Topic : Life Insurance management system

Group no : MLB_03.01_01

Campus : Malabe / Metro / Matara / Kandy / Kurunegala / Kandy / Jaffna

Submission Date : 20/05/2022

We declare that this is our own work and this Assignment does not incorporate without acknowledgment any material previously submitted by anyone else in SLIIT or any other university/Institute. And we declare that each one of us equally contributed to the completion of this Assignment.

Registration No	**Name**	**Contact Number**
IT21225406	Jayasinghe J.A.P.M	0719986708
IT21222986	Fernando N.D.H	0767670044
IT21223594	Thalangama T.P	0723572147
IT21225574	Jayasinghe.J.A.J.M	0741407560
IT21224898	Wijethilaka I.G.R.S.D	0713191163

Contents

Requirements

1. Registered customers and Administrators are the two types of end users who uses the system
2. Registered customer can access the branches details, insurance details, system information and contact information of the company
3. Registered customers can claim insurance using system and if there is an issue, customers also can file complains
4. Administrators can create events in the system
5. Administrators can view other administrators' contact information within the system
6. Account manager and analyzer can create reports
7. There are multiple analyzers who works in a single analyzer department
8. Analyzer can use the analytical calculator in the system

<u>**Noun & Verb Analysis**</u>

- System has two types of Customers such as Registered Customers and Unregistered Customers.
- Unregistered Customers Can Register to the system by providing details such as First name, Last name, NIC and Email.
- Administrators, Analyzers, Managers and other Members of Management can log into the system using their User name and Password.
- Registered Customers can log into the system by entering the correct Username and Password.
- They can view their Insurance Details, and visit Branches, System Details anyway.
- And Registered Customers can create the complaints if they want by using customer complaints page.

- And they can visit the Contact Information and contact the Agents as they need.
- Account manager can access the Details of Registered Customers and check their complaints.
- After the checking Customer Details and Account Manager can issue the Claims for the Customers.
- Administrators can Create and publish Events.
- They have permission to access the Database and they can make any changes.
- If they want any supports, they can request supports by using support Page.
- Analyzer and Account manager can make Reports and they can use any Reports.

Identified Classes

- End users
- Administrators
- Registered customers
- Analyzer department
- Analyzers
- Account manager
- Events
- Reports
- Analytical calculator
- System support details
- Customer complaints
- Customer claims
- Branches
- Insurance details
- System details
- Contact information

Why we rejected other nouns

According to the above analysis we found that there were

Redundant: Members, registered customers, Agents ,administrators

Outside scope of system: system

Meta-language : They

Identified Methods

- Verify login
- create events
- view administrators contacts information
- display admin details
- access branches details
- access insurance details
- access system information
- access contact information
- claim insurance
- file complains
- display customer details
- create reports
- use the analytical calculator
- analyze report
- create reports
- validate Complaints
- issue claims
- Display event details
- Display reports
- Calculate mathematical calculations
- Print result
- Display system support details
- Display customers complain details
- Calculate insurance claim
- Display claim detailsDisplay branch details
- Display insurance details
- Display system details

- Display contact information

End users	
Responsibilities	collaborations
Verify login	

Administrators	
Responsibilities	collaborations
create events	Events
view administrators contacts information	Contact information
display admin details	

Registered customers	
Responsibilities	collaborations
access branches details	Branches
access insurance details	Insurance details
access system information	System details
access contact information	Contact information
claim insurance	Customer claims
file complains	Customer complaints
display customer details	

Analyzer department	
Responsibilities	collaborations
Create reports	Reports
use the analytical calculator	Analytical calculator

Analyzers	
Responsibilities	collaborations
create reports	Reports
use the analytical calculator	Analytical calculator
analyze report	Reports

Account manager	
Responsibilities	collaborations
create reports	Reports
validate Complaints	Customer complaints
issue claims	Customer claims

Events	
Responsibilities	collaborations
Display event details	

Reports	
Responsibilities	collaborations
Display reports	

Analytical calculator	
Responsibilities	collaborations
Calculate mathematical calculations Print result	

System support details	
Responsibilities	collaborations
Display system support details	

Customer complaints	
Responsibilities	collaborations
Display customers complain details	

Customer claims	
Responsibilities	collaborations
Calculate insurance claim Display claim details	

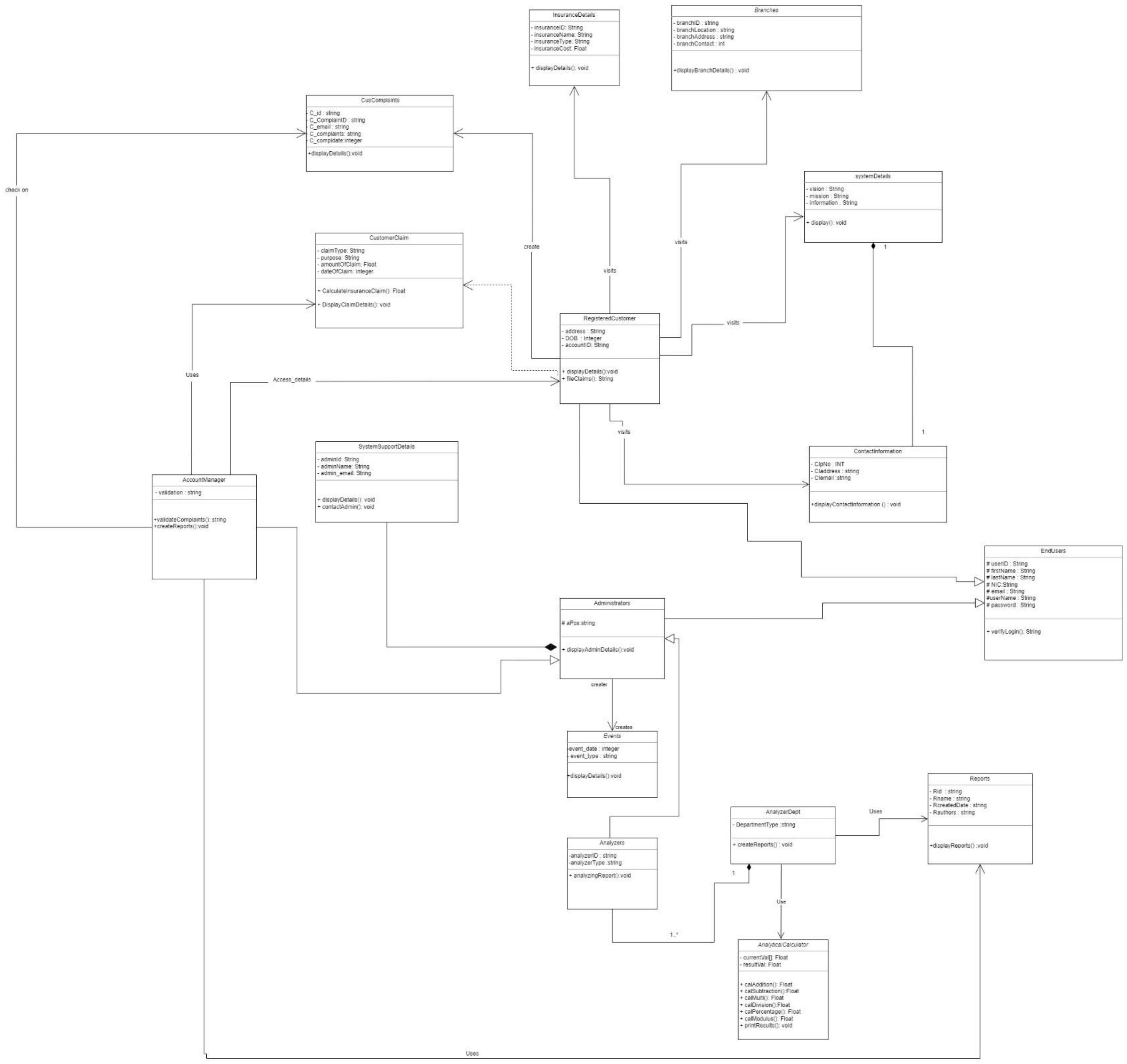
Branches	
Responsibilities	collaborations
Display branch details	

Insurance details	
Responsibilities	collaborations
Display insurance details	

System details	
Responsibilities	collaborations
Display system details	

Contact information	
Responsibilities	collaborations
Display contact information	

Class Diagram



Class header files

AccountManager.h

```
#include <string>
#include "RegisteredCustomer.h"
#include "CustomerClaim.h"
#include "CusComplaints.h"
#include "Reports.h"
using namespace std;

Class AccountManager : public Administrators
{
    private :
        string validation;
        CusComplaints* cuscomp[2];
        Reports* reports[2];
        CustomerClaim* cusclaim[2];

    public :
        AccountManager();
        AccountManager(string val , CusComplaints * cusCom ,Reports * rep   );
        string validateComplaints();
        void createReports();
        string issueClaims();
        ~AccountManager();

};
```

Administrators.h

```
#include "endUser.h"
#include <string>
using namespace std;

class Administrators : public endUser{

  private:
    string aPos;

  public:
    Administrators();
    void setaPos(string pus);
    void displayAdminDetails();
    ~Administrators();
};
```

AnalyticalCalculator.h

```
class AnalyticalCalculator {

private:
    float value[10];
    float returnVal;
public:
    AnalyticalCalculator();
    AnalyticalCalculator(float val[10], float retVal);

    float calAddition();
    float calSubtraction();
    float calMulti();
    float calDivision();
    float calPercentage();
    float calModulus();
    void printResults();

    ~AnalyticalCalculator();
};
```

AnalyzerDepartment.h

```
#include<iostream>
#include<string>
using namespace std;

class AnalyzerDept
{
    private:
        string DepartmentType;

        Analyzers* analyzers[3];

    public:
        void analyzingReport();
        ~AnalyzerDept();

}
```

Analyzers.h

```
#include<string>
using namespace std;

class Analyzers : public Administrators
{
    private:
        string analyzerID;

    public:
        ~Analyzers();
};
```

Branches.h

```
 #include <string>
using namespace std;

class branch
{
private:

  string branchID;
  string branchLocation;
  string branchAddress;
  int branchContact;

public:

  branch();

  void setBranchID(string branchID);
  void setBranchLocation(string branchLocation );
  void setBranchAddress(string branchAddress);
  void setBranchContact(int branchContact);

  void displayBranchDetails();
  ~branch();

};
```

ContactInformation.h

```
#include <string>
using namespace std;

class ContactInformation {
private:
  int CIpNo;
  string CIaddress;
  string CIemail;

public:
  ContactInformation();
  ContactInformation(int number, string address, string email); // overloaded
  void displayContactInformation();
  ~ContactInformation();
};
```

CustomerClaims.h

```
#include<iostream>
#include<string>
using namespace std;

class Registeredcustomer
{
    private:
        string address;
        int DOB;
        string accountId;
    public:
        Registeredcustomer();
        Registeredcustomer(string add, int dob, str accId)
        {
          address=add;
          DOB=dob;
          accountId=addId;
        }
        void displayDetails();
        string fileClaims();
};

class customerClaim
{
    private:
        string claimType;
        string purpose;
        float amountOfClaim;
        int dateOfClaim

    public:
        customerClaim();
        float calculateInsuranceClaim();
        void displayClaimDetails();
        ~customerClaim();
};
```

CustomerComplaints.h

```
#include<string>
using namespace std;

class CusComplaints
{
private:
  string C_id;
  string C_ComplainID;
  string C_email;
  string C_complaints;
  int C_compldate;

public:
  CusComplaints();
  void displayDetails();
  ~CusComplaints();
};
```

Enduser.h

```
#include <string>
using namespace std;

class endUser
{
protected:
  string userID;
  string FirstName;
  string NIC;
  string email;
  string UserName;
  string password;

public:

  endUser();
  endUser(string u_ID ,string f_Name ,string nic, string e_mail ,string U_name
 ,string pass);

  string veryfyLogin();

  ~endUser();


}
```

Events.h

```cpp
#include <string>
using namespace std;

class Event
{
private:
  int event_date;
  string event_type;

public:
  Event();
  void displayDetails();
  ~Event();
}
```

InsuranceDetails.h

```cpp
#include <string>
using namespace std;

class InsuranceDetails {

private:
    string insuranceID;
    string insuranceName;
    string insuranceType;
    float insuranceCost;
public:
    InsuranceDetails();
    InsuranceDetails(string id, string name, string type, float cost);

    void displayDetails();

    ~InsuranceDetails();
};
```

RegisteredCustomer.h

```
#include <string>
#include "EndUser.h"
#include "CusComplains.h"
#include "InsuranceDetails.h"
#include "Branches.h"
#include "ContactInformation.h"
#include "SystemDetails.h"
using namespace std;

class RegisteredCustomer : public EndUsers
{
private:
    string address;
    int DOB;
    string accountID;

    CusComplains* complain;
    InsuranceDetails* insDet;
    Branches* branch;
    ContactInformation* Cinfo;
    SystemDetails* SDetails;


public:
    RegisteredCustomer();
    RegisteredCustomer(string address, int dob, string accID, CusComplain*
cmpl, InsuranceDetails* insdtl, Branches* pBranch, ContactInformation* PCinfo,
SystemDetails* PSDetails);

    void displayDetails();
    string fileClaims();

    ~RegisteredCustomer();
};
```

Reports.h

```cpp
#include <string>
using namespace std;

class Reports {
private:
  string Rid;
  string Rname;
  string RcreatedDate;
  string Rauthors;

public:
  Reports();
  Reports(string name, string date, string authors); // overloaded
  void displayReports();
  ~Reports();
};
```

SystemDetails.h

```cpp
#include<iostream>
#include<string>
using namespace std;

class systemDetails
{
    private:
        string vision;
        string mission;
        string information;
    public:
       systemDetails();
       void display();
       ~systemDetails();
};
```

SystemSuppourDetails.h

```
#include<iostream>
using namespace std;

class systemsSupportDetails
{
    private:
        string adminId;
        string adminName;
        string adminEmail;
    public:
        systemsSupportDetails();
        systemsSupportDetails(string id, string name, string email);
        void displayDetails();
        void contactAdmin();
        ~systemsSupportDetails();
};
```

Class .cpp files

AccountManager.cpp

```
#include<iostream>
#include <cstring>
#include "RegisteredCustomer.h"
#include "CustomerClaim.h"
#include "CusComplaints.h"
#include "Reports.h"
#include "AccountManager.h"

using namespace std;

AccountManager::AccountManager()
{
        validation = "";
}

AccountManager::~AccountManager()
{

}
```

Administrators.cpp

```cpp
#include <iostream>
#include <string>
#include "Administrators.h"
using namespace std;

Administrators::Administrators()
{
  aPos = "0";
}


Administrators::~Administrators()
{

}
```

AnalyticalCalculator.cpp

```cpp
#include "AnalyticalCalculator.h"
#include <iostream>
using namespace std;


AnalyticalCalculator::AnalyticalCalculator() {
     value[10] = {0};
     returnVal = 0;
}

AnalyticalCalculator::~AnalyticalCalculator() {
    cout << "destructor" << endl;
}
```

Analyzers.cpp

```cpp
#include<iostream>
#include<string>
using namespace std;
Analyzers::Analyzers()
{
    analyzerID="";
}
Analyzers::~Analyzers()
{
}
```

Branches.cpp

```cpp
#include <iostream>
#include <string>
#include "Branches.h"
using namespace std;

branch::branch()
{
    branchID ="";
    branchLocation="";
    branchAddress="";
    branchContact=0;
}


branch::~branch()
{

}
```

ContactInformation.cpp

```cpp
#include <string>
#include "ContactInformation.h"
using namespace std;

ContactInformation::ContactInformation()
{
    number = 0 ;
    address = "" ;
    email ="" ;

}

ContactInformation::~ContactInformation()
{

}
```

Enduser.cpp

```
#include <string>
#include <iostream>
#include<Enduser.h>
using namespace std;

endUser::endUser()
{
	userID ="";
	FirstName="";
	NIC="";
	email="";
	UserName="";
	password="";

}



endUser::endUser(string u_ID ,string f_Name ,string nic, string e_mail ,string 
U_name  ,string pass)
{
	userID = u_ID;
	FirstName= f_Name;
	NIC = nic;
	email=e_mail;
	UserName= U_name;
	password= pass;
}




endUser::~endUser()
{

}
```

Eents.cpp

```
#include<iostream>
#include<string>
using namespace std;

 Event::Event()
{
  event_date=0;
  event_type="";
}
Event::~Event(){

}
```

InsuranceDetails.cpp

```
#include "InsuranceDetails.h"
#include <iostream>
#include <string>
using namespace std;

InsuranceDetails::InsuranceDetails() {

    insuranceID = "";
    insuranceName = "";
    insuranceType = "";
    insuranceCost = 0.00;
}

InsuranceDetails::~InsuranceDetails() {

    cout << "Destructor called!" << endl;
}
```

RegisteredCustomer.cpp

```cpp
#include "RegisteredCustomer.h"
#include <iostream>
#include <string>
using namespace std;

RegisteredCustomer::RegisteredCustomer() {
    string address = "";
    int DOB = 0;
    string accountID = "";
}

RegisteredCustomer::~RegisteredCustomer() {
    cout << "RegisteredCustomer destructor called!" << endl;
}
```

Reports.cpp

```cpp
#include <iostream>
#include <string>
#include "Reports.h"

using namespace std;



Reports::Reports() {
  Rid = "";
  Rname = "";
  RcreatedDate = "";
  Rauthors = "";
}

Reports::~Reports() {

    }
```

systemDetails.cpp

```cpp
#include <iostream>
#include <cstring>
#include "SystemSupportDetails.h"
using namespace std;

SystemSupportDetails::SystemSupportDetails()
{
    strcpy(adminId, "");
    strcpy(adminName, "");
    strcpy(adminEmail,"");
}

void SystemSupportDetails::displayDetails()
{
    cout << adminId << endl;
    cout << adminName << endl;
    cout << adminEmail << endl;
}

void SystemSupportDetails::contactAdmin()
{

}

SystemSupportDetails::~SystemSupportDetails()
{

}
```

systemSupportDetails.cpp

```cpp
#include <iostream>
#include <cstring>
#include "SystemSupportDetails.h"
using namespace std;

SystemSupportDetails::SystemSupportDetails()
{
    strcpy(adminId, "");
    strcpy(adminName, "");
    strcpy(adminEmail,"");
}

void SystemSupportDetails::displayDetails()
{
    cout << adminId << endl;
    cout << adminName << endl;
    cout << adminEmail << endl;
}

void SystemSupportDetails::contactAdmin()
{

}

SystemSupportDetails::~SystemSupportDetails()
{

}
```

Main.cpp

```cpp
#include <iostream>
#include "Administrators.h"
#include "RegisteredCustomer.h"
#include "InsuranceDetails.h"
#include "AnalyticalCalculator.h"
#include "Branches.h"
#include "ContactInformation.h"
#include "Reports.h"
#include "AccountManager.h"
using namespace std;


int main()

{
 //---------IT21223594---------------
    RegisteredCustomer regCustomer;
  InsuranceDetails insDetails;
  AnalyticalCalculator cal;
  //----------------------------------

  //---------IT21223594---------------

    regCustomer.displayDetails;
    string claimInfo = regCustomer.fileClaims;
  cout << claimInfo << endl;

  insDetails.displayDetails();
    insDetails:InsuranceDetails("ID001", "Apex Insurance", "Life insurance",
195000.00);

  cal.calAddition();
    cal.printResults();
  //----------------------------------

  //----------IT21225406---------------
  //administrators class
   Administrators a1,a2,a3;
    a1.setaPos("a001");
    a1.displayAdminDetails();

    return 0;
  //---------IT21225406----------------
  //branches class
   branch br1 , br2;
  br1.setBranchID("ab001");
  br1.serBranchLocation("colombo");
  br1.setBranchAddress("1,silverstreet,colombo");
  br1.setBranchContact(0111241236);
  br1.displayBranchDetails();
  //---------IT21225406---------------
  a1.veryfyLogin();
  a2.veryfyLogin();


    //----------IT21222986---------------
    ContactInformation ci1  ;

    ci1.displayContactInformation ();
    //-----------------------------------

    //----------IT21222986---------------
     Reports r1 ;

    r1.displayReports();
    //-----------------------------------

    //----------IT21222986---------------
     AccountManager am1 ;

    am1.validateComplaints();
    am1.createReports();
    am1.string issueClaims();
    //-----------------------------------

  //---------IT21225574----------------
  //Event class
  Events evnt ;

  evnt.displayEvents();
  //-----------------------------------

  //--------IT21225574-----------------
  //Customer Complaints
CustomerComplaints Ccompl;

Ccompl.displayDetails();
//-------------------------------------

//----------IT21225574---------------
//AnalyzerDepartment
AnalyzerDepartment ADep;
```

```
ADep.createReports();
//------------------------------------

//--------IT21225574------------------
//Analyzers
Analyzers Anlyzr;

Anlyzr.analyzingReport();

//--------IT21224898------------------
//Customer Claims
customerClaims cusClaims;

cusClaims.displayDetails();

//--------IT21224898------------------
//System Details
systemDetails systDet;

systDet.display();

//--------IT21224898------------------
//System Support Details
systemSupportDetails ssdet;

  ssdet.displayDetails();
  ssdet.contactAdmin();
```

Individual Contribution

IT21225406 - J.A.P.M. Jayasinghe

- Created requirements for the project
- Created classes with the help of other team members
- Class diagram
    - Administrators
    - Branches
- Added relationships in class diagram with group members
- Coded classes
    - Administrators
    - Branches
    - Enduser
- Finalized the report

IT21222986 – N.D.H.Fernando

- Created CRC cards
- Class diagram
    - Reports
    - Contact information
    - Account manager
- Coded classes
    - Reports
    - Contact information
    - Account manager
- Added relationships in class diagram with group members

IT21223594 – T.P.Thalangama

- Added relationships in class diagrams with group members
- Class diagram
    - End user
    - Analytical Calculator
    - Registered Customer
    - Insurance Details
- Coded classes
    - Analytical Calculator
    - Registered Customer
    - Insurance Details
- Helped other members in coding classes

IT21224898 – Wijethilaka I.G.R.S.D.

Created Noun & Verb Analysis Page with support of Jayasinghe J.A.J.M – (IT21225574)

- Class Diagrams
    - Customer Claims
    - System Details
    - System Support Details
- Coded Classes
    - Customer Claims
    - System Details
    - System Support Details
- Added relationships in class diagrams with group members.

IT21225574-J.A.J.M Jayasinghe

- Class diagram
    - Analyzer class
    - Customer complaints class
    - Events class
- Code class
    - Analyzer class
    - Customer complaints class
    - Events class
- Created Noun & Verb Analysis Page with support of wijethilaka I .G.R.S.D – (IT21224898)
- Added relationships in class diagrams with group members.