FROM:	**KONSTANTINOS MITSARAKIS** **CHARIS VAIRLIS** **DAN ŠILHAVY**
TO:	ANTONIOS SIDIROPOULOS MICHALIS SALAMPASIS STEFANOS UGIAROGLOU
SUBJECT:	**Stock Exchange IEX API**
DATE:	SUNDAY 13/01/2019

Table of contents

1. TradeStock - Project description

This application is a student project for the graduate course in MSc. under the supervision of Dr. Michaelis Salampasis. It is developed by a team of 3: Konstantinos Mitsarakis (GR), Charalabos Vairlis (GR) and Dan Šilhavý (CZ).

The aim of this project is to build an API application and for that we have chosen the IEX (Investors Exchange) API solution.

Application is available here:

www.382235420.linuxzone118.grserver.gr/m102/

Application **admin panel** is here:

www.382235420.linuxzone118.grserver.gr/m102/backend

Email: kostas.mitsarakis@gmail.com

Password: 22222222

GitHub repository is public too:

https://github.com/kostmits83/m102/

Project management is available here:

https://app.asana.com/0/885036296387213/885036296387213

Documentation for the project is here:

https://github.com/kostmits83/m102/wiki

2. IEX API

is a free and reliable set of services designed for developers and engineers that are functioning and surrounding the **IEX stock exchange** (for further explanation how the stock market works see www.iextrading.com). The API can be used to build both high-quality or prototype application and services.

2.1. Basic stock market concepts

1. **Stocks**
2. **Markets**

2.2. Features of IEX API

1. **Active-Active** Multiple copies of every IEX server to ensure high availability and to protect against regional failures.
2. **Redundant** 5 data centers with multiple DNS providers.
3. **Auto-Scaling** Scaling of servers on-demand based on custom metrics.

2.3. Performance measures

- 99.981% web uptime.
- 100% database uptime.
- 3.5 million messages per second peak & Data loss protection
- 262 TB data transferred per month
- RegSCI System that regulates the API for the highest level of integrity, resiliancy, and security.
- Google Cloud. High performance, scale, and reliability ensured by Google, LLC.
- We developed and patented multiple ways to ensure data integrity.
- 1.1 trillion database records with 72 TB database size
- 600,000 queries per second peak

- Redis caching: 180 K average operations per second
- 10.51 B operations per day

You can read more about IEX API at the following link: www.iextrading.com/developer/.

3. User Stories

Borrowing from agile terminology, action-oriented documentation is often based on user stories. A user story is a definition of a process whereby a goal is accomplished. We used the most common template for identifying user stories: **As <type of user>, I want <what?>, so that <why?>.**

For our project, from the user's point of view some prototype features were described and are presented as follows:

- As a **user**, I want to be able to **sign up and have an account,** so that I can **add and keep my favorites stokes in a place.**
- As a **user**, I want to be able to **reset my account password,** so that I can **recover it easily in case I forget it.**
- As a **user**, I want to be able to **delete my account,** so that I can **keep my privacy and utilize the right of erasure (GDPR).**
- As a **user**, I want to **view all the available stocks in a list**, so that **I can get more information about every stock I'm interested in**.
- As a **user**, I want to be able to **add multiple stocks in a comparison list**, so that I can **compare metrics of the markets more clearly**.
- As a **user**, I want to be able to **add a stock in favorites list,** so that I can **navigate easily each time I want to get statistics about this stock**.
- As a **user**, I want to be able to **contact to the app support by submitting a contact form,** so that I can **discuss further needs or making a comment.**
- As an **administrator**, I want to **have access on submitted contact forms** so that **I can read them and reply** to the senders if it is essential.

- As an **administrator**, I want to **have access to stocks list,** so that **I can create or manage existing stocks**.

4. Technologies

1. **Yii** / PHP framework
2. **Bootstrap** / Front-end framework
3. **SASS** / CSS preprocessor

4.1. Yii

Yii is an open source, object-oriented, **component-based MVC web application PHP framework**. Yii is originally Chinese framework and the names is pronounced as "Yee" or [ji:] or [γιː], which means "simple and evolutionary".

Yii 1.1 was released in January 2010 adding a form builder, relational Active record queries, a unit testing framework and more. The Yii community continues to follow the 1.1 branch with PHP7 support and security fixes. The last release was version 1.1.20 in July 2018, stable version.

In May 2011 the developers decided to use new PHP versions and fix architectural shortcomings, resulting in **Yii version 2.0**. In May 2013 the **Yii 2.0** code went public, followed by the first stable release in October 2014. PHP7 is supported since version 2.0.9.

Features

- Model-View-Controller (MVC) design pattern.
- Generation of complex WSDL service specifications and management of Web service request handling.

- Internationalization and localization (I18N and L10N), comprising message translation, date and time formatting, number formatting, and interface localization.
- Layered caching scheme, which supports data caching, page caching, fragment caching and dynamic content. The storage medium of caching can be changed.
- Error handling and logging. Log messages can be categorized, filtered and routed to different destinations.
- Security measures include prevention of cross-site scripting (XSS), cross-site request forgery (CSRF) and cookie tampering.
- Unit and functionality testing based on PHPUnit and Selenium.
- Automatic code generation for the skeleton application, CRUD applications, through the Gii tool.
- Code generated by Yii components and command line tools complies to the XHTML standard.
- Designed to work well with third-party code. For example, it's possible to include code from PEAR or the Zend Framework.

More about Yii can be read at the following link: www.yiiframework.com.

4.2. Bootstrap

Bootstrap is a free and open-source front-end framework for designing websites and web applications. It contains HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many earlier web frameworks, it concerns itself with front-end development only.

Features

- Support for the latest versions of the Google Chrome, Firefox, Internet Explorer, Opera, and Safari (except on Windows).
- Support for responsive web design (mobile-first design attitude)

- Layout of web pages adjusts dynamically taking into account the characteristics of the device used (desktop, tablet, mobile phone).
- Sass and flexbox support.

More about bootstrap can be read at: www.getbootstrap.com.

4.3. SASS, CSS preprocessor

SASS, CSS preprocessor stands for *Syntactically awesome style sheets* and it is a style sheet language initially designed by Hampton Catlin and developed by Natalie Weizenbaum.

SASS is interpreted or compiled into Cascading Style Sheets (CSS). SassScript is the scripting language itself. Sass consists of two syntaxes. The original syntax, called "the indented syntax", uses a syntax similar to Haml. It uses indentation to separate code blocks and newline characters to separate rules. The newer syntax, "SCSS" (Sassy CSS), uses block formatting like that of CSS. It uses braces to denote code blocks and semicolons to separate lines within a block. The indented syntax and SCSS files are traditionally given the extensions .sass and .scss, respectively.

Features

- Logical Nesting of styles
- Variables with these data types: Numbers, Strings, Colors & Booleans
- Functions (Mixins)
- Math

Read more about SASS at the following link: www.sass-lang.com.

5. Interactions

5.1. Components of the application

Frontend

Frontend is composed of all things the browser can read, display and/or run. Literally we speak about php file containing HTML views, and separate CSS and JavaScript files that client (browser) can read. All the components are written in **HTML, CSS and JavaScript.** We used mainly jQuery and Ajax as for JavaScript libraries and Bootstrap as a frontend framework**.**

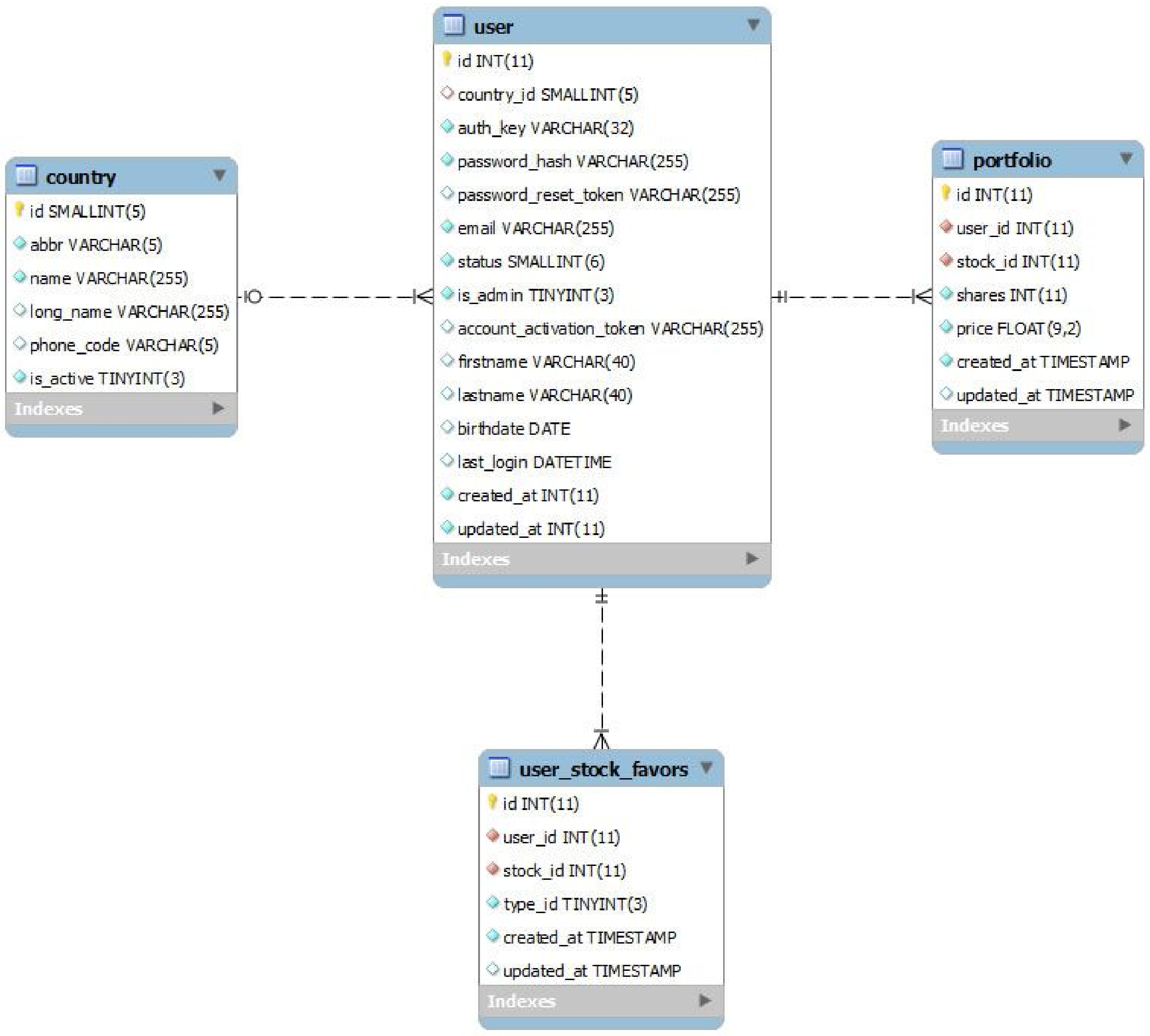
Backend

Backend is composed of all things that run on a server side. This side consists of two more parts: app logic (the main control center), and database, where all persistent data is stored. For our backend we used PHP programming language. We hosted MySQL database.

API

Our API was written in PHP programming language and it is composed out of methods (for instance $stats which retrieves key statistics from IEX database) and endpoints, which are linked with Web Services Server listeners.

E-R Diagram, demonstrating individual elements in the database and its relations



5.2. Interaction between the components

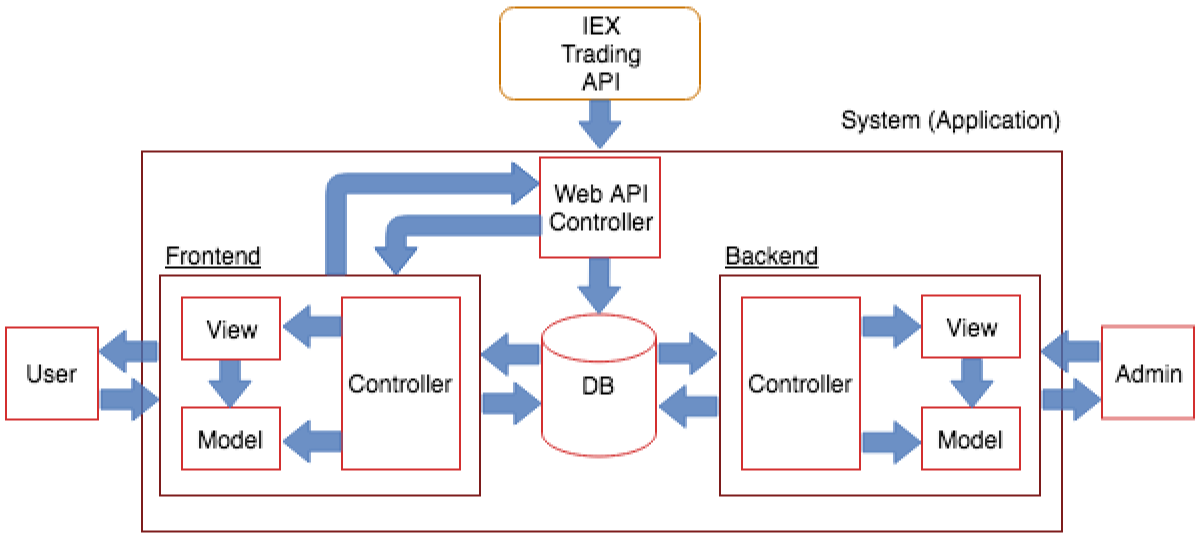
In our case the There are two main architectures today that define how your backend and frontend interact:

Server-rendered apps The first is straight up HTTP requests to a server-rendered app. This is a system whereby the browser sends a HTTP request and the server replies with a HTML page.

Between receiving the request and responding, the server usually queries the database and feeds it into a template (ERB, Blade, EJS, Handlebars).

Once the page is loaded in the browser, HTML defines what things are, CSS how they look and JS any special interactions.

General overview of application data flow and interactions with terminators



This overview is not a formal UML diagram and serves only as a complement to the actual diagrams and project's systematic analysis.

6. Team management & collaboration

6.1. Asana / Project management system

We worked on Asana in order to organize our team project. Asana is a free web and mobile application designed to help teams organise, track, and manage work. www.Asana.com

6.2. Slack / Communication tool

Slack was meanwhile our place for public conversation and place ideas. Slack is generally a cloud-based set of proprietary team collaboration tools and services.

6.3. GitHub / Versioning Control System

We use GitHub as our version-control system. It enables us tracking of changes in computer files and coordinating work on those files among multiple people. It is primarily used for source-code management in software development, but it can be used to keep track of changes in any set of files. As a distributed revision-control system, it is aimed at speed,data integrity, and support for distributed, non-linear workflows.

For better control over the team collaboration and state of the code we use a lot of branches, see GitHub repository here.

6.4. Google Drive / Google Docs / Google presentation

Google Drive helped us organize shared folders and files in which we worked together and simultaneously. More specifically we worked on Google Docs to write down our report and on Google presentation to create a brief presentation for our project.

6.5. Draw.io / Diagram Draw Tool

Draw.io is free online diagram software for making flowcharts, process diagrams, org charts, UML, ER and network diagrams. We used draw.io in order to design the use case diagram of our project. **Use case diagram** is presented later in this document.

7. Stages of development

The team was established in October of 2018, consequently these steps were taken:

1. Planning: of the technologies and calculating the strengths and weaknesses of the project. Project management and communication tools were set up.

2. Analysis: of the SW performance at various stages and making notes on additional requirements. Analysis is very important to proceed further to the next step.

3. Design: Once the analysis was complete, the step of designing took over, which was basically conceptualising the architecture of the project. This step helped remove possible flaws by setting a standard and attempting to stick to it.

4. Development & Implementation: Once the software has being developed, the stage of implementation comes in where the application goes through a study (eg. UX analysis) to see if it's functioning properly.

5. Testing: The testing stage assessed the software for errors and consequently fixing them.

6. Completion: All the relevant documentation was uploaded to GitHub.

8. System requirements

The TradeStock application runs on Apache, PHP, MySQL and it utilizes the technologies that have already been mentioned.

As minimum requirements by Yii, we have:

- Web server to support PHP 5.1.0 or above.
- Apache HTTP server, version 1.1.0c and above (It may also run on other Web servers and platforms provided PHP 5 is supported).

As minimum requirements by Database, we need:

- MySQL version 5.7 above. This is because we use json as data type and this json alias is available on this MySQL version or above.

9. Installation guide

9.1. Server

You can install WAMP (http://www.wampserver.com/en/) or XAMPP or MAMP (https://www.apachefriends.org/index.html) or you can use Vagrant.

9.2. Yii

The proposed way is using Vagrant. Vagrant will isolate dependencies and their configuration within a single disposable, consistent environment, without sacrificing any of the tools you are used to working with (editors, browsers, debuggers, etc.).

Once you or someone else creates a single Vagrant file, you just need to vagrant up and everything is installed and configured for you to work. Other members of your team create their development environments from the same configuration, so whether you are working on Linux, Mac OS X, or Windows, all your team members are running code in the same environment, against the same dependencies, all configured the same way.

So, having set up a web server e.g. WAMP you have to follow the steps below:

1. Open a console terminal.
2. Go to your www folder.
3. Run the following command:
    - composer create-project --prefer-dist yiisoft/yii2-app-advanced tradestock
4. Execute the init command and select dev as environment.
5. Create a new database and adjust the components['db'] configuration in /path/to/tradestock/common/config/main-local.php accordingly.
6. Apply migrations using terminal command /path/to/php-bin/php /path/to/tradestock/yii migrate.

All database changes and modifications will be handled using migrations. Because a database structure change often requires some source code changes, migration feature allows you to keep track of database changes in terms of database migrations which are version-controlled together with the source code.

9.3. SCSS

To install SCSS you can simply pick the way you like best from https://sass-lang.com/install

9.4. Compass

We use a useful CSS Authoring Framework called Compass which requires Ruby (https://rubyinstaller.org/downloads/).

Then install Compass from http://compass-style.org/install/.

When doing development on your project, you can run the compass watcher to keep your CSS files up to date as changes are made. Open terminal console, go to your CSS folder and execute the following:

run watch /path/to/tradestock/common/common/web/css/sass

10. Use cases

The use case model developed during screen and interaction analysis is mainly related to how users interact with the application through the screen.

All web applications have at least one human user, most often anonymous. In our project, we have three actors: visitors, registered users of the frontend TradeStock system and administrators of the system enabled to control the stocks, the submitted forms etc via backend TradeStock system.

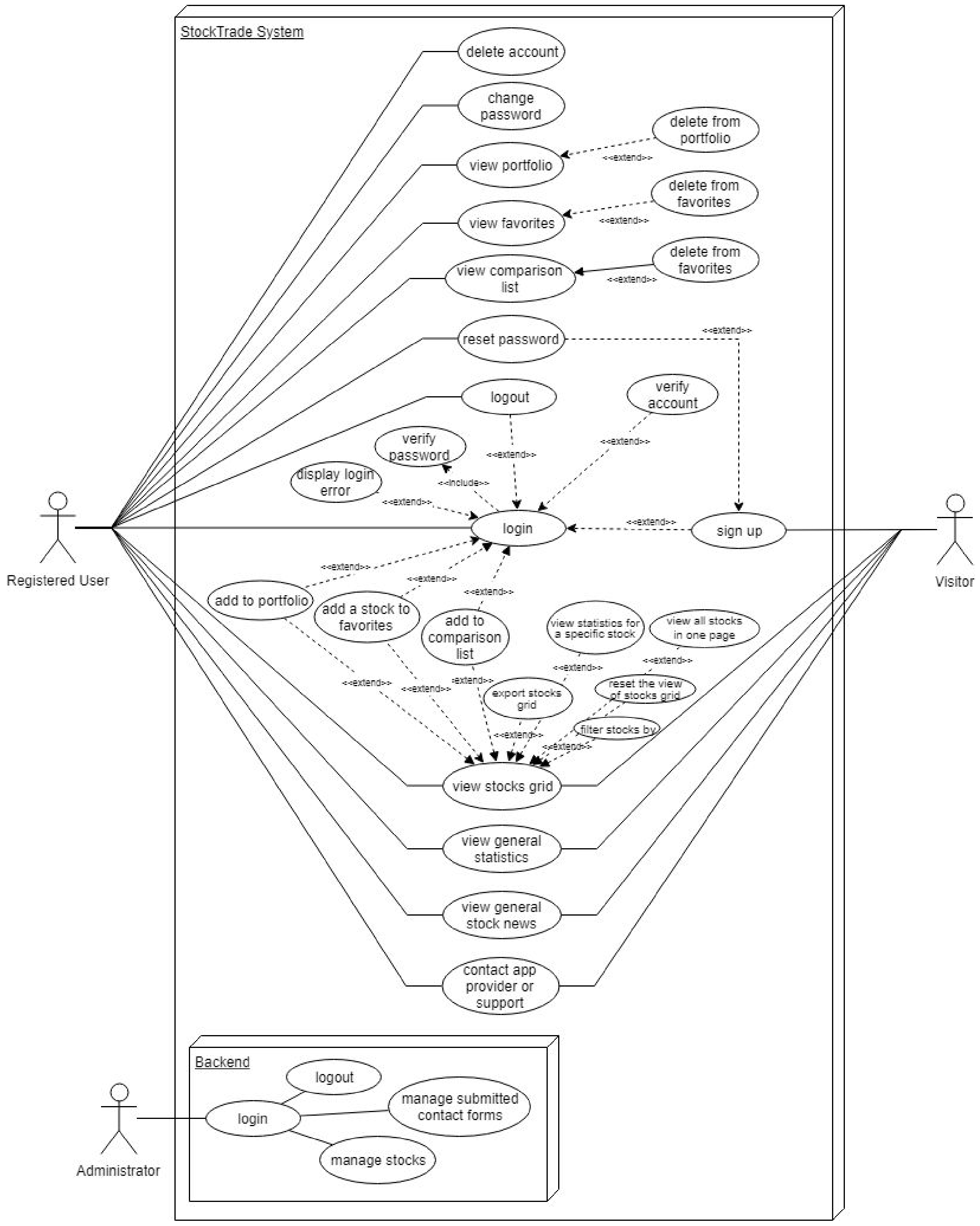
10.1. List of use cases

We have implemented the following **use cases** for our TradeStock app:

Use case	Actor / Actors
View stocks grid	Visitor, Registered user
Reset the view of stocks grid	Visitor, Registered user
Export stocks grid	Visitor, Registered user
Filter stocks by	Visitor, Registered user
View all stocks in one page	Visitor, Registered user
View statistics for a specific stock	Visitor, Registered user
View general statistics	Visitor, Registered user
View general stock news	Visitor, Registered user
Contact app provider or support	Visitor, Registered user
Sign up	Visitor, Registered user
Verify account	Visitor
Login	Registered user
Verify password	Registered user
Display login error	Visitor, Registered user
Logout	Registered user
Reset password	Registered user
Change password	Registered user
Delete account	Registered user
View favorites	Registered user
Remove from favorites	Registered user
Add a stock to favorites	Registered user
Add a stock to comparison list	Registered user
Administrator login	Administrator
Administrator logout	Administrator
Manage submitted contact forms	Administrator
Manage stocks	Administrator

10.2. Use case diagram

The following figure shows the use case diagram representing the use case model for the TradeStock system. It serves as the starting point for further modeling. Navigational requirements supplementing functional requirements are made explicit through the stereotype navigation in the use case diagram.



10.3. Sub processes use cases in textual form

The most significant widely-used sub use cases are described in textual form as follows:

1. Use Case Name		Sign Up
Actors		Visitor (= user of the application)
Preconditions		None
Normal Flow	**Description**	Visitor types his email, at least 6 character length password containing at least one lower case, one numeric character, retypes the password at the next field, types correctly the security code and presses the Sign up button. The system informs the visitor on the next screen that to be able to log in, he needs to confirm his registration. To do that he must go to his email account inbox and click at the activation link from the system.
	Postconditions	Visitor is informed about registering an account.
Alternative flows and exceptions		Visitor inputs email & password incorrectly. The validation won't let him register.
Non functional requirements		Visitor should be informed about results of input validation immediately while the validation email should be sent in 8 seconds and visitor should be informed about successful or unsuccessful Sign Up in 2 seconds most.

2. Use Case Name		Log In
Actors		Visitor
Preconditions		Sign Up
Normal Flow	**Description**	Visitor inputs his email & password; submits the input. Consequently the system informs visitor about the result -- either he is logged in and the system automatically loads user administration page or informs the visitor about unsuccessful log in.
	Postconditions	Visitor is / is not logged.
Alternative flows and exceptions		None.
Non functional requirements		Visitor should be informed about results of input validation immediately while the resulting information of successful or unsuccessful log in should appear in 2 seconds at most.

3. Use Case Name		View stock grid
Actors		Visitor
Preconditions		None
Normal Flow	**Description**	Visitor accesses /stock/index site either through main navigation bar or through URL or through user administration. Visitor can view the stock grid and interact with it.
	Postconditions	Visitor views stock grid.
Alternative flows and exceptions		None.
Non functional requirements		Data about the stocks froms servers DB should be loaded in 3 seconds most while the whole content of the webpage should be sent to the viewer(client) in 6 seconds most.

4. Use Case Name		Add a stock to favorites
Actors		Visitor
Preconditions		1) Visitor is logged in. 2) Visitor views stock grid.
Normal Flow	**Description**	Visitor adds a stock to his favored while viewing stock grid and clicks on button “Add to Favorites”. Visitor is informed about adding a stock to his favorites.
	Postconditions	Logged in visitor has the stock in his favorites.
Alternative flows and exceptions		Stock is already not accessible and visitor is informed about unsuccessful addition to his favorites.
Non functional requirements		The whole process of adding a favorite stock should take at maximum 2 seconds and it should inform the visitor accordingly.

5. Use Case Name		Logout
Actors		Visitor
Preconditions		Visitor is logged in.
Normal Flow	**Description**	Visitor clicks on log out button in main navigation bar. Visitor is logged out from his account and the system automatically loads applications home page.

	Postconditions	Logged out visitor
Alternative flows and exceptions		Visitor is already logged out and the system automatically loads applications home page.
Non functional requirements		Visitor should be informed about the results of logging out immediately while the loading of mentioned homepage should be finished in 2 seconds at most.