Oundle School Transport Reservation System

Matthew Riegels Computing Coursework 2025

Contents

[Analysis 3](#_Toc172627572)

[Project description 3](#_Toc172627573)

[Current system details 3](#_Toc172627574)

[Description of the problem 4](#_Toc172627575)

[Computational Solution 4](#_Toc172627576)

[Stakeholders 6](#_Toc172627577)

[Stakeholder 1: Kristian Fewster 6](#_Toc172627578)

[Research 8](#_Toc172627579)

[Essential features 9](#_Toc172627580)

[Limitations 10](#_Toc172627581)

[Hardware & software requirements 11](#_Toc172627582)

[Success criteria 12](#_Toc172627583)

[Design 13](#_Toc172627584)

[Solution Decomposition 13](#_Toc172627585)

[Tables 16](#_Toc172627586)

[TblVehicles 16](#_Toc172627587)

[TblUsers 16](#_Toc172627588)

[TblRequests 18](#_Toc172627589)

[Entity relationship diagram 19](#_Toc172627590)

[Structure 20](#_Toc172627591)

[Generic page design 20](#_Toc172627592)

[Colours 21](#_Toc172627593)

[Usability Features 22](#_Toc172627594)

[Detailed page design 23](#_Toc172627595)

[Development 31](#_Toc172627596)

[Table SQL Code: 32](#_Toc172627597)

[TblUsers 32](#_Toc172627598)

[TblVehicles 32](#_Toc172627599)

[TblRequests 32](#_Toc172627600)

[Connection Page 33](#_Toc172627601)

[Install Page 35](#_Toc172627602)

[Vehicle pages 36](#_Toc172627603)

[Validation 37](#_Toc172627604)

# Programming project (Component 03 or 04) marking criteria AO 2.2 Analysis (maximum 10 marks) marks The candidate Will have: • Identified some features that make the problem solvable by computational methods. • Identified suitable stakeholders for the project and described them and some of their requirements. • Identified some appropriate features to incorporate into their solution. • Identified some features of the proposed computational solution. • Identified some limitations of the proposed solution. • Identified some requirements for the solution. • Identified some success criteria for the proposed solution. 3—5 marks • Described the features that make the problem solvable by computational methods. • Identified suitable stakeholders for the project and described how they will make use of the proposed solution. • Researched the problem looking at existing solutions to similar problems identifying some appropriate features to incorporate into their solution. • Identified the essential features Of the proposed computational solution. • Identified and described some limitations of the proposed solution. • Identified most requirements for the solution. • Identified some measurable success criteria for the proposed solution. — 70 marks marks • Described the features that make the problem solvable by computational methods and why it is amenable to a computational approach. • Identified suitable stakeholders for the project and described them and how they will make use of the proposed solution and why it is appropriate to their needs. • Researched the problem in depth looking at existing solutions to similar problems identifying and describing suitable approaches based on this research. • Identified and described the essential features of the proposed computational solution. • Identified and explained any limitations of the proposed solution. • Specified the requirements for the solution including (as appropriate) any hardware and software requirements. • Identified measurable success criteria for the proposed solution. 9—10 marks • Described and justified the features that make the problem solvable by computational methods, explaining why it is amenable to a computational approach. • Identified suitable stakeholders for the project and described them explaining how they will make use of the proposed solution and why it is appropriate to their needs. • Researched the problem in depth looking at existing solutions to similar problems, identifying and justifying suitable approaches based on this research. • Identified the essential features of the proposed computational solution explaining these choices. • Identified and explained with justification any limitations of the proposed solution. • Specified and justified the requirements for the solution including (as appropriate) any hardware and software requirements. • Identified and justified measurable success criteria for the proposed solution. O marks = no response or no response worthy of credit. Analysis

## Project description

*Described and justified the features that* make *the problem solvable by computational methods, explaining why it is* amenable *to a computational approach.*

### Current system details

When a teacher at school wants to organise transportation for a group of students, they have to email the school’s transportation office and request a minibus with a certain capacity for a certain time. Staff requesting transport can either drive themselves or choose to use a hired driver. The school has a small number of hired drivers on casual contracts. Additionally, the school sometimes hires minibuses instead of using those from its own fleet. When a staff member makes a request, they must provide the same information every time: details like the destination and required capacity of the vehicle.

* Required information for a request:
  + Vehicle type,
  + Date/time required,
  + Destination/use,
  + Date/time returned,
  + Whether a hired driver is required.
* Booking process:
  + Provide all above information for each vehicle.
  + If you request a driver, the job is added to a shared spreadsheet and an email is sent to the group of drivers notifying them; the drivers choose to take jobs for times when they are available.
    - The drivers are not full-time employees, and as such are not obliged to take the jobs. As a result, this part can take a few days.
  + Once a driver has accepted a job, the requestor will be informed that their request has been accepted.
  + The current system aims to respond within 4 days of a request.
  + Requestors can talk to drivers directly, but any change of plan must still be reflected in the spreadsheet so that the drivers are appropriately paid, and the office knows when they are busy.
* Additional details:
  + The school currently has four drivers currently hired.
  + They have the same recruitment checks as permanent staff, but they have casual contracts (they can work as much or little as they want).
  + There are some drivers currently on holiday (general point being that driver time off should be accounted for).

### Description of the problem

The problems with the current system that I am trying to solve are as follows:

* Requests come through by email to the office that manages transportation, and those emails can get lost in the inbox, or take a long time to be actioned on by the office.
  + The new system would need to make unread requests more obvious.
* The number of staff that can drive large minibuses is dwindling due to some recent legislation change in the UK which means that staff need special training to legally drive the minibuses on their licence.
  + The current system can’t deal with the heightened demand for drivers.
  + The new system would need to streamline requests, allowing reduced numbers of drivers to handle a greater number of requests.
* It is often unclear whether all drivers are aware that a job is available.
  + The new system will make jobs obviously and readily visible to drivers on the main page. This will ensure that all drivers can clearly see what jobs are available to them.
  + I should add a means by which drivers can indicate that they have read a job so that the coordinator can confirm that all drivers are aware of their options.

### Computational Solution

I aim to upgrade this system, and to build the new solution I will use a web-based database. Users will fill in an online digital form to provide request information.

* Since the same information must be given each time in the same format, a digital form is suitable for this use. Furthermore, since this information will be stored in an online database, the fields of the form can be easily added to a database table, further streamlining the process of requesting. Also, since all required information will be asked for in the form, the office won’t have to send reply emails asking for more information, which slows down the process of requesting.
* The requests all being added to the same database table means that all requests are in a single centralized location which can easily referenced, and the requests can be sorted based on whether they have been read or accepted or neither. This will allow the transport office to keep track of all requests and respond more quickly.

A web-based database system can easily monitor whether a driver has viewed the details of a job with a simple tick box for each driver, and can send automated emails to drivers when jobs are available to them to remind them to check the system.

## Stakeholders

Identified suitable stakeholders for the project and described them explaining how they will make use of the proposed solution and why it is appropriate to their needs.

In general, any stakeholders will be in one of the following categories:

* Coordinator.
  + The only candidate for this category is Kristian Fewster ([kfewster@oundleschool.org.uk](mailto:kfewster@oundleschool.org.uk)), the school’s transport manager.
  + A coordinator uses the system to ensure that the driver knows about any details of the job, and that the requestor knows that their job has been taken. The coordinator is also responsible for making sure that the drivers are aware of the jobs available.
  + The system will be able to send automated emails to relevant people, as well as store request information online in a centralized location to make the details of a particular job clear.
* Driver.
  + Drivers need to see the details of the jobs they can choose to take. It should also be clear which jobs are available and which have already been taken.
  + Available jobs will be posted on the online system with all relevant information provided.
* Requestor.
  + Requestors need to provide the information relevant to their request.
  + There will be a fixed online form to enter this information.

### Stakeholder 1: Kristian Fewster

Kristian Fewster is the school’s transport manager. His role in school is to oversee and manage all of the school’s transportation. The system will be useful to him because it will collect all information pertaining to transport arrangements in one centralized location for him to reference. He will also be able to change some of the arrangements if necessary. The system is especially suited to his needs due to the ease of access provided by a centralized web-based database. Not only is it easy for him to access and reference, it is also easy for staff members and drivers to make and respond to requests, which is convenient to his role since it is essential that everyone is up to date on information and referencing the same plan.

Kristian helpfully answered some of my questions:

* Could you shed some light on the recent changes in British licencing laws and how they have affected transportation at school?
  + “As far as British Driving legislation, I would suggest you go on the DVLA Website as this explains everything to do with transport licencing laws and is up to date.”
* How much do the aesthetics of the website matter to you?
  + “Aesthetically, it needs to be simple to use and have clear steps as all levels of IT competencies would have access including the more mature drivers.”
* What problem with the current system would you most like to avoid in the new system?
  + “The problem we currently have is a that we use a shared spreadsheet that I populate with details of a trip, then the casual drivers look through and put their names against them, but I need to know if all drivers have seen them. Below is part of our current sheet and details I need.”



Considering this feedback and the screenshot, I will

* Ensure the design of the website is straightforward and simple, so that technologically inexperienced users will not have difficulty using the system.
* Create a feature that lets Kristian know whether drivers have seen that a job is available and read the details of said job.
* Give the request form a field for a postcode for destinations outside of school.
* Similarly, add a field for the general purpose of the trip (such as “CA”, “Cricket”, “Learning for Life”) in the screenshot above.

## Research

Researched the problem in depth looking at existing solutions to similar problems, identifying and justifying suitable approaches based on this research.

## Essential features

Identified the essential features of the proposed computational solution explaining these choices.

Essential features of the website will include the following:

* A page where users can log in to the website.
  + This will be the first page shown to the user, and no other page will be accessible until the user has logged in.
  + The pages that are accessible after this will depend on the role of the user logged in.
* A page where Kristian can view and modify all transport arrangements.
  + This will be laid out like a calendar/timetable, each day being a column with vertical space representing the time of day.
  + He should be able to view both pending requests and active jobs.
  + He should be able to see whether a driver has read a job or not.
    - This is so that he can confirm whether a driver has read a job and is unavailable or whether they have not checked available jobs.
  + This is all so that Kristian can oversee and manage all transportation arrangements as an ultimate authority.
* A page where the school’s drivers can view all available jobs that haven’t been accepted by anyone else, and accept them if they choose to.
  + This is so that the drivers can volunteer for their own jobs, since they are on casual contracts.
* A page where staff members can submit requests.
  + This page will have a digital form with all required information, so that .
* A way for staff members to remove their requests once they’re made in case plans change.
  + If only some details have been changed, the staff member can make another request with the modified details.

## Limitations

Identified and explained with justification any limitations of the proposed solution.

With regards to paying the drivers, I will be unable to process payments directly on the website. This is because there are some legal issues I’d like to avoid, along with user data security issues. Additionally, I would have to buy an SSL certificate which can cost up to a thousand pounds annually. Given my lack of experience with web-based development, I think that navigating the complex issues that accompany handling money transfers would be too difficult for me to be a feasible part of the system.

Instead, I will likely make use of an online based payment solution, since I will be able to integrate it into the website without having to do the payment processing myself. This will circumvent many of the issues mentioned above, since I can pass the role of managing user data onto the company.

I will not create a downloadable sister app for the website. While this may be a convenience to customers, it is beyond my capabilities and it would be especially difficult to make the app available on all devices since it would have to function on multiple different operating systems, thereby requiring two separate coded solutions to function on these different devices. This limitation can be mitigated by making the website easily accessible on a phone by ensuring that my CSS code uses mobile-friendly Bootstrap styles.

Furthermore, this limitation is not likely to be a major hindrance to the project on the whole, since the database will be stored on a central server which cannot be accessed without an Internet connection. Given that an internet connection is required for the system to work, one will always be able to access the website in any situation where the system could work on an app.

## Hardware & software requirements

Specified and justified the requirements for the solution including (as appropriate) any hardware and software requirements.

The software requirements to implement the final project will include making use of Visual Studio Code for the sake of easy traversal of the many files involved in the final product. I will code the solution using PHP and MySQL to access database tables and output the results of these queries. I will also be using HTML to create the structure of the website, using Bootstrap and CSS in order to ensure the final website has a pleasant and appropriate look and feel by styling the HTML. I am choosing to use these languages as these are the languages that we have been taught in school and so I am most proficient with, in addition to the fact that they lend themselves very well to web-based development, allowing me to easily code the final solution with them.

As for the Hardware requirements of the final project, I will use Xampp for a local hosting app, which I need in order to create and host the databases and tables used in the final system. Xampp is the local hosting app that I am most familiar with, and it is also free and pre-installed on the school computers, which I will be using to complete this project. Xampp also allows for an easy view of the tables, so that I can easily see the data within them. Eventually once I am ready to host the final project, I will need to use a free online web server to host the website so it is available for customers to access from anywhere at all times.

## Success criteria

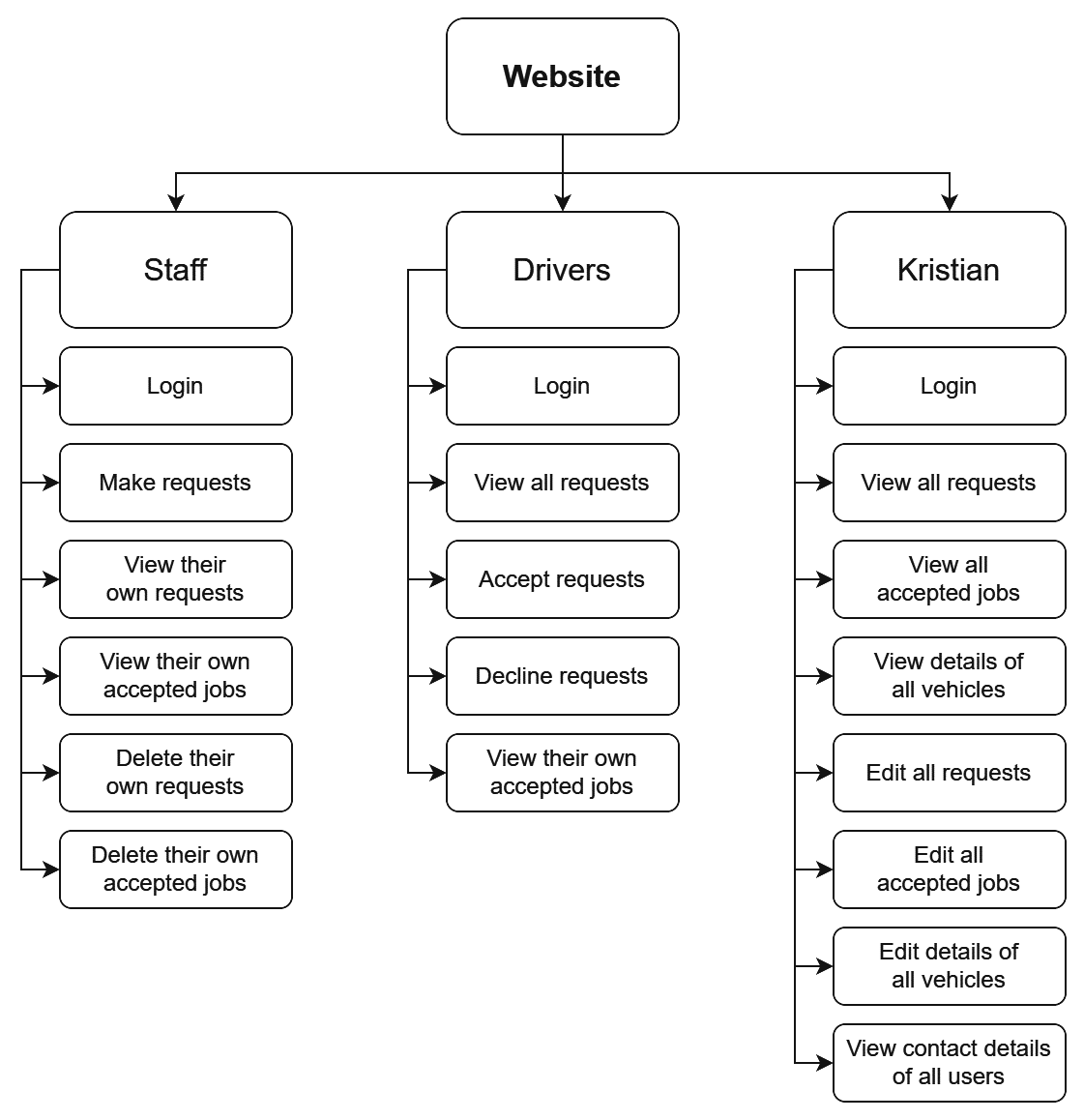
Identified and justified measurable success criteria for the proposed solution.

* My main success criterion is that the website must be easy and intuitive to use and understand.
  + This will mean that users will be able to make use of the website without requiring any help or training, thus significantly streamlining the process of implementing the new system.
  + This will be measured by asking users how easy they find the website to use, and then they will provide a personal rating on a scale from 1-10 on a usability questionnaire. I aim to achieve an average score of 7 or above.
* Another success criterion is that making a request must take less than 10 minutes.
  + This is to make the new system more attractive for staff members, who will not want to waste time adapting to a new method of requesting.
* Users must be able to log in to their account in less than a minute.
  + This is to ensure that users are not frustrated by the process of wasting vast quantities of time logging in to their account.
* Users must not be able to access any pages their role doesn’t allow them to (for example, a staff member shouldn’t be able to view and manage all transport arrangements, but Kristian should).
  + This is to ensure that the only people who can modify a job are Kristian and whoever requested it.
* Drivers must be able to view all their active jobs in one place easily.
  + This is so that they can manage their working hours effectively, and is also a convenience that will make the new system more attractive to them.

# AO 3.1 Design (maximum 15 marks) 1—4 marks The candidate will have: Described elements of the solution using algorithms. Described some usability features to be included in the solution. • Identified the key variables / data structures / classes (as appropriate to the proposed solution). • Identified some test data to be used during the iterative or post development phase Of the process. 5 8 marks • Broken the problem down systematically into a series of smaller problems suitable for computational solutions describing the process. • Defined the structure of the solution to be developed. • Described the solution fully using appropriate and accurate algorithms. • Described the usability features to be included in the solution. • Identified the key variables / data structures / classes (as appropriate to the proposed solution) and any necessary validation. • Identified the test data to be used during the iterative development of the solution. • Identified any further data to be used in the post development phase. 9—12 marks • Broken the problem down systematically into a series of smaller problems suitable for computational solutions explaining the process. Defined in detail the structure of the solution to be developed. • Described the solution fully using appropriate and accurate algorithms explaining how these algorithms form a complete solution to the problem. Described, explaining choices made, the usability features to be included in the solution. • Identified and justified the key variables / data structures / classes (as appropriate to the proposed solution) explaining any necessary validation. • Identified and justified the test data to be used during the iterative development of the solution. • Identified and justified any further data to be used in the post development phase. 13—15 marks • Broken the problem down systematically into a series of smaller problems suitable for computational solutions, explaining and justifying the process. Defined in detail the structure of the solution to be developed. Described the solution fully using appropriate and accurate algorithms justifying how these algorithms form a complete solution to the problem. Described, justifying choices made, the usability features to be included in the solution. • Identified and justified the key variables / data structures / classes (as appropriate to the proposed solution) justifying and explaining any necessary validation. • Identified and justified the test data to be used during the iterative development of the solution. • Identified and justified any further data to be used in the post development phase. O marks = no response or no response worthy of credit. Design

## Solution Decomposition

Broken the problem down systematically into a series of smaller problems suitable for computational solutions, explaining and justifying the process.

I will use decomposition to break down the problem with a top-down design chart as shown on the right to display this. I am doing this in order to make the final coding of the solution easier for myself by simplifying the design and both showing and justifying the final structure of the site.

The first page every user sees is the login page where they enter their details and can then (if the details are correct) view other pages.

The website will have a navigation bar visible at all times that links to every page a user has access to.

Members of staff will have access to:

* A page that contains the digital form for submitting requests.
  + This is so that they can easily submit requests without omitting any information, which streamlines the entire process and helps Kristian and the driver too.
* A page that contains all the requests they have made.
  + This is so that they are reminded not to forget about the details of the jobs they’ve requested.
  + This is also so that they are able to cancel their requests if they change their mind about the details of any of them.
* A page that contains all the accepted jobs they have requested.
  + This page will likely be merged with the previous page for the sake of simplicity, and since it serves all the same purposes – those being to remind staff members about their jobs and also to allow them to cancel them.
* They will be able to delete their requests and their active jobs on the pages listed above, so separate pages are not required to provide that capability.

Drivers will have access to:

* A page that allows them to view the details of all requests that require a hired driver, and either accept or decline these requests.
  + Drivers will not be able to view requests they cannot accept. This includes staff members who are driving themselves (but still need a vehicle) and staff members who are already being driven by another driver.
  + Drivers will be able to view all details of a request including the time and date when it starts and finishes, the destination and purpose of the trip, the number of people being transported, and the staff member who requested it. They will also be able to view the details of the vehicle that has been assigned to that job, if any.
    - This is to ensure that they can make the most informed decision possible when they choose whether to volunteer for the job.
* A page that allows them to view all jobs they currently have active.
  + This page will display all the above-mentioned details of all the jobs they have accepted and not completed.
  + This is so that drivers can easily manage their working hours and avoid accidentally overcommitting.
  + This page may also have a feature where drivers can view their jobs on a timetable-like setup where the jobs are shown visually, where days are columns and time of day is represented in vertical space.

Kristian will have access to:

* A page that shows the details of all active jobs and requests.
  + This includes staff members that are driving themselves and those that need a hired driver.
  + This also includes jobs that both have and have not been accepted by any driver.
  + On this page Kristian will be able to see all the details of all jobs including the time and date when they start and finish, the destination and purpose of the trip, the number of people being transported, the staff member who requested it, the driver who accepted the job (if any), the drivers that declined the job (if any), and any drivers that have done neither (ie haven’t read it).
  + On this page he will also be able to modify the details of or delete any request or active job.
  + He will also be able to see which vehicles have been assigned to which jobs, and view their details.
  + This page may be split into several subpages – one for accepted jobs and one for requests, for instance. Alternatively, I may include a feature whereby Kristian can filter to show jobs based on their details.
* A page that shows the details of all vehicles in the school’s fleet.
  + On this page Kristian would be able to view the details of every vehicle at the school’s disposal including their registration number, capacity, and the date (if any) after which they will not be available for use.
  + He will also be able to view and modify which jobs any vehicle is currently assigned to.
  + This is so that Kristian can ensure that the vehicles are used efficiently and that there are vehicles free at all times to accept new jobs.
* A page that displays all users on the system and some of their details.
  + On this page Kristian will be able to view some of the account details of both staff members and drivers. These will include their full name, email address, telephone number, role (driver or staff), and (in the case of drivers) the number of hours they’ve worked and haven’t been paid for.
  + This is so that Kristian can contact people to ask them to provide additional information about jobs and requests if required, or to inform them of any changes made to any jobs or requests they’re involved in.

## Tables

For the validation, I plan to make sure that certain fields are required, as this will ensure that some data is always entered into relevant fields, which avoids the issues that occur when I try to retrieve the data stored in the tables. When a maximum character limit is set, this is to ensure that no ridiculous lengths of values are input which would take up unnecessary table space, or perhaps even crash the table if a suitably immense amount of data was provided. In places where a range of values is required by the table, this is to make sure that no absurd values are input.

### TblVehicles

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field name** | **Data type** | **Validation** | **Typical data** | **Justification** |
| VehicleID | INT(6), auto increment, Primary key | Auto-generated. | 14 | Unique identifier for each vehicle on the system. This will be used as a foreign key in the jobs table. |
| RegNumber | VARCHAR(7) | Required. Must exactly be two uppercase letters, followed by two numbers, then three uppercase letters. | BD15SMR | Registration number of each vehicle for tax and legal purposes. |
| Capacity | INT(2) | Required. Must be an integer. | 10 | Indicates how many students can ride on the vehicle (driver and front seat not included). |
| NotAvailableFrom | DATE | Must be a date in the future when entered.  YYYY-MM-DD | 2025-04-22 | Indicates date after which the vehicle will be unavailable (if it has been hired or needs maintenance etc.). |

### TblUsers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field name** | **Data type** | **Validation** | **Typical data** | **Justification** |
| UserID | INT(6), auto increment, Primary key | Auto-generated. | 19 | Unique identifier for each user. This will be used as a foreign key in the jobs table. |
| Password | VARCHAR(60) | Required. Plaintext passwords must be entered with at least 6 characters. Since they are hashed, they will always be exactly 60 characters long when stored. | $2y$10$.vGA1O9wmRjrwAVXD98HNOgsNpDczlqm3Jq7KnEd1rVAGv3Fykk1a | Password is required to log in to the system, which is required to view or do anything on the system. Will be hashed for security reasons. |
| Email | VARCHAR(50) | Required. Max 50 characters. Must have exactly one ‘@’, and at least one ‘.’ afterwards. | smith.j@oundleschool.org.uk | Allows contacting users to remind them of upcoming jobs, confirm submission of requests, etc. |
| TelephoneNumber | VARCHAR(11) | Required. Must be 11 characters, all numbers. | 07305712268 |  |
| Forename | VARCHAR(30) | Required. Max 30 characters. | John | First name of user |
| Surname | VARCHAR(30) | Required. Max 30 characters. | Smith | Surname of user |
| IsDriver | TINYINT(1) | Required. Boolean. | False | Whether the user can access the driver pages or not. |
| IsAdmin | TINYINT(1) | Required. Boolean. | False | Whether the user has access to admin pages or not. Only Kristian will have this toggled to True. |
| IsRequestor | TINYINT(1) | Required. Boolean. | True | Whether the user has access to the request pages or not. |
| HoursWorked | INT(3) | Auto-generated. | False | Will be used to calculate driver wages since they are on casual contracts. |

### TblRequests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field name** | **Data type** | **Validation** | **Typical data** | **Justification** |
| RequestID | INT(6), auto increment, Primary key | Auto-generated. | 12 | Unique identifier for each request. |
| DateTimeOut | DATETIME | Required. Must be in the future at time of entry.  YYYY-MM-DD HH:MM:SS | 2024-05-17 10:30:00 | Time and date when the job starts. |
| TimeIn | TIMESTAMP | Required. Must be after DateTimeOut.  HH:MM:SS | 12:40:00 | Time and date when the job ends. Jobs never span multiple days. |
| Destination | VARCHAR(20) | Required. | Uppingham | Name of destination so drivers know where they’re going. |
| Postcode | VARCHAR(8) | Required. Min length 5, max 8. Must be a valid postcode. | LE15 9SE | Postcode that can be input into a navigation device if needed. |
| Purpose | VARCHAR(20) | Required. | Netball | The department making the request, or the reason why the trip is necessary. |
| ReqCapacity | INT(2) | Required. The number of people needing to be taken. | 7 | This is so that an appropriate vehicle can be assigned to the request. |
| DriverID | INT(6) | This will be managed by the system, not input. | 9 | This is the UserID of the user driving the job. It may be one of the hired drivers or the requestor themselves. |
| VehicleID | INT(6) | This will be managed by the system, not input. | 14 | This is the VehicleID of the vehicle assigned to the job. Left blank if no vehicle has been assigned. |
| RequestorID | INT(6) | Auto-generated. | 19 | This is the UserID of the requestor. |

### TblDeclinedDrivers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field name** | **Data type** | **Validation** | **Typical data** | **Justification** |
| DriverID | INT(6) | Auto-generated. | 10 | This is the UserID of the driver that has declined the request. |
| RequestID | INT(6) | Auto-generated. | 13 | This is the RequestID of the request that was declined by the driver. |

### Entity relationship diagram

DriverID  
RequestID

**DeclinedDrivers**

UserID  
Password  
Email  
Telephone number  
Forename  
Surname  
IsDriver  
IsAdmin  
IsRequestor  
Hours worked

**Users**

RequestID  
Date/time out  
Time in  
Destination  
Postcode  
Purpose  
Required capacity  
DriverID  
VehicleID  
RequestorID

**Requests**

VehicleID  
Registration number  
Capacity  
NotAvailableFrom

**Vehicles**

## Structure

Defined in detail the structure of the solution to be developed.

The Structure of the final product will be a website coded with HTML and styled with CSS and bootstrap that uses PHP and MySQL to access the tables in the database.

Kristian has advised that the website have a simple and intuitive layout that is easy to understand for those users that may be less familiar with online technology. Following this guidance, I have decided to include a navbar that is always visible at the left-hand side of the page that contains clearly labelled links to all pages that a user has access to.

### Generic page design

As mentioned previously, I will design each page with a navbar on the left side always, so all pages will look something like this, with specifics filled in of course.

Page contents will be in here. Most pages will consist of listed objects. This will be displayed like below.

Another page

Pages will be listed down the side like this.

Page two

**Current page**

**Current page**

John Smith Log out



**List item 1**

*Relevant information provided here.*

**List item 2**

*Relevant information provided here.*

**List item 3**

*Relevant information provided here.*

Since this is intended for internal school use, it should be appropriately branded. Since the majority of school products we use have the school logo in the top left corner of the page, I decided to follow suit to ensure the branding is consistent. The links displayed on the navbar will depend on the role of the currently logged in user.

* Staff members need pages to:
  + Make and submit a request.
  + View and potentially cancel their currently logged requests.
* Drivers need pages to:
  + View all requests made by staff members that need drivers.
  + View all requests they’ve accepted.
* Kristian needs pages to:
  + View all requests whether or not they need a driver, have been allocated a driver, or allocated a vehicle.
  + View all vehicles.
  + View contact details for all users.

### Colours

Since this will be a product used internally by the school, it should be appropriately branded as mentioned above. This also means that the website should be made with the school’s official colours. In order to find the exact right colours, I took a screenshot of the school website and then colour-picked the appropriate colours using Microsoft Paint.



**#9F1532**

**#EAECF3**

**#012F60**

The navy-blue colour used as the background for the “Follow us on Instagram” banner has hex value #012F60 and RGB values (1, 47, 96).

The maroon colour used for “Interactive map” and “Menu” in the header as well as the play button of the video in the lower right corner has hex value #9F1532 and RGB values (159, 21, 50).

The pale blue-grey colour used in the background has hex value #EAECF3 and RGB values (234, 236, 243).

These are the colours I will use for the final product since they will properly reflect the school’s branding. I will use the pale blue-grey as a background color for header pages (to help distinguish them from the main body sections, which will be in white). I will use the maroon to distinguish on the sidebar which page the user is currently on, since it stands out well against the navy blue, which I will use for the navbar (on which will be white text).

### Usability Features

The navbar at the left-hand side of the page will be visible on every page on the website, so it provides easy access to everywhere else on the site. The navbar will also stay in the same place on the screen even if users scroll down the page, further increasing the ease of navigation to anywhere on the site.

All links will be clearly labelled and easy to understand so that even technologically inexperienced users will be able to understand where to go easily.

I will use a plain background with black text for main body content since this is easy to read for users who may have eyesight issues, and the slight color tint of the background also helps readability for users who have dyslexia.

I will use 16px font (the default for HTML), since this size is not so small that users would have to strain to read it, but so large as to be impractical for displaying the information that needs to be viewed by users on the system. Since most pages will have quite little information shown (e.g. the page listing all of a staff member’s active jobs), I considered increasing this even more, but since other pages will have a large amount of information I decided to stick with 16px everywhere because having a different font size on different pages would be too jarring and may confuse some users.

Much of the website pages will be displaying lists of information (e.g. the list of jobs a driver can accept or the list of vehicles), and as such I will ensure that sequential items on any list are clearly divided to prevent confusion of information between different items. To this end I may also decide that list items alternate between background colours between adjacent items, although this will depend on the difficulty of implementing such a feature.

I will have the navbar always visible with the full page names shown at all times. The reason I am not using a hamburger menu button is that the extra step of opening a menu to access the links is often confusing to technologically inexperienced individuals.

These usability features will make sure that all users are able to use this website easily, since some staff members may be less confident with IT skills. This is also especially important because Kristian informed me that some of the elderly drivers may be less competent with IT. Additionally, some partially visually impaired users may need to access the final product, and by choosing to include these features I will be making sure that they are able to use the site without unnecessary hindrance, and without having to overly strain themselves.

### Detailed page design

I will now go through each page one at a time discussing:

* The page overview,
* Page layout,
* Usability features,
* Data structures and variables,
* Validation and algorithms.

#### Login page

##### Page overview

The login function allows drivers and staff members to log in to the site and access its services depending on their role. It also allows Kristian to log in which allows him to access his admin pages. This page will have a straightforward design with the input boxes in the centre of the screen and the school logo centred above them. The log in for the website will authenticate users with their email address and password.

##### Page layout

email

password



##### Usability features

Since this page is very skeletal and has little functionality, there isn’t much to say here.

##### Data structures and variables

Data structures:

Tables:

TblUsers – to verify the login function.

No variables are used on this page

##### Validation

As per table designs.

##### Algorithms

Login function:

* When the submit button is pressed:
* Get the user’s password from TblUsers with the SQL command (“SELECT Password FROM TblUsers WHERE Forename = Username”).
* IF the password is verified to be correct:
  + Create a session variable.
  + Determine which pages the user can access by getting their role from the database.
  + (“SELECT IsDriver, IsAdmin, IsRequestor FROM TblUsers WHERE Forename = Username”)
  + Redirect to appropriate pages depending on role of user.
* ELSE:
  + Print “Credentials not entered correctly”

#### Request page

##### Page overview

This page will contain the digital form that will allow staff members to submit requests to the system. The layout of this page will make the form as easy to fill in as possible, without taking up vast quantities of space.

##### Page layout

Purpose:

Required capacity:

Postcode:

Destination:

Time in:

Time out:

Date of job:

View active jobs

**Submit a request**

**Submit a request**

John Smith Log out



**Submit**

##### Usability features

This page will follow the main usability features of the generic page design.

For this page I need to ensure that the inputs are easy to use and edit for the user. I will ensure that the input fields explain what they are for through the use of labels and placeholders, to ensure the user knows what to put in what input field.

I will also make use of dropdown selection menus so the user can quickly and easily make their choice for these inputs, as well as ensuring they don’t enter erroneous values which might occur if it was just a text input box.

##### Data structures and variables

Data structures:

This page will submit the requests to TblRequests with the filled in information, adding the RequestorID automatically.

Variables:

Session variable – to get the ID of the requestor.

DateTimeOut – will be assembled into one variable from the two input boxes date and time.

All other form inputs will be stored in a record in TblRequests.

##### Validation

As per table designs.

##### Algorithms

Loading in the navbar:

* If no user logged in:
  + Redirect to login page.
* If user logged in:
  + Get role values from TblUsers:
    - If IsRequestor is True: show requestor pages on navbar.
    - If IsDriver is True: show driver pages on navbar.
    - If IsAdmin is True: show admin pages on navbar.

Request submission:

* Upon press of the submit button:
  + Redirect to requests\_add.php with all details from the fields entered.
  + Info is then validated on that page.
  + If all validation checks are passed:
    - The new request is added to TblRequests with all input fields.
  + If any data is invalid:
    - No new request is added to TblRequests, and an error message is printed to the screen: “data entered incorrectly – please try again”.

#### Active requests page

##### Page overview

This page will list all the requests of the currently logged in user, separated into those that have and haven’t been accepted. Each item on the list will display some cursory information about the request (enough to identify it), and the user can click on any of the listed requests to open the request details page (design below) for that request. On that page will be a full description of all relevant information pertaining to that request.

##### Page layout

Accepted jobs:

Requested jobs:

**08/05/2025**

*13:30 – 17:00 Rowing regatta*

**19/04/2025**

*12:00 – 17:00 Cricket away fixture*

**07/02/2025**

*18:00 – 22:00 House cinema trip*

**View active jobs**

Submit a request



**View active jobs**

John Smith Log out

##### Usability features

This page will follow the main usability features of the generic page design.

The requests are clearly marked as interactable by the arrows on their right side, helping those users who may be technologically inexperienced to navigate the site.

Jobs will be arranged in date order to help prioritize reminding users of the soonest jobs.

##### Data structures & variables

This page will read the data for the list of requests from TblRequests, using the UserID of the currently logged-in user (from the session variable) to ensure that only their requests are shown.

This page needs to use the session variable, and will also need to use variables to split the DateTimeOut variable read from TblRequests into a date and a time component for displaying as in the above page layout.

##### Validation

Since this page never writes any data, no validation is required.

##### Algorithms

Loading in the navbar:

* If no user logged in:
  + Redirect to login page.
* If user logged in:
  + Get role values from TblUsers:
    - If IsRequestor is True: show requestor pages on navbar.
    - If IsDriver is True: show driver pages on navbar.
    - If IsAdmin is True: show admin pages on navbar.

Loading user requests:

* If no user logged in, or if user is not a staff member:
  + Redirect to login page.
* If staff member logged in:
  + Get all requests made by that staff member from TblRequests.
  + I will use the MySQL command (“SELECT DateTimeOut, TimeIn, Purpose FROM TblRequests WHERE UserID = current\_user\_id”).
    - Where current\_user\_id stands for the UserID of the user that is currently logged in.

Clicking on a particular request:

* This will not be a link using an <a> tag, but will instead be a button that redirects to the request details page and posts the RequestID of the request that was clicked on, so that the details of that request can be shown on that page.

#### Request details page

##### Page overview

This page will display all the details of a particular job. This page will display the driver’s name and contacts, and the vehicle’s reg number and max capacity if either have been assigned. From this page, the user that made the request will be able to cancel it.

##### Page layout

Back to previous page

**Request details.**

Date: 07/02/2025  
Time: 18:00 – 22:00  
Destination: Corby Cinema  
Postcode: NN17 1QG  
Purpose: House cinema trip  
Required capacity: 14  
Requested by: John Smith

Cancel request

**Job assignment details.**

Driver name: Ronald Ferret  
Driver contacts: 07305712268  
ferret.ro@oundleschool.org.uk  
  
Vehicle reg number: AB34EFG  
Vehicle capacity: 16

**Request details**

John Smith Log out

View active jobs

Submit a request



Alternative design to represent different possibilities:

Back to previous page

**Request details.**

Date: 08/05/2025  
Time: 13:30 – 17:00  
Destination: Henley  
Postcode: RG9 3DB  
Purpose: Rowing Regatta  
Required capacity: 12  
Requested by: John Smith

Decline request

Accept request

**Job assignment details.**

No driver assigned  
  
No vehicle assigned

**Request details**

Ronald Ferret Log out

Your accepted jobs

**View pending jobs**



##### Usability features

The ‘cancel request’ button will ask for confirmation to prevent accidental deletion of requests, and it is coloured red to clearly indicate that it is important. This will help technologically inexperienced users to use the site.

The request details and job assignment details are separated into coloured text boxes to increase readability.

The ‘accept request’ and ‘decline request’ buttons are placed far apart and are coloured appropriately to make sure one is not clicked by accident when the other is intended.

##### Data structures & variables

This page will read data from all three tables. It will also write to TblRequests if the request is cancelled by the user.

This page will have to use variables to split DateTimeOut into date and time parts for display as above.

##### Validation

Since no data is entered into this page, no validation is required.

##### Algorithms

Loading in the navbar:

* If no user logged in:
  + Redirect to login page.
* If user logged in:
  + Get role values from TblUsers:
    - If IsRequestor is True: show requestor pages on navbar.
    - If IsDriver is True: show driver pages on navbar.
    - If IsAdmin is True: show admin pages on navbar.

Loading the job details:

* Query TblRequests to get all details of the selected request.
  + I will use this MySQL command: (“SELECT \* FROM TblRequests WHERE RequestID = request\_id”).
    - Where request\_id stands for the RequestID of the request that the user clicked on. This will be sent from the active requests page to the request details page by using post.
* To get the job assignment details, I will query both TblUsers and TblVehicles using the following MySQL commands. Either of these commands will only be executed as long as a vehicle/driver is assigned in the first place.
  + (“SELECT Forename, Surname FROM TblUsers WHERE UserID = driver\_id”)
    - Where driver\_id stands for the UserID of the driver assigned to this job, which was retrieved by the previous MySQL command.
  + (“SELECT RegNumber, Capacity FROM TblVehicles WHERE VehicleID = vehicle\_id”)
    - Where vehicle\_id stands for the VehicleID of the vehicle assigned to this job, which was retrieved by the previous MySQL command.

‘Cancel request’ button:

* If the UserID in the session variable matches the RequestorID of the request:
  + The ‘cancel request’ button is added to the page.
    - This will not happen otherwise to prevent staff members from deleting each others’ requests by accident.
    - The only exception is Kristian, who can cancel any request at will.
* When the ‘cancel request’ button is clicked:
  + Remove the request from TblRequests by using the MySQL command (“DELETE FROM TblRequests WHERE RequestID = request\_id”)
    - Where request\_id stands for the RequestID of the request being viewed.

‘Accept/decline request’ buttons:

* If the UserID in the session variable belongs to a driver, and the request needs a driver (ie DriverID is empty):
  + The ‘accept request’ and ‘decline request’ buttons will be added to the page.
* When the ‘accept request’ button is clicked:
  + The system does a final check to see if the request still needs a driver. If not, an error message will be printed. If so:
    - The request currently being viewed is updated such that the driver who clicked accept is assigned to it.
    - I will do this with the following MySQL command: (“UPDATE TblRequests SET DriverID = session\_var\_id WHERE RequestID = request\_id;”)
      * Where session\_var\_id is the UserID from the session variable, and request\_id is the RequestID of the request currently being viewed (read from $POST).
    - The requestor is automatically notified that the request has been accepted.
* When the ‘decline request’ button is clicked:
  + TblDeclinedDrivers is updated with the pair of keys of the DriverID of the currently logged in driver, and the RequestID of the request currently being viewed.
    - I will use the MySQL command (“INSERT INTO TblDeclinedDrivers (DriverID, RequestID) VALUES (session\_var\_id, request\_id)”).
    - Where session\_var\_id is the UserID from the session variable, and request\_id is the RequestID of the request currently being viewed (read from $POST).
  + The system will then check whether every driver has declined the request. If there are any drivers that haven’t declined, then no action is taken, but if there aren’t any:
    - The request is deleted from TblRequests, all records in TblDeclinedDrivers with this RequestID are deleted, and both the requestor and Kristian will be automatically notified (the notification will include the details of the request for reference).
    - I will use several MySQL statements for this.
    - For checking the request: (“SELECT DriverID FROM TblDeclinedDrivers WHERE RequestID = request\_id”).
    - For deleting from TblRequests: (“DELETE FROM TblRequests WHERE RequestID = request\_id”).
    - For deleting from TblDeclinedDrivers: (“DELETE FROM TblDeclinedDrivers WHERE RequestID = request\_id”).

#### Pending requests page

##### Page overview

On this page drivers will be able to view all pending requests made by staff members so they can choose which ones to accept. Drivers may click on requests to view them in full detail. This will redirect to the request detail page described above.

##### Page layout

Pending jobs:

**08/05/2025**

*13:30 – 17:00 Rowing regatta*

**10/04/2025**

*14:00 – 16:00 Fives away fixture*

**12/03/2025**

*09:00 – 15:00 Quadrivium law trip*

Your accepted jobs

**View pending jobs**



**View pending jobs**

Ronald Newton Log out

##### Data structures & variables

This page will read the data for the list of requests from TblRequests, using the UserID of the currently logged-in user (from the session variable) to ensure that only their requests are shown.

This page needs to use the session variable, and will also need to use variables to split the DateTimeOut variable read from TblRequests into a date and a time component for displaying as in the above page layout.

##### Validation

Since this page never writes any data, no validation is required.

##### Algorithms

Loading in the navbar:

* If no user logged in:
  + Redirect to login page.
* If user logged in:
  + Get role values from TblUsers:
    - If IsRequestor is True: show requestor pages on navbar.
    - If IsDriver is True: show driver pages on navbar.
    - If IsAdmin is True: show admin pages on navbar.

Loading user requests:

* If no user logged in, or if user is not a driver:
  + Redirect to login page.
* If driver logged in:
  + Get all requests made by that driver from TblRequests.
  + I will use the MySQL command (“SELECT DateTimeOut, TimeIn, Purpose FROM TblRequests WHERE UserID = current\_user\_id”).
    - Where current\_user\_id stands for the UserID of the user that is currently logged in.

Clicking on a particular request:

* This will not be a link using an <a> tag, but will instead be a button that redirects to the request details page and posts the RequestID of the request that was clicked on, so that the details of that request can be shown on that page.

#### Accepted jobs page

##### Page overview

On this page drivers will be able to view the jobs they have accepted, and will be able to click on individual jobs to view them in full detail.

##### Page layout

Accepted jobs:

**19/04/2025**

*12:00 – 17:00 Cricket away fixture*

**07/02/2025**

*18:00 – 22:00 House cinema trip*

**Your accepted jobs**

View pending jobs



**View pending jobs**

Ronald Newton Log out

##### Data structures & variables

This page will read the data for the list of requests from TblRequests, using the UserID of the currently logged-in user (from the session variable) to ensure that only their requests are shown.

This page needs to use the session variable, and will also need to use variables to split the DateTimeOut variable read from TblRequests into a date and a time component for displaying as in the above page layout.

##### Validation

Since this page never writes any data, no validation is required.

##### Algorithms

Loading in the navbar:

* If no user logged in:
  + Redirect to login page.
* If user logged in:
  + Get role values from TblUsers:
    - If IsRequestor is True: show requestor pages on navbar.
    - If IsDriver is True: show driver pages on navbar.
    - If IsAdmin is True: show admin pages on navbar.

Loading user requests:

* If no user logged in, or if user is not a driver:
  + Redirect to login page.
* If driver logged in:
  + Get all requests made by that driver from TblRequests.
  + I will use the MySQL command (“SELECT DateTimeOut, TimeIn, Purpose FROM TblRequests WHERE UserID = current\_user\_id”).
    - Where current\_user\_id stands for the UserID of the user that is currently logged in.

Clicking on a particular request:

* This will not be a link using an <a> tag, but will instead be a button that redirects to the request details page and posts the RequestID of the request that was clicked on, so that the details of that request can be shown on that page.

# Development

For the beginning of my project, it makes sense to begin by creating the Database and the corresponding Tables.

A screenshot of a computer

Description automatically generated

Now that the Database is created, I will create the Tables using the following SQL code:

## Table SQL Code:

### TblUsers

CREATE TABLE TblUsers (  
UserID INT(4) UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
Password VARCHAR(60) NOT NULL,  
Email VARCHAR(50) NOT NULL,  
TelephoneNumber Varchar(11) NOT NULL,  
Forename VARCHAR(30) NOT NULL,  
Surname VARCHAR(30) NOT NULL,  
IsDriver TINYINT(1) NOT NULL,  
IsAdmin TINYINT(1) NOT NULL,  
HoursWorked INT(3) UNSIGNED);

### TblVehicles

CREATE TABLE TblVehicles (  
VehicleID INT(4) UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
RegNumber VARCHAR(7) NOT NULL,  
Capacity INT(2) UNSIGNED NOT NULL,  
NotAvailableFrom DATE);

### TblRequests

CREATE TABLE TblRequests (  
RequestID INT(4) UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
DateTimeOut DATETIME NOT NULL,  
TimeIn TIMESTAMP NOT NULL,  
Destination VARCHAR(20) NOT NULL,  
Postcode VARCHAR(8) NOT NULL,  
Purpose VARCHAR(20) NOT NULL,  
ReqCapacity INT(2) NOT NULL,  
DriverID INT(6),  
VehicleID INT(6),  
RequestorID INT(6));

Once this SQL code has been run, the terminal looks like this:

A screen shot of a computer program

Description automatically generated

## Connection Page

Before I go any further, I will need a page to connect to the database so that the website can access the database.



I wrote this PHP code to connect the website to the database. This code can be included in every other page in a modular way, so any maintenance or bug fixes can happen once on one page. Additionally, the important variables are declared above the main code so that they can be edited easily.

$servername is set to localhost so I can test the website and database with xampp; it will be reset later during implementation.

I tested this code and it produced the following error:



I think this means that the database name is incorrect. This is because I accidentally set it to be the server path instead of the MySQL database name on xampp. I’ll change $dbname to be “transport\_coursework”.

A black text on a white background

Description automatically generatedHaving made this change, the connection page now works. This is the success report and final working code:



Since this page is now fully functioning, I will move on.

## Install Page

Instead of inputting all tables into Xampp manually, I will make an install page that automatically drops all current tables then creates new tables with the correct details. I will not yet put test data into this install page as it will be easier and more manageable to have blank tables for the purposes of testing.

I wrote this PHP code (also using MySQL) to drop each table and recreate it with the appropriate data fields.



This code ran without producing any error messages on the web page, and created all tables correctly. Since it seems to be functioning perfectly well, I’ll move on to the input pages.

## Vehicle pages

Each table requires a page to input records. However, each of these pages requires an additional backend page to actually edit the tables on the database. Simply put, adding to any table requires one frontend page and one backend page.

I will name these pages with a consistent system. The frontend pages will be named after the table they are connected to (in lowercase, with “Tbl” removed. For example: “vehicles.php”), and the backend pages will have ‘\_add’ appended to the end of this name. This is for the sake of easy maintenance and intuitive readability.

This is the first draft of code I created for vehicles.php (the frontend page for adding vehicles to the database).



This is the first draft of code I created for vehicles\_add.php (the backend page for adding vehicles to the database).



When this code is run, The following is output to the website:



### Validation

“Reg number” and “NotAvailableFrom” both need validation. The validation for Reg number is much easier to do on the backend page, as well as some of the validation for NotAvailableFrom. However, since dates can be tricky to enter and typos or mistakes in formatting can be both frequent and destructive, I will change this input to be a calendar dropdown. This will completely prevent formatting errors and will reduce mistaken entries.

This is the current draft of code and corresponding output for vehicles.php.

A computer code on a black background

Description automatically generated



Now I must implement the validation on the backend page vehicles\_add.php.

Registration numbers follow a certain pattern. They are all seven characters long. The first two characters are always letters (indicating the region of origin of the vehicle), the next two characters are always numbers (indicating the age of the vehicle), and the final three characters are always random letters for the sake of uniqueness. I will not validate whether the initial letters correspond to an actual area code, nor will I check whether the numbers correspond to a reasonable date range, since this would be an unnecessary level of detail for this project.

The only validation required for NotAvailableFrom is that it is in the future.

This is the first draft of code to check both of these validations.

A screen shot of a computer program

Description automatically generated

I will first test by inputting valid information:

A screenshot of a computer

Description automatically generated

This is the output on the vehicles\_add.php page.

A close up of a number

Description automatically generated

This is the output on vehicles.php afterwards.

A screenshot of a computer

Description automatically generated

The code falsely flagged both RegNumber and NotAvailableFrom as invalid. However, the code did not add the ‘invalid’ data to the database, which is good.

I modified part of the code to print more information in order to solve this problem.

A screenshot of a computer program

Description automatically generated

Output to vehicles\_add.php.

A white background with black text

Description automatically generated

I figured out that the bracket on line 24 was in the wrong place, so the string RegNumber itself was being compared to 7, not its length.

I have fixed the bug which caused RegNumber to be falsely invalidated. I changed line 24 – old and new versions shown below.





Since that bug is fixed, I removed the extra echo statements for debugging. However, NotAvailableFrom is still being falsely flagged as invalid, as shown here:

A close up of a number

Description automatically generated

In order to fix this I once again modified part of the code to output more information.

A computer screen shot of text

Description automatically generated

This caused the following error:



This error is being thrown because the DateTime variable $current\_date cannot be converted into a string to be printed. Since line 12 of the code has been running all along without producing this error, it must mean that $\_POST[“NotAvailableFrom”] is not actually a DateTime variable and so the comparison on line 36 would not work.

I edited line 40 of the code so that it would print out the type of the variable $\_POST[“NotAvailableFrom”].

A computer screen with colorful text

Description automatically generated

As you can see in this output on vehicles\_add.php, its type is string.

A close up of a number

Description automatically generated

I added this line farther up the code, which creates a variable $NotAvailableFrom\_converted, which is $\_POST[“NotAvailableFrom”] converted into a DateTime object.

A screen shot of a computer

Description automatically generated

This modification produces this output on vehicles\_add.php.

A close up of numbers

Description automatically generated

And on vehicles.php I can see that the entry has been added to the table:

A screenshot of a computer

Description automatically generated

Now I’ve established that the code will no longer falsely label data as invalid, I will test whether it falsely labels data as valid. These tests are shown below. If the entry is added to the table, I will show the output on vehicles.php; otherwise I will show vehicles\_add.php.

A black text on a white background

Description automatically generated

A screenshot of a computer

Description automatically generated

A black text on a white background

Description automatically generated

A close up of a number

Description automatically generated

For this test, I entered ‘-1’ into the capacity box.

A screenshot of a computer

Description automatically generated

For this test I entered ‘sixteen’ into the capacity box.

A screenshot of a computer

Description automatically generated

A close up of a number

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A close up of a number

Description automatically generated

For this test I entered '16.66’ into the capacity box.

A screenshot of a computer

Description automatically generated

For this test I entered ‘9+7' into the capacity box.

A screenshot of a computer

Description automatically generated

For this test I entered ‘apple’ into the capacity box.

A screenshot of a computer

Description automatically generated

For this test I entered ‘1sixteen6’ into the capacity box.

A screenshot of a computer

Description automatically generated

For this test I entered ‘sixteen16’ into the capacity box.

A screenshot of a computer

Description automatically generated

Conclusions from these tests:

* RegNumber
  + Length will always be seven characters.
  + First two characters and last three characters will always be letters.
  + Letters may be lowercase, which should be corrected.
  + The third and fourth characters may be non-number numeric characters, such as ‘.’ and ‘-’. This should be corrected.
* NotAvailableFrom
  + Validation working correctly.
* Capacity
  + Negative values are stored as 0.
  + Decimal values round to the nearest integer.
  + Any entry with letters in it is only read until the letters start, after which point everything is ignored. If the first character is a letter, then 0 is stored.
    - E.g. ‘sixteen16’ is stored as 0.
    - ‘9+7’ is stored as 9.
    - ‘1sixteen6’ is stored as 1.

I think the best way to make the validation for capacity more consistent is to change the input box so that it only accepts positive numbers.

I can do this quite simply by adding type=“number” and min=“1” to the <input> tag for capacity on line 10 in vehicles.php. Below are the old and modified versions of the form code in vehicles.php.

Old version:

A computer code with text

Description automatically generated with medium confidence

Modified version:

A screen shot of a computer code

Description automatically generated

This simple change will prevent all confusion with strings. I will now conduct more tests to see if further validation is required. I ran install.php to clear the table for the sake of readability.

A computer screen shot of a computer

Description automatically generated

A close-up of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

Although it cannot be shown in screenshots, the textbox will not allow letters to be typed into it. These tests show that all validation is now working for Capacity.

Now the validation for RegNumber needs to be fixed. The problem is that decimal numbers and negative numbers like ‘.1’ and ‘-9’ are being allowed since is\_numeric(‘.1’) returns True. However the symbol parts alone will return False, so this issue can be fixed by looking at both characters individually. This is the new draft of validation code for RegNumber.

A computer screen shot of a code

Description automatically generated

I will now put this through the same tests that the old code failed.

A close up of a number

Description automatically generated

A close up of a number

Description automatically generated

Clearly, the new validation works, so I will move on.