed to log in
            //they would still be able to view the code for the main menu and gain unauthorized entry
            {

                $qry2= "SELECT * FROM user WHERE username='$user' AND password='$pw'";
                $result2=mysql_query($qry2);
                $row = mysql_fetch_array($result2);
                    //sets the session data for this user

                    session_regenerate_id ();

                       $_SESSION['valid'] = 1;
                       $_SESSION['type'] = $row['type'];

                           $qry= "SELECT login FROM news WHERE newsid = '1'";
                           $result=mysql_query($qry);
                           $row = mysql_fetch_array($result);

                           // login message
                           echo "<h2>".$row['login']."</h2>";

                           echo"<h2>Your ID is ".$_SESSION['userid'].".</h2>";
                           echo"<h2>Your account level is ".$_SESSION['type'].".</h2>";
                           echo"<br>";
                           echo"<p><a href='train.php'= data-role='button'
                           data-inline='true' data-theme-'b'>Press to continue!</a></p>";
        echo"</div><!-- /content -->";

echo"<div data-role='footer' data-position='fixed' data-theme='a'>";

echo"<h6>This site is constructed as coursework for Oxford Brookes University</h6>";

echo"</div><!-- /footer -->"; // end of footer

echo"</div><!-- /page one --></body></html>";

        }
?>
```

Sbcheckform.php

```
<!DOCTYPE html>
<html><head>
<title>Sandbox Check Form</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet"  href="../../css/themes/default/jquery.mobile-1.2.0.css" />
    <link rel="stylesheet" href="../_assets/css/jqm-docs.css"/>
    <script src="//code.jquery.com/jquery-1.7.1.min.js"></script>
    <script src="../../docs/_assets/js/jqm-docs.js"></script>
    <script src="../../js/jquery.mobile-1.2.0.js"></script>
</head>
<body>

<div data-role="dialog">

    <script>
    function val(url, action) {
    switch(action)
    {
        case 'val':

        var sb= document.getElementById('sbLevel').value
            var defect= document.getElementById('defect').value
            var comment= document.getElementById('comment').value


            string= 'level=' + escape(sb)
            + '&defect=' + escape(defect) + '&comment=' + escape(comment);
        break;
    }
    xmlHttp = new XMLHttpRequest();
    xmlHttp.open('POST', url, true);
    xmlHttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

    xmlHttp.onreadystatechange = function() {

    if (xmlHttp.readyState == 4) {

        document.getElementById("response").innerHTML=xmlHttp.responseText
        // if the validation passes, a redirect javascript is echo'd back.
        eval(xmlHttp.responseText)
        // it needs to be eval'd to to be exectuted.
        // else the validation messages are displayed.
        }

    }
    xmlHttp.send(string);
    }

    </script>

    <div data-role="header" data-theme="d">

        <?php
        include('function.php'); // includes all functions.
        session_name('mwa');
        session_start();

        $sb = empty($_GET['sb']) ? 'nosbselected' : $_GET['sb'];
        // gets the sandbox id that is going to be checked.

        $key=array_search($sb, $_SESSION['sbArray']);
        // search the array of all of the
        //possible sandboxes for that user with the sandbox selected

        if($key !== false) {  // if the result isn't false then find out the train id, to help the user.

        db_connect();

            // formulate query
            $qry=sprintf("SELECT * FROM sandbox WHERE sbID='%s'",
            mysql_real_escape_string($_SESSION['sbArray'][$key]));

            // execute query
            $result = mysql_query($qry);

            while($row = mysql_fetch_assoc($result))
            {

                echo"<h1>Sandbox #".$_SESSION['sbArray'][$key];echo ", Train #".$row['trainID']; echo"</h1></div>";

                    $checked= $row['checked'];
                    $timechecked= $row['timechecked'];
                    $level= $row['sbLevel'];
                    $defect= $row['sbDefect'];
                    $comment= $row['sbDefectComment'];
                    $_SESSION['sbChecked']=$_SESSION['sbArray'][$key];

                echo"<div data-role='content' data-theme='c'>
                <p id='response'> </p>";
                echo"<form id = 'sbform' action='train.php' method='post'>";
```

```
                //echo"<form id = 'sbform'>";
                echo"<fieldset>

                    <ul data-role='listview' data-inset='true'>
                        <li data-role='fieldcontain'>
                            <label for='slider'>Sand Level (prior to filling):</label>";
                            //echo"<input type='number' name='sbLevel' id='sbLevel'";

                            echo"<input type='range' name='sbLevel' id='sbLevel' min='0' max='100'";

                            if ($checked != 'Not checked') {
                                // if already checked, set the level that was input.
                                echo " value='".$level."'/>";
                            } else {
                                // else have a default value of 0
                                echo" value='0' />";
                            }

                            echo "</li>
                            <li data-role='fieldcontain'>
                            <label for='slider2'>Equipment Fault?:</label><br>
                                <select name='defect' id='defect' data-role='slider'>";

                                    if ($checked != 'Not checked') {

                                        if ($defect == 1) {

                                            echo"<option value='0'>No</option>
                                        <option selected='selected' value='1'>Yes</option>";

                                        } else {

                                            echo"<option selected='selected' value='0'>No</option>
                                        <option value='1'>Yes</option>";
                                        }

                                    } else {

                                        echo"<option value='0'>No</option>
                                        <option value='1'>Yes</option>";
                                        }

                                    echo"</select>
                            </li>
                            <li data-role='fieldcontain'>
                            <label for='textarea'>Fault Comment:</label>
                                <textarea cols='40' rows='8' name='defectText' id='comment'>";

                                if ($checked != 'Not checked') {  echo $comment;
                                } else {
                                // dont put anything
                                }
                                echo"</textarea>
                                    </li>";
                                if ($checked != 'Not checked') {
                                    echo "<h3>Form filled at ".$timechecked."</h3>";
                                } ?>

                                <input value="Submit" data-theme='b' type="button" onclick='val("validate.php","val")'>

                            <?php  // above calls the ajax function, also setting the destination script.
                            echo "<a href='train.php' data-role='button' data-rel='back' data-theme='c'>Go Back</a>

                </fieldset>";
                echo"</form>
                </div>
                </div>";

                }
            } ?>
</body>
</html>
```

Schedule.php

```
<!DOCTYPE html> <html> <head>

<title>Schedule</title>

<?php
        include('function.php'); // includes all functions.
        session_name('mwa');
        session_start(); // resume session
        error_reporting(E_ALL ^ E_NOTICE);
        RefOpeningTags(); // includes the rest of <head>, opens page + header
        isLoggedIn(); // check if user is logged in
        db_connect(); //connect to db
        $userID = $_SESSION['userid'];
?>

        <h1>Schedule</h1>

     <a href="index.php?do=logout" data-icon="home" data-iconpos="notext" >Main Menu</a>

     <a href="" data-icon="search" data-iconpos="notext" data-rel="dialog" data-transition="fade">Search</a>
   </div><!-- /header -->

<div data-role="content" data-theme="b">

<h2> Displaying all Trains registered to you </h2>

<table data-role="table"  class="ui-body-d ui-shadow table-stripe ui-responsive" data-mode="reflow"  >
          <thead>
            <tr class="ui-bar-d">
              <th data-priority="2">Train Name</th>
              <th data-priority="6"># of SB</th>
              <th data-priority="5">Train Type</th>
              <th data-priority="1">Entry</th>
              <th data-priority="3">Vacate</th>
            </tr>
          </thead>
          <tbody>

<?php

    // generates the schedule table.

   $qry = "SELECT * FROM train WHERE userID = $userID ORDER BY vacatet";
   $result = mysql_query($qry);
   while($row = mysql_fetch_assoc($result)) {

        echo"<tr>
             <th>".$row['trainName']."</th>
             <td>".$row['numberSb']."</td>
             <td>".$row['trainType'] ." 
             <img src='images/".$row['trainType'].".jpg' width='75' height ='50' /> </td>
             <td><b>".$row['entryt']."</b></td>
             <td><b>".$row['vacatet']."</b></td>
             </tr>";  }

    echo"</tbody>
        </table> ";

     footer();

     ?>
```

Train.php

```php
<?php
    echo "<!DOCTYPE html> <html> <head> <title>Check Sandboxes</title> ";
    include('function.php'); // includes all functions.
    session_name('mwa');
    session_start(); // resume session
    error_reporting(E_ALL ^ E_NOTICE);
    RefOpeningTags(); // includes the rest of <head>, opens page + header
    db_connect(); //connect to db
    isLoggedIn(); // check if user is logged in
    $userID = $_SESSION['userid']; // set user id, this is buggy and value is sometimes lost.

// updates the scheduling for the trains.
// If a train has a schedule #2 and now (+ 2hours) > schedule #1 then #2 is made #1
update_schedule();

//Sets sbArray and trainArray.
//it is calculated later on in the script, but values are lost upon login.

$qry4=sprintf("SELECT trainID FROM train WHERE userID='%s'",
                    mysql_real_escape_string($userID));
$result4= mysql_query($qry4);
if(mysql_num_rows($result4)== 0){ noTrain();
} else { // if no trains error message is displayed

getSettings(); // get user preferences - NOT really used anymore.

$qry4="SELECT * FROM train
            WHERE DATE(vacatet)=  CURDATE()
                AND userID= $userID ORDER BY vacatet ASC";
$result4= mysql_query($qry4);

if(mysql_num_rows($result4)== 0){ noTrainToday(); } else {
// no trains on this day error displayed. trains still exist for that user however

    while($row4 = mysql_fetch_assoc($result4)) {
        $sbchecktrainID[] =$row4['trainID'];

        // place vacate datetime into array
        $vacate_DT[]=$row4['vacatet'];
        $_SESSION['vacateDT'] = $vacate_DT;

        // create array of train IDs
        $_SESSION['trainIDArray'] = $sbchecktrainID;
        $hooha = implode(',', $_SESSION['trainIDArray']);

$qry5=("SELECT sbID FROM sandbox WHERE trainID IN ({$hooha})");
$result5=mysql_query($qry5);
    while($row5 = mysql_fetch_assoc($result5)) {
        $blahsbarray[]=$row5['sbID'];
        $_SESSION['sbArray'] =$blahsbarray;
    }
}

// NUMBER OF SBS LEFT CALCULATIONS
if(isset($_SESSION['sbArray']) && $_SESSION['sbArray'])  {

//implode array to comma sepearted string for query
            $sblistquery = implode(',', $_SESSION['sbArray']);

            $qrySB=("SELECT count(sbID) FROM sandbox WHERE sbId
                IN ({$sblistquery}) AND checked = 'Not checked'");

            $sb_checked_qry = mysql_query($qrySB);

                while($sbrow = mysql_fetch_assoc($sb_checked_qry))
                {
                    $notcheckedassoc = $sbrow;
                }

} // end ifsset Sbarray

if (isset($notcheckedassoc)) {

foreach ($notcheckedassoc as $key=>$val)
$notchecked=$val;
}

if ($notchecked == 0) {
        echo "<h1 class=green>Checks COMPLETE</h1>";
        }

    else {
        echo"<h1 class=red>Checks INCOMPLETE</h1>";

        }
```

```
    echo"<a href='main.php' data-icon='home' data-iconpos='notext' >Main Menu</a>

        <a href='index.php?do=logout' data-icon='search' >Logout</a>
    </div><!-- /HEADER -->

    <div data-role='content' data-theme='b'>";

    if ($notchecked>0) {
    echo"<h4 class=red>Sandboxes UNCHECKED: ".$notchecked;
    }

echo"</h4><h4>";

    // TIME DISPLAY
    $timedisplay = new DateTime();
    echo date_format($timedisplay, 'H:i:s\, l jS F Y');
    echo "</h4><h4>Displaying trains that are due to vacate depot today.</h4>
    <h4>Trains due to leave first are at the top. </h4>";

// ALL BELOW THIS IS FOR LISTING THE TRAINS AND SANDBOXES
//define search variables
$userID=$_SESSION['userid'];
// EXECTUE MAIN TRAIN DISPLAY QUERY
$result = mysql_query($qry4);
//define arrays
$trainarray = array();
$sbarray = array();

while($row = mysql_fetch_assoc($result)) // GET TRAIN ID'S FOR USER
// Begin TRAIN PRINTING OUT WHILE (FIRST)
{
    // train type
    $trainType= $row['trainType'];
    // total number of sandboxes for each train
    $numberSb = $row['numberSb'];
    //set checked
    $checked = "Checked";
    $checkedat =  $row['checkedat'];
    // THIS BELOW WAS THE FIX TO MY BUG - IT TURNS OUT I HADNT SET $trainID until a later loop
    $trainID  = $row['trainID'];
    //gets count of number of checked sandboxes and calls it sbComplete
    $qry3=sprintf("SELECT COUNT(sbID) AS sbComplete FROM sandbox WHERE trainID='%s' AND checked='%s'",
                    mysql_real_escape_string($trainID),
                    mysql_real_escape_string($checked));

        $result3 = mysql_query($qry3);

            while($row3 = mysql_fetch_assoc($result3)) {

            $sbComplete = $row3['sbComplete'];

            } // end while row 3

            // get datetimes
            $entryRaw=$row['entryt'];
            $vacateRaw=$row['vacatet'];

            // format them
            $entry = new DateTime($entryRaw);
            $vacate = new DateTime($vacateRaw);

            // splits datetime into seperate date and time
            $ed = $entry->format('d-m-Y');
            $et = $entry->format('H:i');
            $vd = $vacate->format('d-m-Y');
            $vt = $vacate->format('H:i');

            // get difference between vacate time and current time
            $now = new DateTime();
            $timeleft = $vacate->diff($now);

            $dont_display = $now->modify('-30 minutes');
            $checked_at = new DateTime($row['checkedat']);

            // IF HAS BEEN CHECKED FOR MORE THAN 30 mins REMOVE.
            if ((($row['checkedat']) > '0000-00-00 00:00:00') && ($dont_display > $checked_at))

            { } // NOT DISPLAYED

            else { // carry on with rest of the script

            // if total number of sandboxes is = to number of sandboxes checked
            if ($numberSb==$sbComplete) {

                // Set the TRAIN selector box to be white instead of blue, indicating all checks complete
                echo"<div data-role='collapsible' data-theme='c'  data-icon='check'>";

            } else{
                // Still blue - checks remain.
                echo"<div data-role='collapsible'  data-icon='check'>";
            }
```

```
            } else{
                // Still blue - checks remain.
                echo"<div data-role='collapsible'  data-icon='check'>";
            }

            if ($checkedat > 1) {
                echo "<h2>".$row['trainName']." check done at ".$checkedat.".</h2>";

                if ($checkedat > $vacate)
                {
                    echo "<h2><font color='red'>Checks were made late.</font></h2>"; // DOESNT WORK

                }


            } else {

                // if current time is greater than train vacate time, it is late
            if ($now > $vacate)
                {
                    echo"<h2><font color='red'>".$row['trainName']." - ";
                    echo $timeleft->format("Checks late by: %h hours, %i minutes").".</font></h2>";
                } else {

        echo"<h2>".$row['trainName']." - "; echo $timeleft->format("%h hours, %i minutes");echo" left.</h2>";

            }// end else for if train delayed

            }

            echo"<h5>Arrived/arriving at: ".$et." on ".$ed; // entry time/date
            echo" and should leave/left at: ".$vt." on ".$vd; // vacate time/date

        echo".</h5><ul data-role='listview' data-inset='true' data-split-icon='gear'
        data-divider-theme='a' data-theme='a' data-split-theme='a'>";

            unset($row2['checked']);

            $trainID  = $row['trainID']; // TRAIN ID'S

            // SANDBOX QUERY

            $qry2=sprintf("SELECT DISTINCT sbID, checked, timechecked, sbName FROM sandbox WHERE trainID='%s'",
                    mysql_real_escape_string($trainID));

            $result2=mysql_query($qry2);

            while($row2 = mysql_fetch_assoc($result2))
            {
                $sbarray[]=$row2['sbID'];

                // array for the check form
                $_SESSION['sbArray']=$sbarray;

                if ($row2['checked'] != "Checked") {
                    // not checked - Black Bar
                    echo "<li>";
                } else{

                    // White bar
                    echo"<li data-theme='c'>";
                }

            echo"<a href='sbcheckform.php?sb=".$row2['sbID']."' data-rel='dialog'>";

            if($row2['checked'] != "Checked") {

                // if sb not checked
                //echo"<img src='images/".$trainType.".jpg' />"; OLD IMAGES
                echo"<img src='images/".$row2['sbName'].".jpg' />";
                echo"<h3>Sandbox #".$row2['sbID']." </h3>
                <p><b>".$row2['checked'].".</b></p>";
            }
            else{
                // checked - place tick for image, and time checked.
                echo"<img src='images/tick.png' />
                <h3>Sandbox #".$row2['sbID']." </h3>
                <p><b>".$row2['checked']." at: ".$row2['timechecked'].".</b></p>";
            }

            echo"</a><a href='sbcheckform.php?sb=".$row2['sbID']." ' data-rel='dialog'>Open Check Form</a>
            </a></li>";

             }// END SECOND while ?>
        </ul>
</div>

<?php

}// END  FIRST WHILE
```

validate.php

```php
<?php
include('function.php'); // includes all functions.
session_name('mwa');
session_start();
db_connect();

$sbLevel = ($_POST['level']);
$sbDefect = ($_POST['defect']);
$sbDefectComment = ($_POST['comment']);
$sbID = $_SESSION['sbChecked'];
$userID = $_SESSION['userid'];

// validate form data, echo error messages if there are any.

if (!isset($sbLevel)) {
    echo "<p><font color='red'>Please enter the sandbox level.</font></p>";
}
if ((isset($sbLevel)) && ($sbLevel == 0)) {
    echo "<p><font color='red'>Are you sure the sandbox is/was empty?</font></p>";
}
if (($sbDefect == 1) && (empty($sbDefectComment))) {
    echo "<p><font color='red'>Please leave a description of the fault.</font></p>";
}
if ((strlen($sbDefectComment) > 0) && ($sbDefect == 0)) {
    echo "<p><font color='red'>You cannot leave a comment if there is no fault.</font></p>";
}
if ((!isset($sbLevel)) || (($sbDefect == 1) && (empty($sbDefectComment)))

    || ((isset($sbLevel)) && ($sbLevel == 0)) || ((strlen($sbDefectComment) > 0) && ($sbDefect == 0)))  { }
else {
// VALIDATION PASSED, NOW PERFORM ALL ACTIONS TO ADD SANDBOX DATA
$checked = "Checked";
// add sandbox level to average, and the number of averages taken.
$occ = "SELECT * FROM fault_occ WHERE sbID = $sbID";
        $occres=mysql_query($occ);
            if(mysql_num_rows($occres) == 0) {
                $level_taken = 1;
                $insert_occ="INSERT INTO `09034276`.`fault_occ`
                    (`sbID`, `avg_level`, `level_taken`)
                    VALUES ('$sbID', '$sbLevel', '$level_taken');";

                        mysql_query($insert_occ) or die ('avg1 insert failed');

              } else {

                while ($occ_row = mysql_fetch_assoc($occres)) {
                    $cur_avg= $occ_row['avg_level'];
                    $new_avg = ($cur_avg + $sbLevel)/2;
                    $level_taken= $occ_row['level_taken'] + 1;
                }

                    $insert_occ = "UPDATE fault_occ SET `avg_level`='$new_avg',
                    `level_taken` ='$level_taken' WHERE sbID = '$sbID' ";

                        mysql_query($insert_occ) or die ('avg2 insert failed');

            }

// NOW CHECK if fault is reported - and generate a fault record in repair table.
if ($sbDefect == '1') {

    $insert = "INSERT INTO `09034276`.`repair`
    (`repairID`, `sbID`, `faultc`, `repairc`, `repairedby`, `repairedat`)
    VALUES ('', '$sbID', '$sbDefectComment', '', '', '');";

    mysql_query($insert) or die ('repair insert failed');

    // take occurance of faults  statistical purposes

    $occ = "SELECT * FROM fault_occ WHERE sbID = $sbID";

        $occres=mysql_query($occ);

            if(mysql_num_rows($occres) == 0) {

                $fault_occ = 1;

                $insert_occ="INSERT INTO `09034276`.`fault_occ`
                    (`sbID`, `fault_occ`)
                    VALUES ('$sbID', '$fault_occ');";

                        mysql_query($insert_occ) or die ('fault1 insert failed');

              } else {

                while ($occ_row = mysql_fetch_assoc($occres)) {
                    $fault_occ= $occ_row['fault_occ'] + 1;
                }
```

```
                }

                    $insert_occ = "UPDATE fault_occ SET `fault_occ`= '$fault_occ'
                            WHERE sbID = '$sbID' ";

                        mysql_query($insert_occ) or die ('fault2 insert failed');
            }

    // now send mail
    $get="SELECT trainID FROM sandbox WHERE sbID= $sbID";
    $result=mysql_query($get);
    while ($row = mysql_fetch_assoc($result)) {
        $mailTid= $row['trainID'];
    }

    $get="SELECT email FROM user WHERE type= 'eng'";
    $result=mysql_query($get);
    while ($row = mysql_fetch_assoc($result)) {
        $to[]= $row['email'];
    }

    $subject = 'Repair attention needed at Train ID: '.$mailTid.', Sandbox ID: '.$sbID.'';
    $body = $sbDefectComment;

    // call send email function, pass array of emails, subject with relevant ID's
    // and lastly the defect comment.

    send_warning_email($to, $subject, $body);

} // end IF DEFECT IS TRUE

$insert=sprintf("UPDATE sandbox SET checked='%s', sbLevel='%s',
                sbDefect='%s', sbDefectComment='%s', timechecked=NOW()
                WHERE sbID ='%s'",
    mysql_real_escape_string($checked),
    mysql_real_escape_string($sbLevel),
    mysql_real_escape_string($sbDefect),
    mysql_real_escape_string($sbDefectComment),
    mysql_real_escape_string($sbID));

        mysql_query($insert) or die('There has been a problem connecting to the DB. Your changes are not saved.');

// UNSET SBCHECKED TO PREVENT BUG - NUMBER 3 BUG.
if(isset($_SESSION['sbChecked']))  unset($_SESSION['sbChecked']);

// checks if all sandbox are checked and then sets the last checktime as the trains finished check time
$qry="SELECT trainID FROM sandbox WHERE sbID= $sbID";
$result=mysql_query($qry);
    while ($row = mysql_fetch_assoc($result)) {
        $getTID= $row['trainID'];
    }
$qry="SELECT numberSb FROM train WHERE trainID = $getTID";
$result=mysql_query($qry);
    while ($row = mysql_fetch_assoc($result)) {
        $getnumberSb= $row['numberSb'];
    }
for ($i=1; $i<=$getnumberSb; $i++)
 {
    $qry="SELECT timechecked, checked FROM sandbox WHERE trainID = $getTID AND sbName = $i";
    $result=mysql_query($qry);
        while ($row = mysql_fetch_assoc($result)) {
            $sbchecktimes[]=$row['timechecked'];
            $checkvalue[]=$row['checked'];
        }
}
$lastsbcheck=max($sbchecktimes);

// if all have been checked - set the last check time as train check time
// checks if all values in array are the same (checked, != not checked)
if (count(array_unique($checkvalue)) == 1) {

    $insert="UPDATE train SET checkedat= '$lastsbcheck' WHERE trainID = $getTID";
    mysql_query($insert);
}

// success
// send js script back which is then eval'd to redirect back to train list.
echo "window.location.replace('train.php');";

}

?>
```

Style.css

```
.red{
color:red;
}
.green {
color:green;
}
.center {
margin:0 auto;
margin-left:auto;
margin-right:auto;
align:center;
text-align:center;
}
body{
  background-color: black !important
}
```

Android app code

manifest.xml

MainActivity.java | activity_main.xml | *SMS Mobile Application Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.sandboxdatalogger"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="17" />
        <uses-permission android:name="android.permission.INTERNET">
        </uses-permission>
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@android:style/Theme.NoTitleBar"
        android:hardwareAccelerated="true">
        <activity
            android:name="com.example.sandboxdatalogger.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
               <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Activity

MainActivity.java | activity_main.xml | SMS Mobile Application Manifest

```
package com.example.sandboxdatalogger;

import android.app.Activity;

public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // load web view layout.
        WebView mainWebView = (WebView) findViewById(R.id.mainWebView);
        // enable Javascript and and apply it with websettings.
        WebSettings webSettings = mainWebView.getSettings();
        webSettings.setJavaScriptEnabled(true);
        // load class that prevents new URL being loaded into web view
        mainWebView.setWebViewClient(new PreventLinks());
        // set the URL to load as SOTS
        mainWebView.loadUrl("http://sots.brookes.ac.uk/~09034276/mobile/");
    }

    private class PreventLinks extends WebViewClient {
        @Override
        public boolean shouldOverrideUrlLoading(WebView view, String url) {
            view.loadUrl(url);
            return true;
        }
    }
}
```

MainActivity.java | activity_main.xml | SMS Mobile Application Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:scrollbars="none"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <WebView android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:id="@+id/mainWebView">
     </WebView>
</LinearLayout>
```

Web Application Code

Index.php

index.php

```php
<?php
// includes all functions.
include('functions/function.php');
//creates/resumes the session.
session_start();
// defaults to index view unless a different view is requested.
$view = empty($_GET['view']) ? 'home' : $_GET['view'];
// set useractivity in user table for login autenthentication
user_activity();

switch($view) {
// when logout or login is pressed, view intercepts and runs the respective function.
    case "logout":
    logout();
    exit;
    break;

    case "login":
    login();
    exit;
    break;
}
// checks if there is an active session. if there is,
//it sets the default layout to the logged in version
// else the logged out layout is used.
if(isset($_SESSION['valid']) && $_SESSION['valid']) {
    include 'views/main/loggedin.php';
}
else {
include 'views/main/layout.php';
}

?>

```

Layout.php and loggedin.php

Used as the skeleton code (the start and end of the html document)

All pages below this are what is placed inside the class ‘content’.

layout.php loggedin.php

```
<!doctype html><html><head><meta charset="utf-8">
<title>SMS - Not logged in.</title>
<?php scripts_css(); ?>
</head><body>
<div class="container">
  <header>
    <a href="?view=home"><img src="images/test.png"
    width="1024" height="88"  style="display: block;" /> </a>
  </header>
  <div class="sidebar">
    <ul class="nav">
      <li><a href="#">
   <form name="login" action="index.php?view=login" method="post">
              <input type="text" name="username" style='width: 120px'>
              <input type="password" name="password" style='width: 120px'>
              <input type="submit" class='loginbutton' value="Click to login">
  </form></a></li>
    </ul>
    <aside>
      <h4> <?php $t=time();  echo(date("F d Y",$t));  ?>  </h4>
      <h4><?php echo(date("G:i:s",$t));  ?>  </h4>
    </aside>
  <!-- end .sidebar --></div>
  <article class="content">
<?php
// includes this page as root
include 'views/layout/'.$view.'.php' ;
 ?>
  <!-- end .content --></article>
  <footer>
    <p>Disclaimer: This site is constructed as a project at
      Oxford Brookes University. It is not a working website and is
      not connected with any site referenced. The views and opinions
      expressed within these pages are personal and should not be
      construed as reflecting the views and opnions of Oxford
      Brookes University.</p>
    <address>
    </address>
  </footer>
  <!-- end .container --></div>
</body>
</html>
```

layout.php loggedin.php

```
<!doctype html><html><head><meta charset="utf-8">
<title>SMS - Logged in.</title>
<?php  scripts_css();
$type = $_SESSION['type']; ?>
</head>
<body>
<div class="container">
  <header>
    <a href="?view=home"><img src="images/test2.png" alt="SMS Logo"
      name="SMS Logo" width="1024" height="49" id="logoo" style="display: block;" /> </a>
  </header>
  <div class="sidebar">

    <ul class="nav">
      <li><a href='?view=trains'>Setup Trains and Scheduling</a></li>
      <li><a href='?view=reports'>View Data or Reports</a></li>
<?php if ($_SESSION['type'] == 'eng' || $_SESSION['type'] == 'admin') {
    echo"<li><a href='?view=eng'>Engineer Panel</a></li>"; }
      if ($_SESSION['type'] == 'admin') {
    echo"<li><a href='?view=users'>Setup Users</a></li>"; } ?>
    <li><a href='?view=settings'>Settings</a></li>
    <li><a href='?view=logout'>Logout</a></li>
  </ul>
          <aside>
            <h4><?php $t=time();  echo(date("F d Y",$t));  ?>  </h4>
            <h4><?php echo(date("G:i:s",$t));  ?>  </h4>
             <h4><?php  echo $_SESSION['name']; ?></h4>
          </aside>
  </div><!-- end .sidebar1 -->
    <article class="content">
        <?php include 'views/layout/'.$view.'.php'; // includes this page as root ?>
    </article><!-- end .content -->
  <footer>
    <p>Disclaimer: This site is constructed as a project at Oxford Brookes
     University. It is not a working website and is not connected with
     any site referenced. The views and opinions expressed within
     these pages are personal and should not be construed as reflecting
     the views and opnions of Oxford Brookes University.</p>
  </footer>
</div><!-- end .container -->
</body></html>
```

change.php

```php
<?php
db_connect();
jquery_scripts();
if(isset($_SESSION['valid']) ) {
db_setup_menu();
?>
<br><div id='dbsetup'><form name='change'>
        <h4>Change User-Train Responsibility</h4>
            <label for='input'>Train ID, Name:</label>
            <select name='TrainUserID' onclick='checktrainuser("ajax/tu.php")'>
        <?php    $qry= "SELECT trainID, trainName FROM train"; // make settings
                    $result = mysql_query($qry);
                     if(mysql_num_rows($result) == 0) {
                      echo "<option value ='no'>No trains registered. </option>"; } else {
                       while($row = mysql_fetch_assoc($result)) {

                        echo "<option value='".$row['trainID']."'>
                        ".$row['trainID'].", ".$row['trainName']."</option>";

                       }  }?>

                    </select>
                    <label id='check'></labeL>


    <?php       echo"<br><br>
             <label for='input'>User ID to register to:</label>
             <select name='ID'>";

                 $qry= "SELECT * FROM user";
                 $result = mysql_query($qry);
                    while($row = mysql_fetch_assoc($result)) {

                echo "<option value='".$row['userid']."'>".$row['firstname'];
                echo" ".$row['surname']; echo ", ID: ".$row['userid']."</option>";

                    }; ?>

          </select>   <input value="Make Changes" type="button"
           onclick='changetrain("ajax/changetrain.php")'> </form> </div> <br>

         <div id='dbsetup'> <div id="cloon"> Please press Make Changes to continue. </div> </div> <br><br>


<script>

function checktrainuser(url) {
// construct post data
var form     = document.forms['change'];
var name = form.TrainUserID.value;
vars = 'tID=' + escape(name);
if (window.XMLHttpRequest) {
request = new XMLHttpRequest();
}
request.open('POST', url, true);
request.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
request.onreadystatechange = function() {
if (request.readyState == 4) {
document.getElementById("check").innerHTML=request.responseText
}
}
request.send(vars);
}


function changetrain(url) {
var form     = document.forms['change'];
var name = form.TrainUserID.value;
var id = form.ID.value
vars = 'tID=' + escape(name) + '&ID=' + escape(id);
var xmlHttp = false;
if (window.XMLHttpRequest) {
xmlHttp = new XMLHttpRequest();
}
xmlHttp.open('POST', url, true);
xmlHttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
xmlHttp.onreadystatechange = function() {
if (xmlHttp.readyState == 4) {
document.getElementById("cloon").innerHTML=xmlHttp.responseText
}
}
xmlHttp.send(vars);
}
</script>
<?php
} else {
  echo "<br><h3>You must be a registered user to use this site. </h3>
      <br><h3>Please contact your administrator if you cannot login.</h3>
       <br><br><br>";
}
```

edit.php | eng.php | home.php | loginfail.p

```
<style>
#news h3 {
font-color: gray;
}
</style>
<?php
if (isset($_SESSION['name'])) {
    // get news
    db_connect();
    $qry = "SELECT * FROM news";
    $result = mysql_query($qry);
        if($result === FALSE) {
            die(mysql_error('Error, database not setup correctly.'));
        }

echo "<div id='news'>";
            while($row = mysql_fetch_assoc($result))
            {
   echo "<br><h4>News #".$row['newsid']."</h4>
   <h4>".$row['header']."</h4>
        <h4>".$row['content']."</h4>";
}
echo "<br></div>";
}
else
{
    echo "<br><h3>You must be a registered user to use this site. </h3>
      <br><h3>Please contact your administrator if you cannot login.</h3>
      <br><br>
    <br>
    ";
}

?>
```

edit.php | eng.php | home.php | loginfail.php

```php
<?php

if(isset($_SESSION['valid']) ) {

if ($_SESSION['type'] == "admin")
{

db_connect();

error_reporting(E_ERROR | E_PARSE);

echo"<h1> Engineers Panel </h1>";

  $initialqry = "SELECT * FROM repair WHERE repairedat = '0000-00-00 00:00:00'";
  $initialresult = mysql_query($initialqry);


echo"<h3> Unfixed Faults </h3>";

 if(mysql_num_rows($initialresult) == 0) {

echo "<p>No unfixed faults found. </p>"; ?>

<script>

function engineeraction(url, action) {
switch(action)
{
case 'view':
string= 'action=view';
  break;
}
xmlHttp = new XMLHttpRequest();
xmlHttp.open('POST', url, true);
xmlHttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
xmlHttp.onreadystatechange = function() {
if (xmlHttp.readyState == 4) {
    document.getElementById("userresponse").innerHTML=xmlHttp.responseText
    }
}
xmlHttp.send(string);
}
</script>

         
<input value="View fixed faults" type="button" onclick='engineeraction("ajax/eng.php","view")'><br><br><br>
<div id='userresponse'> </div><br><br><br><br><br><br><br><br>

<?php

} else {

// create table
echo "<br><form><div id='dbsetup'><table id='table'>";
echo "<tr><th>Repair #</th><th>Fault Comment</th><th>recorded at</th><th>Train</th><th>SB #</th></tr>";


while($repair = mysql_fetch_assoc($initialresult))
{

  $sbID = $repair['sbID'];

  $qry="SELECT trainID, timechecked, sbName FROM sandbox WHERE sbID= $sbID";
  $result=mysql_query($qry);
  while ($row = mysql_fetch_assoc($result)) {
    $getTID= $row['trainID'];
    $fault_recorded = $row['timechecked'];
    $fault_sbname = $row['sbName'];
  }

  $qry2="SELECT trainName, userID FROM train WHERE trainID= $getTID";
  $result2=mysql_query($qry2);
  while ($row2 = mysql_fetch_assoc($result2)) {
    $fault_train = $row2['trainName'];
  }

   echo "<tr><td><b>".$repair['repairID']."</b>
    </td><td><b>".$repair['faultc']."</b>
    </td><td><b>".$fault_recorded."</b>
    </td><td><b>".$fault_train."</b></td>
    <td><b>".$fault_sbname."</b></td></tr>" ; }    ?>

</table><br>
```

edit.php | eng.php | home.php | loginfail.php

```
82
83  </table><br>
84
85  <h4>Select a repair job before pressing report or request.</h4>
86
87  <?php echo" <select id='modify'> ";
88  $qry= "SELECT repairID FROM repair WHERE repairedat = '0000-00-00 00:00:00'"; // make settings
89  $result = mysql_query($qry);
90  while($row = mysql_fetch_assoc($result)) {
91  echo "<option value='".$row['repairID']."'>".$row['repairID']."</option>";
92  }
93  echo "</select>"; ?>
94
95       
96
97  <input value="Report a fault FIXED" type="button"  onclick='engineeraction("ajax/eng.php","repair")'>
98  <input value="View fixed faults" type="button"  onclick='engineeraction("ajax/eng.php","view")'>
99
100 </div></form><br>
101
102 <div id='dbsetup'>  <div id="userresponse"> Press a button to carry out an action. </div> </div><br><br>
103
104 <script>
105
106 function engineeraction(url, action) {
107
108 var modify= document.getElementById('modify').value
109
110
111 switch(action)
112 {
113
114 case 'repair':
115 string= 'action=repair' + '&modify=' + escape(modify);
116   break;
117
118 case 'view':
119 string= 'action=view' + '&modify=' + escape(modify);
120   break;
121
122   case 'repairfix':
123  var comment= document.getElementById('repairc').value
124  string= 'action=repairfix' + '&comment=' + escape(comment) + '&modify=' + escape(modify);
125   break;
126
127 }
128
129 xmlHttp = new XMLHttpRequest();
130 xmlHttp.open('POST', url, true);
131 xmlHttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
132
133 xmlHttp.onreadystatechange = function() {
134
135 if (xmlHttp.readyState == 4) {
136
137     document.getElementById("userresponse").innerHTML=xmlHttp.responseText
138
139     }
140
141 }
142 xmlHttp.send(string);
143 }
144
145 </script>
146
147 <?php } // else for if no repairs found.
148
149 } else { // IF USER DOES NOT HAVE CORRECT ADMIN ACCOUNT ?>
150
151 <h1>Engineer Panel</h1>
152 <br>
153 <h2> You do not have the correct level of access to view this page. </h2>
154 <br>
155 <h2> Please contact your administrator if you have any questions. </h2>
156 <br><br><br><br><br><br><br><br><br><br>
157 <?php }
158
159
160
161 } else {
162
163   echo "<br><h3>You must be a registered user to use this site. </h3>
164
165     <br><h3>Please contact your administrator if you cannot login.</h3>
166
167     <br><br>
168   <br>
169   ";
170 }
```

loginfail.php | reports.php | sbreports.php

```
<script>

function send(url) {

var term= document.getElementById('terms').value

data = 'term=' + escape(term);

var xmlHttp = false;

if (window.XMLHttpRequest) {
    xmlHttp = new XMLHttpRequest();
}

xmlHttp.open('POST', url, true);
xmlHttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

    xmlHttp.onreadystatechange = function() {

        if (xmlHttp.readyState == 4) {

            document.getElementById("visualchart").innerHTML=xmlHttp.responseText
        }

    }

xmlHttp.send(data);
}

</script>

<h1> View Data in List </h1>

<a href='?view=sbreports'class='button'>View Data in List Format</a>
<a href='?view=visual'class='button'>Visualized Reports</a>

<br><br><br>

<div id='visual'>
 <form>
    <label for='input'>View in list format:</label>

        <select name='chart' id='terms'>
            <option value='user'>Users</option>
            <option value='train'>Trains</option>
            <option value='sandbox'>Sandbox</option>
            <option value='traintypes'>Train Types</option>
            <option value='repair'>Repairs</option>

        </select>

<input value="View" type="button" onclick='JavaScript:send("ajax/list.php")'>
</div>
<div id='visualchart'> Please select something to view. </div> </form>

<br><br><br>
<br><br><br>
<br><br><br>
<br><br><br>
<?php
} else {

  echo "<br><h3>You must be a registered user to use this site. </h3>

      <br><h3>Please contact your administrator if you cannot login.</h3>

     <br><br><br>";
} ?>
```

schedule.php

```php
<?php
db_connect();
jquery_scripts();
if(isset($_SESSION['valid']) ) {
db_setup_menu();
?>
<br><div id='dbsetup'><form name'setupschedule'>

<h4>Setup Schedule</h4>

<label for='input'>Select train (ID, name):</label>

<select id='tIDs'>

<?php    $qry= "SELECT trainID, trainName FROM train"; // make settings
         $result = mysql_query($qry);

             if(mysql_num_rows($result) == 0) {
                echo "<option value='no'>No trains registered. </option> </select> <br><br> </div> <br><br><br><br><br>";

                } else {

                while($row = mysql_fetch_assoc($result)) {

                echo "<option value='".$row['trainID']."'>".$row['trainID'].", ".$row['trainName']."</option>";

                } ?>
</select>

        <input value='Lookup' type='button' onclick='schedule("ajax/schedule.php","check")'> </form> </div><br>

<div id='dbsetup'><div id='schedule'> Press Lookup to continue. </div> </div><br><br><br><br>

<?php } ?>

<script>

function schedule(url, form) {

var tid = document.getElementById('tIDs').value

switch(form)
{

case 'check':
string= 'action=check' + '&tID=' + escape(tid);
break;

case 'apply':

var sch = document.getElementById('sch').value
var e = document.getElementById('entry').value
var ed = document.getElementById('entryD').value
var v = document.getElementById('vacate').value
var vd = document.getElementById('vacateD').value

string= 'action=apply' + '&tID=' + escape(tid)
+ '&sch=' + escape(sch)+ '&e=' + escape(e)
+ '&ed=' + escape(ed)+ '&v=' + escape(v)+ '&vd=' + escape(vd);

break;

case 'schedule':
string= 'action=schedule' + '&tID=' + escape(tid);
break;

case 'schedule2':
string= 'action=schedule2' + '&tID=' + escape(tid);
break;

}

var xmlHttp = false;

if (window.XMLHttpRequest) {
xmlHttp = new XMLHttpRequest();
}

xmlHttp.open('POST', url, true);
xmlHttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
xmlHttp.onreadystatechange = function() {

if (xmlHttp.readyState == 4) {

    switch(form)
        {

        case 'check':
        document.getElementById("schedule").innerHTML=xmlHttp.responseText

        break;
```

```
            break;

            case 'apply':
            document.getElementById("confirm").innerHTML=xmlHttp.responseText
            break;

            case 'schedule':

            document.getElementById("schedule").innerHTML=xmlHttp.responseText

            $( "#entryD" ).datepicker({ minDate: 0, maxDate: "1M" });
            $( "#vacateD" ).datepicker({ minDate: 0, maxDate: "7D" });
            $('#entry').timepicker({ 'scrollDefaultNow': true });
            $('#vacate').timepicker({ 'scrollDefaultNow': true });

            break;

            case 'schedule2':

            document.getElementById("schedule").innerHTML=xmlHttp.responseText

            $( "#entryD" ).datepicker({ minDate: 0, maxDate: "1M" });
            $( "#vacateD" ).datepicker({ minDate: 0, maxDate: "7D" });
            $('#entry').timepicker({ 'scrollDefaultNow': true });
            $('#vacate').timepicker({ 'scrollDefaultNow': true });

            break;

            }

        }
    }
    xmlHttp.send(string);
    }

    </script>
    <?php
    } else {

      echo "<br><h3>You must be a registered user to use this site. </h3>

          <br><h3>Please contact your administrator if you cannot login.</h3>

         <br><br><br>";
    } ?>
```

settings.php

```php
<?php
if(isset($_SESSION['valid']) ) { ?>
<script>
function send(url, form) {
var head= document.getElementById('heading').value
var cont= document.getElementById('content').value
var numb= document.getElementById('numb').value
var max= document.getElementById('max').value
var loginm= document.getElementById('logmessage').value
var mobhead= document.getElementById('mobhead').value
var mobcont= document.getElementById('mobcont').value

switch(form)
{

case 'news':
string= 'head=' + escape(head) + '&cont=' + escape(cont) + '&numb=' + escape(numb);
  break;
case 'max':
string= 'max=' + escape(max);
  break;
    case 'mnews':
string= 'head=' + escape(mobhead) + '&cont=' + escape(mobcont);
  break;
     case 'mlogin':
string= 'message=' + escape(loginm);
  break;
}

xmlHttp = new XMLHttpRequest();
xmlHttp.open('POST', url, true);
xmlHttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

xmlHttp.onreadystatechange = function() {

if (xmlHttp.readyState == 4) {

        switch(form)
        {

        case 'news':
         document.getElementById("news").innerHTML=xmlHttp.responseText
          break;
        case 'max':
        document.getElementById("maxset").innerHTML=xmlHttp.responseText
          break;

           case 'mnews':
        document.getElementById("mnewsresponse").innerHTML=xmlHttp.responseText
          break;

          case 'mlogin':
        document.getElementById("mloginresponse").innerHTML=xmlHttp.responseText
          break;

        }

    }

}
xmlHttp.send(string);
}

</script>

<h1> Settings </h1>

<form id='dbsetup'><h4><u>Post news on front page:</u></h4>

<p>Note: there are a maximum of 3 news items, to delete, simply save new news over the existing number.</p>

<label>Heading:</label><br><input type='text' value='Put heading here...' style='height: 25px; width: 250px;' id='heading'/>

<label>Save as news number:</label>
    <select id='numb'><option value='1'>1</option><option value='2'>2</option><option value='3'>3</option>
 </select><br><br>

<label>Content:</label><input type='text' value='Put content here...' style='height: 25px; width: 500px;' id='content'/><br>

<input value="Save" type="button" onclick='JavaScript:send("ajax/settings/news.php","news")'>

<div id="news"> </div> </form>




<br><form id='dbsetup'>
    <h4><u>Set mobile application login message.</u></h4>
    <input type='text' value='Login Message' style='height: 25px; width: 500px;' id='logmessage'/>
      <input value="Set" type="button" onclick='JavaScript:send("ajax/settings/moblogin.php","mlogin")'>
    <div id="mloginresponse"> </div>
```

```php
<?php
// BUTTONS http://www.cssbuttongenerator.com
db_connect();
jquery_scripts();
if(isset($_SESSION['valid']) ) {
db_setup_menu();

?>
<br><div id='dbsetup'><h4>Add Train</h4><form name ='addtrain'>

<div id='gap'>
<label for='input'>Train Name:</label>
<input type='textarea' name='TrainName' id='textarea'>
</div><br>


<div id='gap'>
<label for='input'>User ID to register to:</label>
<select name='ID'>
<?php
$qry= "SELECT * FROM user";
$result = mysql_query($qry);
while($row = mysql_fetch_assoc($result)) {
echo "<option value='".$row['userid']."'>".$row['firstname']; echo " ".$row['surname']; echo ", ID: ".$row['userid']."</option>";
}
echo "</select><br> <br> <label for='input'>Type:</label><select name='trainType'>";
$qry= "SELECT * FROM traintypes"; // make settings
$result = mysql_query($qry);
while($row = mysql_fetch_assoc($result)) {
echo "<option value='".$row['trainType']."'>".$row['trainType']."</option>";
}
echo "</select><br> ";

$o=$_SESSION['maxsb']; // get max sb
echo "<br>
<label for='input'>Number of Sandboxes:</label>
<select name='numberSb'>";

for ($i=1; $i<=$o; $i++)
{
echo"<option value='".$i; echo"'>".$i; echo"</option>";
}

echo"</select></div>"; ?>

<br>

<label for='input'>Entry time:</label>
<input type='text' id= 'entry'  class='time' style='width: 98px'>

<label for='input'>Vacate time:</label>
<input type='text' id='vacate'  class='time' style='width: 98px'>

<br><br>

<label for='input'>Entry Date:</label>
<input type='text' id= 'entryD' name='entryD'  style='width: 98px'>

<label for='input'>Vacate Date:</label>
<input type='text' id='vacateD' name='vacateD'  style='width: 98px'>
<br><br>

<input value="ADD" type="button"  onclick='trainAdd("ajax/addtrain.php")'></form></div><br>


<div id='dbsetup'>  <div id="addresponse"> Fill in the fields and click ADD. </div> </div><br>


<script>

$(function() {

$('#entry').timepicker({ 'scrollDefaultNow': true });
$('#vacate').timepicker({ 'scrollDefaultNow': true });
});

</script>

<script>

$(function() {

$( "#entryD" ).datepicker({ minDate: 0, maxDate: "1M" });
$( "#vacateD" ).datepicker({ minDate: 0, maxDate: "7D" });
});


function trainAdd(url) {
var xmlHttp = false;
if (window.XMLHttpRequest) {
xmlHttp = new XMLHttpRequest();
}
```

users.php

```php
<?php
if(isset($_SESSION['valid']) ) {

 $type = ($_SESSION['type']);
if ($type == "admin")
{

db_connect();

echo"<h1> User Setup </h1> ";

  $qry = "SELECT * FROM user";
  $result = mysql_query($qry);

// create table
echo "<br><form><div id='dbsetup'><table id='table'>";
echo "<tr><th>Firstname</th><th>Surname</th><th>User Type</th>
<th>Username</th><th>Password</th><th>Email</th></tr>";

// no need for validation as there must be a user to be on this page

while($row = mysql_fetch_assoc($result))
{

  echo "<tr>
  <td><b>".$row['firstname']."</b></td>
  <td><b>".$row['surname']."</b></td>
  <td><b>".$row['type']."</b></td>
  <td><b>".$row['username']."</b>
  <td><b>".$row['password']."</b>
  <td><b>".$row['email']."</b></tr>"; }  ?>

</table>
           
</form>
<h4>To perform an action on a user, please select one from
  the drop down menu: (you need to refresh to reflect changes of Add and Modify)</h4>
<?php

echo"<select id='modify'>";

$result = mysql_query($qry);

while($row = mysql_fetch_assoc($result)) {

echo "<option value='".$row['userid']."'>".$row['surname']."</option>"; }

echo "</select>"; ?><br>

    <input value="Add New User" type="button"  onclick='user("ajax/user.php","add")'>
<input value="Modify Selected User" type="button"  onclick='user("ajax/user.php","modify")'>
<input value="Delete Selected User" type="button"  onclick='user("ajax/user.php","delete")'> </div>

<br><br><div id='dbsetup'>  <div id="userresponse"> Press a button to carry out an action. </div> </div><br><br>


<?php } else { // IF USER DOES NOT HAVE CORRECT ADMIN ACCOUNT ?>

<h1>Users</h1>
<br>
<h2> You do not have the correct level of access to view this page. </h2>
<br>
<h2> Please contact your administrator if you have any questions. </h2>
<br><br><br><br><br><br><br><br><br><br><br>
<?php } ?>


<script>

function user(url, form) {

switch(form)
{

case 'add':
  string= 'action=add';
break;
```

users.php

```
case 'adduser':

  var fn= document.getElementById('firstname').value
  var sn= document.getElementById('surname').value
  var um= document.getElementById('username').value
  var pw= document.getElementById('password').value
  var type= document.getElementById('type').value
  var mail= document.getElementById('email').value

  string= 'action=adduser' + '&fn=' + escape(fn) + '&sn=' + escape(sn)
  + '&um=' + escape(um) + '&pw=' + escape(pw) + '&type=' + escape(type) + '&mail=' + escap
break;

case 'modify':
  var modify= document.getElementById('modify').value
  string= 'modify=' + escape(modify) + '&action=modify';
break;

case 'modifyuser':
  var modify= document.getElementById('modify').value
  var fn= document.getElementById('mfirstname').value
  var sn= document.getElementById('surname').value
  var um= document.getElementById('username').value
  var pw= document.getElementById('password').value
  var type= document.getElementById('type').value
  var mail= document.getElementById('email').value

  string= 'action=modifyuser' + '&fn=' + escape(fn) + '&sn=' + escape(sn)
  + '&um=' + escape(um) + '&pw=' + escape(pw) + '&type=' + escape(type)
  + '&mail=' + escape(mail) + '&modify=' + escape(modify);

break;

case 'delete':
  var modify= document.getElementById('modify').value
  string= 'modify=' + escape(modify) + '&action=delete';
break;

case 'deleteuser':
  var modify= document.getElementById('modify').value
  string= 'modify=' + escape(modify) + '&action=deleteuser';
break;

}

xmlHttp = new XMLHttpRequest();
xmlHttp.open('POST', url, true);
xmlHttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

xmlHttp.onreadystatechange = function() {

if (xmlHttp.readyState == 4) {

      switch(form)
        {

        case 'add':
        document.getElementById("userresponse").innerHTML=xmlHttp.responseText
        break;

        case 'adduser':
          document.getElementById("addvalidate").innerHTML=xmlHttp.responseText

        break;

        case 'modify':
         document.getElementById("userresponse").innerHTML=xmlHttp.responseText
        break;

        case 'modifyuser':
          document.getElementById("modifyvalidate").innerHTML=xmlHttp.responseText

        break;

        case 'delete':
         document.getElementById("userresponse").innerHTML=xmlHttp.responseText
        break;

        case 'deleteuser':
          document.getElementById("userresponse").innerHTML=xmlHttp.responseText
          window.setTimeout(function(){location.reload()},3000)
        break;

        }

    }

}
xmlHttp.send(string);
}

</script>
```

function.php

```php
<?php

function user_activity()
{
    // time to log out in seconds - 8 mins
    $inactivity = 480;

    if(isset($_SESSION['timeout']) ) {

        $active_session = time() - $_SESSION['timeout'];

        if($active_session > $inactivity) {
            session_destroy();
            timeout();
        }
    }

    $_SESSION['timeout'] = time();

}


function login()
{
    $user = ($_POST['username']);
    $pw = ($_POST['password']);
    session_start(); //must call session_start before using any $_SESSION variables
    db_connect();
    $qry = "SELECT userid FROM user WHERE username='$user' AND password='$pw'";
    $result = mysql_query($qry);

        if(mysql_num_rows($result) < 1) //no such user exists
        {
            header('Location: ?view=loginfail');
            die();
        }
            else
        {
            $qry= "SELECT * FROM user WHERE username='$user' AND password='$pw'";
            $result = mysql_query($qry);
            $row = mysql_fetch_array($result);

                //sets the session data for this user
                session_regenerate_id (); //this is a security measure
                $_SESSION['valid'] = 1;
                $_SESSION['userid'] = $result;
                $_SESSION['type'] = $row['type'];
                $_SESSION['name'] = $row['firstname']. ' ' .$row['surname'];

            $qry = "SELECT * FROM settings";
            $result = mysql_query($qry);
            $row = mysql_fetch_array($result);

                $_SESSION['maxsb'] = $row['addMaxSB'];

        }
                //redirect to another page or display "login success" message
                header('Location: ?view=home');
}



function logout()
{
    $_SESSION = array(); //destroy all of the session variables
    session_destroy();
    header('Location: ?view=home');
}

function timeout()
{
    $_SESSION = array(); //destroy all of the session variables
    session_destroy();
    header('Location: ?view=timeout');
}
```

```php
function db_connect()
//connects to sots and selects database
 {

  $connection = mysql_pconnect('localhost', '09034276', 'Ticket1');

     if(!$connection)
     {
         return false;
     }

     if(!mysql_select_db('09034276'))
     {
          return false;
     }

     return $connection;

 }

function db_setup_menu()
{

     echo"<h1> Train and Scheduling Setup </h1>
     <div id='center'>
     <table><tr>
     <td><a href='?view=trains' class='button'>Add Train</a></td>
     <td><a href='?view=change' class='button'>Change User-Train</a></td>
     <td><a href='?view=schedule' class='button'>Setup Schedule</a></td>
     <td><a href='?view=edit' class='button'>Edit/Delete Train</a></td>
     </tr></table>
     </div>";

}

function scripts_css()
{

     echo"<link rel='stylesheet' href='styles/style.css'  type='text/css'>";

}


function jquery_scripts()
{

echo"
<script type='text/javascript' src='http://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js'></script

  <script src='//ajax.googleapis.com/ajax/libs/jqueryui/1.10.1/jquery-ui.min.js'></script>

  <script type='text/javascript' src='js/jquery.timepicker.js'></script>
  <link rel='stylesheet' type='text/css' href='styles/jquery.timepicker.css' />
    <link rel='stylesheet' type='text/css' href='styles/jquery.ui.datepicker.css' />
  <link rel='stylesheet' type='text/css' href='styles/datepicker.css' />
  <link rel='stylesheet' type='text/css' href='styles/base.css' />";


}


?>
```

visual.php

```php
<?php
if(isset($_SESSION['valid']) ) { ?>

<link rel='stylesheet' href='styles/datepicker.css'  type='text/css'>
<script>

// jquery datepicker initialized

  $(function() {
    $( "#datefrom").datepicker({
      showOtherMonths: true,
        changeMonth: true,
      changeYear: true,
      selectOtherMonths: true
    });
    $( "#datetill").datepicker({
      showOtherMonths: true,
        changeMonth: true,
      changeYear: true,
      selectOtherMonths: true
    });
  });

function ajax(url) {
// gets the element with the ID of 'report'.
var report = document.getElementById('report').value
// builds the variable sent via $_POST
  vars = 'rep=' + escape(report);
// allows the updating of the page asynchronously
if (window.XMLHttpRequest) {
// defines the request as xmlHttp
  xmlHttp = new XMLHttpRequest();
  }
// sets the request up as POST, the url passed to this function
// is the destination php script that recieves the post data.
xmlHttp.open('POST', url, true);
xmlHttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
xmlHttp.onreadystatechange = function() {
// when the readystate is reached,
// this means the script has sent back its response
if (xmlHttp.readyState == 4) {
  // this places the responsetext into the div called visualchart.
      document.getElementById("visualchart").innerHTML=xmlHttp.responseText
    }

  }
// sends the built variable containing the POST data.
  xmlHttp.send(vars);
}
</script>

<?php
echo"<h2> Visualized Data </h2>
<a href='?view=sbreports'class='button'>View Data in List Format</a>
<a href='?view=visual'class='button'>Visualized Reports</a>

<br><br><br>
      <form id='dbsetup'>
    <label>Report type:</label>
        <select name='chart' id='report'>
            <option value='1'>Missed Sandboxes</option>
            <option value='2'>Trains Delayed</option>
            <option value='3'>Fault Occurance</option>
            <option value='4'>Average sand levels prior to filling</option>
        </select>";  ?>

      <input value="Get Charts" type="button" onclick='JavaScript:ajax("ajax/chart.php")'>

</form>

      <div id='visualchart'> Chart not yet generated.  </div>
<br><br><br>
<br><br><br>
<br><br><br>

  <?php
} else {
  echo "<br><h3>You must be a registered user to use this site. </h3>

      <br><h3>Please contact your administrator if you cannot login.</h3>

     <br><br><br>";
} ?>
```

Ajax – addtrain (adds a train), changetrain (change train – user), chart(calculates charts)

addtrain.php

```php
<?php
include('function.php');
session_start();
db_connect();

// validation

$formvalue = array('trainName', 'entry', 'vacate', 'entryD', 'vacateD');

foreach($_POST AS $key => $value)
{
    if(in_array($key, $formvalue) && $value == '')
    {
        $missed[] = "<h4><font color='red'>$key is required.</font></h4>";
    }
}

if (isset($missed)) {

    echo "<h4><font color='red'>Cannot add train, please enter missing data.</font></h4>";

            foreach($missed as $missed)
            {
                echo $missed;
            }
    }

$trainName= ($_POST['trainName']);

if (strlen($trainName) >= 11) {
    echo "<h4><font color='red'>Train name must be less than 12 characters.</font></h4>";
}

if ((isset($missed)) || (strlen($trainName) >= 11))  { }
else {

$numberSb = ($_POST['numberSb']);
$trainType = ($_POST['trainType']);
$userID = ($_POST['ID']);

$e = ($_POST['entry']); // entry time
$v = ($_POST['vacate']); // vacate time

$entryTime  = date("H:i:s", strtotime($e)); // formatted time
$vacateTime  = date("H:i:s", strtotime($v));


$ed =($_POST['entryD']); // dates
$vd =($_POST['vacateD']);

$edf = date("Y-m-d", strtotime($ed)); // formatted dates
$vdf = date("Y-m-d", strtotime($vd));

$entryt = $edf." ".$entryTime; // combine date and time
$vacatet = $vdf." ".$vacateTime;

$checkname="SELECT trainName FROM train WHERE trainName = '$trainName'";
$nameresult = mysql_query($checkname);
if(mysql_num_rows($nameresult) > 0){

    echo"<h4><font color='red'>Please use a different Train Name</font></h4>";

}
else {

$addt=("INSERT INTO  `09034276`.`train` (
`trainID` ,
`trainName` ,
`numberSb` ,
`trainType` ,
`userID` ,
`entryt`,
`vacatet`)

VALUES (NULL ,  '$trainName',  '$numberSb',  '$trainType', '$userID', '$entryt', '$vacatet');");
```

```php
$qry="SELECT trainID FROM train ORDER BY trainID DESC LIMIT 1";
$result = mysql_query($qry);
while($row = mysql_fetch_assoc($result)) {
$trainID=$row['trainID'];
}


$o=$numberSb;

for ($i=1; $i<=$o; $i++)
  {

$insert=("INSERT INTO  `09034276`.`sandbox` (
`sbName` ,
`checked` ,
`trainID`)

VALUES ('$i', 'Not checked', '$trainID');");

            mysql_query($insert) or die('query2 DIED');

}
                echo "<h4><font color='green'> Database updated with train: ".$trainName; echo".</font></h4>";

                    echo "<h4> Last 5 trains added: </h4>";

                    $qry="SELECT trainID, trainName, numberSb, trainType FROM train ORDER BY trainID DESC LIMIT 5";

                    $result = mysql_query($qry);

                    echo"<table id='table'><thead><th scope='col'>Train ID</th><th scope='col'>TrainName</th><th scope

                    while($row = mysql_fetch_assoc($result)) {

                            echo "<tr><td><b>".$row['trainID']."</b>
                                    </td><td><b>".$row['trainName']."</b>
                                    </td><td><b>".$row['numberSb']."</b>
                                     </td><td>".$row['trainType']."</td></tr>";
                                }

                            echo "</tbody></table>";

}

}


?>
```

changetrain.php

```php
<?php

include('function.php');
session_start();
db_connect();

$tid = ($_POST['tID']);
$uid = ($_POST['ID']);

            $update="UPDATE  `train` SET  `userID` =  $uid WHERE  `train`.`trainID` =$tid";

            if ($tid == 'no') {

                echo "No changes made, as no trains are registered.";
            } else {

        mysql_query($update);

    echo "<h3><font color='green'>Success: Updated Train ".$tid." with User ".$uid."</font></h3>"; }

?>
```

CHART

chart.php

```php
<?php

require ('function.php');
db_connect();
$rep= ($_POST['rep']); // report type
require ('gChart.php');

switch($rep) {
// when logout or login is pressed, view intercepts and runs the respective function.
    case "1":

    missedsb();
    exit;
    break;

    case "2":
    latetrain();
    exit;
    break;

    case "3":
    faultregularity();
    exit;
    break;

    case "4":
    averagesandlevel();
    exit;
    break;
}


function latetrain() {
// checks trains that were not finished checking before they were meant to depart.

$sql = "SELECT vacatet, checkedat, userID FROM train WHERE checkedat > 0 AND checkedat > vacatet";
$result = mysql_query($sql);
$total = (mysql_num_rows($result));
if ($total < 1) {
    echo "There are no trains that have been checked late, or there is not enough data to generate a report.";
} else {

while ($row = mysql_fetch_assoc($result)) {
    $blahsbarray[]=$row['userID'];
}

$dataset = implode(',', (array_count_values($blahsbarray)));
$sql = 'SELECT surname FROM user WHERE userid IN (' . implode(',', array_map('intval', $blahsbarray)) . ')';
$result = mysql_query($sql);
while ($row = mysql_fetch_assoc($result)) {
    $surnamea[]=$row['surname'];
}

$ppiChart = new gPieChart(800,350);
$ppiChart->addDataSet(array($dataset));
$ppiChart->setLabels(array($surnamea));
$ppiChart->setLegend(array($surnamea));
$ppiChart->setColors(array("ff3344", "11ff11", "22aacc"));
$surnames = implode(', ', $surnamea);

echo "<h4>This chart shows the number of times each user has been late finishing the sandbox checks for a train.";

echo "<br><br>Late in this case, means that the time at which checks were complete was after the train should have left.</h4>";

echo "<img src="; echo $ppiChart->getUrl(); echo" />";

echo "<br><br>Users Involved: <b>".$surnames."</b>";
echo "<br># of times final checks were late: <b>".$dataset."</b>";

} // end no results

}
```

chart.php

```php

function missedsb() {
// chart to show number of occurences where sandbox were completly missed before the train had to depart

$qry = "SELECT DISTINCT userID FROM train WHERE NOW() > vacatet AND checkedat = '0000-00-00 00:00:00' ORDER BY userID";
$result = mysql_query($qry);
$total = (mysql_num_rows($result));

if ($total < 1) {
    echo "No missed sandboxes.";
} else {

while ($row = mysql_fetch_assoc($result)) {

    $users[]=$row['userID'];
}

$missedcount = array();

foreach($users as $id) {

            $sql = "SELECT trainID, vacatet, checkedat FROM train
            WHERE NOW() > vacatet AND checkedat = '0000-00-00 00:00:00' AND userID = $id";
            $result = mysql_query($sql);

            while ($row = mysql_fetch_assoc($result)) {

                $TIDa[]=$row['trainID'];
            }

            $trains = implode(', ', $TIDa);

            $qry=("SELECT sbID FROM sandbox WHERE trainID IN ({$trains})");
            $result=mysql_query($qry);

                while($row = mysql_fetch_assoc($result)) {
                    $sbarray[]=$row['sbID'];
                }

            $sblist= implode(',', $sbarray);

                $qrySB=("SELECT count(sbID) FROM sandbox WHERE sbId IN ({$sblist}) AND checked = 'Not checked'");

                    $sb_checked_qry = mysql_query($qrySB);

                        while($sbrow = mysql_fetch_assoc($sb_checked_qry))
                        {
                            $notcheckedassoc = $sbrow;
                        }

            if (isset($notcheckedassoc)) {

            foreach ($notcheckedassoc as $key=>$val)
            $notchecked=$val;
            $missedcount[] = $val;
            }

            echo "<br>User ID: <b>".$id."</b>, Occurance of missed sandboxes after train has left: <b>".$notchecked."</b>";

}

$sql = 'SELECT surname FROM user WHERE userid IN (' . implode(',', array_map('intval', $users)) . ')';
$result = mysql_query($sql);

while ($row = mysql_fetch_assoc($result)) {
    $surnamea[]=$row['surname'];
}

$barChart = new gBarChart(800,350,'g');

foreach($missedcount as $key => $value) {

        $barChart->addDataSet(array($value));

    }

$barChart->setLegend(array($surnamea));
$barChart->setColors(array("ff3344", "11ff11", "22aacc"));
$barChart->setGradientFill('c',0,array('FFE7C6',0,'76A4FB',1));
$barChart->setAutoBarWidth();

echo "<br><br><img src="; echo $barChart->getUrl(); echo" />";

} // end no results

}
```

chart.php

```php

function missedsb() {
// chart to show number of occurences where sandbox were completly missed before the train had to depart

$qry = "SELECT DISTINCT userID FROM train WHERE NOW() > vacatet AND checkedat = '0000-00-00 00:00:00' ORDER BY userID";
$result = mysql_query($qry);
$total = (mysql_num_rows($result));

if ($total < 1) {
    echo "No missed sandboxes.";
} else {

while ($row = mysql_fetch_assoc($result)) {

    $users[]=$row['userID'];
}

$missedcount = array();

foreach($users as $id) {

            $sql = "SELECT trainID, vacatet, checkedat FROM train
            WHERE NOW() > vacatet AND checkedat = '0000-00-00 00:00:00' AND userID = $id";
            $result = mysql_query($sql);

            while ($row = mysql_fetch_assoc($result)) {

                $TIDa[]=$row['trainID'];
            }

            $trains = implode(', ', $TIDa);

            $qry=("SELECT sbID FROM sandbox WHERE trainID IN ({$trains})");
            $result=mysql_query($qry);

                while($row = mysql_fetch_assoc($result)) {
                    $sbarray[]=$row['sbID'];
                }

            $sblist= implode(',', $sbarray);

                $qrySB=("SELECT count(sbID) FROM sandbox WHERE sbId IN ({$sblist}) AND checked = 'Not checked'");

                    $sb_checked_qry = mysql_query($qrySB);

                        while($sbrow = mysql_fetch_assoc($sb_checked_qry))
                        {
                            $notcheckedassoc = $sbrow;
                        }

            if (isset($notcheckedassoc)) {

            foreach ($notcheckedassoc as $key=>$val)
            $notchecked=$val;
            $missedcount[] = $val;
            }

            echo "<br>User ID: <b>".$id."</b>, Occurance of missed sandboxes after train has left: <b>".$notchecked."</b>";

}

$sql = 'SELECT surname FROM user WHERE userid IN (' . implode(',', array_map('intval', $users)) . ')';
$result = mysql_query($sql);

while ($row = mysql_fetch_assoc($result)) {
    $surnamea[]=$row['surname'];
}

$barChart = new gBarChart(800,350,'g');

foreach($missedcount as $key => $value) {

        $barChart->addDataSet(array($value));

    }

$barChart->setLegend(array($surnamea));
$barChart->setColors(array("ff3344", "11ff11", "22aacc"));
$barChart->setGradientFill('c',0,array('FFE7C6',0,'76A4FB',1));
$barChart->setAutoBarWidth();

echo "<br><br><img src="; echo $barChart->getUrl(); echo" />";

} // end no results

}
```

chart.php

```php

function faultregularity() {

$qry = "SELECT sbID, fault_occ FROM fault_occ ORDER BY sbID";
$result = mysql_query($qry);

$total = (mysql_num_rows($result));
if ($total < 1) {
    echo "Not enough data to generate report.";
} else {

while ($row = mysql_fetch_assoc($result)) {

    $sb[]=$row['sbID'];
    $fault[]=$row['fault_occ'];
}

if (!array_filter($fault)) {
   echo "Not enough data to generate report.";
} else {

echo "<h4>Fault Occurance by Sandbox (ID):</h4>";

$barChart = new gBarChart(800,350,'g');
$barChart->addDataSet(array($fault));
$barChart->setLegend(array($sb));
$barChart->setColors(array("ff3344", "11ff11", "22aacc"));
$barChart->setGradientFill('c',0,array('FFE7C6',0,'76A4FB',1));
$barChart->setAutoBarWidth();

echo "<br><br><img src="; echo $barChart->getUrl(); echo" />";

echo "<br><br><h4>Fault Occurance by Train (ID):</h4>";
$qry= "SELECT
  sandbox.trainID AS id
FROM sandbox
  INNER JOIN fault_occ ON fault_occ.sbID = sandbox.sbID
GROUP BY sandbox.trainID";

$result = mysql_query($qry);

while ($row = mysql_fetch_assoc($result)) {
    $id[]=$row['id'];
}

$qry= "SELECT AVG(fault_occ.fault_occ) AS average
FROM sandbox
  INNER JOIN fault_occ ON fault_occ.sbID = sandbox.sbID
GROUP BY sandbox.trainID";

$result = mysql_query($qry);

while ($row = mysql_fetch_assoc($result)) {

    $avg2[]=$row['average'];

}

$tbarChart = new gBarChart(800,350,'g');
$tbarChart->addDataSet(array($avg2));
$tbarChart->setLegend(array($id));
$tbarChart->setColors(array("ff3344", "11ff11", "22aacc"));
$tbarChart->setGradientFill('c',0,array('FFE7C6',0,'76A4FB',1));
$tbarChart->setAutoBarWidth();

echo "<br><br><img src="; echo $tbarChart->getUrl(); echo" />";

} // end array not empty

} // end if total > 0

} // end function

```

chart.php

```php
} // end if total > 0

} // end function



function averagesandlevel() {

echo "<h4>Average sandlevels by Sandbox (ID) (scale 1-100):</h4>";

$qry = "SELECT sbID, avg_level FROM fault_occ ORDER BY sbID";
$result = mysql_query($qry);

$total = (mysql_num_rows($result));
if ($total < 1) {
    echo "Not enough data to generate report.";
} else {

while ($row = mysql_fetch_assoc($result)) {

    $sb[]=$row['sbID'];
    $avg[]=$row['avg_level'];
}

if (!array_filter($avg)) {
   echo "Not enough data to generate report.";
} else {

$barChart = new gBarChart(800,350,'g');
$barChart->addDataSet(array($avg));
$barChart->setLegend(array($sb));
$barChart->setTitle("All Tickets by Issue Type");
$barChart->setGradientFill('c',0,array('FFE7C6',0,'76A4FB',1));
$barChart->setAutoBarWidth();

echo "<img src="; echo $barChart->getUrl(); echo" />";

echo "<br><br><h4>Average sandlevels by Train (ID) (scale 1-100):</h4>";
$qry= "SELECT
  sandbox.trainID AS id
FROM sandbox
  INNER JOIN fault_occ ON fault_occ.sbID = sandbox.sbID
GROUP BY sandbox.trainID";

$result = mysql_query($qry);

while ($row = mysql_fetch_assoc($result)) {
    $id[]=$row['id'];
}

$qry= "SELECT AVG(fault_occ.avg_level) AS average
FROM sandbox
  INNER JOIN fault_occ ON fault_occ.sbID = sandbox.sbID
GROUP BY sandbox.trainID";

$result = mysql_query($qry);

while ($row = mysql_fetch_assoc($result)) {

    $avg2[]=$row['average'];

}

$tbarChart = new gBarChart(800,350,'g');
$tbarChart->addDataSet(array($avg2));
$tbarChart->setLegend(array($id));
$tbarChart->setColors(array("ff3344", "11ff11", "22aacc"));
$tbarChart->setGradientFill('c',0,array('FFE7C6',0,'76A4FB',1));
$tbarChart->setAutoBarWidth();

echo "<img src="; echo $tbarChart->getUrl(); echo" />";

} // end array not empty

} // end if total > 0

} // end function



?>
```

…. around 50% more Ajax code exists, but please look at it on the cd.