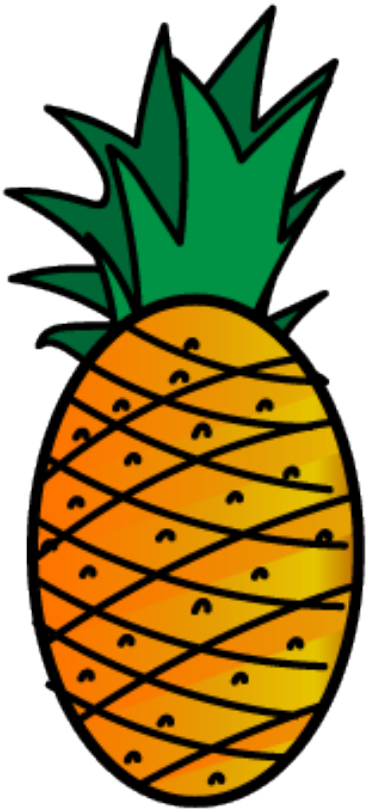


Pineapple

Architecture and Design Document

Version: V1.2
Created: 18th May 2017
Sponsor: Ali Yavari/Reza Soltanpoor
Number: 003
Author: Josh Gerlach
Commercial-in-Confidence

Changelog

Date	Version	Description	Author
18th April 2017	1.0	Created Document. Finished first draft to be sent to supervisor	Josh Gerlach
10th June 2017	1.1	Updated Architecture View to include User interaction and clear separation between layers (section 5.1)	Josh Gerlach
12th June 2017	1.2	Updated according to feedback. Added changelog page and added details to Architecture Overview (section 5.1)	Josh Gerlach

Table of Contents

1 Introduction

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.2 Scope

This Software Architecture Document provides an architectural overview of the Pineapple Stock Buddying Game. Pineapple is being developed by SticksAndStocks to help buddying stock market traders get a feel for the risks and rewards of stock market trading, without the use of real money.

This Document has been made directly from the Pineapple implementation that uses Laravel PHP Framework and the MySQL database.

1.3 Definitions, Acronyms and Abbreviations

Pineapple makes use of various definition, acronyms and abbreviations as follows:

- Pineapple System: A high level term used to describe the Pineapple Stock Buddying Game.
- Guest: An unregistered visitor of the Pineapple System.
- User: A registered and authenticated visitor of the Pineapple System
- Admin: A registered User that has elevated privileges inside the Pineapple System.
- Client: A term for any visitor (guest/user/admin) of the system.
- ORM: Object Relational Mapping, an abstraction of the database tables and relations into Classes that can be used by the Laravel system.

2 Architectural Representation

This document presents the architecture as a series of views; use case view, logical view, process view and deployment view. There is no separate implementation view described in this document. These are views on an underlying Unified Modeling Language (UML) model.

3 Architectural Goals and Constraints

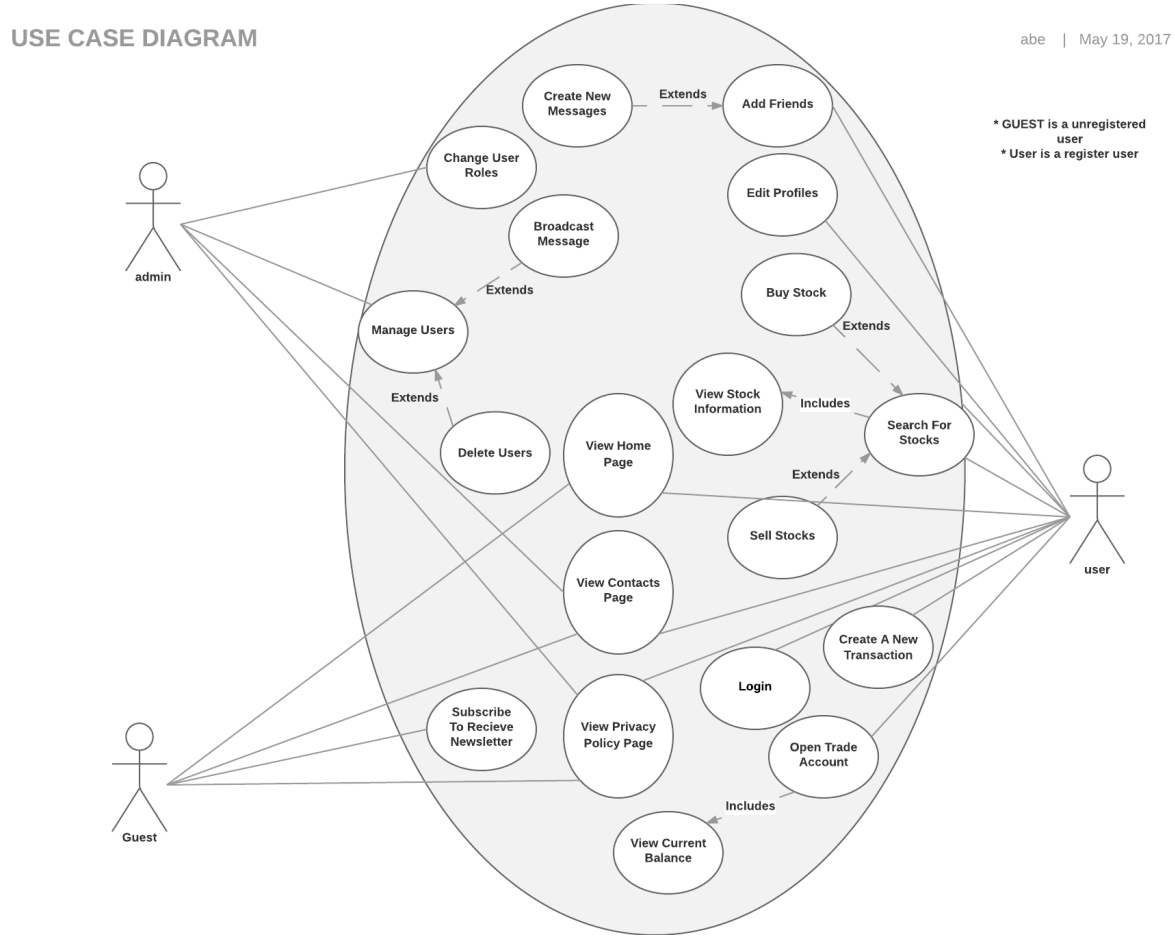
There are some key requirements and system constraints that have a significant impact on the way that the Pineapple System was designed and implemented. These include:

- Using a common, open source, and easy to maintain framework to build the system off. Using Laravel that uses the PHP scripting language and a Model, View, Controller architecture, and using the MySQL Relational Database System, means that the system is able to be easily deployed and maintained.
- Users should be able to view the Pineapple System on any device that has a web browser. This also meant that on different device sizes the Pineapple System had to look and show the requested information properly.
- The added requirement to allow the searching, viewing, buying and selling of other stock markets meant that we had to have a very elastic system that could be easily manipulated to incorporate these changes. The use of Yahoo! Finance meant that we had such flexibility.
- Cost of development was also a major factor. We had specified that we wanted a server for the customer to view progress and provide immediate feedback if necessary. Being a PHP and MySQL framework meant that we could deploy a Linux server to host the development system that would reflect the real life implementation.

4 Use-Case View

The Use Case Diagram demonstrates how different actors interacts with in the application in order to accomplish a goal.
Precondition:

- The administrator is registered actor and has elevated privileges
- User has to register in order to login and use the application
- Guest is a unregistered user that can view index page



5.1 Logical View

A description of the logical view of the architecture. Describes the most important classes, their organization in service packages and subsystems, and the organization of these subsystems into layers. Also describes the most important use-case realizations, for example, the dynamic aspects of the architecture. Class diagrams may be included to illustrate the relationships between architecturally significant classes, subsystems, packages and layers.

The logical view of the Pineapple Game is comprised of 3 main packages: Model, View, Controller. There is also 4 sub main packages that help control system management, including Global Reusable functions and classes: Artisan Commands, Laravel Framework, Custom Middleware and Custom Helpers.

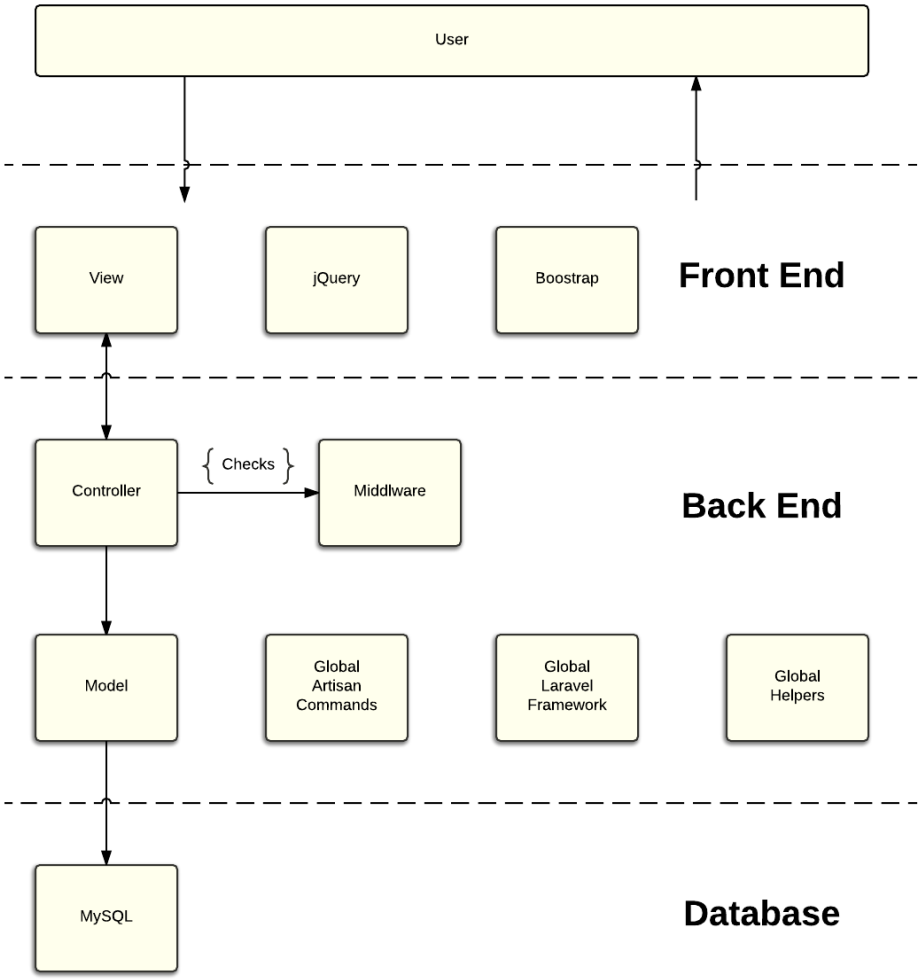
The Model Package contains classes for interfacing to the database through ORM and providing convenience functionality for Controllers and Views to get data dynamically. This package is also responsible for providing an abstraction layer to model the external data collected from external APIs.

The View Package contains classes and a rendering engine (called Blade) that pre-renders Model data into HTML DOM and/or Javascript objects. This provides a very easy and convenient way to organise data representation to the Guest/User/Admin of the client device.

The Controller Package contains classes that act as a mediary between the Model and the View packages. This package controls the client's interaction to the model, through manipulation and validation the Controller package makes sure that the View that the client interfaces to cannot directly manipulate data on the Model. The Controller Package also manipulates data from the Model through to the view, providing a business logic layer.

The Laravel Framework was used to control the flow of the Pineapple System, mainly through the heavy use and enforcing of the MVC architecture pattern. The Laravel Framework provides functionality for testing and separation of web and API implementation, making development very modular and easy to maintain.

5.1 Architecture Overview



5.1.1 Frontend

Layer

Represents the HTML/CSS/Javascript to be rendered on the client's browser. Provides a pre-rendering engine to interface to Model components through the Blade templating engine, which runs Laravel and PHP commands prior to rendering. Also encapsulates libraries for dynamic styling and scripting such as jQuery and Bootstrap.

5.1.1.1 jQuery

Library

A Javascript library that runs inside the view. Provides convenience functionality for common tasks such as AJAX API calls, autocomplete and other dynamic features. It also helps with code maintainability through the use of instance control and single point DOM manipulation.

5.1.1.2 Bootstrap

Library

CSS/Javascript library to help make styling easier, and is driven by a mobile first approach, meaning that it is designed with responsive functionality inbuilt. The library was overridden and modified in some sections to create a uniform and consistent look over the Pineapple System. See the Style Guide for more information on specific modifications made.

5.1.2 Controller

Layer

Represents client actions, responses and data transfer between the Model and View layers. Used to create a single point of contact between the User and Model abstractions. It is also the controller's responsibility to provide most of the business logic between User and Model representations, including but not limited to: currency conversion, wrapping messages and error codes into packages that can be used in the view, collating data across different representational types, etc...

5.1.3 Model

Layer

An ORM representation of the Database, with validation of passed data. Provides abstract functionality to insert/update/read/delete objects in database. The Model layer also provides basic functionality for pre-processing of ORM data so it can be more efficiently used by the Controller or View engine. Model ORM objects and migrations are used as an abstraction to the database so the User or System Administrators do not need to maintain the database schema directly, this provides a secure and maintainable interface when dealing with complex data and analysis.

5.1.4 Database

Layer

Represents the database implementation and schema. The Pineapple System implements the database using the MySQL relational database type. Management of the database is through the Laravel Framework migrations and ORM implementations.

5.1.5 Middleware

Layer

The wrapping layer used by the Controller to restrict access to certain functionality of the Pineapple System. The Laravel Framework provides default Middleware checkers for Authentication, but custom Middleware validation was created to manage Administrative rights.

5.1.6 Global Artisan Commands

Layer

Laravel provided abstraction of server functionality. Provides ways to add custom commands for use system wide at a command line level, with a built in server for local testing and development. Custom commands were created as runner tasks to update stock values, provide facades for email notifications and stock market analysis, and update stock histories. This also provides easy testing and command pattern interfaces for faster and more efficient development.

5.1.7 Global Laravel Framework

Layer

Framework that the Pineapple System organisation. Provides a plethora of functionality and boilerplate code to get the Pineapple System running and reliable.

5.1.8 Global Helpers

Global Functions and Classes

Represents the functions and classes that are used system wide, but do not require the use of low level system calls provided by the Artisan Commands. Used when a ORM object function was used in the same way across other unrelated ORM objects. Provides sophisticated abstraction of these common tasks, wrapped into functions and classes for grouping of functionality and maintainability.

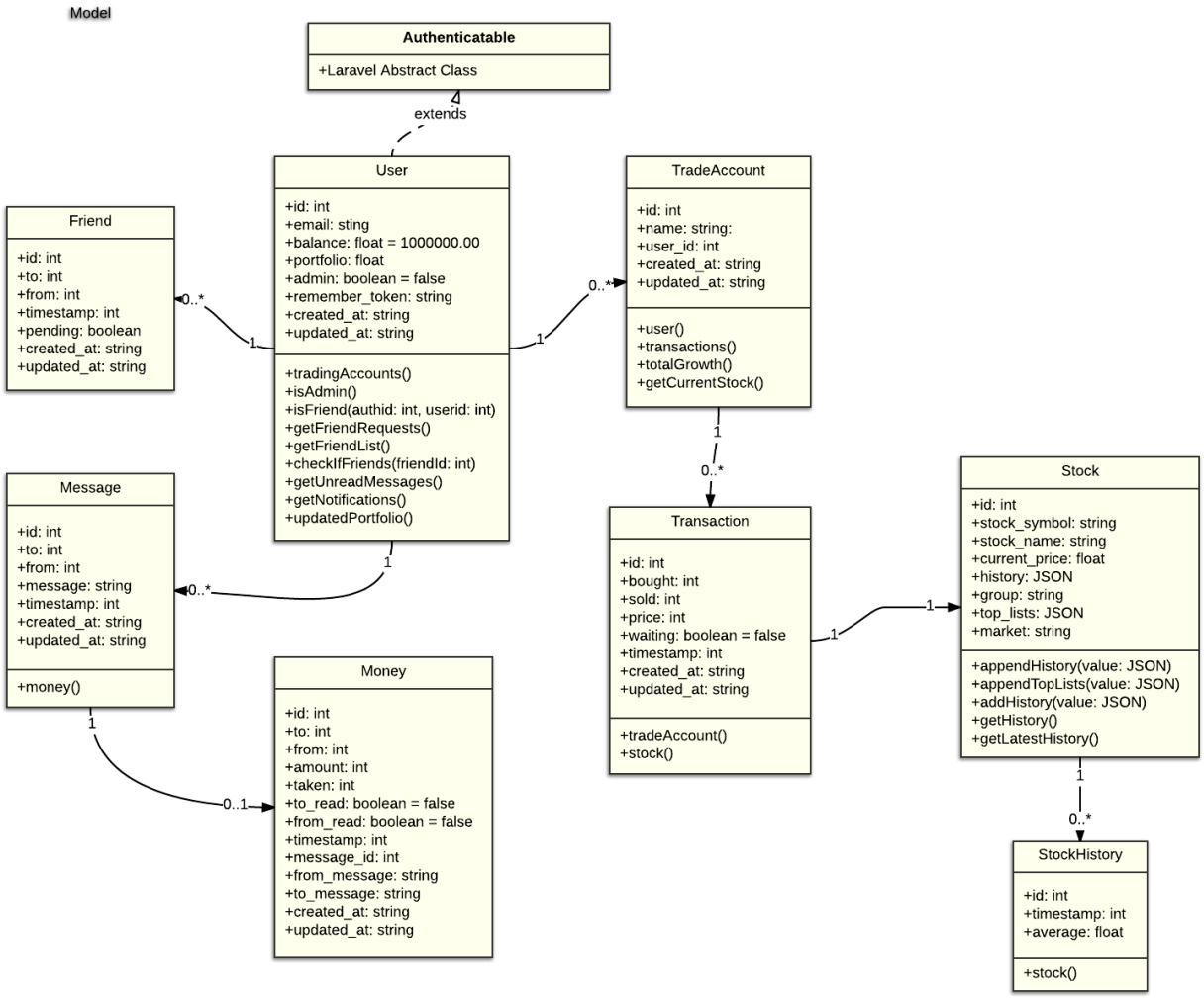
6 Module Interconnection Architecture View

A description of relationships using Class Diagrams of the three major elements (Model, View and Controller), and the major sub components (Middleware, Artisan Commands and Global Helpers).

6.1 Model

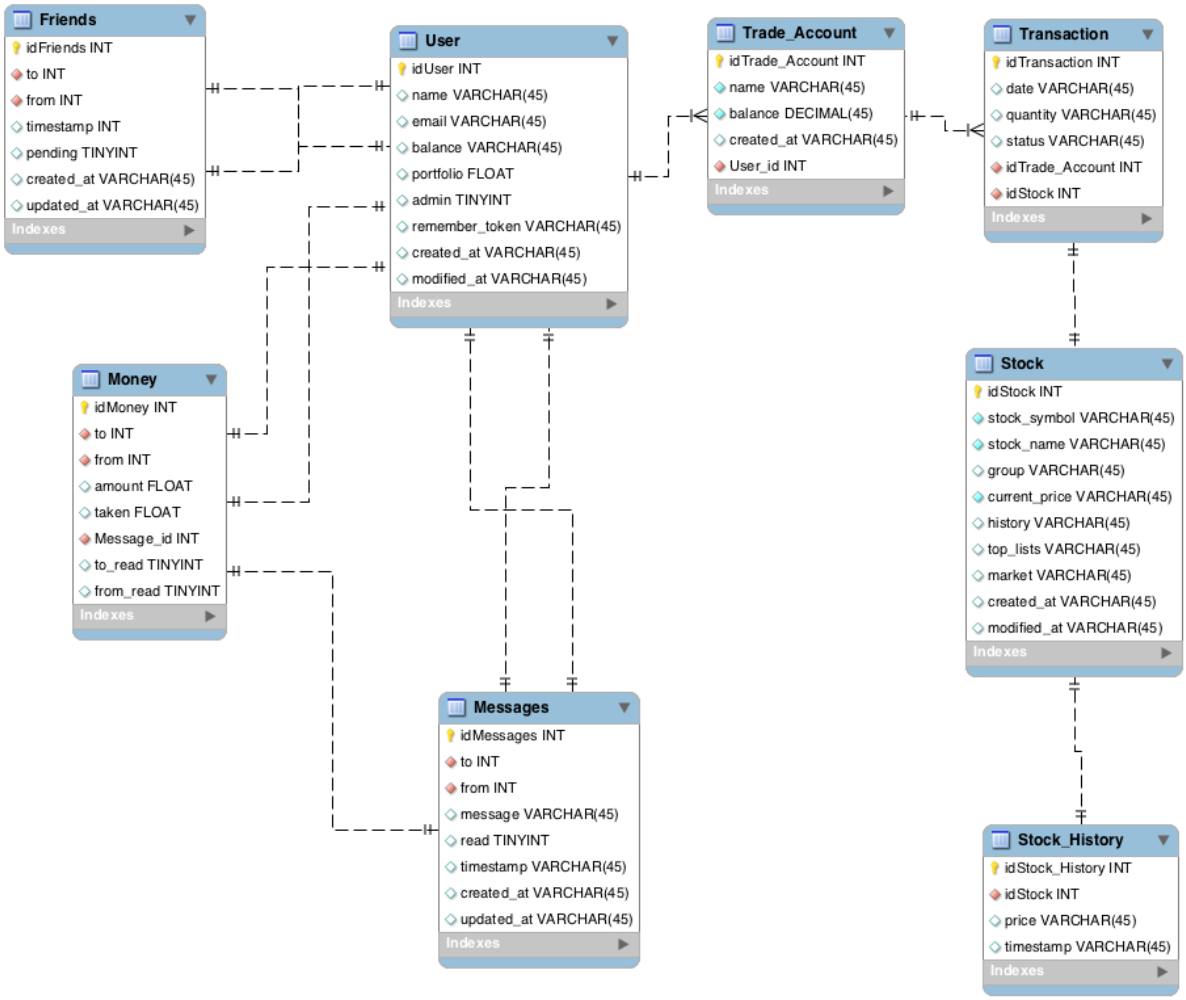
6.1.1 Model Class Diagram

Show the layout of the Model that maps through an ORM to the database. Model controls the constraints and validation of the data relating to the database. The Model also provides functions and methods to Controllers and Views to easily and clearly retrieve and model data from the database.



6.1.2 Database ER Diagram

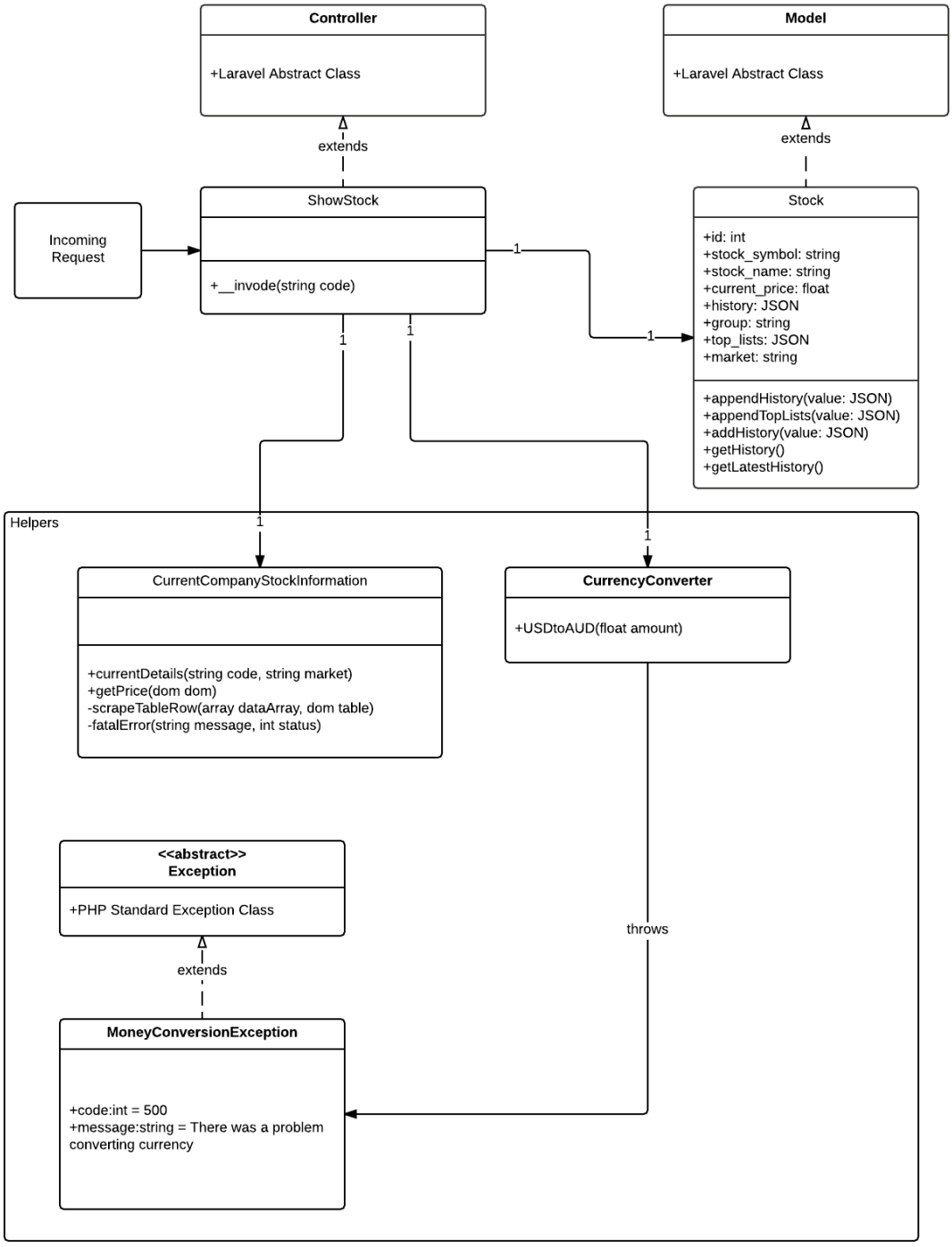
Shows the layout of the database and the relationships between tables. This maps one-to-one with the Model diagram, thus completing the ORM.



6.2 Controllers

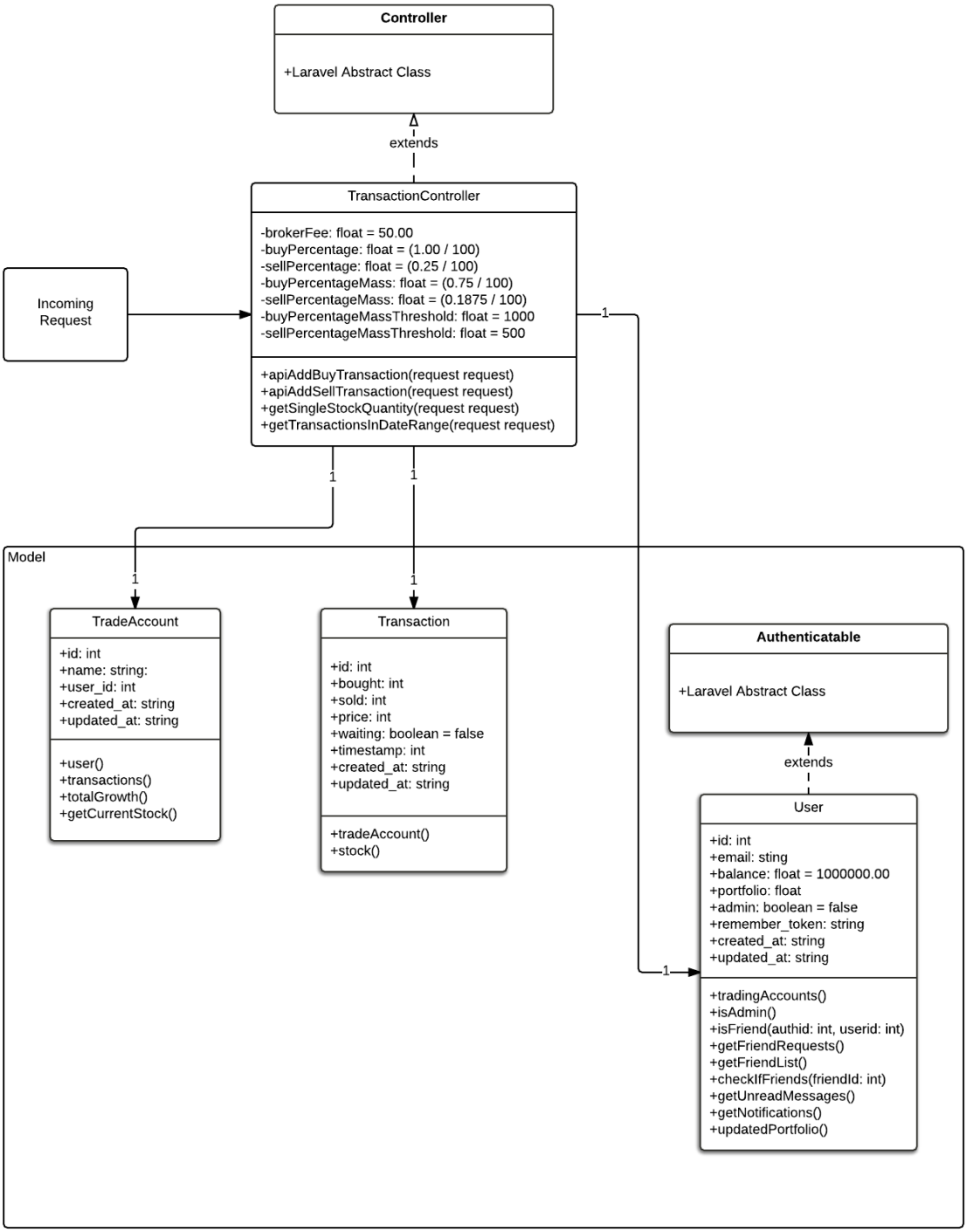
6.2.1 Show Stock Controller

Shows the interaction between classes when an incoming request asks for information on a Stock Company. This only shows the internal workings of getting Stock, not the API calls as they are contained within the functions of the classes shown.



6.2.2 Transaction Controller

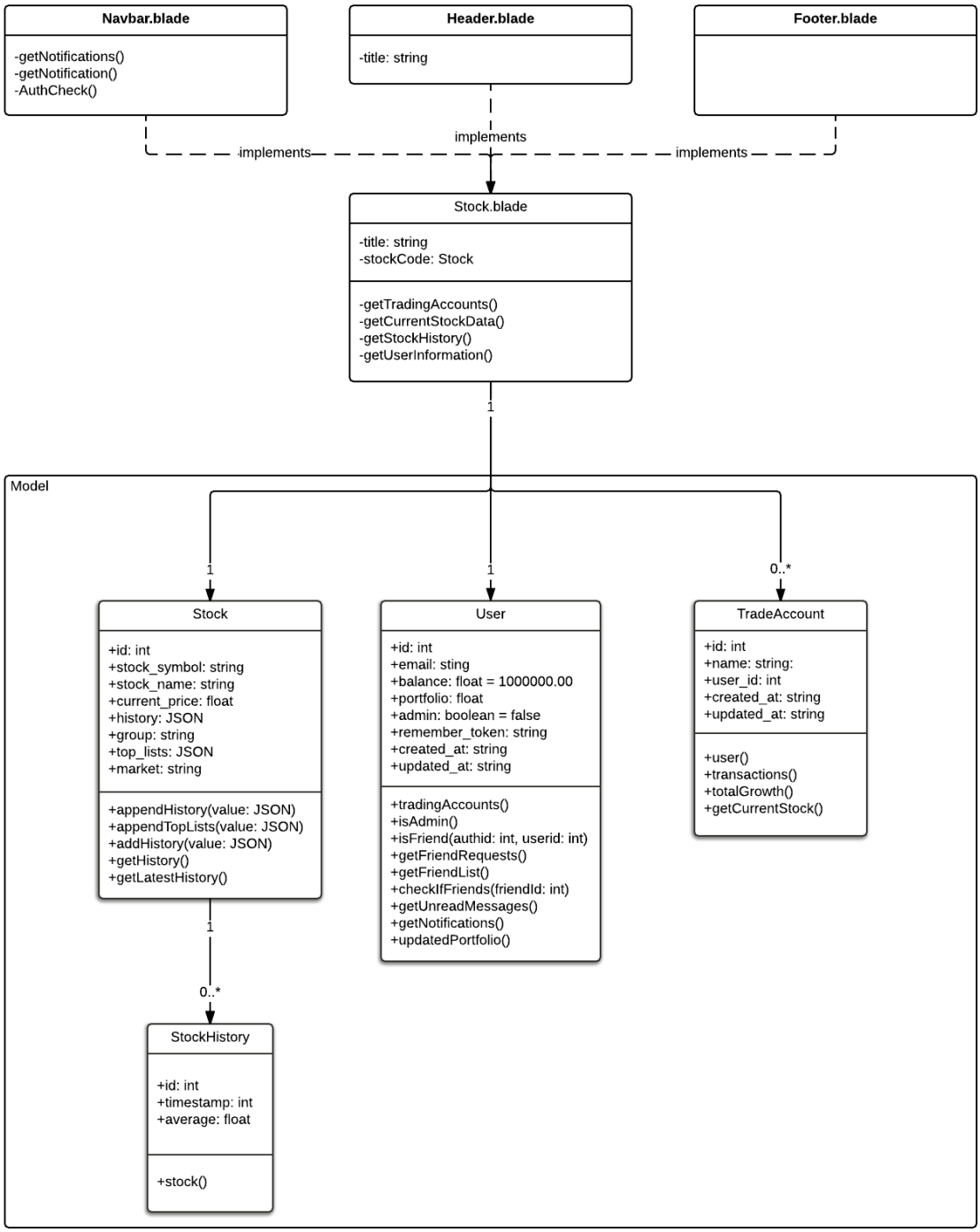
Show the class interaction when buying and selling shares, how it interacts with the Model to update User and Trade Account information. The Transaction controller mediates the data manipulation and model abstraction, with data and user validity checking. The class diagram for both buying and selling are identical, as such they also share common functionality and implementation to help with maintainability.



6.3 Views

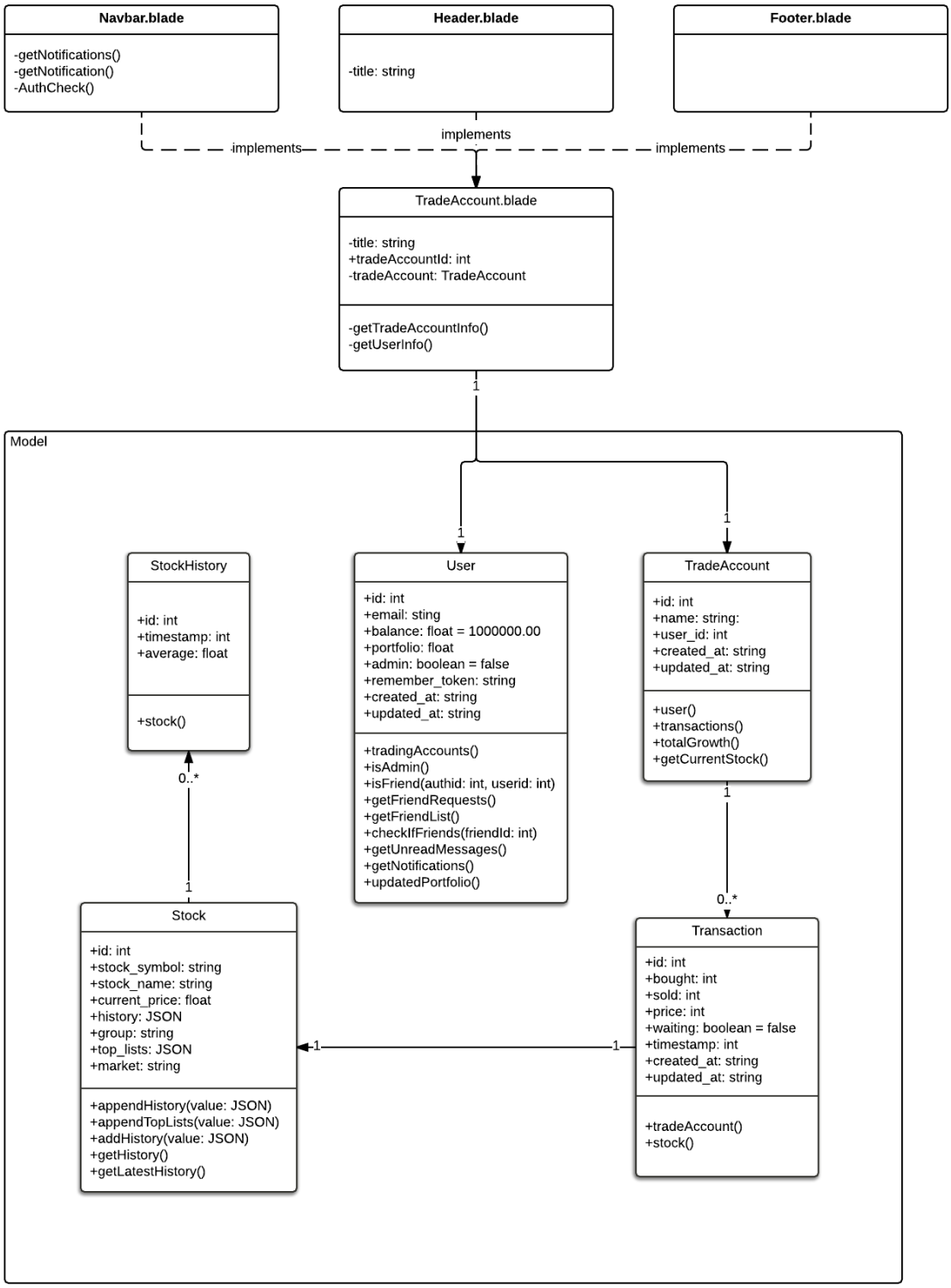
6.3.1 Stock View

Describes the class interaction between the Stock.blade class, the included files and the calls to the Model. This does not cover API or AJAX calls directly, however it does include the collection of all information that is required to perform all of these tasks.



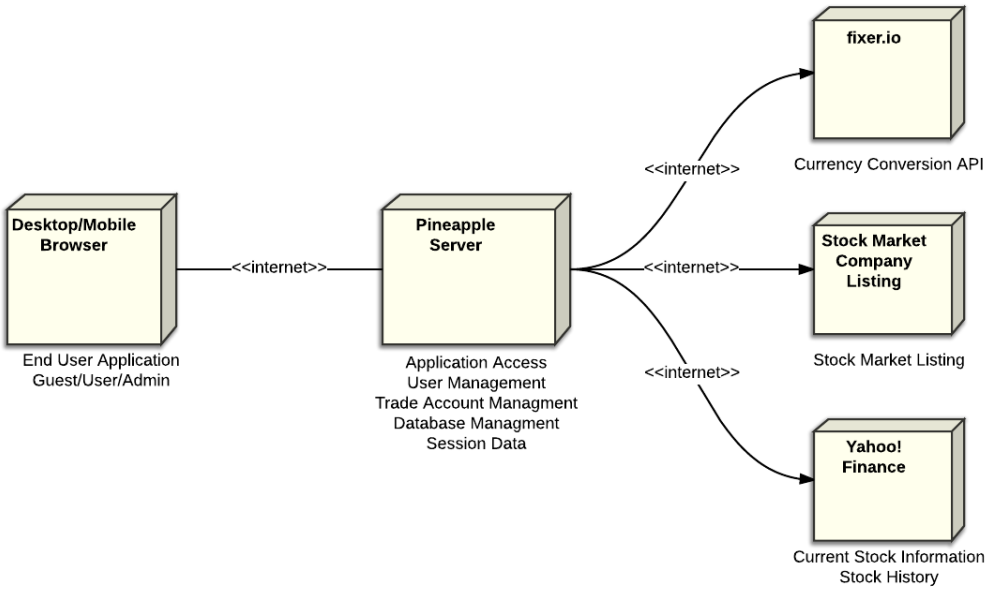
6.3.2 Trade Account View

Describes the class interaction between the trade-account.blade class, the included files and the Model. Shows retrieval of information passed from Controller to the View, then how the View can call the Model to get the information to display.



7 Deployment View

A high level view of the deployment architecture of the Pineapple System. Describes the various physical nodes for the platform integration configurations. The idea is to get an overview of how the Client, Pineapple Server and remote APIs come together over internet connections to present and process data requested by the client.



7.1.1 Desktop/Mobile Browser

Guests can register as Users who interact with the server create Trade Accounts to buy, sell and view Stock information. Users can also use the social media features such as Add/Remove Friends, message Friends and send/receive money from Friends.

7.1.2 Pineapple Server

The Pineapple Server is the server that orchestrates the interaction between Clients and associated APIs and Database functionality. All Clients have varying degrees of access to different features of the server depending on privileges assigned. Also retrieves and manipulates data from external API providers.

7.1.3 Fixer.io API

A third party REST API that returns any common current currency conversion rates in JSON format. Access is controlled by Pineapple Server.

7.1.4 Stock Market Company Listing

A collection of third party CSV files that contain a list of all the currently listed Stock Exchange companies for ASX, NASDAQ, NYSE and AMEX. CSV files are extracted, sorted and inserted into the database for use by Clients.

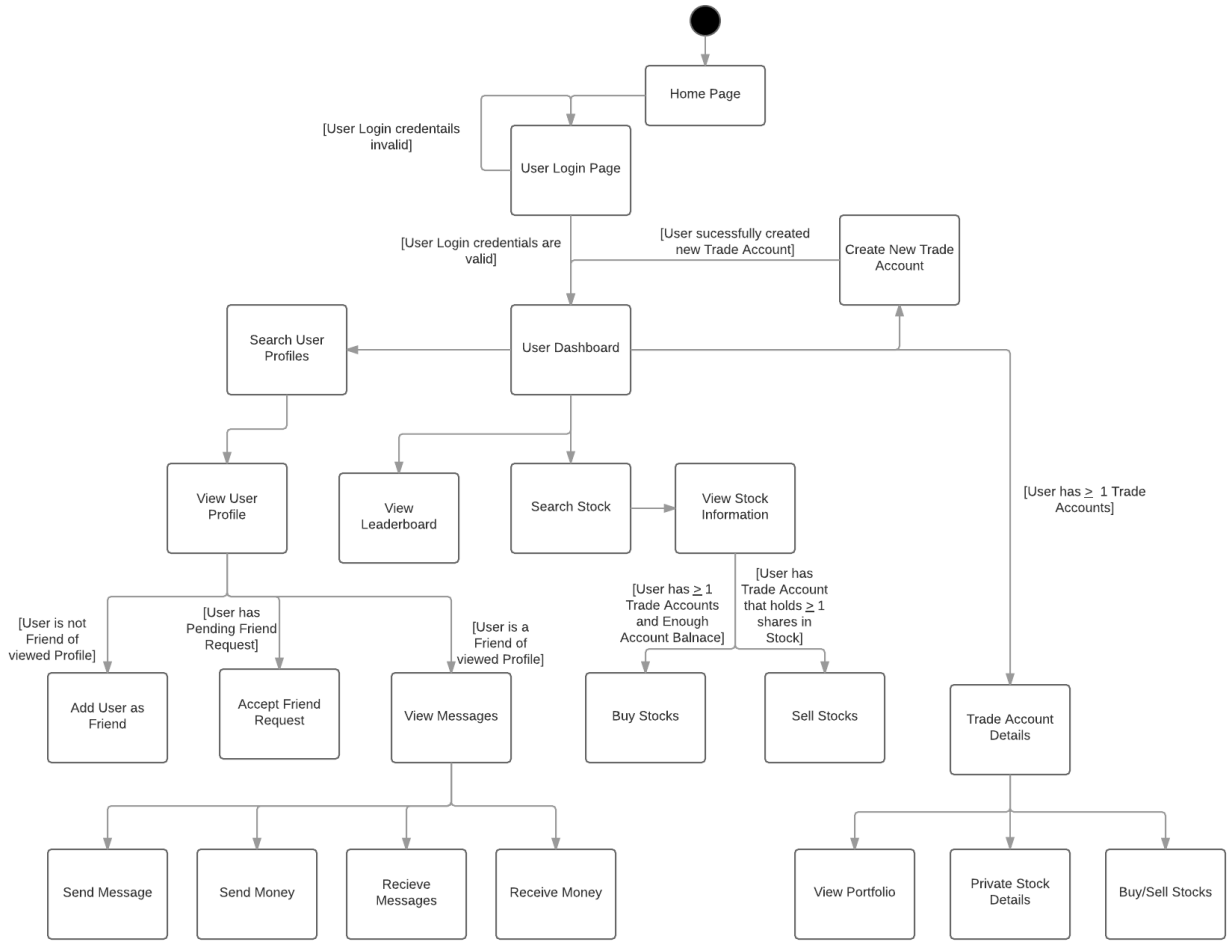
7.1.5 Yahoo! Finance and Yahoo! Finance Chart API

Yahoo! Finance provides extensive and up to date data on Stock Companies worldwide. It is where the Pineapple System gets all of its latest Stock data from. Yahoo! Finance Chart API provides a REST API to get bulk histories of Stock Companies in CSV format, that is then gathered and collated by the Pineapple Server.

8 Activity Flow

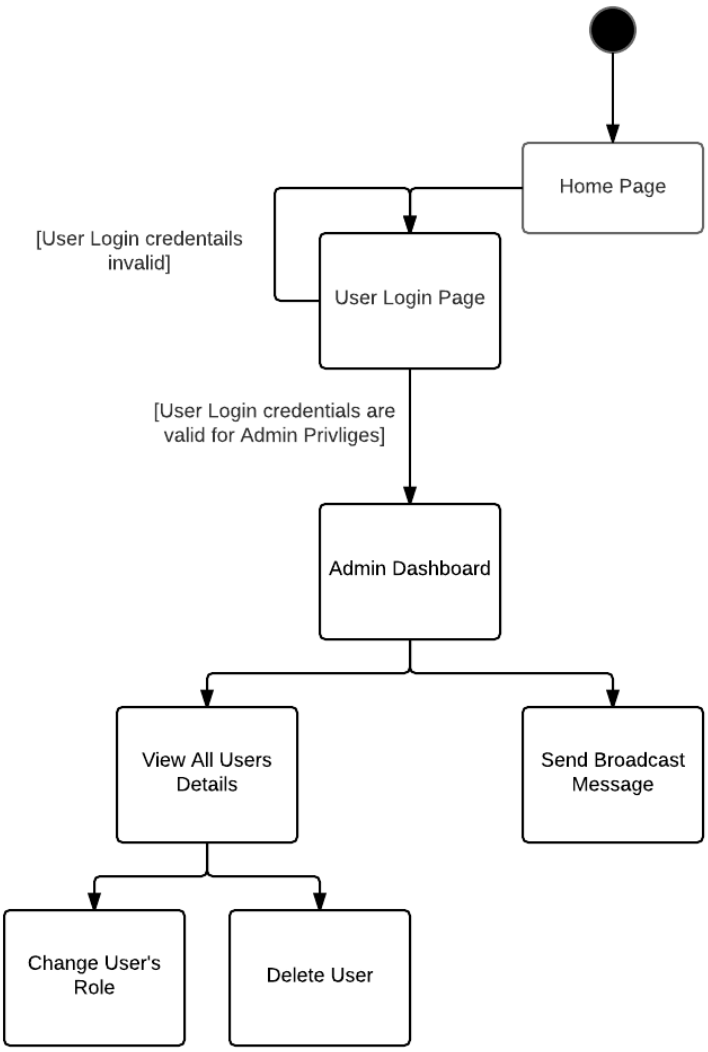
8.1 User Activity Diagram

Describes the User interaction with the Pineapple System. Shows what a User can do, view and manipulate within their defined privileges. This includes the sign in process, the purchase and selling of Stocks, viewing other user profiles and more.



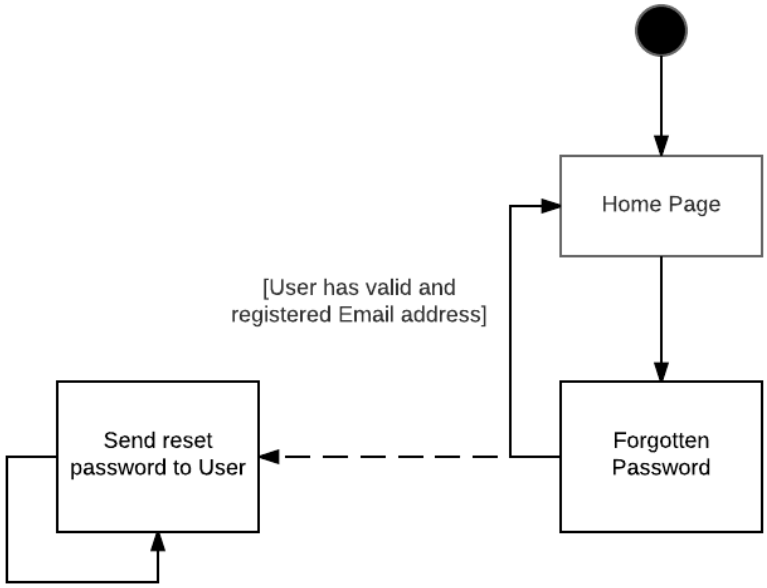
8.2 Administrator Activity Diagram

Describes the Administrator interaction with the Pineapple System. Show the flow of modifying a User's role, deleting a User and sending a broadcast message to all registered Users in the Pineapple System.



8.3 Forgotten Password Activity Diagram

Describes the interaction a Guest who is registered with the system but has forgotten their password to log into the system. Shows the flow of forgetting their password and recovering login information.



Appendix

References

Software Documentation Templates and Examples. 2017. Software Documentation Templates and Examples. [ONLINE] Available at: http://sce2.umkc.edu/BIT/burrise/pl/appendix/Software_Documentation_Templates. [Accessed 18 May 2017].

Software architecture description - Wikipedia. 2017. Software architecture description - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Software_architecture_description. [Accessed 18 May 2017].

Software Architecture | Tools & Methods | Documenting the Architecture | Views and Beyond: The SEI Approach to Architecture Documentation. 2017. Software Architecture | Tools & Methods | Documenting the Architecture | Views and Beyond: The SEI Approach to Architecture Documentation. [ONLINE] Available at: http://www.sei.cmu.edu/architecture/tools/document/viewsandbeyond.cfm. [Accessed 18 May 2017].

Example: Software Architecture Document. 2017. Example: Software Architecture Document. [ONLINE] Available at: http://www.ecs.csun.edu/~rlingard/COMP684/Example2SoftArch.htm. [Accessed 18 May 2017].

Software documentation - Wikipedia. 2017. Software documentation - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Software_documentation. [Accessed 18 May 2017].