**Software Architecture Coursework Specification**

**Chapter 1 – Deciding an Architecture:**

Section 1 – Understanding the Project:

When starting work on a new project requiring the use of design architectures, multiple potential architectures should be compared and evaluated to ensure the use of one which best fits the scope and confines of the project. The context of this project is a development bid for a distributed system to be used as a retail management network, used to provide better coordination for business being done. The system is named “DE-Store” and must be able to handle; modification of item prices and deals, inventory/stock management, a customer loyalty card system, interconnectivity to an online finance payment portal for approving customer finance choices, and lastly, the tracking of customer activity for use in reports for store performance analysis.

Section 2 - Potential Architectures:

The first and most immediate architecture taken into consideration is three-tiered architecture. Three-tier architecture is a variant of Multi-tier architecture (also known as n-tier architecture), an architecture type which focuses on a modular approach to development. The key feature of the three-tier architecture is, as suggested by the name, it’s three tiers, also known as layers. The tiers of the architecture each encompass a different level of abstraction from the highest layer, that which is accessible by a user, so that the user is not directly interacting with lower layer information such as stored data.

In the most common format of the three-tiered architecture, the highest layer is the Presentation Layer. This is the layer which provides the direct interfacing capabilities of the application to a given user. These interfacing capabilities can take various forms such as a Web Page or a Graphical User Interface (GUI). The whole purpose of this layer is to communicate directly with the user of the application, both in terms of receiving commands and inputs from the user, as well as displaying outputs in an adequate format for the user to understand. This layer only communicates with the one layer below it, and essentially only serves the purpose of being a messenger between the user and the application, without actually doing any serious logic of its own.

The next layer down from the Presentation Layer is the Business Layer, also known as the Application Layer, Logic Layer or Middle Layer. This is the layer where the actual “hard work” of the application takes place, such as processing information and commands, making logical decisions, and performing calculations. This layer behaves as a “middleman” between the user of the application and the data contained within.

The lowest layer of the architecture is the Data Layer. This is the layer where all persisted data within the application gets stored and retrieved from. It contains a means of communicating with the Business Layer to allow its methods of managing, storing and retrieving data to be accessed by the Business Layer when desired. The data within the later is most commonly stored within a database, often saved within its own separate secure server for the sake of data security in the context of commercial applications.

Key Advantages of the three-tiered architecture;

* Easily managed and worked on by multiple different developers/teams.
* Centralised maintenance for many end-users.
* Allows the application to easily be rescaled to fit different levels of engagement and usage.
* Supports Multi-threading to improve performance and reduce network traffic build-up.
* Modular approach allows a stronger focus on security in the most vital points within the application.
* Lightweight for client use, meaning it’s accessible for a wider range of hardware, especially since it can often take an ultra-light client-side format such as a website in many cases.

Key Disadvantages of the three-tiered architecture;

* More complex to develop than many of the more basic architectures using singular or two-tiered approaches.
* Will be more commercially expensive to implement due to its more complex nature, though arguably for good reason.
* Can prove difficult to test due to lack of available tools and resources.

The second architecture of consideration is the Client/Server architecture. This is also a variant of the Multi-Tiered architecture format, this time taking a two-tiered approach. The key focus of this architecture is having a dedicated application for the client, aka the user/consumer of the application. A second application is then being run off of a server, which contains the appropriate persisted data and logical capabilities to be utilised by a connected client. Within the context of this project, the Client/Server architecture will refer to the Thick/Fat-client model, as it’s the most common and appropriate within the context of modern commercial Client/Server systems where the intended capabilities of the Client are known in advance. This model also reduces the load and requirements of the sever by offloading part of the “hard work” on to the client, which is effective when multiple individual concurrent users are each trying to utilise the application.

Within this version of a Client/Server system, the client side of the application manages the presentation and application processing. The presentation aspect includes the communication of information to a client, such as text or images, as well as taking in input from the client to then be processed by the application. The processing aspect involves managing input from the user, possibly validating said input, then transforming it to a form to be sent to and further processed by the server. This aspect also involves receiving information from the server then converting it to an adequate form to be presented back to the user of the application.

The server side of the application manages its own application processing as well as data management. The application process aspect covers much of the same ground as the same aspect within the client side of the application, with the likely additions of security aspects, in attempt to avoid potential cyber-attacks aimed towards the server, especially if it manages the transferal of money or products. The data management aspect of the server involves the storage, persistence and retrieval of information required by the server, often to be used by both the sever and client.

Key advantages of Client/Server architecture;

* Simplifies design to expediate development.
* Very effective in situations where the number of clients vastly outnumbers the number of servers.
* Server components are easily replicated for multiple servers.
* Separates some aspects for modular design approach, allowing certain elements of the application to be easily developed/modified my multiple different developers/teams.
* Easy and simple method of creating a distributed system/environment.

Key disadvantages of Client/Server architecture;

* Security can be difficult to properly integrate in the most necessary areas.
* Clients are on the difficult side to manage, maintain and upgrade due to Presentation and Business/Processing aspects being stuck together.
* Somewhat limited client hardware choices as processing is done before sending information to the server.
* Potential issues when running on devices which it isn’t directly developed for and tested on.

Section 3 – Selecting an Architecture:

Make choice and explain why the choice was made

**Chapter 2 – Designing the System:**

Section 1 - TitleHere:

ContentHere

**Chapter 3 – Implementing the Prototype:**

Section 1 - TitleHere:

ContentHere

**Chapter 4 – Evaluating the Prototype:**

Section 1 - TitleHere:

ContentHere