**Section 1: Analysis of the design problem**

Reading the description of the design problem the following functional requirements, quality requirements and design constraints have been established:

|  |  |  |
| --- | --- | --- |
| **ID** | **Functional requirements:** | **Priority** |
| F01 | The system shall automatically lock pump from use after transaction | Essential |
| F02 | The staff shall be able to unlock pump for use after payment of previous transaction. | Essential |
| F03 | Spare |  |
| F04 | The system shall detect the type of fuel being taken | Essential |
| F05 | The system shall be able to detect pump number | Essential |
| F06 | The system shall automatically measure volume of fuel drawn from pump. | Essential |
| F07 | The system shall  provide appropriate display for users to read the volume of fuel drawn | Essential |
| F08 | The system shall be able to store fuel values per litre | Essential |
| F09 | The system shall be able update new fuel values per litre | Essential |
| F10 | The system shall automatically calculate total fuel cost per transaction | Essential |
| F11 | The system shall provide appropriate method  for users to read amount to pay | Essential |
| F12 | The system shall enable user login | Essential |
| F13 | The system shall be able to automatically clear previous  user transaction on the pump | Non-Essential |
| F14 | The system shall store all transactions | Essential |
| F15 | The system shall be able to check any transaction status | Preferred |
| F16 | The system shall be able to display any transaction status | Preferred |
| F17 | The system shall read keypad inputs | Essential |
| F18 | Check out staff shall be able to Modify transactions | Essential |
| F19 | The system shall be able to process bank/card transactions as well as cash transactions | Essential |
| F20 | The system shall print receipt on local printer | Essential |
| F21 | They system shall be able to handle multiple transaction simultaneously | Essential |
| F22 | The system shall be able to archive completed transactions | Essential |

|  |  |  |
| --- | --- | --- |
| **ID** | **QUALITY REQUIREMENT** | **Priority** |
| Q01 | System will be able to send transaction from pump to database in < 5 seconds. | Essential |
| Q02 | The system shall be able to calculate amount to pay < 1 second including display. | Essential |
| Q03 | System shall be able to print receipt < 2 seconds once transaction is complete. | Essential |
| Q04 | System shall provide appropriate user clearance level | Essential |
| Q05 | System shall be compatible with a range of hardware configurations | Prefered |
| Q06 | The System  shall send transactions details to server accurately | Essential |
| Q07 | The system shall process transactions securely | Essential |
| Q08 | the system shall handle personal details securely | Essential |
| Q09 | The system shall be secure |  |
| Q10 | The system shall not store any customer bank credentials | Essential |
| Q11 | The system should be aesthetically pleasing to all users | Non-Essential |
| Q12 | The system should be user friendly/Intuitive interface | Essential |

|  |  |  |
| --- | --- | --- |
| **ID** | **Design Constraints** | **Priority** |
| C01 | Communication protocol over the site must be able to transmit over the given distance, ie, between pumps and server. (internal) |  |
| C02 | The user interface must be easy to use (Internal) |  |
| C03 | The system must have a live connection to the internet for accounting purposes. (External) |  |
| C04 | The connection to the internet must be secure. (Internal) |  |
| C05 | The login must be secure. (internal) |  |
| C06 | The database for the transactions should be done in SQL |  |
| C07 | The database must be on a single server. |  |
| C08 | The distance of the pump and the tills should be at least 10 meters. |  |

With the above characterisation of the design problem we can now proceed with the designs in 5 different styles. We have chosen the following styles and allocated them to the following group members:

Data Centered Blackboard Architecture - Lee hudson

Pipe and filter - Terry Mukonka

Object Orientated - Ryan Varley

Implicit Invocation - Shaz Ashraf

Client Server - Tanaka Chimbuya

**Section 2: Designs**

**Data Centered “Blackboard” Architectural Style**

**Rationale:**

I have chosen to use the data centered blackboard style. This will consist of a central data repository and other software components monitoring and communicating through it. The nature of computation of the desired system and the nature of computation of the blackboard system are compatible for the following reasons:

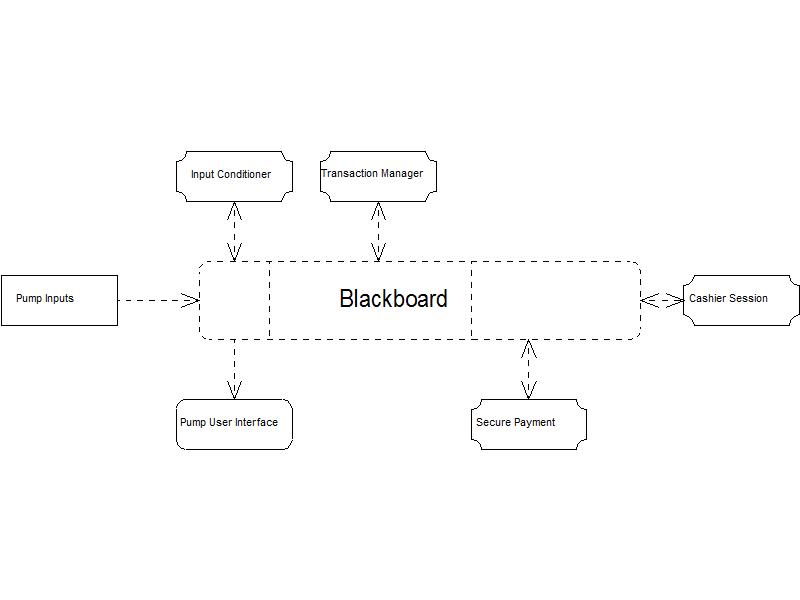
* A data centered architecture has a heavy emphasis on data storage. In order for the system to function the data must be well structured. This is useful for the proposed system as a record of transactions could be kept in the central data structure.

Using the data centered black board architecture also has some quality merits that would prove useful in the proposed system.

* The black board architecture makes use of implicit invocation. This allows a simplistic and modular approach to the design. If a component requests another component to perform a task it will simply ask for “someone” to perform that task via the data repository / blackboard. This means that a component can be replaced as long as it responds to these implicit requests correctly.
* As the data is all in one place it would be easy to implement another component that does some analysis on the data for marketing / finance purposes.
* Data centered black board architecture allows for easy up-scaling. The amount of processes interacting with the data can be easily increased. In the context of the proposed system this means additional pumps or cashiers can be added with little effort.

**Component and connector description:**

1. **Pump Input**
   1. These are the raw inputs coming from the real world. These include the pump pick up sensors to detect if a pump has been picked up and which one, and the fuel flow sensors.
2. **Input conditioner**
   1. This is an active software component that looks at the raw information that has been placed in the data repository by the input sensors and converts the inputs into something meaningful. It will provide the following functions:
      1. It will take the raw flow sensor input (12bit float representing flow rate), convert it into liters per second and be responsible for performing the integration required to get the volume of fuel taken / being taken in real time.
      2. Calculating the cost of the fuel being taken using the volume as calculated from the flow sensors, the type of fuel being taken inferred by the nozzle pick up sensor and the price per liter for a given fuel as stored in the data repository.
3. **Pump user interface**
   1. This module will take the conditioned inputs from the data repository (as conditioned by the input conditioner module) and provide a user display at the pump. This is a passive component and will be triggered by the input conditioner module (nozzle pickup). It will display the current amount of fuel taken and the current price for that volume of fuel.
4. **Transaction Manager**
   1. This is an active software component. It will perform the following functions:
      1. It will continuously monitor the data being accessed and modified by the input conditioner module. Once it detects a volume of fuel has been taken and the nozzle has been put back in the holder it will generate a transaction record in the data repository. This will contain the pump ID, the type of fuel taken, the amount taken and the price of the transaction.
5. **Cashier session**
   1. This is an active software component that becomes active once a cashier has logged in, once the cashier has logged in this will stay active until the cashier has logged out. It will provide the following functionality:
      1. If a transaction is not currently in the control of another instance of the cashier session it can take control of that transaction in the data repository created by the transaction manager. It will then mark this transaction record as “locked” to stop concurrent access. It will do this by setting a bit in a status word contained within the transaction record.
      2. Request a payment to the “secure payment” module via the data repository. In practical terms this will set a bit in a status word in the transaction record that the “Secure Payment” module will be looking for.
      3. Mark the transaction as being completed or void. Again, this will be done by setting some bits in a status word in the transaction record and depends on the outcome of the operations performed by the secure payment module.
6. **Secure Payment**
   1. This is an active software component that is continuously monitoring the transaction records status word. When it detects a transaction that needs to be paid it will carry out the secure payment. Once completed it will set the status word to “paid” or “void” so the cashier session can complete the transaction and unlock the record.



# DATAFLOW (PIPE & FILTER) ARCHITECTURAL STYLE

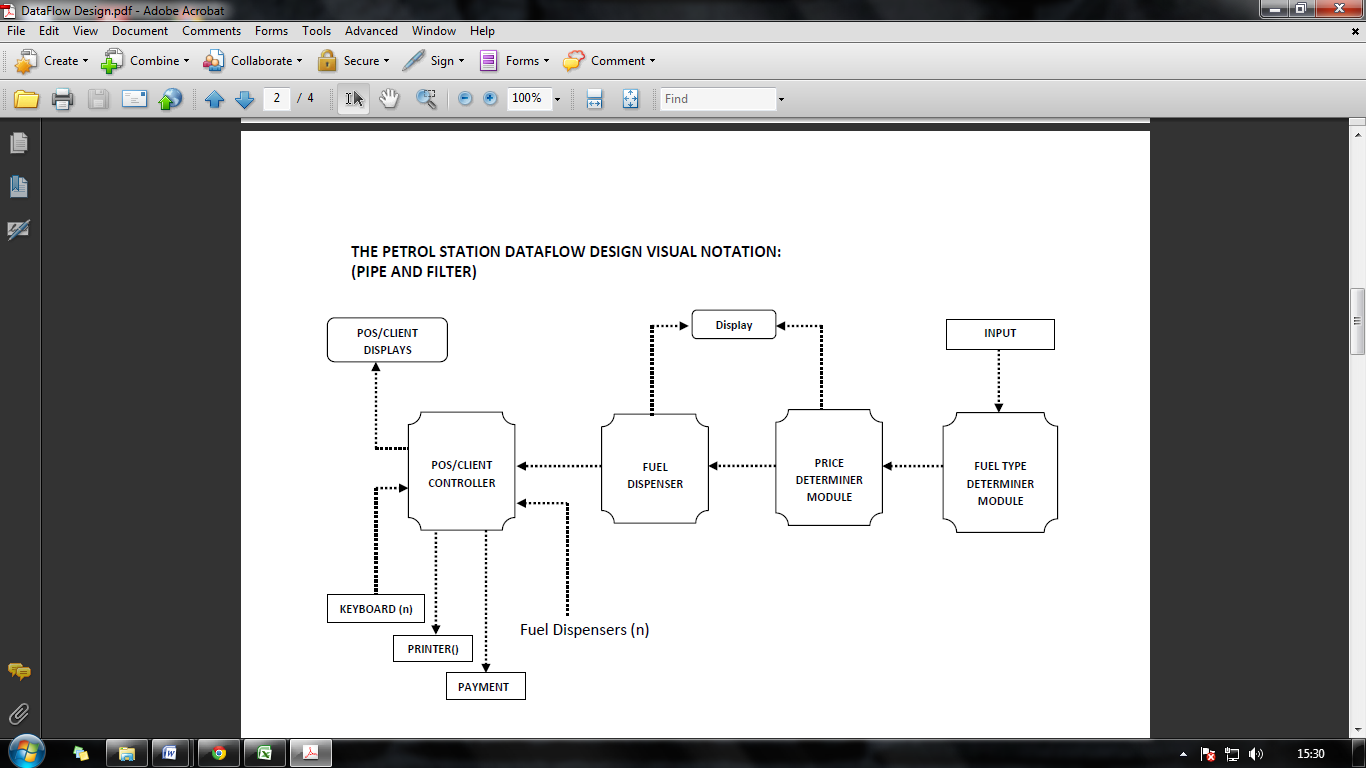
**RATIONALE:**

The use of the Pipe and Filter architectural pattern provides a structure of the data flow within the petrol station. The pipe and filter style shows components that process a stream of data (filters) and connections that transmit data between adjacent components (pipes). This architecture style was chosen as it provides reusability, maintainability, and decoupling of system processes by having distinctive, identifiable and independent tasks that can be rearranged.

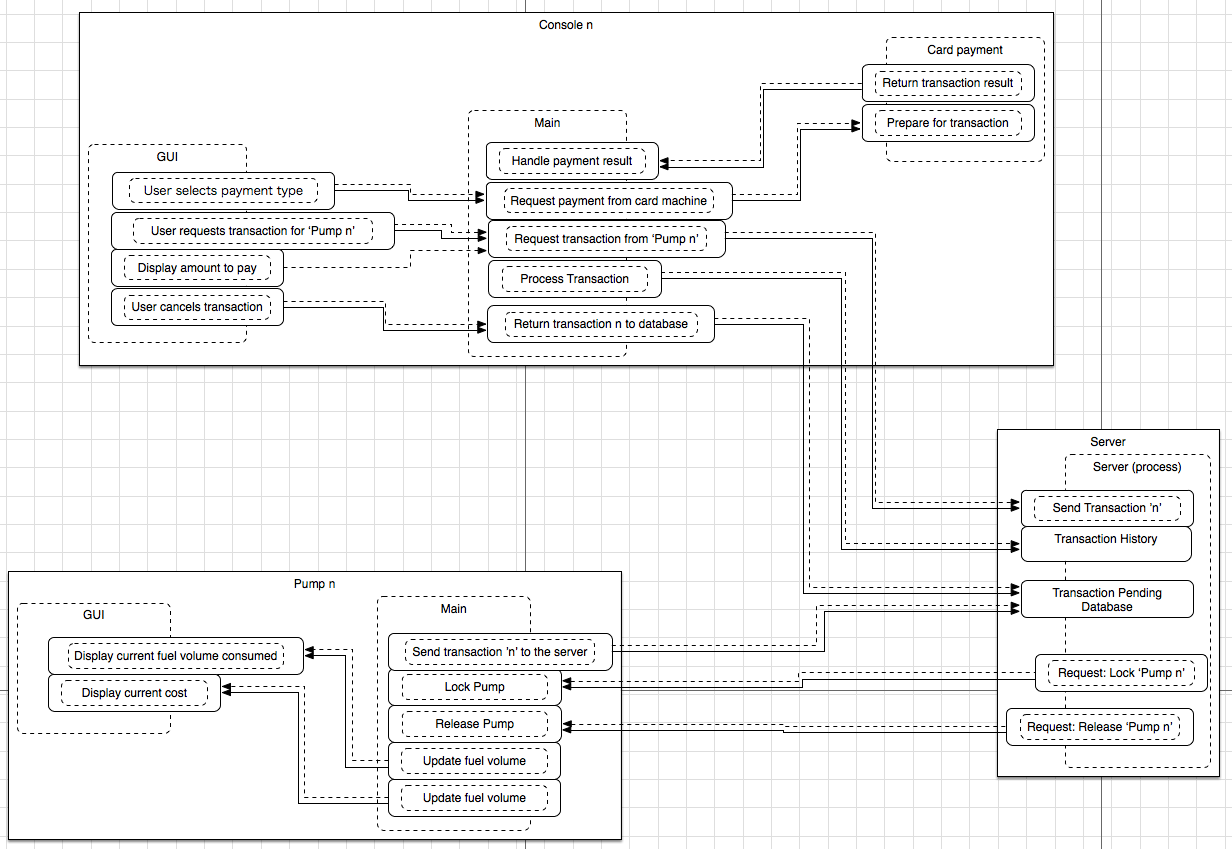
The pipe and filter style works to this system, as the order in which filters are processed is determined and sequential in nature. It applies to the problem where it is natural to decompose the computation into a collection of semi-independent tasks such as the task carried out at the pump and through to the POS controller.

**COMPONENT AND CONNECTOR DESCRIPTION:**

1. **Input**
   1. This is the action from the nozzle; actions include picked up, placed back and pulling of the trigger to dispense the additive.
2. **Fuel Type Determiner Module**
   1. The fuel determiner module detects the nozzle picked and determines the fuel type that is used.
3. **Price Determiner Module**
   1. The price determiner sets the unit price for type of fuel selected.
   2. It also sends data to the display (The display resets and shows the unit price).
4. **Fuel Dispenser**
   1. With the price determined and the fuel type determined, the fuel dispenser starts to dispense fuel when further input data is received.
   2. The data on the fuel withdrawn passed onto the display.
   3. The Fuel dispenser holds the transaction until it is complete (by the nozzle placed down) and sends it to the POS Controller.
5. **POS/Client Controller**
   1. The POS/Client controller stores the transaction for the each pump on the forecourt.
6. **Keyboard (Input)**
   1. Input from the cashier selects the pump and the transaction from the POS/Client controller.
7. **POS/Client Display**
   1. The POS/Client display shows the transaction details that are pulled from the POS/Client controller.
8. **Payments**
   1. When the payment is received processed the transaction data is sent to the POS controller.
   2. In addition, a prompt to issue a receipt is ordered.
9. **Printer**
   1. The printer prints the transaction.

****

**Object Oriented Architectural Style**



**Rationale:**

I have chosen to use the ‘Object Oriented’ design style. This will consist of a main class with which all other classes, in this case the pump and console classes, interact.

The nature of computation of the desired system and the nature of computation of the object oriented architecture are compatible for the following reasons:

* Due to its data abstraction based methodology, the object oriented approach to software design lends itself to the implementation of object oriented programming in the sense that the design will be closely related to the inherent structure of an object oriented programming language.
* It is relatively easy to design a modular system when utilising object orientation.
  + Modular system design is essential for **//TODO** as it allows for individual components to be tested, debugged, upgraded or replaced with minimal risk of negatively impacting or even damaging the rest of the system.
* The various classes do not simply represent one class, but multiple instances of that class, allowing for a further development and hardware diversity in the future.

**Component and connector description:**

1. **Server**
   1. Release Pump
      1. Once the server has started, the first action would be to release all of the pumps for use. It would achieve this by sending a command to ‘2.1 Release Pump’.
   2. Lock Pump
      1. Adversely, should a pump require locking, i.e. as the server shuts down/crashes, the server would send a lock request to ‘2.2 Lock Pump’.
   3. Send Transaction
      1. Upon request from ‘3.1 Request Transaction’, the server will send transaction data from a specific pump (the data to which will be stored in ‘1.4 Fuel Taken / Cost of Fuel’) to a specific console.
   4. Fuel Taken / Cost of Fuel
      1. The data store for pump specific, unclaimed transactions. This component would receive data from ‘2.3 Send Data’. Once the transaction is complete (i.e. the Transaction ID matches that of a Transaction ID in ‘1.5 Transaction History’, it will remove the previously ‘unclaimed transaction’ from ‘1.4 Fuel Taken / Cost of Fuel’.
   5. Transaction History
      1. The data store for completed transactions. This component would receive data from ‘3.2 Process Transactions’.
2. **Pump**
   1. Release Pump
      1. Once this component receives the command from ‘1.1 Release Pump’, it will interface with the hardware with the hardware to release / unlock the pump.
   2. Lock Pump
      1. Once this component receives the command from ‘1.2 Lock Pump’, it will interface with the hardware with the hardware to lock the pump.
   3. Send Data
      1. Once the hardware signals that it’s transaction is ready, it will send the transaction data to ‘1.4 Fuel Taken / Cost of Fuel’.
3. **Console**
   1. Request Transaction
      1. Upon user interaction the console will request a transaction from a specific pump via ‘1.3.
   2. Process Transaction
      1. Once the transaction has been completed, the transaction data would be sent to ‘1.5 Transaction History’.

**Individual Component Implicit Invocation Architectural Style**

**Rationale:**

The chosen architectural style I am going to implement is an independent component event based system using implicit invocation. The justification for the chosen style is because one component can continue to make progress independent of the states of other components. The system is to be run on a network of computer systems consisting of multiple consoles, pumps and a single server.

The way in which this architectural style works is that individual components announce data that they wish to share with their environment as a set of unnamed components and can be invoked into a class of data. The structure makes use of event handlers to manage the communication amongst components.

Using the independent component event based system there are a number of quality aspects that need to be taken into consideration. There are two main factors that need to be analysed:

* Modifiability, especially when performance tuning via reallocating work among processes and processes to computers is necessary.
* Performance, especially to achieve maximal utilisation of the computational resource is important.

The interface of a component in this style usually contains multiple events that can assess and activate the use of procedures and functions that other components are dependent upon. This enforces the quality aspects that need to be taken into consideration above.

**Component and connector description:**

1. Master Control: This module will take the conditional inputs from the pump and the console based on hardware assumptions that the pump details are entered into the system upon the nozzle being lifted. A further assumption can be made from the console that upon a key press a confirmation can be made when the payment method has been called.
2. Input:

2.1 Pump ID: The identity of the pump is assigned when the customer arrives at the pump chosen and once one of the pump station nozzles has been lifted.

2.2 Fuel Type: Is an assigned value to each nozzle on the pump station and will reflect on the Unit Price later in the system.

2.3 Payment details (Pump Payment): This is a set of details obtained from the card reader and will extract the card number and account information etc.

1. Pump Start:

This is a hardware component that causes an event to occur. This event drives the Pump ID and Fuel Type to create a transaction record which updates the cost dependant on the Volume and Unit price selected at the pump.

1. Transaction: This is an active software component that will perform the following functions

4.1 Volume: The amount of fuel being taken at the pump will be constantly assessed and stored for calculation by this class.

4.2 Unit Price: This is a variable storing the data that is pulled and which is dependent upon the Fuel Type selected on the nozzle uplift.

4.3 Cost: The cost of the transaction is determined by the Volume and Unit Price multiplied together. The transaction class performs this calculation and pulls

1. Pump Finished: This is a hardware component that locks the pump from further use by any other customers. The component then sends a status reflections to the Pump Status method based on where payment details have been submitted or whether the nozzle has been returned.
2. Pump Status: The component processes all the Pump Definition data, Transaction data, Status of the pump and submits the record to the database. If no payment has been made a function within the pump status will output a message to the console alerting staff that the status of a single pump has not been completed due to no payment being made.
3. Database: The database will include a single entry for each transaction made through the system and store them for retrievable at the interface element of the console.

1. Output: The output will collate the relevant information from the database and output the request at the master control.

**Structure:**

DB

Transaction

Cost

Volume

Payment details

Unit price

Pump ID

Fuel type

Pump Start

Pump Finished

Pump Payment

Pump Definition

Input

Pump Status

Output

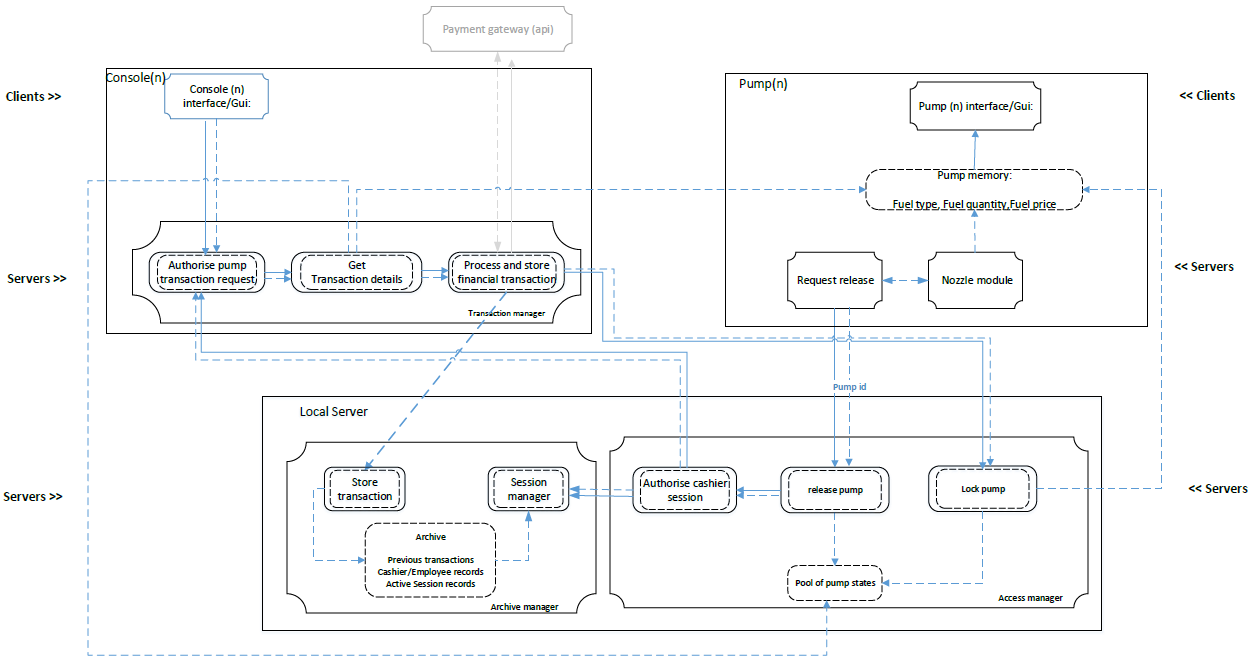
Master Control

**Client Server Architectural Style**

Definitions:

The diagrammatic representation below represents the interaction between a given pump (n) and a given cashier console (n). It is assumed once a console grants access the unlock request is removed from view of other consoles in the scope of unlocking a pump.

The style chosen is Client - Server. The rationale behind the choice being that the architecture can support expansion through modularity. This is a quality that would be useful for the proposed fuel station as the number of pumps is not defined.



**Components and Connector descriptions**

Clients**:**

1. Console interface: The console will receive input from physical interface devices such as a keypad and or a touch screen interface. The role of the console interface will be to allow cashiers to respond to request for pump access/activation and for processing completed transactions financially.

The console display will provide information to cashiers such as:

* A list of pumps and their statuses , inclusive of activation requests
* For any given fuel transaction information inclusive of:

Pump id

Transaction time

Fuel price per unit

Volume of fuel dispensed

Total payable, based on price per unit for the units dispensed.

# Pump Interface:

It’s will provide customers with the ability to request pump unlocks to enable them to use the pump. This however will not be an explicit action, the user will realise this function by lifting the pump nozzle. This action will automatically trigger a request to be sent to a console(s) for release.

A small lcd panel displays transaction information inclusive of:

fuel price per unit

volume of fuel dispensed

total payable , based on price per unit for the units dispensed.

Type of fuel

No customer interaction is required apart from lifting, using the nozzle and putting the nozzle back down.

# Servers:

# Transaction manager

3.1 Authorise pump transaction request:

Cashier response to a pump release request. Achieved through physical interaction with GUI by cashier and an automatic authorisation check.

* 1. Get transaction details

Given pump(n) has completed dispensing fuel, signified by customer placing nozzle back down, get transactions data inclusive of pump id, fuel price per unit, volume of fuel dispensed, type of fuel, and total payable based on price per unit for the units dispensed. Pump for selection determined by data in “pool of pump statuses”.

* 1. Process and store financial transaction

Using data returned from in 3.2 finalise transaction state by processing financial payment, and store transaction record in archive.

# Archive manager

4.1. Store transaction

Store a paid transaction into permanent memory on the server in the form of a database, inclusive of cashier id.

4.2. Session manager

Each cashier is required to log into their console at the start of their shift. Successful logon will start a session which expires when after a given duration unless they specifically log out.

4.3 Archive

Permanent storage containing previous transaction histories, employee details, and active session records.

# Access manager:

5.1. Authorize cashier session:

Given that a cashier has already logged into a console. An active session number will be present and the archive manager will reference this to verify whether the console is authorised to carry out an action.

5.2. Release pump

After receiving a release request from pump, send request to console and wait for a response. On response arrival check session information and proceed is valid.

5.3 Lock pump

Return pump state to locked and clear pump memory from previous transaction, ready for the next transaction. Display should reset and show default values.

# Request release

Receives input from nozzle module when nozzle is lifted and sends a request to the access manager.

# Nozzle module

On-board module that reads sensor information on whether the nozzle is in it’s default location, i.e. in pump, or has been lifted and sends a data to “Request release” if lifted.

# Pump memory

Temporary memory responsible for keeping information from the pump nozzle and information derived from the pump nozzle information, to be display by the Pump display.

**Section 3: Design Evaluation and Comparison**

The designs from the previous section are now to be analysed using the SAAM technique.

**Definition of scenarios related to modifiability:**

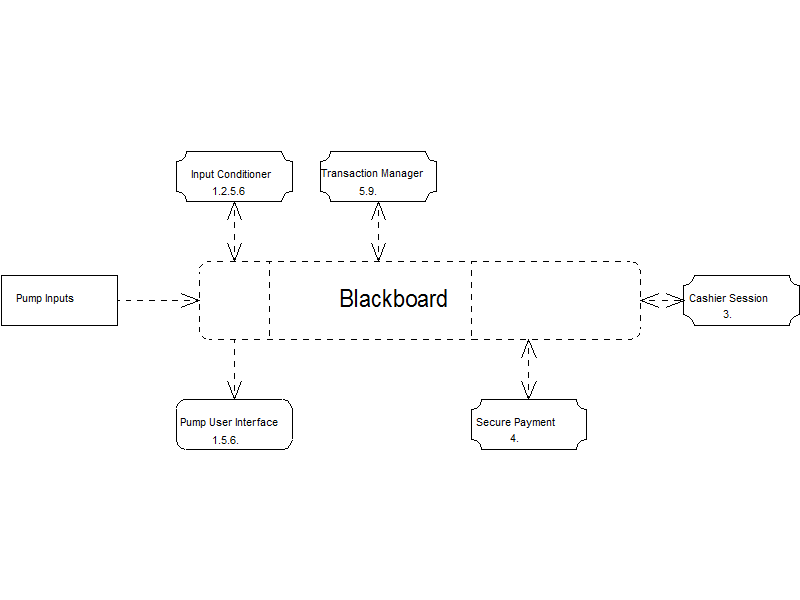
The following scenarios we conceived in relation to modifiability. The stake holder in each of these scenarios is the user but specifically the business owner not the customer of the business, ie, motorist.

|  |  |  |  |
| --- | --- | --- | --- |
| Scenario No | Description | Weight | Weight Justification |
| 1 | User wants to add the facility to charge electric vehicles. | 15 | This will probably happen in the future. |
| 2 | User wants to add a new pump from a different manufacturer. | 10 | There is a good chance this will happen. |
| 3 | User wants a new interface for the cashier with touch screen | 15 | This will probably happen in the future. |
| 4 | User wants to accept payment via an NFC device | 10 | This may happen. |
| 5 | User wants to add pay at pump facility | 10 | This is a common function that may be desired. |
| 6 | User wants to add an additional grade of fuel | 10 | This may happen as other fuels become available. |
| 7 | User wants to add another pump. | 20 | This will probably happen in the future. |
| 8 | User want to add another cashier console | 5 | This probably won’t happen. |
| 9 | User want to add the ability to combine multiple pump usages into a single transaction | 5 | There is no real use for this so it probably won’t happen. |

**Data Centered “Blackboard” Architectural Style Evaluation:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario | | | Change | |
| No. | Description | Type | Component | Change |
| 1 | User wants to add the facility to charge electric vehicles. | In-Direct | Input Conditioner | Input conditioner will have to be modified to deal with the different format |
|
| Pump UI | UI will have to change slightly to reflect the new units |
| 2 | User wants to add a new pump from a different manufacturer. | In-Direct | Input Conditioner | Only this component will need to be modified to interpret the new signals |
| 3 | User wants a new interface for the cashier with touch screen | In-Direct | Cashier Session | This component would need to be modified as it contains the user interface |
| 4 | User wants to accept payment via an NFC device | In-Direct | Secure Payment | This would need to have the NFC protocol added. |
| 5 | User wants to add pay at pump facility | In-Direct | Input Conditioner | This would change to take payment details input |
| Pump UI | This would change to display payment status |
| Transaction manager | This would need to be changed to interact directly with the secure payment component. This would cut out the cashier component. |
| 6 | User wants to add an additional grade of fuel | In-Direct | Input Conditioner | This would need to be changed to recognise a different pump is used |
| Pump UI | This would need to be changed to reflect the additional fuel. |
| 7 | User wants to add another pump. | Direct | No change necessary |  |
|
|
|
| 8 | User want to add another cashier console | Direct | No change necessary |  |
|
|
|
| 9 | User want to add the ability to combine multiple pump usages into a single transaction | In-Direct | Transaction manager | This would need to be modified to be able to combine multiple pump uses into a single transaction. This module decides what constitutes a transaction so should be the only component that needs to be modified. |
|
|
|

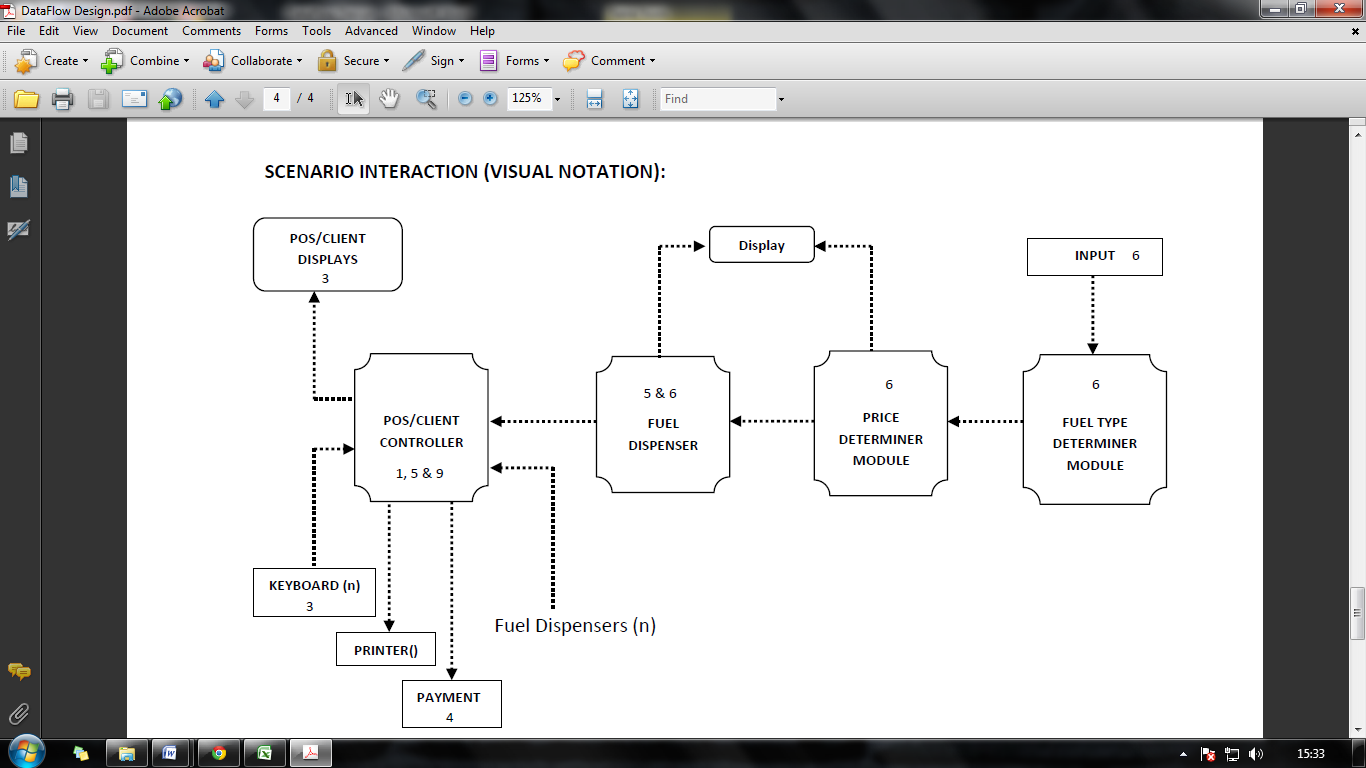
Scenario Interaction:



From the above diagram we can see the interactions of the different scenarios. Note that the blackboard itself isn’t technically modified, it is a central data structure that remains constant. Modifications are only applied to the processes.

**Pipe and Filter Architectural Style Evaluation:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario | | | Change | |
| No. | Description | Type | Component | Change |
| 1 | User wants to add the facility to charge electric vehicles. | In-Direct | POS/Client Controller | It will need to show the amount or electricity used. |
| 2 | User wants to add a new pump from a different manufacturer. | Direct | - | - |
| 3 | User wants a new interface for the cashier with touch screen | In-Direct | POS/Client Display | This component would need to be modified as it is the interface for the cashier |
| Keyboard | This component will no longer be required. |
| 4 | User wants to accept payment via an NFC device | In-Direct | Payment | NFC technology would have to be implemented in order to allow the use of such devices. |
| 5 | User wants to add pay at pump facility | In-Direct | Fuel Dispenser | It will need modifying in order to enable it to process a transaction and add card-reading technology. |
| POS/Client Controller | This would have to handle transactions that are closed at the pump. |
| 6 | User wants to add an additional grade of fuel | In-Direct | Input | Extra Nozzle will have to be installed |
| Fuel Type Determiner | This will have to be modified in order to enable it to determine the new grade. |
| Price Type Determiner | Will have to be modified in order to enable it to determine the new price. |
| Fuel Dispenser | Will have to be modified in order to enable it to dispense the new grade. |
| 7 | User wants to add another pump | Direct | - | - |
|
|
|
| 8 | User want to add another cashier console | Direct | - | - |
|
|
|
| 9 | User want to add the ability to combine multiple pump usages into a single transaction | In-Direct | POS/Client Controller | They will be the need to modify the POS/Client Controller, to enable it to combine two or more transaction. |
|
|
|

****

**Object Orientated Architectural Style Evaluation:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scenario** | | | **Change** | |
| **No.** | **Description** | **Type** | **Component** | **Change** |
| **1** | User wants to add the facility to charge electric vehicles. |  | Send transaction ’n’ to the server | This component is the driver interface between the pump and the software system. Updating the driver to interact with the new ‘pump’. |
| **2** | User wants to add a new pump from a different manufacturer. |  | Send transaction ’n’ to the server | This component is the driver interface between the pump and the software system. Updating the driver to interact with the new ‘pump’. |
| **3** | User wants a new interface for the cashier with touch screen |  | User selects payment type / User requests transactions for ‘Pump n’ / Display Amount to pay / User cancels transaction | All components in the GUI need updating so that they can interact with a new hardware. |
| **4** | User wants to accept payment via an NFC device |  | Prepare for transaction / Return transaction | The card payment components need changing at the driver level in order to interact with the new hardware |
| **5** | User wants to add pay at pump facility |  | 7 functions that match those of the main and card payment classes in console need adding to the system | 7 functions that match those of the main and card payment classes in console need adding to the system |
| **6** | User wants to add an additional grade of fuel |  | Fuel Value | The fuel value database would need updating with the new type and price |
| **7** | User wants to add another pump |  | None |  |
| **8** | User wants to add another cashier console |  | None |  |
| **9** | User wants to add the ability to combine multiple pump usages into a single transaction |  | None |  |

**Implicit Invocation Architectural Style Evaluation:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario | | | Change | |
| No. | Description | Type | Component | Change |
| 1 | User wants to add the facility to charge electric vehicles. | Indirect | Input, Pump Definition, Transaction | A new input type will be required which therefore changes the pump definition and fuel type desired. This in turn requires a modification to the transaction where a different unit price and volume measurement will need to be added. |
|
|
|
| 2 | User wants to add a new pump from a different manufacturer. | Indirect | Master Control, Input, Pump Definition | The master control works as the user interface and the main program. This also requires modifications in the input and pump definition components to accommodate for compatible functionality. |
|
|
|
| 3 | User wants a new interface for the cashier with touch screen | Indirect | Master Control, Output | The master control and output will require modification as they directly affect the interface of the cashier. |
|
|
|
| 4 | User wants to accept payment via an NFC device | Indirect | Input, Pump Payment, DB | The input component will need to be changed to accommodate for the interface change on the pump. Secondly, the pump payment component contains all the payment details and manipulates them into a format acceptable for the database. In this case details may be stored in a different format or more details may be required. |
|
|
|
| 5 | User wants to add pay at pump facility | Indirect | Input, Pump Definition, Pump Payment, Transaction, DB | The pay at pump facility requires a different input type to be assigned to the pump, this requires modifications to be made to the pump definition and pump payment components to accommodate for a merger of the data contained within those components. This will also impact the transaction, DB and the flow of data from input to transaction. |
|
|
|

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario | Change |  |  |  |
| No. | Description | Type | Component | Change |
| 1 | User wants to add the facility to charge electric vehicles. | In-Direct | Pump memory | Amount of electricity used to be added. Pump lock algorithm would need to be adapted such that a pump can be locked in use – due to the lengthy recharge process of electric vehicles. |
| 2 | User wants to add a new pump from a different manufacturer. | Direct | - | None – Under this architecture the pumps can be added in a modular fashion, provided the method of interfacing and data access and transmission remains the similar. |
| 3 | User wants a new interface for the cashier with touch screen | Direct | Console(n) interface | None – proposed system requires I/O streams and event such as clicks. The source and generation does not impact system. |
| 4 | User wants to accept payment via an NFC device | Direct | Process and store financial transaction | None - new physical hardware will still provide the same information required to complete a transaction. |
| 5 | User wants to add pay at pump facility | In-Direct | Pump interface (gui) | Physical card reader will need to be added to pump and a secure connection required to access external payment gateways. |
| 6 | User wants to add an additional grade of fuel | In-Direct | Archive | Changes to data, however the system remains semantically correct. |
| 7 | User wants to add another pump | Direct | - | None – another pump may be added modularly. |
| 8 | User want to add another cashier console | Direct | - | None – another pump may be added modularly. |
| 9 | User want to add the ability to combine multiple pump usages into a single transaction | In-Direct | get transaction details, process and store financial details, | Get transaction details must changed to obtain transaction of n pumps and not process payment prior to all pump data being received.  Store financial details needs to be modified to not associate a single transaction to a single pump id |
|  |  |  |  |  |

**Client Server Architectural Style Evaluation:**

Comparison of Designs:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Scenarios | | Architectures | | | | |
| No | Weight | Data Centred Blackboard | Client Server | Implicit Invocation | Object Oriented | Pipe and Filter |
| 1 | 15 | 2/7 |  | 3/8 |  | 1/10 |
| 2 | 10 | 1/7 |  | 3/8 |  | 0/10 |
| 3 | 15 | 1/7 |  | 2/8 |  | 2/10 |
| 4 | 10 | 1/7 |  | 3/8 |  | 1/10 |
| 5 | 10 | 3/7 |  | 5/8 |  | 2/10 |
| 6 | 10 | 2/7 |  | 3/8 |  | 4/10 |
| 7 | 20 | 0/7 |  | 0/8 |  | 0/10 |
| 8 | 5 | 0/7 |  | 0/8 |  | 0/10 |
| 9 | 5 | 1/7 |  | 3/8 |  | 1/10 |
| Overall | | 17.143 |  | 28.750 |  | 12.000 |

**Section 3: Teamwork Exercise**

Teamwork was an important part of this assignment. In order to maintain an effective working arrangement we held weekly meetings, the minutes from these were documented, see appendix 1.

As a large portion of the coursework involved splitting the work load evenly and allocating a single design to each group member it was important that work from each member was cross checked. Work would be allocated at a meeting, the following week the work was reviewed and commented on. Changes were applied if all group members were in agreement. In addition to these reviews the group used google docs to maintain constant involvement with all work. Each piece of work would have a comment box on the last page where other group members could comment and any changes to that document could be recorded.

The following table displays who specifically did what:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Shaz Ashraf | Terry Mukonka | Ryan Varley | Tanaka Chimbuya | Lee Hudson |
| Identify requirements and constraints | √ | √ | √ | √ | √ |
| Pipe and filter design |  | √ |  |  |  |
| Object orientated design |  |  | √ |  |  |
| Client server design |  |  |  | √ |  |
| Implicit invocation design | √ |  |  |  |  |
| Blackboard design |  |  |  |  | √ |
| Scenario definitions | √ | √ | √ | √ | √ |
| Pipe and filter design evaluation |  | √ |  |  |  |
| Object orientated design evaluation |  |  | √ |  |  |
| Client server design evaluation |  |  |  | √ |  |
| Implicit invocation design evaluation | √ |  |  |  |  |
| Blackboard design evaluation |  |  |  |  | √ |
| Designs comparison | √ | √ | √ | √ | √ |
| Meeting minutes | √ | √ | √ | √ | √ |

As you can see from above each member took responsibility for a design and evaluation of that design. That is not so say other group members weren’t involved in these designs. Each piece of work has had some kind of input from other members in an attempt to increase the quality of the designs and maintain consistency throughout this design document.

**Section 4: Individual Reports**

Lee Hudson 09092543

I will assess my performance based on the following criteria:

1. Interacting professionally within the group, including engaging effectively in discussions and/or debating in a professional manner, negotiating in a professional context and managing conflict;
2. Recognising, supporting or being proactive in leadership;
3. Being autonomous in taking responsibility for own work and criticising it;
4. Seeking and making use of feedback on his/her own work, and reflecting on action, the application of own criteria of judgement and, when appropriate, challenging received opinion;
5. Contributing to the technical development of the project;
6. Producing detailed and coherent project reports;

For the above criteria I feel the following grades are correct:

1. A - I attended all meeting except for one. I took minutes for multiple meetings and set the agenda for following meetings. During the group meetings I was an active member who contributed to discussions and debates on how best to proceed with the work.
2. A - Ryan was our group leader and we all supported this. Still, I was able to have an impact on decisions within the group. During the meetings the way we decided to proceed with work was collaborative. It is important to realise that not everyone in the group can be the leader but all group members can have an active role in the group and make substantial contributions.
3. A+ - I always had my work completed for the dates set in the meeting agendas. As a group we all cross checked each other’s work but I also critically assessed my own work. During the design stage I went through each element of the control system to ensure myself that I could code the system with the final design. Also, it is hard to realise the practical application for both the design and evaluation but I made sure I understood the practical implications of every piece of my work.
4. A - Through each stage of the coursework I got feedback from the lecturer and from the other members of the group. I also feel that I gave strong and accurate advice / feedback to group members. I did all the reading recommended to ensure I had a good grasp of all design styles.
5. A+ - Throughout the assignment I had a key role in the technical development. I had a good understanding of the design task and as such was able to get a design finished in good time to assist other members of the group with their development. I feel I had an exceptional grasp of the technicalities of the control system.
6. A - I believe my written work is concise and applicable to the practical implementation and understanding of my design. All aspects of my design were explained with the aid of diagrams and descriptions.

# 

Terry Mukonka 09081449

In this project, we as a group worked to our full potential, and this was achieved by the way we interacted and communicated with one another. Some research shows that effective team communication is a vital aspect of a positive outcome to any project that requires a significant amount of a collaborative effort, and this was no different.

In this project I found myself interacting professionally compared to most of my previous projects during the course of the degree. My attitude and behavior towards the task at hand was that of positivity and willingness to learn with the focus of doing the task right, this was reciprocated in the way I engaged with the group in meetings and debates and in the way we addressed issues regarding the task, my input to the task was well heard and applied by my group and so did there input. Not only did this show that I was engaging with the group, it showed that I could present my opinion in a clear and professional manner in that I listened to what others had to say too and debated reasonably if necessary. However, individuals in the group all acted professional resulting in good team dynamics; this sometimes always is the case in real world.

Having gone into this project with a positive attitude and the willingness to learn, it’s only fair to say that this helped in some situations in which I found myself frustrated as I recognized some of my weakness however this did not hinder me in actively supporting the group, I attended group meetings and actively debated and contributed in tasks. The group itself was formed fairly easy and when it came to choosing the group leader, one member quickly called for the position and we, as a group had no objections to that, I personally would have had no objection if I were to be elected leader.

As a group we each had individual tasks to accomplish. The degree of autonomy in this project could be considered high, as we each had a choice as to what we felt capable of doing in terms of styles. As a group, we agreed to a deadline to when we was to present the chosen style we was to work on, I went for the dataflow and presented it to the group. It was my responsibility to insure that I met that deadline and that the work was to be up to date which I can say I did meet, we further agreed that each member can criticize another’s deign in order to allow room for development and to enhance the learning experience of different designs and views. From this experience, the feedback from the group members was used to enhance and correct the design and in many causes, I was not far off the concept. I further contributed to the technical development of the project in that I carried out the evaluation of the design and this was also double checked by the group and so did I too check those of others.

It is fairly easy to say we, as a group have performed to high stands and being part of that group my contribution, effort, commitment, communication and positive attitude played a part in archiving what we have achieved. It is therefore only logical, fair and sincere that my self- assessment grade in this assignment should be noted down as an “A”. I feel that I have supported my group by achieving set deadlines, attending and contributing to meetings both before and after the lecture and all debates that we as a group had and capitalized on the feedback put in by my group members.

Appendix 1: Documentation of meeting minutes

**Date:** 2014/02/04

**Time:** 16:00 - 16:30

**Attendees:**

* Ryan Varley (Team Leader)
* Lee Hudson (Technical Specialist)
* Terry Mukonka
* Tanaka Chimbuya

**Apologies:**

* Shaz Ashraf

**Scribe:**

* Ryan Varley

**Agenda**

New Business:

* Arrange Group dynamics
* Document (non)functional requirements
* Create a list of constraints
  + Based on Lawson's theory
* Find existing schematics for a similar system
* Post individual research to a google doc by Friday 7th February

**Next Meeting**

Agenda:

* Consolidate individual research
* Create exercise report from research

Location: Wheatley C228

Time: 16:00

**Date:** 2014/02/11

**Time:** 16:00 - 17:00

**Attendees:**

* Ryan Varley (Team Leader)
* Terry Mukonka
* Tanaka Chimbuya
* Shaz Ashraf

**Apologies:**

* Lee Hudson

**Scribe:**

* Ryan Varley

**Agenda**

Previous Business:

* Document (non)functional requirements
* Create a list of constraints
  + Based on Lawson's theory
* Find existing schematics for a similar system
* Post individual research to a google doc by Friday 7th February

New Business:

* Consolidate the group's functional and nonfunctional requirements
* Consolidate the group’s constraints

**Next Meeting**

Agenda:

Location: Wheatley C228

Date: 2014/02/18

Time: 16:00

**Date:** 2014/02/18

**Time:** 16:00 - 17:00

**Attendees:**

* Ryan Varley (Team Leader)
* Lee Hudson (Technical Specialist)
* Terry Mukonka
* Tanaka Chimbuya
* Shazaad Ashraf

**Apologies:**

* N/A

**Scribe:**

* Shazaad Ashraf

**Agenda**

Previous Business:

* Consolidate the group's functional and nonfunctional requirements
* Consolidate the group’s constraints

New Business:

* Ensure functional requirements are clear and precise, include an actor, action and result
* Non functional requirements really need to include quality factors (McCalls model)
* Constraints (Reference recording) - A majority of these were factored in as non functional requirements. Therefore constraints are more factors that the developer feels the system may incur. Eg Server is required = Impact
* Diagram and brief paragraph describing assumptions made on the hardware.

**Next Meeting**

Agenda:

* Access Google Docs, enforce the new business objectives as a team and agree a document development period of time.

Location: Wheatley C228

Time: 16:00

**Date:** 2014/02/25

**Time:** 16:00 - 17:00

**Attendees:**

Ryan Varley (Team Leader)

Lee Hudson (Technical Specialist

Tanaka Chimbuya

Shazaard Ashraf

**Apologies:**

Terry

**Scribe:**

Lee

**Agenda**

Previous Business:

Check through requirement specs

New Business:

Lee to read requirement spec and add notes before the rest of the group meet with hong on thursday 27/02

Requirements to be finally reviewed with hong on thursday

Each do a design of the control system in our own chosen style:

Lee: Data Centered - Blackboard

Ryan: Call and return - Object Orientated

Shazzard: Individual component - Event Systems

Tanaka: Individual Component - Client Server

Terry: Data Flow - Pipe and filter

**Next Meeting**

Agenda: Check on progress of the above

Location: Wheatley C228

Time: 16:00

**Date:** 2014/03/04

**Time:** 15:00 - 16:00

**Attendees:**

* Ryan Varley
* Lee Hudson
* Terry Mukonka
* Tanaka Chimbuya

**Apologies:**

* Shazaard Ashraf

**Scribe:**

* Ryan

**Agenda**

Previous Business:

* Each do a design of the control system in our own chosen style:
* Lee: Data Centered - Blackboard
* Ryan: Call and return - Object Orientated
* Shazzard: Individual component - Event Systems
* Tanaka: Individual Component - Client Server
* Terry: Data Flow - Pipe and filter

New Business:

* Check in on progress, direction and planning

**Next Meeting**

Agenda:

* Create basic outline of system design:
  + Each do a design of the control system in our own chosen style:
  + Lee: Data Centered - Blackboard
  + Ryan: Call and return - Object Orientated
  + Shazzard: Individual component - Event Systems
  + Tanaka: Individual Component - Client Server
  + Terry: Data Flow - Pipe and filter

Location: Wheatley C228

Time: 16:00

**Date:** 2014/03/11

**Time:** 15:00 - 16:00

**Attendees:**

* Ryan Varley
* Lee Hudson
* Terry Mukonka
* Tanaka Chimbuya
* Shazaard Ashraf

**Apologies:**

**Scribe:**

* Lee

**Agenda**

Previous Business:

* Check in on progress, direction and planning

New Business:

* Apply changes to each design as discussed to create the final revision of the designs

**Next Meeting**

Agenda:

Check final revisions of the designs ready for evaluation

Location: Wheatley C228

Time: 16:00

**Date:** 2014/03/18

**Time:** 15:00 - 16:00

**Attendees:**

* Ryan Varley
* Lee Hudson
* Terry Mukonka
* Tanaka Chimbuya
* Shazaard Ashraf

**Apologies:**

**Scribe:**

* Lee

**Agenda**

Previous Business:

* Apply changes to each design as discussed to create the final revision of the designs

New Business:

* All designs now checked and mostly approved and in a state when they can be evaluated. For next week all designs should be evaluated so the comparisons can be done in the meeting as a group

**Next Meeting**

Agenda:

Check evaluations and do the comparison of the designs as a group.

Location: Wheatley C228

Time: 16:00

**Date:** 2014/03/25

**Time:** 15:00 - 16:00

**Attendees:**

* Ryan Varley
* Lee Hudson
* Terry Mukonka
* Tanaka Chimbuya
* Shazaard Ashraf

**Apologies:**

**Scribe:**

* Lee

**Agenda**

Previous Business:

* Evaluate designs

New Business:

* All group members completed the design evaluations. Comparison of designs performed during this meeting as a group.

**Next Meeting**

Agenda:

Individual reports to be done by next week. Teamwork report and correlation of complete document to be completed in next meeting

Location: Wheatley C228

Time: 16:00