Chapter 5

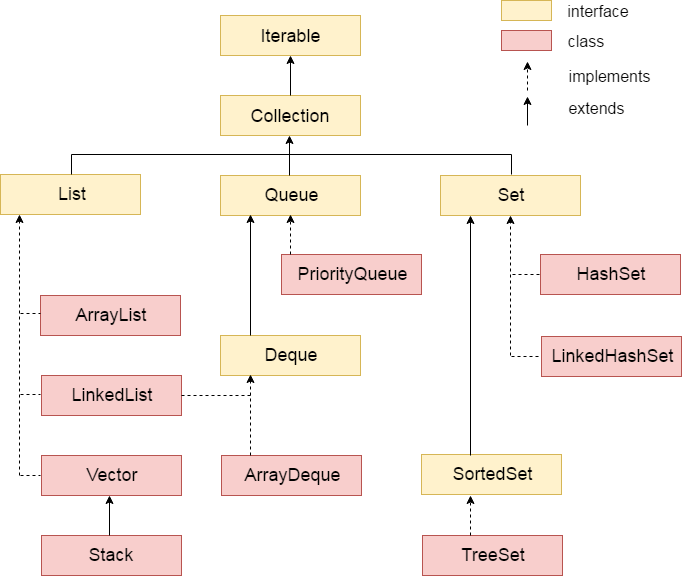
**Implementation**

**5.1 Java Spring Boot**

* Spring Boot provides a good platform for Java developers to develop a stand-alone and production-grade spring application that you can just run. You can get started with minimum configurations without the need for an entire Spring configuration setup.
* Spring framework provides flexibility to configure beans in multiple ways such as XML, Annotations, and JavaConfig. With the number of features increased the complexity also gets increased and configuring Spring applications becomes tedious and error-prone.
* **Spring is very popular for several reasons**
  + Spring’s dependency injection approach encourages writing testable code
  + Easy to use but powerful database transaction management capabilities
  + Spring simplifies integration with other Java frameworks like JPA/Hibernate ORM, Struts/JSF/etc. web frameworks
  + State of the art Web MVC framework for building web application

**5.2 Collections**

* Collections in java is a framework that provides an architecture to store and manipulate the group of objects. All the operations that you perform on a data such as searching, sorting, insertion, manipulation, deletion etc. can be performed by Java Collections.
* Java Collection simply means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque etc.) and classes (Array List, Vector, LinkedList, Priority Queue, HashSet, Linked HashSet, Tree Set etc.).



**Fig.5.1 Java Collections**

**5.3 Lambda Expression**

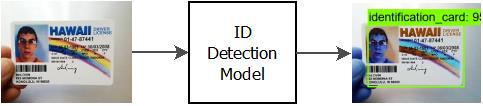
* The Lambda expression is used to provide the implementation of an interface which has a functional interface. It saves a lot of code. In the case of lambda expression, we don't need to define the method again for providing the implementation. Here, we just write the implementation code.
* Java lambda expression is treated as a function, so compiler does not create .class file.
* One issue with anonymous classes is that if the implementation of your anonymous class is very simple, such as an interface that contains only one method, then the syntax of anonymous classes may seem unwieldy and unclear. In these cases, you're usually trying to pass functionality as an argument to another method, such as what action should be taken when someone clicks a button. Lambda expressions enable you to do this, to treat functionality as method argument, or code as data.
* A lambda expression represents an anonymous function. It comprises of a set of parameters, a lambda operator (->) and a function body.

**5.4 Python Models**

Models Used

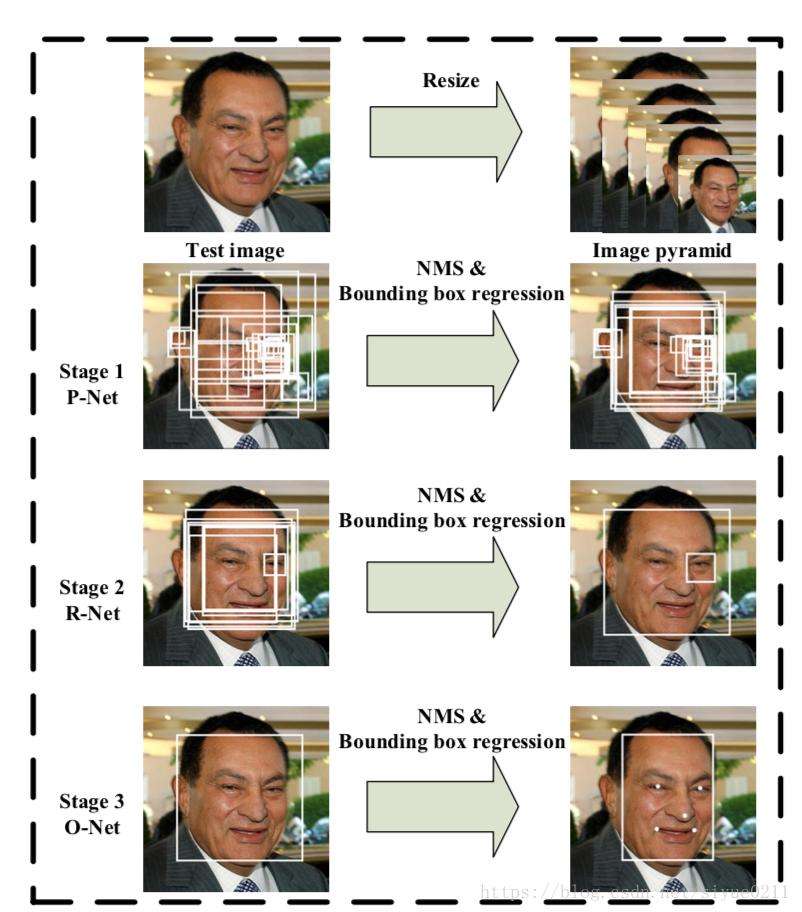
1. ID Detection
2. MTCNN
3. Face Net

* ID Detection
  + Specially trained model to detect ID card in an Image and return cropped image of just the ID card if detected.
  + It also gives confidence score of image being an ID card



**Fig.5.2 ID Card Detection Model**

* MTCNN
  + MTCNN works in 3 stages.
  + Stage 1 is called P-net (Proposal Net). Image is passed through shallow CNN. This will detect lot of faces in an image and also give a confidence score. It will make boundary boxes around face it detected.
  + Stage 2 is called R-net (Refine Net). This network will filter out a large number of boxes with poor effect, and finally select the candidate frames. And coincident boxes are removed using Non-Maximum Suppression (NMS) method.
  + Stage 3 is called O-net (Output Net). This network is similar to R-net. The difference is that this layer structure will identify the area of ​​the face through more supervision. It will pin point the location of facial feature like eyes, mouth and nose.



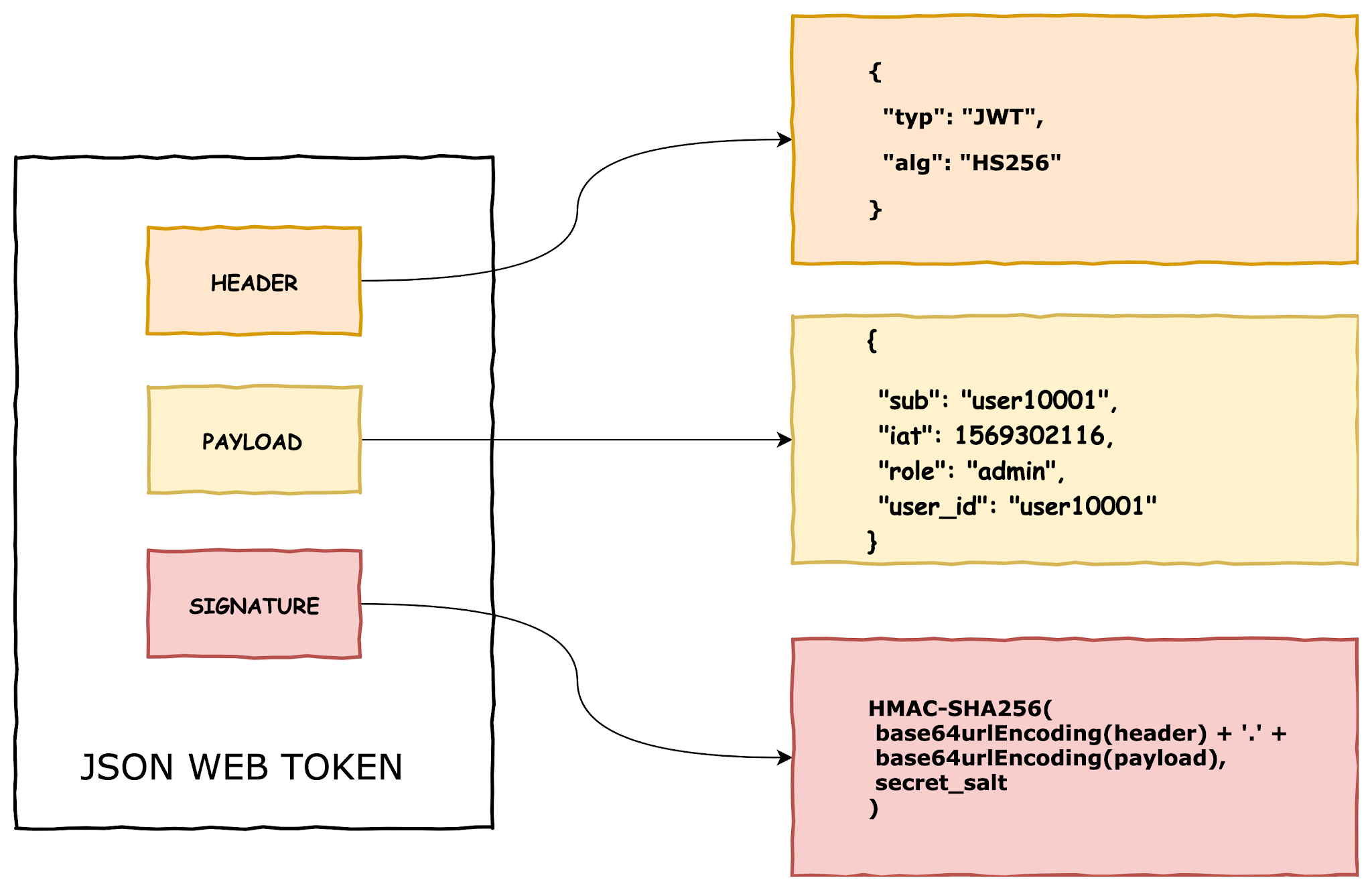
**Fig.5.3 MTCNN Stages**

* Face Net
  + Face Net uses deep convolutional networks along with triplet loss to achieve state of the art accuracy.
  + It maps each face image into a Euclidean space such that the distances in that space correspond to face similarity, i.e. an image of person A will be placed closer to all the other images of person A as compared to images of any other person present in the dataset.
  + Once we have embedding, we can perform different tasks like classification, verification or clustering.

**5.5 JSON Web Token**

For authentication we have used JWT token. JSON Web Token (JWT) is an open standard ([RFC 7519](https://tools.ietf.org/html/rfc7519)) for securely transmitting information between parties as a JSON object.

It is compact, readable and digitally signed using a private key/ or a public key pair by the Identity Provider (IdP). So, the integrity and authenticity of the token can be verified by other parties involved.



**Fig.5.5 JWT Structure**

**5.6 Semantic UI**

Semantic UI is a front-end development framework similar to bootstrap designed for theming. It contains pre-built semantic components that help create beautiful and responsive layouts using human-friendly HTML.

**5.7 SockJs**

For getting pricing data for Stock, we are using WebSocket. Client Subscribe to a particular Organization and receive data from WebSocket for that we are using sockjs client.

**Chapter 6**

**Testing**

**6.1 Testing Plan**

The testing is a technique that is going to be used in the project is black box testing the expected inputs to the system are applied and only the outputs are checked.

**6.2 Testing Strategy**

The development process repeats this testing sub process a number of lines for the following phases.

* Unit Testing
* Integration Testing

Unit Testing tests a unit of code after coding of that unit is completed. Integration Testing tests whether the previous programs that make up a system, interface with each other as desired. System testing ensures that the system meets its stated design specifications. Acceptance testing is testing by users to ascertain whether the system developed is a correct implementation of the software requirements specification. Testing is carried out in such a hierarchical manner that each component is correct and the assembly/combination of components is correct. Merely testing a whole system at the end would most likely throw up errors in components that would be very costly to trace and fix. We have performed both Unit Testing and System Testing to detect and fix errors.

**6.3 Testing Methods**

We have performed Black-box testing for the testing purpose. A brief description is given below:

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied to virtually every level of software testing unit, integration, system and acceptance. It typically comprises most if not all higher-level testing, but can also dominate unit testing as well.

**6.4 Library & Frameworks**

* Front-End

We have used Chai and Mocha for testing react components.

Chai is a BDD / TDD assertion library for node and the browser that can be delightfully paired with any JavaScript testing framework.

Mocha is a feature-rich JavaScript test framework running on Node. js and in the browser, making asynchronous testing.

* Backend

We have used Mockito and Junit for testing Java controllers.

Junit is an open-source unit testing framework for Java. Mockito is used when class under test should not be Mocked. Let's assume you are connecting to a remote service which is being built by one of your engineering team, and you are not familiar with its internal functionality but you know what requests and responses it returns. In that case, you can create a Mock of that Object, and define it with a set of responses returned in different situations. Each situation should get its own different test and for each response you should check separately the reaction of the code.

**Chapter 7**

**Conclusion and Future Extension**

**7.1 Conclusion**

All of the modules in this project developed by us using React+SpringBoot frameworks. For the front-end design, we used React.js along with SemanticUI to implement all views. We are passing data in JSON format to the backend and parsing it using JACKSON. All database operations are done using JPA. All modules were developed separately and were then integrated.

The main concept behind creating communication between two systems lies in the usage of API of the respective system and the stability of that API. These features were developed by keeping in mind all the modules of the system. Hence the quality of features desired by the user is received.

**7.2 Future Extension**

* More User attractive UI.
* Live video recording in KYC instead of image.
* Host application on AWS.
* User can create multiple Watchlists instead of single per Assets.
* Users can perform Short Sells.
* Connect to third party API for pricing.
* Integration with Bank API.

**8**

**Bibliography**

Java documentation - <http://docs.oracle.com/en/>

Stack Overflow - <http://stackoverflow.com/>

Collection - <https://www.javatpoint.com/collections-in-java>

Hibernate - <https://www.javatpoint.com/hibernate-tutorial>

Spring Boot - <https://www.baeldung.com/spring-boot>

React - <https://reactjs.org/tutorial/tutorial.html>

Nextjs - <https://nextjs.org/>

Semantic UI - <https://react.semantic-ui.com/>