Trade Net Trade & Portfolio Management Software:

Software Architecture Description

Hunter Harrison hh342

Lauren Taylor lft23

Eleashia Hodges ejh122

Frank Haecker fph7

CSE 4233/6233

Fall 2014

1. Architectural Documentation (Document Control Information)

1.1 *Date of issue and status*

Issued on November 25th 2014, and is Complete

1.2 *Issuing organization*

Trade Net Brokerage and Financial Services

1.3 *Change history*

Date	Version	Description	Author
11/29/14	1.0	Initial Draft	Lauren Taylor, Hunter Harrison
12/4/14	1.1	Iteration	Eleashia Hodges, Lauren Taylor
12/5/14	1.2	Further Iteration	Frank Haecker, Eleashia Hodges
12/8/14	1.3	Content Filling	Hunter Harrison, Frank Haecker
12/10/14	1.4	Final Draft	All members

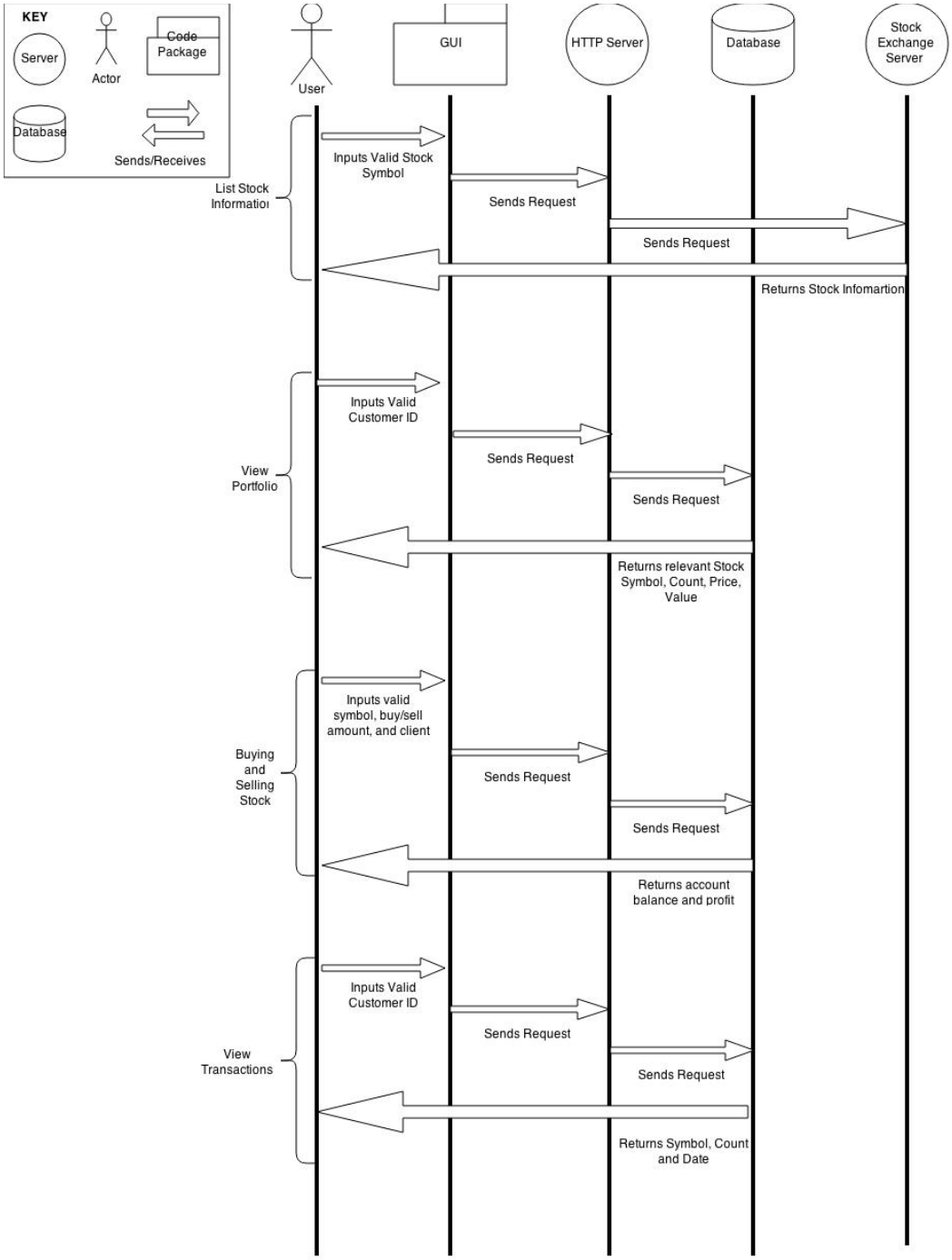
1.4 *Summary (System Overview)*

This document is an architecture-description document for the Trade Net software application. The purpose of this document is to provide an a comprehensive view of all the architecture designed for the Professional Practice Assignments 1-4.

These views largely consists of the trading application, external data used by the trading application, user access of the trading application, the account management server, and the databases where this information is stored.

The Uses Style and the Service Oriented Style give the most clear overhead view of the project and should be referenced for an overview of the system. The context diagram (we chose a use case diagram) will show how the system is used.

1.5 *Context*



There are four scenarios in which have been described above in the use case model. The first is List Stock Information. The user inputs a valid stock symbol into the GUI.

The GUI then sends a request to the HTTP server, the server then sends the request to the Stock Exchange Server. The Stock exchange server returns the relevant information to the user.

The second situation depicted above is the View Portfolio scenario. The user first inputs a valid user ID into the GUI. The GUI sends a request to the HTTP Server. The HTTP Server then passes the request to the database. The Database then returns the relevant stock symbol, count, price, and value.

The third situation depicted above is the Buy/Sell scenario. The user first inputs a valid user ID, valid stock symbol, and buy/sell amount into the GUI. The GUI sends a request to the HTTP Server. The HTTP Server then passes the request to the database. The Database then returns account balance and profit to the user.

The fourth situation depicted above is the View Transactions scenario. The user first inputs a valid user ID into the GUI. The GUI sends a request to the HTTP Server. The HTTP Server then passes the request to the database. The Database then returns the relevant stock symbol, count, and date.

Use cases are used during requirements elicitation to model external behaviors of the system.This view would be useful for the maintainers of the system. This will be helpful by giving the maintainers an idea of how the outward system should work to detect any changes. This view will also be useful for the infrastructure support panel. This view will help them determine which elements should be accessible to the user and determine the risks associated with each point of entry.

1.6 *Glossary*

Stakeholder- an individual, organization,or team that has some sort of interest or “stake” in the projects completion.

App- Shorthand of application, which is a program that is being used either by a user or another program.

ERD- Entity Relationship Diagram. A diagram of the entities of a database, the attributes within them, and the relationships between those entities.

External- Some element outside the control of Tradenet.

1.7 *References*

Byron Williams Class slides for the Fall semester of 2014
IEEE International Standard given by Byron Williams for Fall semester of 2014

1.8 *Acknowledgements*

We would like to thank Byron Williams for helping in the completion of this project.

2. Identification of Stakeholders and Concerns

2.1 Stakeholders

2.1.1 Project Manager

The project manager leads a team of workers to completing a software project.

2.1.2 Designers of other systems

Designers of other systems can be helpful with the web designs and other possible designs.

2.1.3 Development Team

A development team is a team of workers who has the responsibility of developing source code for the software project.

2.1.4 Maintainers

Maintainers have the ability to keep the system's external functions working.

2.1.5 Customers

Customers will be the potential buyers of the software product.

2.1.6 End-Users

The end-users are the people who actually uses the software product.

2.1.7 Analysts

Analysts are the people who conducts analysis on a particular product.

2.1.8 Infrastructure Support Personnel

Infrastructure personnel helps with problems that may occur in building the product.

2.1.9 Future Architects

Future architects are people who will be interested in learning about the architecture views of the product.

2.2 Concerns

2.2.1 Project Manager's Concern

The Project manager will be concerned with the schedule of the project, making sure it will be done within time and budget. They will also be concerned about the overall purpose and constraints of the product. Also, they will want to know about other systems interactions with the product.

2.2.2 Designer's Concerns

Designers' concerns will include what set of operations are required and or provided for the product and the protocols that must be taken.

2.2.3 Development Team's Concerns

A development team will have concerns dealing with the constraints of how to do their jobs properly.

2.2.4 *Maintainers' Concerns*

Maintainers ' concerns are with the areas that change will affect. They also have concerns about how to do their job effectively within a software project.

2.2.5 *Customers' Concerns*

A customer will have concerns with the coat and progress of the product. Also, they will have concerns on whether the product will be designed correctly or not.

2.2.6 *End-Users' Concerns*

The end-users' concerns will be how to use the system.

2.2.7 *Analysts' Concerns*

An analyst will be concerned with the system, itself. The will want to know whether the system meet the required quality attributes.

2.2.8 *Infrastructure Support Personnel's Concerns*

The infrastructure support personnels are concerned about helping the system administrators.

2.2.9 *Future Architects' Concerns*

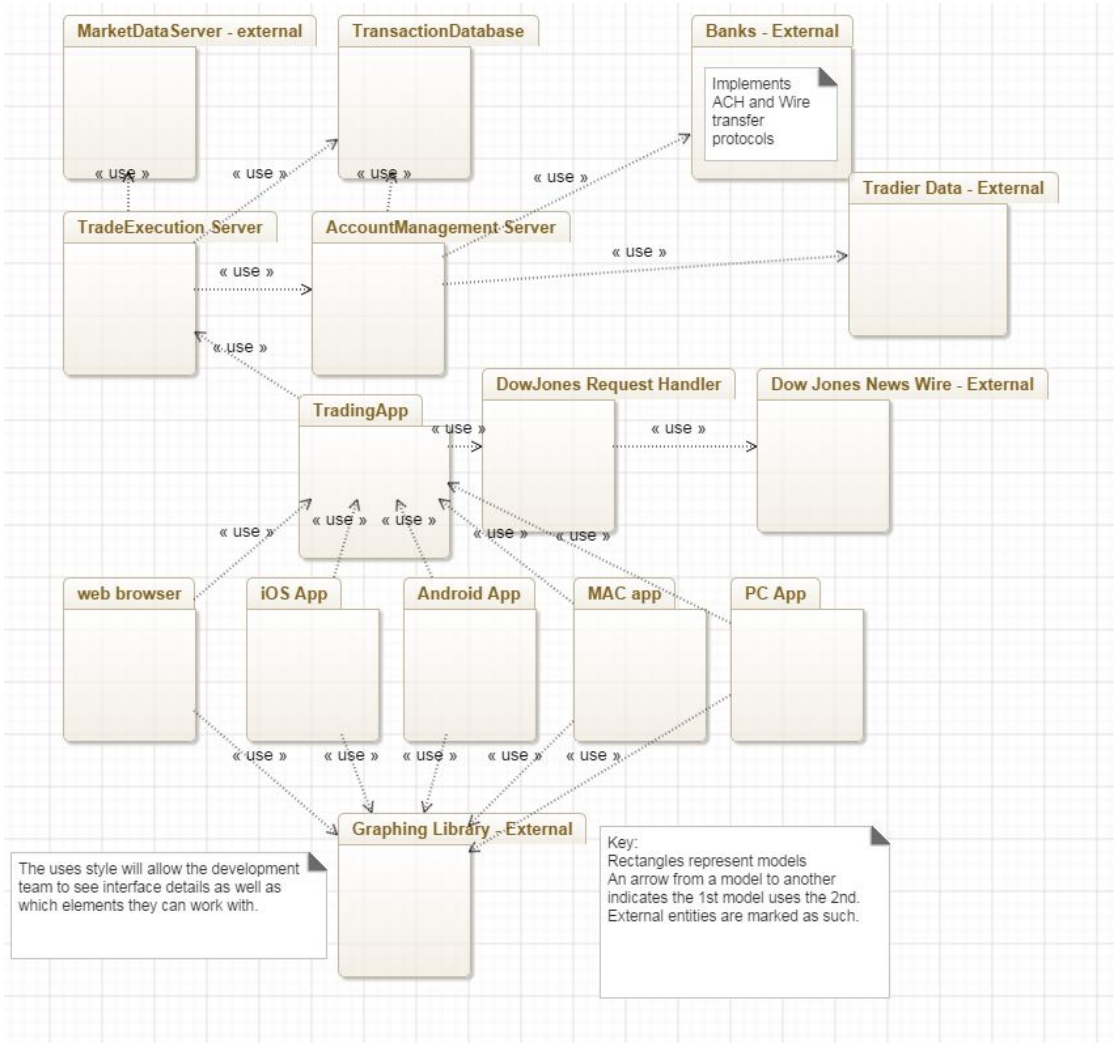
Future architects' concerns deals with the rationale and design information of the system's architecture and the aspects of it.

3. Architectural views

3.1 Logical views

This section of the architecture document covers the logical views of architecture that we created throughout the profession practice assignments. The first view is the Uses Style View, followed by Layered Style View , the Aspects Style View, and finally the Data Model Style View. Each view has the element catalog listed below it. If appropriate is contains the relationship between them and how it maps to services required in the SRS.

3.1.1 Uses Style View



Element Name	**Uses**	**Service Mapping to SRS**
TradingApp	DowJones Request Handler, TradeExecution Server	Allows access to the markets via several different means
DowJones Request Handler	Dow Jones News Wire - External	Requests Access to Dow Jones Newswire
web browser	TradingApp, Graphing Library- External	Allows user to access on web browser
iOS App	TradingApp, Graphing Library- External	Allows user access on iOS application

Android App	TradingApp, Graphing Library- External	Allows user access on Android Application
MAC	TradingApp, Graphing Library - External	Allows user access on MAC computers
PC App	Trading App, Graphing Library-External	Allows user access on PC computers
Graphing Library - External	External	Allows graphing utility for users
Dow Jones News Wire - External	External	Allows access to Dow Jones Newswire Information
TradeExecution Server	AccountManagement Server, MarketDataServer-External, TransactionDatabase	Manages requests between users and the real time market data
MarketData Server - External	External	Stores the external market data for access
TransactionDatabase	Is only used	Stores transactions
AccountManagement Server	Banks - External, Tradier Data - External	Integrates with external Tradier and bank data
Banks - External	External	Stores information for access on banks
Tradier Data - External	External	Stores the Trade data and is updated with every interaction

Trading application - allows access to the markets via a web browser, a standalone (Mac and PC) application, and both iOS and Android mobile apps.

The client apps enable stock charting, technical analysis for strategy development, access to company fundamental data (e.g., financial reports), and trade execution.

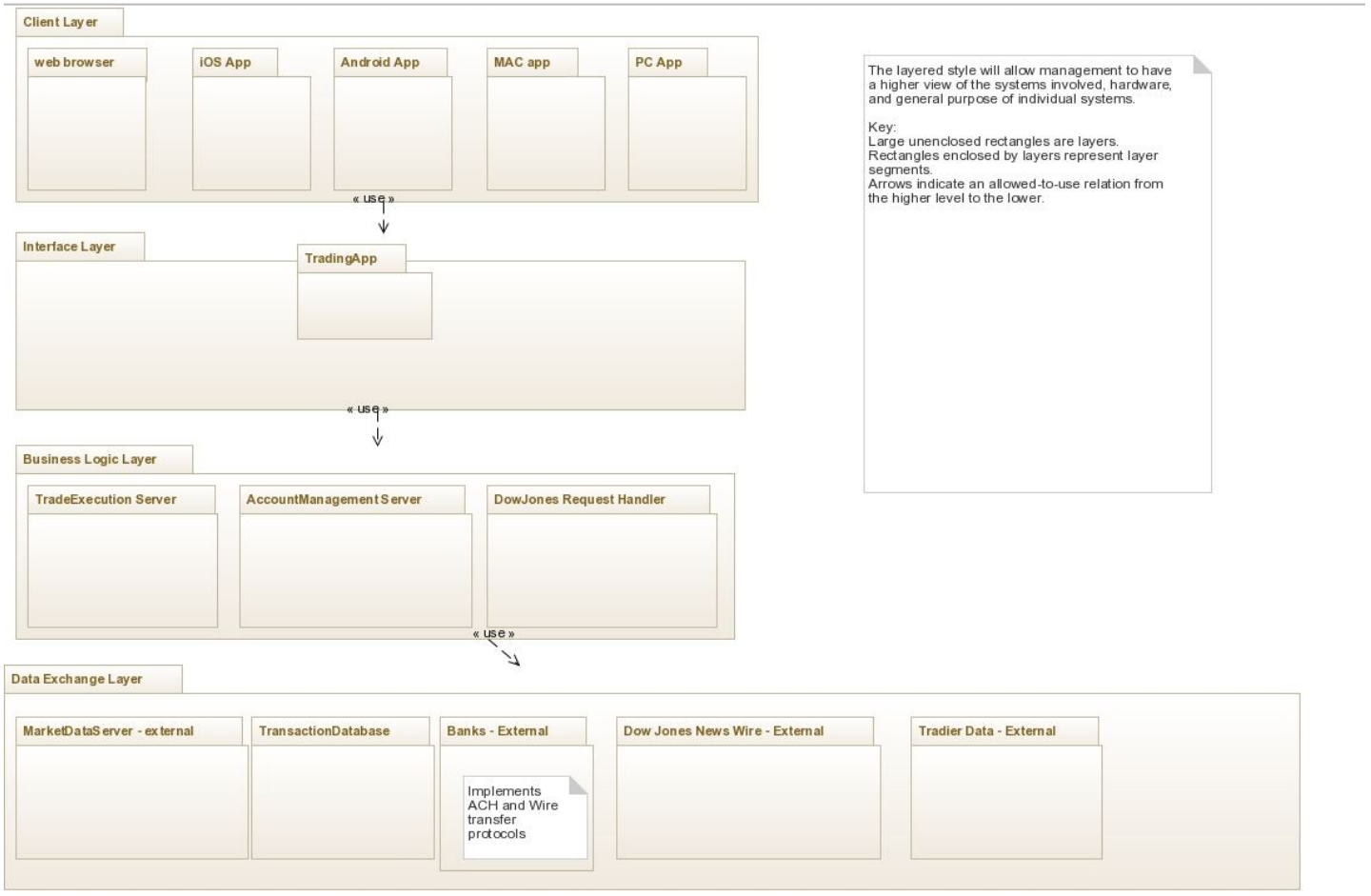
In addition to the client apps, Trade Net has to extend its existing legacy Account Management (AM) server, which will integrate with a new Trade Execution server.
The Trade Execution (TE) server will access real-time market data via a direct stream to the stock exchanges' market data feed servers to obtain share price information and the best stock bid and ask prices for its customers.
The TE server will also execute trades on the exchanges after verifying the availability of customer funds with the AM server. The TM server will authenticate customer login info using the AM server. Both servers will store their data in a central database. This data includes historical market data, all transactions, and logs required by federal regulators.

Each client application relies on a third party charting and financial graphing library to generate charts and integrate charting technical analysis.
The desktop client applications also allow direct access the Dow Jones newswires for real-time financial news updates.

The AM server facilitates cash transfers using standard banking ACH and Wire Transfer protocols and can transfer funds with any other US-based bank implementing the protocols. Securing customer data and transactions is vital to company success. All communication between nodes should be conducted using encrypted transmissions and verified, open authentication protocols.

The Uses Style View very directly relates to the Use View Packet Class diagram submitted with professional practice number three. The Use View Packet covers all three class that we built for the code, and each part of the Uses Style View could easily fit into any of those three classes.

3.1.2 Layered Style View

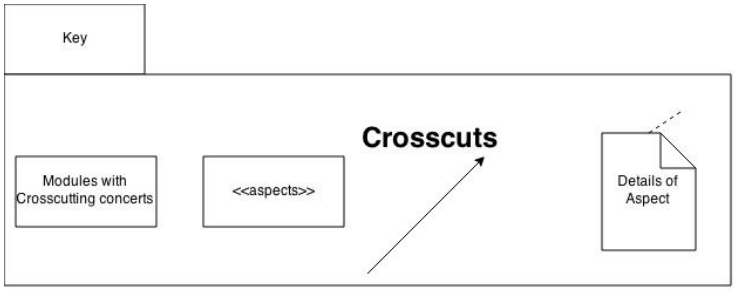


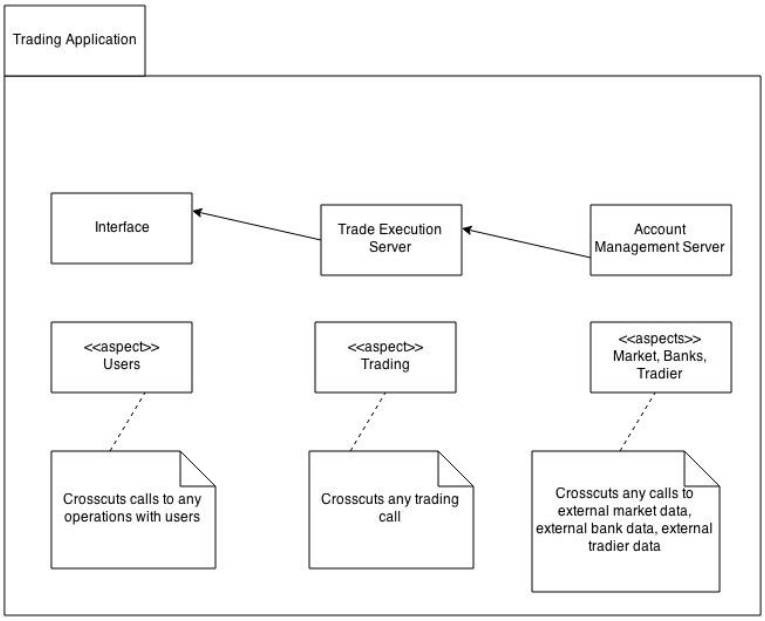
Element Name	**Uses**	**Service Mapping to SRS**
Client Layer	Interface Layer	Allows Users to access the program through a number of means
Interface Layer	Business Logic Layer	Allows the User to access an interface to us the application
Business Logic Layer	Data Exchange Layer	Enforces business rules and attempts to access data that is authorized

Data Exchange layer	None	Access stored data and return what is requested

The layered style will allow management to have a higher view of the systems involved, hardware, and general purpose of individual systems. It is easy to parse the four simple layers of the program. This view was choses to allow a very simple overview of the program, simple enough for anyone to understand.

3.1.3 Aspects Style View

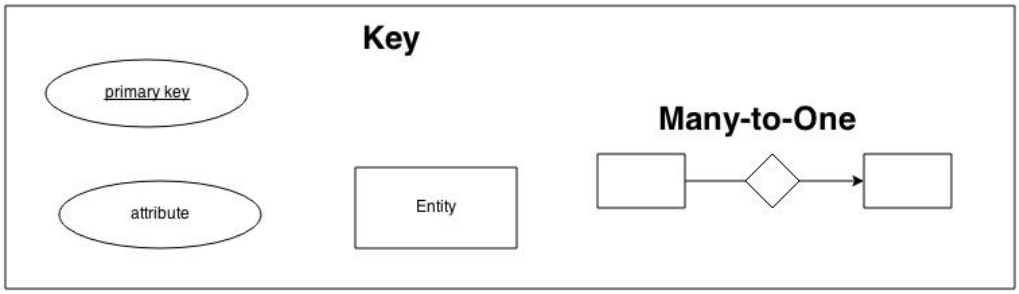


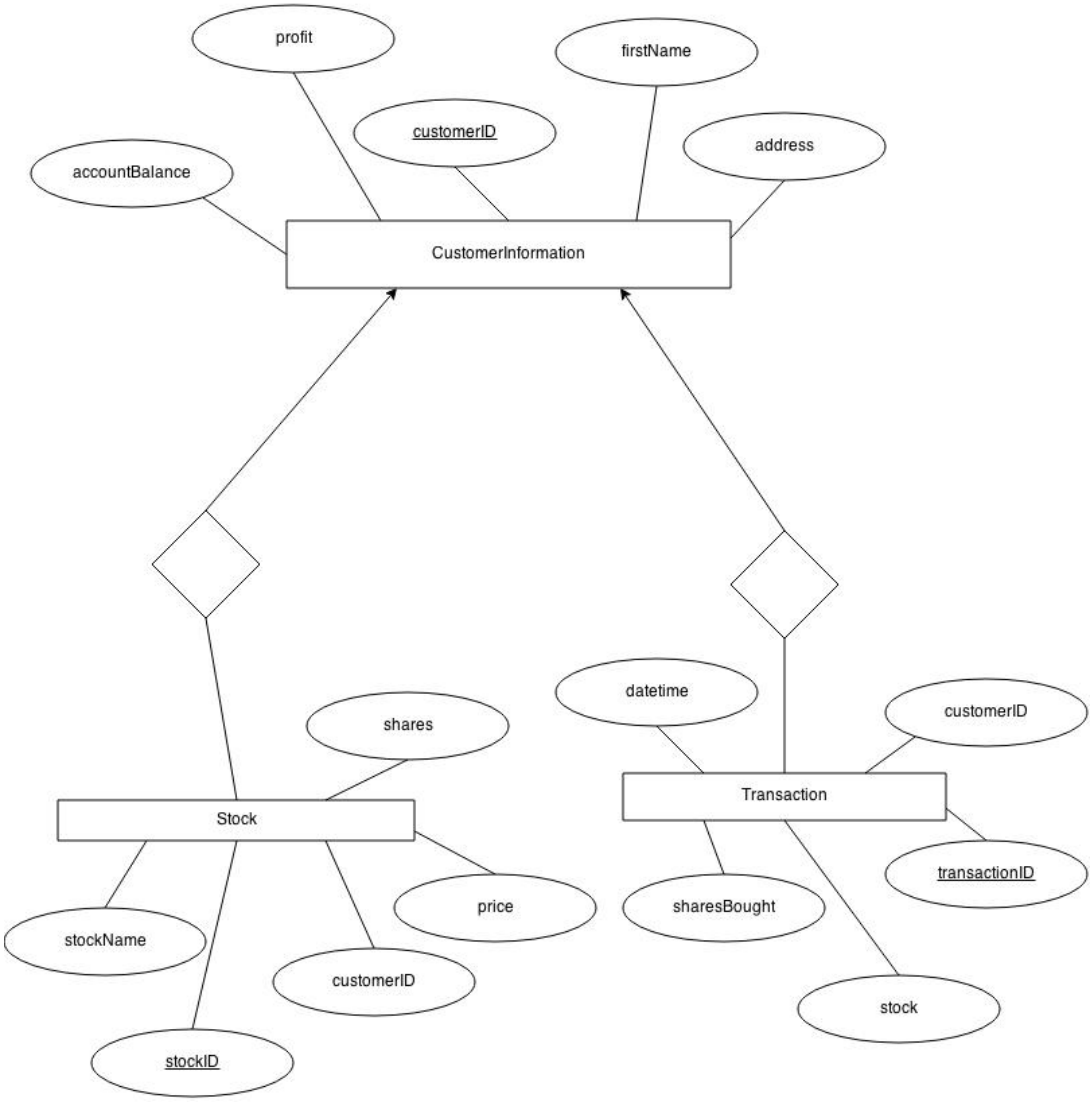


Element Name	Service Mapping to SRS
Interface	Handles all interactions with the users
Trade Execution Server	Handles all interactions with trading calls
Account Management Server	Handles all interactions with external market data, bank data, external tradier data

This is an aspect style model for the trading application. We decided on this view so it it would be easy to isolate where cross cutting concerns are occurring. It will be useful for the development team so they can see what parts of the design will have crosscutting concerns for development. It will also be useful for analyst, because they will be able to tell if cross cutting concerns are being met according to the design.

3.1.4 Data Model Style View





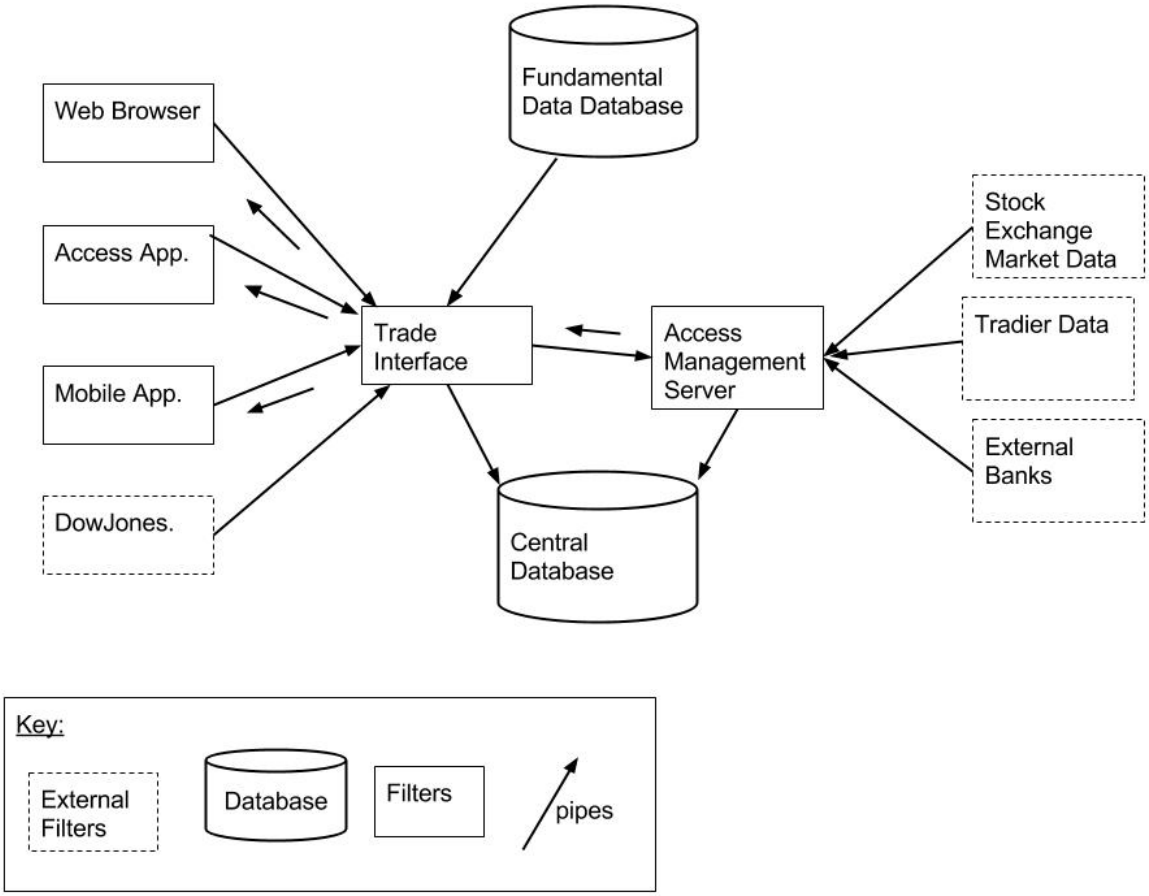
Element Name	Uses	Service Mapping to SRS
CustomerInformation	Houses foreign keys from Stock and Transaction	Directly maps to CustomerInformation table in database
Stock	Many to one relationship with CustomerInformation	Directly maps to Stock table in database
Transaction	Many to one relationship with CustomerInformation	Directly maps to Transaction table in database

This is a Data Model Style View in the form of an ERD (Entity Relationship Diagram). This view is useful for mapping out the tables of databases and seeing what attributes are in each of those tables. It is also useful for showing what the relationships between the tables are. We created this view in order to help map out the database.

3.2 Run-time Structure and Dynamic Views

This section of the document begins with the Pipe and Filter View, followed by the Tiered Client-Server View, and finally the Service Oriented Style. The main point of this section is to view architecture that shows what processes will be in the system. It will also explain where the modules are assigned to each process. The four processes that they can be assigned to will be Client, Server, External, and Database. The Service Oriented Styles servers as the best overhead view of the architecture in this section.

3.2.1 *Pipe and Filter View*

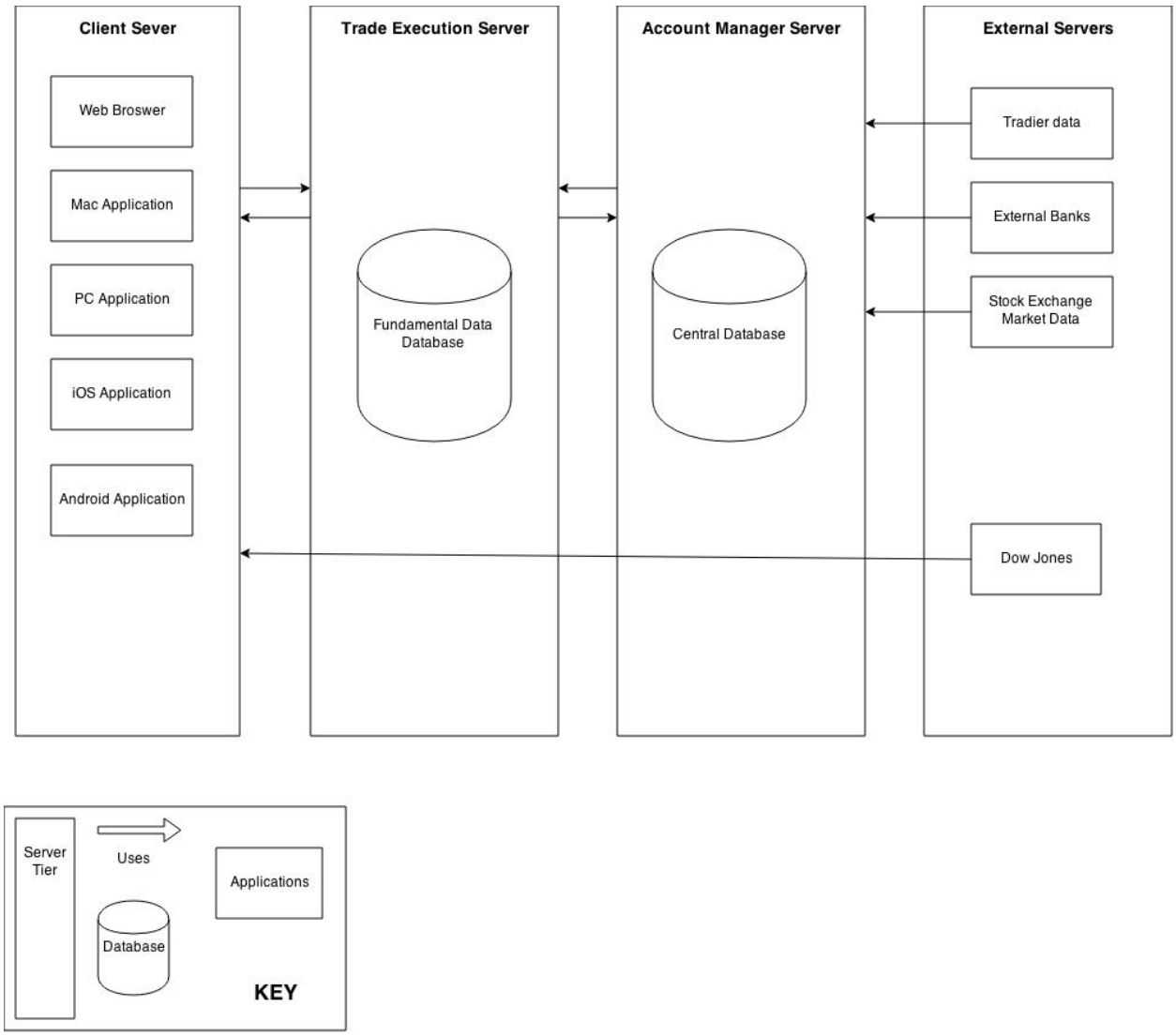


Element Name	**Uses**	**Process Assigned To**
Web Browser	Http/https to Interface	Client
Standalone Application	Http/https to Interface	Client
Mobile Application	Http/https to Interface	Client
Interface	Dow Jones, Trade Execution Server, Fundamental Data Database	Client
Dow Jones	External	External

Fundamental Data Database	Stores data for Interface	Database
Trade Execution Server	Central Database, Account Management Server	Server
Central Database	Stores data for Trade Execution Server and Account Management Server	Database
Account Management Server	Tradier Data, Stock Exchange Market Data, External Banks	Server
Tradier Data	External	External
Stock Exchange Market Data	External	External

This style illustrates the overall function performance. It shows the inputs and outputs that will be sent back and forward using pipes and filters. The stakeholders that will be concerned with this style is development team and maintainers because they care about how everything interacts with one another; basically, they will want to know the general idea behind the system.

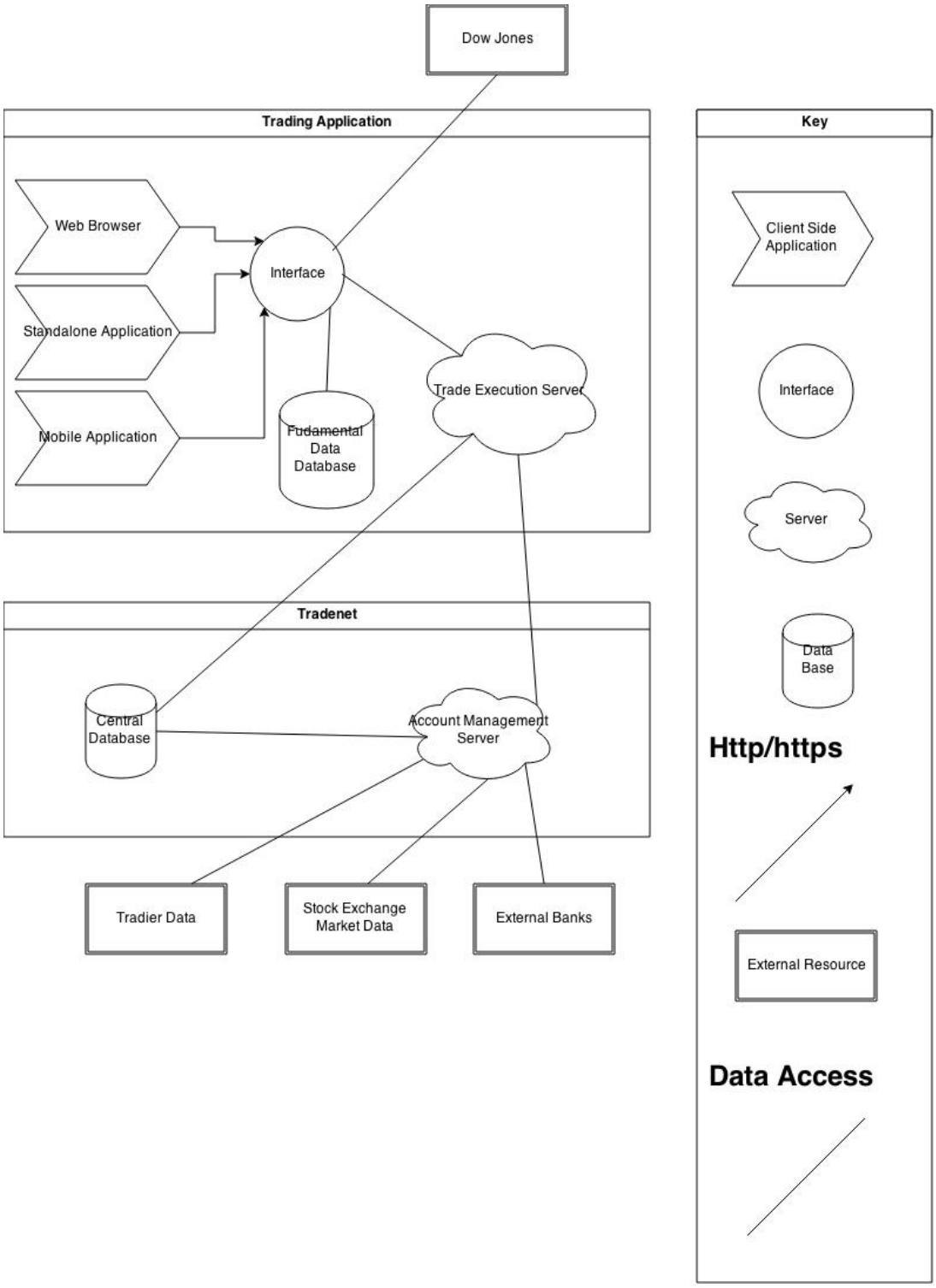
3.2.2 Tiered Client-Server View



Element Name	Uses	Process Assigned To
Client Server	Http/https to Trade Management Server	Client
Trade Execution Server	Http/https to Client Server and Account Manager Server	Database
Account Management Server	Http/https to Trade Execution server and External Servers	Database
Tradier Data	Http/https to Account Management Server	Exeternal
Stock Exchange Market Data	Http/https to Account Management Server	External
Dow Jones	Http/https to Client Server	External
External Banks	Http/https to Account Management Server	External
Central Database	Stores data for Trade Execution Server and Account Management Server	Database
Fundamental Database	Stores data for Interface	Database
Web Browser	Creates the interface	Client
Mac Application	Creates the interface	Client
PC Application	Creates the interface	Client
iOS application	Creates the interface	Client
Android Application	Creates the interface	Client

We chose client server because the requirements mention at least two servers that will be invoked by user. (Only one will actually interface directly the client) This will address the development team’s concerns for the general idea behind the system and the elements they have to interface with. Also, the maintainers will want to see the this view. It will be helpful for maintaining the system as they use the systems architecture as a starting point.

3.2.3 Service Oriented Style View



Element Name	Uses	Process Assigned To
Web Browser	Http/https to Interface	Client
Standalone Application	Http/https to Interface	Client
Mobile Application	Http/https to Interface	Client
Interface	Dow Jones, Trade Execution Server, Fundamental Data Database	Client
Dow Jones	External	External
Fundamental Data Database	Stores data for Interface	Database
Trade Execution Server	Central Database, Account Management Server	Server
Central Database	Stores data for Trade Execution Server and Account Management Server	Database
Account Management Server	Tradier Data, Stock Exchange Market Data, External Banks	Server
Tradier Data	External	External
Stock Exchange Market Data	External	External
External Banks	External	External

The Trading Application square represents the actual application, while Tradenet represents the company in general. The data access between Trade Execution Server and Account management serve also handles the authentication. All transactions

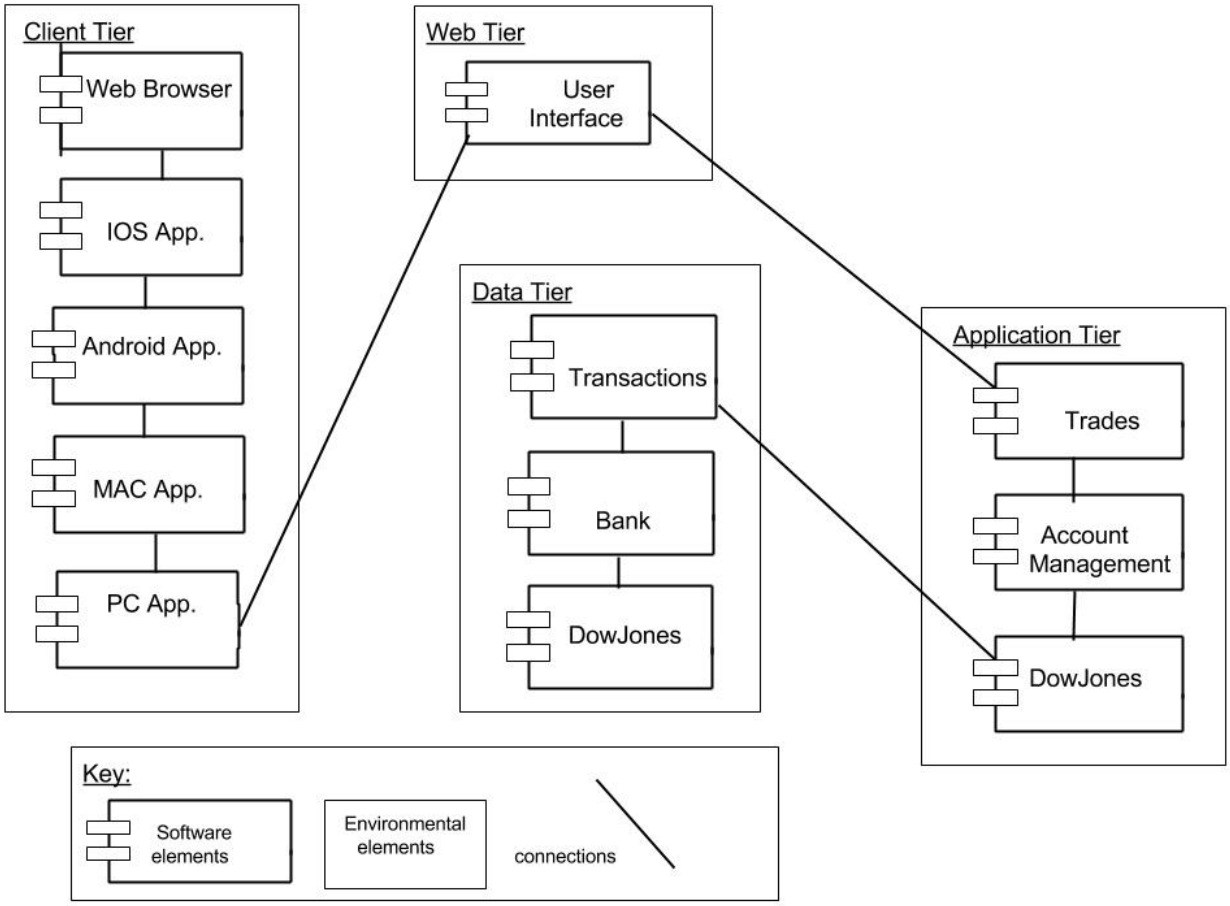
between external resources are encrypted. The Central Database includes the account management server database.

This view will be useful for the project manager, because it will allow them to see the overall structure of each service at a high level. It will also be good for analyst as a key part to see where services take place and see if they correspond accurately.

This view easily maps to the Use View Packet submitted with professional practice number three. This view servers as a good overhead view of the project, so the individual elements can be mapped to the ButtonHandler,GetClient, or DisplayPanel classes.

3.3 Allocation Views

3.3.1 *Deployment View*

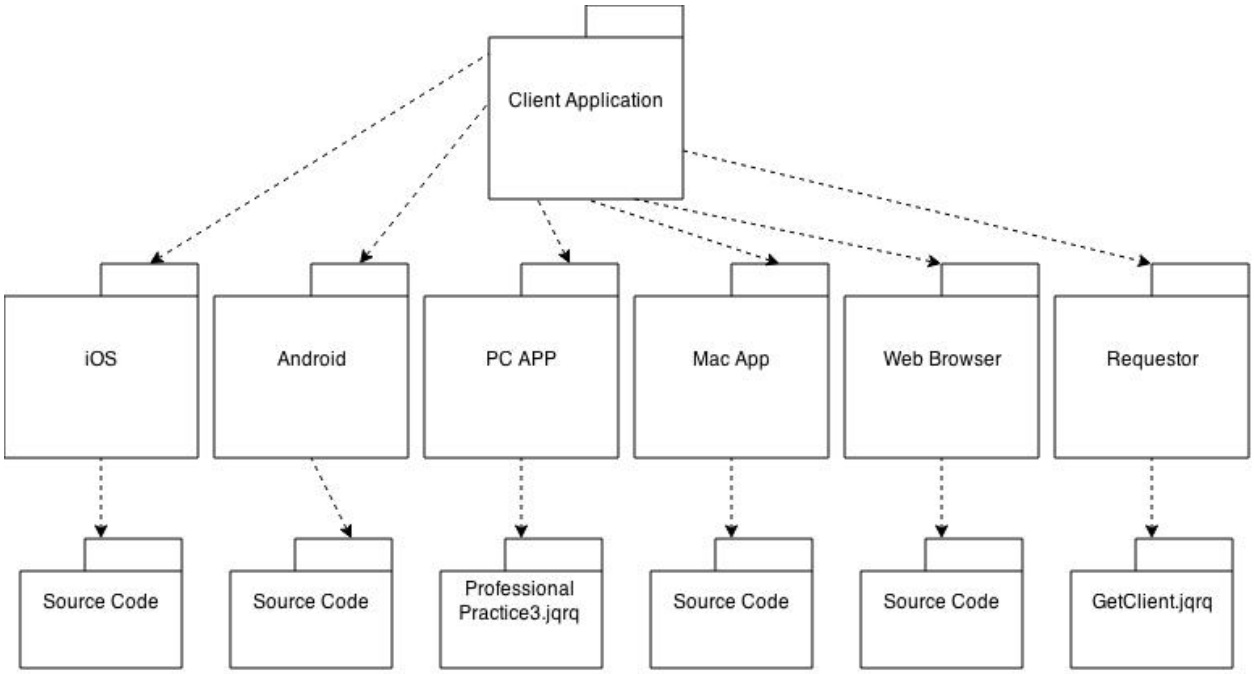


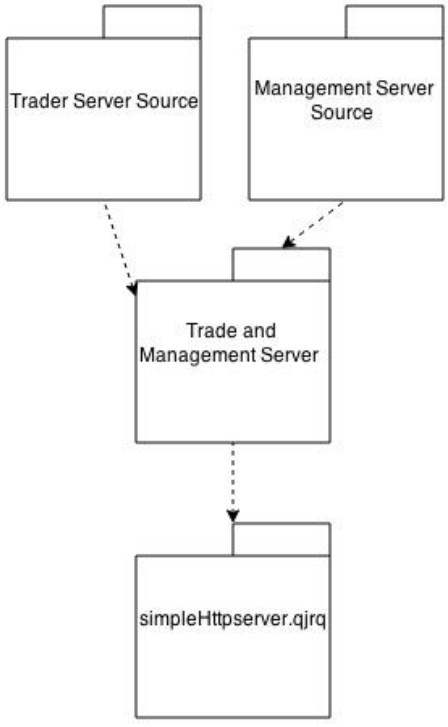
Element Name	**Description**	**Type**
Web Browser	Allows access from a web page	Client
IOS App.	Allows access from and IOS application	Client
Android App.	Allows access from an Android application	Client

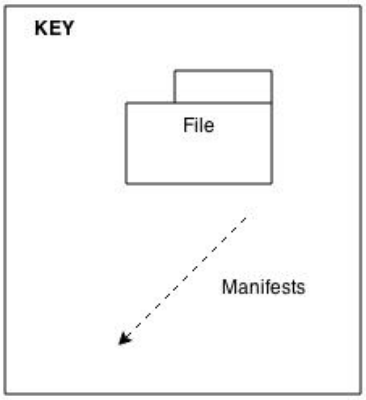
MAC App.	Allows access from a macintosh application	Client
PC App.	Allows access from a Windows Personal Computer application	Client
User Interface	The interface that all users access the system with	Server, Database
Trades	Allows the user to complete trades then stores info	Server,Database
Account Management	Allows the user to edit information to their account then stores info	Sever, Database
DowJones	Allows the user to get info from DowJones network	Server
Transactions	Stores the users transactions	Database
Bank	Stores the external bank data for access	External
DowJones	Holds DowJones network data for access	External

This style allocates the elements from connector and component view to a computing platform. Shows how the environmental and software elements interact together. The stakeholders that will be concerned with this style is development team and maintainers because they care about how everything interacts with one another; basically, they will want to know the general idea behind the system.

3.3.2 *Implementation View*


Client Application
iOS
Android
PC APP
Mac App
Web Browser
Requestor
Source Code
Source Code
Professional Practice3.jqrq
Source Code
Source Code
GetClient.jqrq


Trader Server Source
Management Server Source
Trade and Management Server
simpleHttpserver.qjrq


KEY
File
Manifests

Element Name	Description	Type
Client Application	GUI that the end user sees	Client
iOS Application	Manifested from the Client Application	Client
Android Application	Manifested from the Client Application	Client
Web Browser	Manifested from the Client Application	Client
PC Application	Manifested from the Client Application	Client
Mac Application	Manifested from the Client Application	Client
Requestor	Manifested from the Client Application	Client
iOS Application Source Code	Manifested from the iOS application	Client
Android Application Source Code	Manifested from the Android Application	Client
Web Browser Source Code	Manifested from the Web Browser Package	Client
ProfessionalPractice3.jqrq	Manifested from the PC Application	Client
Mac App Source Code	Manifested from the Mac Application	Client
GetClient.jqrq	Manifested from the Requestor package	Client
Tradier Server Source	Package that contains the source for the Tradier Server	External
Management Server Source	Package that contains the source for the Management server	Database

Trade and Management Server	Manifested from a combination of The Tradier Server Source and Management Server Source packages.	Database
simplehttpserver.jqrq	Manifested from the Trade and Management Server Package	Database

This style is a mapping of models to development infrastructure. We need to know which files will be installed on target machines. This view is good for configuration and version controlling. This system will address the development team, they will be concerned with the mapping of the system and this view will address this concern. This will also be beneficial for the infrastructure support panel. They need to see which parts of the infrastructure are accessible and might be vulnerable to security attacks.

The Implementation View relates to the Use View Packet Class diagram submitted with professional practice number three. Element comes directly from the code, so it could very easily fit into one of the three classes that we built into the view packet.

3.3.3 *Deployment Structure*

Equipment:

There will only be a few equipment items necessary for the deployment of this project. There will need to be a centralized database to store data, two servers to handle requests and access data, and there will also need to be connections to four outside sources. Of course, the various users will also need to have one of the acceptable devices to access the application.

Location:

It would be wise to place the database in a centralized, safe, and dry location. It would also be a good idea to have a backup database located somewhere else that is constantly updating with the main database. This way if something were to happen to one database, there would be another to step up in its place. The two servers should be placed within the same building as the databases, but in a different room. It would also be wise to have two backup servers in the secondary location. It does not matter where the external data is stored as long as a there is a secure connection to that data.

Communication Capabilities:

Communications between these different pieces of equipment will take place over secure and encrypted connections over the internet.

Processes Assigned:

Databases will be assigned the database processes. Servers will be assigned the server processes. Individual clients will be assigned the client processes. All external items will be assigned the external processes.

Distribution to Development Team:

For this project we recommend having two teams. Please reference the Service Oriented Style View at 3.2.3 in this document. One team should be assigned the "Trading Application" portion of the architecture. The second team should be assigned the "Tradenet" portion of the architecture. This distribution should allow the project to move at a speedy clip. However, it is very important that communication is constant between the two teams in order to maintain compatibility.

4. Consistency among Architectural Views (Mapping Between Views)

We believe that this architecture is very consistent and the views should be able to build off of each other. There are only a few inconsistencies worth noting.
The "transaction database" in the Uses Style View is rolled into the "Central Database" in the Service Oriented Style View.
The "Stock Exchange Market Data" form the Uses Style View goes through the "Trade Execution Server" in the Uses Style View.
The "Transaction Database" in Uses Style View is now rolled into "Tradier Data" for the Service Oriented Style View.

5. Architectural Rationale

We decided on these architecture view, because we believed that each of these views were the most helpful for creating this project. Specific rationale is given under each of the views, so this section will be more of an overview of our rationale.

We generally put a lot of thought into our views before we chose one, so it was not often that we had to resort to an alternative. However, there is one example where we tried to create a Data Model Style View before professional practice number four. This meant

that we had not yet created the database, so it was difficult to know exactly what attributes and entities there were. We had to to drop this view until profession practice number four was assigned. This is a unique scenario, because all of our other models were completed as we decided.

The Context (Use Case) diagram is the view that makes it easy to see a use scenario for the project, and is good starting off point. The document then moves on to the Logical Views section. This section is largely designed to see how the overall project is designed. Each view gives an understanding of what is happening without going into too much specific detail. These views will be perfect for stakeholders trying to get a better understanding of the project, as well as see how all the pieces fit together.

The Run-time Structure and Dynamic views are more helpful for seeing where the data is going and coming from. It allows stakeholders to see where data is going, and through what means it is moving. This will be useful for a variety of stakeholders, but especially useful for analyst and developers. This more detailed view allows for a better analysis of how the system is working.

The Allocation Views are primarily for the mapping of internal elements to their technical environment. The two views that we chose will be very useful developers to see what technology goes where. This allows developers to focus on their specific technology and then see at a glance where it belongs. It can also be useful for any stakeholders who have planning responsibilities. They can much more easily assign work if they can see where the technology belongs.