MSc Applied Computer Science

Postgraduate Diploma Project Brief

Part I – Application Design & Design Assessment

(Module FDM05)

Revision: September 2020

Table of Contents

1. Purpose and Scope

The project specified in this document is intended for use by students on their second year of the FDM MSc Applied Computer Science Programme who are studying at Postgraduate Diploma level. The demands of this project are expected to provide the necessary opportunities for students to fulfil Learning Goals FDM05 and FDM06 of the Programme. Instructions provided in this document are sufficient to complete Part I of the Project and correspond to the requirements of Learning Goal FDM05 only. The Functional Requirements set out in Section 7 are required for both parts of the project.

2. Relationship to Learning Goals FDM05 & FDM06

This project requires the student to translate the functional requirements detailed in a Requirements Specification into two or more technical design options. Decisions taken during the translation process will determine both the architectural and performance characteristics (elegance and efficiency) of the finished application, productivity, and the ease with which subsequent modifications can be made. Students are expected to research and present their own understanding of what architectural and performance characteristics contribute to the elegance and efficiency of application designs and then use this understanding to set their own design goals for their project. By setting these goals, students will create criteria to qualitatively assess selected functional areas of their subsequent technical design options. Examples of architectural characteristics would include cohesion, coupling, dependency and modularisation. Students are encouraged to methodically experiment with different architectural options by, for example, utilising different design patterns to implement identical functionality and then compare the outcomes using their selected design criteria.

This project is split into two distinct parts, each intended to allow the student to achieve the learning objectives set out in FDM05 and FDM06, respectively. Part I deals with the Application Design Process and the critical assessment of Application Designs. Part II deals with implementation and testing of applications, taking into account Design Goals which have been set out in Part I.

Both FDM05 and FDM06 will be assessed at the same time, and should be submitted together in-line with the deadline set for you by the MSc Programme Co-ordinator.

3. Instructions

These instructions give direction for how to commence and progress your project to the design and assessment stage. They are likely to be supplemented by further instructions and support by email, as and when the need arises. It is very important that you contact the MSc coordinator if you have any questions, so that instructions can be clarified and issues arising resolved.

Keep a journal of your activities, challenges and learning points as you progress through this project. It will help you to log your progress and to remember key points when you come to reflect on your work at the end of the project.

Project Tasks

1. Familiarise yourself with the project scenario and read the descriptions of the application Client and Server requirements, and the Stock Handling functionality.

Minimum Resources:	Functional Requirements in this document
Required Deliverables:	none

2. Using suitable literature sources, establish what criteria can be used to measure, define or assess *elegance* (good design practice) and *efficiency* in application design.

   These may include the measures to allow ease of extension, refactoring or re-use of code. Cohesion, coupling, dependency and modularisation are examples of other measures encountered. The SOLID design principles you will have encountered in your initial FDM training are a good starting point for a guiding list, but do not be afraid to investigate further afield. There are many flavours of design principles claiming to favour faster design pathways, to give a purer implementation of object-oriented design, or other desired outcomes.

Minimum Resources:	Suitable literature on application architecture and design
Required Deliverables:	none

3. Select at least three distinct measures from step (2) as design goals that you are interested in pursuing and are prepared to investigate, apply, compare and use as a basis for critical assessment over the duration of this project. These should be relatively high-level and general at this stage.
   Email a description of your Design Goals to your MSc Programme Co-ordinator.

   The *efficiency* of a particular application design and its associated implementation are often intrinsically linked to architectural elegance, with one sometimes being played off against the other, or sometimes going hand-in-hand with the other. The architectural impact of using Design Patterns can improve productivity but at the cost of performance or other attributes. You may want to investigate these further when reasoning the selection of criteria for your own Design Goals.

Minimum Resources:	Knowledge acquired from step (2)
Required Deliverables:	Design Goals

4. Analyse the requirements you read in step (1), and propose an initial technical solution that applies your design goals.

   Read step (5) and its notes before commencing this work.
   Evidence for this work would include the presentation of an appropriate set of UML or similar models, providing different perspectives of your solution (structural, behavioural, etc). Additionally, you will require evidence of a designed data model. Provide a written account of the reasoning behind your design decisions and how they met your design goals.

Minimum Resources:	Functional Requirements in this document Design Goals
Required Deliverables:	An initial Technical Design Option

5. Repeat step (4) using an alternative design strategy to arrive at a different solution. Again, provide a written account of the reasoning behind your design decisions and how they demonstrated a different outcome.

   This process may have an architectural impact on part of, or the whole of, your original solution from step (4). An *alternative strategy* could mean using a different design pattern or no pattern at all, or perhaps a differently structured data model, or different levels of modularisation. Whichever design strategies you adopt, they need to tangibly demonstrate how your design goals have been variously met or not met in a measurable way.
   Email both Technical Design Options as professionally formatted reports to your MSc Programme Co-ordinator.

Minimum Resources:	Functional Requirements in this document, Design Goals
Required Deliverables:	A second Technical Design Option

6. Critically assess your two alternative solutions using your design goals and their associated indicators as a basis for comparison. Prepare a written report on your assessment of how your alternative strategies have impacted on your solution outcomes.
   Email your Critical Design Assessment to your MSc Programme Co-ordinator.

   Your report should include as much detail as possible of your comparison methodology. For example, describe how you compared, measured or otherwise evidenced the impact of alternative design strategies. Similarly, comment on what parts of your solution were included or excluded when measuring the impact of your design changes.

Minimum Resources:	2x Technical Design Options, Design Goals
Required Deliverables:	Critical Design Assessment Report

7. Produce a Test Plan.
   Email your Test Plan to your MSc Programme Co-ordinator.

   Define a test strategy including a plan for suitable Functional *and* Performance Testing to be able to prove to yourself and others that all requirements have been met, in both terms of functionality and efficiency of the application. Consider suitable methods to verify that all requirements have been met in full. Given that both of your Technical Design Options should meet all the requirements of Section 7, you should be able to apply a common Test Plan.

Minimum Resources:	Functional Requirements in this document, Technical Design Options
Required Deliverables:	Test Plan

4. Completion and Delivery Plan

Part I of this project has been designed to meet all the Learning Objectives specified in FDM05. The submission of work will be phased and has been scheduled according to the following plan:

Deliverable *Project issued to student*	No of weeks
Design Goals	5
2x Technical Design Options	6
Critical Design Assessment	4
Functional Test Plan	3

It is essential that students keep to the delivery schedule so that sufficient time is available for them to complete their MSc dissertation work within the period allowed by the University (three years from date of registration). Subject to satisfactory completion of Part I of the project and FDM approval, students will then continue directly on to Part II.

5. Suggested Tools

A UML CASE tool (e.g. Visual Paradigm for UML) will be required for this part of the project.

6. Assessment Scheme for FDM05

Your response to this project assignment will be assessed against the criteria set out in the table below.

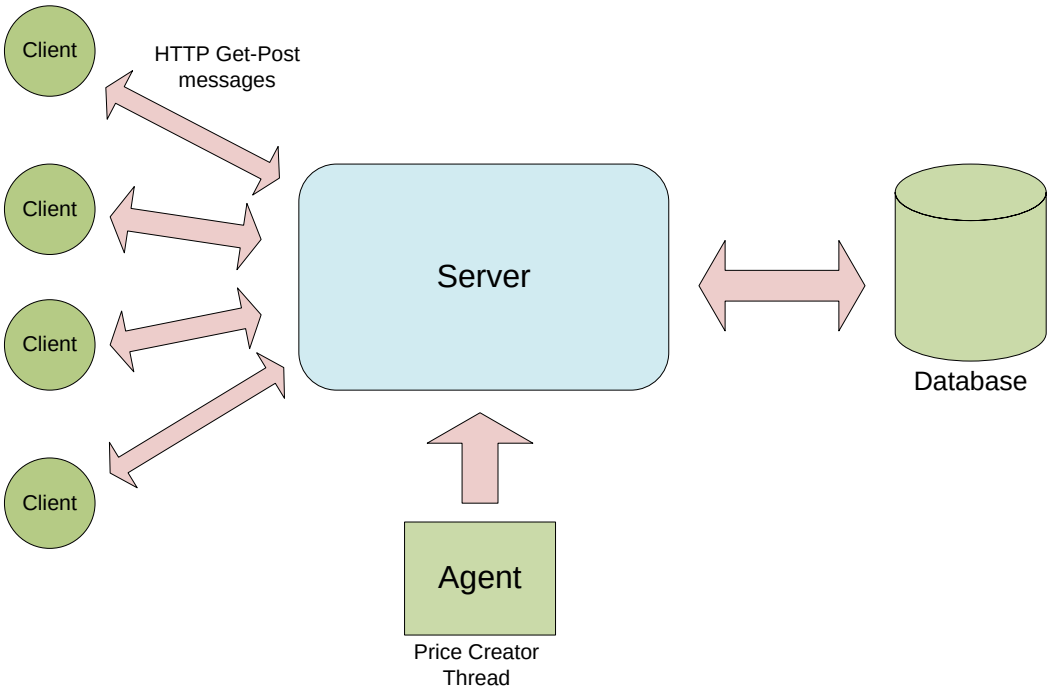
Learning Goal: To develop a deep understanding of a specific programming language (e.g. Java or C#.NET) and to be able to utilise a range of applications, libraries and tools to efficiently design and develop high quality applications.

Objective No	Learning objective	Tasks	Indicative deliverable(s)	Assessment criteria
FDM05a	Create applications which demonstrate a detailed knowledge and understanding of the appropriate usage of programmatic commands, constructs and libraries.	- Translate FRS into Technical Design Options	- Technical Design Options	- Propose designs that will exploit commands, constructs and technologies from a chosen paradigm (e.g. Java or .NET) to produce elegant and efficient programmatic solutions
FDM05b	Analyse functional requirements and assess alternative solutions to isolate elegant and efficient implementations to demonstrate a comprehensive awareness of issues related to application design.	- Define Design Goals - Translate FRS into - Technical Design Options - Assess Design Alternatives - Plan Functional Tests	- Design Goals - Technical Design Options - Critical Design Assessment - Functional Test Plan	- Comprehensive translation of functional requirements into technical design - Significant issues related to application design (eg extensibility, cohesion, refactoring, coupling, dependency, modularisation) are explained and applied.
FDM05c	Critically assess the qualitative aspects of application design and technical implementations for complex functional requirements.	- Assess Design Alternatives	- Critical Design Assessment	- Recognition of the comparative implications of various application design approaches, and implementations.

7. Project Requirements

What follows is a description of the overall scenario for this project, requirements for application Client and Server components, and detailed requirements for Stock Handling functionality. Students are expected to deliver a project that meets all these requirements in order to attain maximum marks.

7.1 Scenario



The application proposed for this project will be a Real-Time Stock Trading Simulator which allows multiple users to view, buy, and sell stocks from a central server. Users will be trading stocks through a broker, whose purpose is to manage requests from users and decide if they are feasible.

The application will maintain user state across sessions. A user will be required to pass account authorisation before performing transactions.

Users may request a new account via registration. Account activation can only occur once the user has confirmed their email is valid. Upon successful registration, users are given a fixed amount of credit. Users will be unable to add credits to their account. They must obtain credit through the process of interacting with stocks. Each user will have a profile area that will detail their past transactions and account details.

The administrator is the super user of the application who can alter the details of the users and companies. Companies should be added by the admin to the database. User access to the application can be blocked by the admin for a period of time.

The server shall broadcast stock data to each of the registered online clients, showing the latest stock data from the server. The user can choose to interact with this data, in doing so there is no guarantee that the request will be accepted. This will be decided by a broker.
A reporting suite will allow users to create a report from a given set of report templates. Each report may be generated into various formats. The report will be generated externally but sent to the client to be saved locally.

7.2 Client Description

The client acts as an interface for the server. No data processing shall occur on the client-side other than validation of input data, and presentation of data. Validation should be performed on all user inputs. Validation must check:

- If the field is mandatory, whether it has a value
- If the input adheres to the constraints
- If the input has the correct data type

Ensure that user interface data validation is used universally in the client. If client-side validation ever fails, prevent the data being sent to the server and inform the user specifically what has failed. Failed data must be preserved to allow the user to edit the errors, with exception to hidden fields.

The client shall be written using a suitable language (e.g. HTML). CSS must be employed to ensure that there is a consistent style for the presentation. There are no restrictions to the actual presentation of the client, but a professional 'look and feel' is required. Ensure that text is descriptive with correct grammar and spelling. To improve the user experience, confusion about the page functionality should be minimised with correctly labelled buttons, links and descriptive text. Navigation should be simple, nested navigational links should be avoided to prevent confusion. It may be necessary to define a set of universal links and have it accessible from every page.

Below are the basic pages that are required for the client:

Page	Details
Stocks	Provides transaction functionality
Registration	Registration for a new user account
Account Details	Provides account details
Reports	Functionality to create reports

Additional pages may be added given reasonable justification. Similarly, additional functionality may be added or split into other pages from these foundational pages if deemed suitable.

7.2.1 Stocks

This page acts as an interface into the available shares on the server. Users will be provided transaction functionality for shares. The user can purchase stocks depending on their available credit.

Shares should only be sellable if the user has current possession of them. Shares should only be buyable if there are any available shares.

The user should be able to buy or sell a set of shares, or a single share. Give the user the ability to select multiple shares, rather than having to purchase singular shares individually.

As sellable shares will be displayed dependent upon the user's possession of shares, if the user sells all the held shares of a stock, the stock will no longer be visible as a sellable share.
It is recommended to present buyable and sellable shares separately, but is not required.

As the data is dynamic and dependent upon several factors, a real-time stream of data is necessary, for example, using AJAX with JQuery. Ideally the server will broadcast stock data towards clients that are registered with the server; it is deemed sufficient to have the client request data from the server periodically with a high frequency.

Table 1: Field Requirements

Field Name	Field Description	Format
Stock symbol	A short abbreviation that identifies the publicly traded shares of a stock.	Must be capitals
Stock Name	Name of the company	No restriction
Last Trade	The time of the last trade	[Hour:Minute:Second]
Gains	Gains for the trading day	[Number]
Value	The current value of the stock	[Number]
Volume	The number of shares available for the stock	[Number]
Open	The value of the stock during stock market opening at 08:00 am	[Number]
Close	The value of the stock during stock market closing at 16:30 pm	[Number]

Table 2: Example Row

Field	Value
Stock Symbol	THSY
Stock Name	Therion Systems
Last Trade	14:48:14
Gains	8.5
Value	244.32
Volume	1400000
Open	235.82
Close	235.82

Table 3: Additional Fields

Field Name	Field Description	Format
Purchased Value	The value of the share at purchase	[Number]

Stocks should be presented in a tabular format. Refer to [Table 1] to understand how stocks should be presented. Refer to [Table 2] for an example of some stock data. Sellable shares shall have an additional field to represent the value of the share at purchase, see [Table 3].

Ensure that there is functionality to split the stock across multiple pages, use the default value of 20 stocks per page. All stocks may be presented on one page if the user wishes.

The user shall have the ability to refine the view of the stocks presented based on certain criteria. Searches should be performed based on:

- Stock Symbol, equal to a variable;
- Stock Name, equal to a variable;
- Gains, greater than or lesser than a variable;
- Value, greater than or lesser than a variable;
- Volume, greater than or lesser than a variable.

Additionally, they shall be able to order in ascending or descending order of a given field. By default, order of the stocks is based on the stock symbol.

Each row of the table will denote a company stock, with information about the stock itself. Transaction functionality must be present for every stock.

7.2.2 Registration

This page will allow a user to register for a new account. All fields must be validated. If any errors occur with the input, ensure that this is highlighted to the user in some form.

Table 4: Registration fields

Field Name	Field Type	Input Type	Details	Restrictions
User name	String	Textbox	The user name will be unchangeable after account creation, unless done so by administrator	Must be more than 3 characters and less than 16 characters. Must be unique
Password	String	Textbox	The user password to login to an account. Must have two field inputs, one of which is a confirmation field	Must contain numbers and letters. Must be more than 7 characters long
Email	String	Textbox	Email connected to the account. Must have two field inputs, one of which is a confirmation field	Must be a valid email address
First Name	String	Textbox	First name of the user	NONE
Last Name	String	Textbox	Last name of the user	NONE
Birthdate*	Date	<ANY>	The birthdate of the user	Must be a valid date

*denotes optional fields

Once submitted, the user shall be highlighted to the fact that they have registered a new account and an activation email has been sent to their email address. The confirmation email shall contain a link that will confirm that the user has access to that email address.

Note: Brokers do not register for the site directly and should instead be added by administrative users.

7.2.3 Account Details

This page will show information provided by the user during registration and details of previous transactions the user has performed.

All fields from registration will be modifiable, with the exception of the username. These must be validated in the same way as done in registration. A new email will have to be confirmed before it can replace the existing email.

Full detailed stock information shall not be visible on this page, only a general overview is displayed. This will include for each stock a:

Field	Description
Stock Symbol	
Stock Company	
Current Value	The current value of the stock
Purchase Value	The value of the stock at purchase
Share Amount	The amount of shares held

Ensure that there is a method to obtain complete stock purchasing history with full details of each stock.

7.2.4 Reports

The reporting page is an area the user can create a set of reports from data gleaned from the database. The following reports need to be provided:

- Currently held shares of the user. It should always be ordered by company name.
- Details from a list of chosen company stock. A stock can be chosen from the stock symbol. It will be ordered by company name.
- A complete list in ascending or descending order of the stock value.

Each of these reports may be presented in an XML or CSV format dependent upon a user's choice. Only the following fields need to be presented:

- Stock Symbol
- Stock Company
- Value
- Volume
- Gains

Below details a set of example outputs of a given stock.

XML Example:

```
<Stock>
    <Symbol> THSY </Symbol>
    <Name> Therion Systems </Name>
    <Value> 244.32 </Value>
    <Volume> 1400000 </Volume>
    <Gains> 8.5 </Gains>
</Stock>
```

CSV Example:

```
"Symbol","Name","Value","Volume","Gains"
"THSY","Therion Systems", "244.32","1400000","8.5"
```

When the reports are generated on the server side, the user is given the generated report as a download.

7.2.5 Further Pages

The application must have login functionality. Users shall have to login in order to access any of the sites content. The exception to this will be pages that won't be interacting with stocks or don't use user details.

When a user attempts to login with invalid values, they must be notified with an error message detailing that their username or password is incorrect.

An administrator page must be implemented to allow the administrator to access administrative functionalities. The administrative functionalities shall consist of:

- Viewing the details of the company and users. The administrator shall be able to view all details related to users and companies presented in a tabular format.
- Changing the details of company and users. The administrator shall be able to modify any details; essentially it will require an interface into the database.
- Adding new companies. New companies can be added, on creation a set of stock values will be created automatically.

7.3 Server Description

The predominant purpose of the server is to handle the requests sent by the clients. The server side of the application should be implemented to serve the different types of requests that will help handle complex functionalities. This section details all the mandatory functionality required prior to final submission.

7.3.1 Architecture

The client cannot retrieve any data from the database directly; instead it has to send a request to the server to communicate with the database.

7.3.2 Login and Logout Facility

Transactions with the stocks cannot take place unless the user logs in. The server should provide login facility to the user. Login facility should be able to handle the login requests by comparing the username and password with user details from database. The server should return an appropriate error messages if the server cannot find any matching candidate.

Logout facility also should be provided to invalidate the session of the user. After logout every object that is related to the user should be removed from the system, pending the completion of transactions.

Different roles should be created for the application. Two mandatory roles are ROLE_USER and ROLE_ADMIN. This should be persisted to the database when the user is created.

7.3.3 Security Measures

User login details should be stored in the database. The user's password should be encrypted before being sent to the database and a one way encryption algorithm should be used. SQL injection should be prevented by taking appropriate measures.

User interaction with the database should incorporate minimum permissions necessary to provide all functionality. Only DML queries should be permitted for users.

7.3.4 Thread Safety

The two areas that should be taken into the consideration of a concurrent application are:

- Thread Safety
- Performance

The server should be implemented to serve multiple users concurrently. To make the application thread safe, locks should be placed where it is necessary. Using more locks than is necessary will degrade the performance dramatically.

7.3.5 Database Interaction

Communication between the server and Database should be handled by the server. Connections to a database server can be a precious resource and should be managed in a sensible way to prevent unnecessary hogging, or disconnection followed by reconnection. SQL exceptions should also be handled.

Provide a facility to create a connection using a Datasource. A DriverManager (Java only) is not the preferred way and shouldn't be used.

7.3.6 Stock Transactions

Users can send a request to buy and sell the stocks. To buy or sell any stock, user should provide the company's symbol and the quantity of the stocks to be bought or sold. The server should handle the request by checking for stock availability. Then the server should check if the user has enough money to complete the transaction.

The stocks cannot be bought or sold directly by the user. All transactions should take place through broker class. A broker is the middle man for all transactions. The broker is responsible for getting the request from the user, validating the request, checking if there is sufficient stock and checking if there is enough money in the user's account.

When all the stocks are sold for the specific company, that company should be removed from the stock availability list. Companies can reappear in the availability list when one of the clients release its stocks back to the market.

7.3.7 User Functionalities

When the user account is created, a minimum amount will be deposited to the users account. The user will start out with £50k. The account will be closed when the user cannot afford to buy any stock and doesn't hold any stock. The user's details shouldn't be removed from the database, but it should be deactivated.

Users cannot buy anything without providing credit card details, which act as identity verification. Users can have more than one credit card and it should be registered to the database to be used later. Credit card details should be encrypted before persisting to the database and only the last four characters should be returned to the user when retrieving the card details. A two way encryption algorithm can be used and a key should be stored in the database to provide extra security. Users cannot be unregistered from the system, but a request can be sent to the administrator by email.

7.3.8 Reports

A request will be sent towards the server, get some data from it, and create an appropriate XML file to use with an XSLT file.

7.3.9 Email Functionality

Different actions and requests will automatically send generated emails.

User registration should produce two emails. The first email shall be sent to the user stating they have successfully registered. The first email will include with an activation link, when the user clicks on the link their email will be activated for this account. A second email shall be sent to the administrator stating the details of the user.

7.3.10 Exception Handling

All exceptions should be handled, such that the server doesn't crash. When an exception occurs, an automated email should be sent towards the administrator. The email will consist of creation time and date, debug messages and the stack trace.

7.4 Stock Handling Function

To simulate a real stock market, dynamic stock values are required. User purchases do not contribute significantly to the stock values as they only represent a partial section of the stock-purchasing population. Therefore, stock values shall be updated from an external source. A threading approach is required to allow modulation of stocks in conjunction with the running application.
Additionally, stock data is not provided; stock values shall be realistically generated.

7.4.1 Generation

To aid in stock data generation, a set of company names are provided with their stock symbols. Consult Appendix 1 for this information.

Stock data, specifically stock value and share volume, must be generated for each of the companies. The data generation will be restricted to a range of values:

Field	Data Type	Restriction
Value	Number	range of 550 GBP to 5 GBP
Volume	Number	range of 2,000,000 – 1,000,000,000

Generation must not occur if the stock data already exists within the database to prevent data being duplicated or overwritten on server reset.

7.4.2 Modulation

Stock calculation and updating will need to occur separately from the server application. Below is a listing of how fields should be modified:

Field	Calculation	Modify
Open	=Value	when the stock market opens at 08:00
Close	=Value	when the stock market closes at 16:30
Gains	=Value-Open	when the Value is modified
Value	=Value*NumberGen()	every 5 minutes

The value and gain of a stock shall all be simultaneously modified, after every 5 minutes. NumberGen() should return a number that has some level of randomness to it to prevent predictability.

Appendix 1 - Companies List for Stock Data

Company	Symbol
Citigroup	C
HSBC	HSBC
Credit Suisse	CS
RBS	RBS
Calyon	CL
Threadneedle	TN
Aviva	AV
Morgan Stanley	MS
Unicredit	L
Caplin	CAP
Catlin	CLNGF
Rabobank	RABO
BNP	BNP
Wildnet	WN
QBE	QBE
Financial Times	FT
News International	NI
Abbey Santander	AS
Dresdner/ Commerz	DC
JP Morgan	JPM
Lloyds of London	LL
HBOS	HBOS
Nomura	N
ING	ING

Company	Symbol
Bank of New York	BNY
Soc Gen	SG
LCH	LCH
Daiwa	DA
Rothschild	RCD
Fortis	FO
Brewin Dolphin	BD
Schroders	SC
St St Bank	SSB
Bank of Tokyo	BTO
London Stock Exchange	LSE
Bank Of America	BOA
Tullet Prebon	TP
New Edge	NE
Fitch	FH
Thomson Reuters	TR
Chubb Insurance	CI
Tri Systems	TS
WTG	WTG
E&Y	EY
Ace Europe	AE
Beazily	BY
Sun	JAVA
Therion Systems	THSY