INTRODUCTION

Electronic voting systems are increasingly replacing the traditional paper-based voting systems. These systems can make the voting process more convenient and may therefore lead to improved turnout. Electronic recording and counting of votes could be faster, more accurate and less labour intensive. The goal of the E-Voting as a product is to automate the voting process, help in solving fraud problems, decreasing the voting time, and the process of counting.
The e-voting system provides a voting service that allows people to vote from any poll site in the country electronically. This system encompasses legal, regulatory, behavioural, and sociological aspects of the current voting system, while adding additional convenience and security to the overall voting process.

This system is designed to improve the current voting process in the following ways:
1. Allow voters to vote from any poll site in the country without the use of absentee ballots.

2. Reduce the number of legitimate votes not counted by reducing the number of over-votes, and eliminating vote tampering.

3. Improve the registration process by allowing voters to check their registration status prior to voting and centralizing registration databases.

4. Increase voter confidence and improve the voting experience.

FUNCTIONAL REQUIREMENTS

VOTER LOGIN :

Description:

Provides login to the voter for the purpose of the voting.

Functional Requirement :

This feature used by the registered voter to login into system. They are required to enter email id and password before they are allowed to enter the system .The email id and password will be verified and if invalid id is there voter is not allowed to enter the system.

REGISTER NEW VOTER :

Description:

This feature allows to add new voter to the system.

Functional requirements :

System must be able to verify voter's information.

System must have the status of voter's past activity of casting vote .

System must be able to not to allow users younger than 18 years of age to cast vote,

VIEW RESULTS :

Description:

This feature allows the user of the system to view the result of the election before and after casting the vote to his/ her desired candidate.

Functional requirements

System must be able to retrieve the results of voting from the database and display it on the console. The user below the legal voting age are also allowed to view the result of the election in progress.

NON-FUNCTIONAL REQUIREMENTS

Usability:

It is expected that the user should be able to vote easily. Administration of the page also should be user friendly. Provide step by step guide for both admin and users. User should complete voting in a few minutes.

Reliability:

The system should be reliable. Security is a major concern for an e-voting system. Process used in this system should be secure enough to be able to meet the requirements mentioned for e-voting. It requires database connections. Changes can be done in the databases to store the votes. All changes needs to be confirmed and if the transfer is complete the confirmation should be displayed. The changes should be monitored. It must be fast to process the request.

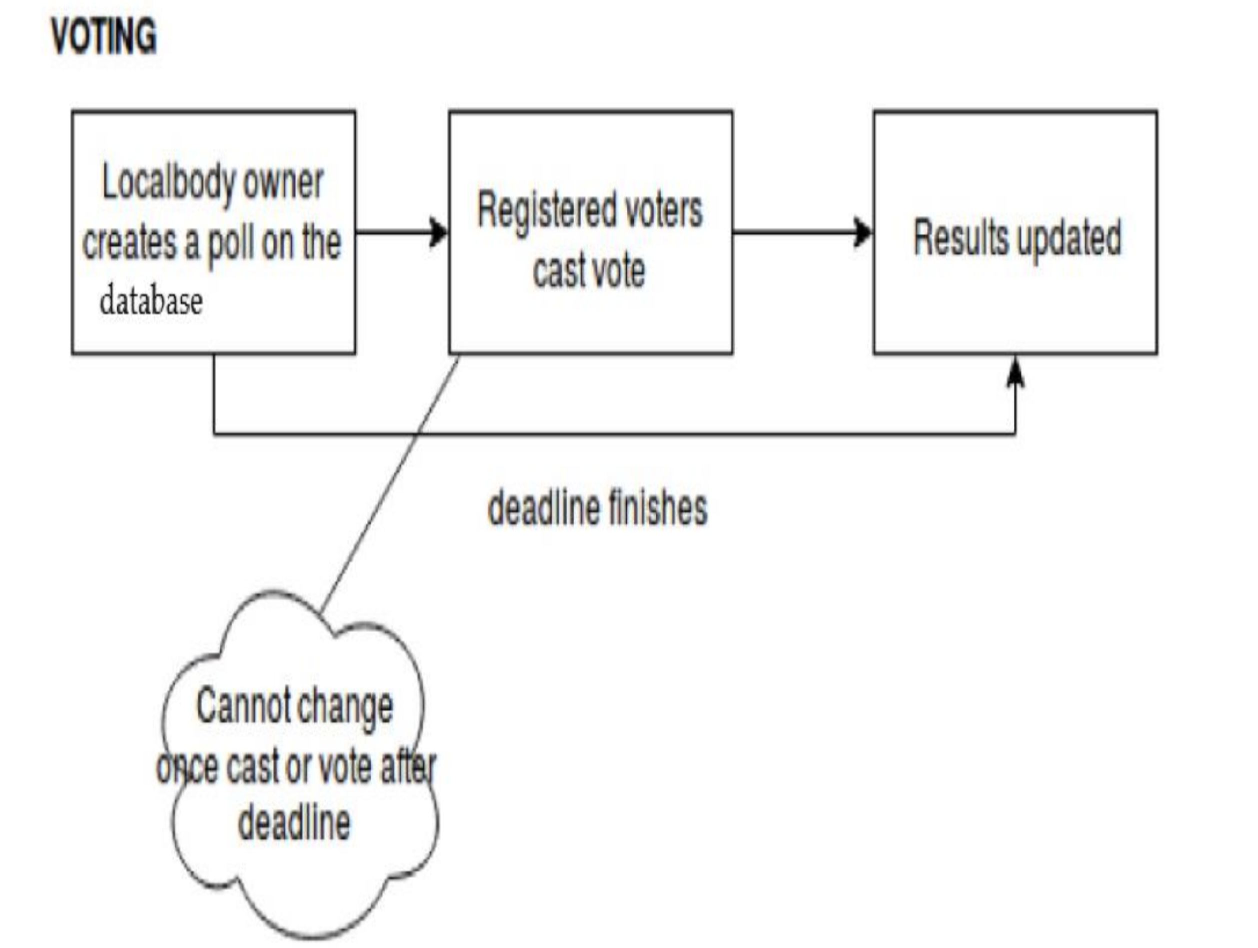
Performance:

There might be many users accessing to the database server simultaneously. As an e-voting tool performance shouldn't be affected much and response time for submitted page should be less than a minute.

Maintenance:

System maintenance must be easy. System will be used be using some additional components. Each of these components might need an update or changes. Applying these changes during the maintenance should be relatively easy.

BLOCK DIAGRAM


VOTING
Localbody owner creates a poll on the database
Registered voters cast vote
Results updated
deadline finishes
Cannot change once cast or vote after deadline

SOURCE CODE

1. **VotingApp.java**

```
package votingapp;

import java.sql.SQLException;
import java.util.Scanner;

public class VotingApp {
	Scanner sc = new Scanner(System.in);

	void adduser() throws ClassNotFoundException, SQLException {
		userinfo ui = new userinfo();
		String name;
		String email;
		String password;
		String age;
		String status = "no";
		System.out.println("Enter Your Name:");
		name = sc.next();
		System.out.println();

		System.out.println("Enter Your Email");
		email = sc.next();
		System.out.println();

		System.out.println("Enter Your Password");
		password = sc.next();
		System.out.println();

		System.out.println("Enter Your Age");
		age = sc.next();
		System.out.println();
		ui.addnew(name, email, password, age, status);
	}

	void login() throws ClassNotFoundException, SQLException {
		userinfo ui = new userinfo();
		votes v = new votes();
		String email;
		String password;
		System.out.println("Enter Your Email: ");
		email = sc.next();
		System.out.println();
		System.out.println("Enter Your Password: ");
		password = sc.next();
		System.out.println();
		if (ui.passval(email, password)) {
			if (ui.ageval(email)) {
				if (ui.statusval(email)) {
					System.out.println("Verification Successfull, Enter ID to Vote");
System.out.println("1. Candidate 1");
                                        System.out.println("2. Candidate 2");
                                        System.out.println("3. Candidate 3");
                                        System.out.println("4. Candidate 4");
                                        System.out.println("5. Candidate 5");
                                        int id = sc.nextInt();
                                        v.vote(id, email);
                                        System.out.println("Voted Successfully");
                                } else {
                                        System.out.println("You Have Already Voted");
                                }
                        } else {
                                System.out.println("You are Too Young to Vote");
                        }
                } else {
                        System.out.println("Invalid Password");
                }
        }

        public static void main(String[] args) throws ClassNotFoundException, SQLException {
                Scanner sc = new Scanner(System.in);
                VotingApp va = new VotingApp();
                votes v = new votes();
                // userinfo ui=new userinfo();
                while (true) {
                        System.out.println("***** WELCOME *****");
                        System.out.println();
                        System.out.println("1. Login");
                        System.out.println("2. Register");
                        System.out.println("3. Result");
                        System.out.println("4. Exit");
                        System.out.println();
                        System.out.println("Enter Your Choice: ");
                        int c = sc.nextInt();
                        switch (c) {
                        case 1:
                                va.login();
                                break;
                        case 2:
                                va.adduser();
                                break;
                        case 3:
                                v.result();
                                break;
                        case 4:
                                break;
                        default:
                                System.out.println("Enter Correct Input");

                        }

                }
```

2. **userinfo.java**

```
package votingapp;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class userinfo {

	boolean addnew(String name, String email, String password, String age, String status)
			throws ClassNotFoundException, SQLException {
		Class.forName("com.mysql.jdbc.Driver");
		Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/stqa?autoReconnect=true&useSSL=false",
				"root", "mohit");
		Statement stm = con.createStatement();
		String sql = "insert into voter values('" + name + "','" + email + "','" + password + "','" +
age + "','"
				+ status + "');";
		boolean b = stm.execute(sql);
		System.out.println("You Have Registered Successfully");
		return b;
	}

	boolean passval(String email, String password) throws SQLException,
ClassNotFoundException {
		Class.forName("com.mysql.jdbc.Driver");
		String em = "";
		Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/stqa?autoReconnect=true&useSSL=false",
				"root", "mohit");
		Statement stm = con.createStatement();
		String sql = "select password from voter where email='" + email.trim() + "';";
		ResultSet rs = stm.executeQuery(sql);
		while (rs.next()) {
			em = rs.getString("password").trim();
		}

		if (em.equals(password)) {
			return true;
		} else {
			return false;
		}
	}

	boolean ageval(String email) throws ClassNotFoundException, SQLException {

		String ag = "";
		Class.forName("com.mysql.jdbc.Driver");


		Connection con = 
DriverManager.getConnection("jdbc:mysql://localhost/stqa?autoReconnect=true&useSSL=false",
				"root", "mohit");
		Statement stm = con.createStatement();
		String sql = "select age from voter where email='" + email.trim() + "';";
		ResultSet rs = stm.executeQuery(sql);
		while (rs.next()) {
			ag = rs.getString("age");
		}
		int a = Integer.parseInt(ag);
		if (a >= 18) {
			return true;
		} else {
			return false;
		}

	}

	boolean statusval(String email) throws ClassNotFoundException, SQLException {

		Class.forName("com.mysql.jdbc.Driver");
		String stat = "";
		Connection con = 
DriverManager.getConnection("jdbc:mysql://localhost/stqa?autoReconnect=true&useSSL=false",
				"root", "mohit");
		Statement stm = con.createStatement();
		String sql = "select status from voter where email='" + email.trim() + "';";
		ResultSet rs = stm.executeQuery(sql);
		while (rs.next()) {
			stat = rs.getString("status").trim();
		}
		if (stat.equals("no")) {
			return true;
		} else {
			return false;
		}
	}

}
```

3. **votes.java**

```
package votingapp;

import java.sql.*;

public class votes {
	int result() throws ClassNotFoundException, SQLException {
		int flag = 0;
		Class.forName("com.mysql.jdbc.Driver");

		Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/stqa?autoReconnect=true&useSSL=false",
				"root", "mohit");
		Statement stm = con.createStatement();
		String sql = "select * from votes;";
		ResultSet rs = stm.executeQuery(sql);
		while (rs.next()) {
			System.out.println(rs.getString("name") + "   " + rs.getString("count"));
		}
		if (rs != null)
			flag = 1;

		return flag;
	}

	int vote(int i, String email) throws SQLException, ClassNotFoundException {
		String inic = "";
		String name = "Candidate " + Integer.toString(i);
		Class.forName("com.mysql.jdbc.Driver");

		Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/stqa?autoReconnect=true&useSSL=false",
				"root", "mohit");
		Statement stm = con.createStatement();
		String sql = "select count from votes where name='" + name + "';";
		ResultSet rs = stm.executeQuery(sql);
		while (rs.next()) {
			inic = rs.getString("count");
		}
		int newc = 1 + Integer.parseInt(inic);
		String n = Integer.toString(newc);
		String sql1 = "update votes set count='" + n + "' where name='" + name + "';";
		int r = stm.executeUpdate(sql1);
		String s = "yes";
		String sql2 = "update voter set status='" + s + "' where email='" + email + "';";
		stm.executeUpdate(sql2);
		return r;
	}
```

4. **VotingAppTest.java**

```
class VotingAppTest {
	Scanner sc = new Scanner(System.in);
	@Test
	void testAdduser() throws ClassNotFoundException, SQLException {
		System.out.println("Enter Name: ");
		String name = sc.next();
		Pattern p = Pattern.compile("^[a-zA-Z]*$");
		Matcher m = p.matcher(name);
		boolean output = m.matches();
		assertTrue("Test Case Failed, INVALID FORMAT FOR NAME",output);
	}
}
```

5. **votesTest.java**

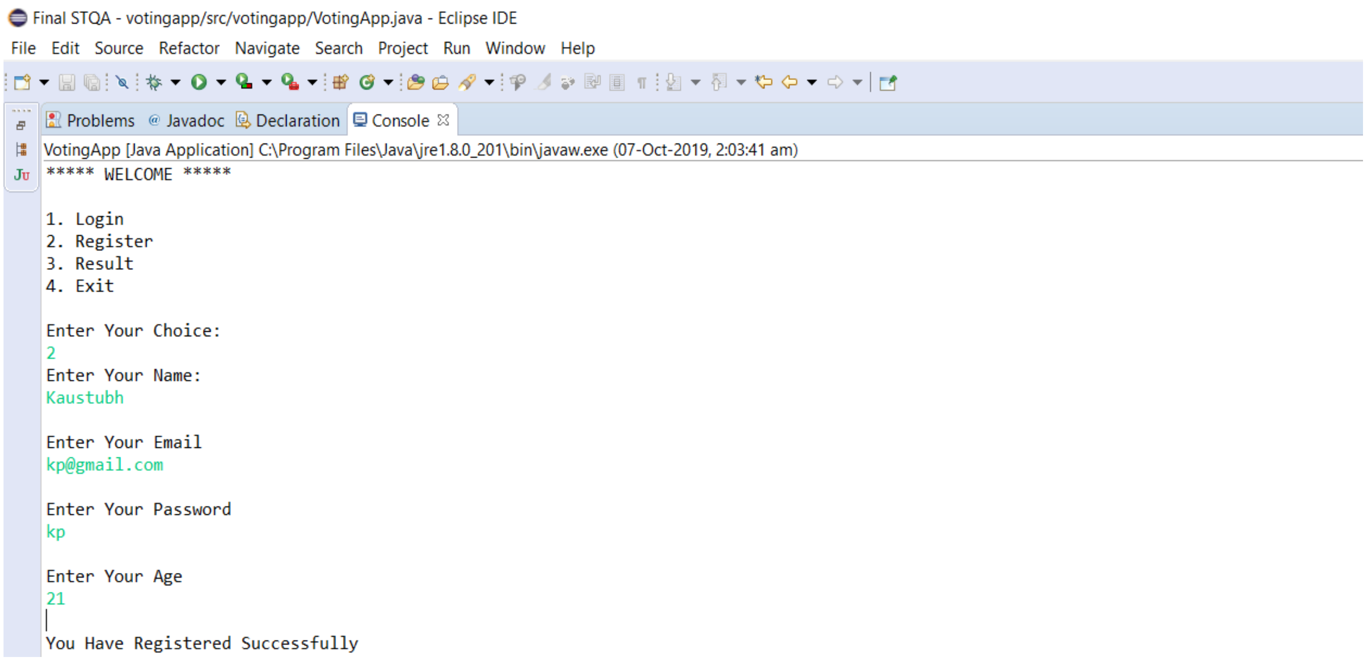
```
package votingapp;

import static org.junit.jupiter.api.Assertions.*;

import java.sql.SQLException;
import java.util.Scanner;

import static org.junit.Assert.assertTrue;

import org.junit.jupiter.api.Test;

class votesTest {
	userinfoTest u = new userinfoTest();
	votes v = new votes();
	Scanner sc = new Scanner(System.in);

	@Test
	void testResult() throws ClassNotFoundException, SQLException {
		int access = v.result();
		assertEquals(1, access,"Test Case Failed, Unable to Access Database");
	}

	@Test
	void testVote() throws ClassNotFoundException, SQLException {
		System.out.println("Enter The Candidate Number: ");
		int canNo = sc.nextInt();
		int flag = 0;
		if(canNo > 0 && canNo < 6)
			flag = 1;
		assertEquals(1,flag,"Test Case Failed, No Such Candidate in List");

	}
}
```

6. **userinfoTest.java**

```
package votingapp;

import static org.junit.Assert.assertTrue;
import static org.junit.jupiter.api.Assertions.*;

import java.sql.SQLException;
import java.util.Scanner;

import org.junit.jupiter.api.Test;

class userinfoTest {
	VotingApp vo = new VotingApp();
	userinfo u = new userinfo();
	Scanner sc = new Scanner(System.in);

	@Test
	void testaddnew() throws ClassNotFoundException, SQLException {

		System.out.println("Enter Name: ");
		String reqName = sc.next();
		System.out.println("Enter Email: ");
		String reqEmail = sc.next();
		System.out.println("Enter Password: ");
		String reqPass = sc.next();
		System.out.println("Enter Age: ");
		String reqAge = sc.next();
		String status = "no";
		boolean expectedOutput = false;
		boolean addedUser = u.addnew(reqName, reqEmail, reqPass, reqAge, status);
		assertEquals(expectedOutput,addedUser,"Test Case Failed, Failed To Register");

	}

	@Test
	void testPassval() throws ClassNotFoundException, SQLException {

		System.out.println("Enter Email: ");
		String reqEmail = sc.next();
		System.out.println("Enter Password: ");
		String reqPass = sc.next();
		boolean passChecked = u.passval(reqEmail, reqPass);
		assertTrue("Test Case Failed, Password Incorrect",passChecked);

	}
	@Test
	void testAgeval() throws ClassNotFoundException, SQLException {
		System.out.println("Enter Email: ");
		String reqEmail = sc.next();
		boolean ageChecked = u.ageval(reqEmail);
		assertTrue("Test Case Failed, Age is < 18 ",ageChecked);
	}

	@Test
	void testStatusval() throws ClassNotFoundException, SQLException {
		System.out.println("Enter Email: ");
		String reqEmail = sc.next();
		boolean expectedOutput = false;
		boolean statusChecked = u.ageval(reqEmail);
		assertEquals(expectedOutput, statusChecked,"Test Case Failed, Voter Already Voted");

	}

}
```

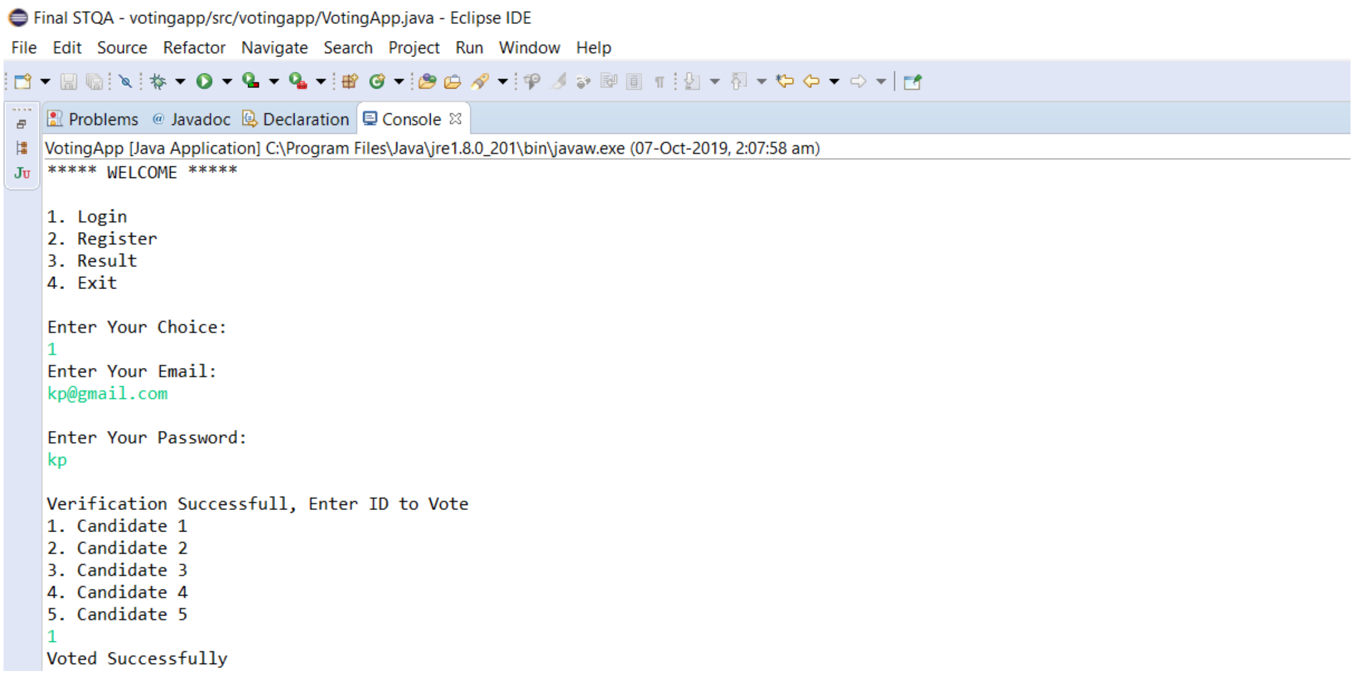
USER INTERFACE

1. REGISTRATION PROCESS


Final STQA - votingapp/src/votingapp/VotingApp.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console

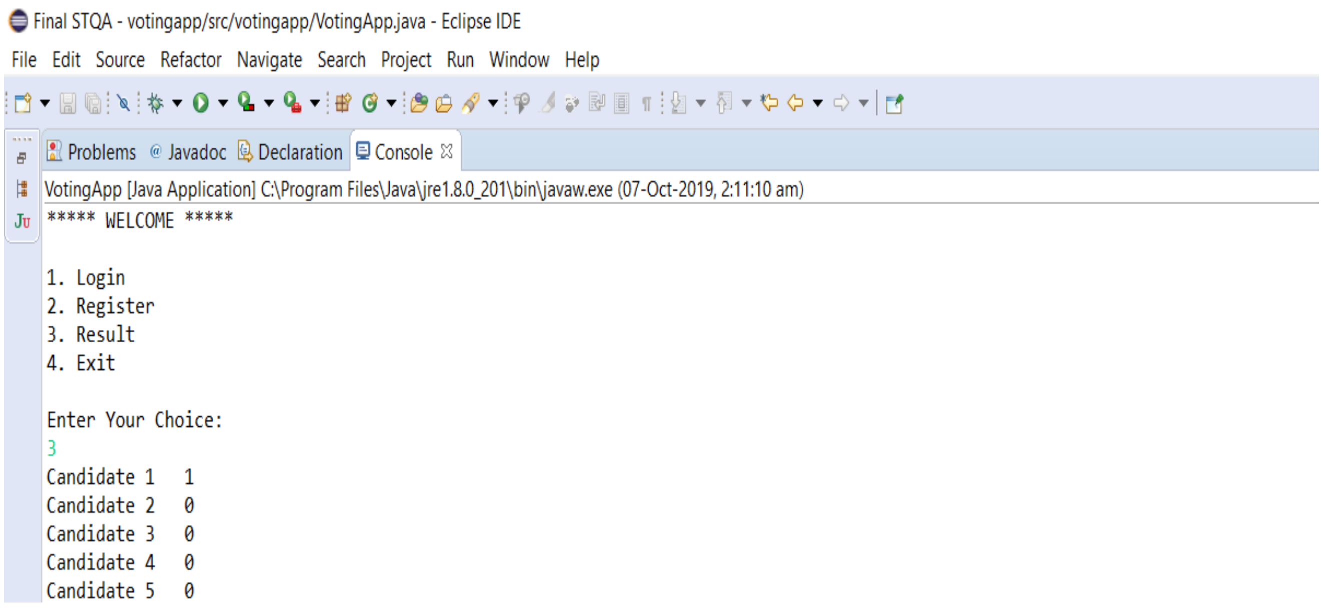
```
VotingApp [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (07-Oct-2019, 2:03:41 am)
***** WELCOME *****

1. Login
2. Register
3. Result
4. Exit

Enter Your Choice:
2
Enter Your Name:
Kaustubh

Enter Your Email
kp@gmail.com

Enter Your Password
kp

Enter Your Age
21

You Have Registered Successfully
```

2. LOGIN PROCESS


Final STQA - votingapp/src/votingapp/VotingApp.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console

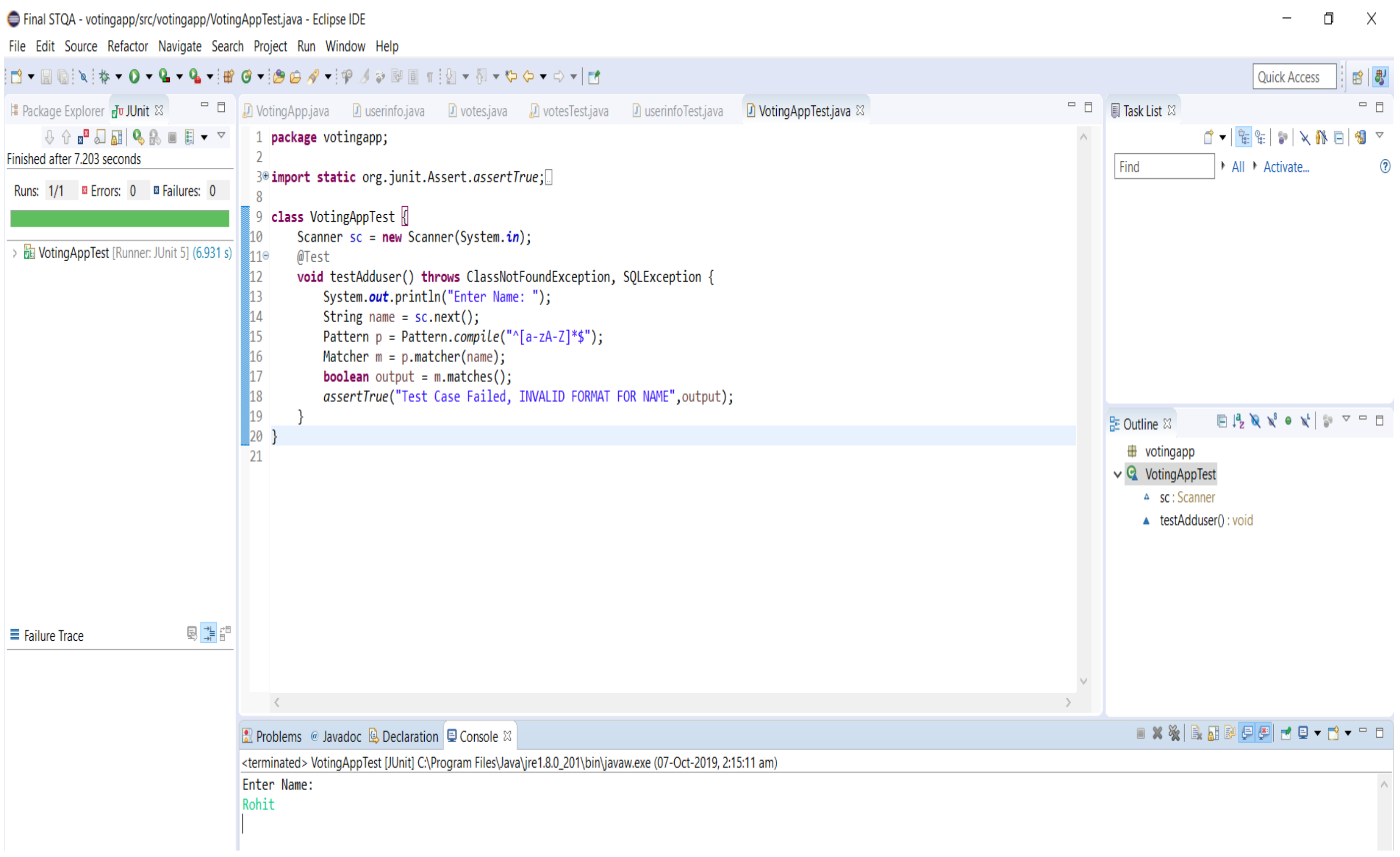
```
VotingApp [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (07-Oct-2019, 2:07:58 am)
***** WELCOME *****

1. Login
2. Register
3. Result
4. Exit

Enter Your Choice:
1
Enter Your Email:
kp@gmail.com

Enter Your Password:
kp

Verification Successfull, Enter ID to Vote
1. Candidate 1
2. Candidate 2
3. Candidate 3
4. Candidate 4
5. Candidate 5
1
Voted Successfully
```

3. VIEWING RESULTS


Final STQA - votingapp/src/votingapp/VotingApp.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console

```
VotingApp [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (07-Oct-2019, 2:11:10 am)
***** WELCOME *****

1. Login
2. Register
3. Result
4. Exit

Enter Your Choice:
3
Candidate 1   1
Candidate 2   0
Candidate 3   0
Candidate 4   0
Candidate 5   0
```

4. PASSING A TEST CASE

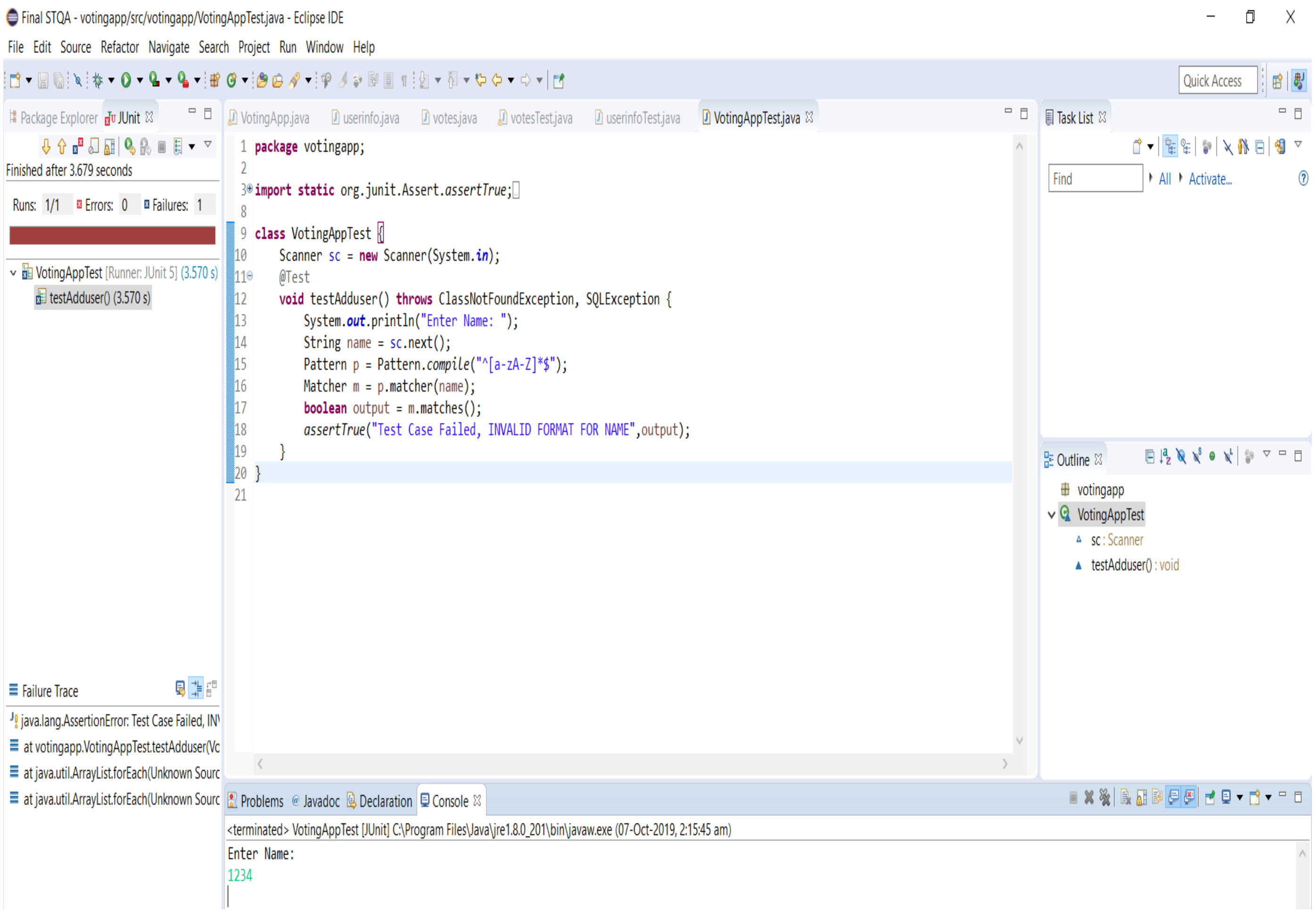


```
Finished after 7.203 seconds
Runs: 1/1   Errors: 0   Failures: 0
VotingAppTest [Runner: JUnit 5] (6.931 s)
```

```java
package votingapp;

import static org.junit.Assert.assertTrue;

class VotingAppTest {
    Scanner sc = new Scanner(System.in);
    @Test
    void testAdduser() throws ClassNotFoundException, SQLException {
        System.out.println("Enter Name: ");
        String name = sc.next();
        Pattern p = Pattern.compile("^[a-zA-Z]*$");
        Matcher m = p.matcher(name);
        boolean output = m.matches();
        assertTrue("Test Case Failed, INVALID FORMAT FOR NAME",output);
    }
}
```

```
<terminated> VotingAppTest [JUnit] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (07-Oct-2019, 2:15:11 am)
Enter Name:
Rohit
```

5. FAILING A TEST CASE



Finished after 3.679 seconds

Runs: 1/1 Errors: 0 Failures: 1

VotingAppTest [Runner: JUnit 5] (3.570 s)
testAdduser() (3.570 s)

```java
package votingapp;

import static org.junit.Assert.assertTrue;

class VotingAppTest {
    Scanner sc = new Scanner(System.in);
    @Test
    void testAdduser() throws ClassNotFoundException, SQLException {
        System.out.println("Enter Name: ");
        String name = sc.next();
        Pattern p = Pattern.compile("^[a-zA-Z]*$");
        Matcher m = p.matcher(name);
        boolean output = m.matches();
        assertTrue("Test Case Failed, INVALID FORMAT FOR NAME",output);
    }
}
```

```
java.lang.AssertionError: Test Case Failed, IN
at votingapp.VotingAppTest.testAdduser(Vc
at java.util.ArrayList.forEach(Unknown Sourc
at java.util.ArrayList.forEach(Unknown Sourc
```

```
<terminated> VotingAppTest [JUnit] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (07-Oct-2019, 2:15:45 am)
Enter Name:
1234
```

TEST PLAN DESCRIPTION

- **Introduction:**
  The goal of the E-Voting as a product is to automate the voting process, help in solving fraud problems, decreasing the voting time, and the process of counting. The e-voting system provides a voting service that allows people to vote from any poll site in the country electronically.

- **Intended Audience** :
  This test plan is made for system testing of this application. The test plan will be referred by,
  - Development manager and development team
  - Election Managers

- **Intended Users** :
  This application will be used by the eligible voters, election commission as well as the non-eligible voters to just view the results.

- **Test Scope** :
  System testing shall cover,
  - User interface testing
  - Functionality testing

- **Out of Scope :**
  The following types of testing are out of scope for this application
  - Security testing
  - Performance, volume, and stress testing
  - Usability testing

- **Test Objectives :**
  - Targeted number of defects—50
  - Targeted number of test scenario—7
  - Targeted number of test cases—10
  - Number of iterations required—5
  - Test scenario writing per day—10
  - Test case writing per day—50
  - Test case execution per day—100
  - Coverage of requirements P1 = 100%
    P2 = 50% & P3 = 10%

- **Assumptions of Testing :**
  - Users understand English and they are capable of using a computer
  - Users understand only integers (they are not expected to use decimal)
  - Application will be working on Windows 2000 professional. Other operating systems, if any, are compatible.
  - Application is tested on Pentium P3, 64MB RAM, 250 MHz processor. Other configurations used, if any, are compatible.
  - Testing considers right-handed user

- **Risk Analysis :**
  - Application will not be able to handle non-integers
  - Compatibility on other operating system and machine configuration can be a problem.
  - Application may not work for left-handed users
  - User may not understand English

- **Workflow** :

The project manager communicates the availability of new build along with the records of reviews and unit testing to test manager. The application is installed on the machine as defined in prerequisites (P3, 64MB RAM, 250 MHz professor). Test manager checks the entry criteria for testing. If the application passes the entry criteria, it is taken for iteration testing.

Test cases are stored in configuration library. Test cases are executed as per the sequence mentioned and results are added in the test case as actual results. Difference between expected result and actual result is taken for evaluation. All differences between expected results and actual results are reproduced. Video clips are created for the defects. Defects are added in defect management tool

.

- **Test Design** :

The application will be tested as per steps. Verification activities will be covered in quality plan. This is a system test plan and other levels of testing are covered by separate plans at each level.

- **Roles and Responsibilities** :

Roles and responsibilities of stakeholders are as defined in Table below

Responsibilities/Role	Test manager	Senior tester	Tester	Customer
test planning	X			
test scenario writing		X		
test cases writing		X	X	
Test data definition			X	
test execution			X	
Query resolution				X

- **Test Data Management**
  - Test plan, test scenario and test cases will be under configuration management control and kept on centralized server
  - Test reports will be stored on the same server.
  - Backup will be taken on tape, once per week.

- **Test Environment**:

  Test environment will be made of individual machines.
  - Windows 10
  - 2GB RAM
  - Intel i3 Processor

- **Test Entry Criteria :**

  - All review comments are closed
  - All unit testing defects are closed
  - Application can be installed and launched
  -

- **Test Exit Criteria** :

  Test cases are completed and all defects found in testing are closed, retested and found to be closed.

- **Test Suspension Criteria** :

Tests will be suspended in the following circumstances,

  - Test cases written do not find sufficient number of defects
  - Test case writing and test data definition are not completed
  - Application cannot be installed
  - Defects found in 1st/2nd/3rd iteration are more than 500.

- **Test Resumption Criteria :**

  - All defects are reviewed and corrective/preventive actions are taken.
  - All review comments are closed.
  - All unit testing defects are fixed, retested and found to be closed.
  - Application can be installed.

- **Tester's Training** :

  The following training is planned for testers.

  - Testers will be trained on basics of triangle geometry

Tester's training for writing test plan, test scenarios, test cases, execution of test cases and defect logging is already done.

TEST CASES

Test Case 1:

- Test Precondition : Application is launched with its main window.
- Test Case Name and Number : Test_Name, TC001
- Type of Testing : input testing
- Objectives : To check whether input provided is of valid format
- Valid/Invalid Conditions Valid condition
- Expected result : Application navigates to the next menu, allowing user to login.

Test Case 2:

- Test Precondition : User of the system has logged in successfully.
- Test Case Name and Number : Test_Database_Access, TC002
- Type of Testing : functionality testing
- Objectives : To check whether we are able to fetch data from database.
- Valid/Invalid Conditions Valid condition
- Expected result : Application displays the result of the election.

Test Case 3:

- Test Precondition : Application is launched with its main window.
- Test Case Name and Number : Test_Candidate_Number, TC003
- Type of Testing : input testing
- Objectives : To check whether correct input is provided by user.
- Valid/Invalid Conditions Valid condition
- Expected result : Application increments the vote count of the selected candidate.

Test Case 4:

- Test Precondition : Application is launched with its main window.
- Test Case Name and Number : Test_Database_Access, TC004
- Type of Testing : functionality testing
- Objectives : To check whether the data entered by user is submitted to the database successfully.
- Valid/Invalid Conditions Valid condition
- Expected result : Application registers a new user into its database.

Test Case 5:

- Test Precondition : Application is launched with its main window.
- Test Case Name and Number : Test_Password, TC005
- Type of Testing : input testing
- Objectives : To check whether correct input is provided by user.
- Valid/Invalid Conditions Valid condition
- Expected result : Application allows the user to vote.

Test Case 6:

- Test Precondition : The user has logged in successfully.
- Test Case Name and Number : Test_Age, TC006
- Type of Testing : condition testing
- Objectives : To check whether valid input is provided by user.
- Valid/Invalid Conditions Valid condition
- Expected result : Application allows the user to vote as he/she is above the legal age of voting.

Test Case 7:

- Test Precondition : The user has logged in successfully.
- Test Case Name and Number : Test_Status, TC007
- Type of Testing : condition testing
- Objectives : To check whether an user is trying to cast vote again even though he/she has already voted.
- Valid/Invalid Conditions Valid condition
- Expected result : Application allows only valid voters to cast the vote.

CONCLUSION

Thus we implemented an e-Voting system with the help of java platform. The motive of developing this application is to automate the process of voting, make it hassle free, make it less prone to errors and reduce the frauds. The application also aims at acquiring voters' trust and giving them the confidence of safety of their casted vote and secrecy of their personal information.

The application was also successfully tested with the help of Junit**.** JUnit is an open source Unit Testing Framework for JAVA. It is useful for Java Developers to write and run repeatable tests. Various parts of the application were tested covering various aspects like input testing, functionality testing and condition testing with the help of automated tests.