

2015

Online Voting System

Vote for your most preferred Automation Tool!

"Online Voting System" is a simple web-based application that will help our college to know statistically which automation tools are preferred by the students for software testing. This application is a centralized system/platform for casting and counting votes by digital means.

Ruchir Kute, Mansi Rathore, Aparna Nagarajan, Sumandeep Kaur
Northwestern Polytechnic University
7/31/2015



1. TEAM MEMBER LIST

Sr. No.	Name	Student ID
1	Ruchir Kute	12457
2	Mansi Rathore	14759
3	Aparna Nagarajan	13452
4	Sumandeep Kaur	14711

2. WORK CONTRIBUTION LIST

Date	Contributor	Description
22 nd July, 2015	Ruchir Kute	Project Development, Test Case – Registration Page, Junit Test Case, PPT, Project Report
23 rd July, 2015	Mansi Rathore	Test Case – Login Page, Equivalence Class Partitioning (ECP) & Boundary Value Analysis (BVA) Conditions, ECP Test Case – Registration Page, BVA Test Case – Registration Page, Statement & Branch Coverage, Exit Criteria, Performance Graph
25 th July, 2015	Aparna Nagarajan	Control Flow Graph, Algorithm, Project Source Code & Enhanced Code, Test Case – Vote Page, Other Test Cases
26 th July, 2015	Sumandeep Kaur	Test Case – Voting Statistics Page, Cause Effect Graph, Decision Table

Table of Contents

1. INTRODUCTION.....	5
1.1. Problem Statement.....	5
1.2. Project Overview.....	5
1.3. Intended Users for Project.....	5
1.4. Services Provided/Features to be Tested/In Scope.....	5
1.5. Out of Scope.....	6
1.6. Project Screens.....	6
2. TEST STRATEGY.....	8
2.1. Test Objective.....	8
2.2. Test Assumptions.....	9
2.3. Test Principles.....	9
2.4. Data Approach.....	10
2.5. Scope and Levels of Testing.....	10
2.5.1.Exploratory.....	10
2.5.2.User Acceptance Test (UAT).....	10
2.5.3.Functional Testing.....	10
3. DESIGN & DEVELOPMENT.....	10
3.1. Test Design.....	10
3.2. Control Flow Graph.....	11
3.3. Algorithm	12
3.4. Source Code & Enhanced Code.....	12
3.5. Database Structure.....	13
3.6. Model-View-Controller.....	13
3.7. Entry and Exit Criteria.....	14
3.8. Defect Tracking and Reporting.....	14
3.9. Test Estimation Technique.....	15
4. TESTING.....	16
4.1. Cause Effect Graph	16
4.2. Decision Table.....	17
4.3. Test Cases for webpages.....	17
4.4. Junit Test Case.....	17
4.5. Performance Metrics.....	19
4.5.1.Coverage Graph.....	19
4.5.2.Calculations for Statement Coverage, Branch Coverage & Cyclomatic Complexity.....	19
4.5.3.Conditions for ECP & BVA.....	20
4.5.4.Equivalence Class Partitioning.....	20
4.5.5.Boundary Value Analysis.....	20
4.5.6.Performance Graph.....	21

5. RISKS.....	21
6. APPENDIX.....	22
6.1. Unit Test.....	22
6.2. Integration Test.....	22
6.3. System Test.....	22
6.4. Performance Test.....	22
6.5. Security Test.....	23
6.6. Stress and Volume Test.....	23
6.7. Back Up and Recovery Test.....	23
6.8. Regression Test.....	23
6.9. User Acceptance Test.....	23

1. INTRODUCTION

1.1. Problem Statement

Every new university college thinks of providing the best education to their students and the best is defined by the most preferred and running technologies in market. The demand for every automation tool is not the same. Every student will have his own preferred software automation tool for testing. To help every college understand the need of, the most preferred testing tool by students, my project "Online Voting System" comes in to picture.

1.2. Project Overview

"Online Voting System" is simple web-based online voting systems that will help our college to know student's most preferred automation tool for testing. Online voting (also known as e-voting) is voting using electronic systems to aid casting and counting votes.

Remote e-Voting is where voting is performed within the voter's sole influence, and is not physically supervised by representatives (e.g. voting from one's personal computer, mobile phone, television via the internet (also called i-voting). Electronic voting technology can speed the counting of ballots and can provide efficiency in statistics.

In this project students have to register first by clicking on "Register me" button in order to vote for their most preferred automation tool. Once registered, they can sign in through the "Student Login page" with their respective username and password. After successful sign in they can go to "Vote Here" tab and register their vote by selecting any one tool of their choice and hitting the "Submit" button. Once done their vote will be registered. One vote per student is allowed. In "Voting Statistics" tab students can see the total number of votes received for each testing tool. This way the college will know the best automation testing tool and they can include it in their syllabus.

1.3. Intended Users for Project

Every University works on providing the right technology to their students which will help them prosper in job market. This project has been developed by keeping in mind mainly the college students and their most preferred automation testing tool. In this project not only professors but even other faculty members can see the statistics for maximum number of votes obtained to a language.

1.4. Services Provided/Features to be tested/in scope

In this project various components have been developed and integrated to work as online voting application. Some of application component details are listed below as:

- Login Page – Students can login to vote with their credentials
- Registration Page – New students have to register first to vote
- Main Page – Students can access all the tabs of voting system
- Voting Page – Students can vote for their preferred software development language here

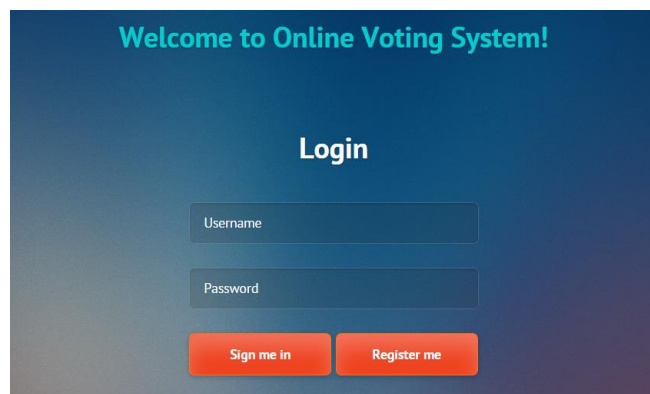
- Voting Statistic Page – Students get to see the total voting statistics here

1.5. Out of Scope

- The user acceptance team requires users who will be interacting with the system and management level staff with approval authority to sign off on the entire testing effort.
- Performance testing will also be carried out to test how the website performs under extreme load

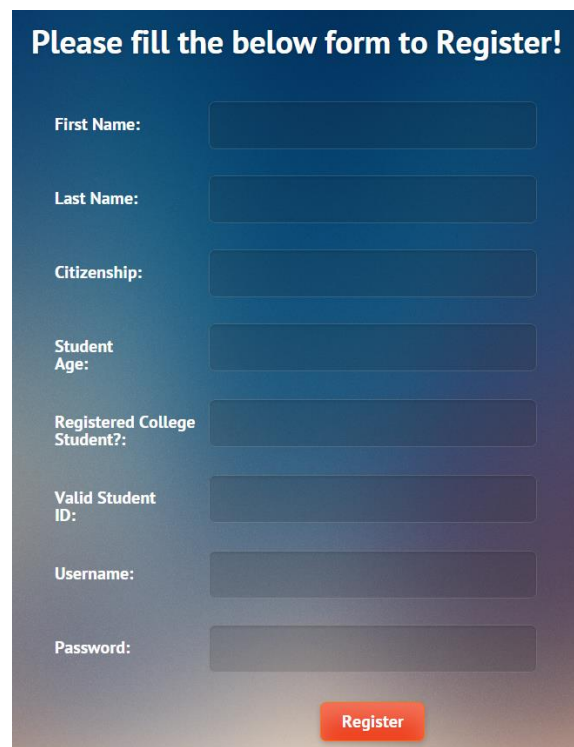
1.6. Project Screens

- Login Page



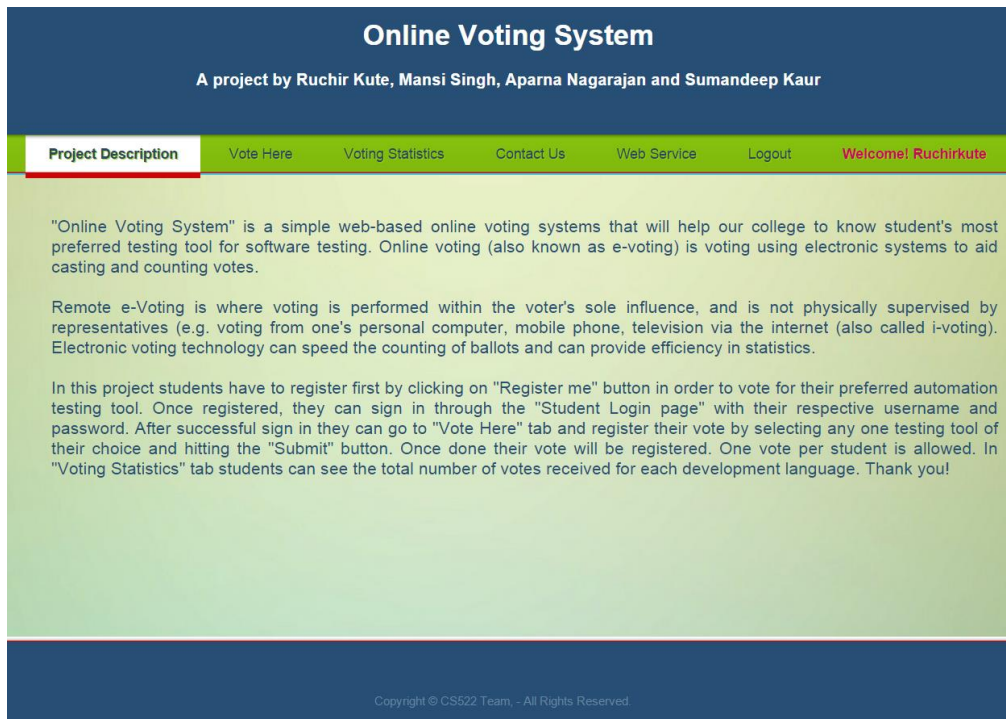
A screenshot of the login page for the Online Voting System. The page has a dark blue gradient background. At the top, it says "Welcome to Online Voting System!" in a light blue font. Below that, the word "Login" is centered in white. There are two input fields: "Username" and "Password", both with light blue borders. At the bottom, there are two orange buttons: "Sign me in" and "Register me".

- Registration Page



A screenshot of the registration page for the Online Voting System. The page has a dark blue gradient background. At the top, it says "Please fill the below form to Register!" in white. Below that, there are several input fields with labels: "First Name:", "Last Name:", "Citizenship:", "Student Age:", "Registered College Student?:", "Valid Student ID:", "Username:", and "Password:". At the bottom, there is an orange button labeled "Register".

- Main Page



The screenshot shows the main page of the Online Voting System. The header is dark blue with the title "Online Voting System" and the subtitle "A project by Ruchir Kute, Mansi Singh, Aparna Nagarajan and Sumandeep Kaur". Below the header is a green navigation bar with links: "Project Description" (highlighted), "Vote Here", "Voting Statistics", "Contact Us", "Web Service", "Logout", and "Welcome! Ruchirkute". The main content area is light green and contains three paragraphs of text describing the system. The footer is dark blue with the copyright notice "Copyright © CS522 Team, - All Rights Reserved."

Online Voting System

A project by Ruchir Kute, Mansi Singh, Aparna Nagarajan and Sumandeep Kaur

Project Description | [Vote Here](#) | [Voting Statistics](#) | [Contact Us](#) | [Web Service](#) | [Logout](#) | [Welcome! Ruchirkute](#)

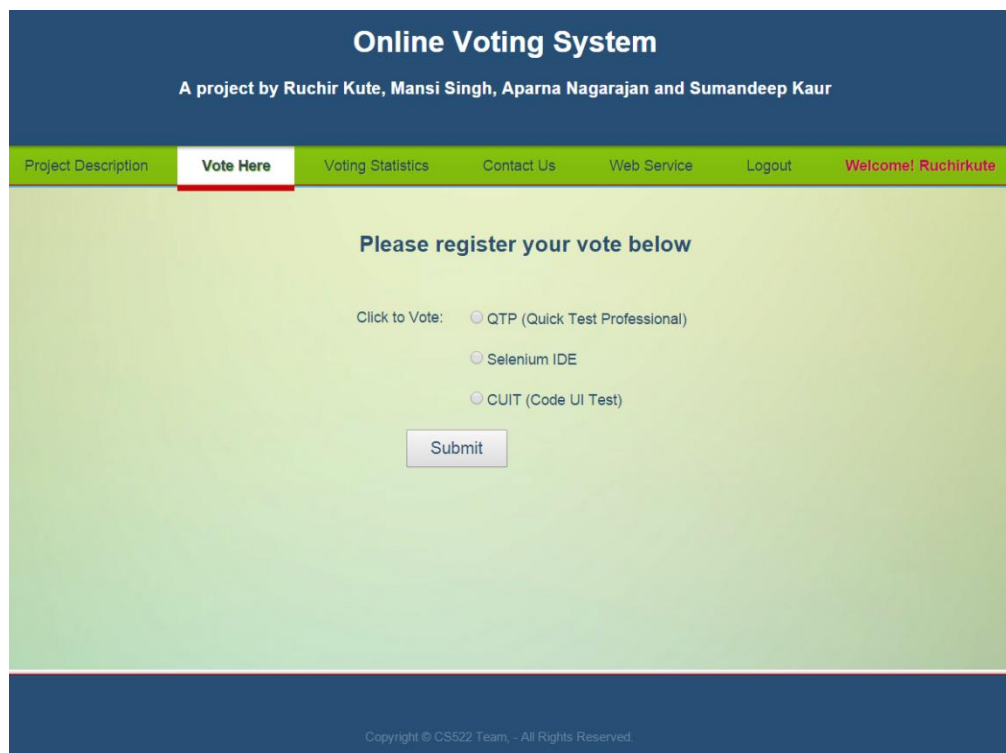
"Online Voting System" is a simple web-based online voting systems that will help our college to know student's most preferred testing tool for software testing. Online voting (also known as e-voting) is voting using electronic systems to aid casting and counting votes.

Remote e-Voting is where voting is performed within the voter's sole influence, and is not physically supervised by representatives (e.g. voting from one's personal computer, mobile phone, television via the internet (also called i-voting). Electronic voting technology can speed the counting of ballots and can provide efficiency in statistics.

In this project students have to register first by clicking on "Register me" button in order to vote for their preferred automation testing tool. Once registered, they can sign in through the "Student Login page" with their respective username and password. After successful sign in they can go to "Vote Here" tab and register their vote by selecting any one testing tool of their choice and hitting the "Submit" button. Once done their vote will be registered. One vote per student is allowed. In "Voting Statistics" tab students can see the total number of votes received for each development language. Thank you!

Copyright © CS522 Team, - All Rights Reserved.

- Vote Here Page



The screenshot shows the "Vote Here" page of the Online Voting System. The header is dark blue with the title "Online Voting System" and the subtitle "A project by Ruchir Kute, Mansi Singh, Aparna Nagarajan and Sumandeep Kaur". Below the header is a green navigation bar with links: "Project Description", "Vote Here" (highlighted), "Voting Statistics", "Contact Us", "Web Service", "Logout", and "Welcome! Ruchirkute". The main content area is light green and contains a heading "Please register your vote below" and a form with three radio buttons for "Click to Vote": "QTP (Quick Test Professional)", "Selenium IDE", and "CUIT (Code UI Test)". A "Submit" button is located below the radio buttons. The footer is dark blue with the copyright notice "Copyright © CS522 Team, - All Rights Reserved."

Online Voting System

A project by Ruchir Kute, Mansi Singh, Aparna Nagarajan and Sumandeep Kaur

[Project Description](#) | **[Vote Here](#)** | [Voting Statistics](#) | [Contact Us](#) | [Web Service](#) | [Logout](#) | [Welcome! Ruchirkute](#)

Please register your vote below

Click to Vote: ☐ QTP (Quick Test Professional)
☐ Selenium IDE
☐ CUIT (Code UI Test)

Copyright © CS522 Team, - All Rights Reserved.

- Voting Statistics Page

The screenshot shows the 'Voting Statistics' page of the 'Online Voting System'. The page has a dark blue header with the title 'Online Voting System' and a subtitle 'A project by Ruchir Kute, Mansi Singh, Aparna Nagarajan and Sumandeep Kaur'. Below the header is a green navigation bar with links: 'Project Description', 'Vote Here', 'Voting Statistics' (highlighted), 'Contact Us', 'Web Service', 'Logout', and 'Welcome! Ruchirkute'. The main content area is light green and displays the text 'Total number of Votes registered!'. Below this, there are three input fields for test tools: 'QTP (Quick Test Professional):' with a value of '1', 'Selenium IDE:' with a value of '2', and 'CUIT (Code UI Test):' with a value of '1'. The footer is dark blue and contains the text 'Copyright © CS522 Team, - All Rights Reserved.'

2. TEST STRATEGY

2.1. Test Objective

The objective of the test is to verify that the functionality of Online Voting System module works according to the user given specifications and meets the customer's expectation.

The scope of testing consists of –

1. Component level validation using unit/white box testing like
 - Statement Coverage using Junit
 - Branch Coverage using Junit
2. Functional Testing
 - BVA
 - ECP
 - Decision Table
 - Exploratory Testing
3. Regression
 - Automation using Selenium (please check)

The goal is to provide a framework that can be used by managers and testers to plan and execute the necessary tests in a timely and cost-effective manner to ship the product with quality.

2.2. Test Assumptions

Following assumptions are crucial for testing team to complete the planned activities and any deviation with following assumption will be considered as risk.

- Functional Testing would be carried out once the build is ready for testing
- Once testing begins, changes to the application and/or overall system are discouraged. If functional changes are required, the proposed changes will be discussed with the Project Manager, Project Team, and be escalated as necessary to appropriate management levels to assess the impact of the change and if/when it should be implemented. The system test team requires experienced testers to develop, perform and validate tests.
- The Test Team will be provided with access to Test environment
- The source code must be unit tested and provided within the scheduled time outlined in the Project Schedule.
- The Test Team assumes all necessary inputs required during Test design and execution will be supported by Development/Business Analysts appropriately.
- Test case design activities will be performed by Quality Assurance Group
- Test environment and preparation activities will be owned by Development Team
- Development team will provide Defect fix plans based on the Defect meetings during each cycle to plan. The same will be informed to Test team prior to start of Defect fix cycles
- Business Analyst will review and sign-off all Test cases prepared by Test Team prior to start of Test execution
- Appropriate PCs (with specified hardware/software) as well as the connectivity to appropriate servers need to be available during normal working hours. Any downtime will affect the test schedule.
- The defects will be tracked if any in a defect tracking tool. Any defect fixes planned will be shared with Test Team prior to applying the fixes on the Test environment
- Project Manager/ Business Analyst will review and sign-off all test deliverables
- The project will provide test planning, test design and test execution support
- Test team will manage the testing effort with close coordination with Project PM/ Business Analyst
- Test data & database should also be made available to the testers for use during testing.
- Project team has the knowledge and experience necessary, or has received adequate training in the system, the project and the testing processes.
- There is no environment downtime during test due to outages or defect fixes.
- The system will be treated as a black box; if the information shows correctly online and in the reports, it will be assumed that the database is working properly.

2.3. Test Principles

- Testing will be focused on meeting the business objectives, cost efficiency, and quality.
- Testing will be divided into distinct phases, each with clearly defined objectives and goals.
- There will be common, consistent procedures for all teams supporting testing activities.
- Testing will be a repeatable, quantifiable, and measurable activity.
- Testing processes will be well defined, yet flexible, with the ability to change as needed.

- Testing environment and data will emulate a production environment as much as possible.
- Testing activities will build upon previous stages to avoid redundancy or duplication of effort.
- Testing will be robust
- There will be entrance and exit criteria.

2.4. Data Approach

Here in functional testing all the pre-loaded data will be already loaded in the system in order to perform testing activities.

2.5. Scope and Levels of Testing

2.5.1. Exploratory

The purpose of this test is to make sure critical defects are removed before the next levels of testing can start.

2.5.2. Unit Testing

Testing team will author unit test code for statement and branch coverage using Junit framework.

2.5.3. Functional Testing

Functional testing will be performed to check the functions of application. The functional testing is carried out by feeding the input and validates the output from the application.

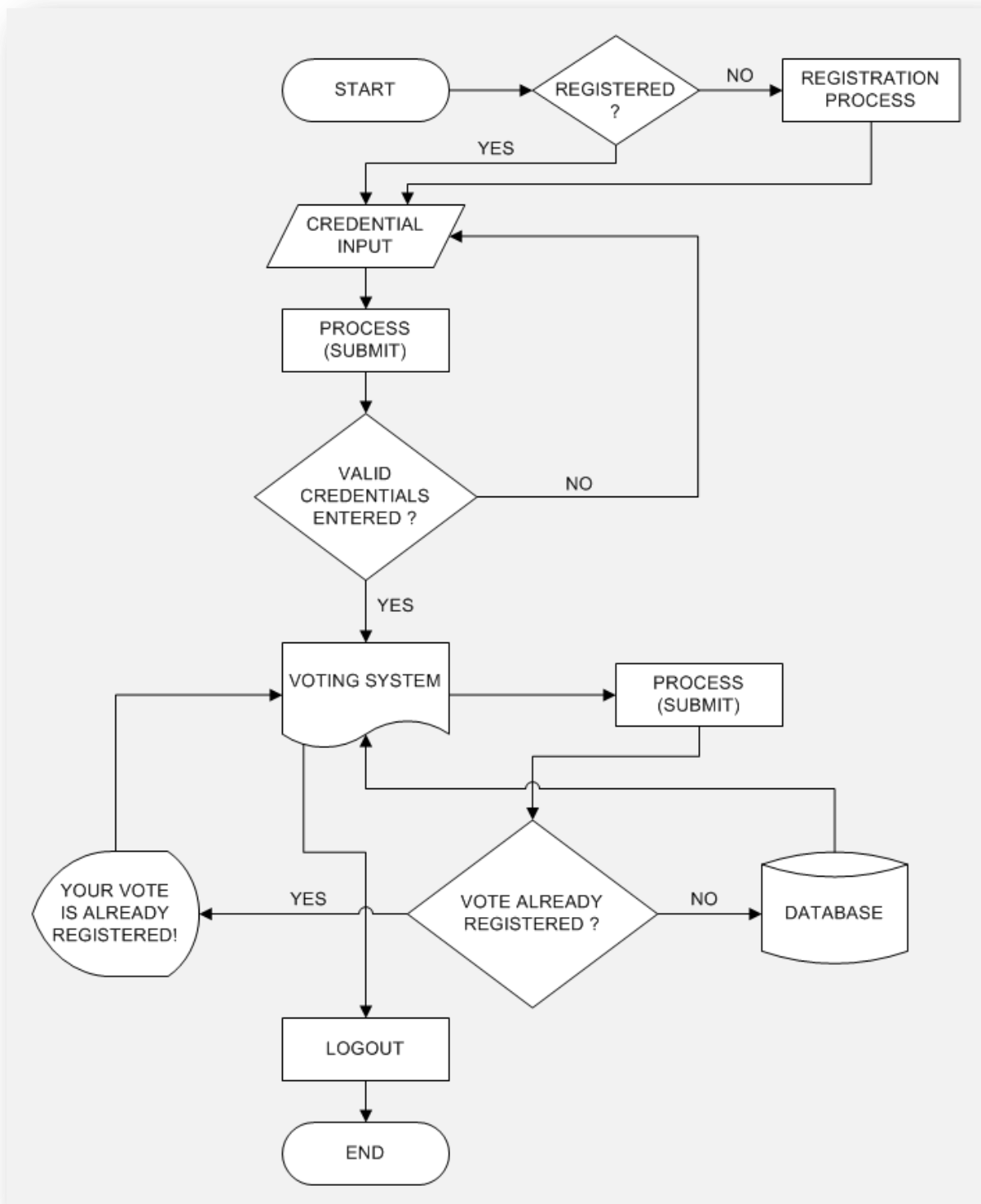
3. DESIGN & DEVELOPMENT

3.1. Test Design

The test developer should understand the requirements and accordingly prepare test cases such that all requirements are covered. The user must not face any issues with the website design in future. Also the QA team should think from the user perspective and design the website in such a way that user should be satisfied

- Each of the Test cases will undergo peer review with functional inputs from the business analyst following the review defects are captured and shared with the Test team. For final sign-off, all defects will be corrected by the testers.
- In the early stages, tester will make use of a prototype and functional specification document (SRS) to create step wise test cases.
- Depending on the type of test defects, change requests will be generated and assigned to the responsible test engineer depending on the priority of the defect.
- Sign-off for the test cases would be communicated through mails by Business Analysts
- Post sign-off, any new changes to the test cases will be directly added to the requests queue and will be reviewed bi-weekly.

3.2. Control Flow Graph



3.3. Algorithm

Step1: Read User choice;
 if(already registered) { **goto** Step2;}
 else { register user;}
 end if
Step2: Read Username & Password;
Step3: **boolean** check(Username & Password);
Step4: **if**(check) {
 Read Vote;
 boolean checkVote(Vote);
 if(checkVote) { Write: Vote already registered; }
 else { Write: Vote to DB; }
 end if
 }
 else { **goto** Step2; }
 end if
Step5: Stop;

3.4. Source Code and Enhanced Code

```
9=<%  
10 String userName = null;  
11 Cookie[] cookies = request.getCookies();  
12 if (cookies != null) {  
13     for (Cookie cookie : cookies) {  
14         if (cookie.getName().equals("username"))  
15             userName = cookie.getValue();  
16     }  
17 }  
18 if (userName == null)  
19     response.sendRedirect("login.jsp");  
20 %>  
21 </head>  
22 <body>  
23     <div id="page">  
24         <div id="header">  
25             <div id="section">  
26                 <div  
27                     style="text-align: center; color: white; margin: 0px 0 0px 324px;">  
28                     <h1 style="margin: 32px 160px 18px 0;">Online Voting System</h1>  
29                     <h3 style="margin: 0px 0 0 -160px;">  
30                         "A project by Ruchir Kute, Mansi Singh, Aparna Nagarajan and Sumandeep Kaur</h3>  
31                     </div>  
32                 </div>  
33             </div>  
34         <ul>  
35             <li class="current"><a href="index.jsp">Project Description</a></li>  
36             <li><a href="vote.jsp">Vote Here</a></li>  
37             <li><a href="vote-stats.jsp">Voting Statistics</a></li>  
38             <li><a href="contact.jsp">Contact us</a></li>  
39             <li><a href="JasonServlet">Web Service</a></li>  
40             <li><a href="LogoutServlet">Logout</a></li>  
41             <li><a class="welcome"><b>Welcome! <%=userName%></b></a></li>  
42
```

This code can be enhanced to achieve better performance by using the "Session" variable in this program. We need to make changes in JSP & Servlet files.

3.5. Database Structure

For this application, the database is created by using the MySQL Workbench. The following below information highlights my database entities –

1. Schema

- votingdatabase

2. Tables & Columns

- studentdata

- id (INT(11) Primary Key, Not Null, Auto Increment)
- fname (VARCHAR(45) Not Null)
- lname (VARCHAR(45) Not Null)
- citizenship (VARCHAR(45) Not Null)
- studentage (INT(3))
- regcollegestudent (VARCHAR(3) Not Null)
- studentid (INT(5))
- username (VARCHAR(45) Not Null, Unique)
- password (VARCHAR(45) Not Null)

- devlang

- id (INT(11) Primary Key, Not Null, Auto Increment)
- username (VARCHAR(45) Unique) votes (VARCHAR(45))
- votes (VARCHAR(45))

3.6. Model-View-Controller

In this project the Model-View-Controller (MVC) implementation has been used in an efficient way. Below is the hierarchy for the MVC implementation –

1. Model (edu.npu.votingsystem.domain)

- Register.java
- Vote.java

2. View (VotingSystem/WebContent folder)

- login.jsp
- register.jsp
- index.jsp
- vote.jsp
- vote-stats.jsp

3. Controller (edu.npu.votingsystem.servlets)

- LoginServlet.java
- LogoutServlet.java
- RegistrationServlet.java
- VotingServlet.java

4. Database (edu.npu.votingsystem.database)

➤ VotingBin.java

5. JUnit (edu.npu.votingsystem.testlogin)

➤ TestLogin.java

3.7. Entry and Exit Criteria

The test plan which is developed need to be tested until and unless the final bug count is at a minimal rate which QA is promised to the customer. Then testing team can stop the testing and give the final output to the customer then we can exit the criteria.

Entry Criteria –

- Build is deployed on test environment with 70% statement coverage.
- Smoke checklist is being executed by development team
- All features under the release scope is being covered with unit test.

Some of the Exit criteria's are as follows –

- 100% Test Scripts executed
- No open high priority, severity and critical defects
- Pre agreed high percentage of Medium severity defects resolved
- All results should be documented
- 90% functional coverage
- 80% Code coverage
- Unfulfilled defects captured in change request document and to be reviewed ahead of future release
- All tests and execution results will be checked into revision control repository
- Test metric report generated using JIRA/Bugzilla
- Test Coverage = Number of Test Cases Passed/Total Number of Test Cases * 100
- Test Coverage (First Phase) = $22/45 * 100 = 48\%$
- Test Coverage (Final Phase) = $41/45 * 100 = 91\%$

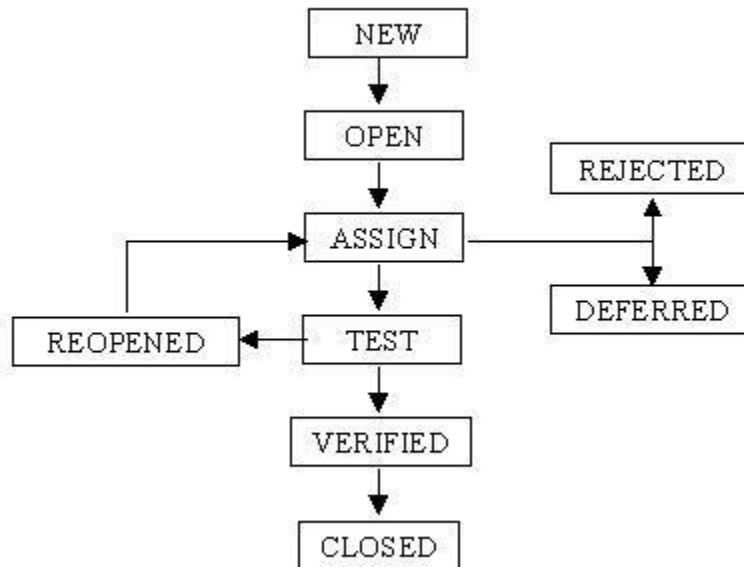
3.8. Defect Tracking and Reporting

All the bugs captured while testing should be organized by using a Bug Tracking Tool for example Bugzilla or Jira etc. They should be reported daily or weekly to the developer team in order to fix it on time.

The bugs can be categorized as per their level of severity based on impact on functionality of application. All the defects would come along with a snapshot in JPEG format for easy understanding.

A bug triage meeting will be conducted between product owner, development and testing team bi-weekly to fill any communication gap and prioritization of defects.

Following defect life cycle will be followed for issues found during test case execution:

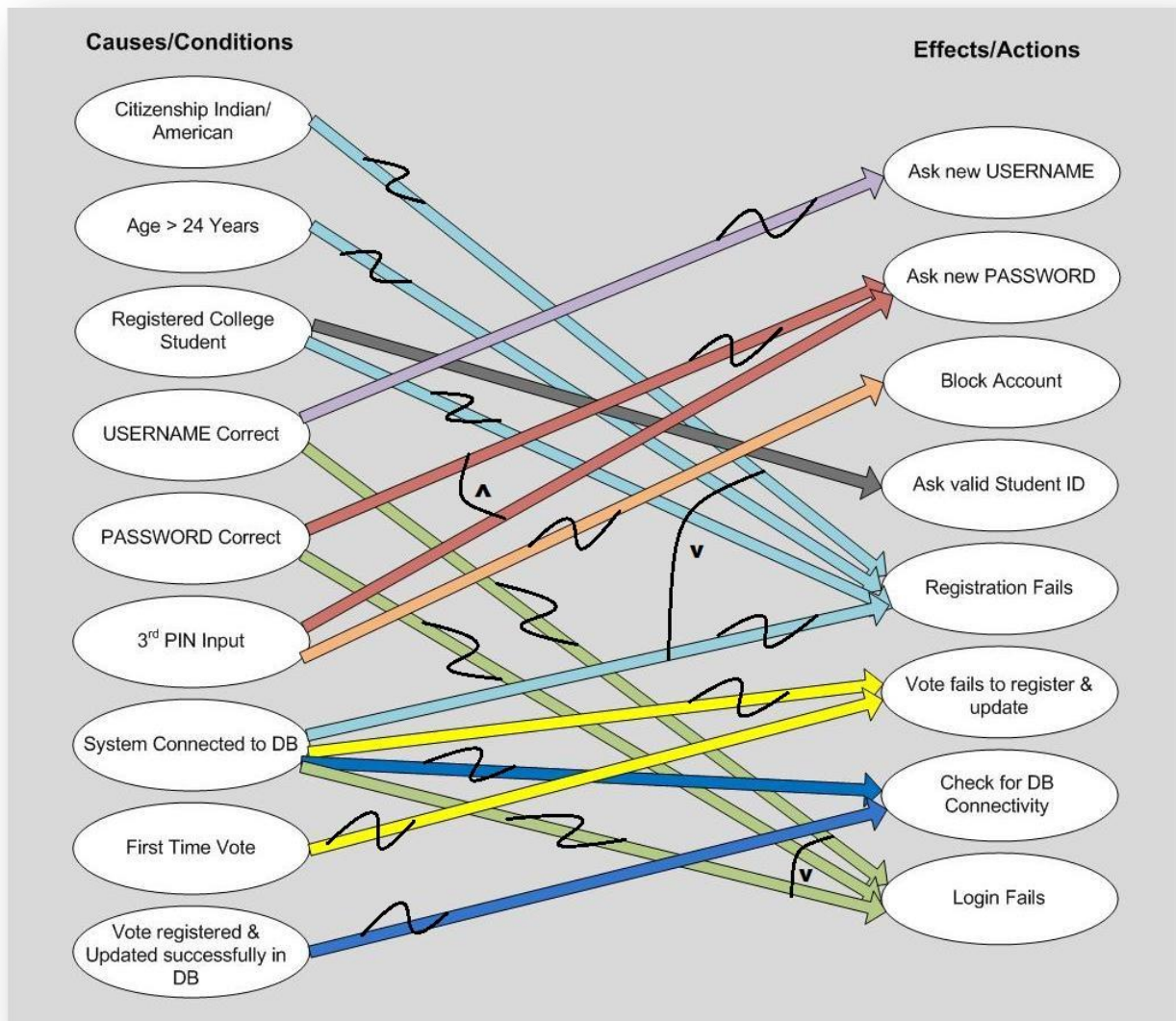


3.9. Test Estimation Technique

Team will consider the gut based test estimation technique based on experience and expertise of team, all estimations will be provided in tasks during planning meeting.

4. TEST ANALYSIS AND EXECUTION

4.1. Cause Effect Graph



4.2. Decision Table

Condition/Cause	TC1	TC2	TC3	TC4	TC5	TC6	TC7	TC8	TC9	TC10	TC11	TC12	TC13	TC14	TC15	TC16
Citizenship is Indian or American	N	Y	Y	Y	Y											Y
Age is above 24 years	Y	N	Y	Y	Y											Y
Registered college Student	Y	Y	N	Y	Y											Y
USERNAME is correct						N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PASSWORD is correct						Y	N	N	Y	Y	Y	Y	Y	Y	Y	Y
3 incorrect PASSWORD attempts								Y								
System connected to the database	Y	Y	Y	N	Y	Y	Y	Y	N	Y	Y	N	Y	Y	Y	Y
A registered college student votes for the first time												N	Y	Y	Y	Y
Vote is registered and updated successfully in database											N	N	Y	N	Y	Y
Effect/Action																
Ask new USERNAME	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N
Ask new PASSWORD	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N
Block Account	N	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N
Ask valid Student ID	Y	Y	N	Y	Y	N	N	N	N	N	N	N	N	N	N	Y
Registration Fails	Y	Y	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N
Login Fails	N	N	N	N	N	Y	Y	Y	Y	N	N	Y	N	N	N	N
Vote Fails to Register and Update	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N
Check for Database Connectivity	N	N	N	Y	N	N	N	N	Y	N	N	Y	N	Y	N	N

4.3. Test Cases

You can see or download the Test Cases by clicking the below link.

<https://drive.google.com/file/d/0B1dZ4mUwq69oaXhNQU9YWIFCWVU/view?usp=sharing>

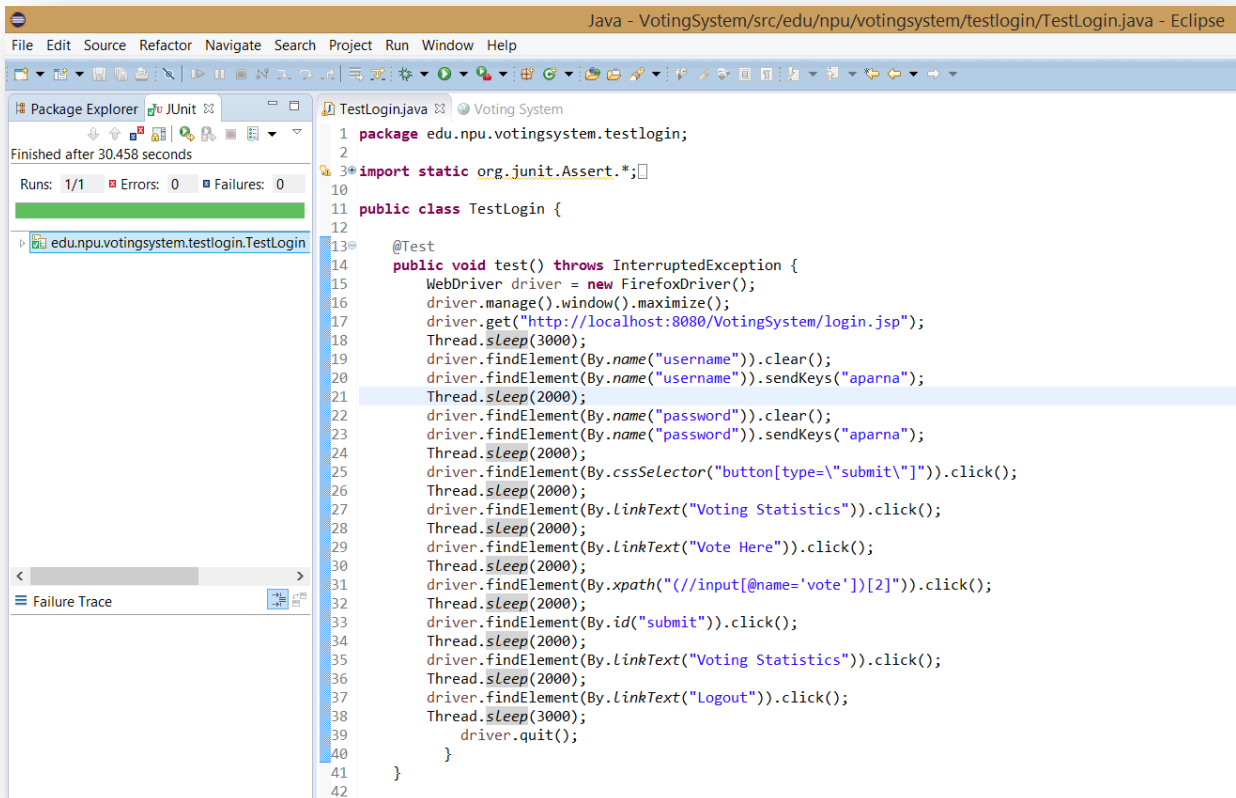
4.4. Junit Test Case

```

1 package edu.npu.votingsystem.testlogin;
2
3 import static org.junit.Assert.*;
4
5
6
7
8
9
10
11 public class TestLogin {
12
13     WebDriver driver;
14     private StringBuffer verificationErrors = new StringBuffer();
15
16     @Test
17     public void test() throws InterruptedException {
18         WebDriver driver = new FirefoxDriver();
19         driver.manage().window().maximize();
20         driver.get("http://localhost:8080/VotingSystem/login.jsp");
21         Thread.sleep(3000);
22         driver.findElement(By.name("username")).clear();
23         driver.findElement(By.name("username")).sendKeys("ruchirkute");
24         Thread.sleep(2000);
25         driver.findElement(By.name("password")).clear();
26         driver.findElement(By.name("password")).sendKeys("kute");
27         Thread.sleep(2000);
28         driver.findElement(By.cssSelector("button[type='submit']")).click();
29         Thread.sleep(2000);
30         driver.findElement(By.LinkText("Vote Here")).click();
31         Thread.sleep(2000);
32         driver.findElement(By.LinkText("Voting Statistics")).click();
33         Thread.sleep(2000);
34         driver.findElement(By.LinkText("Logout")).click();
35         Thread.sleep(3000);
36         driver.quit();
37     }
38 }
39

```

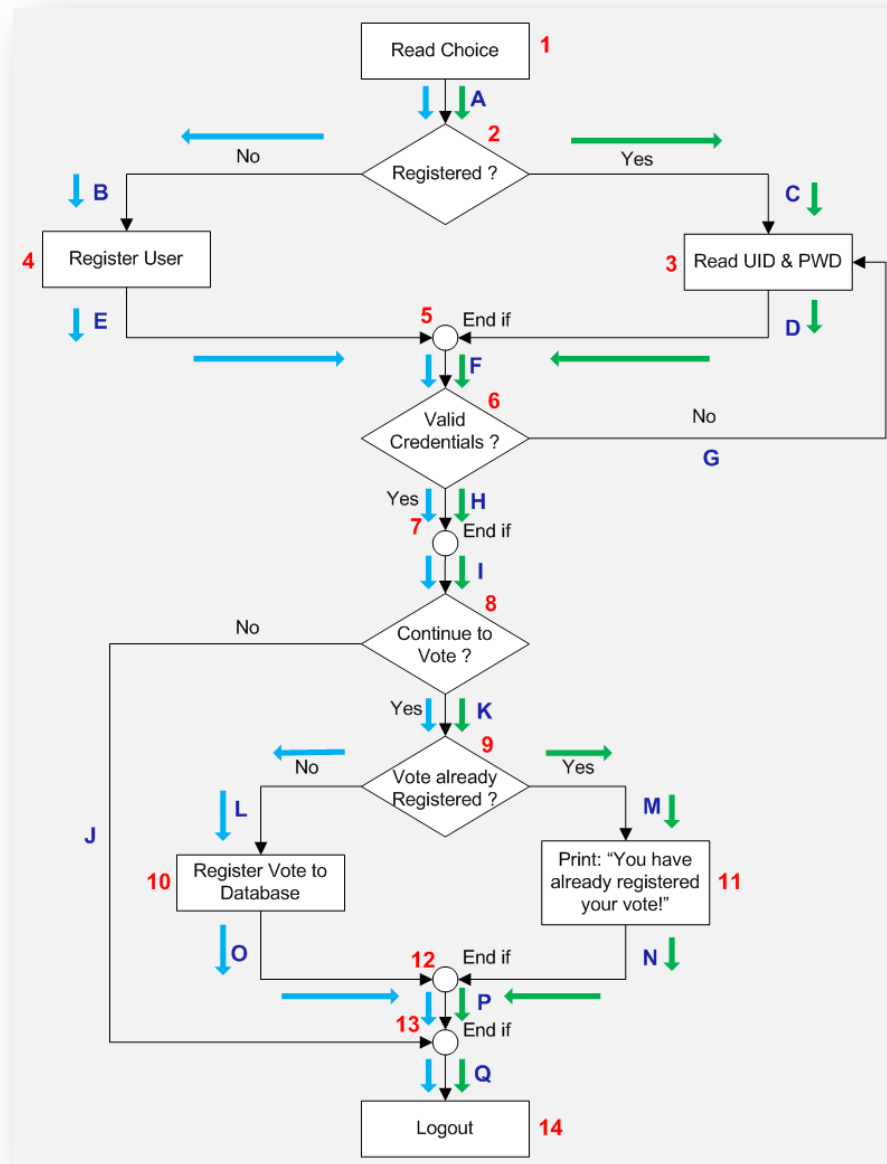
The screenshot shows the Eclipse IDE with the 'TestLogin.java' file open. The code is a JUnit test case that automates the login process. It uses Selenium WebDriver to interact with the Voting System's login page. The test includes steps for maximizing the browser window, navigating to the login page, entering the username 'ruchirkute' and password 'kute', clicking the submit button, and then clicking on 'Vote Here', 'Voting Statistics', and 'Logout' links. The test is annotated with @Test and throws InterruptedException. The Package Explorer on the left shows the project structure, and the JUnit icon indicates that the test is runnable.



```
Java - VotingSystem/src/edu/npu/votingsystem/testlogin/TestLogin.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
TestLogin.java Voting System
1 package edu.npu.votingsystem.testlogin;
2
3 import static org.junit.Assert.*;
4
10
11 public class TestLogin {
12
13     @Test
14     public void test() throws InterruptedException {
15         WebDriver driver = new FirefoxDriver();
16         driver.manage().window().maximize();
17         driver.get("http://localhost:8080/VotingSystem/login.jsp");
18         Thread.sleep(3000);
19         driver.findElement(By.name("username")).clear();
20         driver.findElement(By.name("username")).sendKeys("aparna");
21         Thread.sleep(2000);
22         driver.findElement(By.name("password")).clear();
23         driver.findElement(By.name("password")).sendKeys("aparna");
24         Thread.sleep(2000);
25         driver.findElement(By.cssSelector("button[type='submit']")).click();
26         Thread.sleep(2000);
27         driver.findElement(By.LinkText("Voting Statistics")).click();
28         Thread.sleep(2000);
29         driver.findElement(By.LinkText("Vote Here")).click();
30         Thread.sleep(2000);
31         driver.findElement(By.xpath("//input[@name='vote']"))[2]).click();
32         Thread.sleep(2000);
33         driver.findElement(By.id("submit")).click();
34         Thread.sleep(2000);
35         driver.findElement(By.LinkText("Voting Statistics")).click();
36         Thread.sleep(2000);
37         driver.findElement(By.LinkText("Logout")).click();
38         Thread.sleep(3000);
39         driver.quit();
40     }
41 }
42
Package Explorer JUnit
Finished after 30.458 seconds
Runs: 1/1 Errors: 0 Failures: 0
edu.npu.votingsystem.testlogin.TestLogin
Failure Trace
```

4.5. Performance Metrics

4.5.1. Coverage Graph



4.5.2. Calculations for Statement Coverage, Branch Coverage & Cyclomatic Complexity

Statement Coverage (SC): The shortest number of paths (P) which covers all the nodes.

P1 = 1A-2C-3D-5F-6H-7I-8K-9M-11N-12P-13Q-14

P2 = 1A-2B-4E-5F-6H-7I-8K-9L-10O-12P-13Q-14

SC = 2

Branch Coverage (BC): The minimum number of paths (P) which covers all the edges.

P1 = 1A-2B-4E-5F-6H-7I-8K-9L-10O-12P-13Q-14

P2 = 1A-2C-3D-5F-6G

P3 = 1A-2C-3D-5F-6H-7I-8J-13Q-14

P4 = 1A-2B-4E-5F-6H-7I-8K-9M-11N-12P-13Q-14

BC = 4

Cyclomatic Complexity:

$V(G) = E \text{ (Edges)} - N \text{ (Nodes)} + 2$

$V(G) = 17 - 14 + 2 = 5$

4.5.3. Conditions for Equivalence Class Partitioning (ECP) & Boundary Value Analysis (BVA)

First Name					Username				
Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries	Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
	2-20 chars	< 2 chars	2, 3 chars	1 char		2-20 chars	< 2 chars	2, 3 chars	1 char
	Valid chars	> 20 chars	19, 20 chars	21 chars		Valid chars	> 20 chars	19, 20 chars	21 chars
		Invalid chars		0 chars			Invalid char		0 char
First Name	Valid Chars	A-Z, a-z, Space, Roman Numerals			Username	Valid Chars	A-Z, a-z, @, ., 0-9		
	Invalid Chars	Numbers & Special Characters				Invalid Chars	All special characters except @, .		
Last Name					Password				
Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries	Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
	2-20 chars	< 2 chars	2, 3 chars	1 char		8-20 chars	< 8 chars	8, 9 char	7 char
	Valid chars	> 20 chars	19, 20 chars	21 chars		Valid chars	> 20 chars	19, 20 char	21 chars
		Invalid chars		0 chars			Invalid char		0 char
Last Name	Valid Chars	A-Z, a-z, Roman Numerals			Password	Valid Chars	A-Z, a-z, 0-9, All special characters		
	Invalid Chars	Numbers & Special Characters				Invalid Chars	Space		
Age					Student ID				
Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries	Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
	24-60	< 24	24, 25	23		5 digits	<4 digits	5 digits eg. 12345	4 digits eg. 1234
	First char non-zero	> 60	59, 60	61			>6 digits		6 digits eg. 123456
		Null				First char Non - Zero			
		Non-numeric	99 digits	100			First char zero		012345
Age	Valid	Non - Zero			Student ID	Valid	Non - Zero, 5 digits		
	Invalid	Zero				Invalid	Zero, < 4 digits, > 6 digits		

4.5.4. Equivalence Class Partitioning

You can see or download the Test Cases by clicking the below link.

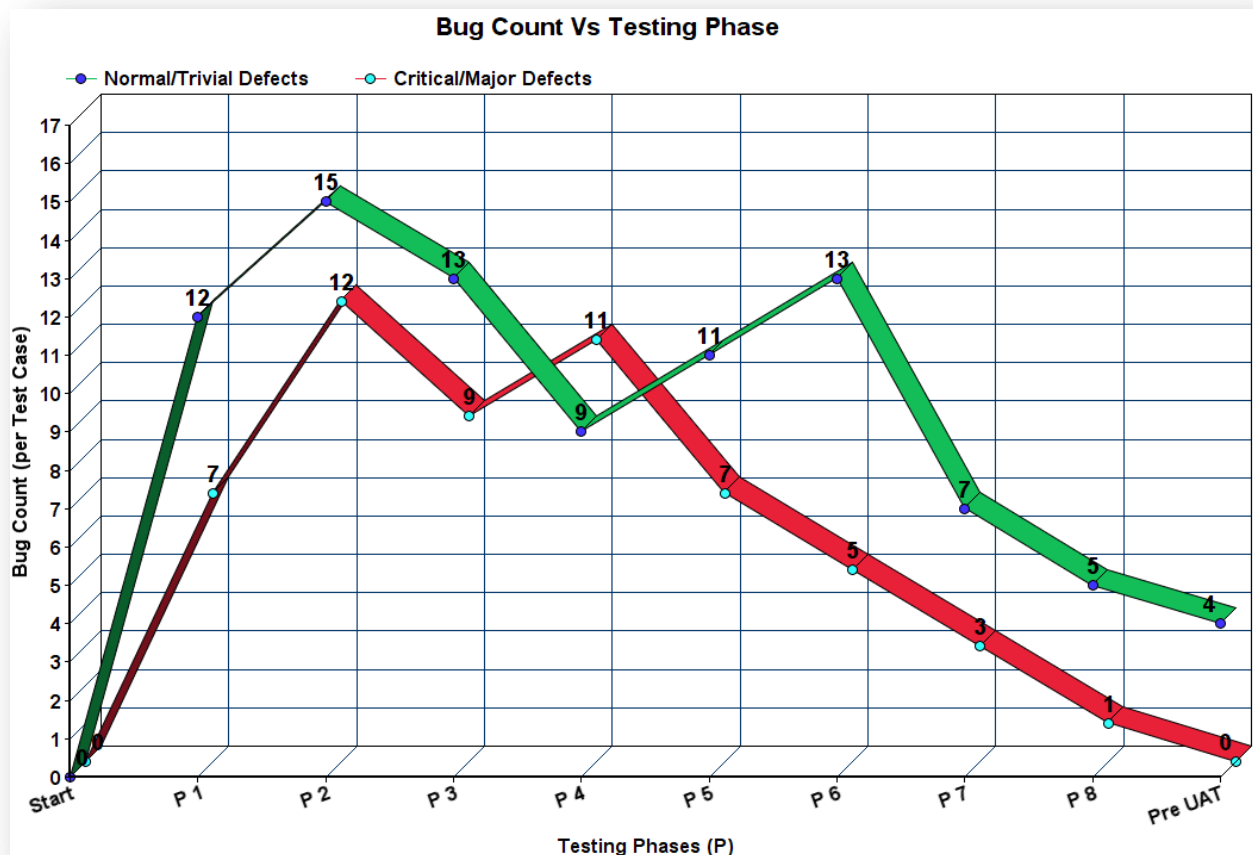
<https://drive.google.com/file/d/0B1dZ4mUwq69oaXhNQ9YWIFCWVU/view?usp=sharing>

4.5.5. Boundary Value Analysis

You can see or download the Test Cases by clicking the below link.

<https://drive.google.com/file/d/0B1dZ4mUwq69oaXhNQ9YWIFCWVU/view?usp=sharing>

4.5.6. Performance Graph



5. RISKS

- **Schedule:** The schedule for each project phase could affect testing. A slip in the schedule in one of the other phases could result in a subsequent slip in the test phase. Close project management is crucial to meeting the forecasted completion date. Have prewritten test cases that cover all scenarios. Therefore, sufficient time must be allocated to write and execute these test cases thoroughly.
- **Technical:** Network connectivity, backups, and ability to recover data will be crucial for the test environment. In addition, if parallel testing is conducted, the legacy system must be operational and available.
- **Management:** Management support is required so when the project falls behind, the test schedule does not get squeezed to make up for the delay. Management can reduce the risk of delays by supporting the test team throughout the testing phase and assigning people to this project with the required time set aside as well as the appropriate skill sets to run the tests and check results.
- **Personnel:** It is very important to have subject matter experts involved in testing. The test cases should be reviewed with these knowledgeable individuals prior to starting testing. It is

also advisable when performing data entry tests that users who are somewhat familiar, but not overly familiar with the software be used.

- **Requirements:** The test plan and test schedule are based on the currently known requirements outlined in the User Requirements documentation. If the documentation of requirements is not complete, we run the risk of not testing all requirements.

6. APPENDIX

5.1. Unit Testing

The developers will unit test their own sections of the application. The developers will create their own test data and test scenarios unless these things are otherwise provided for them.

5.2. Integration Testing

Ensure that parts of the application that need to communicate or have some relationship to each other work properly together. This testing will be performed as a coordinated effort among the developers or will be conducted by the testing lead. System testing should not begin until integration testing is complete. List the functions that should be integration tested below.

- Integrated function 1
- Integrated function 2

5.3. System Testing

The System test will focus on the behavior of the application and system as a whole. Scenarios will be executed

- Within the application
- Through file transfers
- Reports generation
- Other outputs or data checks
- To verify that the system successfully accomplishes the functions included in the scope of the project
- Test cases and scripts/scenarios should be mapped to business requirements outlined in the User Requirements document. This will ensure that all requirements have been addressed and tested.

5.4. Performance Test

Performance tests will be conducted to ensure that

- The data entry process can be completed in a timely manner for each step. The response time required should be determined with the user community.
- File transfers are occurring at the right times and are completing with little to no interruption for the user community
- Batch jobs are completing in a reasonable amount of time and fit in appropriately with other batch schedules occurring for this and other systems
- Consider testing on a server with multiple databases to see how performance will be in a production environment where several databases are on the same server.
- Test with large enough volume to simulate higher data volume to see how the database response time will be.

5.5. Security Test

The security testing will be conducted as a part of system testing.

5.6. Stress and Volume Test

- Simulate the stress on the application and server that is expected during certain active times such as during the workday, workweek, month, etc.
- Simulate the volume of data that is expected in a production environment.
- Test scenarios for various numbers of users.
- Include test cases that cover special situations like holidays, major events, etc. that could affect the timing and/or volume of entries.

5.7. Backup and Recovery Test

During testing, we may encounter problems that require us to restore data in our test environment.

- Perform regular backups
- Test the recovery process to ensure that we can restore data back to specific points in time.
- This test will also be useful in ensuring that the production system can be restored to a point in time from backup when we go live. It is vitally important that all data is recovered after a system failure & no corruption of the data occurred.

5.8. Regression Test

Regression testing is done in order to ascertain whether fixes to defects have caused errors elsewhere in the application/process.

- If possible, create a standard set of tests that can be run in an automated way. For instance, if there is a batch job to kick off certain processing from beginning to end, this would be a good regression test after major fixes or a new release has been implemented.

5.9. User Acceptance Test

After system testing is complete, user acceptance testing will be done. Testers should -

- Be knowledgeable business users who are familiar with the scope of the project
- Create test cases to cover appropriate scenarios of system use. Oftentimes, IT resources may need to assist the user acceptance test team in how to develop the test cases
- Confirm that the system is developed according to the specified user requirements. Test cases and scripts/scenarios should be mapped to business requirements outlined in the User Requirements document
- Document “bugs” that are found
- Retest after bugs have been fixed
- Confirm that the system is ready for operational use.

Black box testing should also be conducted as part of user acceptance testing. This testing by individuals who do not necessarily know the business, can ensure that the application doesn’t fail in areas that business users may not think to try.