Software Architecture & Design

Final Project: Milestone 1

Team 1

Hannah Church (hds109)

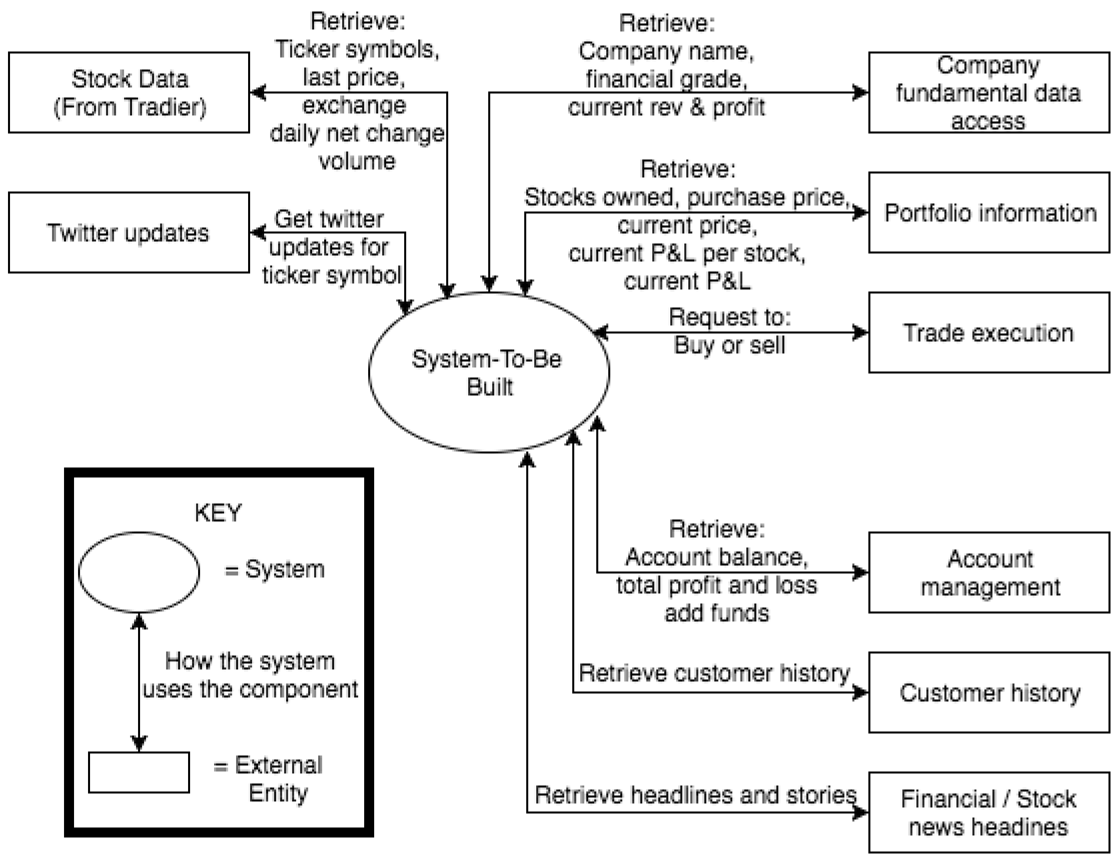
JJ Kemp jhk114

Ryan Sweeney ras536

Steven Lightle sel266

Jimmy Bailey jb2188

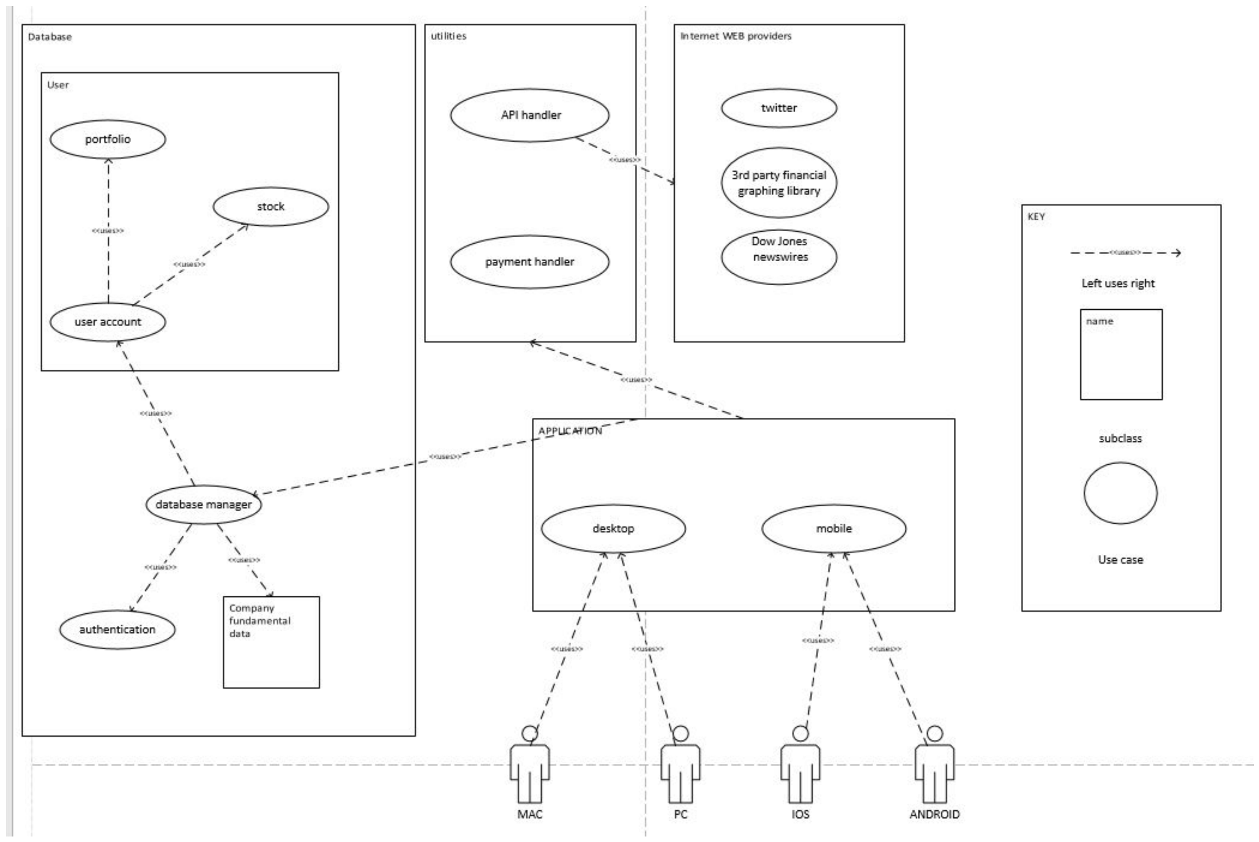
Context Diagram


Stock Data (From Tradier)
Retrieve: Ticker symbols, last price, exchange daily net change volume
Twitter updates
Get twitter updates for ticker symbol
Retrieve: Company name, financial grade, current rev & profit
Company fundamental data access
Retrieve: Stocks owned, purchase price, current price, current P&L per stock, current P&L
Portfolio information
System-To-Be Built
Request to: Buy or sell
Trade execution
Retrieve: Account balance, total profit and loss add funds
Account management
Retrieve customer history
Customer history
Retrieve headlines and stories
Financial / Stock news headines
KEY
= System
How the system uses the component
= External Entity

The system context view is a high-level view of the system. It shows how the system uses the external entities and how they are modularly separated. It is needed because it depicts the entities outside the system. Anything that we do not directly write with our code is an external entity, and must be interacted with as such. The relationship lines show the commands and stimuli that are processed and produced. We will code the relationship lines to interact with these entities. j6sFor example, we have no control over what is on Twitter, so we must interact with Twitter by pulling updates and using them in our system.

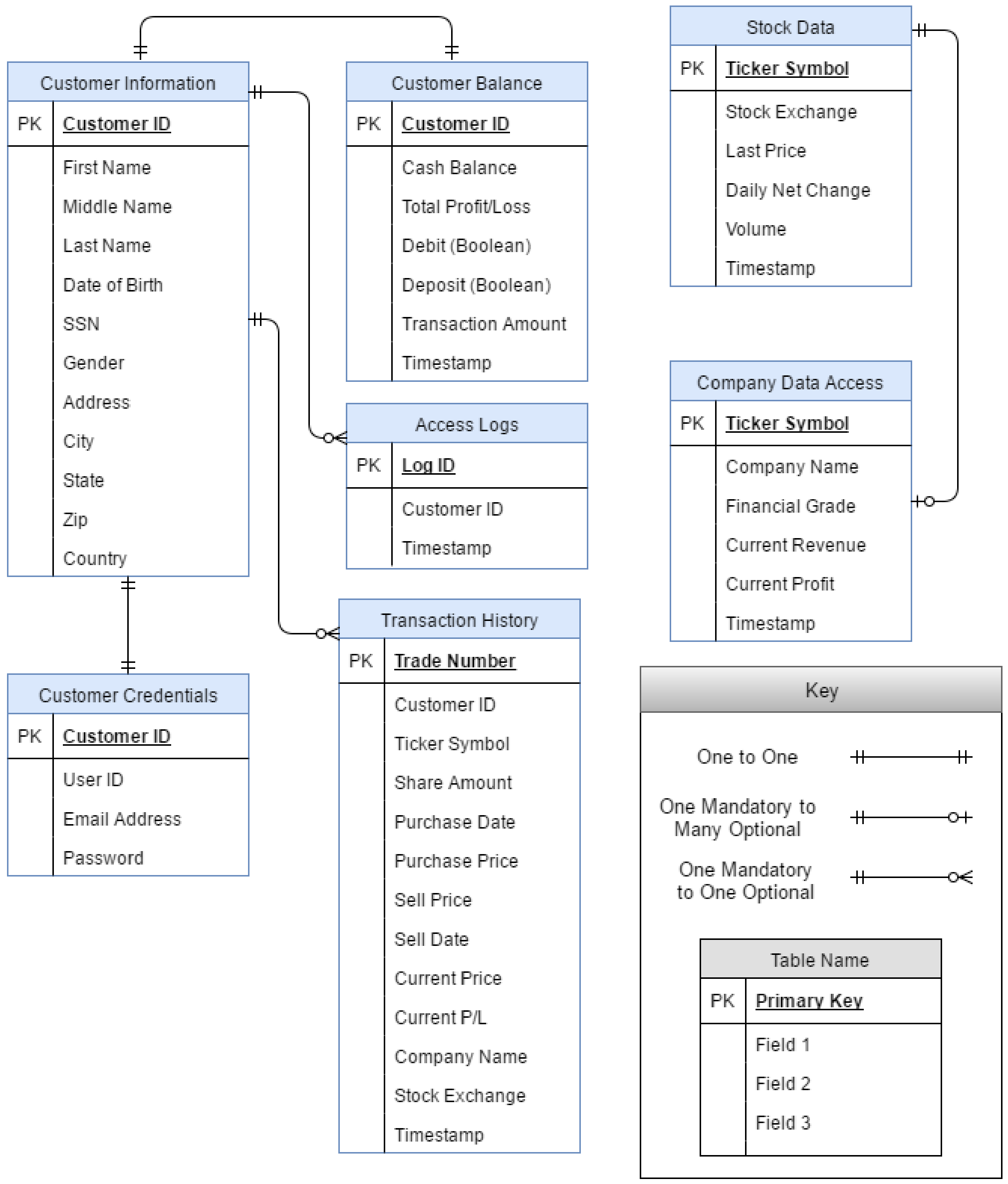
Module views (2)

Use Case Model



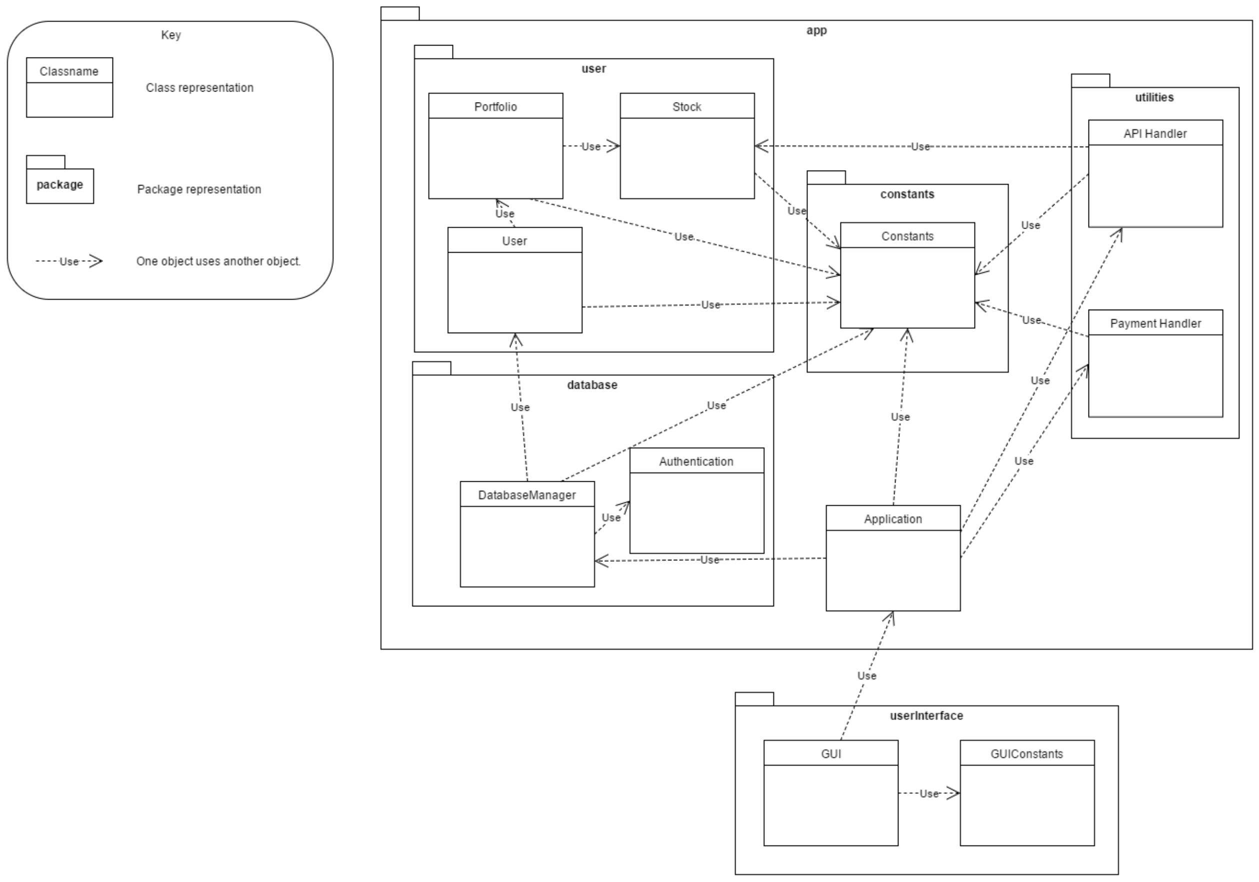
The Use Case is a diagram that shows the basic interactions of different parts of the system. It shows a basic view of how the parts interacts with each other and gives a general understanding of the permissions. We chose to the Use Case diagram because of the many different aspects of the program and how they coordinate with each other. Another benefit of using this view is that it allows you to view security flaws if parts of the system has permissions where it should not.

Data-model (Logical)



The Logical Data Model is a diagram that expresses how the data will be stored for the system. It contains the entities, attributes and relationships of the database. This can be helpful as it is a visual representation of all of the data that is intended to be stored. The diagram allows for the relationships of each table to be easily seen and understood. Furthermore, it helps to avoid data redundancy through overseeing each of the tables. For our system, we have five tables related to customers and two tables related to stock information. This diagram reflects our system in that it depicts the relationship of these customer tables and stock tables so that the data flow can be easily viewed.

<u>Class/Object Diagram</u>



This class diagram illustrates how the user interface will interact with the application. Within the user interface package, there exists a GUI object that is created upon program execution that uses the application object and uses the GUI constants file. Contained within the application package is 4 subpackages that are needed for code structure. In the user package, a user, portfolio, and stock class exist. Each of these classes in the user package use one another to achieve the goal of maintaining state of a user that is using the program. The user package uses the constants file to maintain consistency throughout the package in the way Strings and Integers are represented. There is a database package that uses the user package to store persistent data for a user in the database. The Authentication class is used by the database manager to ensure the correct data points are given for logging in. The constants package contains any constants that are used throughout the program. Almost every class uses the constants file in some way. The Utilities package is responsible for handling interactions with external modules. The API Handler class will handle all interactions with external APIs. The

Payment Handler class will handle all interactions with the external payment system that is needed to confirm transfer of stocks.

Work-Assignment View

Module 1

Diagrams

Class/Object/Package	JJ Kemp
Use Cases	Steven Lightle
Data model (logical)	Ryan Sweeney
System Context	Jimmy Bailey

Other

Work Assignment View	Hannah Church
Discussion Questions	All

Module 2

Diagrams

C&C View	Hannah Church
Allocation View	Ryan Sweeney
MVC View	Jimmy Bailey; Ryan Sweeney

Initial Coding

Database Setup	Hannah Church; Jimmy Bailey
Framework/Git Setup	JJ Kemp; Steven Lightle

Milestone 3

Implementation

Application Files	Hannah Church; Ryan Sweeney; Jimmy Bailey

GUI	JJ Kemp; Steven Lightle

Testing

Unit Testing	Steven Lightle; Jimmy Bailey; JJ Kemp
System Testing	Hannah Church; Ryan Sweeney

<u>Questions</u>

- What is the Design purpose?
    - The design phase helps provide a robust foundation for the remainder of the project. The implementation phases will have clearly-defined goals and tasks thanks to the organization and discussion documented during the design phase.
- What is the system’s Primary Functionality? Describe.
    - The system’s primary functionality is allowing customers to manage their local TradeNet accounts and perform stock trading.
- Describe system’s quality attributes.
    - Security
        - Due to the high amount of sensitive information tied to customer’s accounts, this data must be stored securely to avoid becoming compromised. The system is also connected to many external modules which could each be compromised as a point of entry for malicious activity.
    - Maintainability
        - The system must be flexible so that it can support future feature implementations such as expansion of customer interactions.
    - Availability
        - Customers require constant, real-time updates and feeds of stocks and the data pertaining to them. The site must be able to withstand high amounts of network traffic to protect this functionality as well as protecting against possible denial-of-service attacks.
- List and describe any derived requirements or architectural concerns.
    - One concern that should be closely monitored is the amount of external connections that the system contains. The security of these connections is crucial to the security of the system. The more external connections, the more opportunities for system infiltration.
    - Another concern is the reliability of those external connections. If the API’s that are used for stock updates fail, there should be a contingency plan in place to ensure that the system’s availability is not affected.

    - Two quality attributes that can be derived from the requirements are encapsulation and modularity. Encapsulation and modularity in the code base will contribute to the flexibility of the system.
- What are the constraints? As the architect, are there any constraints you would impose? Discuss.
    - Must use Tradier as the market data source
    - Must use mailboxlayer API for email validation
    - Must utilize twitter updates for ticker symbol
    - Utilizes 2 separate servers for account management (existing) and trade executions.
    - System will be Java-based, so every client needs Java on their system for full functionality.