A Project Report

On

FAMILY TREE MANAGEMENT SYSTEM

By

Samruddhi Naik	T120238590
Priyanka Suryagan	T120238632
IshaWalimbe	T120238642

Under the guidance of

Prof. K.B. Sadafale

Department of Information Technology

Sinhgad College of Engineering

SAVITRIBAI PHULE PUNE UNIVERSITY

2014-2015

Sinhgad Technical Education Society,
Department of Information Technology
Sinhgad College of Engineering , Pune-41

Date:

CERTIFICATE

This is to certify that,

SamruddhiNaik (T120238590)
PriyankaSuryagan(T120238632)
IshaWalimbe (T120238642)

of class T.E IT; have successfully completed theirproject work on “FAMILY TREE MANAGEMENT SYSTEM’’at SINHGAD COLLEGE OF ENGINEERINGin the partial fulfillment of the Graduate Degree course in T.E at the Department of Information Technology, in the academic Year 2014-2015Semester – I as prescribed by the SavitribaiPhulePune University.

Prof. K.B. SadafaleDr.N. J. Uke
Guide Head of the Department
(Department of Information Technology)

Acknowledgement

I feel great pleasure in expressing my deepest sense of gratitude and sincere thanks to my guide **Prof. K.B.Sadafale**for his valuable guidance during the Project work, without which it would have been very difficult task. I have no words to express my sincere thanks for the valuable guidance, extreme assistance and cooperation extended to all the **Staff Members** of my Department.

This acknowledgement would be incomplete without expressing my special thanks to **Dr. N. J. Uke** Head of the Department (Information Technology) for his support during the work.

I would also like to extend my heartfelt gratitude to my **Principal, Dr. S. D. Lokhande** who provided a lot of valuable support, mostly being behind the veils of college bureaucracy.

Last but not least I would like to thank all the Teaching, Non- Teaching staff members of my Department, my parent and my colleagues those who helped me directly or indirectly for completion of this Project successfully.

Ms. PriyankaSuryagan
Ms. IshaWalimbe
Ms. SamruddhiNaik

Contents

ANCESTRY MANGEMENT SYSTEM

<u>Abstract</u>

Family trees are an age old concept, primarily used to maintain a record of the vast families existing in society. Starting from trees drawn on papers, to extensive tapestries describing all family members, family trees have been an integral part of society. They are one of the few Database Management Systems that were needed even in the early decades to document all the members of a community.

Taking into account the importance of family documentation, this project focuses on an easy and simple interface provided for easy definition and manipulation of entered data. Instead of tedious writing and drawings, this system helps one to build his/her own Family Tree and keep record of the same.

The screens and menus are very user friendly. A person having basic knowledge of computers will be able to operate this system.

Introduction

Detailed problem definition:-

The system has been designed for database maintenance of a Family Tree. It will store all the personal basic information of a person and his family. This data will be readily available to update delete modify as required.

The information will be entered by the user but it will be totally at the administrator's discretion how to manipulate it.

Need for system:-

The project which we have done comes due to the need of automation. In a Database Management System a lot of time is spent in data processing and along with time there is a substantial cost involved in doing so, hence comes the need of automation.

Thus due to this package we can reduce the work load which will result in accuracy and efficiency. It will provide efficient storage and management of large amount of data.

Scope

Thisproject was made keeping in mind the need for awareness of database management systems and their importance in our day to day life. The system extends a ways and means for the data of a certain individual and his family to be stored and retrieved at will.

It also aims to increase speed, accuracy, and efficiency of recording all information and generating all the required reports accurately and promptly.In this system, we have provided the database wherein the administrator can maintain information about the persons, their personal data and their relationship with each other.

Thus the proposed system caters to computerization of all major activities of the Family Tree Management process.

SPECIFIC REQUIRMENTS

- **Hardware Interface**

Processor: Pentium 4 or higher

RAM: min 512 MB

Hard Disc: min 80 GB

- **Software Interface**

Operating System: Microsoft Windows XP/Vista/7

Back End : MongoDB

Front End : Java(JDK)

THEORY OF SOFTWARE USED

❖ WindowsOperatingSystem:-

Windows offers great features like consistent GUI interface; exchange of data between different programs, and it is the most preferred software used today. Most people are comfortable working with Windows.

Although other Operating Systems, like Linux, are slowly becoming very popular, they still lack support, as well as a consistent GUI interface.

❖ Java(JDK):-

The **Java Development Kit** (**JDK**) is an implementation of either one of the Java SE, Java EE or Java ME platforms released by Oracle Corporation in the form of a binary product aimed at Java developers on Solaris, Linux, Mac OS X or Windows. The JDK includes a private JVM and a few other resources to finish the recipe to a Java Application. Since the introduction of the Java platform, it has been by far the most widely used Software Development Kit (SDK). On 17 November 2006, Sun announced that it would be released under the GNU General Public License (GPL), thus making it free software. This happened in large part on 8 May 2007, when Sun contributed the source code to the OpenJDK.

▸ JDK contents

The JDK has as its primary components a collection of programming tools, including:

- **appletviewer** – this tool can be used to run and debug Java applets without a web browser
- **apt** – the annotation-processing tool[4]
- **extcheck** – a utility which can detect JAR-file conflicts
- **idlj**– the IDL-to-Java compiler. This utility generates Java bindings from a given Java IDL file.

- **jabswitch** – the Java Access Bridge. Exposes assistive technologies on Microsoft Windows systems.
- **java**– the loader for Java applications. This tool is an interpreter and can interpret the class files generated by the javac compiler. Now a single launcher is used for both development and deployment. The old deployment launcher, jre, no longer comes with Sun JDK, and instead it has been replaced by this new java loader.
- **javac** – the Java compiler, which converts source code into Java bytecode
- **javadoc** – the documentation generator, which automatically generates documentation from source code comments
- **jar**– the archiver, which packages related class libraries into a single JAR file. This tool also helps manage JAR files.
- **javafxpackager** – tool to package and sign JavaFX applications
- **jarsigner**– the jar signing and verification tool
- **javah** – the C header and stub generator, used to write native methods
- **javap**– the class file disassembler
- **javaws** – the Java Web Start launcher for JNLP applications
- **JConsole** – Java Monitoring and Management Console
- **jdb** – the debugger
- **jhat** – Java Heap Analysis Tool (experimental)
- **jinfo** – This utility gets configuration information from a running Java process or crash dump. (experimental)
- **jmap** – This utility outputs the memory map for Java and can print shared object memory maps or heap memory details of a given process or core dump. (experimental)
- **jmc**– Java Mission Control
- **jps**– Java Virtual Machine Process Status Tool lists the instrumented HotSpot Java Virtual Machines (JVMs) on the target system. (experimental)
- **jrunscript** – Java command-line scriptshell.
- **jstack**– utility which prints Java stack traces of Java threads (experimental)
- **jstat** – Java Virtual Machine statistics monitoring tool (experimental)
- **jstatd** – jstat daemon (experimental)
- **keytool** – tool for manipulating the keystore
- **pack200** – JAR compression tool
- **policytool** – the policy creation and management tool, which can determine policy for a Java runtime, specifying which permissions are available for code from various sources

- **VisualVM** – visual tool integrating several command-line JDK tools and lightweight[*clarification needed*] performance and memory profiling capabilities
- **wsimport**– generates portable JAX-WS artifacts for invoking a web service.
- **xjc** – Part of the Java API for XML Binding (JAXB) API. It accepts an XML schema and generates Java classes.

Experimental tools may not be available in future versions of the JDK.

The JDK also comes with a complete Java Runtime Environment, usually called a *private* runtime, due to the fact that it is separated from the "regular" JRE and has extra contents. It consists of a Java Virtual Machine and all of the class libraries present in the production environment, as well as additional libraries only useful to developers, such as the internationalization libraries and the IDL libraries.

Copies of the JDK also include a wide selection of example programs demonstrating the use of almost all portions of the Java API.

Ambiguity between a JDK and an SDK

The JDK forms an extended subset of a software development kit (SDK). It includes "tools for developing, debugging, and monitoring Java applications". Oracle strongly suggests that they now use the term "JDK" to refer to the Java SE Development Kit. The Java EE SDK is available with or without the "JDK", by which they specifically mean the Java SE 7 JDK.

❖ MongoDB:-

MongoDB (from "hu**mongo**us") is an open-source document database, and the leading NoSQL database. Written in C++, MongoDB features:

- **Document-Oriented Storage »**

  JSON-style documents with dynamic schemas offer simplicity and power.

- **Full Index Support »**

Index on any attribute, just like you're used to.

- **Replication & High Availability »**

  Mirror across LANs and WANs for scale and peace of mind.

- **Auto-Sharding »**

  Scale horizontally without compromising functionality.

- **Querying »**

  Rich, document-based queries.

- **Fast In-Place Updates »**

  Atomic modifiers for contention-free performance.

- **Map/Reduce »**

  Flexible aggregation and data processing.

- **GridFS »**

  Store files of any size without complicating your stack.

- **MongoDB Management Service »**

  Monitoring and backup designed for MongoDB.

- **Partner with MongoDB »**

  Reduce cost, accelerate time to market, and mitigate risk with proactive support and enterprise-grade capabilities.

DATABASE COLLECTION FORMAT

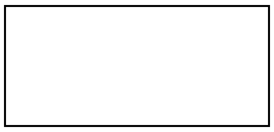
Field Name	Data Type
Firstname	Varchar()
Lastname	Varchar()
Gender	Varchar()
Relationship	Varchar()
Date of Birth	Date
Age	Int()
Status	Varchar()

<u>ER DIAGRAM</u>

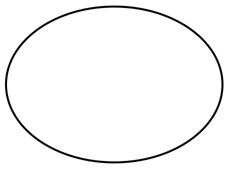
Symbols used in ER Diagram :

Following are the symbols used for drawing the Context Level Diagram and the Data Flow Diagrams.

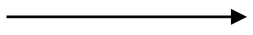
- **Collection**

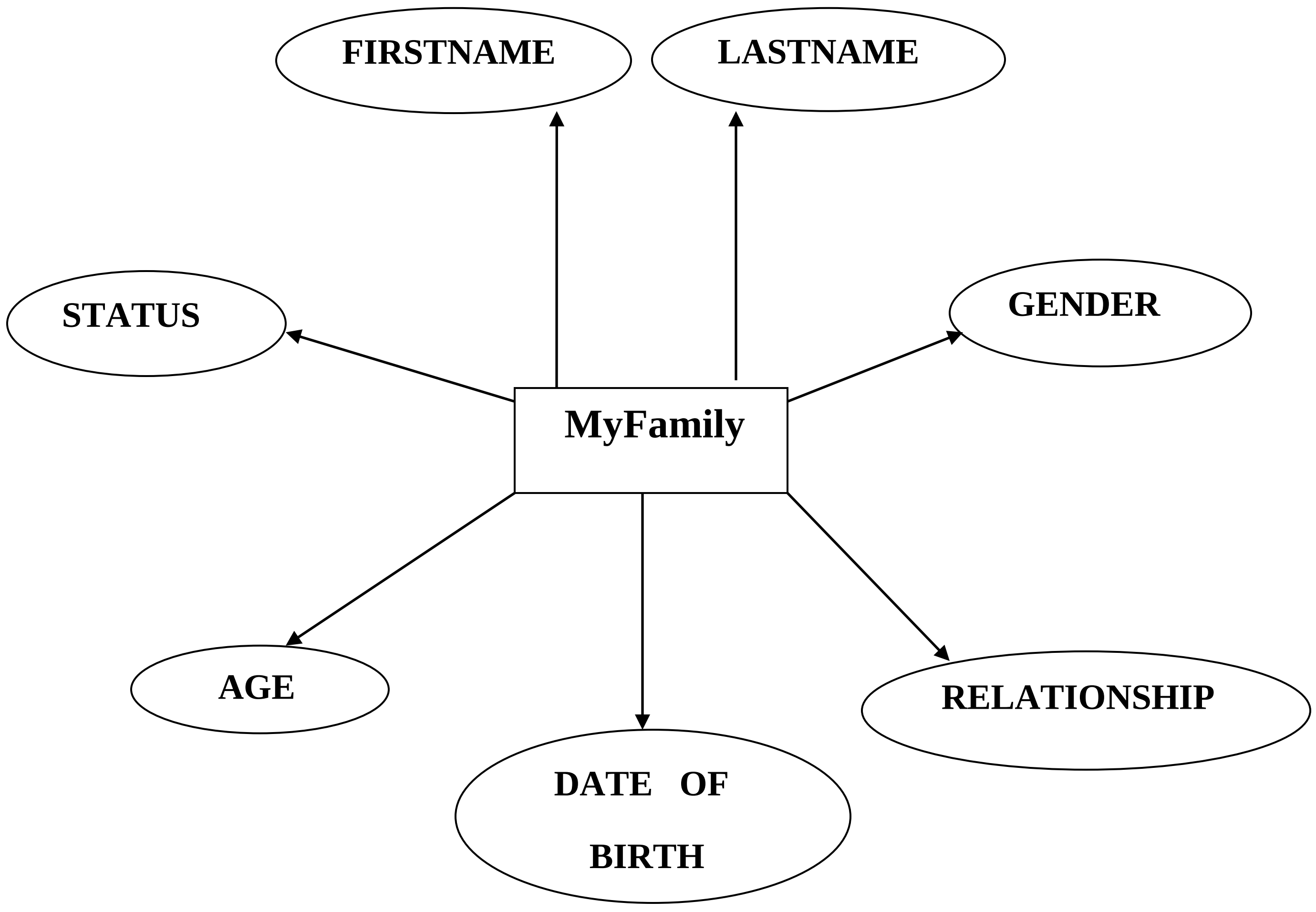


- **Attributes**



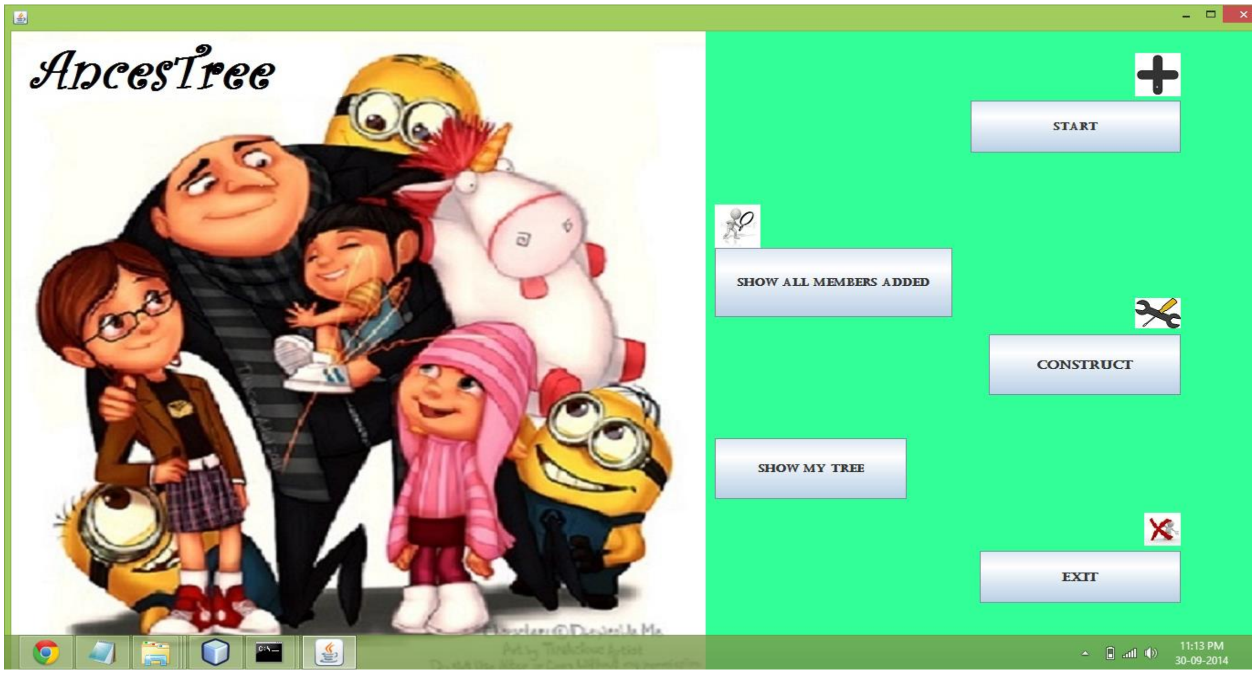
- **Data Flow**



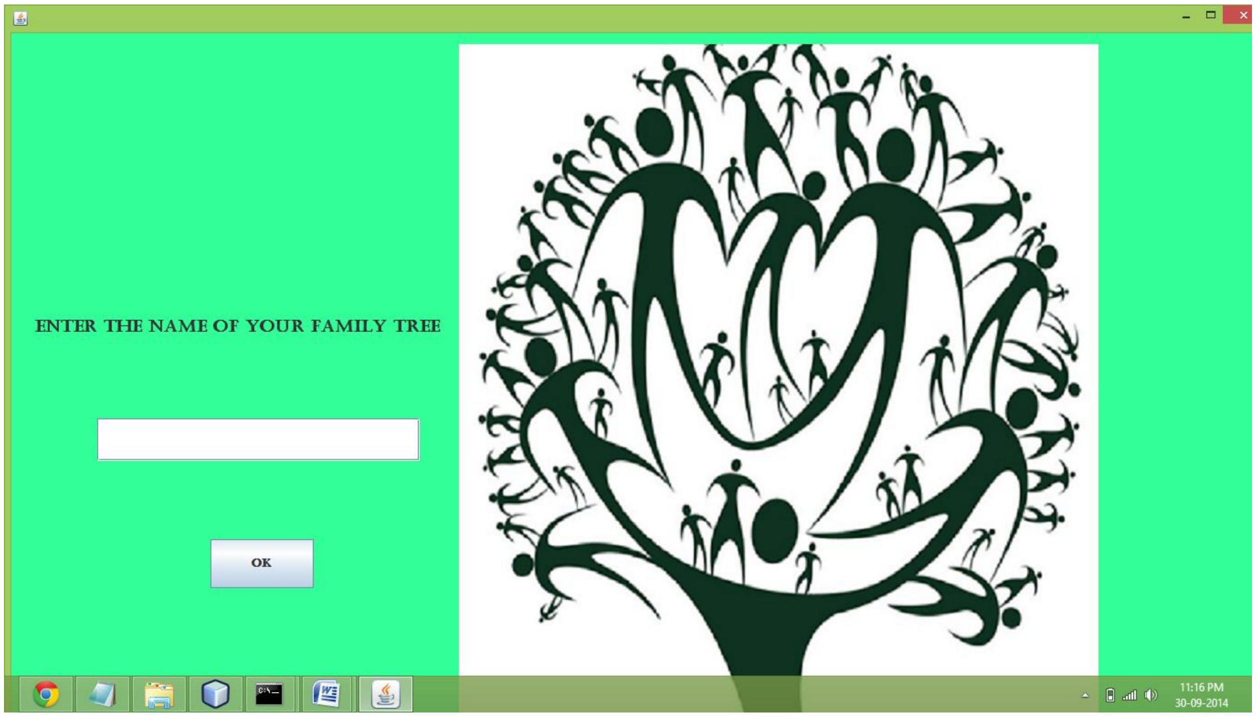

FIRSTNAME
LASTNAME
STATUS
GENDER
MyFamily
AGE
RELATIONSHIP
DATE OF BIRTH

OUTPUT SCREEN (GUI)

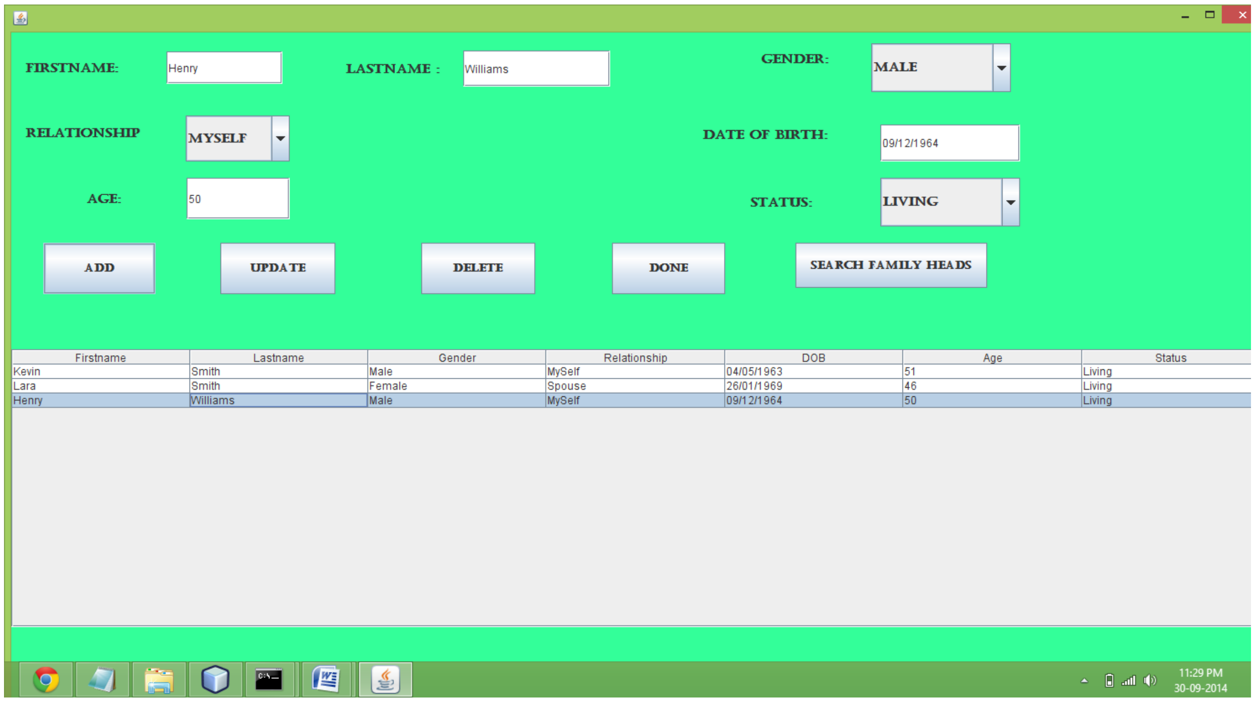
1. **Home Page**



2. **Start**



3. **Add a member.**



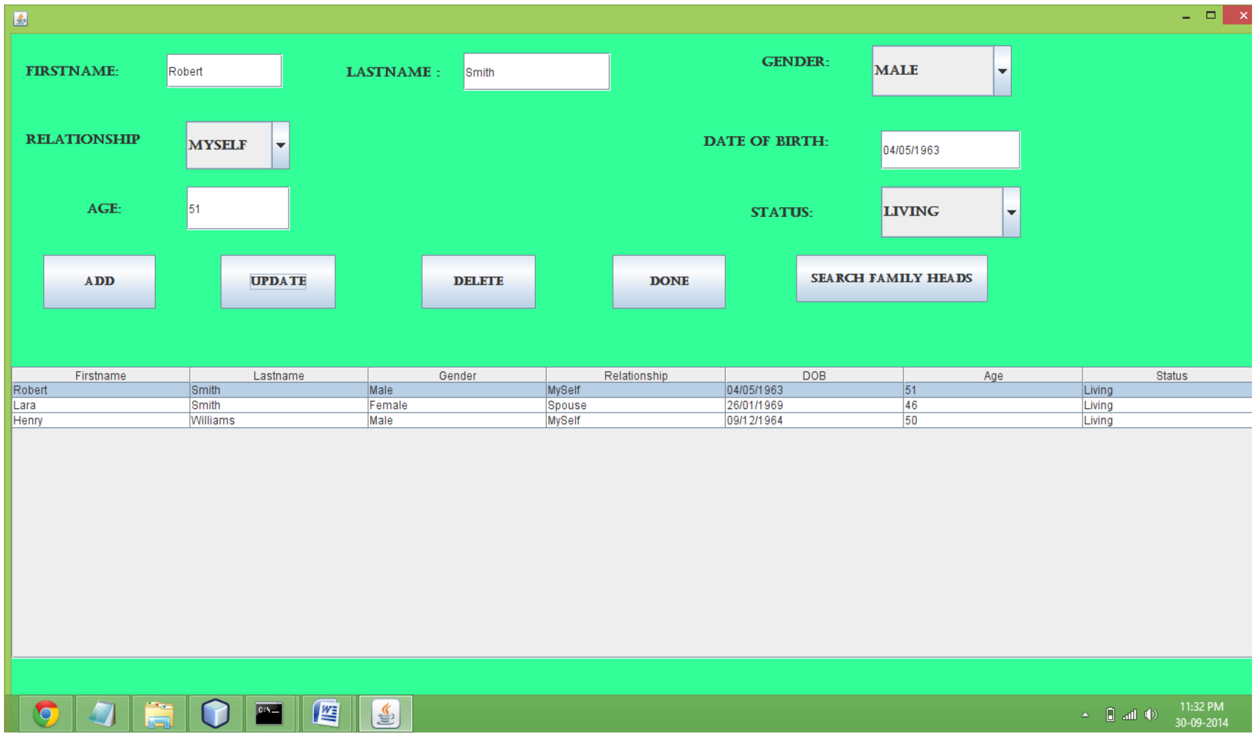
FIRSTNAME: Henry LASTNAME : Williams GENDER: MALE

RELATIONSHIP MYSELF DATE OF BIRTH: 09/12/1964

AGE: 50 STATUS: LIVING

Firstname	Lastname	Gender	Relationship	DOB	Age	Status
Kevin	Smith	Male	MySelf	04/05/1963	51	Living
Lara	Smith	Female	Spouse	26/01/1969	46	Living
Henry	Williams	Male	MySelf	09/12/1964	50	Living

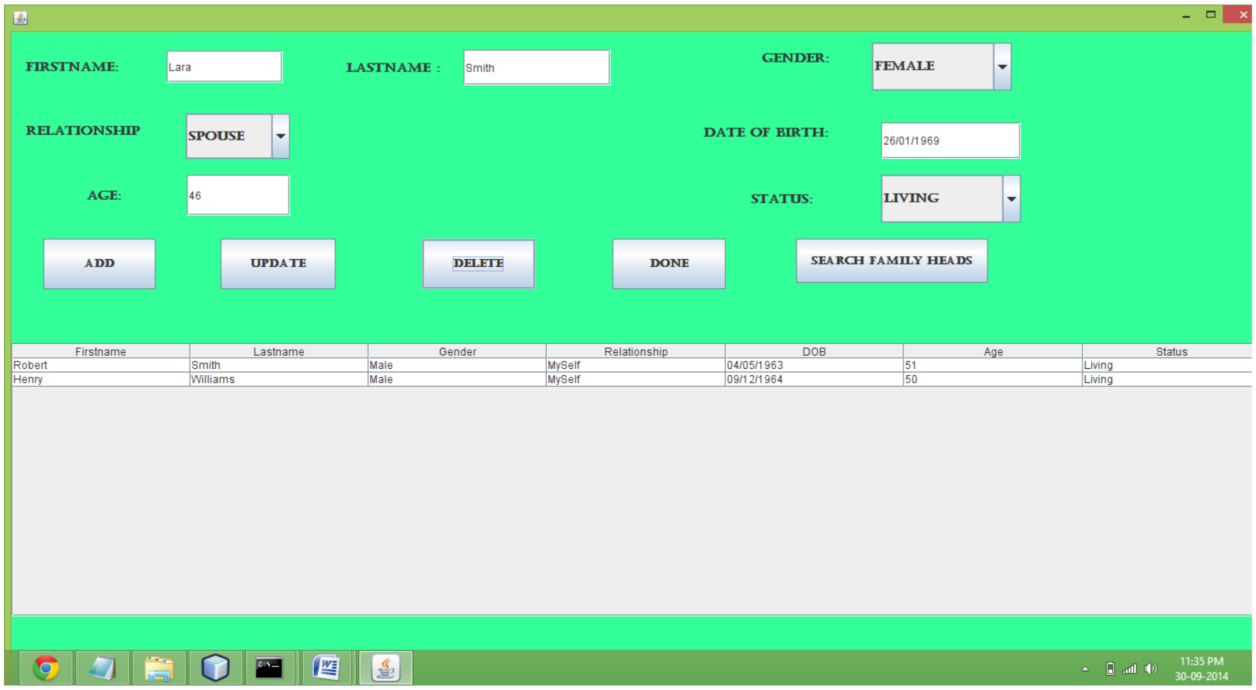
4. **Update a member's information.**



FIRSTNAME: Robert
LASTNAME : Smith
GENDER: MALE
RELATIONSHIP: MYSELF
DATE OF BIRTH: 04/05/1963
AGE: 51
STATUS: LIVING

Firstname	Lastname	Gender	Relationship	DOB	Age	Status
Robert	Smith	Male	MySelf	04/05/1963	51	Living
Lara	Smith	Female	Spouse	26/01/1969	46	Living
Henry	Williams	Male	MySelf	09/12/1964	50	Living

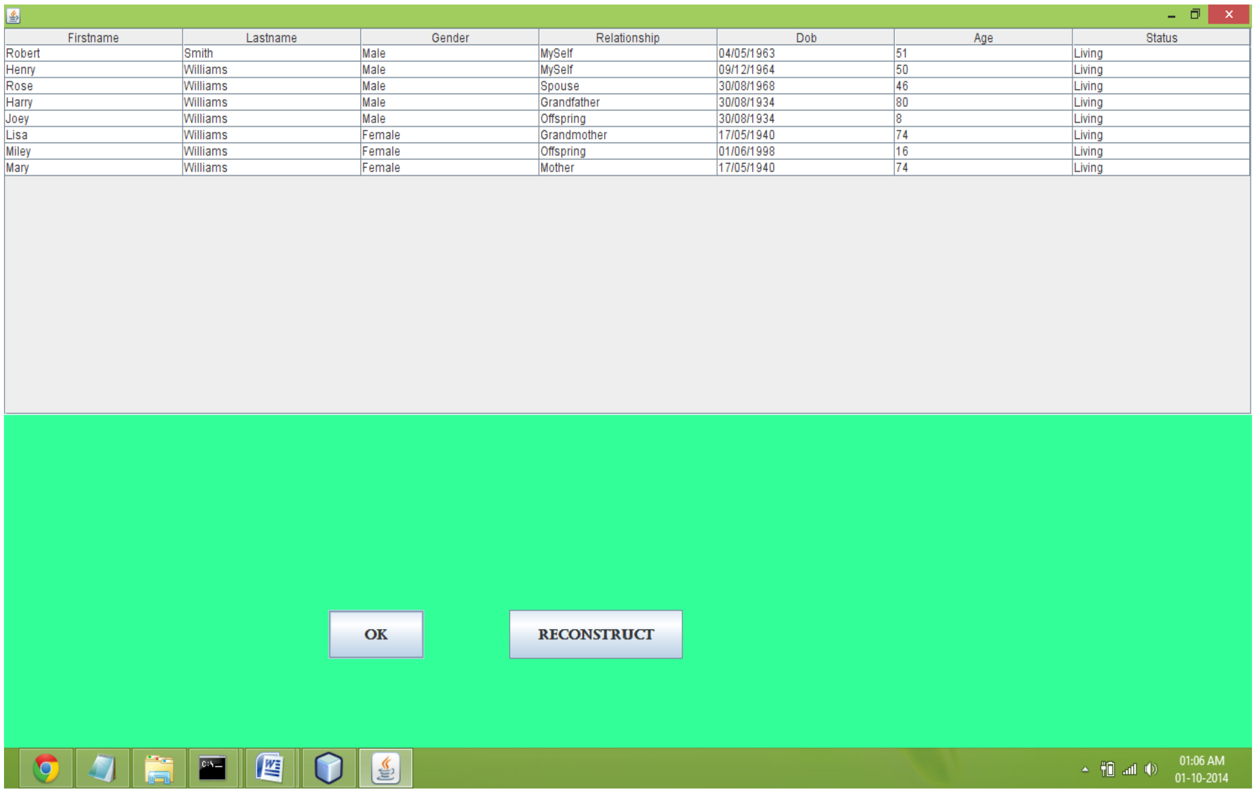
5. Delete a member.



FIRSTNAME: Lara
LASTNAME : Smith
GENDER: FEMALE
RELATIONSHIP: SPOUSE
DATE OF BIRTH: 26/01/1969
AGE: 46
STATUS: LIVING

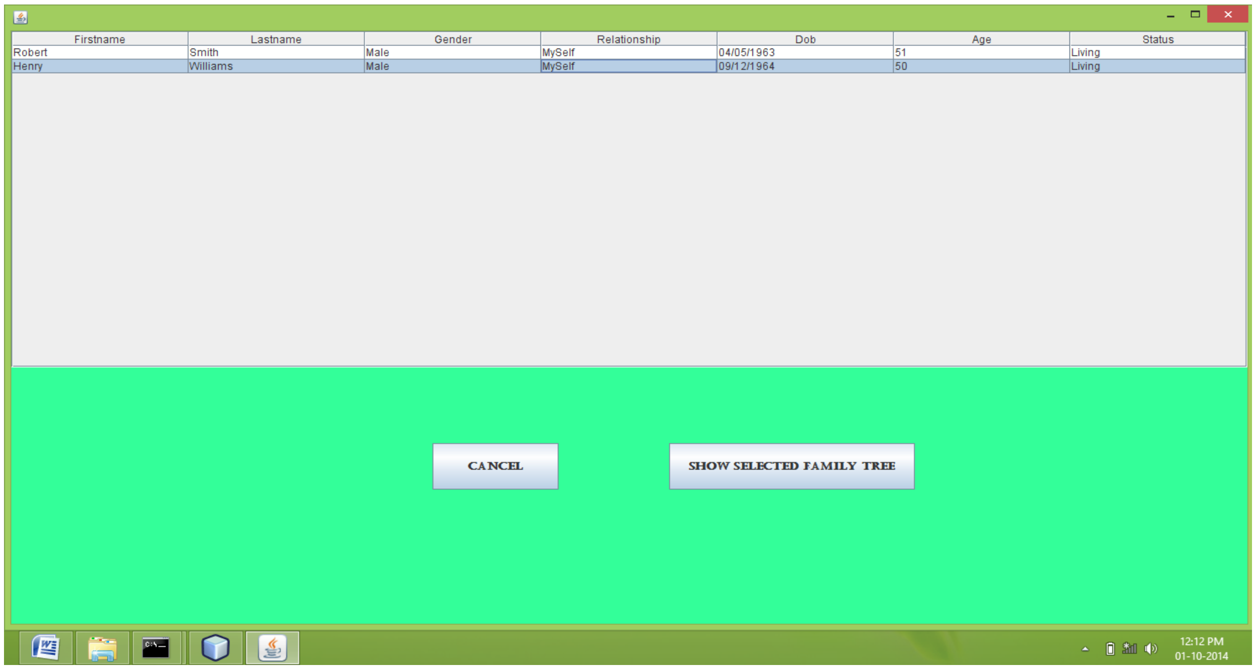
Firstname	Lastname	Gender	Relationship	DOB	Age	Status
Robert	Smith	Male	MySelf	04/05/1963	51	Living
Henry	Williams	Male	MySelf	09/12/1964	50	Living

6. Show all members.



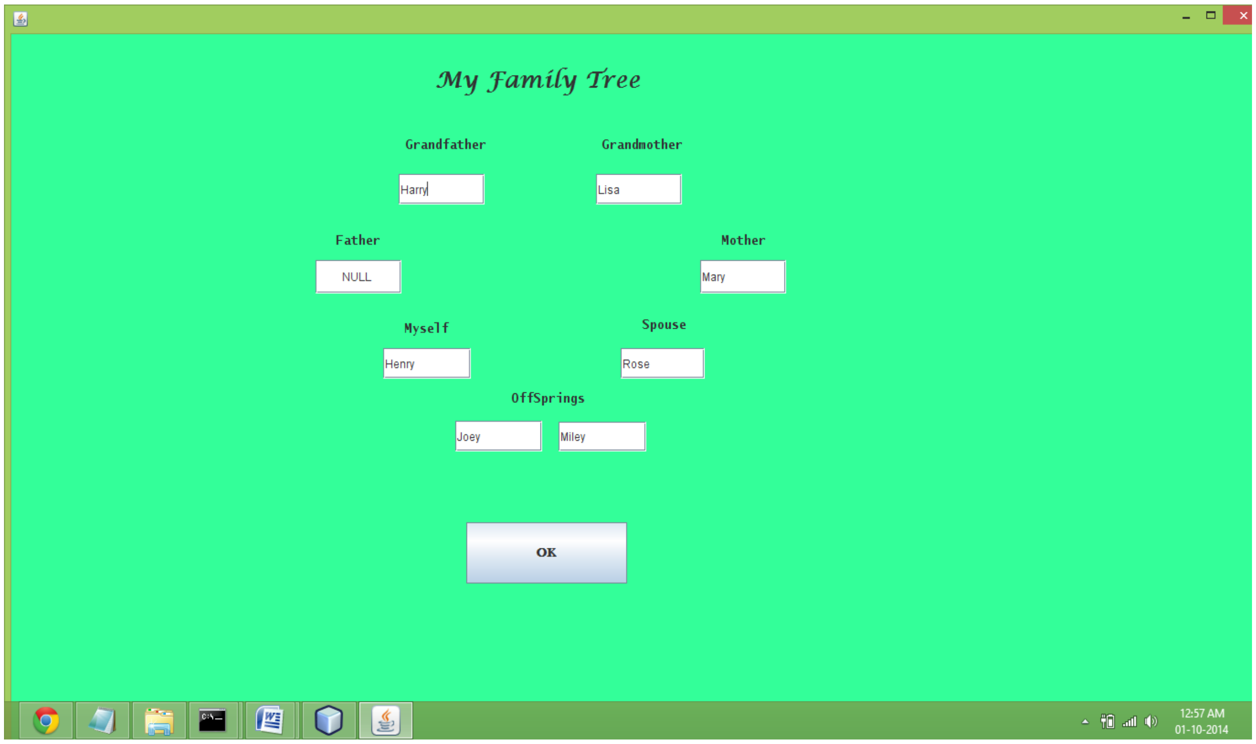
Firstname	Lastname	Gender	Relationship	Dob	Age	Status
Robert	Smith	Male	MySelf	04/05/1963	51	Living
Henry	Williams	Male	MySelf	09/12/1964	50	Living
Rose	Williams	Male	Spouse	30/08/1968	46	Living
Harry	Williams	Male	Grandfather	30/08/1934	80	Living
Joey	Williams	Male	Offspring	30/08/1934	8	Living
Lisa	Williams	Female	Grandmother	17/05/1940	74	Living
Miley	Williams	Female	Offspring	01/06/1998	16	Living
Mary	Williams	Female	Mother	17/05/1940	74	Living

7. Search Family Heads.



Firstname	Lastname	Gender	Relationship	Dob	Age	Status
Robert	Smith	Male	MySelf	04/05/1963	51	Living
Henry	Williams	Male	MySelf	09/12/1964	50	Living

8. Show Family Tree.



SAMPLE CODE

1)Insert:

```
private void jButton2ActionPerformed(java.awt.event.ActionEventevt)
 {//GEN-FIRST:event_jButton2ActionPerformed

        try {

                Firstname =j2.getText();
                Lastname= jTextField1.getText();
                Relationship = jComboBox2.getSelectedItem().toString();
                Gender = jComboBox3.getSelectedItem().toString();
                Dob = jTextField4.getText();
                Age= jTextField5.getText();
                Status=jComboBox1.getSelectedItem().toString();

                DefaultTableModel model = (DefaultTableModel)jTable1.getMode

                if(!this.j2.getText().trim().equals(""))
                {
                        model.addRow(
                        new
                        Object[]{j2.getText(),jTextField1.getText(),jComboBox3.getSelectedItem().toStrn
                        g(),jComboBox2.getSelectedItem().toString(),jTextField5.getText(),jTextField4.ge
                        tText(),jComboBox1.getSelectedItem().toString()});
                        MongoClientmongoClient = new MongoClient("localhost", 27017);
                        DB db = mongoClient.getDB("FamilyDb");

                        DBCollectioncoll = db.getCollection("MyFamily");
                        BasicDBObject doc = new BasicDBObject ("Firstname",
                        Firstname).append("Lastname", Lastname).append("Relationship", Relationship)
                        .append("Gender",Gender).append("Dob",Dob).append("Age",Age).append("St
                        atus", Status);

                        coll.insert(doc);
                        System.out.println(doc);

                }
        }
        catch (UnknownHostException ex)
         {
                Logger.getLogger(Add1.class.getName()).log(Level.SEVERE, null, ex);
        }

}
```

2) Update

```
private void jButton3ActionPerformed(java.awt.event.ActionEventevt)

 {

        try

        {

                DefaultTableModel model = (DefaultTableModel) jTable1.getModel();

                MongoClientmongoClient = new MongoClient("localhost", 27017);

                DB db = mongoClient.getDB("FamilyDb");

                DBCollectioncoll = db.getCollection("MyFamily");

                DBCursor cursor = coll.find();
```

```
if(jTable1.getSelectedRow()!=-1)

{

BasicDBObject newdoc1 = new BasicDBObject();

        newdoc1.append("$set",newBasicDBObject().append("Firstname",this.j2.getTex
        t()));

        BasicDBObject search1 = new
        BasicDBObject().append("Firstname",model.getValueAt(jTable1.getSelectedRow
        (), 0).toString());

coll.update(search1, newdoc1);

model.setValueAt(this.j2.getText(),jTable1.getSelectedRow(), 0);

        BasicDBObject newdoc2 = new BasicDBObject();

        newdoc2.append("$set",newBasicDBObject().append("Lastname",this.jTextField
        1.getText()));

         BasicDBObject search2 = new
        BasicDBObject().append("Lastname",model.getValueAt(jTable1.getSelectedRow
        (), 1).toString());

coll.update(search2, newdoc2);

        model.setValueAt(this.jTextField1.getText(),jTable1.getSelectedRow(), 1);



        BasicDBObject newdoc3 = new BasicDBObject();

        newdoc3.append("$set",new
        BasicDBObject().append("Gender",this.jComboBox3.getSelectedItem().toString()
        ));

         BasicDBObject search3 = new
        BasicDBObject().append("Gender",model.getValueAt(jTable1.getSelectedRow(),
        2).toString());

coll.update(search3, newdoc3);
```

```
        model.setValueAt(jComboBox3.getSelectedItem().toString(),jTable1.getSelectedRow(),
        2);


        BasicDBObject newdoc4 = new BasicDBObject();

        newdoc4.append("$set",new
        BasicDBObject().append("Relationship",this.jComboBox2.getSelectedItem().toString()));

        BasicDBObject search4 = new
        BasicDBObject().append("Relationship",model.getValueAt(jTable1.getSelectedRow(),
        3).toString());

coll.update(search4, newdoc4);

        model.setValueAt(jComboBox2.getSelectedItem().toString(),jTable1.getSelectedRow(),
        3);


        BasicDBObject newdoc5 = new BasicDBObject();

        newdoc5.append("$set",newBasicDBObject().append("Age",this.jTextField5.getText()));

        BasicDBObject search5 = new
        BasicDBObject().append("Age",model.getValueAt(jTable1.getSelectedRow(),
        4).toString());

coll.update(search5, newdoc5);

        model.setValueAt(this.jTextField5.getText(),jTable1.getSelectedRow(), 4);


        BasicDBObject newdoc6 = new BasicDBObject();

        newdoc6.append("$set",newBasicDBObject().append("Dob",this.jTextField4.getText()));

        BasicDBObject search6 = new
        BasicDBObject().append("Dob",model.getValueAt(jTable1.getSelectedRow(),
        5).toString());

coll.update(search6, newdoc6);

        model.setValueAt(this.jTextField4.getText(),jTable1.getSelectedRow(), 5);
```

```
                BasicDBObject newdoc7 = new BasicDBObject();

                newdoc7.append("$set",new
                BasicDBObject().append("Status",this.jComboBox1.getSelectedItem().toString()));

                BasicDBObject search7 = new
                BasicDBObject().append("Status",model.getValueAt(jTable1.getSelectedRow(),
                6).toString());

        coll.update(search7, newdoc7);


        model.setValueAt(jComboBox1.getSelectedItem().toString(),jTable1.getSelectedRow(),6);

        }

}

        catch (UnknownHostException ex)

        {

                Logger.getLogger(Add1.class.getName()).log(Level.SEVERE, null, ex);

        }

}
```

3) Delete

```
private void jButton4ActionPerformed(java.awt.event.ActionEventevt)
 {
        Try
        {
                DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
        MongoClientmongoClient = new MongoClient("localhost", 27017);
        DB db = mongoClient.getDB("FamilyDb");
        DBCollectioncoll = db.getCollection("MyFamily");
        DBCursor cursor = coll.find();
                BasicDBObject search = new
                BasicDBObject().append("Firstname",model.getValueAt(jTable1.getSelectedRow(),
                0).toString());
        coll.remove(search);
        model.removeRow(jTable1.getSelectedRow());
           }
        catch (UnknownHostException ex)
         {
                Logger.getLogger(Add1.class.getName()).log(Level.SEVERE, null, ex);
}
  }
```

CONCLUSION

This system will allow the user to display all the Family information ,and make the access, retrieval and storage of data easy.