Montclair State University

[04/30/2018]

Inga Bondarenko, Priyanka Phapale

CSIT 691 Independent Study

Project “Fixed income portfolio management”

Instructor Prof. John Jenq

Table of Contents

I. Project Summary

The “Fixed Income Portfolio Management” project is an Independent Study for the Master Program and has two main goals for educational and practical purposes.

- The very first aim is to learn new technologies and their combination for understanding how frontend and backend development works in reality. The entire work is divided between two developers for simulation User Interface (UI) and Server Side implementation close to real practical approach when both parts are developed separately and merged at the end for final testing and release.
- The second goal is to become familiar with, at least, a minimum of understanding of the work of financial market. To become more proficient with the terminology of the industry, which could be helpful for promotions in one of the most paid field, calculations and coding, and how it could be implemented in the database structure and as views for the users.

The project is based on the simulation of buying and selling bonds on the financial market. The process is simplified for users best understanding. We use just one user, the client, to show how end users will see the entire trading process and numbers. In compare to the real operations, we are excluding involving a real brokerage account for a client, mediator processes as a trader’s actions, as well as banks related responsibilities for controlling all money processing functionality.

II. Project Requirements

Requirements Specification

Problem of Definition

The “Fixed income portfolio management” system is intended to work as a tutorial simulator for users who would like to learn how financial trades work. The research study has shown that we can find many very complicated educational systems for trading on the financial market and paid which can be very confusing for the beginners.

Purpose

So the purpose of this proposed system is to provide a fundamental and simple understanding how financial trades work for those who just started to explore this market for making investments. For this purpose, we chose bonds as an example of one of the most popular trades after stocks. This system can also be expanded in future for using all other kinds of trades.

Requirement Overview

The proposed system will enable end-users to view, analyze, sell and buy bonds and create their own very first portfolio. The data, which will be used for storing, processing and calculations, is obtained from the real financial source for the purpose to get a user more close to the reality.

Basic Requirement of the proposed system role wise defined as below.

Roles:

- Customer

Responsibilities Role wise:

Customer:

- Customer can register in the system
- Customer can buy/sell bonds on the market

- Customer can view his/her own activity on the market and historical data
- Customer can make a deposit/withdrawal from his/her account

System Flow Diagram

Below is the system flow diagram (Figure 1) which shows all of the system modules and the description for the modules (Table 1).

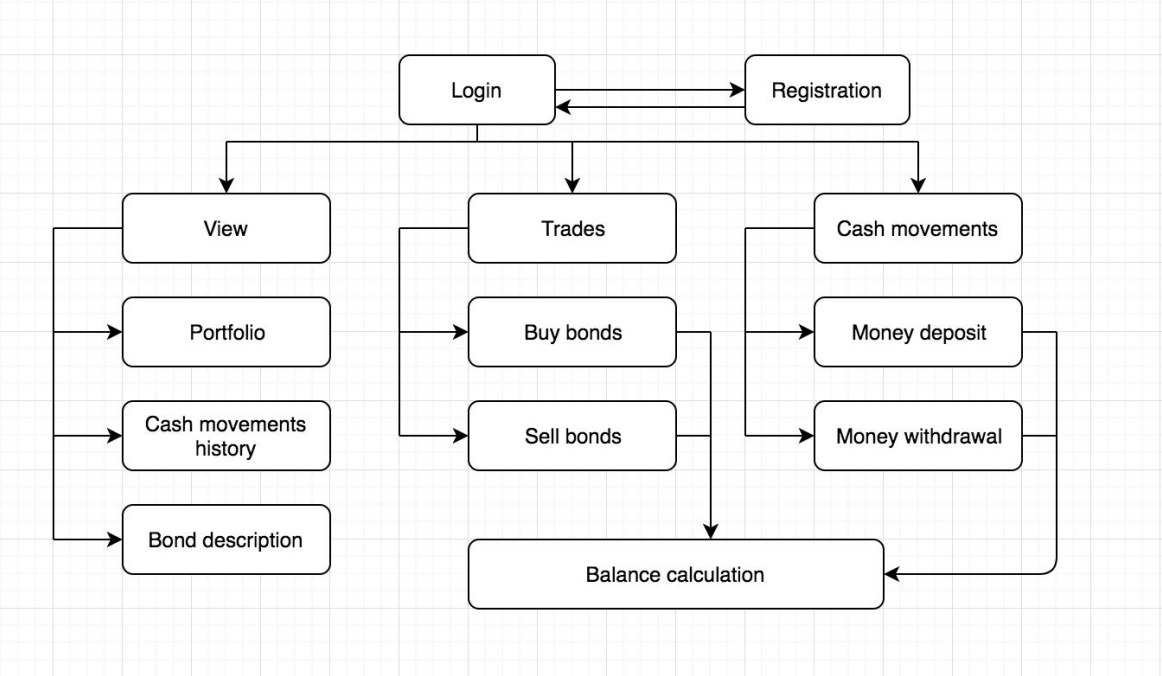


Figure 1. System Flow Diagram.

Module name	**Description**
Login	Login module should perform standard features for an intuitive interface. Client should be able to login by using email and password.
Registration	Registration module should allow to a client to input his/her last name, first name, email, password to be registered. At least one account for this user should be created at the same time.
Portfolio	The main summary for all active securities in possession.

Cash movements history	Cash movements page should display all transactions of one account ordered by date (all positive and negative movements).
Bond description	Bond description page should show all the information about selected security
Buy bonds	Buy bond module should implement a form with ability to bid the price and chose quantity of bonds to buy. All necessarily calculations and records should be inserted to the database.
Sell bonds	Sell bond module should implement a form with ability to choose quantity of bonds to sell. All necessarily calculations and records should be inserted to the database.
Money deposit	Money deposit module should implement a form with ability to add some money to the user account.
Money withdrawal	Money withdrawal module should implement a form with ability to withdraw some money from the user account with validation if the balance is sufficient.
Balance calculation	Balance calculations should be performed after each transaction and displayed on the user portfolio.

Table 1. Modules for the required system.

Use Case Diagram

Below is a Use Case diagram based on the system requirements (Figure 2).

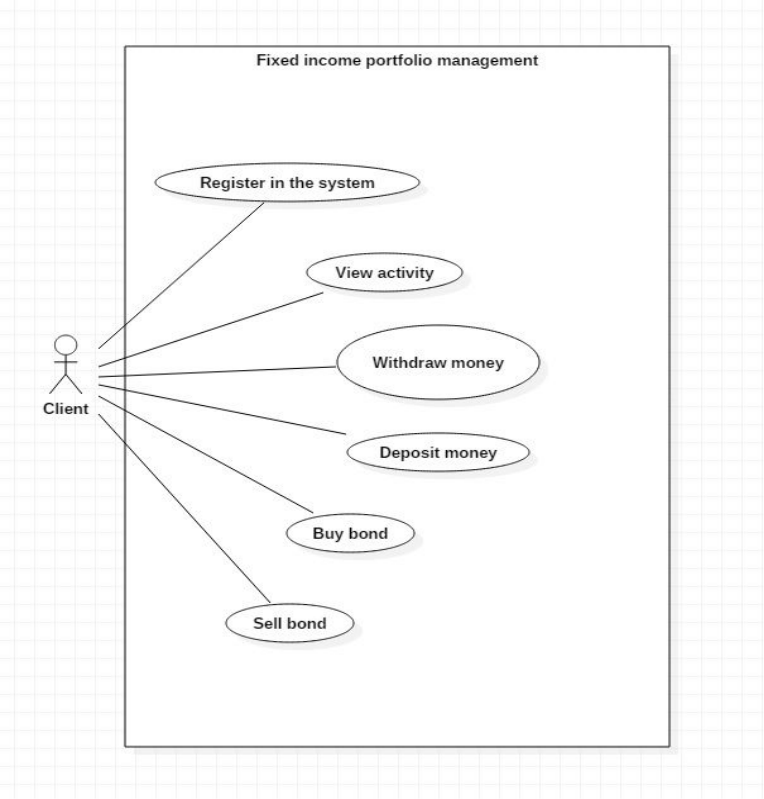


Figure 2. Use Case Diagram.

Actor:

- Client of the system

Use cases:

- Register/Login into system
- View activity
- Withdraw money
- Deposit money
- Buy bonds
- Sell bonds

III. Project development

Technologies in use

- NodeJS (nodejs.org) - is an open-source, cross-platform JavaScript runtime environment for executing JavaScript code server-side
- Angular 5 (angular.io) - is a TypeScript-based open-source front-end web application platform led by the Angular Team at Google and by a community of individuals and corporations.
- Java (java.com) - is a concurrent, class-based and object-oriented computer-programming language that.
- TomCat (tomcat.apache.org) - is an open source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies.
- Oracle Database (oracle.com) - is a multi-model database[4] management system.
- JBoss RESTful API - is a framework developed as a part of the JBoss Application Server. It implements the JAX-RS specifications. JAX-RS (Java API for RESTful Web Services) is a Java API that supports the creation of Representational State Transfer (REST) web services, using annotations.

Setting up the development environment

There are initial steps for setting up the development environment:

- Download and install Java JDK from Oracle website [1]
- Install TomCat web server from website [2]. We were using the instructions from this link http://editrocket.com/articles/tomcat_mac.html. We installed tomcat directory in the Application. So, our global variables have the next parameters:

```
export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_161.jdk/Contents/Home
export CATALINA_HOME=/Applications/tomcat
```

- As SDK we use IntelliJ Idea Community version [3].
- Install Oracle VM VirtualBox for Mac from here (to be able to install and use Oracle on Mac) [4].

- For Oracle installation from this link [5] we are choosing Database App Development VM from the table. Download from the instructions link, accept the agreement and start downloading (create a free account).
- Install Oracle SQL Developer [6].
- For Angular installation, we've used the instructions from official website [7].
- Install the Angular CLI globally by using command npm install -g @angular/cli. During the installation process, we ran into a problem that it should be done under the root user. Under the root user use the command mentioned above. The next problem we found that my installation folder wasn't granted permissions. So, we've used the next command

  chmod -R 777 /usr/local/lib/node_modules/@angular

  to change that.
- Adding Bootstrap to Angular Project. We were using the instructions from this tutorial [8]. During the installation we faced two errors, which are not mentioned on the website.

  - Error in ./NgbTabset class NgbTabset - inline template:12:20 caused by: Cannot read property 'templateRef' of undefined.

  The solution is to change from template to ng-template in app.component.html file.

  - Error: Bootstrap tooltips require Tether (http://tether.io/) at eval.

  The solution is to add this code below in index.html file above bootstrap.min.css:

  <!-- To Fix : Error: Bootstrap tooltips require Tether (http://tether.io/) at eval -->
  <script src="https://npmcdn.com/tether@1.2.4/dist/js/tether.min.js"></script>

For the frontend configuration

- Before we move to Oracle database, we were using this tutorial [9] for the initial steps.

For the backend configuration

- Launch IntelliJ and create Maven project. Use settings from this command

  mvn archetype:generate -DgroupId=edu.bonds -DartifactId=E-bonds
  -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false

- Make configurations in pom.xml: install RESTEasy API

- Make a new Run configuration. Give it a name (Ebonds), command line is tomcat7:run. Use this new run configuration to run inside tomcat. Localhost:8888.
- Create lib folder and place ojdbc8.jar in it (from sql developer installed folder, right click - show package content, then resources-sql developer-jdbc-lib). It is for the certification.
- Run the next command to use .jar file mentioned above in Execute Maven Goal

  install:install-file -Dfile=lib/ojdbc8.jar -DgroupId=com.oracle.jdbc -DartifactId=ojdbc8 -Dversion=12.2.0.1 -Dpackaging=jar -DgeneratePom=true

For the backend and frontend connection

- In the root of the frontend folder create proxy.conf.json file with the following code:

```
{
 "/api/*": {
  "target": "http://localhost:8084",
  "secure": false,
  "logLevel": "debug",
  "changeOrigin": true
 }
}
```

This code provides “tunneling” between the UI and backend (from port 4200 to 8080, where TomCat is running)

- In package.json file change the next directive as mentioned below

"start": "ng build && ng serve --proxy=proxy.conf.json"

Database structure

Below is the database structure which was created for the system needs:

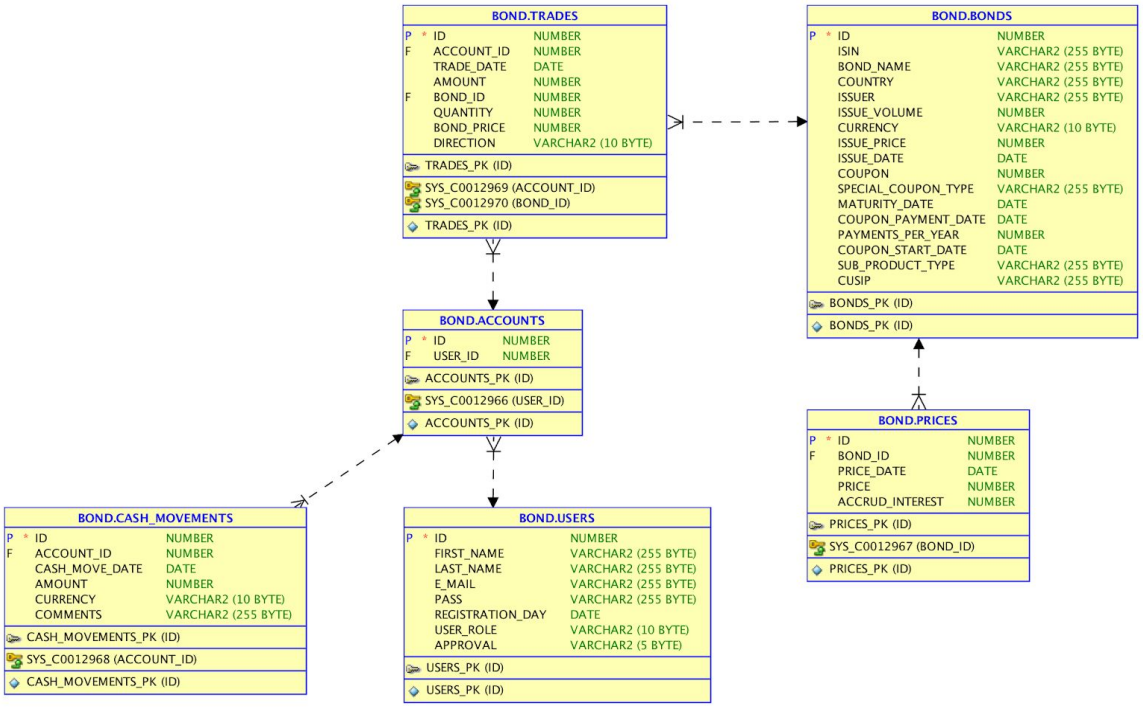


Figure 3. Database structure.

Table Name	Description
Users	Contains all user related information
Accounts	Contains all accounts information, one user can have more than one account in a real life
Bonds	Contains all descriptional information about bonds
Prices	Contains all prices related to bonds. In real life prices are getting updated every day
Trades	Contains all records regarding to all trades user made with bonds
Cash Movements	Contains all records related cash movements for user account: positive records for selling bonds and making deposits; negative records for buying bonds and money withdrawals

IV. Functionality of the project

Functionality at the final phase includes the following features listed below in the table

Function	**Description**	**Contribution**
Login/Logout	Function allows to login/logout for registered users	Inga
User registration	Function allows to register a new client. At the same time, a new account is created for a new user.	Inga
Make a deposit	Function allows to add more money to the user account	Priyanka
Withdraw money	Function allows to perform a simulation for money withdrawal with validation if balance allows to do that	Priyanka
View Client's Portfolio	A table on the client panel which summarizes all bought bonds related information	Inga
View a selected bond information/description	A page which contains all finance information and displayed in our application	Priyanka
Buy bond	Function allows to buy a bond and checks if a client has sufficient amount on his/her account	Inga
Sell bond	Function simulates a process which sells bonds on market	Inga
Client Activity	Function shows the detailed information and day by day history about client's trades, and calculates possible profit/loss.	Priyanka

V. How to deploy the Project

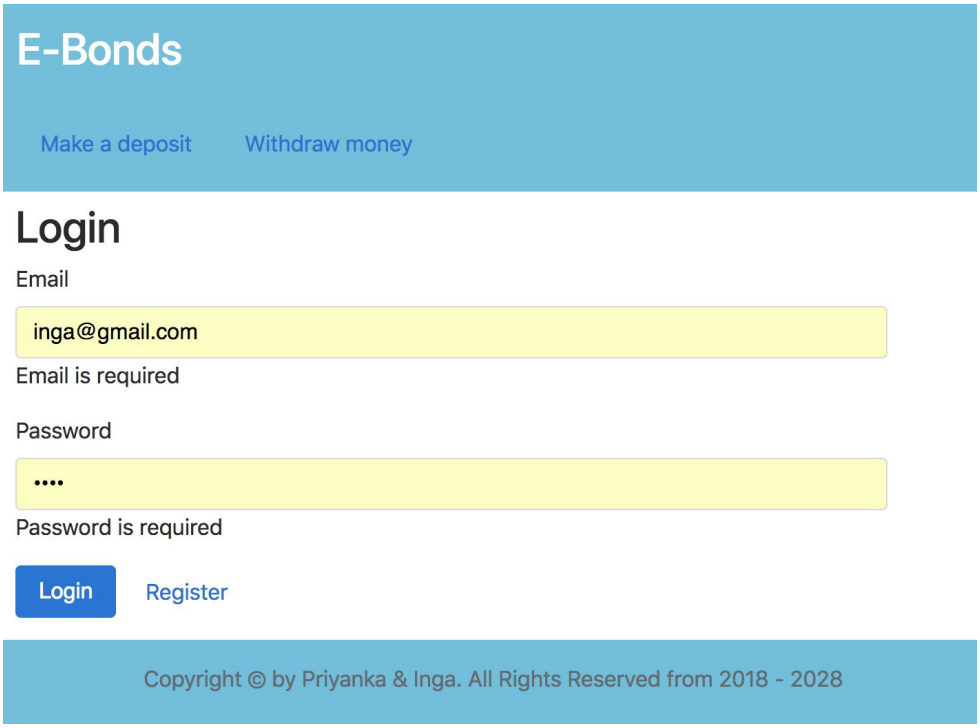
There are two options to deploy the project to Tomcat. One is to use web archive (war file), another one (mostly used for a development phase) is to use an exploded (non archived) directory structure. Our backend part is a maven project and supports both ways.

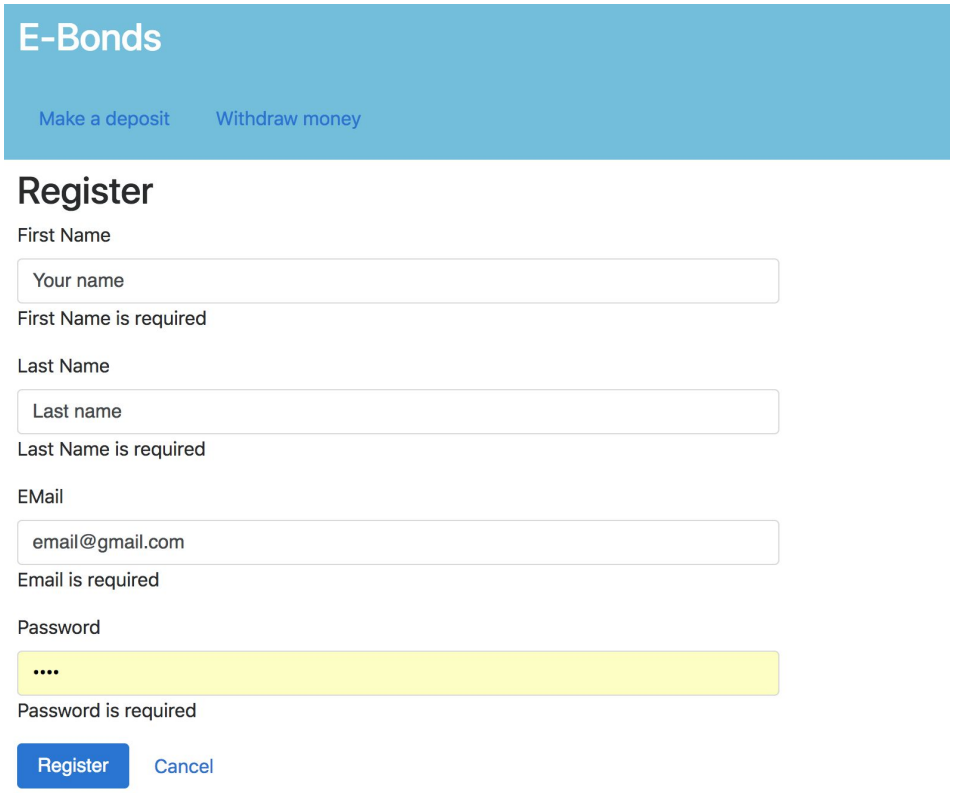
Here the following instructions we used to make a deployment. We used an exploded directory goal and completed the following steps:

- We have Maven integrated into our Intellij Idea, so we were able to edit the run configuration by using *compile war:exploded* for command line and run.
- An exploded directory was placed directly to the our target Root folder.
- At the frontend run *ng build -prod*
- Copy everything from dist folder from the frontend to the ROOT folder in Tomcat directory
- Run localhost:8080 and enjoy!

VI. User manual with test run screenshots

Below listed all major screen with description and steps to follow while using the system. All test run data also presented.

Login/Register Module
Login Page

Login page allows to use your credentials (Email and Password) to login to the system
Registration Page



Put your First name, Last name, Email and password. Register button will create a record for you in the system and a new account.

Client portfolio

The main board displays all bonds which were bought by client. All other options are available, like, selling, buying, as well as making money deposit and withdrawal. As a user you can also view a history of all trades and to date calculated balance.

Client Activity/History Panel

CASH MOVEMENT DATE	AMOUNT	COMMENTS
03-31-2018	-5	Money withdrawal
03-31-2018	-100	Money withdrawal
03-31-2018	200	Money deposit
03-31-2018	5	Money deposit
03-31-2018	200	Money deposit
03-31-2018	500	Money deposit
03-31-2018	-200	buy 2 of US Airways 6.250 10/22/2024 at 100

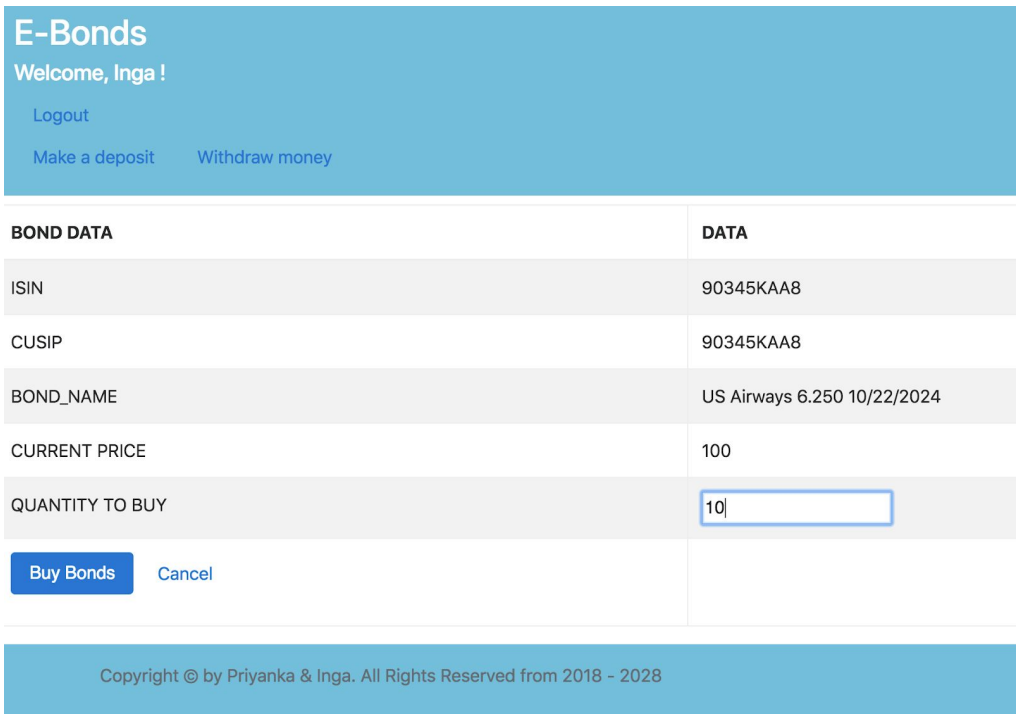
History page displays all cash movements on a client account. Negative number means buying bonds or money withdrawal. Positive number means selling bonds or money deposit. All information ordered by date (time).

Bond information Page

BOND DATA	DATA
ISIN	90345KAA8
CUSIP	90345KAA8
BOND_NAME	US Airways 6.250 10/22/2024
COUNTRY	USA
ISSUER	US Airways, Inc.
ISSUE_VOLUME	262857
CURRENCY	USD
ISSUE_PRICE	100
ISSUE_DATE	12-15-2010
COUPON	6.25
MATURITY_DATE	10-22-2024
COUPON_START_DATE	12-21-2010
COUPON_PAYMENT_DATE	
SPECIAL_COUPON_TYPE	
PAYMENTS_PER_YEAR	2
SUB_PRODUCT_TYPE	Corporate Bond

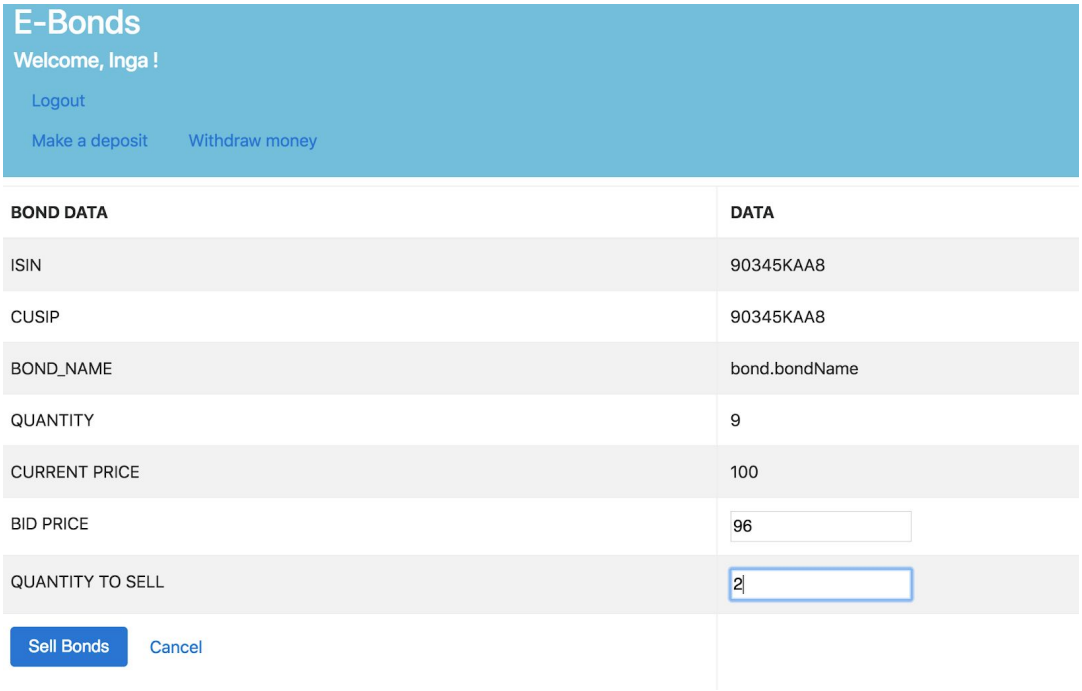
Bond information page contains all real world related information about bond just for user information.

Panel for buying bonds



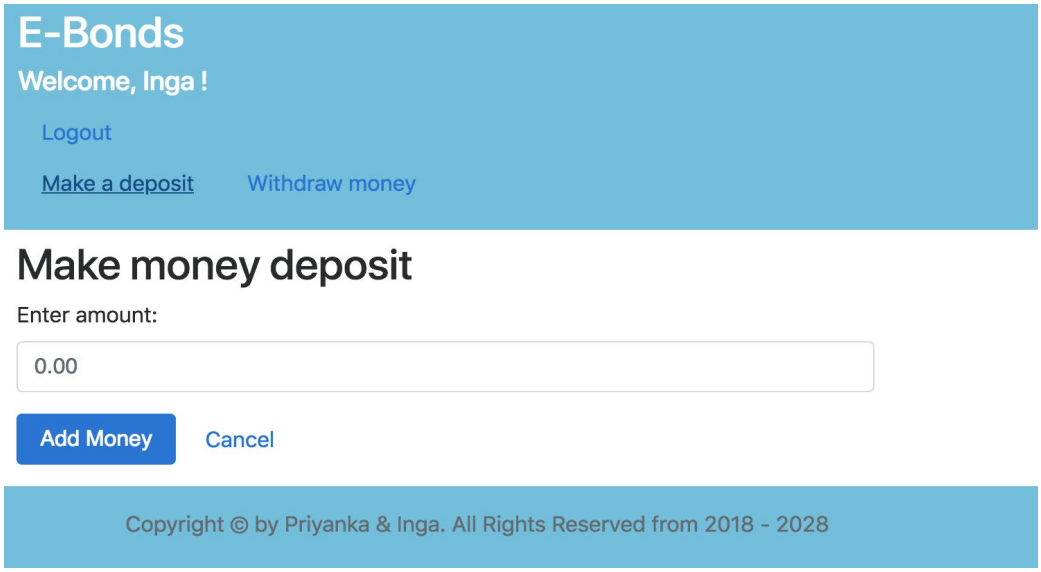
After a client have chosen an option to buy a bond, he/she will be redirected to the buying panel. This form allows to choose a quantity for bonds to be bought. All necessarily validation will be performed at the backend.

Panel for selling bonds



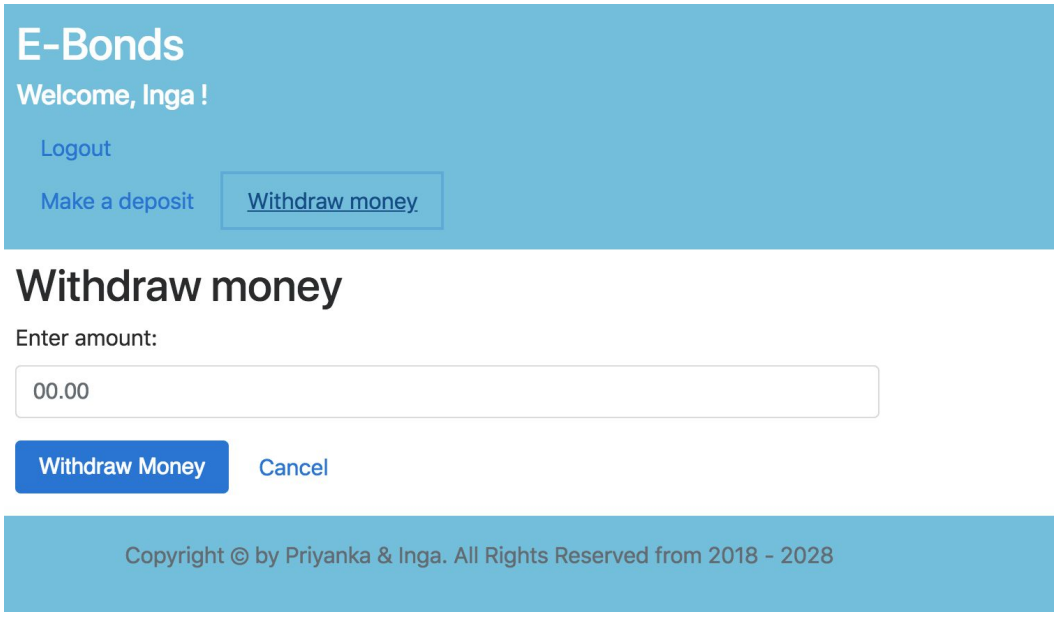
After a client have chosen an option to sell a bond, he/she will be redirected to the selling panel. This form allows to choose a quantity for bonds to be sell and the bid price. All necessarily validation will be performed at the backend.

Money deposit



Any client can add more money to his/her balance in the system

Money withdrawal



An example of money withdrawal simulation with all validations needed

IX. References.

[1]. Download Java JDK http://www.oracle.com/technetwork/java/javase/downloads/index.html
[2]. Tomcat web server http://tomcat.apache.org
[3]. Intellij Idea https://www.jetbrains.com/idea/download/#section=mac
[4]. Oracle VM VirtualBox for Mac
http://www.oracle.com/technetwork/server-storage/virtualbox/downloads/index.html
[5]. Oracle Installation
http://www.oracle.com/technetwork/community/developer-vm/index.html
[6]. Oracle SQL Developer
http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html
[7]. Angular Installation https://angular.io/guide/quickstart
[8]. Adding bootstrap to Angular Project
https://codingthesmartway.com/using-bootstrap-with-angular/
[9]. How to connect Oracle with frontend
https://coursetro.com/posts/code/84/Setting-up-an-Angular-4-MEAN-Stack-(Tutorial)