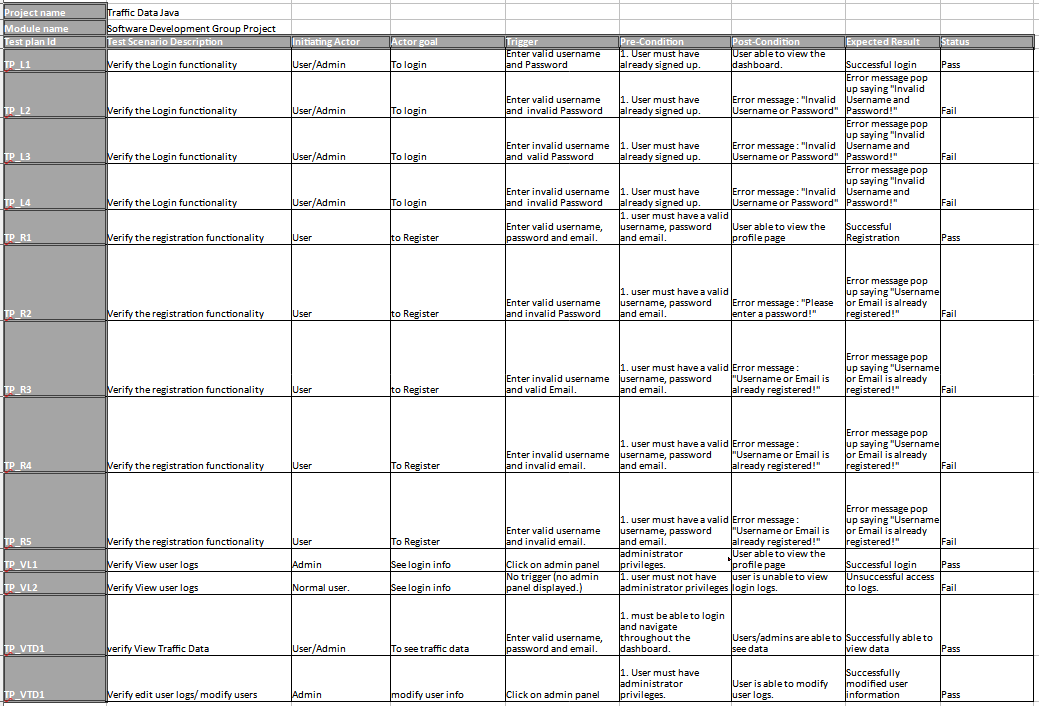
|  |  |  |
| --- | --- | --- |
| 5COSC003W\_cwk1\_group\_Ciaran\_Lyne\_w1725430  * Use this template to structure the GROUP element of coursework 1. Ensure that the correct information is in each white box. The advice for each box is basic guidance to help you focus your answer. * YOU MUST USE THIS TEMPLATE FOR THE GROUP WORK OF COURSEWORK 1.  The current size of the boxes is not indicating how much you should write; change their size as you need but ensure that you make complete points, concisely, without repetition or going off topic.When you save the file, put your name and registration number in the file name, eg ‘5COSC003W\_cwk1\_group\_Kelly\_Garret\_12345678.doc’.  * A reminder of plagiarism: If you use bits of another’s group report in yours or if you give your report to another group to use this will be the academic offence called ‘collusion’. * In each section you are asked to write who worked on this. **EVERYONE is expected to work in ALL sections.** If someone didn’t do work in one section, do not say that they did. Rewarding those who didn’t do any work is not good for you and is not good for them. | | |
| **Surname** | Lyne | |
| **Forename** | Ciaran | |
| **Registration No:** | W1725430 | |
| **By submitting this coursework you agree to the following:** | | |
| I confirm that I understand what plagiarism is and have read and understood the section on Assessment Offences in the Essential Information for Students. The work that I have submitted is entirely my own. Any work from other authors is duly referenced and acknowledged. | | I confirm |
| **List here the other members of your group** | Aykut Inalan, Steven Naaba, Tevin Awuakye, Burak Kavus | |

|  |  |
| --- | --- |
| Conceptual and Logical ERD (10 marks) | |
| **Which group members worked on this:** | Ciaran Lyne, Burak Kavus |
| **Guidance:** Put here the ERD model of the data you are using. This should include the specific data set you have for your group and should include all the attributes in your dataset. | |
| **CONCEPTUAL ERD:**    The conceptual ERD can be broken down into 3 sections: user system, log system and traffic data. The user tables attributes store data on the user and have a specialisation on optional for users that can access admin functionality. The main log table (ActionLog) has is specialised into 3 different types of log activity: login attempts, logout and edit user. Any given ActionLog will have a corresponding user who performed the action thus the one to many relationship. A one to many relationship is also found between ActionEditUser and user, where the modified user will be referenced from the ActionEditUser table. Below these, we see the data structures for traffic data. As the csv provided contained many lines of repeated data from the same count points, it seemed logical to split the tables and instead of bringing all the information in from the CSV to create a count points table to store the locations of counts then create a table to store count data. The table between them (Count) will splits any given count point up into the counts per hour in a day and links the two tables together with two more one to many relationships. A simple way of viewing this would be a count happens at a count point and records a new count data entry with every hour.  **LOGICAL ERD:**    Conversions from conceptual. countID from count data already existed in the composite key but was set as a foreign key also due to the translated one to many relationship. After multiple conceptual ERD drafts and conversions to logical, this was our final representation of the database and how we plan to implement it into SQL. | |

|  |  |
| --- | --- |
| User Interaction Design (10 marks) | |
| **Which group members worked on this:** | Steven Naaba |
| **Guidance:** Include here the wireframe and mock up of your interface design | |
| **Wireframe:**                  Initially, an issue arose with the wireframe due to a misunderstanding of the tutorials and was made as a mobile app. Luckily, this was early on and was identified and sorted quickly. Looking at the general structure of the wireframes will show a clear inspiration from mobile design and it seems this small error at the start helped a lot in the ease of use of the designs. As they were initially made for a platform with a smaller screen, there is a distinct lack of excess information giving the elements that are present more room to breathe and increasing the readability of the page.  **MOCK-UP:**                  Moving from the wireframe to the mock-up, the most noticeable feature is the choice of colour pallet. Given reading data can take a while, it’s expected the user may spend a good amount of time looking at the application, so it makes sense to ensure readability, usability and decrease the intensity on the eyes.  The choice of the darker and lighter blue used for backgrounds and buttons while always ensuring when the harsher blue is used, it’s always contrasted using white. Colours are most important  Combining this with Java swing allowing for rounded edges on containers allows us to create a certain softness to the application. Trying to increase how easy the page is on the eyes by removing 90-degree corners has worked quite well.  Combining this with the colour scheme and the spacious nature of the design has set us in good stead for the upcoming challenge of creating the program. It is a good balance between simplicity for the user and still allows us to look forward to task of transferring this into code.  **Document continues on next page** | |

|  |  |
| --- | --- |
| UML diagrams (40 marks) | |
| **Which group members worked on this:** | Ciaran Lyne, Aykut Inalan, Tevin Awuakye |
| **Guidance:** Put here your use case diagrams, sequence diagrams and class diagrams | |
| **USE CASE DIAGRAM:**  Much like the ERDs, the use case diagram is split up into three sections: human actors, traffic software interaction, and system actors. These titles are created to make it easier upon the viewer to understand what the system will be doing. The actors are colour coded to allow easier way to follow along what each actor is doing. we have made sure the interaction between all the data and the actor makes sense we had 5 drafts all-together before ending up with this version. As a group, we have used all the tools appropriate to create a complex system that pulls from many aspects of what we’ve learnt and, as a starting point, the use case diagram forced us to view from system from a top down perspective. This in turn allowed for better fluidity between documents as we had a good reference to look back on that gave a good overview of how our designed systems interacted as we created and improved upon them.    Use case specification has attributes from both sequence diagram and the test plan. The above table is used to show the simple relationships that happens during the user’s interaction with the software. All the table is intended to do is display all the important data and furthermore, disregards any data that may not be necessary to display. We can use the use case table to gain a better understanding of what’s happening inside the use case diagram.  **SEQUENCE DIAGRAM:**    The sequence diagram displayed here is a representation of the processes that occur during our program. We made sure this was easy to follow by using the arrows appropriately wherever they were needed. As you see login and register are at the top, for login to work you must first have an account so your email and password could be rejected.  For register to work you must save account information that has not previously existed which will cross reference with the database server to allow verification.  We also want administrators to be able to give administrator of other users that are displayed in our admin panel, so we allow administrators to edit users that are registered in our application. When a user tried to access traffic data before we display, we must verify the existence of the data if it is successful then we can let the user the ability to see the data.  **CLASS DIAGRAM:**    **(Sorry for small text size, this was the only way we could get it to fit. It’s high resolution so zooming should work.)**  Most classes here are self-explanatory. In order to save space, instead of putting getters and setters as methods for all the classes, the notation <get> and <get/set> is used to denote attributes that will have these methods.  The data structures for traffic counts have remained fairly like those from the logical ERD. Theoretically, this will allow us to make the process of creating the dashboards simpler for those less familiar with SQL and object oriented.  Another aspect of our system design used in this way is how GUI classes are handled. By trying to establish all the classes required for the GUI early on before any code is even touched, we can abstract the problem class by class. Creating a panel model that all the different panels of the application can fit into while displaying the navbar next to it will allow us to have more control over how these are displayed while compartmentalising in a way that should prevent an error in one class messing up an unrelated one. An issue with displaying a component in Dashboard1 has no way of spreading to the navbar for instance.  This is also reminiscent of an MVC (model view controller) framework. The idea being that the user interacts with the controller, which then manipulates the model in someway which in turn updates the view for the user and the cycle repeats.  While the attempt here to do something similar isn’t quite a full realisation of MVC, the inspiration from this way of doing things can be seen across all documents but is highlighted best here in the class diagram. Another example is how the individual dashboards will control the display while the dashboard model is left to interact with the traffic data.  **Document continues on next page** | |

|  |  |
| --- | --- |
| Use case test plans (10 marks) | |
| **Group members** | Aykut Inalan |



The test plans look complicated on first look but it is not, first thing you will notice is similar scenarios have similar ID’s to show that those relate to each other. The test plan is used to assess how the application should interact with the users that use the application.

The first four row are on the login functionality as you see there the test plan indicates the importance of entering the correct password. We also specify that the user must already have an account for this functionality for it to work properly as shown on the test plan.

While reading this test plan you might notice the status, this allows you to see if the user has succeeded in their task. For example, if a non-administrator user tries to access an administrator section of the application it will be a failure because they will be unable to do so.