Name: Ung Xuan Dat

Student number: s3932156

Lecturer: Vu Thanh Minh

Github link: <https://github.com/xuandat2001/Insurance-Claims-Management-System.git>

**Insurance Claims Management System**

**Application Description**

**1 Introduction**

The insurance system plays a critical role in ensuring the safety of individuals. Furthermore, it has become more important in this dynamic world now because of the demand for guarding assets and businesses to avoid the risks and the unexpected. As a result, many people have trusted the insurance system, and dealing with claims that are generated is genuinely a big problem. Catching these concerns, the system is developed to manage, track and process claims.

The system is build based on the close relationships of three main entities : “Customer, Insurance Card and Claim”.

**2 Purpose**

With this system, insurers can comprehend claims, decrease manual operation, and accelerate the process of claims. The system supports claim management, adding, deleting, and updating the claim for customers.

**3 Step by step Instruction**

The system is designed for Admin to manage the Customers, Insurance Card and Claims. When the program starts, the information of three entities will be load from the data files and store them into three lists: “PolicyHolderList”, “InsuranceCardList”, “ClaimList”.

Then each Policy Holder object in PolicyHolderList will be added with each Insurance Card object in InsuranceCardList correspondingly.

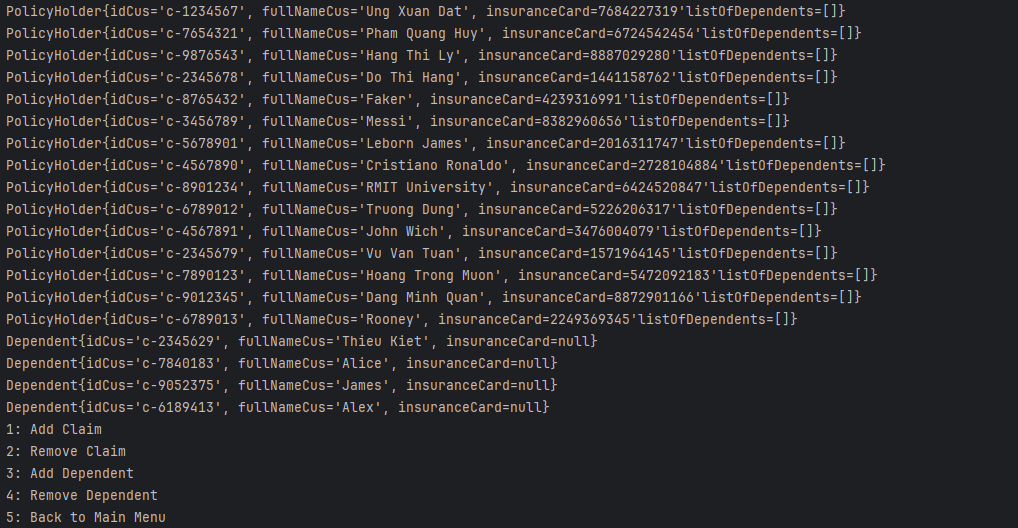
Then the Policy Owner also will be added for all Insurance Cards.

The system starts four choices : “View All Customers”, “View All InsuranceCards”, “View All Claims” and “Exit”.



**Figure1: Main Menu**

In the first choice, all of Customers will be printed into screen and five choices: “Add Claim”, “Remove Claim”, “Add Dependent”, “Remove Dependent” and “Back”



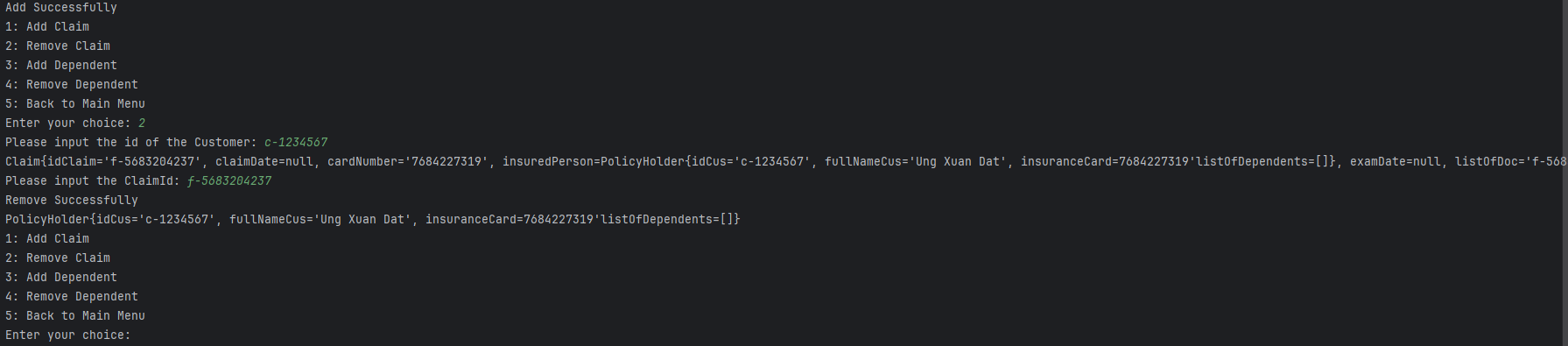
**Figure2**

If the admin choose the Add Claim, the system will ask the admin to enter the Customer ID and Claim ID. If they both are found in the lists, the claim will be added into the list of claims of the customer successfully.



**Figure3**

If the admin choose the Remove Claim, the system will operate similar with the Add Claim. If If they both are found in the lists, the claim will be removed from the list of claims of the customer successfully.



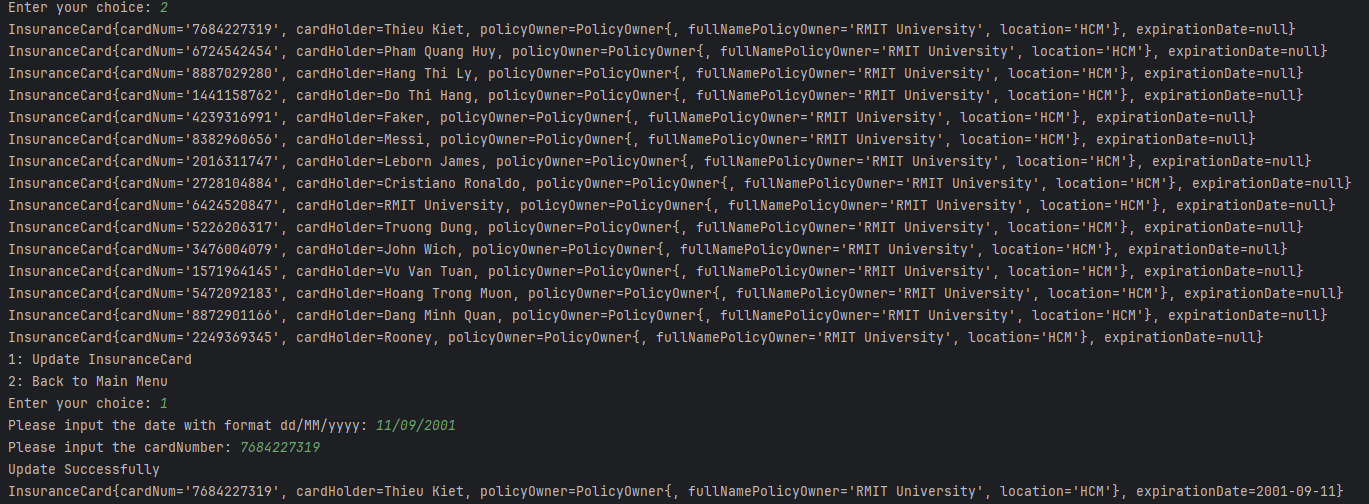
**Figure4**

The Add and Remove Dependent also work similar with two functions above. The system will ask the admin to enter the Customer ID and Dependent ID. If they both are found in the lists, the dependet will be added into or remove from the list of dependents of the customer successfully.



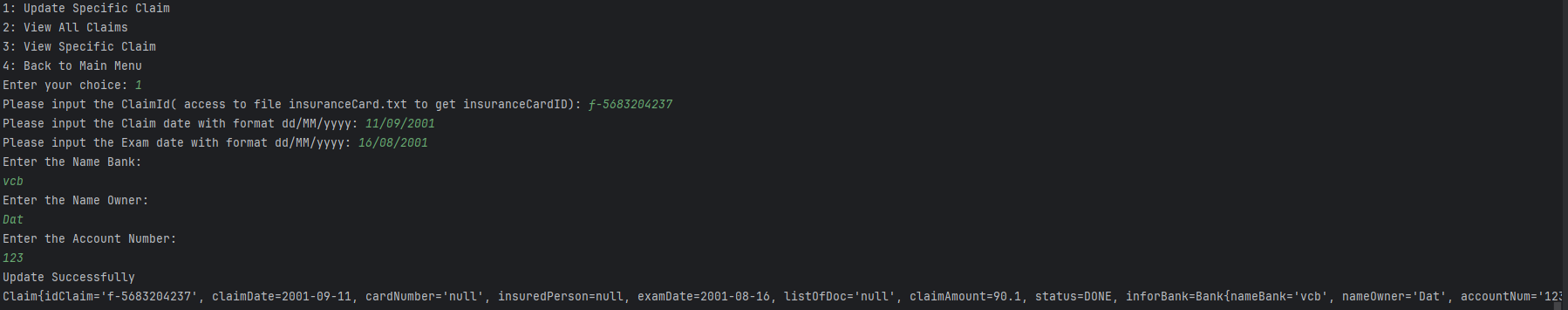
**Figure5**

Back to the main menu, if the admin choose “View all InsuranceCards”, the system will displayed all Insurance Cards and two choices “Update Insurance Card” and “Back”. If the admin choose the “Update Insurance Card”, the system will ask the admin to enter the cardNumber. If the cardNumber is found, the system will ask the admin to enter the date with format(dd//mm/yyyy) to set the expirationDate for the Insurance Card



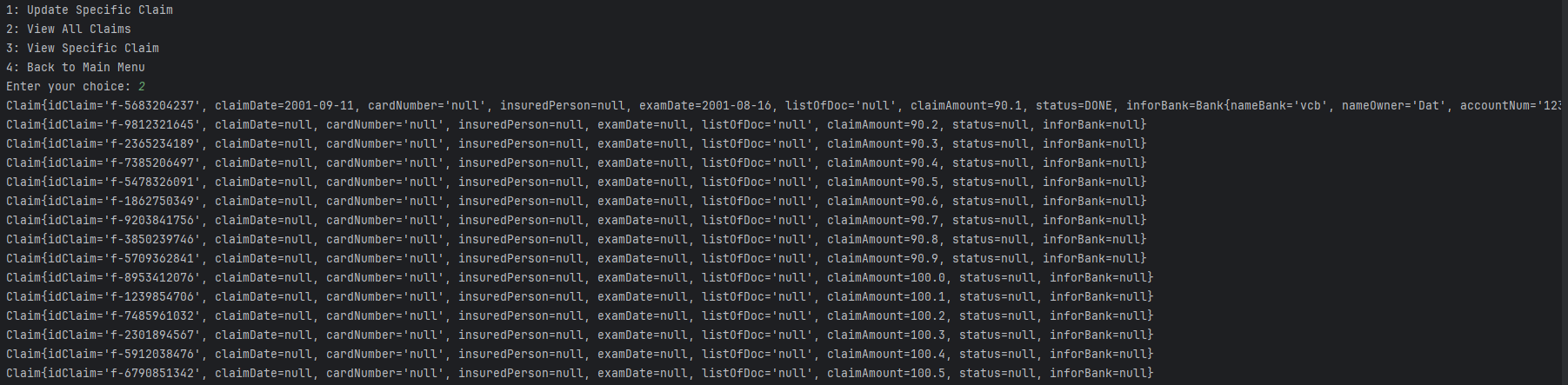
**Figure6**

Back to the main menu, if the admin choose “View all Claims”, the system will displayed all Claim Id and four choices: “Update Specific Claim”, “View one Claim”, “View Specific Claim”, “Back”. If the admin choose the first choices, the system will run updateClaim() method and ask user to enter the ID claim. If the ID claim is found, the system will ask admin promote the claim Data, Exam Date and Information Bank.



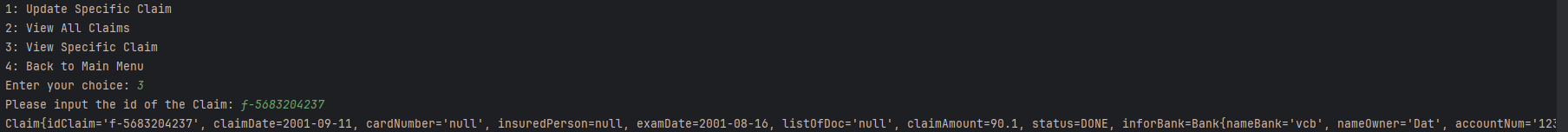
**Figure7**

If the admin choose “View all Claim”, all of detail Claim will be printed on screen.



**Figure8**

If the admin choose “View one Claim”, the system will aske the admin to enter the Id of Claim and print it on the screen.



**Figure10**

**Application Flow (Diagram)**

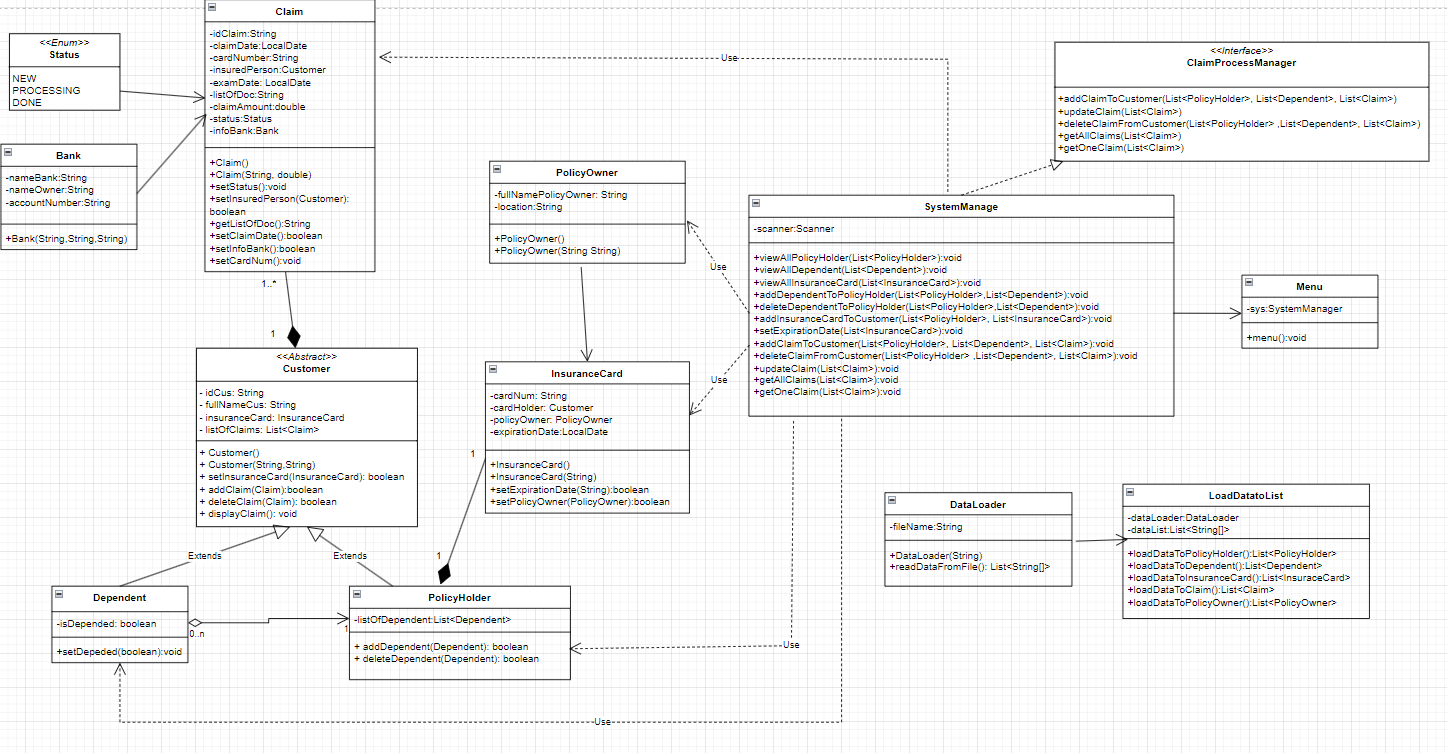


Figure11: The Insurance Claim System diagram

**Description Each Class**

|  |  |  |  |
| --- | --- | --- | --- |
| Class : **Customer** | Name and Data Type | Description | Explanation |
| Attributes | idCus : String | Id of Customer |  |
| fullNameCus: String | Full name of Customer |  |
| insuranceCard:InsuranceCard | InsuranceCard of Customer |  |
| listOfDependent:List<Dependent> | List of Dependent of Customer | Store dependents |
| Methods | setInsuranceCard(InsuranceCard) | Set a new InsuranceCard | Set a new InsuranceCard |
| addClaim(Claim) | Add Claim to the Claim List | Add Claim to the Claim List |
| removeClaim(Claim) | Remove Claim to the Claim List | Remove Claim to the Claim List |
| displayClaim() | Display all Claim in ClaimList | Display all Claim in ClaimList |

|  |  |  |  |
| --- | --- | --- | --- |
| Class:  **Dependent** | Name and Data Type | Description | Explanation |
| Attributes | isDepended:boolean | Check the dependent is depended or not | Check the dependent is depended or not |
| Methods | setDepended | Set Depended | Set Depended |

|  |  |  |  |
| --- | --- | --- | --- |
| Class:  **PolicyHolder** | Name and Data Type | Description | Explanation |
| Attributes | listOfDependent: List<Dependent> | List of Dependent | List of Dependent |
| Methods | addDependent(Dependent) | Add Dependent to the List | Add Dependent to the List |
| removeDependent(Dependent) | Remove Dependent to the List | Remove Dependent to the List |

|  |  |  |  |
| --- | --- | --- | --- |
| Class:  **Claim** | Name and Data Type | Description | Explanation |
| Attributes | idClaim: String | Id of Claim | Id of Claim |
| claimDate: LocalDate | claimDate of Claim |  |
| cardNum: String | cardNum of Claim |  |
| examDate:LocalDate | examDate of Claim |  |
| insurePerson:Customer | insurePerson of Claim |  |
| listOfDoc:String | listOfDoc of Claim |  |
| claimAmount: double | claimAmount of Claim |  |
| status:Status | status of Claim |  |
| infoBank:Bank | infoBank of Claim |  |
| Methods | setStatus() | setStatus for Claim |  |
| setInsuredPerson(Customer) | setInsuredPerson for Claim |  |
| getListOfDoc() | getListOfDoc for Claim |  |
| setClaimDate() | setClaimDate for Claim |  |
| setExamDate() | setExamDate for Claim |  |
| setBank() | setBank for Claim |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Class:  **InsuranceCard** | Name and Data Type | Description | Explanation |
| Attributes | CardNum:String | CardNum of Insurance Card |  |
| CardHolder:Customer | CardHolder of Insurance Card |  |
| policyOwner:PolicyOwner | policyOwner of Insurance Card |  |
| expirationDate:LocalDate | expirationDate of Insurance Card |  |
| Methods | setExpirationDate(String) | Set expirationDate for Insurance Card |  |
| setPolicyOwner(PolicyOwner) | Set PolicyOwner for Insurance Card |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Class:  **System Manage** | Name and Data Type | Description | Explanation |
| Attributes | Scanner:Scanner | obtaining the input of the primitive types like int, double, etc. and strings | obtaining the input of the primitive types like int, double, etc. and strings |
| Methods | viewAllPolicyHolder(List<PolicyHolder>) | View all PolicyHolders |  |
| viewAllDependent(List<Dependent>): | View all Dependents |  |
| viewAllInsuranceCard(List<InsuranceCard>) | View all InsuranceCards |  |
| addDependentToPolicyHolder(List<PolicyHolder>,List<Dependent>) | Add Dependent to PolicyHolder | Use two lists to find specific policyHolder and Dependent |
| deleteDependentToPolicyHolder(List<PolicyHolder>,List<Dependent>) | Delete Dependent to PolicyHolder | Use two lists to find specific policyHolder and Dependent |
| addInsuranceCardToCustomer(List<PolicyHolder>, List<InsuranceCard>) | Add InsuranceCard to PolicyHolder | Use two lists to find specific policyHolder and Dependent |
|
| setExpirationDate(List<InsuranceCard>) | Set ExpirationDate | Use the list to find specific InsuranceCard |
| addClaimToCustomer(List<PolicyHolder>, List<Dependent>, List<Claim>) | Add Claim to PolicyHolder | Use two lists to find specific policyHolder and Claim |
| deleteClaimFromCustomer(List<PolicyHolder> ,List<Dependent>, List<Claim>) | Delete Claim to PolicyHolder | Use two lists to find specific policyHolder and Claim |
| updateClaim(List<Claim>) | Update Claim |  |
| getAllClaims(List<Claim>) | View all Claim |  |
| getOneClaim(List<Claim>) | Get one claim |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Class:  **DataLoader** | Name and Data Type | Description | Explanation |
| Attributes | fileName:String | Name of File | Contain the direction of file |
| Methods | readDataFromFile() | Read data from a file | Readata |

|  |  |  |  |
| --- | --- | --- | --- |
| Class:  **LoadDataToList** | Name and Data Type | Description | Explanation |
| Attributes | dataLoader:DataLoader | dataLoader variable |  |
| dataList:List<String[]> | List of string array |  |
| Methods | loadDataToPolicyHolder() | Load Data To PolicyHolder | To return the PolicyHolderList |
| loadDataToDependent() | Load Data To Dependent | To return the DependentList |
| loadDataToInsuranceCard() | Load Data To InsuranceCard | To return the InsuranceCard List |
| loadDataToClaim() | Load Data To Claim | To return the Claim List |
| loadDataToPolicyOwner() | Load Data To PolicyOwner | To return the PolicyOwner List |

**API list (With brief description)**

|  |  |
| --- | --- |
| **Name of API** | **Brief description** |
| “**java.util.Scanner**” | It is used for obtaining the input of the primitive types like int, double, etc. and strings  <https://www>.geeksforgeeks.org/scanner-class-in-java/ |
| “**java.time.LocalDate**” | It is used to get the current date  <https://www>.geeksforgeeks.org/java-time-localdate-class-in-java/ |
| “**java.time.format.DateTimeFormatter**” | It is to format and parse date and time  <https://www>.geeksforgeeks.org/java-time-localdate-class-in-java/ |
| “**java.util.ArrayList**” | It is used to provide the functionality of a dynamic array where the size is not fixed as an array |
| **“java.util.List”** | It is used to provide a way to store the ordered collection  <https://www>.geeksforgeeks.org/list-interface-java-examples/ |
| “**java.io.File**” | It is a  representation of files and directory pathnames  <https://www>.geeksforgeeks.org/file-class-in-java/ |
| “**java.io.IOException**” | It is used to signal that an I/O exception of some sort has occurred. |

**Table:** Application Programming Interface(API)

**Any drawback and Future Work**

Overall, the system has implemented some basic functions of CRUD and set relationships between Customers, Insurance Cards, and Claims. However, there are some limitations in designing the system. Firstly, loading data from external files has some disadvantages, like what if the admin wants to update new entities in the system or access the specific object in data files? To address this problem, I recommend using relational database management systems such as MySQL, Oracle, PhpMyAdmin, etc. The data will be managed and organized as tables consisting of columns and rows with relationships defined between tables. Another drawback is that the system only allows the admin to operate with claims. I suggest that the system should develop more users instead of the admin, such as customers and policy owners.

Creating fake claims is also a persistent problem in almost all insurance systems. Therefore, applying machine learning algorithms is one of the future plans to improve the fraud detection ability in this system. This enhancement can detect suspicious requests from users by analyzing their behavior.