CHAPTER 1

1 Introduction

1.1 Introduction to Technologies

1.1.1 Angular 2

Angular 2 is the new improved version of the ever popular JavaScript framework Angular Js. Angular2 is a re-imagining of Angular applying all lessons learned from v1.x and promotes a component based architecture while leveraging new features of ES2015 (or Typescript) like classes and modules.

- **Speed and Performance**

  Achieve the maximum speed possible on the Web Platform today, and take it further, via Web Workers and server-side rendering.

- **Incredible Tooling**

  Build features quickly with simple, declarative templates. Extend the template language with your own components and use a wide array of existing components. Get immediate Angular-specific help and feedback with nearly every IDE and editor. All this comes together so you can focus on building amazing apps rather than trying to make the code work.

- **Loved by Millions**

  From prototype through global deployment, Angular delivers the productivity and scalable infrastructure that supports Google's largest applications.

1.1.2 Typescript

Typescript is a free and open source programming language developed and maintained by Microsoft. It is a strict superset of JavaScript, and adds optional static typing and class-based object-oriented programming to the language. Anders Hejlsberg, lead architect of C# and creator of Delphi and Turbo Pascal , has worked on the development of Typescript .Typescript may be used to develop JavaScript applications for client-side or server-side(Node.js) execution.

Typescript is designed for development of large applications and trans compiles to JavaScript .As Typescript is a superset of JavaScript, any existing JavaScript programs are also valid Typescript programs.

1.1.3 Node.js

Node.js is an open-source, cross-platform JavaScript runtime environment for executing JavaScript code server-side, and uses the Chrome V8 JavaScript engine. Historically, JavaScript was used primarily for client-side scripting, in which scripts written in JavaScript are embedded in a webpage's HTML, to be run client-side by a JavaScript engine in the user's web browser. Node.js enables JavaScript to be used for server-side scripting, and runs scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js has become one of the foundational elements of the "JavaScript everywhere" paradigm ,allowing web application development to unify around a single programming language, rather than rely on a different language for writing server side scripts. Though .js is the conventional filename extension for JavaScript code, the name "Node.js" is not referring to a particular file in this context—it's just the name of the product.

1.1.4 JavaScript

JavaScript is a high-level, dynamic, untipped and interrupted programming language .It has been standardized in the ECMAScript language specification . Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production; the majority of websites employ it, and all modern Web browsers support it without the need for plug-ins-.JavaScript is prototype-based with first-class functions, making it a multi-paradigm language,supporting object-oriented, imperative, and functional programming styles. It has an API for working with text, arrays, dates and regular expressions, but does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.

Although there are strong outward similarities between JavaScript and Java, including language name, syntax, and respective standard libraries, the two are distinct languages and differ greatly in their design. JavaScript was influenced by programming languages such as Self and Scheme.

1.1.5 Web Services

A web service is a service offered by an electronic device to another electronic device, communicating with each other via the World Wide Web. In a Web service, Web technology such as HTTP, originally designed for human-to-machine communication, is utilized for machine-to-machine communication, more specifically for transferring machine readable file formats such as XML and JSON. In practice, the web service typically provides an object-oriented web-based interface to a database server, utilized for example by another web server, or by a mobile application, that provides a user interface to the end user. Another common application offered to the end user may be a mashup, where a web server consumes several web services at different machines, and compiles the content into one user interface.

Web services may use SOAP over HTTP protocol, allowing less costly interactions over the Internet than via proprietary solutions like EDI/B2B. Besides SOAP over HTTP, web services can also be implemented on other reliable transport mechanisms like FTP. In a 2002 document, the W3C Web Services Architecture Working Group defined a Web Services Architecture, requiring a standardized implementation of a "web service."

We can identify two major classes of web services:

- REST- compliant web services , in which the primary purpose of the service is to manipulate XML representations of web resources using a uniform set of "stateless" operations; and
- Arbitrary web services, in which the service may expose an arbitrary set of operations.

— W3C, Web Services Architecture

1.1.6 XML

In computing, Extensible Mark up Language (XML) is a mark-up language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The W3C's XML 1.0 Specification and several other related specifications—all of them free open standards—define XML.

The design goals of XML emphasize simplicity, generality, and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary data structures such as those used in web services.

1.1.7 JSON

In computing, JavaScript Object Notation or JSON , is an open-standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is the most common data format used for asynchronous browser/server communication, largely replacing XML, and is used by AJAX.

JSON is a language-independent data format. It was derived from JavaScript, but as of 2017 many programming languages include code to generate and parse JSON-format data. The official Internet media type for JSON is application/json. JSON filenames use the extension .json.

1.1.8 HTTP

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, and hypermedia information systems.[1] HTTP is the foundation of data communication for the World Wide Web.

Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.

1.1.9 REST

Representational state transfer (REST) or RESTful Web services are one way of providing interoperability between computer systems on the Internet. REST-compliant Web services allow requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations. Other forms of Web service exist, which expose their own arbitrary sets of operations such as WSDL and SOAP. "Web resources" were first defined on the World Wide Web as documents or files identified by their URLs, but today they have a much more generic and abstract definition encompassing everything or entity that can be identified, named, addressed or handled, in any way whatsoever, on the Web. In a RESTful Web service, requests made to a resource's URI will elicit a response that may be in XML, HTML, JSON or some other defined format. The response may confirm that some alteration has been made to the stored resource, and it may provide hypertext links to other related resources or collections of resources. Using HTTP, as is most common, the kind of operations available include those predefined by the HTTP verbs GET, POST, PUT, DELETE and so on. By making use of a stateless protocol and standard operations, REST systems aim for fast performance, reliability, and the ability to grow, by re-using components that can be managed and updated without affecting the system as a whole, even while it is running.

1.2 Introduction to Financial Services

1.2.1 Clearing House

In banking and finance, clearing denotes all activities from the time a commitment is made for a transaction until it is settled. Clearing of payments is necessary to turn the promise of payment (for example, in the form of a cheque or electronic payment request) into the actual movement of money from one account to another.

In trading, clearing is necessary because the speed of trades is much faster than the cycle time for completing the underlying transaction. It involves the management of post-trading, pre-settlement credit exposures to ensure that trades are settled in accordance with market rules, even if a buyer or seller should become insolvent prior to settlement. Processes included in clearing are reporting/monitoring, risk margining, netting of trades to single positions, tax handling, and failure handling.

The Clearing House Association, L.L.C. is a New York-headquartered trade group and the nation's first and oldest banking association representing 24 of the world's largest commercial banks, which collectively employ over 2 million people and hold more than half of all U.S. deposits. It is a nonpartisan organization that advocates on regulatory, legislative, and legal public policy issues on behalf of its Owner Banks before policymakers, courts of law, and standards setters in the U.S. and abroad.

The Clearing House seeks a level playing field among similarly situated market participants, in which a legal and regulatory framework promotes systemic stability, economic growth, and a safe and sound banking system. Unique among trades for its sole focus on large-scale commercial banking and payments issues, the Association and its Owner Banks form strong consensus positions on issues vital to the banking industry that are technically detailed and research- and data-driven.

1.2.2 Pledge

Pledge is used when the lender (pledgee) takes actual possession of assets0028 i.e. certificates, goods). Such securities or goods are movable securities. In this case the pledgee retains the possession of the goods until the pledger (i.e. borrower) repays the entire debt amount. In case there is default by the borrower, the pledgee has a right to sell the goods in his possession and adjust its proceeds towards the amount due (i.e. principal and interest amount). Some examples of pledge are Gold /Jewellery Loans, Advance against goods,/stock, Advances against National Saving Certificates etc.

1.2.2 Release

A pledger bank will only be allowed to release collateral when there are no longer any state funds on deposit or the current market value of any remaining collateral is equal to or greater than the maintenance percentage (110% of the amount deposited by the State Treasurer with the pledger bank plus the interest due at maturity, in excess of the FDIC-insured limit).

CHAPTER 2

2 Problem Definition and Background

2.1 Existing Approach

In the existing system to perform the pledge and release process , one should follow many steps.

- Firstly the client should mention all the required details like bank details , pledge amount to the broker.
- Secondly the broker logs on to the system and requests the bank for money on behalf of the client.
- Thirdly, the backend team does the actual pledging work updating the database and managing the shares. This seems to be a very tedious process. To overcome this we have this single page collateral pledge application to ease the end user.
- Finally the history about the pledged and the released securities will be updated in another system as explained in fig 2.1.

.

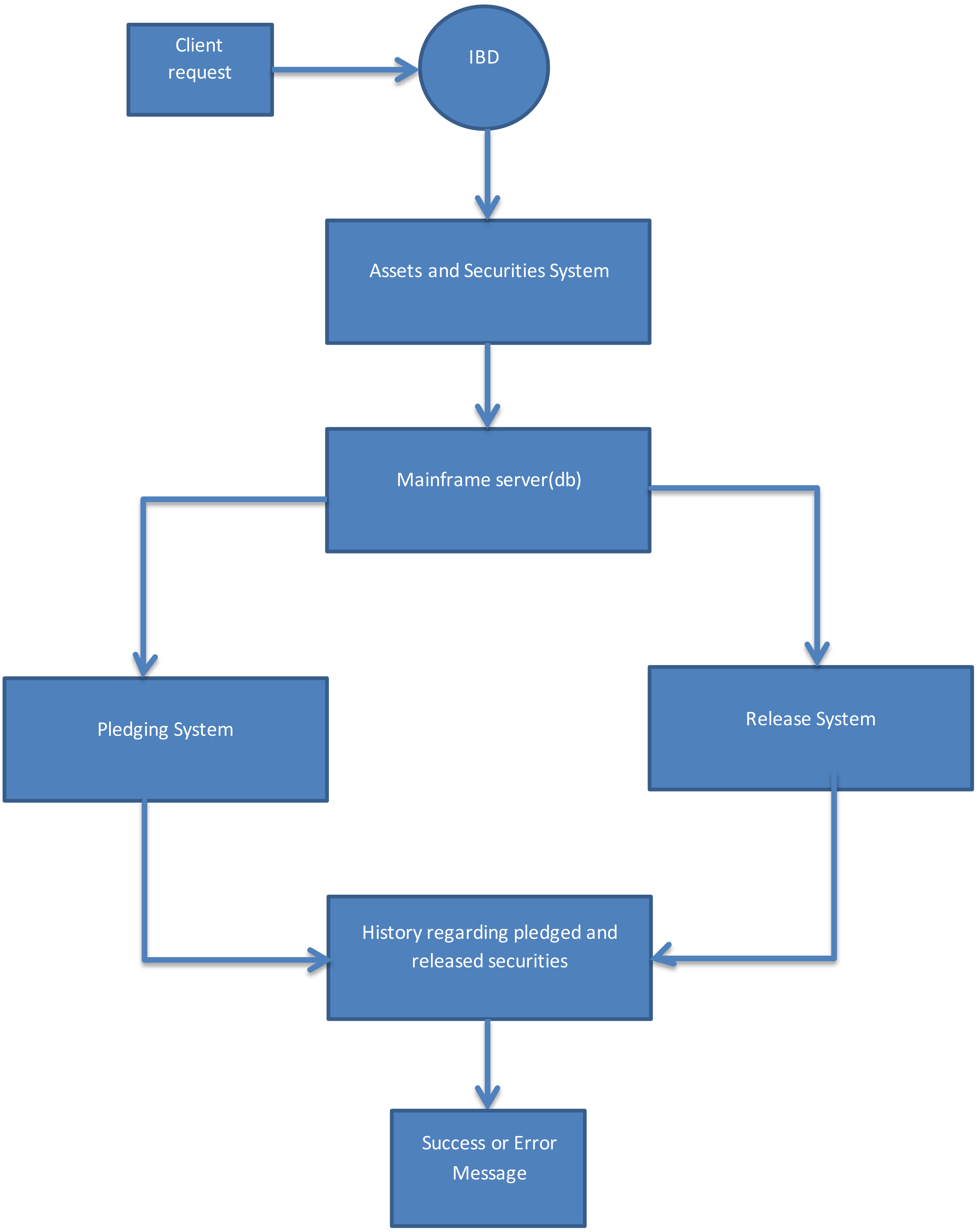


Fig:2.1 Block diagram of the existing approach

2.2 Problem Statement

A Stock Pledge is nothing but a transfer of stocks against a debt. It is an agreement between the client and the bank. The debtor pledges the shares as an asset against the amount of money taken from a lender and promises to return the amount within specific period. The debtor pledges the stocks as a security against the debt. According to the law, after the payment of the obligation the bank in which stocks are ledged must return the stocks to the debtor and the agreement stands void.

In Early, it was very difficult to manage this pledge process for the individual clients. To view the pledge details, the client has to navigate to different modules which are tedious process. Since, many components are needed to check the status of the individual user's pledge process, the convenience of the user and the response time is affected.

In order to overcome all the burden of the user, a single page is developed to fetch all the details from various components and displayed in the required format. User can specify their own requirements and can change the columns to be displayed. Every shares and securities associated with each banks for the particular broker id can be viewed with great ease. The pledging process can be scalable for existing and new client.

CHAPTER 3

3 Requirements analysis

3.1 Hardware Support

3.1.1 Client Side:

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirement list is often accompanied by a Hardware Compatibility List (HCL), especially in case of operating systems. An HCL lists tested, compatible and sometimes incompatible hardware devices for a particular operating system or application.

Table: 3.1.Hardware Specification

Processor	Intel core 2 duo and advance
Speed	2.0 GHz
Hard Disk Drive	250 GB and above.
Operating System	Windows, linux
Memory	2 GB RAM and above
System Type	32,64 bit Operating System

3.1.2. Server Side:

Mainframe:

A Mainframe Computer is a high performance Multi User computer system which is the most scalable, available, reliable and secured machine in the world capable of performing some Million Instructions per second (up to 569,632 MIPS).

Characteristics:

1) Reliable single-thread performance
2) Maximum I/O connectivity
3) Maximum I/O bandwidth
4) Reliability, Availability & Serviceability (RAS)
5) Unbreakable Security & Scalability (USS)

Z13 Mainframe:

The z13 processor has faster I/O and the ability to address up to 10144 GiB of RAIM memories -- three times as much as its predecessor. It can house up to 168 processor units in a single system and run as many as 8,000 virtual servers.

At a maximum 5GHz, the z13's processor is slower in terms of clock speed than the chip in the z12, but IBM says it more than compensates for that with other improvements. The chip has eight cores compared with six for its predecessor, and it's manufactured on a newer, 22 nanometer process, which should mean smaller, faster transistors.

The z13 supports up to 8000 Linux images simultaneously for cloud computing. For the mobile economy the z13 does real-time encryption and can process 2.5 billion transactions per day. For the z13, IBM spent over 1 billion dollars and five years of development and with more than 500 new patents

.

Table:3.2. Mainframe Specification

Available	March 9,2015
Memory	Up to 10TB
Number Of Models	5-NE1,NC9,N96,N63,N30
Channels	-PCIe Gen3 16GBps channel buses -SIX CSSs, upto 85 LPARs -4 Sub channels sets per CSS -Flash Express
Operating Systems	z/OS, z/VM, z/VSE, z/TPF, Linux on z Systems

3.2. Software Support

The list of software required for the project is:

(i) Windows Operating System

(ii) Java8

(iii)Visual Code

(iv)Eclipse IDE

(v)Bootstrap Framework

3.2.1. Windows Operating System

The project was decided to be developed using Java and R language. JVM is available for Linux, Windows, MAC, Solaris operating systems. Out of those, we decided to choose Windows 8 operating system,, as it is more stable and reliable with long term support.

3.2.2. Java8

Java is platform independent meaning project compiled in one machine can be run in any other machine that has a version of JVM available for it regardless of the underlying architecture. It is an object oriented GPL licensed programming language.

Java 8 is the only version of Java that is currently supported. Main reason for choosing Java 8 is because of ease of prototyping in it.

3.2.3 Visual Code

VS code is a new type of tool which combines the simplicity of a code editor with what developers need for their core edit-build-debug cycle. Code provides comprehensive editing and debugging support, an extensibility model, and lightweight integration with existing tools.

The general code structure looks like:

- -src
- -app
- -app.component.ts
- -app.module.ts
- -main.ts

app.component.ts :

It is the root component of what will become a tree of nested components as the application evolves.

app.module.ts:

Defines AppModule, the root module that tells Angular how to assemble the application. Right now it declares only the AppComponent.

main.ts:

Compiles the application with the JIT compiler and bootstraps the application's main module (AppModule) to run in the browser.

3.2.4 Eclipse IDE

Eclipse is an IDE (integrated Development Environment) used in computer programming, and most widely used in Java IDE. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and the primary use is for developing Java applications, but this may also be used to develop applications in other programming languages. It can also be used to develop documents with LaTeX (via a TeXlipse plug-in). Development environments include the Eclipse JDT (Java development tools) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

3.2.5 Bootstrap Framework

Bootstrap is an open-source collection of tools which is used in creating websites and web applications. It contains HTML and CSS based design templates for forms, buttons, and navigation and so on, as well as optional Java Script extensions. This aims to ease the development of dynamic websites and also web applications.

Bootstrap is a front end web framework which is an interface for the user, unlike the server-side code which lies on the "back end" or server. Bootstrap comes with several JS(JavaScript) components in the form of jQuery plugins. They provide additional UI(user interface) elements such as dialog boxes, tooltips, and so on. They also extend the functionality of existing interface elements, including auto-complete function for input fields. The following JavaScript plugins are supported by bootstrap: Dropdown, Scroll spy, Modal, Tab, Tooltip, Popover, Collapse, Alert, Button, Carousel and Type ahead.

ADVANTAGES OF BOOTSTRAP

- Ease of Use
- Highly Flexible
- Responsive Grid
- Comprehensive List of Components
- Leveraging JavaScript Libraries
- Frequent Updates
- Detailed Documentation and Vast Community
- Consistency

CHAPTER 4

4 Proposed Approach

4.1 Overview of Proposed Design

In the proposed system, we create a standalone pledger management for our clients. It is easy to create a Pledge within the system and then to apply subsequent donation receipts to the pledge. The system allows you to monitor the status of a specific pledge or to view all pledge activity as a whole on printed reports.

The main purpose is to create a Pledge Management page which is used by administrators and staff members to manage pledge records. From this page, you can view clients who have pledged to a specific fund, add and delete pledge records, view pledge details, and make adjustments to existing pledges. The organization administrator can see all funds and their associated pledges while staff members can see only those pledges associated with the funds they granted permission to access. Every clients will be having their own broker dealer and every dealer will have their own unique IBD .Using that IBD, clients can either pledge or release shares and securities.

4.2 BLOCK DIAGRAM:

4.2.1 Control Flow

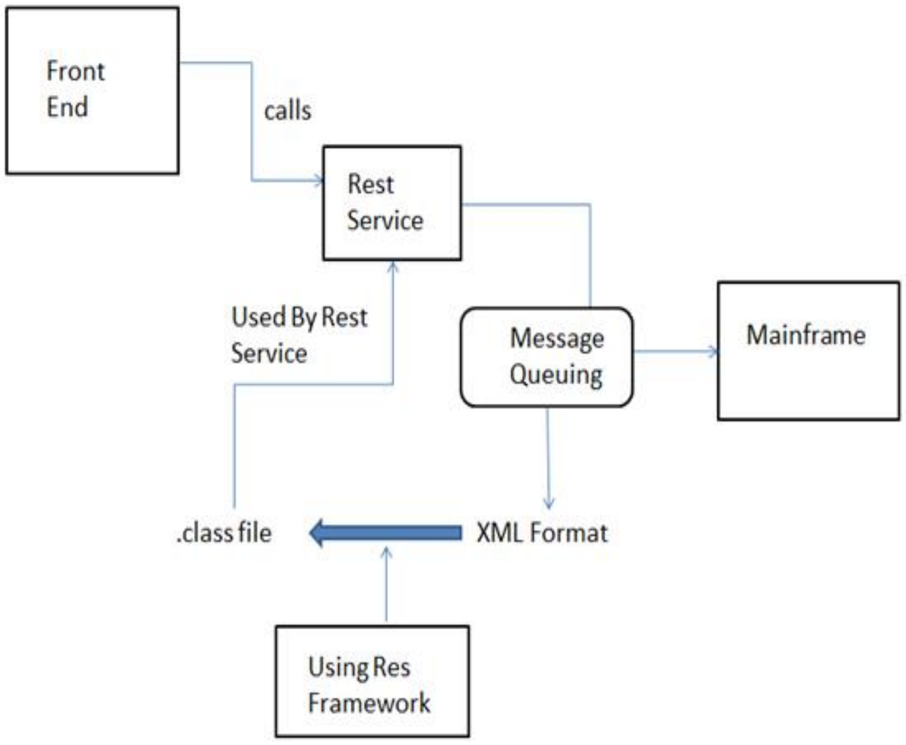


Fig:4.1 Block diagram showing flow of control from front end to back end mainframe

The flow in which the client interacts with the user interface shown in fig 4.1. Initially, upon client request, the front end calls the rest service. The REST service also called as web service is used because of its low response rate when compared to SOAP API. To fetch data, REST API calls the Mainframe. Here message queuing is used to retrieve data from the mainframe, and the format is XML. A framework called Res Framework is used which generates equivalent java class file for the XML file. This class objects can be used by the initially invoked REST Service. The REST service returns the object in the JSON format. Angular JS 2 technology is implemented in viewing the retrieved data in single page application.

4.2.1 Pledge and Release

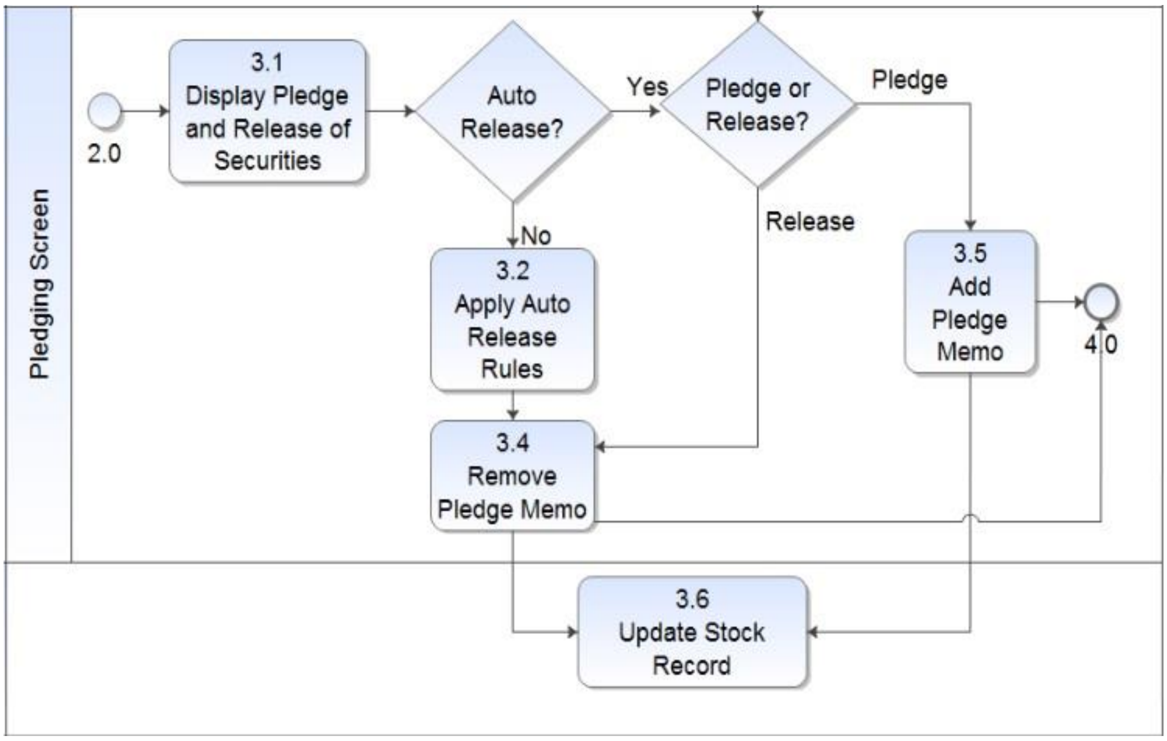


Fig 4.2: Flowchart of Pledging and Release Screen

. In the first place, the broker selected by the client sees the pledge and release security records under each bank. There is an option of Auto Release. In Auto Release, After 24 hours of time, the pledged amount will be automatically released and the amount is deducted from the client account. If Auto release is not applied, then the broker can release the pledged stock manually. After the above mentioned process is completed, the record is updated to view the changes made, as shown in fig 4.2.

CHAPTER 5

5 Interaction Scenario

5.1 Class Diagram

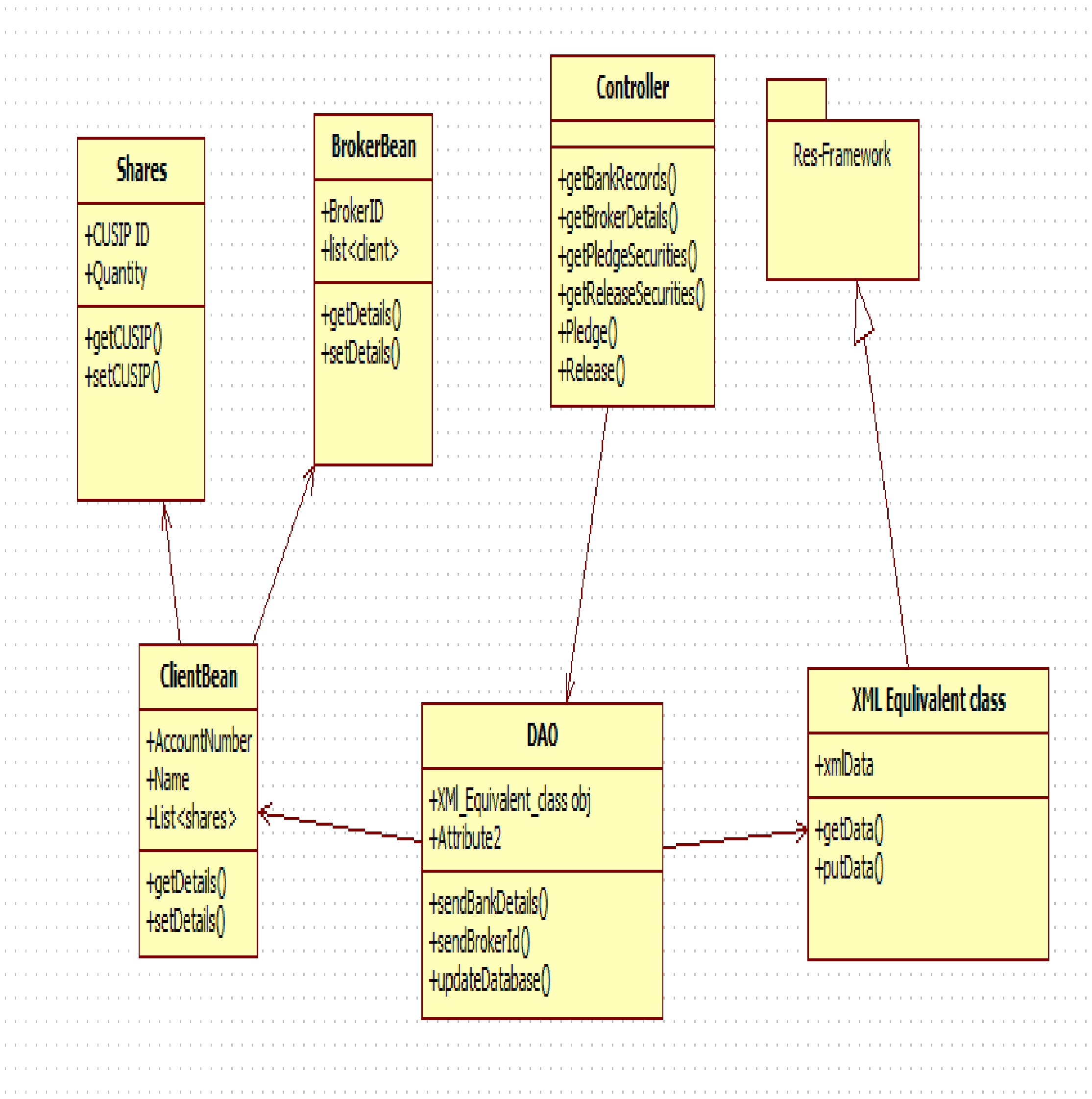


Fig 5.1: Class Diagram for Pledging and Releasing Process

Six classes are available namely, Shares, BrokerBean, Controller, ClientBean, DAO, XMLEquivalentClass. These six classes form main contents of project and help in achieving goals.

Each class has several attributes which determines properties of class respectively. These classes interact with each other for successful functioning of the application, as explained in fig 5.1.

Class Diagram Process Flow:

1. Controller to DAO.

2. Class Diagrams contains Attributes and Functions.

EXAMPLE:

- ➢ DAO contains attributes and functions as follows:
- ➢ Attributes:
    - XMl_Equivalent_class obj
- ➢ Functions:
    - sendBankDetails()
    - sendBrokerId()
    - updateDatabase()

3. In this way all the class Diagrams are linked :

    - Controller to DAO.

- DAO uses ClientBean, Shares , BrokerBean, XMLEquivalentClass.
- ClientBean will have Shares and BrokerBean Object.
- XMLEquivalentClass uses classes present is Res Framework package.

5.2 Use Case Diagram

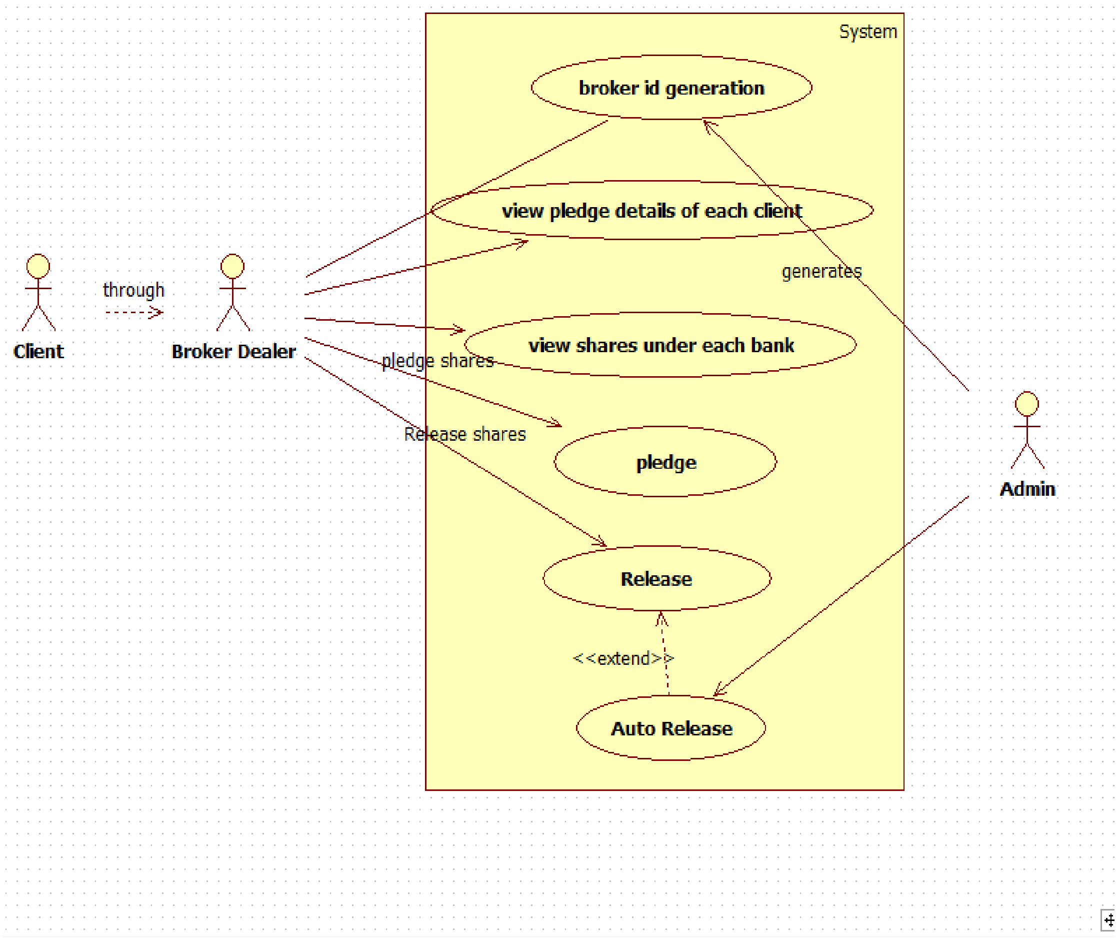


Fig: 5.2 : Use case Diagram for Pledging and Releasing

Client, broker and admin are the actors who are interacting with the application software. The client interacts with the application through a broker who acts as a mediator to pledging and releasing process.

Since, same transaction can be performed concurrently in two or more systems, some transaction may fail. For every unsuccessful process, an error popup message is thrown with corresponding error statements. The log of every unsuccessful transactions are saved and can be viewed later, shown in fig 5.2

Use case Diagram Process flow:

1. Client to Broker Interaction.

2. Unique Broker id will be generated.

3. Broker views all the pledge details of each client.

4. Broker places the shares in the pledge.

5. Broker can also have the option to release the shares from the bank.

6. Finally there is an Auto release option to make the release of equities with 24 hours from the stocks pledged.

5.3 Activity Diagram

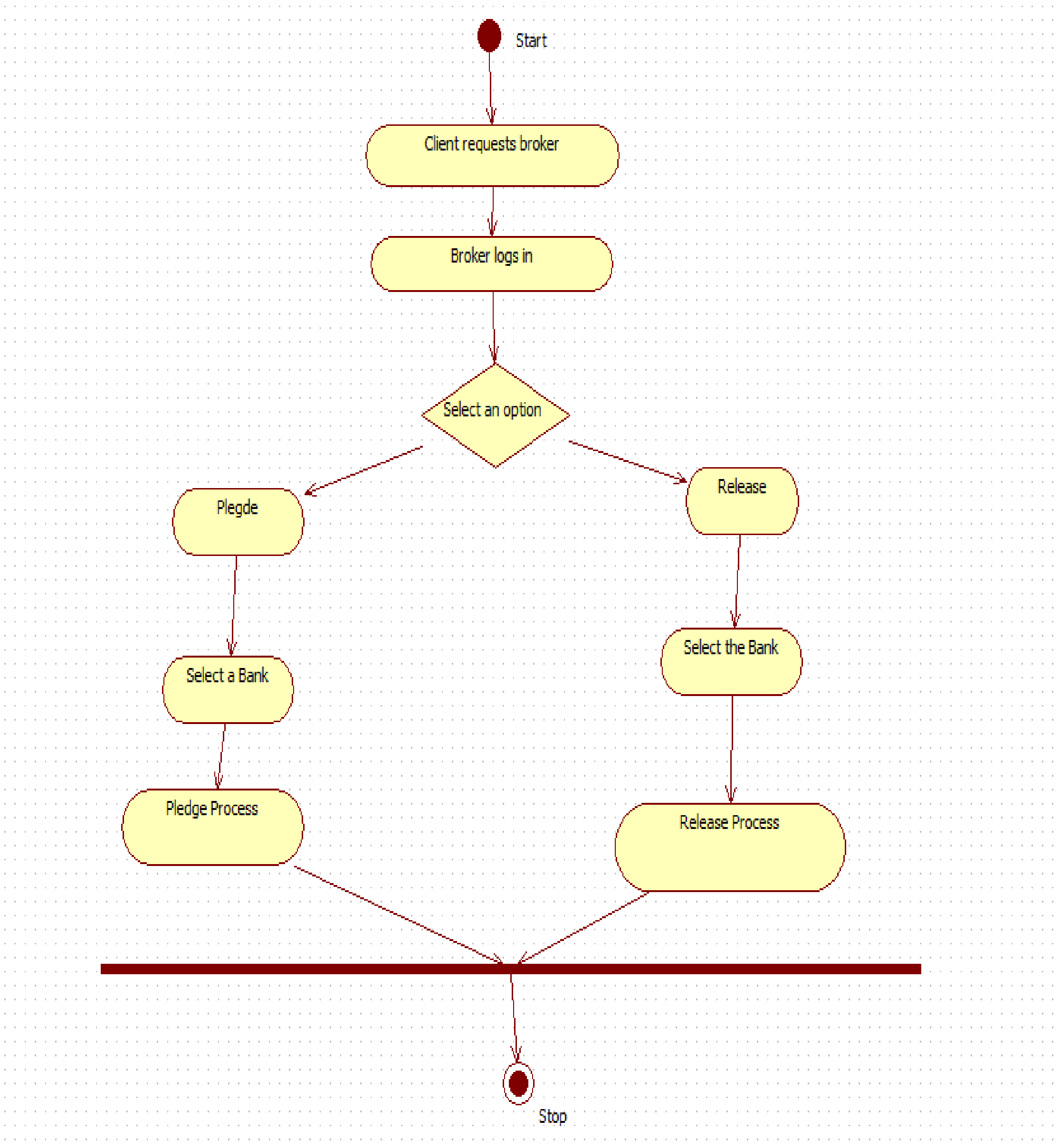


Fig 5.3 Activity Diagram for overall pledging and releasing process.

Activity Diagram Process flow:

1. Client requests Broker and gives required information.

2. Broker logs in to the application.

3. If the Pledge option is selected, the available bank is listed down along with the details of all the client's shares under the specific bank.

4. In pledge process, the clients' shares are given to the bank and in return bank gives the money to the client based on share value.

5. If release option is selected, then broker selects the user share's CUSIP ID to be released and proceeds with Release process.

6. In release process the client pays the money back to the bank and takes back his shares. When failed to lend the money to the bank, then the bank puts penalty on the client which is to be paid, as explained in fig 5.3.

5.4 Sequence Diagram

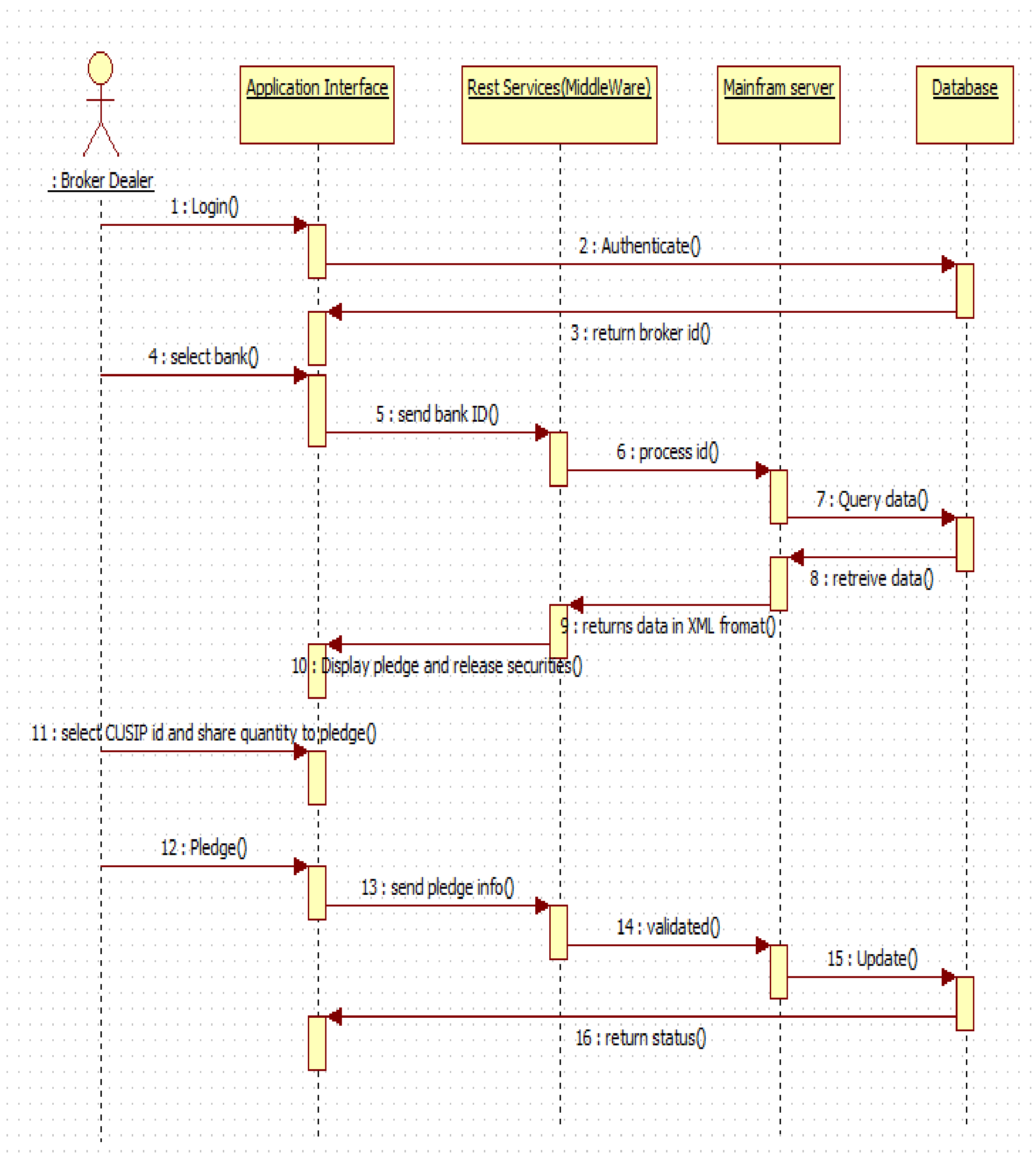


Fig 5.4 Sequence Diagram for Pledging and Release Process

Sequence Diagram process flow:

1. End user(broker dealer) to Application Interface.

2. End user provides inputs.

3. The inputs are sent to REST services in JSON format.

4. Mainframe server processes this inputs from REST Services.

5. Database gets updated based on the inputs provided.

6. Finally, the status of the transaction is returned back to the end user, as explained in fig 5.4.

5.5 Collaboration Diagram

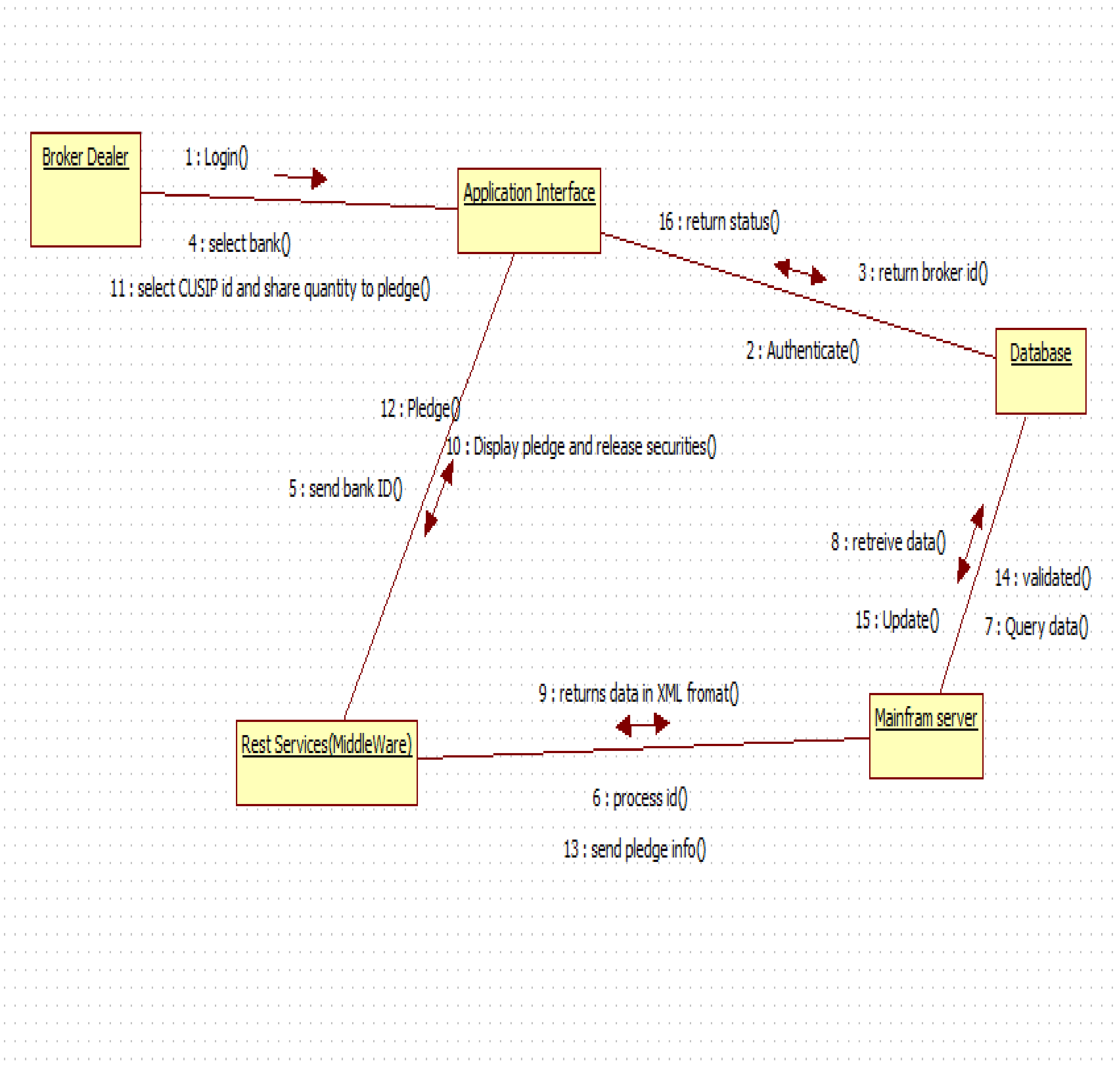


Fig 5.5: Collaboration Diagram for Pledging and Releasing Process

Process Flow:

1. Broker Dealer logs on to application interface and authentication is done with the database.

2. A Broker id is returned to the application interface from the database.

3. Broker Dealer selects the bank and sends to middle ware which sends the process id to mainframe.

4. With the given requirements a query is made to the database which returns the data as output.

5. The data is returned from mainframe to middle ware in XML format

6. The Interface displays the pledge and release securities.

7. Now the broker dealer selects the cusip id and the quantity of shares to perform pledge.

8. The pledge request is sent to REST Service.

9. The information of the pledge is sent to mainframe.

10. The requested details that are sent to the database are validated.

11. Finally if the validation is true then the database gets updated and the status report is sent back to the application interface, shown in fig 5.5

CHAPTER 6

6. Implementation

6.1 npm and node js Installation

Node.js and npm are essential to modern web development with Angular2 and other platforms. Node powers client development and build tools. The npmpackage manager, itself a node application, installs JavaScript libraries.

Version:

- **Node js v4.x.x or higher**
- **npm 3.x.x or higher**

The versions of node and npm can be checked using :

- **node -v**
- **npm –v**

Older versions produce errors.

After installing node and npm, the following steps are performed

1. Create a project folder named quickstart.

2. Clone the Quick Start seed into project folder.

3. Install npm packages.

4. Run npm start to launch the sample application.

git clone https://github.com/angular/quickstart.git quickstart

The above command is given in GIT bash console to clone the quickstart folder from the GitHub link. We can also download it separately and save it in a folder.

cd quickstart

Switch to the quickstart folder

npm install

Install all the packages necessary to run Angular JS 2 application

npm start

Once the command is executed, the npm server is started and https://localhost:3000 path is set by default.

6.1.1 Code Snippet

The following code snippet is used to display the table in front end using Angular 2 technology:

app.component.js

```
angular.module('ngTableTutorial', ['ngTable']).controller('tableController',
function ($scope, $filter, ngTableParams)

{

        $scope.records=/*data fetched from rest services*/;

        $scope.usersTable = new ngTableParams({

              page: 1,

              count: 10

           }, {

           total:$scope.records.length,getData:function ($defer, params)

                 {

                       $scope.data=params.sorting()?
                       $filter('orderBy')($scope.records, params.orderBy()) :
                       $scope.records;

                       $scope.data = $scope.data.slice((params.page() - 1) *
                       params.count(), params.page() * params.count());

                    $defer.resolve($scope.data);

                 } });});
```

ngtable.html

```
<div ng-controller="tableController">

<table ng-table="releaseTable" showfilter=”true” class="table table-
striped">

  <tr ng-repeat="row in data">

      <td data-title="'CUSIP'" >{{row.cusip}}</td>

      <td data-title="'SYMBOL'" >{{row.symbol}} </td>

      <td data-title="'DESCRIPTION'" >{{row.description}}</td>

     <td data-title="'RELEASE QUANTITY" >{{row.release_quantity}}

      </td>

      <td data-title="'PLEDGE QUANTITY'">{{row.pledge_quantity}}

      </td>

      <td data-title="'PLEDGE VALUE'" >{{row.pledge_value}}</td>

       <td data-title="'PLEDGE BANK'" >{{row.pledge_bank}}</td>

 </tr>

</table>

</div>
```

6.2 Interfacing Client FrontEnd and RESTful Services Using Spring MVC framework:

6.2.1 SPRING MVC

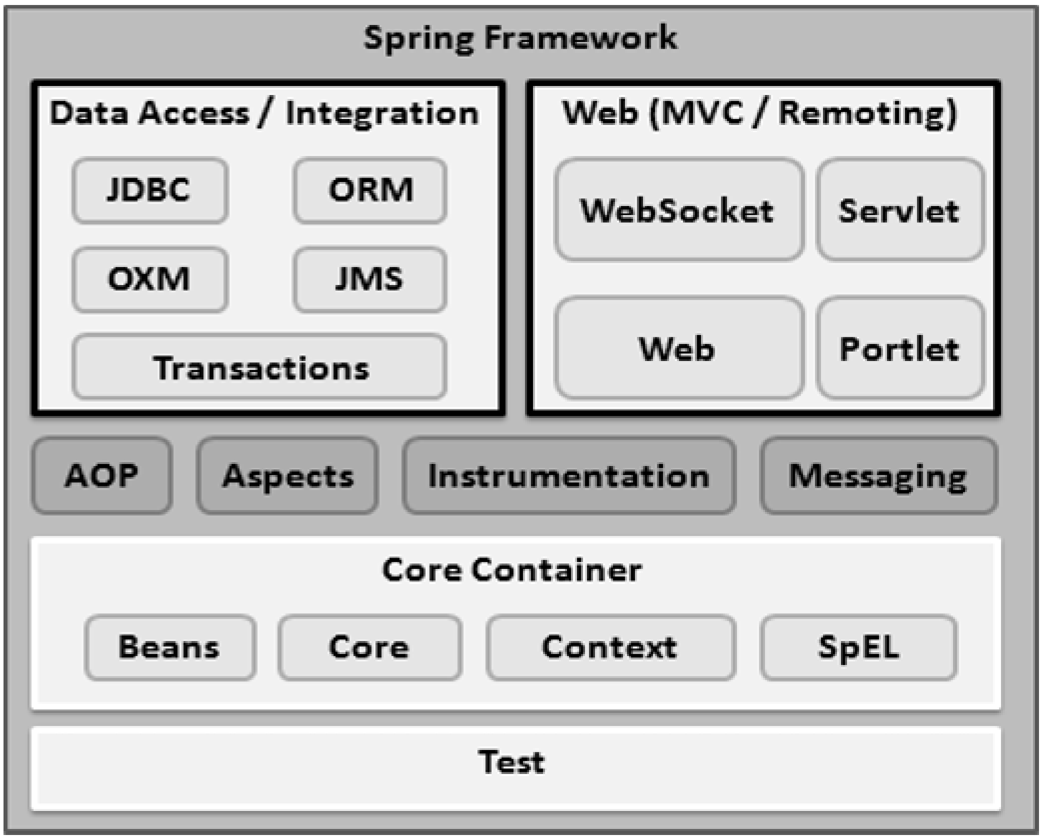


Fig: 6.1. Spring Framework

The spring web MVC framework provides model-view-controller architecture and ready components that can be used to develop flexible and loosely coupled web applications. The MVC pattern results in separating the different aspects of the application (input logic, business logic, and UI logic), while providing a loose coupling between these elements, explained in fig 6.1.

Model - encapsulates the application data and in general they will consist of POJO.

View - responsible for rendering the model data and in general it generates HTML output that the client's browser can interpret.

Controller - responsible for processing user requests and building appropriate model and passes it to the view for rendering.

6.2.2 Interfacing RESTful Services and Backend Mainframe Server Using Message Queuing:

The following are the configuration files used :

- Pom.xml
- Spring-Servlet.xml
- Web.xml

Pom.xml:

This file contains all the dependencies that are required to get the application work. The code dependencies and the properties tag are given below.

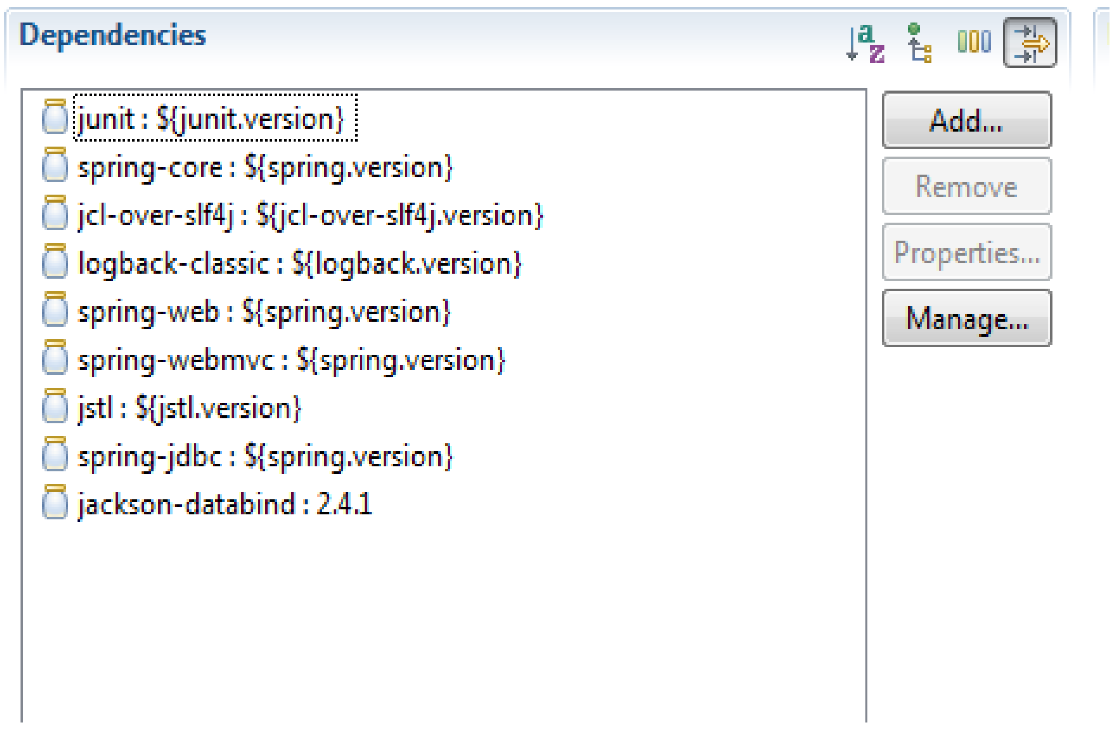
```
<projectxmlns="http://maven.apache.org/POM/4.0.0"xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>pledge</groupId>
  <artifactId>pledge.test</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <name>Test Pledge Maven Webapp</name>
  <url>http://maven.apache.org</url>
  <properties>
                <jdk.version>1.7</jdk.version>
                <spring.version>4.1.1.RELEASE</spring.version>
                <jstl.version>1.2</jstl.version>
                <junit.version>4.11</junit.version>
                <logback.version>1.0.13</logback.version>
                <jcl-over-slf4j.version>1.7.5</jcl-over-slf4j.version>
        </properties>
```

```
<dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>${junit.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>${spring.version}</version>
            <exclusions>
              <groupId>commons-logging</groupId>
              <artifactId>commons-logging</artifactId>
            </exclusions>
        </dependency>

</dependencies>
<build>
   <finalName>Test</finalName>
  </build>
</project>
```

Maven Dependency :

Maven is a powerful tool that allows users to import dependencies into their software projects and also automatically manage transitive dependencies. In order to use Maven, it is necessary to explicitly add dependencies to the Maven pom.xml file. Once added to the Maven pom.xml file, dependencies will be automatically downloaded as shown in fig 6.2, updated, and have their sub-dependencies managed by Maven.



junit : ${junit.version}
spring-core : ${spring.version}
jcl-over-slf4j : ${jcl-over-slf4j.version}
logback-classic : ${logback.version}
spring-web : ${spring.version}
spring-webmvc : ${spring.version}
jstl : ${jstl.version}
spring-jdbc : ${spring.version}
jackson-databind : 2.4.1

Fig: 6.2. Dependency injection

Spring-servlet.xml:

```
<beans>

        xmlns="http://www.springframework.org/schema/beans"

        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

        xmlns:context="http://www.springframework.org/schema/context"

        xmlns:mvc="http://www.springframework.org/schema/mvc"

        xmlns:tx="http://www.springframework.org/schema/tx"

        xsi:schemaLocation="http://www.springframework.org/schema/mv
        c http://www.springframework.org/schema/mvc/spring-mvc.xsd

        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd

        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx-4.1.xsd

        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-
        context.xsd”

</beans>
```

The above shown configuration file contains a beans tag which is used to download the required packages from the url provided.

6.3 Setting up the Initial Software and configuring all the configuration files

In the Eclipse IDE, all the API files like servlet.api are added into the build path. The configuration files like pom.xml, spring-servlet.xml, and web.xml are edited. All the required files are included. The created project is converted into Maven which helps in downloading the required dependencies from the internet. The versions of the required dependency are mentioned in pom.xml file.

6.4 Performing the Pledging Process

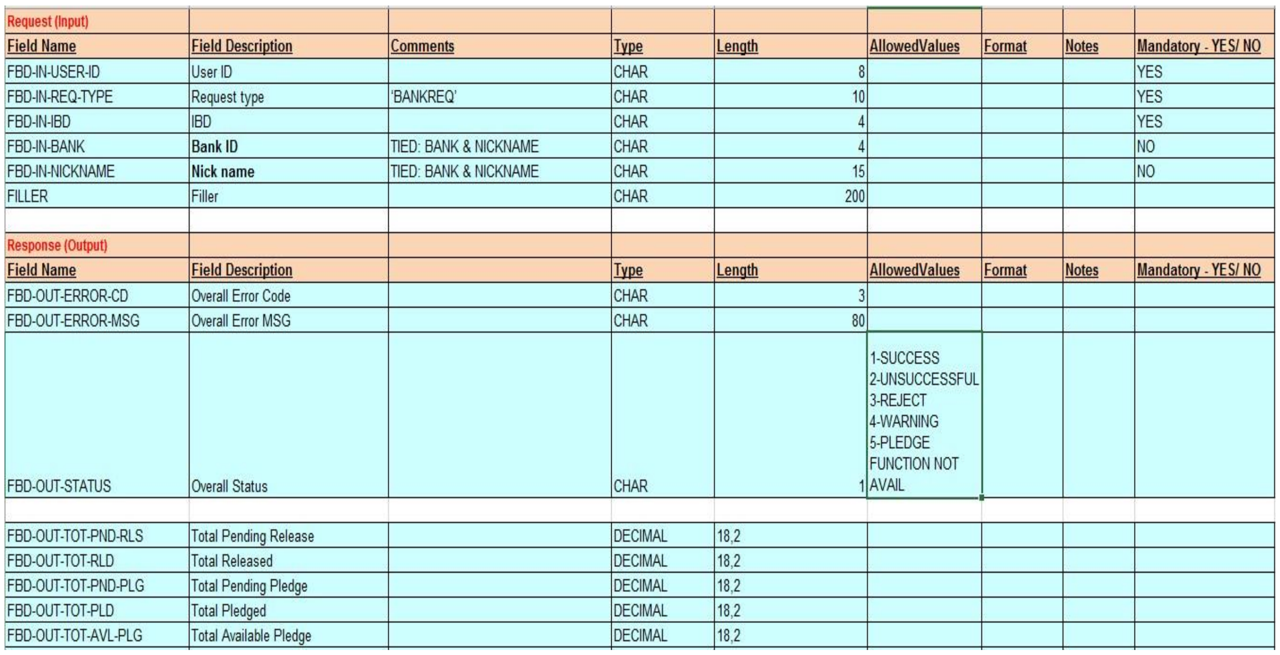
Initially the broker selects the bank to perform the pledging process and the user interface displays the pledge and release security policies.

The Broker then selects the CUSIP id of the client shares and enters the quantity of each share to pledge. If the quantity of the shares to be pledged under the selected bank is less than or equal to the available quantity, then the stocks are successfully pledged and given to the bank.

Here, bank becomes the owner of the stocks and client borrows money from the bank. If the quantity is not available, then the application prompts the popup message stating insufficient quantity and it is not accepted.

Data from the front end is integrated with the backend Mainframe server using the Messaging Queue as a middleware service.

In the messaging queue the data in sent as the JSON object using the pLink function, in this a specified format is designed through which the data need to be sent.



Request (Input)								
Field Name	Field Description	Comments	Type	Length	AllowedValues	Format	Notes	Mandatory - YES/ NO
FBD-IN-USER-ID	User ID		CHAR	8				YES
FBD-IN-REQ-TYPE	Request type	'BANKREQ'	CHAR	10				YES
FBD-IN-IBD	IBD		CHAR	4				YES
FBD-IN-BANK	Bank ID	TIED: BANK & NICKNAME	CHAR	4				NO
FBD-IN-NICKNAME	Nick name	TIED: BANK & NICKNAME	CHAR	15				NO
FILLER	Filler		CHAR	200				
Response (Output)								
Field Name	Field Description		Type	Length	AllowedValues	Format	Notes	Mandatory - YES/ NO
FBD-OUT-ERROR-CD	Overall Error Code		CHAR	3				
FBD-OUT-ERROR-MSG	Overall Error MSG		CHAR	80				
FBD-OUT-STATUS	Overall Status		CHAR	1	1-SUCCESS 2-UNSUCCESSFUL 3-REJECT 4-WARNING 5-PLEDGE FUNCTION NOT AVAIL			
FBD-OUT-TOT-PND-RLS	Total Pending Release		DECIMAL	18,2				
FBD-OUT-TOT-RLD	Total Released		DECIMAL	18,2				
FBD-OUT-TOT-PND-PLG	Total Pending Pledge		DECIMAL	18,2				
FBD-OUT-TOT-PLD	Total Pledged		DECIMAL	18,2				
FBD-OUT-TOT-AVL-PLG	Total Available Pledge		DECIMAL	18,2				

Fig: 6.3. Data format

Fig 6.3 shows the format through which the data is send to the backend mainframe . pLink automatically sorts the message that is send to message queue.

6.5 Performing the Release Process

Initially the broker selects the bank to perform the release process and the user interface displays the pledge and release security policies.

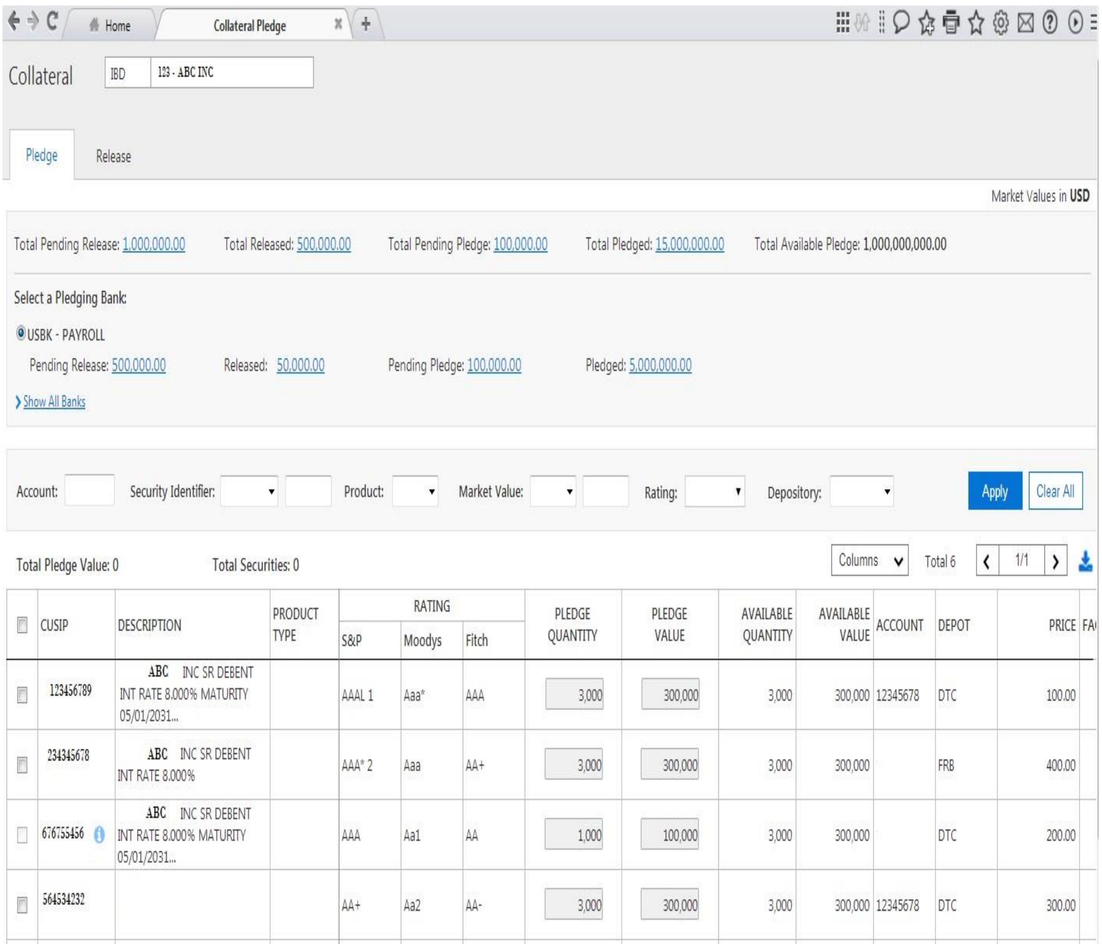
The Broker then selects the CUSIP id of the client shares and enters the quantity of each share to release. If the quantity of the shares to be released under the selected bank is less than or equal to the pledged quantity, then the stocks are successfully released and given back to the client.Here, bank returns the stocks back to the owner upon receiving the money. Once the release process is done, the amount is deducted from the client's account depends on the number of share they released.

Since, same transaction can be performed concurrently in two or more systems, some transaction may fail. For every unsuccessful process, an error popup message is thrown with corresponding error statements. The log of every unsuccessful transactions are saved and can be viewed later.

CHAPTER 7

7 Results

7.1 Pledge


Home | Collateral Pledge

Collateral IBD 123 - ABC INC

Pledge | Release

Market Values in USD

Total Pending Release: 1,000,000.00 Total Released: 500,000.00 Total Pending Pledge: 100,000.00 Total Pledged: 15,000,000.00 Total Available Pledge: 1,000,000,000.00

Select a Pledging Bank:

● USBK - PAYROLL

Pending Release: 500,000.00 Released: 50,000.00 Pending Pledge: 100,000.00 Pledged: 5,000,000.00

› Show All Banks

Account: Security Identifier: Product: Market Value: Rating: Depository: Apply Clear All

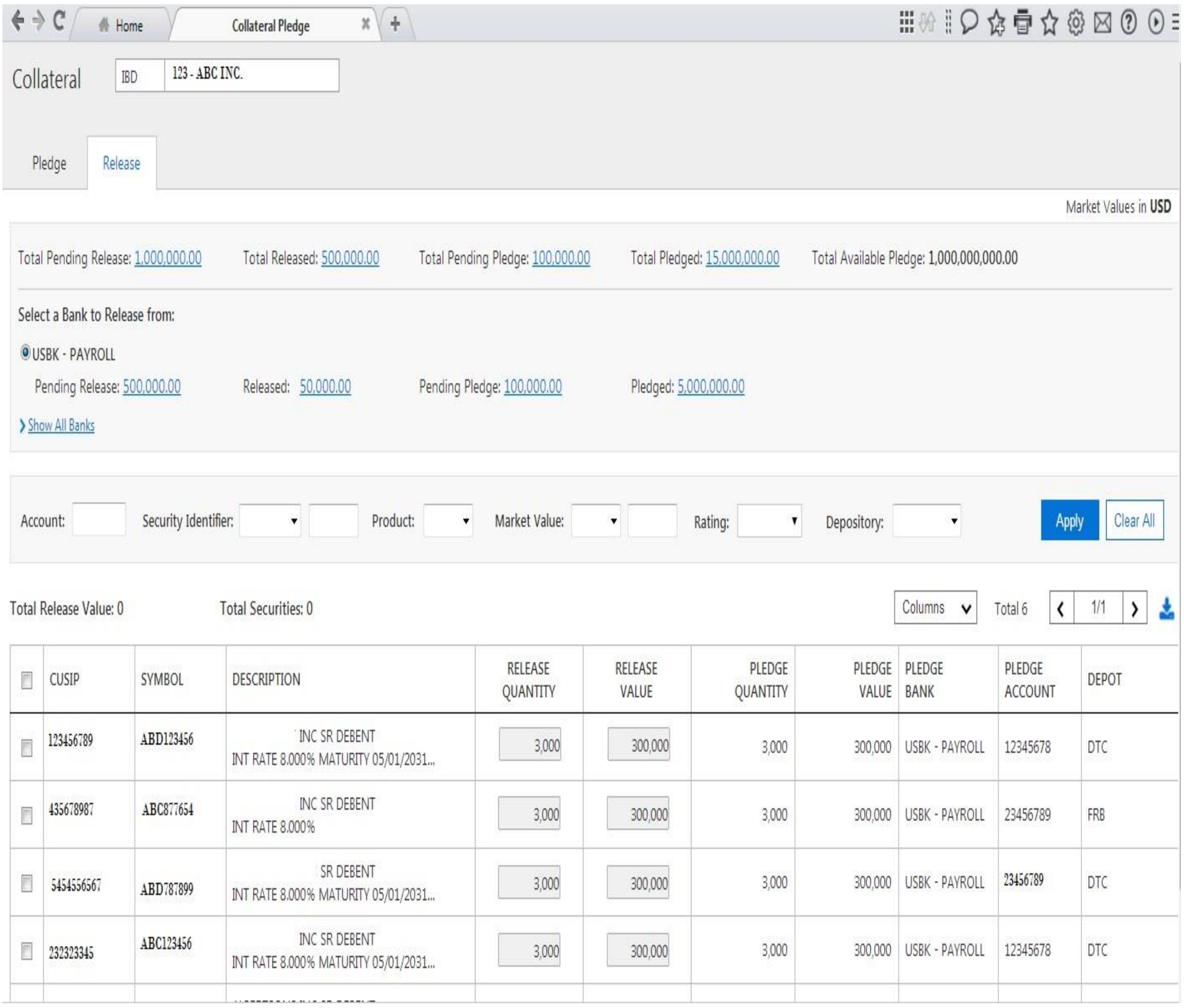
Total Pledge Value: 0 Total Securities: 0 Columns Total 6 1/1

	CUSIP	DESCRIPTION	PRODUCT TYPE	RATING			PLEDGE QUANTITY	PLEDGE VALUE	AVAILABLE QUANTITY	AVAILABLE VALUE	ACCOUNT	DEPOT	PRICE	FA
				S&P	Moodys	Fitch								
☐	123456789	ABC INC SR DEBENT INT RATE 8.000% MATURITY 05/01/2031...		AAAL 1	Aaa*	AAA	3,000	300,000	3,000	300,000	12345678	DTC	100.00	
☐	234345678	ABC INC SR DEBENT INT RATE 8.000%		AAA* 2	Aaa	AA+	3,000	300,000	3,000	300,000		FRB	400.00	
☐	676755456	ABC INC SR DEBENT INT RATE 8.000% MATURITY 05/01/2031...		AAA	Aa1	AA	1,000	100,000	3,000	300,000		DTC	200.00	
☐	564534232			AA+	Aa2	AA-	3,000	300,000	3,000	300,000	12345678	DTC	300.00	

Fig 7.1: Output Screen of Pledging Process

In the Fig 7.1: output screen of Pledge Process is shown. The broker id is displayed on the top. The details like total pending release, total release and total pending pledge and total available pledge are available. Filtering facilities are provided and the table is shown.

7.2 Release



Collateral IBD 123 - ABC INC.

Pledge | Release

Market Values in USD

Total Pending Release: 1,000,000.00 Total Released: 500,000.00 Total Pending Pledge: 100,000.00 Total Pledged: 15,000,000.00 Total Available Pledge: 1,000,000,000.00

Select a Bank to Release from:

● USBK - PAYROLL

Pending Release: 500,000.00 Released: 50,000.00 Pending Pledge: 100,000.00 Pledged: 5,000,000.00

Show All Banks

Account: Security Identifier: Product: Market Value: Rating: Depository:

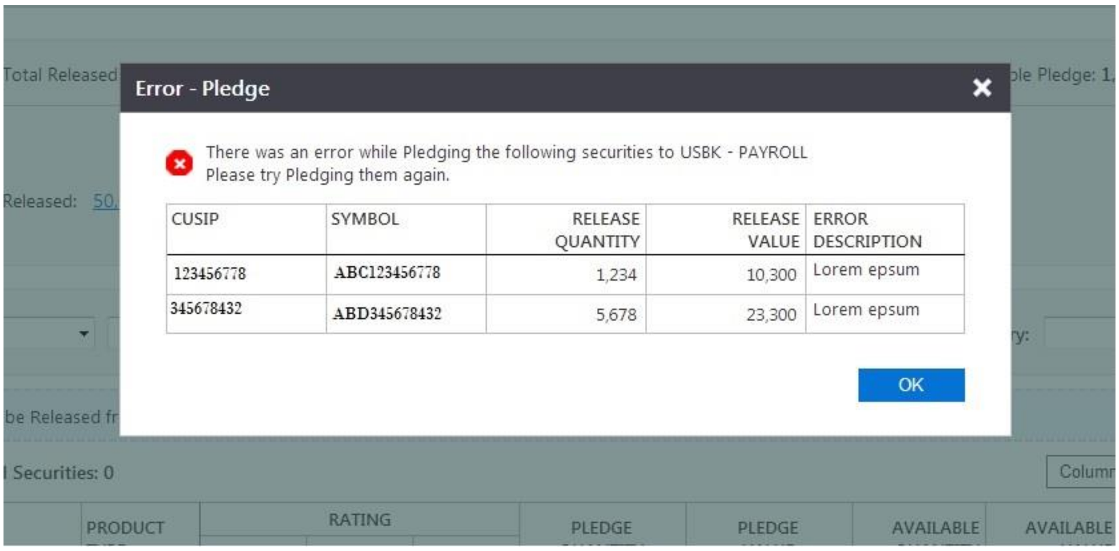
Total Release Value: 0 Total Securities: 0 Total 6 1/1

	CUSIP	SYMBOL	DESCRIPTION	RELEASE QUANTITY	RELEASE VALUE	PLEDGE QUANTITY	PLEDGE VALUE	PLEDGE BANK	PLEDGE ACCOUNT	DEPOT
☐	123456789	ABD123456	INC SR DEBENT INT RATE 8.000% MATURITY 05/01/2031...	3,000	300,000	3,000	300,000	USBK - PAYROLL	12345678	DTC
☐	435678987	ABC877654	INC SR DEBENT INT RATE 8.000%	3,000	300,000	3,000	300,000	USBK - PAYROLL	23456789	FRB
☐	5454556567	ABD787899	SR DEBENT INT RATE 8.000% MATURITY 05/01/2031...	3,000	300,000	3,000	300,000	USBK - PAYROLL	23456789	DTC
☐	232323345	ABC123456	INC SR DEBENT INT RATE 8.000% MATURITY 05/01/2031...	3,000	300,000	3,000	300,000	USBK - PAYROLL	12345678	DTC

Fig 7.2: Output Screen of Releasing Process

In the Fig 7.2: output screen of Release Process is shown. It is similar to the pledge screen except that few columns in the table differ. Here, the pledged quantity and release quantity is displayed.

7.3 Error



There was an error while Pledging the following securities to USBK - PAYROLL
Please try Pledging them again.

CUSIP	SYMBOL	RELEASE QUANTITY	RELEASE VALUE	ERROR DESCRIPTION
123456778	ABC123456778	1,234	10,300	Lorem epsum
345678432	ABD345678432	5,678	23,300	Lorem epsum

Fig 7.3: Output Screen of Error Message

Error message may occur when,

- If bank restricts the security not be pledged at that time.
- Or else if more than one broker pledges the same security.

CHAPTER 8

8 Conclusion

Thus, the main aim of designing this application software is to ease out the user experience in dealing with the pledging and release process. This application also displays all the security policies of the software and makes the user interface self-explanatory, thereby increasing the comfort level of the user. The application is very secure since end to end encryption and decryption algorithms are used. This developed module responds to user queries in a fraction of time because of highly sophisticated technologies like Angular JS 2, Node JS . The best part of the application software is that there is no external agent is required to perform clearing process, instead a single module will take care of all the settlement process completely. The development and maintenance job of the application is also very easy. This application also acts as an intermediary between buyers and sellers of financial instruments. Such systems are called as Clearing House System.