

“ONLINE VOTING SYSTEM ”

**A Project Report Submitted in Partial Fulfillment of the Requirements for the Degree
of
Bachelor of Technology
in
Computer Science & Engineering
By**

	Student Name	Roll Number	Branch and Section
1	ABHISHEK SHARMA	2301650100010	B-TECH(CSE) ‘A’
2	ANAND KESHRI	2301650100030	B-TECH(CSE) ‘A’
3	ABHISHEK GUPTA	2301650100008	B-TECH(CSE) ‘A’
4	ANKIT SHARMA	2301650100037	B-TECH(CSE) ‘A’
5			

**Under the Supervision of
Mr. ANAND SHARMA (Assistant Professor)**



**Kanpur Institute of Technology, Kanpur
Dr. A.P.J Abdul Kalam Technical University, Lucknow**

DECLARATION

We hereby declare that this submission of our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma or the university or other institute of higher learning, except where due acknowledgement has been made in the text.

Signature:

Name: ABHISHEK SHARMA

Roll no: 2301650100010

Date:

Signature:

Name: ABHISHEK GUPTA

Roll no 2301650100008

Date:

Signature:

Name: EEEEEEE Roll no:

555555555

Date:

Signature:

Name ANAND KESHRI Roll no:

2301650100030

Date:

Signature:

Name: ANKIT SHARMA

Roll no: 2301650100037

Date:

CERTIFICATE

*This is to certify that the Project Report entitled “online voting system which is submitted by **Abhishek Sharma (2301650100010)**, **Anand Keshri (2301650100030)**, **Abhishek Gupta (2301650100008)**, **Ankit Sharma (2301650100037)**, **EEEEEE (5555555)** of the 3rd semester, in the year 2024-2025.*

*In partial fulfillment of the requirement for the award of degree of B.Tech in department of **Computer Science & Engineering** of **Dr. APJ Abdul Kalam Technical University** is a record of the candidate own work carried out by him under my supervision. The matter embodied in this is original and is not been submitted for the award of any degree.*

Signature
Mr. Rahul Singh)
(Head of Department)
CSE Department
KIT Kanpur

Signature
(Mr. XYZ)
(Assistant Professor)
CSE Department
KIT Kanpur

Date:

ACKNOWLEDGEMENT

*It gives us a great sense of pleasure to present the report of the B.Tech Project undertaken during **B.Tech Final/Third/Second Year**. We owe special debt of gratitude to our guide **Mr.***

***Anand sir** of Department of Computer Science & Engineering, Kanpur Institute of Technology, Kanpur (U.P) for his constant support and guidance throughout course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have been light of the day.*

*We also take the opportunity to acknowledge the contribution of **Mr. Rahul Singh , Headof Department of Computer Science & Engineering, Kanpur Institute of Technology, Kanpur (U.P.)** for his full support and assistance during the development of the project.*

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their assistance and cooperation during the development of my project. Last but not the least, we acknowledge our friends for their contributions in the completion of the project.

Signature:

Name: ABHISHEK SHARMA

Roll no: 2301650100010

Date:

Signature:

Name: ANANDKESHRI

Roll no: 2301650100030

Date:

Signature:

Name: ABHISHEK GUPTA

Roll no2301650100008

Date:

Signature:

Name: ANKIT SHARMA

Roll no: 2301650100030

Date:

TABLE OF CONTENTS

1. Declaration	ii
2. Certificate	iii
3. Acknowledgement	iv
4. Abstract	v
5. List of Figures	viii
INTRODUCTION	1
1.1 General Introduction	1
1.2 Problem Statement	1
1.2.1 Problem Detection	1
1.2.2 Problem Solving	2
1.3 Proposed System Overview	2
1.4 Methodology	3
1.4.1 Principles of Agile	4
1.4.2 Flow Chart	5
1.5 What is Steganography?	5
1.5.1 Steganography techniques	6
1.5.2 Advantages over cryptography	6
1.5.3 Types of Steganography	7
1.5.3.1 Image Steganography	7
1.5.3.2 Audio Steganography	7
1.5.3.3 Text Steganography	8
1.5.3.4 Video Steganography	8
1.6 Scope of the Project	8
2. Feasibility Report	9
2.1 Feasibility Study	9
2.1.1 Technical Feasibility	9
2.1.2 Economic Feasibility	9
2.1.3 Operational Feasibility	9
2.1.4 Legal Feasibility	10
2.2 System Design	10
2.3 Description	10
3. Requirement Elicitation	12
3.1 Requirement Analysis	12
3.1.1 Requirements Gathering	13
3.1.2 Analyzing Requirements	13
3.2 Classification of Requirement	13
3.2.1 Functional Requirements	13
3.2.2 Non-Functional Requirements	14
4. Designing	16
4.1 Modular Architecture Design	16
4.2 Detailed Design	16
4.3 ER Diagram	16
4.3.1 The components and features	17
4.3.1.1 Entity	
4.3.1.2 Relationship	
4.3.1.3 Attribute	

4.4 Data Flow Diagram

4.4.1 Level 0 DFD

4.4.2 Level 1 DFD

4.4.3 Level 2 DFD

4.5 Use Case Diagram

5. Coding & Implementation

5.1 Implementation

5.2 Hardware

5.3 IDE

5.3.1 jQuery Code – Encrypting and Decrypting Messages

5.3.2: HTML Code – Front end

5.3.3: Python Code – Backend of Messaging Service

5.4 Machine learning Model

5.4.1 Working

5.4.2 Algorithm used

5.4.3 Dataset used

5.4.4 Uploading data set

5.4.5 Training the Model

5.4.6 Output of the model

5.4.7 Steganography

5.5 Snapshots

6. Testing

6.1 Description of Agile

6.2 Diagram of Agile Model

6.3 Types of Testing

6.3.1 Black Box Testing

6.3.2 White Box Testing

7. Conclusion

7.1 Future Scope

7.2 Uses & Targets

7.3 Applications

7.4 Result & Analysis

7.5 Limitations

References

ABSTRACT

The advent of digital technology has revolutionized various sectors, including the electoral process. An online voting system (OVS) is a digital platform that enables voters to cast their ballots over the internet, enhancing accessibility, convenience, and efficiency in elections. This abstract explores the design, implementation, and implications of an OVS, focusing on its potential to increase voter participation, streamline the voting process, and reduce operational costs associated with traditional voting methods.

Key features of an effective OVS include robust security measures to protect against fraud and hacking, user-friendly interfaces to accommodate diverse voter demographics, and comprehensive verification systems to ensure the integrity of the voting process. The paper also addresses challenges such as digital divide issues, cybersecurity threats, and the need for legal frameworks to govern online voting.

Through case studies and empirical data, this research highlights successful implementations of OVS in various jurisdictions, demonstrating improved voter turnout and satisfaction. Ultimately, the findings suggest that while online voting systems offer significant advantages, careful consideration of security, accessibility, and regulatory compliance is essential for their successful adoption in democratic processes.

User friendly interface —

- *online voting system (OVS) is crucial to ensure that all voters, regardless of their technical proficiency, can easily navigate the platform and cast their votes. Here are key principles and features to consider when designing a user-friendly interface for an OVS:*
- **Key Principles for a User-Friendly Interface:**
 - **Simplicity:**
 - *The interface should be straightforward and uncluttered, allowing users to focus on the voting process without distractions.*
 - **1.Intuitive Navigation:**
 - *2. Clear and logical navigation paths should guide users through the voting process, with easily identifiable buttons and menus.*
 - **Accessibility:**
 - *The design must accommodate users with disabilities, adhering to accessibility standards (e.g., WCAG) to ensure that everyone can participate in the voting process.*
 - **Responsive Design:** *The interface should be optimized for various devices (desktops, tablets, smartphones) to allow voters to access the system from their preferred platform.*
 - **Visual Clarity:** *Use of high-contrast colors, readable fonts, and clear icons can enhance visibility and comprehension, making it easier for users to understand their options.*
- **Essential Features of a User-Friendly OVS Interface:**
 - **User Onboarding:** *A guided tutorial or walkthrough for first-time users can help them understand how to navigate the system and cast their votes.*
 - **Clear Instructions:** *Provide concise and clear instructions at each step of the voting process, including how to select candidates, review choices, and submit votes.*
 - **Progress Indicators:** *Display a progress bar or step indicators to inform users of their current position in the voting process, helping them understand how many steps remain.*
 - **Confirmation Screens:** *After users make their selections, a confirmation screen should summarize their choices before final submission, allowing them to review and make changes if necessary.*
 - **Help and Support Options:** *Easily accessible help resources, such as FAQs, live chat support, or helpline numbers, should be available for users who encounter difficulties.*

- **Multi-Language Support:** *Offering the interface in multiple languages can accommodate a diverse electorate, ensuring that language barriers do not hinder participation.*
- **Feedback Mechanism:** *Incorporating a feedback option allows users to report issues or suggest improvements, fostering a sense of community and ongoing enhancement of the system.*
- **Secure Authentication:** *Implement a straightforward yet secure authentication process (e.g., two-factor authentication) to verify voter identity without complicating the user experience.*

- **Auto fill functionality —**

- *online voting systems (OVS) that enhances user experience by simplifying data entry and reducing the likelihood of errors. This feature can streamline the voting process, making it more efficient and user-friendly. Below are key aspects and considerations for implementing auto-fill functionality in an OVS:*

- **Key Aspects of Auto-Fill Functionality:**

- **User Profile Management:**

- **Account Creation:**

-

- *When users create an account, they can enter personal information such as name, address, and voter ID. This information can be securely stored and retrieved for future voting sessions.*

- **Profile Updates:**

- *Users should have the option to update their profiles easily, ensuring*

Search Functionality:

- *Implementing a search feature that auto-suggests candidates or measures as users type can make it easier for them to find their selections quickly.*

2. Security and Privacy:

- **Data Encryption:** *Ensure that all personal information used for auto-fill is encrypted and stored securely to protect voter privacy.*
- **User Consent:**
- *Obtain explicit consent from users for storing and utilizing their information for auto-fill purposes, ensuring compliance with data protection regulations.*

3. *User Control:*

- *Opt-in/Opt-out Options: Allow users to enable or disable auto-fill features according to their preferences, giving them control over their data.*
- *Manual Override:*
- *Users should always have the option to manually enter or edit their information, ensuring accuracy and allowing corrections if needed.*

Benefits of Auto-Fill Functionality:

- *Increased Efficiency:*
- *Reduces the time required to complete the voting process, making it more convenient for users.*
-
- *Error Reduction:*
- *Minimizes the chances of typographical errors, ensuring that the information submitted is accurate.*
-
- *Enhanced User Experience:*
- *A smoother and faster voting process can lead to higher voter satisfaction and increased participation.*
-
- *Accessibility:*
- *Simplifies the voting process for individuals with disabilities or those who may struggle with complex forms.*

Benefits —

- An online voting system (OVS) offers numerous benefits that can enhance the electoral process, improve voter participation, and streamline administrative tasks. Here are some of the key advantages:

1. Increased Accessibility

- **Convenience:**
- Voters can cast their ballots from anywhere with internet access, eliminating the need to travel to polling places. This is particularly beneficial for those with mobility challenges or those living in remote areas.
- **Extended Voting Period:**
- Online voting can allow for a longer voting period, enabling voters to participate at their convenience rather than being restricted to a single day.

2. Higher Voter Turnout

- **Ease of Use:** The simplicity and convenience of online voting can encourage more individuals to participate in elections, potentially leading to higher voter turnout rates.
- **Engagement of Younger Voters:**
- Younger generations, who are more accustomed to digital platforms, may be more inclined to vote if the process is accessible online.
-

3. Cost Efficiency

- **Reduced Operational Costs:** Online voting can lower costs associated with traditional voting methods, such as printing ballots, staffing polling places, and transporting materials.
- **Streamlined Administration:** Automating various aspects of the voting process can reduce the workload for election officials and minimize the potential for human error.
-

4. Enhanced Security and Accuracy

- **Advanced Security Measures:** Modern online voting systems can implement robust security protocols, including encryption and multi-factor authentication, to safeguard the voting process against fraud and tampering.
- **Real-Time Validation:** Voter identities can be verified in real time, ensuring that only eligible individuals can cast votes and that each vote is counted accurately.

5. Immediate Results

- **Faster Vote Counting:** Online voting allows for quicker tallying of votes, leading to faster election results compared to traditional methods.
- **Real-Time Reporting:** Election officials can monitor voting progress and detect any irregularities in real time, allowing for immediate intervention if necessary.

6. Environmental Benefits

- **Reduced Paper Usage:** Online voting significantly decreases the need for printed ballots and other paper materials, contributing to environmental sustainability.
- **Lower Carbon Footprint:** By minimizing travel to polling places, online voting can reduce the carbon emissions associated with transportation.

4. Flexibility in Voting Methods

- **Multiple Voting Options:** OVS can accommodate various voting methods, including ranked-choice voting or proportional representation, which can enhance democratic participation.
- **Remote Voting:** Voters who are overseas, in military service, or otherwise unable to vote in person can participate in elections easily.

5. Improved Voter Education

- **Access to Information:** Online platforms can provide voters with easy access to information about candidates, measures, and the voting process itself, empowering them to make informed decisions.
- **Interactive Tools:** Educational resources, such as videos or interactive guides, can help voters understand how to navigate the online voting system.

9. Audit and Transparency

- **Enhanced Audit Trails:** Digital systems can maintain detailed logs of the voting process, facilitating audits and ensuring transparency in the electoral process.
- **Public Trust:** Transparent processes and the ability to audit results can enhance public confidence in the integrity of elections.

Online voting systems (OVS) have a wide range of potential applications beyond traditional electoral processes. Here are several key areas where online voting can be effectively implemented:

1. Public Elections

- **National and Local Elections:** OVS can be used for federal, state, and local elections, allowing voters to cast their ballots conveniently from anywhere.
- **Referendums and Ballot Initiatives:** Citizens can vote on specific policy issues or proposed amendments to laws through secure online platforms.

2. Corporate Voting

- **Shareholder Meetings:** Companies can use online voting to facilitate shareholder votes on important issues, such as mergers, acquisitions, or board member elections.
- **Employee Voting:** Organizations can conduct internal elections for employee representatives, committees, or other governance structures.

3. Non-Profit Organizations

- **Board Elections:** Non-profits can utilize online voting to elect board members or make decisions on organizational policies, ensuring greater participation from stakeholders.
- **Membership Voting:** Organizations can allow members to vote on issues such as bylaw changes or funding allocations.

4. Student Government Elections

- **University and College Elections:** Educational institutions can implement OVS for student government elections, making it easier for students to participate and increasing turnout.
- **Class Representative Elections:** Online voting can streamline the process for electing class representatives or other student leaders.

5. Community and Local Governance

- **Neighborhood Associations:** Local community organizations can use online voting to make decisions on community projects, budgets, and leadership positions.
- **Town Hall Meetings:** Residents can vote on local issues or initiatives discussed in town hall meetings, enhancing civic engagement.

6. Polling for Surveys and Feedback

- **Public Opinion Polls:** Organizations can conduct public opinion polls on various issues, gathering data to inform policy decisions or marketing strategies.
- **Customer Feedback:** Businesses can use online voting to gather feedback on products, services, or new initiatives, allowing customers to express their preferences.

7. Political Party Primaries and Caucuses

- **Party Elections:** Political parties can use online voting to select candidates for elections, allowing party members to participate conveniently.
- **Policy Decisions:** Party members can vote on key policy positions or platform issues, promoting grassroots involvement.

8. International Voting

- **Voting for Expatriates:** Citizens living abroad can participate in elections of their home country through secure online voting, ensuring their voices are heard.

- **International Organizations:** Organizations like the United Nations or the European Union can use online voting for member states to vote on resolutions or policy matters.

9. Civic Engagement Initiatives

- **Community Polling:** Governments can conduct online polls to gauge public opinion on community issues, encouraging citizen participation in governance.
- **Crowdsourcing Ideas:** Online platforms can facilitate voting on community projects or initiatives proposed by residents, fostering collaboration and innovation.

10. Research and Academia

- **Academic Research:** Researchers can use online voting to gather data on public opinion or conduct studies on electoral behavior and preferences.
- **Student Surveys:** Universities can conduct online surveys for research purposes, allowing students to provide feedback on various academic and administrative matters.

❖ INDEX.HTML FILE

```
<!DOCTYPE html>
<html>
<head>
  <meta charset='utf-8'>
  <meta http-equiv='X-UA-Compatible' content='IE=edge'>
  <title>online voting system</title>
  <link rel="stylesheet" href="css/style.css">
  <meta name='viewport' content='width=device-width, initial-scale=1'>
  <link rel='stylesheet' type='text/css' media='screen' href='main.css'>
  <script src='main.js'></script>
</head>
<body>
  <center>
    <div id="header section">
      <h1>online voting system</h1>
    </div>
    <hr>
    <div id="bodysection">
      <form action="api/login.php"method="POST">
        <h2>login</h2>
        <br><br>
        <input type="text" name="email" placeholder="Enter email"><br><br>
        <input type="number" name="mobile" placeholder="Enter mobile
number"><br><br>
        <input type="password" name="password" placeholder="Enter password
number"><br><br>
        <select id="dropbox">
          <option value="1">voter</option>
          <option value="2">group</option>
        </select>
        <br><br>
        <button id="Loginbtn">Login</button><br><br>
        new user?<a href="routes/register.html">register here</a>
      </form>
    </div>
  </center>
</body>
</html>
```

Online Voting System

Login

Enter mobile

Enter password

Voter ▼

[Login](#)

New user? [Register here](#)

❖ REGISTRATION.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>online voting system - registration</title>
  <link rel="stylesheet" href="../css/styleSheet.css">
</head>
<body>
  <style>
    #address{
      width: 35%;
    }
    #Imagepart{
      border: 2px solid black;
      border-radius: 5px;
      padding: 10px;
      width: 30%;
    }
    #role{
      border: 2px solid black;
      border-radius: 5px;
      padding: 10px;
      width: 30%;
    }
    #role select{
```


Online Voting System

Registration

Name Mobile

Password Confirm Password

Address

Upload image: No file chosen

Select your role:

Already user? [Login here](#)

border-radius: 5px;
padding: 10px;

```

    }
</style>
<center>
<div id="header section">
    <h1>online voting system</h1>
</div>

<hr>
<div id="bodysection">

<h2>registration</h2>
<form action="../api/register.php" method="post" enctype="multipart/form-data">
<input type="text" name="Name" placeholder="Enter name">
<input type="number" name="mobile" placeholder="Enter mobile number">
<br><br>
<input type="password" name="password" placeholder="Enter password">
<input type="password" name="confirm password" placeholder="Enter confirm
password">
<br><br>
<input id="address" type="address" name="Address" placeholder="Address">
<br><br>

<div id="Imagepart">
uploade image : <input type="file" name="photo" >

```

```
</div>

<br><br>
<div id="role">

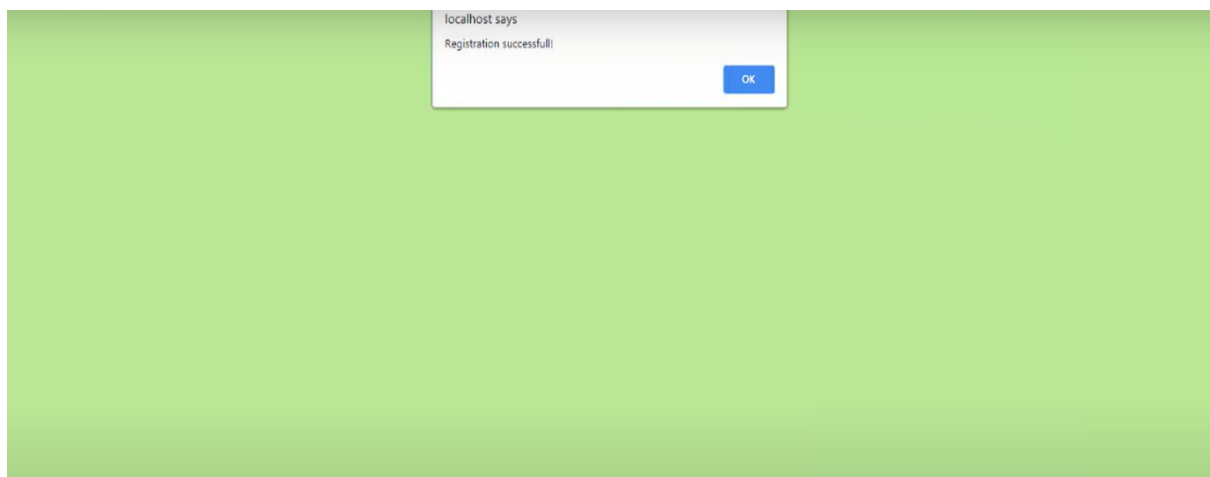
  select role:<select name="role">
    <option value="1">voter</option>
    <option value="2">group</option>
  </select>
</div>

<br><br>
<button
  style="padding-block: 5px;
        padding-inline: 5px;
        background-color: #0984e3;
        color: white;">
  Register</button><br><br>
  already user?<a href="../index.html">Login here</a>
</form>
</div>
</center>
</body>
</html>
```

—

❖ LOGIN.PHP

```
<?php
include('connect.php');
$mobile=$_POST['mobile'];
$password=$_POST['password'];
$role=$_POST['role'];
$check=mysqli_query($connect,"SELECT * FROM user WHERE mobile='$mobile' and
password='$password' and role='$role' " );
if(mysqli_num_rows($check)>0){
$userdata=mysqli_fetch_array($check);
$groups=mysqli_query($connect,"SELECT * FROM user WHERE role=2 ");
$groupsdata=mysqli_fetch_all($groups, MYSQLI_ASSOC);
$_SESSION['userdata']=$userdata;
$_SESSION['groupsdata']=$groupsdata;
echo'
    <script>
        window.location='../routes/dashboard.php';
    </script>
    ';
}
else{
    echo'
    <script>
        alert("invalid credential or user not found!");
        window.location='../';
    </script>
    ';
}
?>
```

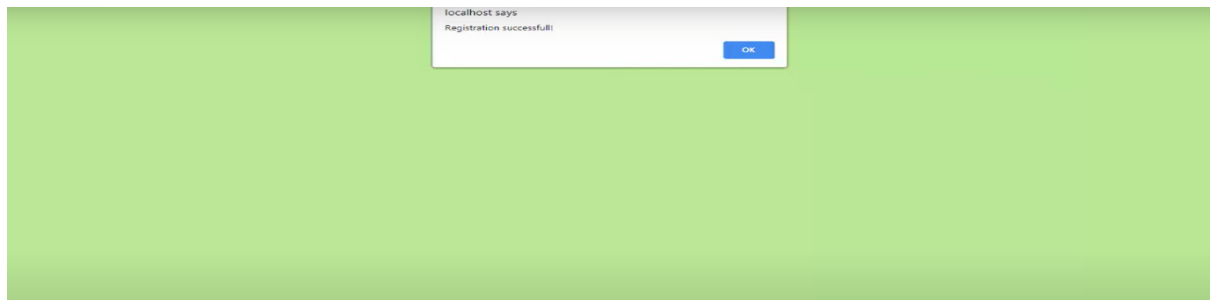


❖ REGISTRATION.PHP

```
<?php

include("connect.php");
$name=$_post['name'];
$mobile=$_post['mobile'];
$address=$_post['address'];
$password=$_post['password'];
// $confirmpassword=$_post['confirm password'];

// $image=$_FILES['name']['photo'];
// $tmp_name=$_FILES['photo']['tmp_name'];
$role=$_post['role'];
if($password==$confirmpassword){
move_uploaded_file($tmp_name,"../uploads/$image");
$insert=mysqli_query($connect,"INSERT INTO user (name,mobile,address,password,role,status,votes) VALUES
('$name','$mobile','$address','$password','$role',0,0)");
if($insert){
    echo'
    <script>
    alert("registration successfull!");
    window.location="../";
    </script>
    ';
}
else{
    echo'
    <script>
    alert("some error occurred!");
    window.location="../routes/register.html";
    </script>
    ';
}
}
else{
    echo'
    <script>
    alert("Password and confirmpassword does not match!");
    window.location="../routes/register.html";
    </script>
    ';
}
?>
```



❖ CONNECTION.PHP

```
<?php
$connect=mysqli_connect("localhost","root","","voting") or die("connection failed!");
if($connect){
    echo "connected!";
}
else {

    echo "not connected!";
}
?>
```

❖ DASHBOARD.PHP

```
<?php
session_start();
if(!isset($_session['userdata'])){
    header("location../");
}
$userdata=$_SESSION['userdata'];
$groupsdata=$_SESSION['groupsdata'];
if($_SESSION['userdata']['status']==0){
    $status='<b style="color:red">not voted </b>';
}
else{
    $status='<b style="color:green">voted </b>';
}
?>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>online voting system - dashboard</title>
  <link rel="stylesheet" href="../css/styleSheet.css">
</head>
<body>
  <style>
    #backbtn{
      padding-block: 5px;
      padding-inline: 5px;
      background-color: #0984e3;
      color: white;
      float:left;
      margin: 10px;
    }
    #logoutbtn{
      padding-block: 5px;
      padding-inline: 5px;
      background-color: #0984e3;
      color: white;
      float:right;
      margin: 10px;
    }
    #profile{
      background-color: white;
      width:40%;
      padding:20px;
      float:left;
    }

    #group{
      background-color: white;
      width:60%;
      padding:20px;
      float:right;
    }

    #votebtn{
      padding-block: 5px;
      padding-inline: 5px;
      background-color: #0984e3;
      color: white;
    }
    #mainpanel{
      padding:10px;
    }
    #headersection{
      padding: 10px;
    }
  </style>

```

```
#voted{
padding-block: 5px;
padding-inline: 5px;
background-color: #0984e3;
color: white;
}
```

```
</style>
```

```
<div id="mainSection">
  <center>
    <div id="headersection">
      <a href=".."><button id="backbtn">back</button></a>
      <a href="logout.php"><button id="logoutbtn">logout</button></a>
    <h1>online voating system</h1>
  </div>
</center>
  <hr>
  <div id="mainpanel">

    <div id="profile">
<center><?php echo $userdata['photo'] ?>"height="100"width="100"></center><br><br>
      <b>name:</b><?php $userdata["name"]?><br><br>
      <b>mobile:</b><?php $userdata["mobile"]?><br><br>
      <b>Address:</b><?php $userdata["address"]?><br><br>
      <b>status:</b><?php $status?><br><br>
    </div>

    <div id="group">

      </div>

</div>

    <div id="group">
<?php
if($_SESSION['groupsdata']){
for( $i=0; $i<count($groupsdata); $i++){
?>
<div>
  <?php echo $groupsdata[$i]["photo"] ?>"height="100" width="100">
  <b>group name:</b> <?php echo $groupsdata[$i]["name"]?><br><br>
  <b>Votes:</b><?php echo $groupsdata[$i]["name"]?><br><br>
  <form action=".."><?php echo $groupsdata[$i]["votes"]?>>
  <input type="hidden" name="gvotes" value="<?php echo $groupsdata[$i]["votes"]?>">
```

```

        <input type="hidden" name="gid" value="<?php echo $groupsdata[$i]["id"]?>">
<?php
if($_SESSION["userdata"]["status"]==0){

    ?>
    <input type="submit" name="votebtn" value="vote" id="votebtn">
    <?php

}
else{
    ?>
    <button disable type="submit" name="votebtn" value="vote" id="voted">voted</button>
    <?php

}

?>

</form>
</div>
<hr>
<?php
}
}
else{

}
?>
</div>
</div>

</body>
</html>

```

❖ LOGOUT.PHP

```

<?php
session_start();
session_destroy();

header("location:../");

?>

```

❖ LOGOUT.PHP

```

<?php
session_start();
include('connect.php');
$votes=$_POST["gvotes"];
$total_votes=$votes+1;

```



```

$gid=$_POST["gid"];
$uid=$_SESSION['userdata']['id'];
$update_votes=mysqli_query($connect,"UPDATE user SET votes=$total_votes'WHERE id=$gid'");
$update_user_status=mysqli_query($connect,"UPDATE user SET status=1 WHERE id=$uid'");
if($update_votes and $update_user_status ){

$groups=mysqli_query($connect,"SELECT * FROM user WHERE role=2 ");
$groupsdata=mysqli_fetch_all($groups, MYSQLI_ASSOC);
$_SESSION['userdata']['status']=1;
$_SESSION['groupsdata']=$groupsdata;

echo'
<script>
alert("voteing succesful!");
window.location="../routes/dashboard.php";
</script>
';

}


else{
    echo'
    <script>
    alert("some error occured!");
    window.location="../routes/dashboard.php";
    </script>
    ';
}
?>

```

Back

Online Voting System

Logout




Name : Shahebaz Khan Shabbir Khan Pathan
Mobile : 9511846837
Address : Andheri East, Near Govt. College, Mumbai
Status : Not Voted

Group Name : Soehl Khan Salim Khan Pathan

Votes : 0


Vote



Group Name : Shoaib Khan Shabbir Khan Pathan

Votes : 0

Vote



INTRODUCTION

General Introduction

An online voting system (OVS) is a digital platform that enables voters to cast their ballots via the internet, streamlining the electoral process and enhancing accessibility. As technology continues to evolve, traditional voting methods are increasingly being supplemented or replaced by online systems, which offer a modern approach to democratic participation.

Key Features of Online Voting Systems

1. **Accessibility:** OVS allows voters to participate in elections from any location with internet access, removing barriers associated with physical polling places. This is particularly beneficial for individuals with disabilities, those living in remote areas, or expatriates who may be unable to return to their home country for elections.
2. **Convenience:** Voters can cast their ballots at their convenience, often over an extended voting period, rather than being restricted to a specific day. This flexibility can lead to increased voter turnout and engagement.
3. **Efficiency:** Online voting systems can streamline the voting process, reducing the time and resources required for ballot distribution, collection, and counting. Automated processes can enhance accuracy and reduce human error.
4. **Security:** Modern OVS implementations incorporate advanced security measures, such as encryption, secure authentication, and audit trails, to protect voter data and ensure the integrity of the election process. These features help to mitigate concerns about fraud and manipulation.
5. **Real-Time Results:** Online voting allows for quicker counting of votes and faster reporting of results, providing immediate feedback to candidates and the public. This can enhance transparency and trust in the electoral process.

Problem Statement

The online voting system faces several critical challenges that can undermine its effectiveness and integrity. Security vulnerabilities expose the system to potential cyberattacks, risking the confidentiality and accuracy of voter data. Additionally, the digital divide limits access for certain populations, leading to unequal participation in elections. Concerns about voter authentication and identity verification can result in fraud or disenfranchisement. Finally, ensuring compliance with legal and regulatory standards adds complexity to the implementation and operation of online voting systems.

Problem detection

online voting systems is crucial to ensuring their integrity, security, and reliability. Here are some key areas where issues may arise, along with methods for detection:

1. Security Vulnerabilities

- **Detection Methods:**
- **Penetration Testing:** Regularly conduct simulated cyberattacks to identify vulnerabilities in the system.

- **Security Audits:** Perform comprehensive audits of the system's architecture, code, and infrastructure to uncover weaknesses.
- **Monitoring Tools:** Implement intrusion detection systems (IDS) to monitor for suspicious activities or unauthorized access attempts.

2. Data Integrity Issues

- **Detection Methods:**
- **Blockchain Technology:** Utilize blockchain for secure and immutable record-keeping, allowing for easy verification of vote integrity.
- **Checksum and Hashing:** Employ cryptographic techniques to ensure that data has not been altered during transmission or storage.
- **Audit Trails:** Maintain detailed logs of all voting activities, enabling independent verification of the election process.

3. Authentication and Identity Verification Problems

- **Detection Methods:**
- **Multi-Factor Authentication (MFA):** Implement MFA to ensure that only authorized voters can access the system.
- **User Behavior Analytics:** Monitor user behavior patterns to detect anomalies that may indicate fraudulent activity.
- **Regular Testing:** Conduct periodic assessments of the authentication mechanisms to identify weaknesses.

4. Accessibility Issues

- **Detection Methods:**
- **Usability Testing:** Conduct tests with diverse user groups, including those with disabilities, to identify barriers to access.
- **Feedback Mechanisms:** Implement user feedback forms to gather insights on accessibility challenges faced by voters.
- **Analytics:** Analyze user engagement data to identify patterns that may indicate difficulties in accessing the system.

5. System Performance Problems

- **Detection Methods:**
- **Load Testing:** Simulate high traffic conditions to assess how the system performs under stress and identify potential bottlenecks.
- **Real-Time Monitoring:** Use performance monitoring tools to track system responsiveness and uptime during the voting period.
- **Error Logging:** Implement comprehensive error logging to capture and analyze system failures or performance issues.

6. Compliance and Regulatory Issues

- **Detection Methods:**
- **Regular Compliance Audits:** Conduct audits to ensure adherence to relevant laws and regulations governing online voting.
- **Legal Reviews:** Engage legal experts to review system policies and procedures to identify areas of non-compliance.
- **Stakeholder Feedback:** Gather feedback from regulatory bodies and stakeholders to ensure that the system meets all necessary standards.

Conclusion

Proactively detecting problems in online voting systems is essential for maintaining public trust and ensuring the democratic process. By employing a combination of security measures, performance monitoring, and user feedback mechanisms, organizations can identify and address potential issues before they affect the integrity of

elections. Regular assessments and updates to the system will further enhance its reliability and security.

Problem Solving

Detecting problems in an online voting system is essential to ensure its integrity, security, and overall effectiveness. Here are several methods and strategies to identify potential issues within such systems:

1. Security Testing

- **Penetration Testing:** Conduct regular penetration tests to simulate attacks on the system. This helps identify vulnerabilities that could be exploited by malicious actors.
- **Vulnerability Scanning:** Use automated tools to scan the system for known vulnerabilities and weaknesses in the software and infrastructure.

2. Monitoring and Logging

- **Real-Time Monitoring:** Implement monitoring tools that continuously track system performance, user activity, and network traffic. Anomalies or unusual patterns can indicate potential security breaches or system failures.
- **Audit Logs:** Maintain detailed logs of all actions taken within the system, including user logins, vote submissions, and administrative actions. Regularly review these logs for suspicious activity or unauthorized access.

3. User Behavior Analytics

- **Anomaly Detection:** Utilize machine learning algorithms to analyze user behavior and detect anomalies that may indicate fraud or system misuse, such as unusual voting patterns or access attempts from unfamiliar locations.

4. Load and Performance Testing

- **Stress Testing:** Simulate high traffic conditions to evaluate how the system performs under peak loads. This helps identify bottlenecks and performance issues that could hinder voter access during critical periods.
- **Response Time Monitoring:** Measure the response times of different components of the system to ensure they meet acceptable performance standards.

5. Accessibility Audits

- **Usability Testing:** Conduct usability tests with diverse user groups, including those with disabilities, to identify accessibility issues that may prevent some voters from participating.
- **Feedback Mechanisms:** Implement surveys or feedback forms to gather insights from users about their experience with the voting system, focusing on accessibility and ease of use.

6. Compliance Checks

- **Regulatory Audits:** Regularly review the system to ensure compliance with relevant laws and regulations governing online voting, such as data protection and voter privacy laws.
- **Legal Reviews:** Engage legal experts to assess the system's policies and procedures for compliance with electoral regulations.

7. Post-Election Audits

- **Independent Audits:** Conduct independent audits of the voting process after elections to verify the accuracy of the results and the integrity of the system.

- **Risk-Limiting Audits:** Implement statistical audits that allow for a sample of votes to be manually verified against the reported results, ensuring that any discrepancies are identified and addressed.

8. User Authentication Checks

- **Multi-Factor Authentication (MFA):** Ensure that robust authentication measures are in place to verify voter identities and prevent unauthorized access.
- **Identity Verification:** Regularly test and update identity verification processes to ensure they are effective and secure.

Proposed System Overview

The proposed Online Form Filling Application is a web-based solution designed to provide a user-friendly, efficient, and secure platform for filling out forms. The system consists of the following components:

User Interface

- Intuitive and easy-to-use interface for users to fill out forms
- Auto-fill functionality to save time and effort
- Real-time validation to ensure accurate and complete information

Form Builder

- Drag-and-drop interface for administrators to create and design forms
- Customizable form templates and fields
- Integration with existing systems and databases

Form Filling Engine

- Automated form filling process using user data and auto-fill functionality
- Real-time validation and error detection
- Integration with payment gateways and third-party services

Database and Storage

- Secure and scalable database to store user data and form submissions
- Data encryption and access controls to ensure security and privacy
- Automated backups and disaster recovery processes

Security and Authentication

- Advanced security measures to protect sensitive user information
- Multi-factor authentication and access controls
- Regular security audits and penetration testing

Reporting and Analytics

- Real-time reporting and analytics to track form submissions and user behavior
- Customizable dashboards and reports
- Integration with existing analytics and reporting tools

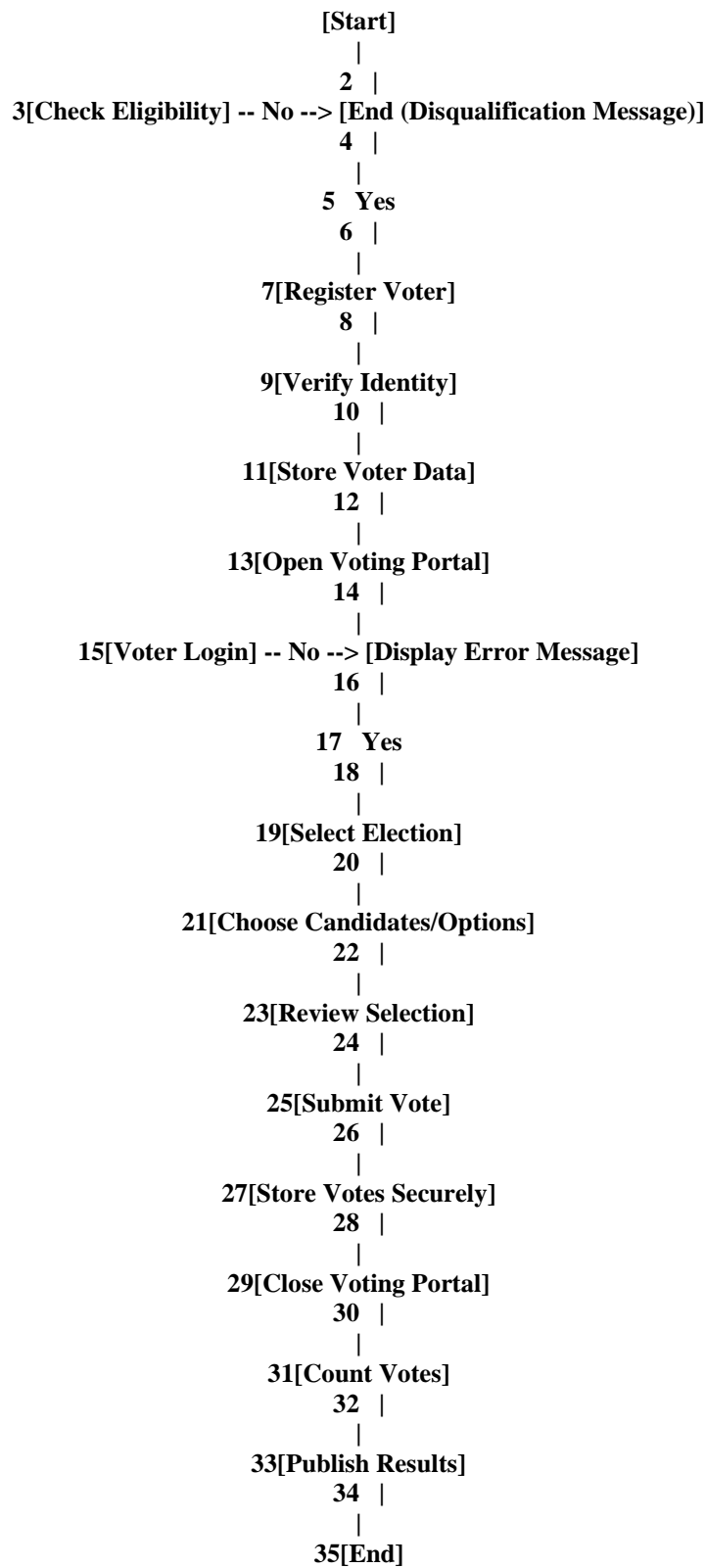
Integration and API

- RESTful API for integration with existing systems and third-party services
- Integration with payment gateways, CRM systems, and other applications
- Customizable API keys and access controls

The proposed Online Form Filling Application provides a comprehensive solution for filling out forms, addressing the issues associated with traditional form filling and providing a user-friendly, efficient, and secure platform for user

an online voting system requires a comprehensive methodology that encompasses security, accessibility, user experience, and compliance. Here's a structured approach to developing a robust online voting system:

Flow Chart



CONCLUSION:

An **online voting system** offers a modern, convenient, and efficient way to conduct elections by allowing voters to cast their ballots remotely via the internet. It enhances accessibility, reduces costs, and streamlines the voting process. However, to ensure its widespread adoption and trust, key challenges such as security, voter authentication, privacy, and resistance to cyber threats must be addressed. When implemented with robust safeguards, online voting can improve voter participation and make elections more inclusive and transparent.

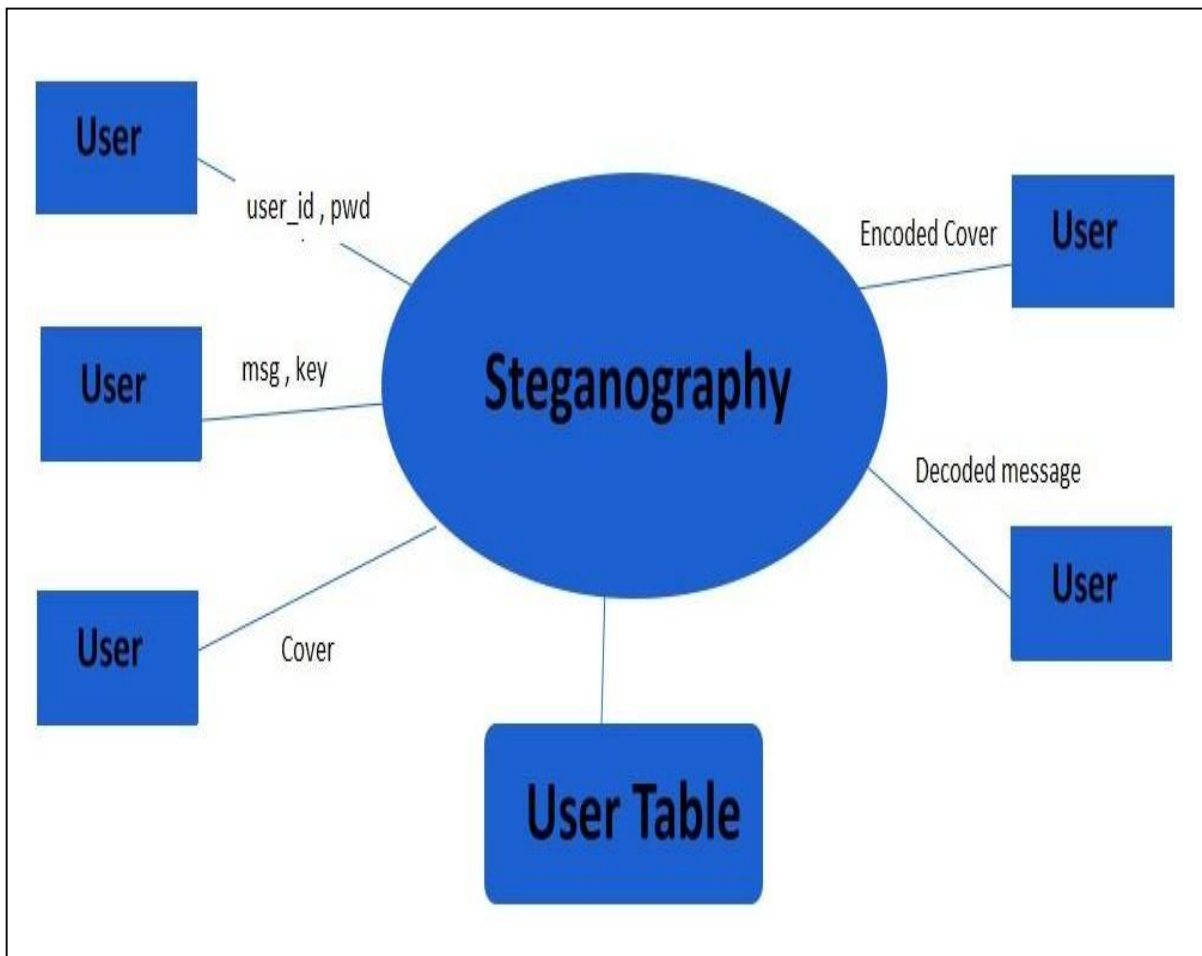




Fig. 1.4.1 Agile Model

Principles of Agile

- Satisfy the customer through early and continuous delivery of valuable software.
 - Welcome changing requirements.
 - Deliver working software frequently.
 - Work together daily throughout the project.
- Build projects around motivated individuals who are supported and trusted to get the job done.
 - Use face-to-face conversation whenever possible.
 - Working software is the primary measure of progress.
 - Maintain a constant pace indefinitely.
 - Give constant attention to technical excellence and good design.
- Simplicity—the art of maximizing the amount of work not done—is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- Reflect on how to become more effective, then tune and adjust accordingly at regular intervals.

Flow Chart

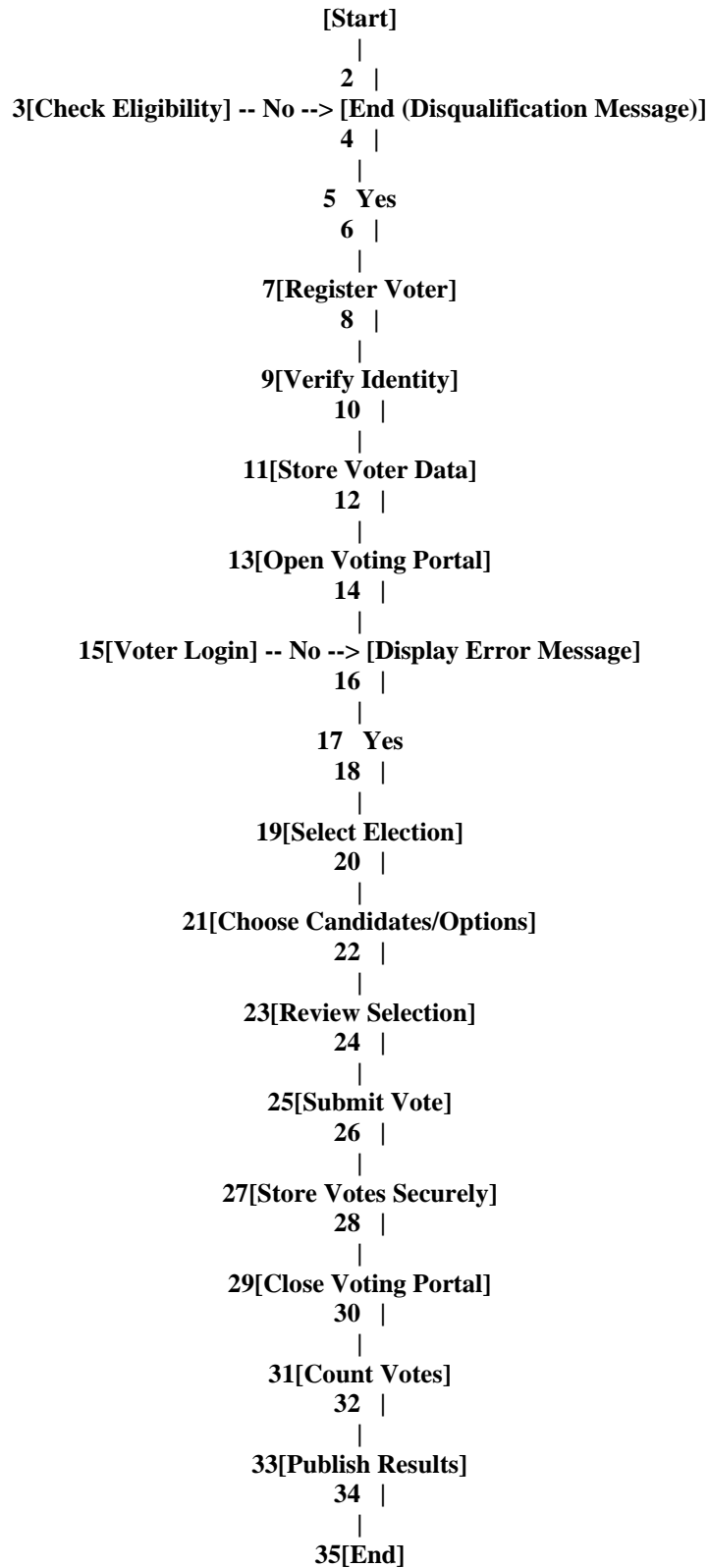


Fig. 1.4.2 Flow Chart

WHAT IS STEGANOGRAPHY?

Steganography means “Masking of message” so that others will have no ability to recognize it.

While cryptography is a way of keeping and conducting data in a specific form so that only those who are authorized recipient can recognize it. Steganography takes a step further by modification of cryptography in the field of hiding encoded information, instead of observer nobody can suspects its presence. Whoever detecting your documents will fail to recognize it.

It consists encrypted data key concept behind steganography.

Steganography includes the concealment of information within computer files. In digital steganography, electronic communications may include steganographic coding inside of a transport layer, such as a document file, image file, program or protocol. Media files are ideal for steganographic transmission because of their large size. For example, a sender might start with an innocuous image file and adjust the color of every hundredth pixel to correspond to a letter in the alphabet. The change is so subtle that someone who is not specifically looking for it is unlikely to notice the change.

Steganography techniques

In modern digital steganography, data is first encrypted or obfuscated in some other way and then inserted, using a special algorithm, into data that is part of a particular file format such as a JPEG image, audio or video file. The secret message can be embedded into ordinary data files in many different ways. One technique is to hide data in bits that represent the same color pixels repeated in a row in an image file. By applying the encrypted data to this redundant data in some inconspicuous way, the result will be an image file that appears identical to the original image but that has "noise" patterns of regular, unencrypted data.

The practice of adding a watermark -- a trademark or other identifying data hidden in multimedia or other content files -- is one common use of steganography. Watermarking is a technique often used by online publishers to identify the source of media files that have been found being shared without permission.

While there are many different uses of steganography, including embedding sensitive information into file types, one of the most common techniques is to embed a text file into an image file. When this is done, anyone viewing the image file should not be able to see a difference between the original image file and the encrypted file; this is accomplished by storing the message with less significant bits in the data file. This process can be completed manually or with the use of a steganography tool.

Advantages over cryptography

Steganography is distinct from cryptography, but using both together can help improve the security of the protected information and prevent detection of the secret communication. If steganographically-hidden data is also encrypted, the data may still be safe from detection -- though the channel will no longer be safe from detection. There are advantages to using steganography combined with encryption over encryption-only communication.

The primary advantage of using steganography to hide data over encryption is that it helps obscure the fact that there is sensitive data hidden in the file or other content carrying the hidden text. Whereas an encrypted file, message or network packet payload is clearly marked

and identifiable as such, using steganographic techniques helps to obscure the presence of the secure channel.

Types of Steganography:

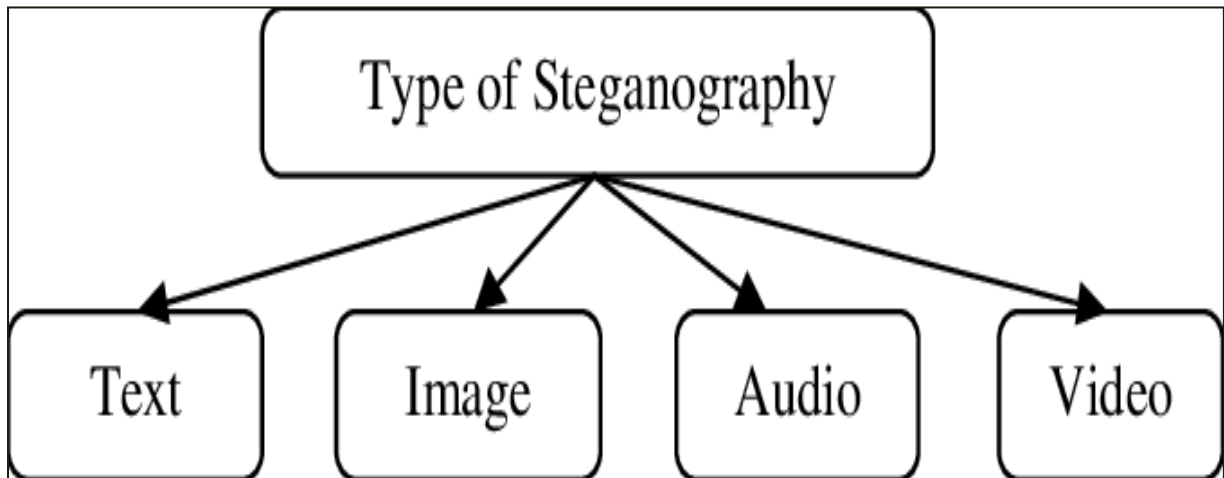


Fig. 1.5.1 Types of Steganography

Image Steganography

Image Steganography is in which data is hidden within an image file. The image selected for it is called as the cover-image and the image obtained after the steganography process is called the stego-image. Image steganography is of various types.

In image steganography an image is usually the carrier that holds the secret information. Cover images shows the image which is used for hiding the secret data as a payload. The embedding technique is the algorithm which is useful for hiding the secret message inside the cover image with the help of stego key. The stego-key needs to be shared with both the ends.

The ‘stego-image’ is the final output image that keeps the secret information hidden. There is also an extraction technique which is used to recover the secret message from stego-image with the help of stego-key. The counter of steganography is steganalysis or an attack on steganography. “Steganalysis is an art and science of detecting the existence or recovering the secret message from stego images.”

Audio Steganography

Audio Steganography is a technique used to transmit hidden information by modifying an audio signal in an imperceptible manner.

It is the science of hiding some secret text or audio information in a host message. The host message before steganography and stego message after steganography have the same characteristics.

Text Steganography

Text steganography can involve anything from changing the formatting of an existing text, to changing words within a text, to generating random character sequences or using context-free grammars to generate readable texts.

Text steganography is believed to be the trickiest due to deficiency of redundant information which is present in image, audio or a video file. The structure of text documents is identical with what we observe, while in other types of documents such as in picture, the structure of document is different from what we observe.

Therefore, in such documents, we can hide information by introducing changes in the structure of the document without making a notable change in the concerned output.

Video Steganography

Video steganography is becoming an important research area in various data hiding technologies, which has become a promising tool because not only the security requirement of secret message transmission is becoming stricter but also video is more favoured.

Scope of the Project

This project is developed for hiding information in any image file. The scope of the project is implementation of steganography tools for hiding information includes any type of information file and image files and the path where the user wants to save Image and extruded file.

- This can be used as base for the use by various intelligent services organizations.
- There are distributed steganography methods including methodologies that distribute the payload through multiple carrier files in diverse locations to make detection more difficult.
- It aims to help certain defence organizations of the country where security is the primary aspect.

CHAPTER 2

FEASIBILITY REPORT

Feasibility Study

The feasibility study is an evaluation and analysis of a proposed project which is based on extensive investigation and research to support the process of decision making.

Feasibility study evaluates the project's potential for success; therefore, perceived objectivity is an important factor in the credibility of the study for potential investors and lending institutions.

It mainly refers to four areas of feasibility:

- Technical
- Economic
- Operational
- Legal

Technical Feasibility

The technical feasibility assessment is focused on gaining an understanding of the present technical resources and their applicability to the expected needs of the proposed system.

To develop this project, developers should have knowledge about working with Python, Django framework, Steganography, Cryptography and Machine Learning.

Economic Feasibility

Economic Feasibility assessment typically involves a cost/benefit analysis. This project is economically feasible because it implements on a cloud server as it is a web application. To access this web application the user needs a phone or a tablet or any device that is connected to the internet and can run an internet browser. This web app is available for free.

Operational Feasibility

Operational feasibility is a measure of how well a proposed system solves the problems and takes advantages of opportunities identified during scope definition and how it satisfies the requirements identified in the requirement analysis phase of development.

This project is efficient, fast and gives better results. This web application can run on any day to day devices which can access the internet and can run an internet browser.

Legal Feasibility

Determines whether the proposed system conflicts with legal requirements and whether the proposed venture is acceptable in accordance to the laws of the land. This project does not conflict with any legal requirements.

System Design

In this project we have designed a web application in which we can send messages securely with a feature of hiding your messages on images and audio files. Also, it can block any spam messages.

This web application let a user in with the help of login id and password. It uses PNG images and wav audio files to hide the encrypted text message with capability of blocking any message that is spammy. The encryption of text messages is done using AES algorithm and machine learning is used to identify and block any spam messages.

Description

A user chats to another user on an ajax based chat system and it is ensured with the help of machine learning algorithms that any user is not spamming to another user and no other third party can access to the chats as we are using encryption and steganography to security and privacy.

A user(sender) is sending another user(receiver) a message which has an important message so the sender uses an image as a hiding means for sending a message to ensure security of the message all the way till the receiver doesn't get the message.

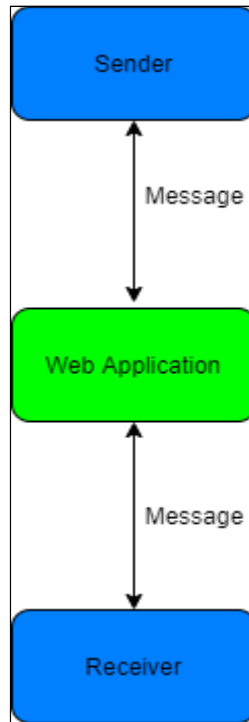


Figure 2.3.1 To indicate Working of model

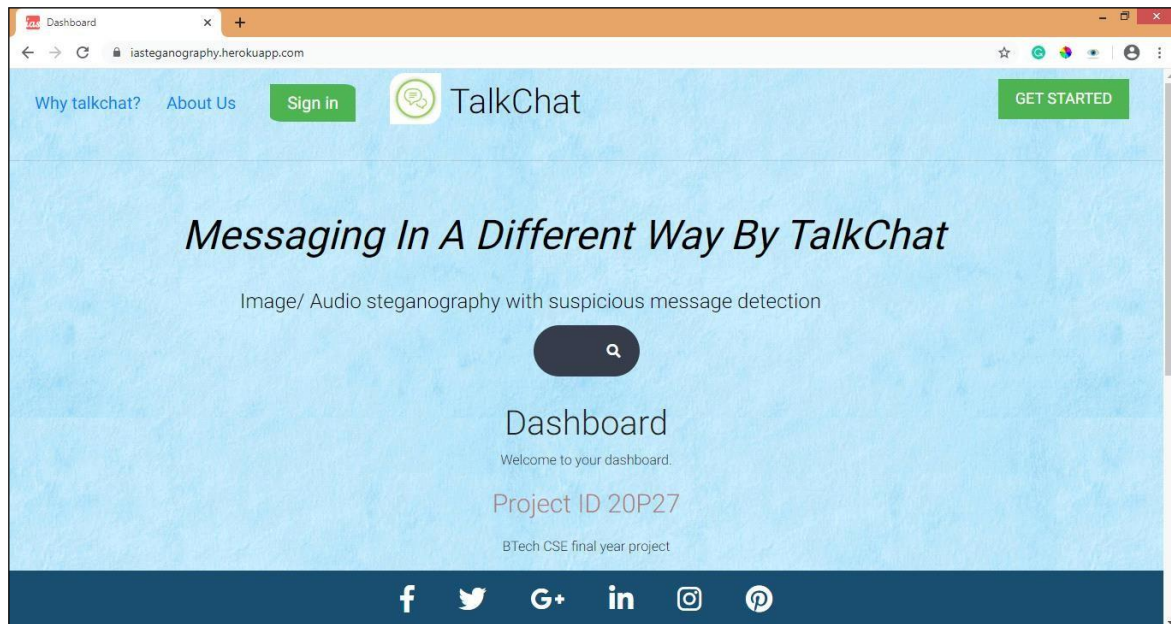


Figure 2.3.2: Screenshot of the application (Desktop View)

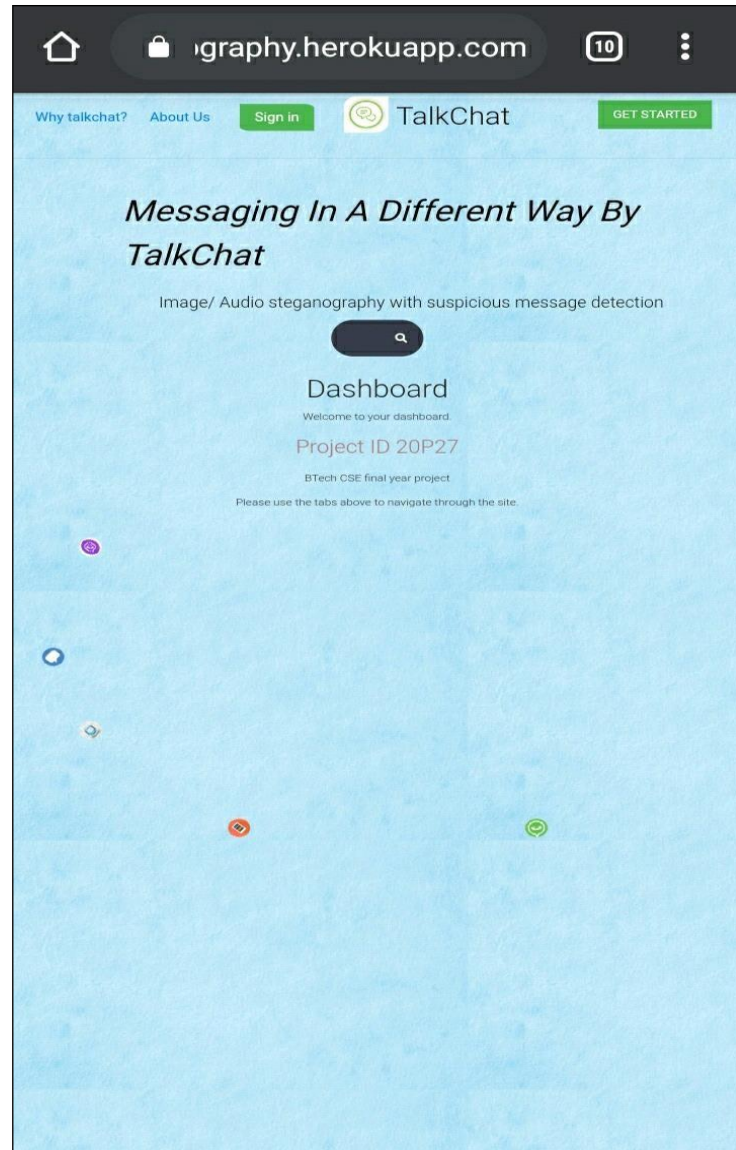


Figure 2.3.3: Screenshot of the application (Mobile View)

CHAPTER 3 REQUIREMENT ELICITATION

Requirement elicitation is the practice of collecting requirements of a system users that are necessary for the Identification system.

Requirement elicitation is non-trivial because you can never be sure you get all requirements for the users and customer by just asking them what the system should do or not do.

The objective of this phase is to clearly understand clients or customer's requirements and to systematically organize these requirements.

Requirements may be documented in various forms, usually including a summary list and may include natural-language documents, use cases, use stories, or process specifications.

Requirement Analysis

Requirements analysis in systems engineering and software engineering, encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product or project, taking account of possibly conflicting requirements of the various stakeholders, analyzing, documenting, validating and managing software or system requirements.

A software requirement is a capability needed by the user to solve a problem or to achieve an objective. In other words, requirement is a software capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documentation. Ultimately, what we want to achieve is to develop quality software that meets customers' real needs on time and within budget.

Perhaps the greatest challenge being faced by software developers is to share the vision of the final product with the customer. All stakeholders in a project - developers, end users, software managers, customer managers - must achieve a common understanding of what the product will be and do, or someone will be surprised when it is delivered. Surprises in software are almost never good news.

Therefore, we need ways to accurately capture, interpret, and represent the voice of customers when specifying the requirements for a software product.

Requirements analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

It consists of following activities:

Requirements Gathering

Eliciting requirements (e.g. The project charter or definitions), business procedure, documentation, and stakeholder interviews. This is sometimes also called requirements gathering.

Analyzing Requirements

It includes determining whether the stated requirements are clear, complete, consistent and unambiguous, and resolving any apparent conflicts and classification of requirements.

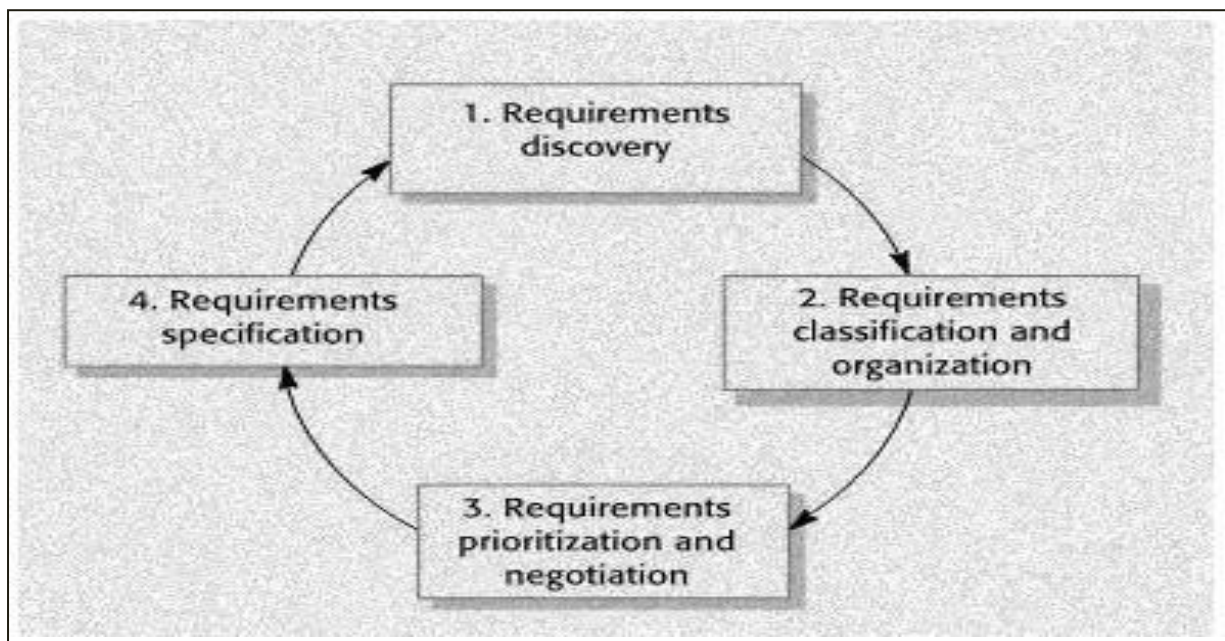


Figure 3.1.2.1 : Requirement Analysis Cycle

Classification of Requirements

The customers/users are not completely sure of what is needed, have a poor understanding of the capabilities and limitations of their computing environment, don't have a full understanding of the problem domain, have trouble communicating needs to the system engineer, omit information that is believed to be "obvious," specify requirements that conflict with the needs of other customers/users, or specify requirements that are ambiguous or untestable.

Functional Requirements

The Functional Requirements Specification documents the operations and activities that a system must be able to perform.

Functional Requirements should include:

- Descriptions of data to be entered into the system
- Descriptions of operations performed by each screen
- Descriptions of work-flows performed by the system
- Descriptions of system reports or other outputs
- Who can enter the data into the system?
- How the system meets applicable regulatory requirements

The Functional Requirements Specification is designed to be read by a general audience. Readers should understand the system, but no particular technical knowledge should be required to understand the document.

The Functional Requirements Specification describes what the system must do; how the system does it is described in the Design Specification.

If a User Requirement Specification was written, all requirements outlined in the User Requirement Specification should be addressed in the Functional Requirements Specification.

The Functional Requirements Specification should be signed by the System Owner and Quality Assurance. If key end-users, developers, or engineers were involved with developing the requirements, it may be appropriate to have them sign and approve the document as well.

Non-Functional Requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions.

The plan for implementing non-functional requirements is detailed in the system architecture, because they are usually Architecturally Significant Requirements. Non-functional requirements are in the form of "system shall be <requirement>", an overall property of the system as a whole or of a particular aspect and not a specific function.

A system may be required to present the user with a display of the number of records in a database. This is a functional requirement. How current this number needs to be, is a non-functional requirement. If the number needs to be updated in real time, the system architects

must ensure that the system is capable of displaying the record count within an acceptably short interval of the number of records changing.

Examples of non-functional requirements are as follows

- Performance
- Platform compatibility
 - Portability
 - Readability
- Fault tolerance
 - Recovery
 - Reliability
- Maintainability
 - Resilience
- Resource constraints
 - Response time
 - Reusability
 - Robustness
 - Scalability
 - Stability
- Supportability
 - Testability
- Transparency
 - Usability

CHAPTER 4DESIGNING

Software design deals with transforming the customer requirement, as describe in Software Requirement Specification document, into a form is implementable using a programming language.

Modular Architecture Design

It is a technique of high-level design technique. Modular design focuses on physical entities on the relationship between them. Identifying the entities containing your logical design constructs and managing dependencies between the units of deployment are example of modular design. Without modular design we may not realize the benefits you expect from your logical design.

Advantage of modular Design are-

- Design software that is extensible, reusable, maintainable and adaptable.
- Design Modular software today, in anticipation of future platform support for modularity.
 - Break large System into a flexible composite of collaborating modules.
 - Understand where to place your architectural focus.

Detailed Design

Once high-level design is done, you have a graphical representation of the structure of your softwaresystem a document that defines high-level details of each module in your system, including a module's interface (including input and output data types) notes on possible algorithms or data structures the module can use to meet its responsibilities a list of any non-functional requirements that might impact the module.

After high-level design, a designer's focus shifts to low-level design Each module's responsibilities should be specified as precisely as possible Constraints on the use of its interfaces should be specified pre and post conditions can be identified module-wide invariants can be specified internal data structures and algorithms can be suggested.

ER Diagram

ER diagram is an entity relationship diagram which shows a has a relationship or is a kind of relationship between the entities and attributes. The entity is represented by rectangular box and attributes by ellipse. The relationship between two entities is represented by diamond.

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases.

The components and features of an ER diagram:

ER Diagrams are composed of entities, relationships and attributes. They also depict cardinality, which defines relationships in terms of numbers. Here's a glossary:

Entity

A definable thing such as a person, object, concept or event—that can have data stored about it. Think of entities as nouns. Examples: a customer, student, car or product. Typically shown as a rectangle.

Relationship

How entities act upon each other or are associated with each other. Think of relationships as verbs. For example, the named student might register for a course. The two entities would be the student and the course, and the relationship depicted is the act of enrolling, connecting the two entities in that way. Relationships are typically shown as diamonds or labels directly on the connecting lines.

Attribute

A property or characteristic of an entity. Often shown as an oval or circle.

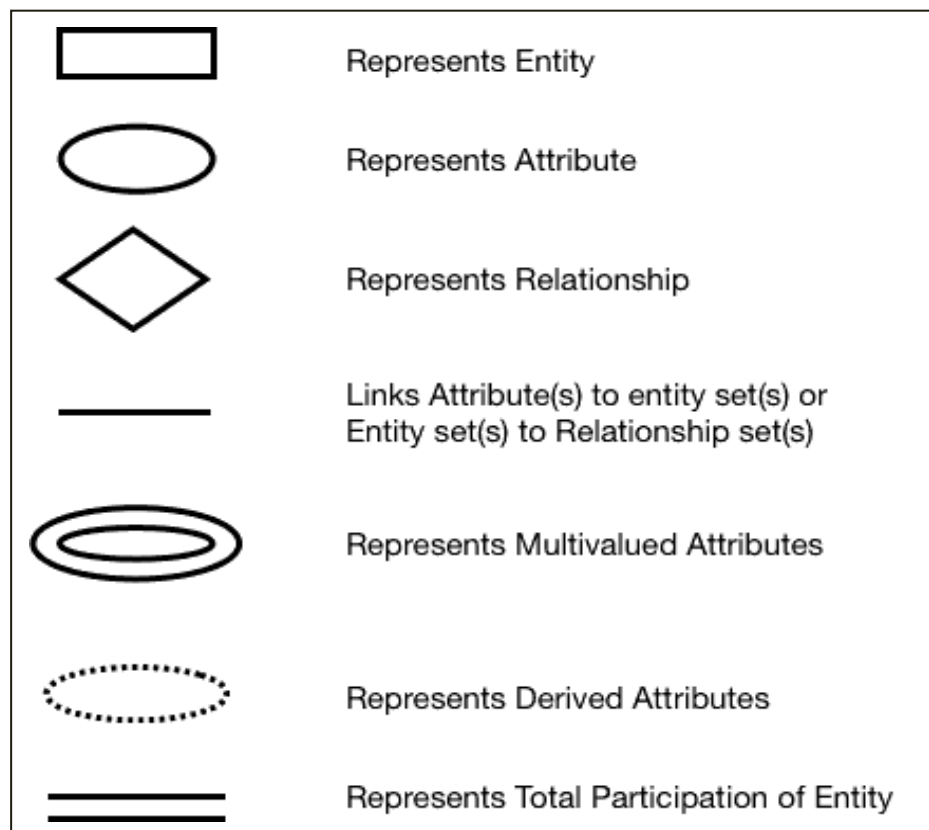


Figure 4.3.1: Components of ER Diagram

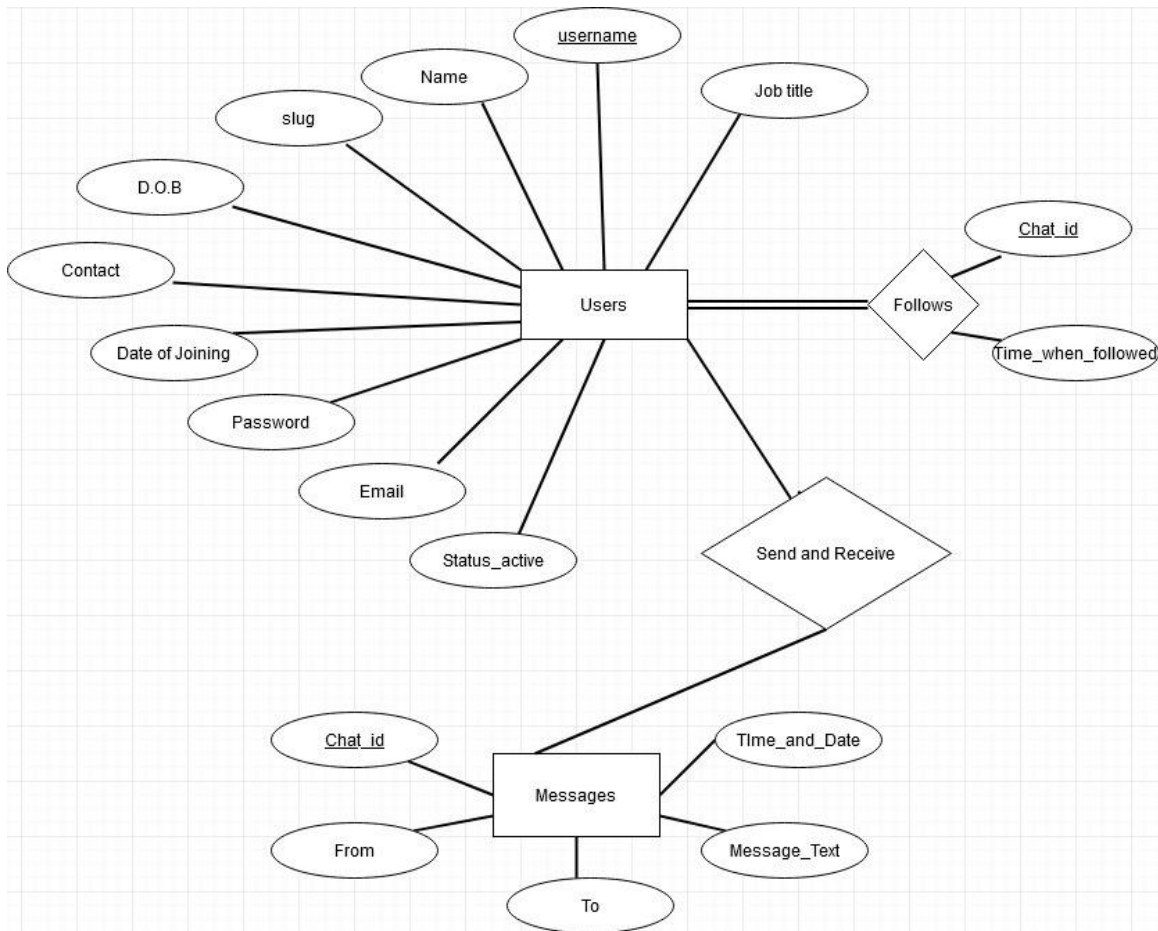


Figure 4.3.2: ER Diagram

Data Flow Diagram

A **data flow diagram (DFD)** is a graphical representation of the "**flow**" of **data** through an information system. A **DFD** is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated.

Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top-down approach to Systems Design. This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the detail of the system being modelled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.

The Various levels of refinements for our project is as follows.

Level 0 DFD

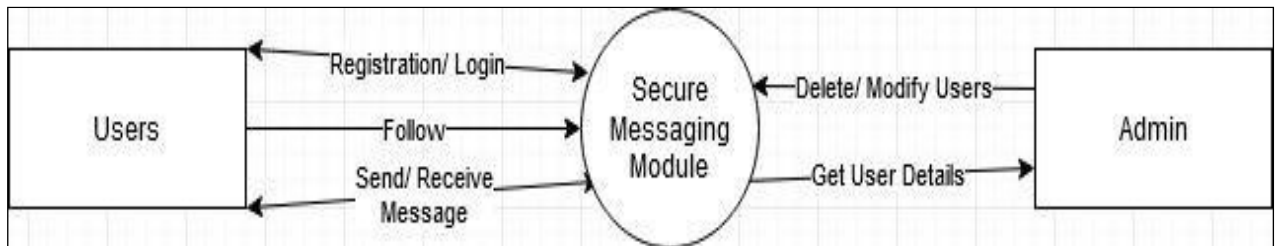


Figure 4.4.1: Level 0 DFD

Level 1 DFD

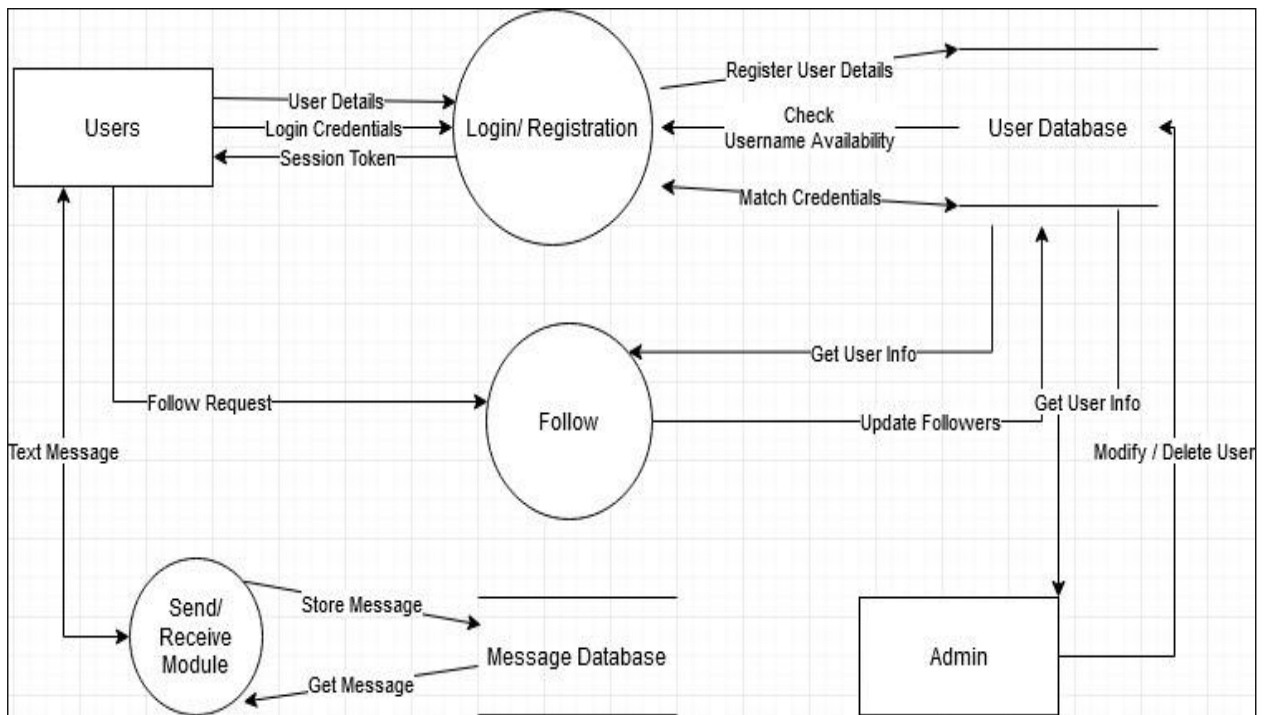


Figure 4.4.2: Level 1 DFD

Level 2 DFD

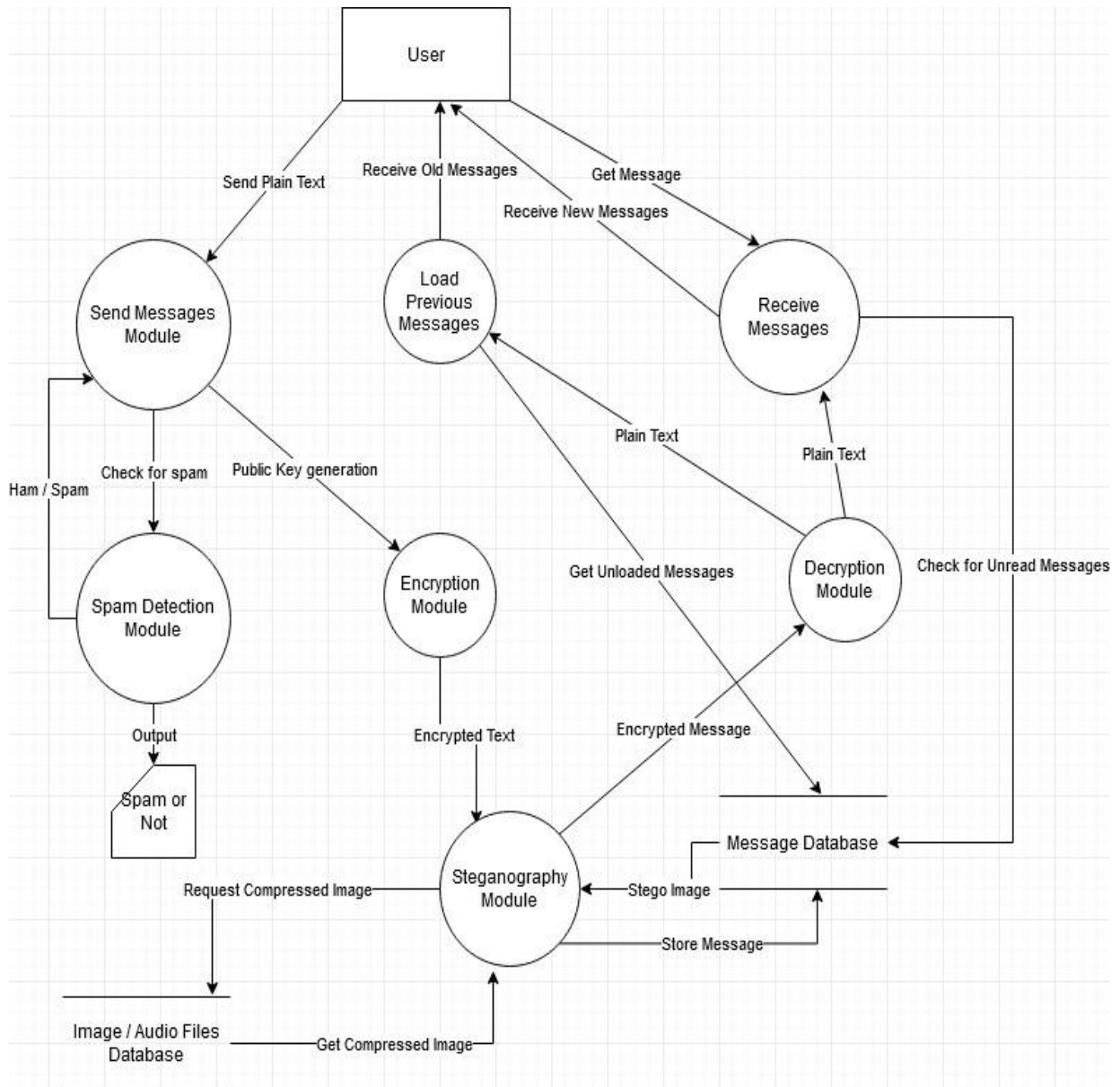


Figure 4.4.3: Level 2 DFD

Use Case Diagram

A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements.

A UML use case diagram is the primary form of system/software requirements for a new software program under development. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textually and visually (i.e. use case diagram). A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

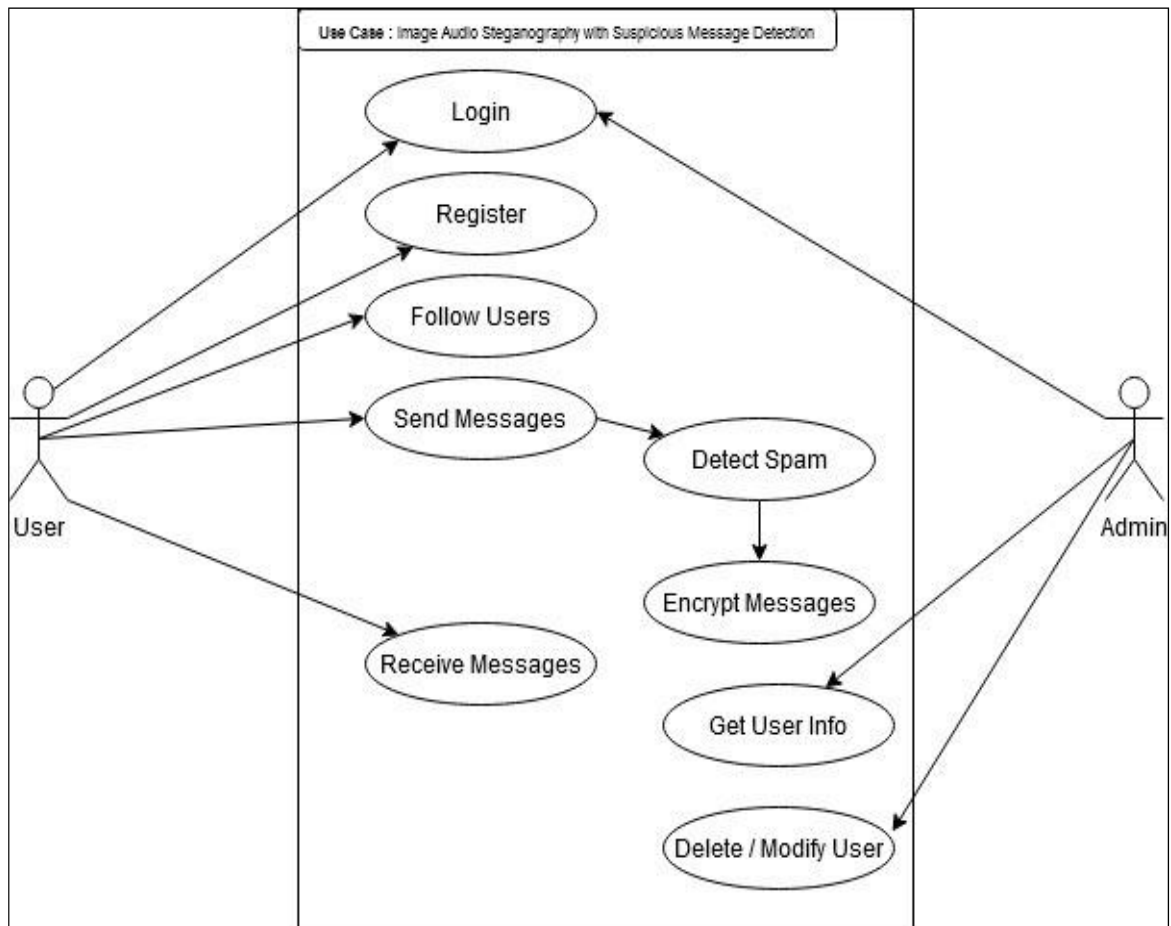


Figure 4.5.1 Use Case Diagram

CHAPTER 5

CODING AND IMPLEMENTATION

Implementation

Implementation is the realization of an application, or execution of a plan, idea, model, design, specification, standard, algorithms.

Implementation is the part of the process where software engineers actually program the code for the project.

Software and Hardware implementation should always be designed with the end user in mind and the implementation process usually benefits from user involvement.

If the users participate in the design and implementation of the system, ideally it will serve their business objectives more accurately and reflect their priorities and the ways in which they prefer to work.

Keeping in mind the need of modern lifestyle, the project is being implemented with the aim of providing a better lifestyle.

Here we have implemented an android application for the user to control the electrical appliances.

Java coding along with php and android has given a better environment. webhost servers contains all the tables and queries to be executed.

Hardware

As of in our project there is no such hardware requirement. It only requires a system with –

- Any 3G/4G internet network
- JavaScript supported browsers like Chrome, Firefox etc.
- OS- Windows, Linux, Mac, Android

IDE (Integrated Development Environment)

An Integrated Development Environment (IDE) is an application that facilitates application development. In general, an IDE is a graphical user interface (GUI)-based workbench designed to aid a developer in building software applications with an integrated environment combined with all the required tools at hand.

All of them have a text editor with syntax highlighter. It allows for code to be written and highlighted(key words, variables, strings, numbers, are in a different color so that they are easier to see and read).

All of them have a builder that translates the code we've written into the actual program.

Depending on the language they allow for real-time show of the result (for example web-pages). Some of them have the ability to see Code Coverage (how much of your code is actually used, really useful and interesting to see the results!).

Even a simple search for IDEs will turn up quite a few choices. IDEs are available from Open Source communities, vendors, and software companies. They range from free to pricing dependent upon the number of licenses required. There isn't a standard for IDEs and each has its own capabilities, along with strengths and weaknesses. Generally, an IDE provides an easy-to-use interface, automates development steps, and allows developers to run and debug programs all from one screen. It can also provide the link from a development operating system to an application target platform, like a desktop environment, smart phone or microprocessor.

Most common features, such as debugging, version control and data structure browsing, help a developer quickly execute actions without switching to other applications. Thus, it helps maximize productivity by providing similar user interfaces (UI) for related components and reduces the time taken to learn the language. An IDE supports single or multiple languages.

The concept of IDE evolved from simple command-based software which was not as useful as menu-driven software. Modern IDEs are mostly used in the context of visual programming, where applications are quickly created by moving programming building blocks or code nodes that generate flowchart and structure diagrams, which are compiled or interpreted.

Selecting a good IDE is based on factors, such as language support, operating system (OS) needs and costs associated with using the IDE etc.

A collection of utilities combined in to single desktop application that does the following-

- check the grammar i.e. syntax of the programming language
- code build- converting the program in to machine readable code i.e. executable code
- Easy maintenance of source code by facilitating dependency management & refactoring
 - documentation about the code
- IDE increases the productivity of the programmer a lot

: jQuery Code – Encrypting and Decrypting Messages

Let's write the code for encrypting the messages before sending it to our server-side systems and decrypting the text the we have received from the server.

To implement encryption and decryption we have used Advanced Encryption Standard (AES) Algorithm as described in previous section.

```

        $.ajaxSetup({ async: false
        });
        $(document).ready(function(){
        // Reference to the chat messages area
        // let $chatWindow = $("#messages");
        let $chatWindow = $(".chat-bubbles-body");

        // Our interface to the Chat service
        let chatClient;

        // A handle to the room's chat channel
        let roomChannel;

        let $chat_id; let $sender; let $receiver; let $chat_key;
        $chat_id = document.getElementById("chat_id").value;
        $sender = document.getElementById("sender").value ;
        $receiver = document.getElementById("receiver").value ;

        function sendmessagetoserver(msg){
            $.getJSON( "/receivemsg",
            {
            chat_id : $chat_id,From : $sender, To : $receiver, message: msg
            },
            function(data){ if(data.status==="success")
            {
            $received_msg = data.message_from_python;console.log($received_msg); printMessage($sender ,
            $received_msg);
            }
            }
            )
            }

        function receivemessgfromserver(){

            $.getJSON( "/sendmsg",
            {
            chat_id : $chat_id,From : $receiver, To : $sender,
            },

            function(data){

            if (data.status==="success")
            {
            for(i=0;i<data.length;i++)
            {
            printMessage( $receiver , data.messages[i]);
            }
            }
            }
            )
            }

```

```

        setTimeout(receivemessgefromserver , 2000);
    }

    )
}

function encrypt(msgString, key) {
// msgString is expected to be Utf8 encoded msgString = CryptoJS.enc.Utf8.parse(msgString);var iv =
CryptoJS.lib.WordArray.random(16);
var encrypted = CryptoJS.AES.encrypt(msgString, key, {
    iv: iv
});
return iv.concat(encrypted.ciphertext).toString(CryptoJS.enc.Base64);
}

function decrypt(ciphertextStr, key) {

var ciphertext = CryptoJS.enc.Base64.parse(ciphertextStr);

// split IV and ciphertext var iv = ciphertext.clone();iv.sigBytes = 16; iv.clamp();
ciphertext.words.splice(0, 4);
ciphertext.sigBytes -= 16;

// decryption
var decrypted = CryptoJS.AES.decrypt({ciphertext: ciphertext}, key, {
    iv: iv
});
console.log(decrypted);
return decrypted.toString(CryptoJS.enc.Utf8);
}

let $form = $("#message-form"); let $input = $("#message-input");let $send = $("#send");
$send.on("click", function(e) {
    e.preventDefault();
    let $messagetosend = $input.val().trim();if($messagetosend.length!=0){
        if(checkForSpam($messagetosend))
        {
            $input.val("");printSpam();return;
        }
    }
var $key = CryptoJS.enc.Utf8.parse($chat_key);let $encrypted = encrypt($messagetosend,$key);
console.log($encrypted); sendmessagetoserver($encrypted);
    $input.val("");
});
loadpreviousmessages(); receivemessgefromserver();

});

```


: HTML Code – Front end

Now let's code the front end of our website. Following codes are written using HTML and JINJA3 in order to work with Django.

Code for Login Snippet :

```
{% extends "base.html" %}
{% load staticfiles %}
{% block title %}Log-in{% endblock %}

{% block content %}
<div class="login">
<h1>Log-in</h1>
{% if notvalid %}

<p>
Your Username and password didn't match. Please try again.
</p>
{% elif notactive %}
<p>
Your account is not activated yet. Kindly check your email.
</p>
{% else %}
{% if loggedout %}
<p>Logged out successfully</p>
{% elif registered %}
<p>Registered Successfully!!</p>
{% endif %}
<p> Please Use the following form to login</p>

{% endif %}
<div class="login-form">

<div class="wrapper fadeInDown">
<div id="formContent">
<div class="fadeIn first">

</div>
<form action="{% url 'login' %}" method = "post">
{% csrf_token %}
<input type="text" id="login" class="fadeIn second" name="username" placeholder="login" required>
<input type="password" id="pass" class="fadeIn third" name="password" placeholder="password"
required>
<input type="submit" class="fadeIn fourth" value="Log In">
<p style="color:dodgerblue ">New User? <a href="{% url 'registration' %}">Register Here</a></p>
</form>
<div id="formFooter">
<a class="underlineHover" href="#">Forgot Password?</a>
</div>
</div>
</div>
</div>
</div>
{% endblock %}
```

Front end Code for Chat window:

```
{% load static %}
<!DOCTYPE html>
<html>
<head>
<title>{{room.name}} | TalkChat</title>
<link href='https://fonts.googleapis.com/css?family=Montserrat' rel='stylesheet'>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
<link rel="stylesheet" href="{% static 'chat/styles/chat.css' %}">
</head>
<body>
<input type="hidden" id="sender" name="variable" value="{{ sender }}">
<input type="hidden" id="receiver" name="variable" value="{{ receiver }}">
<input type="hidden" id="chat_id" name="variable" value="{{ chat_id }}">
<div class="top-bar">
<div class="back-button">
<a href="{% url 'friends' %}">
<button><i class="fa fa-arrow-left"></i></button>
</a>
</div>
<div class="name-and-status">
<div class="name">
{{receiver}}
</div>
<div class="status">
Online
</div>
<div class="chat-id">
Chat ID : {{chat_id}}
</div>
</div>
<br><br>
<div class="chat-body-outer">
<div class="chat-bubbles-body">
<!--The Bubble Body-->
</div>
</div>
<div class="chat-entry">
<form id="message-form">
<div class="input-box">
<input type="text" name="" class="chat-entry-input-box" id="message-input" placeholder="Type your
message here..... ">
</div>
<div class="send-button">
<button type="submit" id="send">
<i class="fa fa-paper-plane"></i>
</button>
</div>
</form>
</div>
```

```

<br>
<script src="https://code.jquery.com/jquery-3.2.1.min.js" crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/3.1.9-1/crypto-js.js"

```

```

crossorigin="anonymous"></script>
<script src="{% static 'chat/scripts/chat.js' %}"></script>
<script type="text/javascript">
    var input = document.getElementById("message-input");
    input.addEventListener("keyup",function(event){
        if(event.keyCode === 13){
            event.preventDefault();
            document.getElementById("send").click();
        }
    })
</script>
</body>
</html>

```

: Python Code – Backend of Messaging Service

Lets code the backend of our chat service which will interact with the existing frontend and it will integrate the spam detector for filtering out the spam messages from our chat box.

```

#!/usr/bin/python

# import framework utilities
from django.conf import settings
from django.http import JsonResponse
from django.shortcuts import render
from django.utils import timezone

# import data entities
from .models import Room , ReadMessages , UnreadMessages
from account.models import Followers,Users

from . import spam_detection
from Crypto.Cipher import AES

def receivevmsg(request):

    chat_id = Followers.objects.filter(chat_id = request.GET.get('chat_id'))
    try:
        if request.session['user'] is None
            or (request.GET.get('From') not in [chat_id[0].user1.username , chat_id[0].user2.username]):

                return render(request,'account/dashboard.html',{'section':'dashboard'})

    except KeyError:

        return render(request,'account/dashboard.html',{'section':'dashboard'})

    From = Users.objects.filter(username = request.GET.get('From'))
    To = Users.objects.filter(username = request.GET.get('To'))
    message = request.GET.get('message')

    row = UnreadMessages(chat_id = chat_id[0] , From = From[0] , To = To[0] , message = message)
    row.save()
    response = {'status':"success" , "message_from_python":message,"time" : str(timezone.now)}
    return JsonResponse(response)

```

```

def room_detail(request, slug):
    room = Room.objects.get(slug=slug)

    user = Followers.objects.get(chat_id = slug)

    if request.session['user'] in [user.user1.username , user.user2.username]:
    receiver = user.user1.username if request.session['user']!=user.user1.username else user.user2.username
    data = {'room': room,'sender':request.session['user'],'receiver':receiver , 'chat_id':slug}

    return render(request, 'chat/chat.html', data)else:

    return JsonResponse({"access":"denied"})


def sendmsg(request):

    chat_id = Followers.objects.filter(chat_id = request.GET.get('chat_id'))

    From = Users.objects.filter(username = request.GET.get('From'))To = Users.objects.filter(username =
    request.GET.get('To'))
    messages = UnreadMessages.objects.filter(chat_id = chat_id[0] , From = From[0] , To = To[0])response =
    {
        'status':'success',
        "messages" : [],
        "length" : 0
    }

    for msg in messages: response["messages"].append(msg.message)
    ReadMessages(chat_id = msg.chat_id , From = msg.From , To = msg.To , message = msg.message).save()

    response["length"] = response["messages"]._len()messages.delete()
    return JsonResponse(response)


def readmsg(request):

    chat_id = Followers.objects.filter(chat_id = request.GET.get('chat_id'))
    messages = ReadMessages.objects.filter(chat_id = chat_id[0]).order_by('time_date')messages =
    messages[max(0 , len(messages)-30):]
    response = {
        'status':'success',
        "messages" : [],
        "length" : 0
    }

    for msg in messages:
    response["messages"].append({"from":msg.From.username , "message":msg.message})
    response["length"] = response["messages"]._len()return JsonResponse(response)


def checkspam(request):

    message = request.GET.get('message') status = (spam_detection.is_spam(message))return
    JsonResponse({'status':status})

```

Machine learning Model

This web application supports the feature of spam detection in which a spam message is being prevented to send it to the user(receiver).

To achieve this feature, we have used the machine learning model.

Working:

- The sender types the message to send it to the receiver.
- Before sending the message, the model is used to detect the spam in the message
 - This is achieved by the algorithm known as 'Logistic Regression'.
- With the help of this algorithm we find out if the message is spam or not. If the message is spam then it is blocked right away otherwise the message is sent to the receiver.

Algorithm used:

The algorithm used is, 'Logistic Regression'. The Logistic Regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous. Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

Since this algorithm figures out whether a message is spam or not (the dependent variable has only two possible types either 1 and 0), therefore, the type of logistic regression that we are using is known as 'Binary Logistic Regression'.

Instead of predicting exactly 0 or 1, logistic regression generates a probability—a value between 0 and 1, exclusive. For this logistic regression model for spam detection, if the model infers a value of 0.932 on a particular email message, it implies a 93.2% probability that the email message is spam. More precisely, it means that in the limit of infinite training examples, the set of examples for which the model predicts 0.932 will actually be spam 93.2% of the time and the remaining 6.8% will not.

Dataset used:

The dataset used for testing and training have more than 5000 training examples. There are two columns in the dataset file. The first column is of text where all the training examples are given and second column is of spam where the value '0' means that the text is not spam and '1' means that the text is spam.

	A	B
1	text	spam
2	Subject: naturally irresistible your corporate identity It is really hard to recollect a company : the marke	1
3	Subject: the stock trading gunslinger fanny is merrill but muzo not colza attainer and penultimate like	1
4	Subject: unbelievable new homes made easy im wanting to show you this homeowner you have been	1
5	Subject: 4 color printing special request additional information now ! click here click here for a printabl	1
6	Subject: do not have money , get software cds from here ! software compatibility . . . ain ' t it great ? g	1
7	Subject: great nnews hello , welcome to medzonline sh groundsel op we are pleased to introduce ours	1
8	Subject: here ' s a hot play in motion homeland security investments the terror attacks on the united st	1
9	Subject: save your money buy getting this thing here you have not tried cialls yet ? than you cannot ev	1
10	Subject: undeliverable : home based business for grownups your message subject : home based busin	1
11	Subject: save your money buy getting this thing here you have not tried cialls yet ? than you cannot ev	1
12	Subject: las vegas high rise boom las vegas is fast becoming a major metropolitan city ! 60 + new high ri	1
13	Subject: save your money buy getting this thing here you have not tried cialls yet ? than you cannot ev	1
14	Subject: brighten those teeth get your teeth bright white now ! have you considered professional tee	1
15	Subject: wall street phenomenon reaps rewards small - cap stock finder new developments expected t	1
16	Subject: fpa notice : ebay misrepresentation of identity - user suspension - section 9 - dear ebay memb	1
17	Subject: search engine position be the very first listing in the top search engines immediately . our con	1
18	Subject: only our software is guaranteed 100 % legal . name - brand software at low , low , low , low pri	1
19	Subject: localized software , all languages available . hello , we would like to offer localized software v	1
20	Subject: security alert - confirm your national credit union information - - >	1
21	Subject: 21 st century web specialists jrghm dear it professionals , have a problem or idea you need a s	1
22	Subject: any med for your girl to be happy ! your girl is unsatisfied with your potency ? don ' t wait until	1
23	Subject: re : wearable electronics hi my name is jason , i recently visited www . clothingplus . fi / and w	1
24	Subject: top - level logo and business identity corporate image can say a lot of things about your compa	1
25	Subject: your trusted source for prescription medication . best prescription generic meds 4 less . anger	1

Implementation:

```

import numpy as np
import pandas as pd
import nltk
from nltk.corpus import stopwords
import string

```

Here we have imported various libraries:

- numpy for numerical computation
 - pandas for file handling
- nltk for natural language processing
 - string for string manipulation

Uploading data set:

```

from google.colab import files # Use to load data on Google Colab
uploaded = files.upload() # Use to load data on Google Colab

```

Choose Files dataset.csv

• dataset.csv(application/vnd.ms-excel) - 8937260 bytes, last modified: 6/10/2020 - 100% done
Saving dataset.csv to dataset.csv

Here we have uploaded the data set to train the model.

Putting dataset to the Pandas framework and removing null values:

```
import io

df = pd.read_csv(io.BytesIO(uploaded['dataset.csv']))

df.head(5)

df.shape

df.columns

df.drop_duplicates(inplace = True)

df.shape

df=pd.DataFrame(df).dropna()

df.isnull().sum()

nltk.download('stopwords')
```

We have stored the dataset file on the pandas frameworks, we have also downloaded the stopword. We have cleaned the data by removing the rows having missing values.

Cleaning:

```
def process_text(text):

    nopunc = [char for char in text if char not in string.punctuation]
    nopunc = ''.join(nopunc)

    clean_words = [word for word in nopunc.split() if word.lower() not in stopwords.words('english')]

    return clean_words

df['text'].head().apply(process_text)
```

```
0    [Subject, naturally, irresistible, corporate, ...
1    [Subject, stock, trading, gunslinger, fanny, m...
2    [Subject, unbelievable, new, homes, made, easy...
3    [Subject, 4, color, printing, special, request...
4    [Subject, money, get, software, cds, software,...
Name: text, dtype: object
```

We have defined a function 'process_text()' to get clean words in lowercase.

Training the Model:

```

from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(analyzer=process_text)
messages_bow = vectorizer.fit_transform(df['text'])

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(messages_bow, df['spam'], test_size = 0.20, random_state = 0)

messages_bow.shape

from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression().fit(X_train,y_train)

print(classifier.predict(X_train))
print(y_train.values)

```

Here, we have vectorized the text we got using count_vectorizer and then created a vector space model. Then we have split it to train 80% of the data that we got from vector space model and the remaining 20% data is used for the testing purpose. After splitting the data we used logistic regression algorithm to train the model and later tested the model.

Report:

```

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
pred = classifier.predict(X_train)
print(classification_report(y_train ,pred ))
print('Confusion Matrix: \n',confusion_matrix(y_train,pred))
print()
print('Accuracy: ', accuracy_score(y_train,pred))

print('Predicted value: ',classifier.predict(X_test))
print('Actual value: ',y_test.values)

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
pred = classifier.predict(X_test)
print(classification_report(y_test ,pred ))

print('Confusion Matrix: \n', confusion_matrix(y_test,pred))
print()
print('Accuracy: ', accuracy_score(y_test,pred))

```

We also went for report of the model we trained.

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	3456
1.0	1.00	1.00	1.00	1098
accuracy			1.00	4554
macro avg	1.00	1.00	1.00	4554
weighted avg	1.00	1.00	1.00	4554
Confusion Matrix:				
[[3456 0]				
[0 1098]]				
Accuracy: 1.0				
Predicted value: [0. 1. 0. ... 0. 1. 1.]				
Actual value: [0. 1. 0. ... 0. 1. 1.]				
	precision	recall	f1-score	support
0.0	0.99	1.00	0.99	869
1.0	0.99	0.96	0.98	270
accuracy			0.99	1139
macro avg	0.99	0.98	0.98	1139
weighted avg	0.99	0.99	0.99	1139
Confusion Matrix:				
[[866 3]				
[10 260]]				
Accuracy: 0.9885864793678666				

The achieved accuracy is 0.98 out of 1.0

Output of the model:

```

text = input("Enter the input data ")

mb = vectorizer.transform([text])
# mb
isspam = classifier.predict(mb)[0]

if isspam==1:
    print("This message is spam")
else:
    print("This message is not spam")

```

Enter the input data Are you broke? buy this software and earn money
This message is spam

Steganography:

Steganography is the technique of hiding secret data within an ordinary, non-secret, file or message in order to avoid detection; the secret data is then extracted at its destination. The use of steganography can be combined with encryption as an extra step for hiding or protecting data. We have used the PNG image files and WAV audio files as a cover to hide the data.

Algorithm used: LSB Algorithm (Naive): In this algorithm, the least significant bit of the pixel is changed to hide the information. Here is its implementation.

```

from PIL import Image
import string
import random
dic= string.ascii_letters+string.punctuation+string.digits
print(dic)
def generate_message(n):
    s = ""
    for i in range(n):
        s+=dic[random.randint(0 , 1000)%len(dic)]
    return s
img = Image.open("0.png")
img.show()
def listoflist(l):
    op=[]
    for i in l:
        op.append(list(i))
    return op
def preprocessimage(listofpixels):
    for i in range(len(listofpixels)):
        for j in range(3):
            listofpixels[i][j]&=254
listofpixels = list(img.getdata())
#print(listofpixels)
listofpixels = listoflist(listofpixels)
print(listofpixels[:10])
preprocessimage(listofpixels)
print(listofpixels[:10])
message = generate_message(1000)
listofbytes = [ord(i) for i in message]
lis = []
for i in listofbytes:
    while i:
        lis.append(i&254)
        i>>=1
listofbytes = lis[:]
def hide(listofpixels , listofbytes):
    key=[]
    k=0
    for i in range(len(listofpixels)):
        for j in range(3):
            try:
                listofpixels[i][j]|=listofbytes[k]
                k+=1
            except IndexError:
                break
hide(listofpixels,listofbytes)
print(listofpixels[:10])
def listoftuple(l):
    op=[]
    for i in l:
        op.append(tuple(i))
    return op
new_image = Image.new(img.mode , img.size)
listofpixels = listoftuple(listofpixels)
new_image.putdata(listofpixels)
new_image.show()
new_image.save("0lsb.png")

```

Input Image:

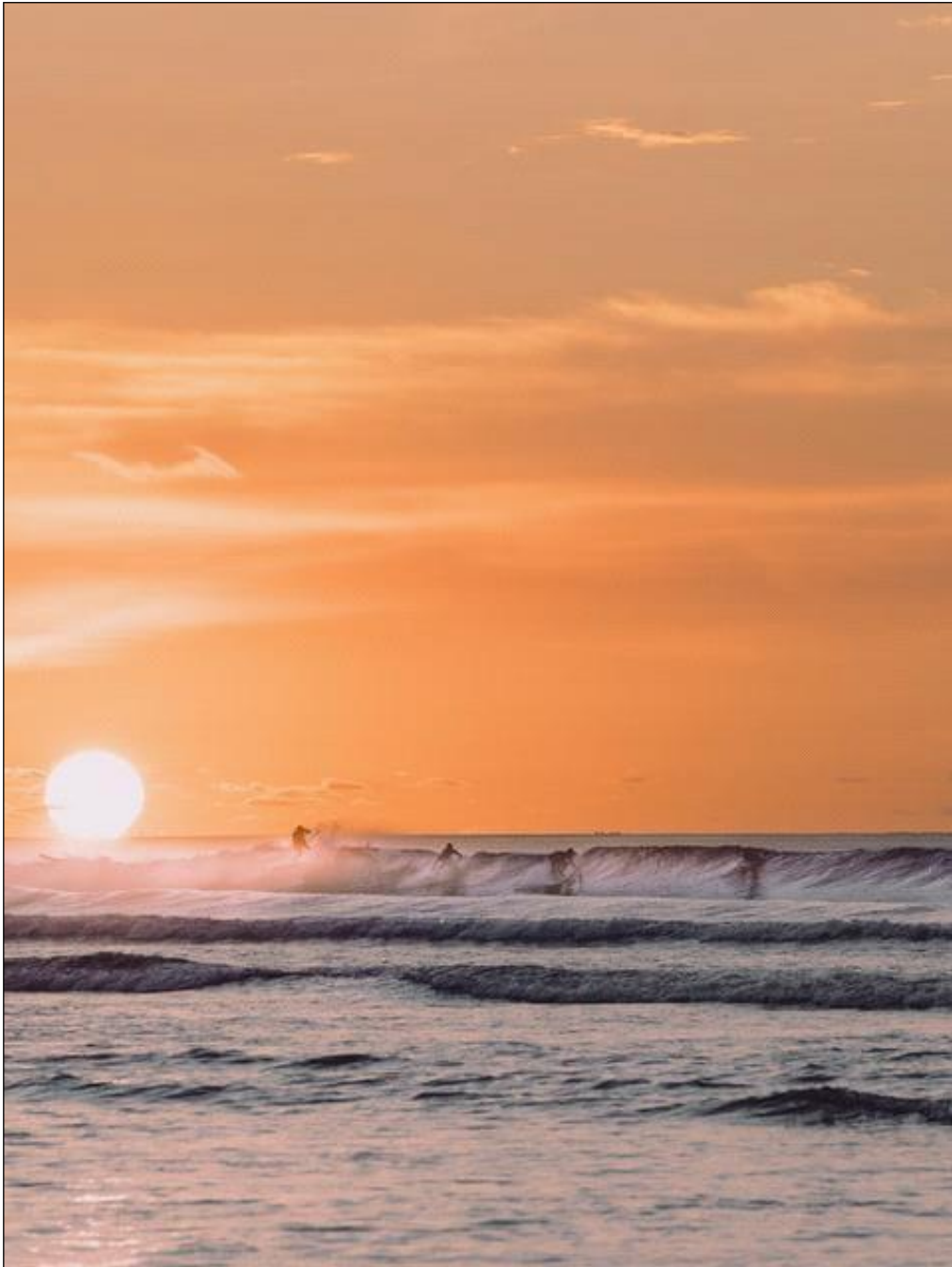


Fig.5.4.6.1 LSB Input

Output Image:

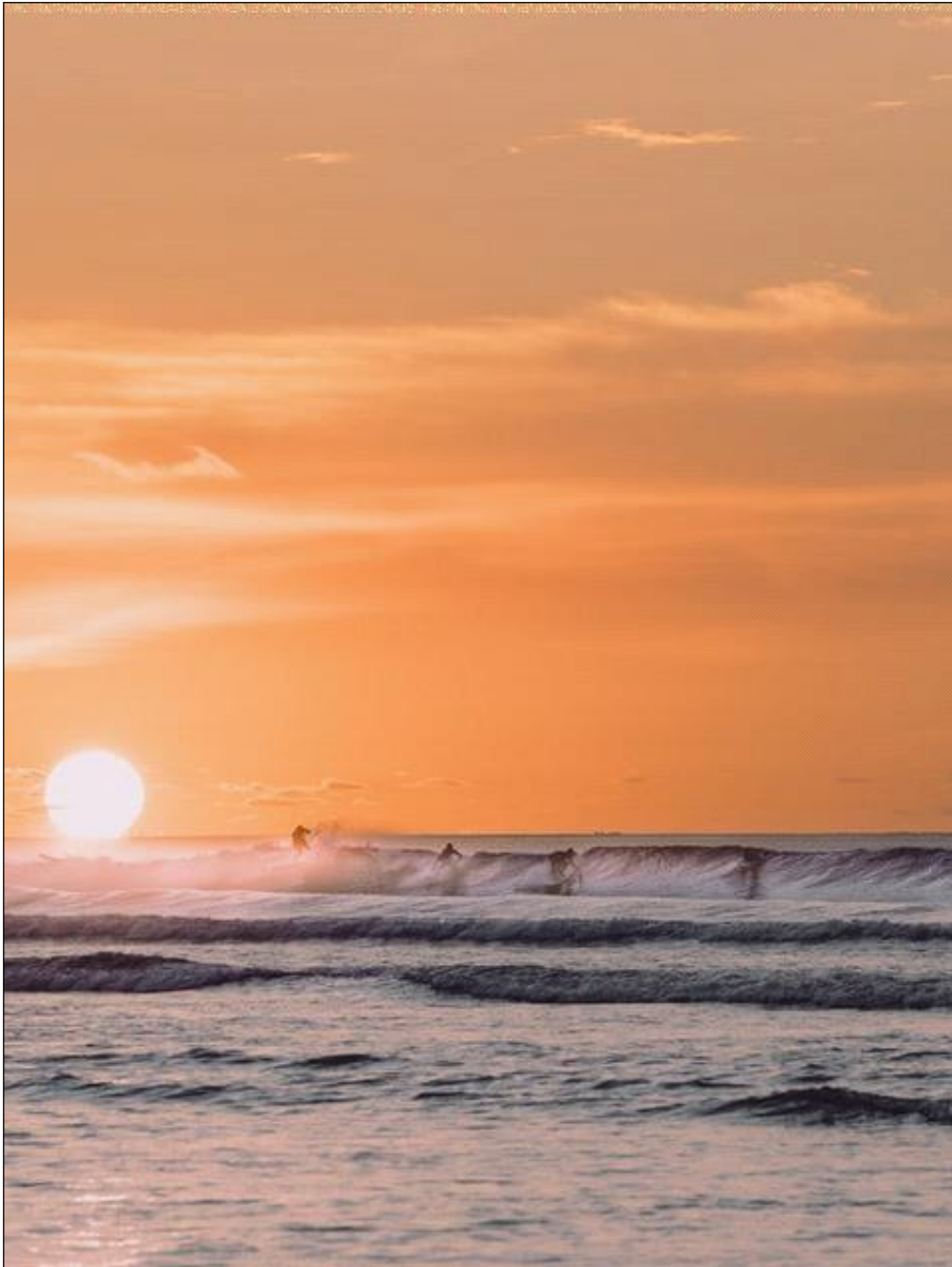


Fig.5.4.6.2 LSB Output

This algorithm works fine but sometimes it also distorts the image to a noticeable extent and to overcome this flaw, we have used another algorithm known as 'Battle Steg Algorithm'.

Battle Steg Algorithm:

In this algorithm, instead of amending the least significant bit of the pixel, we now can amend any bit of a pixel. Here is its implementation:


```

from PIL import Imageimport string
import randomimport pickle
dic= string.ascii_letters+string.punctuation+string.digitsprint(dic)
def generate_message(n):s = ""
    for i in range(n):
s+=dic[random.randint(0 , 1000)%len(dic)]return s
img = Image.open("2.png")img.show()
def listoflist(l):op=[]
    for i in l:
        op.append(list(i))return op
def preprocessimage(listofpixels): for i in range(len(listofpixels)):
    for j in range(3): listofpixels[i][j]&=254
    listofpixels = list(img.getdata())#print(listofpixels)
    listofpixels = listoflist(listofpixels)print(listofpixels[:10])
    preprocessimage(listofpixels) print(listofpixels[:10])
message = generate_message(25) print("message generated is : ",message)listofbytes =
    [ord(i) for i in message]lis = []
    print(listofbytes) for i in listofbytes:
for k in range(8):lis.append(i%2)i//=2
    listofbytes = lis[:]print(listofbytes)
    def next_position(key,n):
pos = (random.randint(0, n-1) , random.randint(0,2)) return pos if pos not in key else
        next_position(key,n)
    def hide(listofpixels , listofbytes):key=[]
        n = len(listofpixels)
for k in range(len(listofbytes)):i,j = next_position(key , n)key.append((i,j))
        listofpixels[i][j]^=listofbytes[k]return key
    key = hide(listofpixels,listofbytes)print(listofpixels[:10])
    def listoftuple(l):op=[]
        for i in l:
            op.append(tuple(i))return op
    new_image = Image.new(img.mode , img.size)

```

```

listofpixels = listoftuple(listofpixels)new_image.putdata(listofpixels)
new_image.show() new_image.save("0bs.png")
f = open("key.pickle","wb")pickle.dump(key,f) new_image=None
new_image = Image.open("0bs.png")msg=[]
list_of_pixels_in_new_image = list(new_image.getdata())for i,j in key:
msg.append(str(list_of_pixels_in_new_image[i][j]&1))print(msg)
msg = "".join(msg)
msg = msg[::-1]ans=[] print(msg)
for i in range(0,len(msg),8): ans.append(int(msg[i:i+8],2))
for i in range(len(ans)):ans[i] = chr(ans[i])
msg = "".join(ans)
print(msg[::-1])

```

Input Image:



Fig. 5.4.6.3 Battlesteg Input

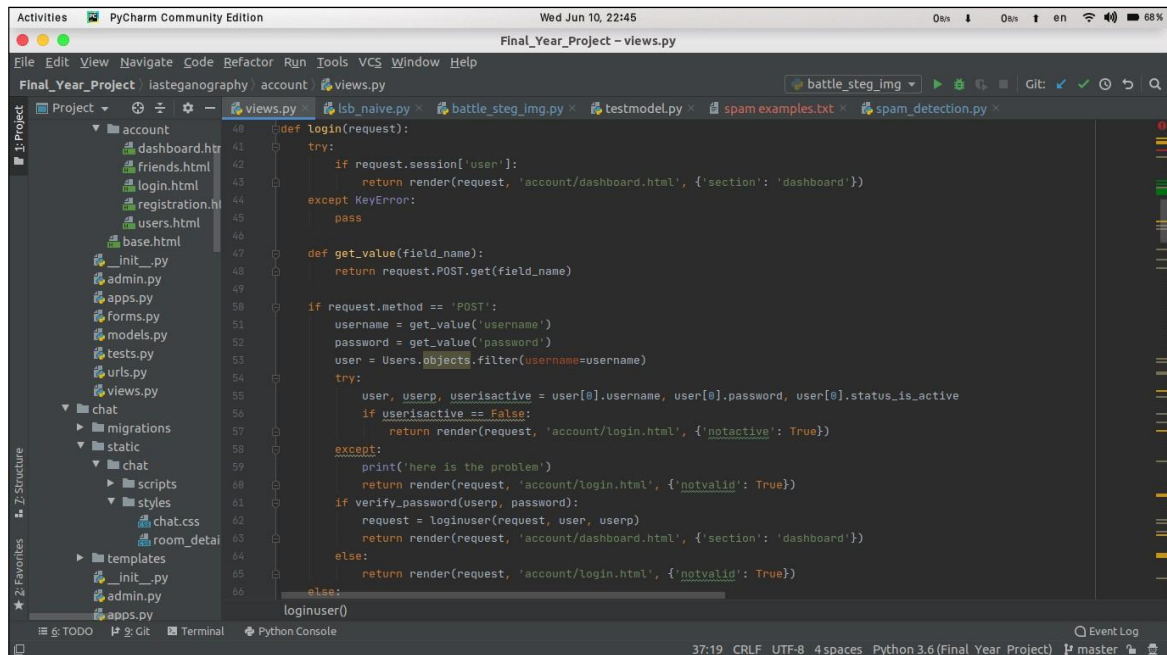
Output Image:



Fig. 5.4.6.4 Battlesteg Output

By using this algorithm, the amendments are not noticeable.

Snapshot



The screenshot shows the PyCharm IDE with the 'Final_Year_Project' open. The file explorer on the left shows the project structure, including 'account' and 'views.py'. The main editor displays the 'login' function in 'views.py'. The function handles a POST request for login, checks if the user is active, and returns the appropriate template based on the login status.

```
def login(request):
    try:
        if request.session['user']:
            return render(request, 'account/dashboard.html', {'section': 'dashboard'})
    except KeyError:
        pass

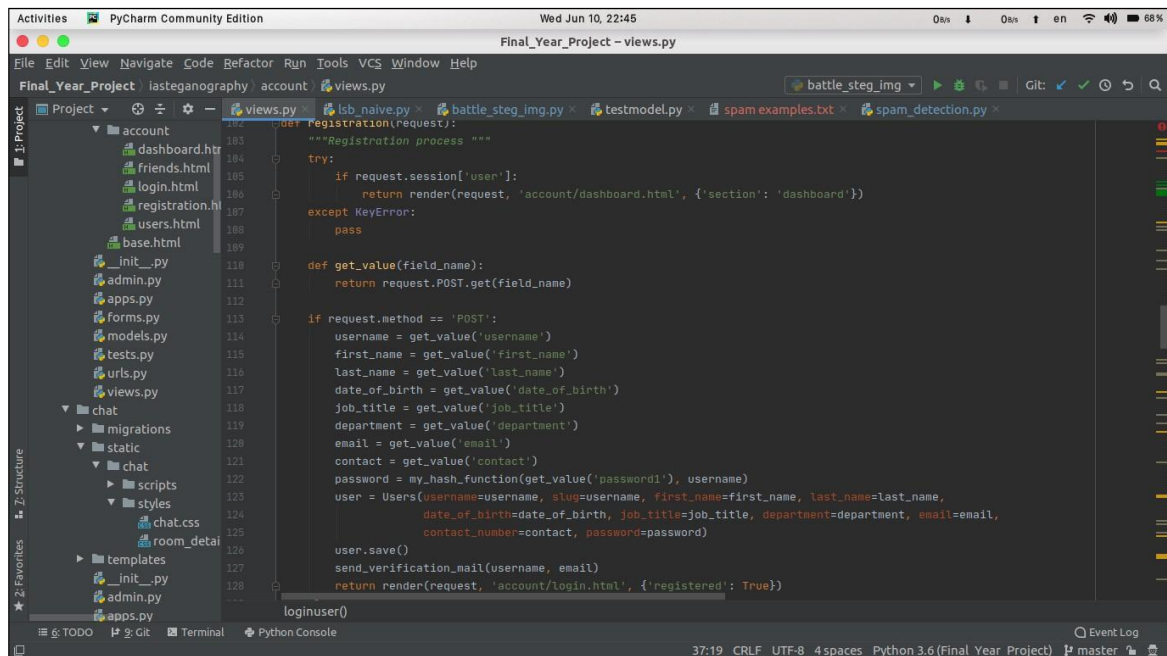
    def get_value(field_name):
        return request.POST.get(field_name)

    if request.method == 'POST':
        username = get_value('username')
        password = get_value('password')
        user = Users.objects.filter(username=username)
        try:
            user, userispactive = user[0].username, user[0].password, user[0].status_is_active
            if userispactive == False:
                return render(request, 'account/login.html', {'notactive': True})
        except:
            print('here is the problem')
            return render(request, 'account/login.html', {'notvalid': True})
        if verify_password(userispactive, password):
            request = loginuser(request, user, userispactive)
            return render(request, 'account/dashboard.html', {'section': 'dashboard'})
        else:
            return render(request, 'account/login.html', {'notvalid': True})

    return render(request, 'account/login.html', {'notvalid': True})

loginuser()
```

Figure: 5.5.1: Login Activity Code



The screenshot shows the PyCharm IDE with the 'Final_Year_Project' open. The file explorer on the left shows the project structure, including 'account' and 'views.py'. The main editor displays the 'registration' function in 'views.py'. The function handles a POST request for registration, collects user details, and saves the user to the database.

```
def registration(request):
    """Registration process """
    try:
        if request.session['user']:
            return render(request, 'account/dashboard.html', {'section': 'dashboard'})
    except KeyError:
        pass

    def get_value(field_name):
        return request.POST.get(field_name)

    if request.method == 'POST':
        username = get_value('username')
        first_name = get_value('first_name')
        last_name = get_value('last_name')
        date_of_birth = get_value('date_of_birth')
        job_title = get_value('job_title')
        department = get_value('department')
        email = get_value('email')
        contact = get_value('contact')
        password = my_hash_function(get_value('password1'), username)
        user = Users(username=username, slog=username, first_name=first_name, last_name=last_name,
                     date_of_birth=date_of_birth, job_title=job_title, department=department, email=email,
                     contact_number=contact, password=password)
        user.save()
        send_verification_mail(username, email)
        return render(request, 'account/login.html', {'registered': True})

    return render(request, 'account/login.html', {'notvalid': True})

loginuser()
```

Figure:5.5.2: Registration Activity Code

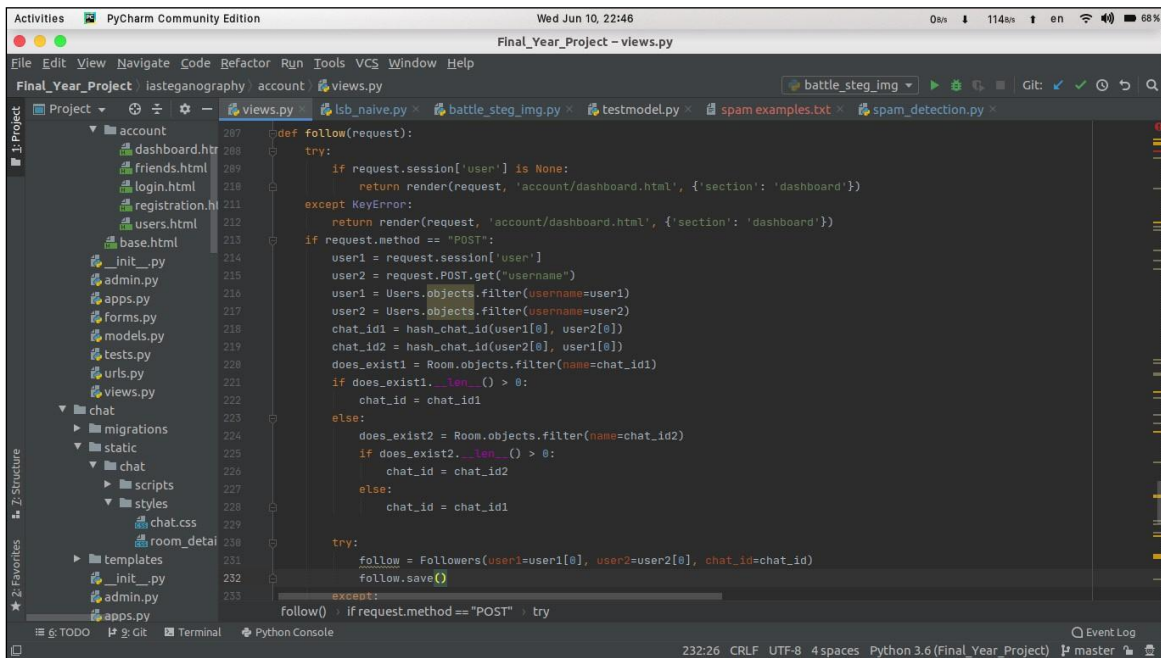


Figure 5.5.3 Follow Activity Code

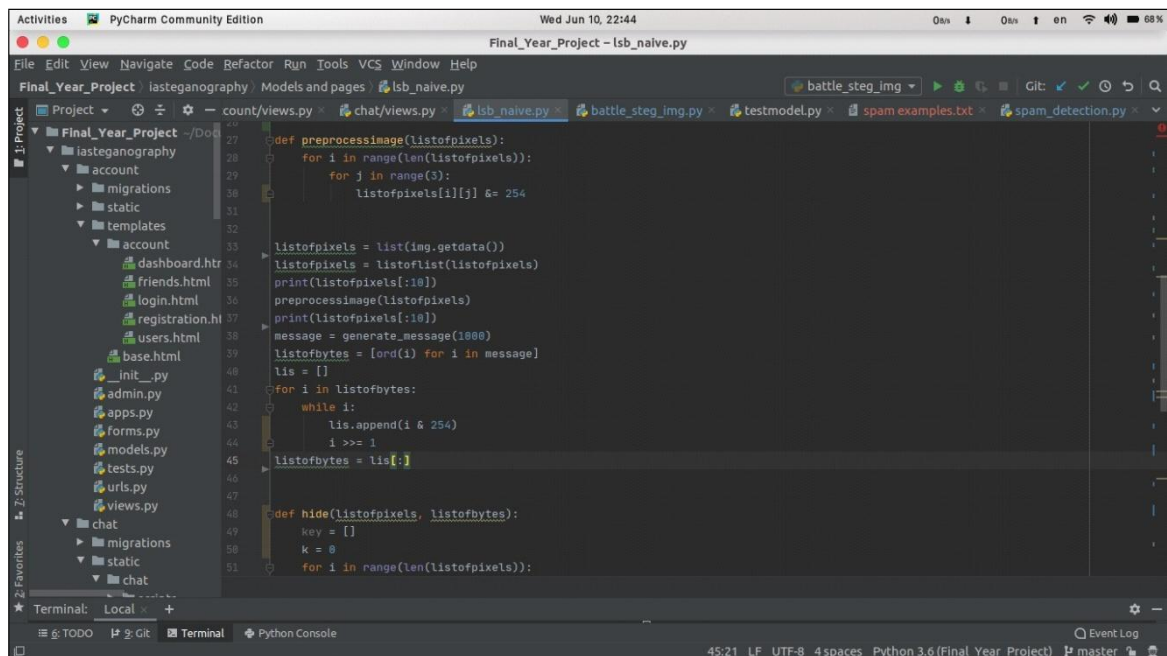


Figure 5.5.4 LSB Steganography Code

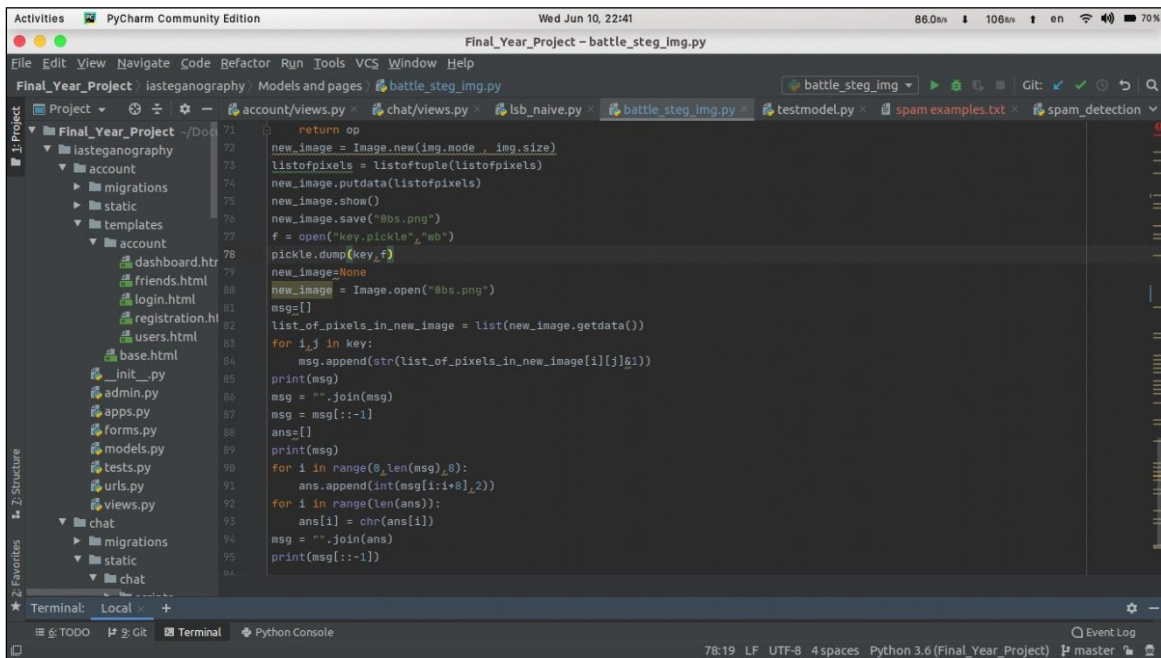
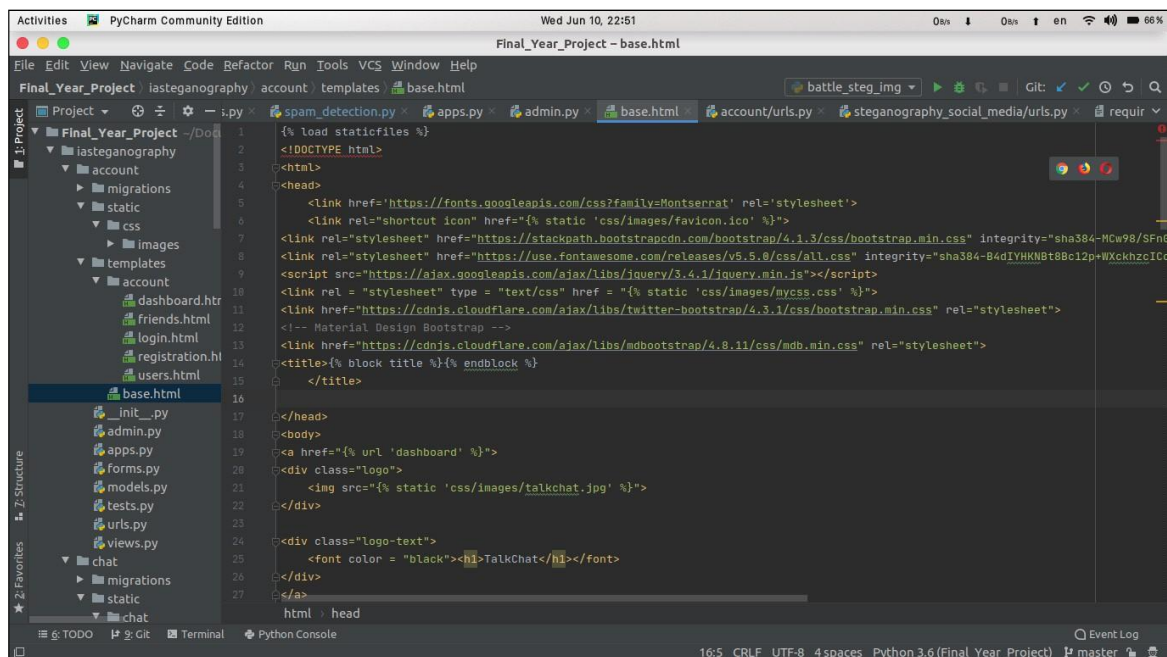


Figure 5.5.5 Battle-Steg Algorithm Code




```

nltk.download('stopwords')

def process_text(text):
    nopunc = [char for char in text if char not in string.punctuation]
    nopunc = ''.join(nopunc)

    clean_words = [word for word in nopunc.split() if word.lower() not in stopwords.words('english')]

    return clean_words

df['text'].head().apply(process_text)

from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(analyzer=process_text)
messages_bow = vectorizer.fit_transform(df['text'])

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(messages_bow, df['spam'], test_size=.20, random_state=0)

messages_bow.shape

from sklearn import tree
classifier = tree.DecisionTreeClassifier().fit(X_train, y_train)

```

```

print(classifier.predict(X_train))
print(y_train.values)

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
pred = classifier.predict(X_train)
print(classification_report(y_train, pred))
print('Confusion Matrix: \n', confusion_matrix(y_train, pred))
print()
print('Accuracy: ', accuracy_score(y_train, pred))

print('Predicted value: ', classifier.predict(X_test))
print('Actual value: ', y_test.values)

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
pred = classifier.predict(X_test)
print(classification_report(y_test, pred))

print('Confusion Matrix: \n', confusion_matrix(y_test, pred))
print()
print('Accuracy: ', accuracy_score(y_test, pred))

text = input("Enter the input data ")
mb = vectorizer.transform([text])
# mb
isspam = classifier.predict(mb)[0]

```

Figure 5.5.8 Decision Tree Implementation

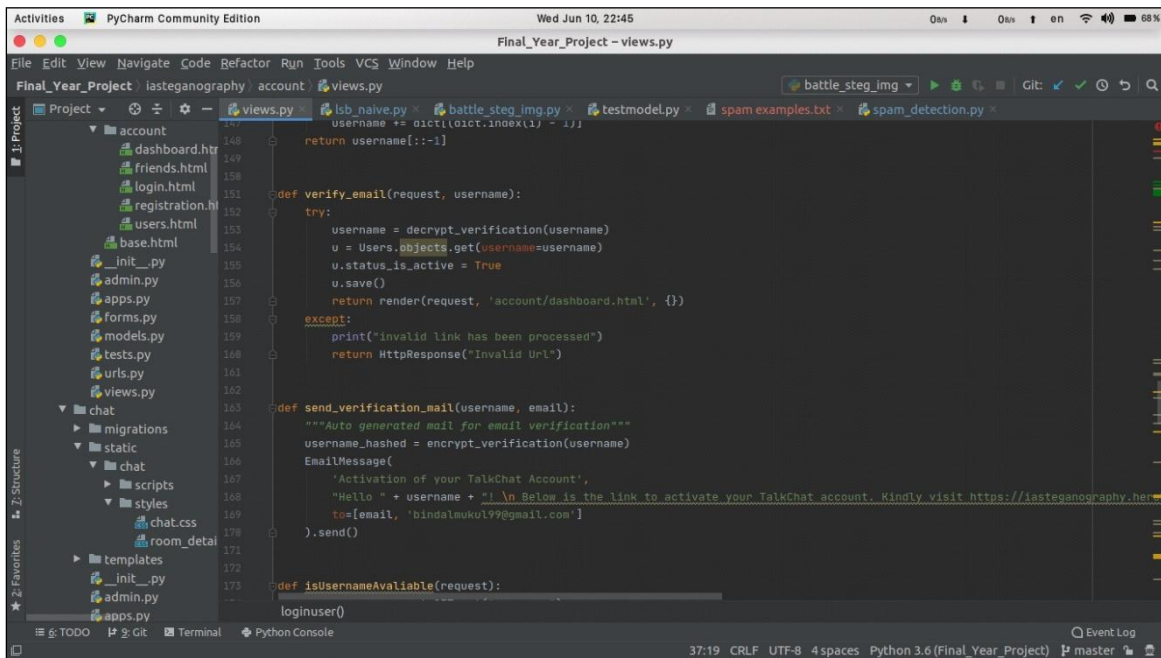


Figure 5.5.9 Email Verification

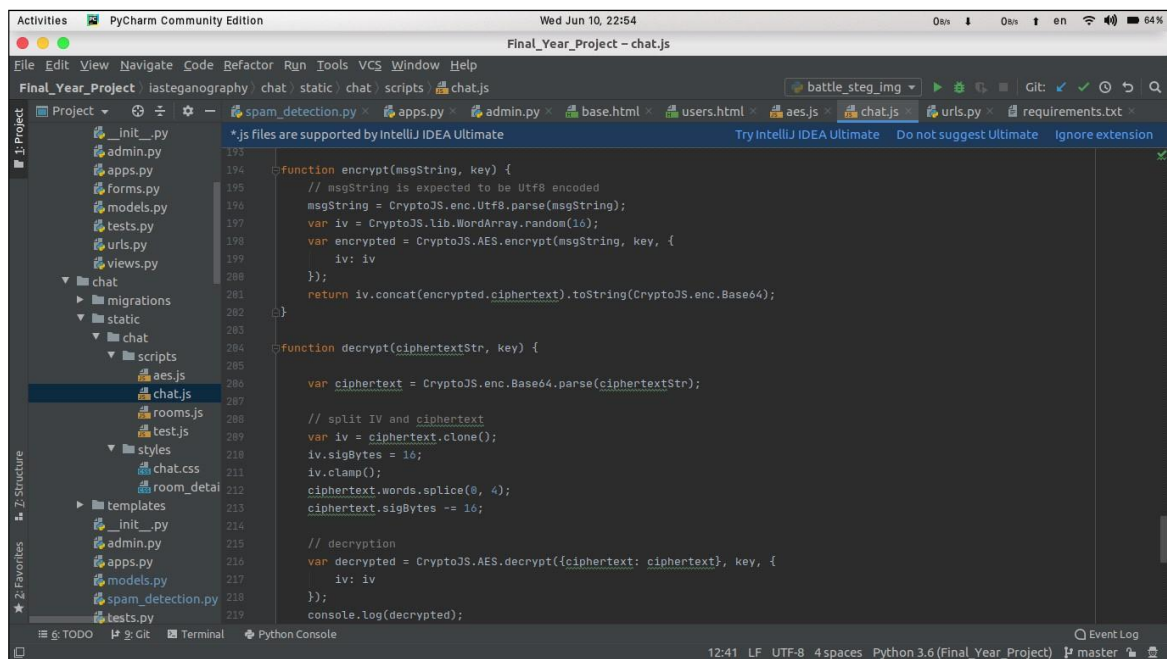


Figure 5.5.10 Encryption and Decryption at client side

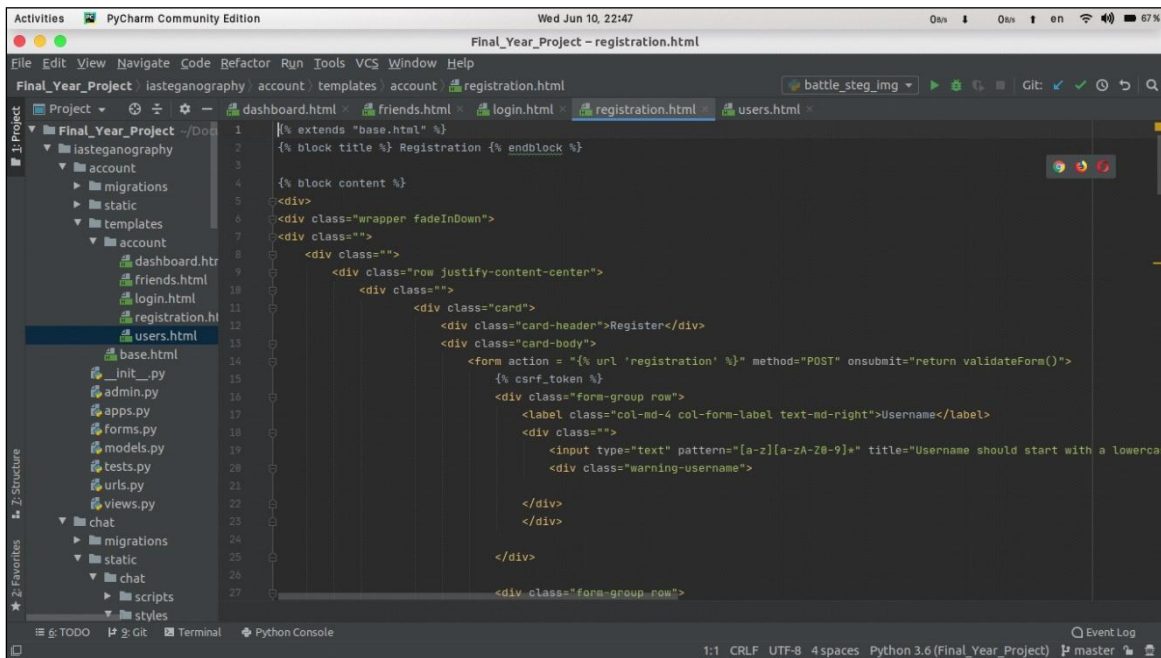


Figure 5.5.11 Front end Registration code

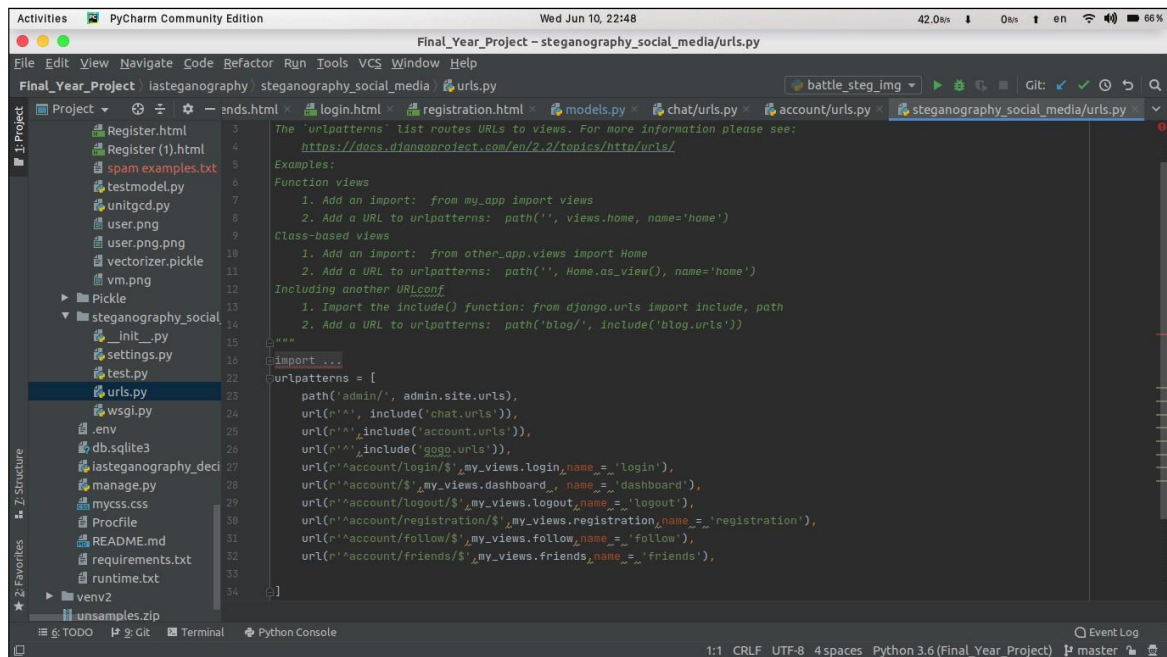


Figure 5.5.12 URL Mapping

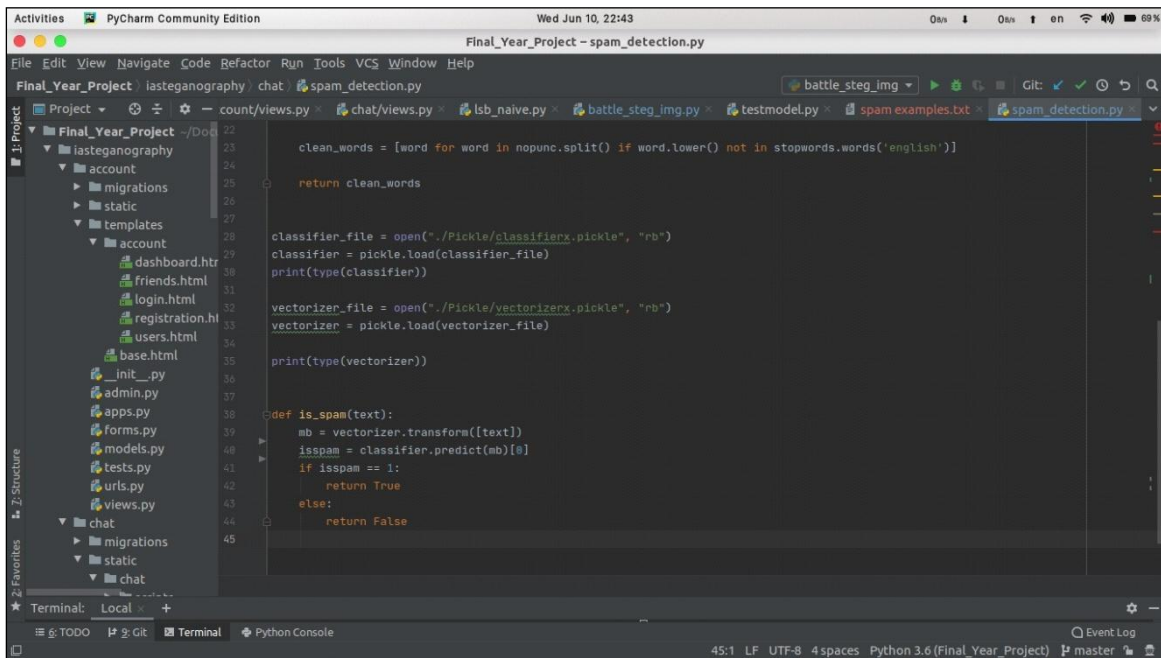


Figure 5.5.13 Spam Detection Code

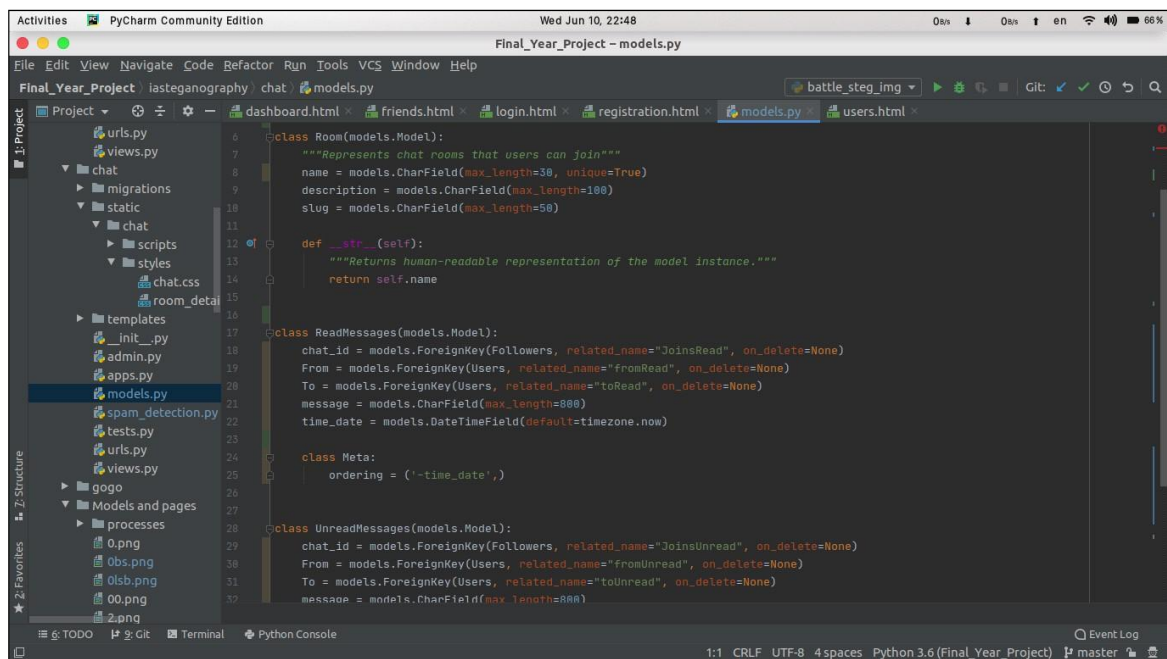


Figure 5.5.14 Chat Database

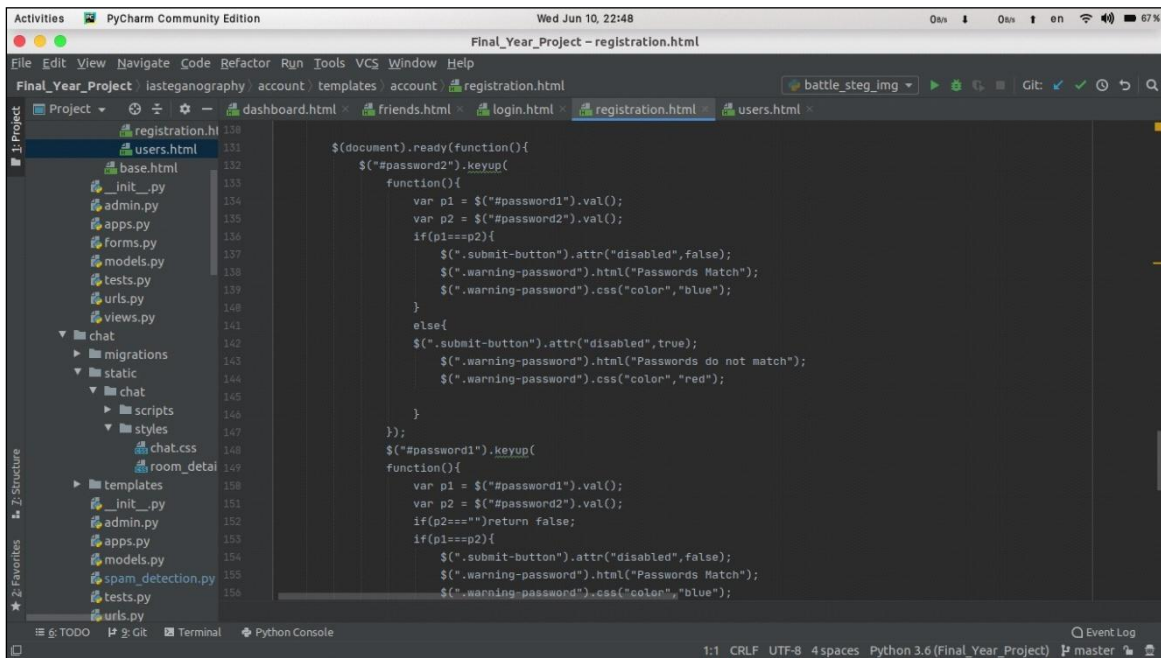


Figure 5.5.15 Registration verification code

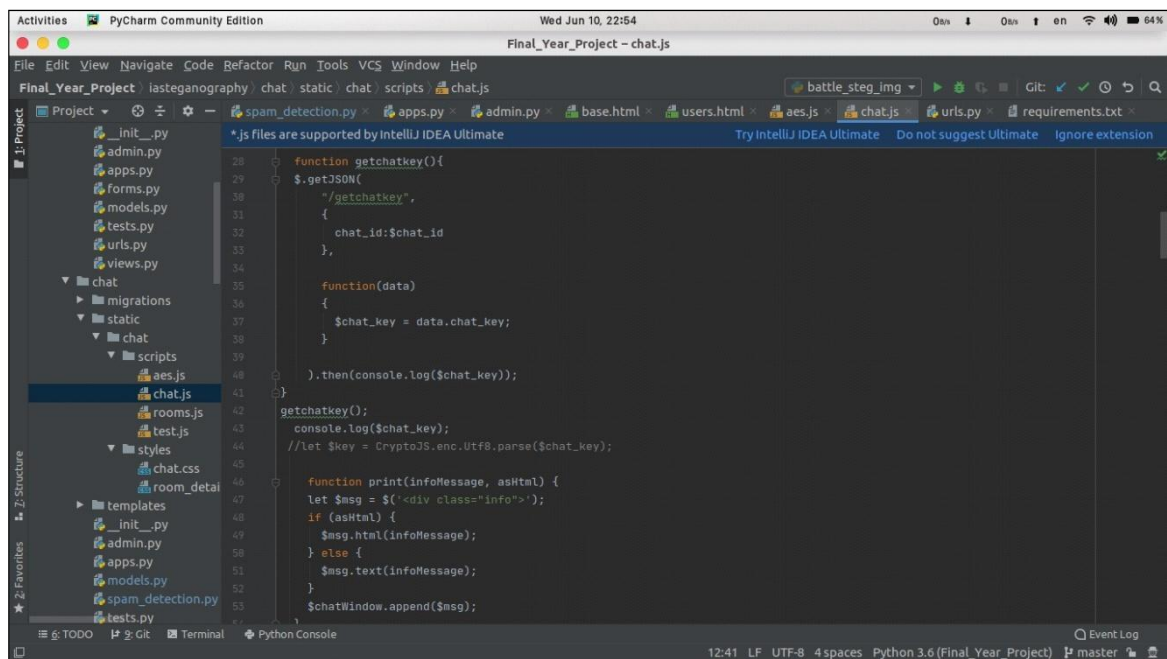


Figure 5.5.16 Token Passing at Client Side

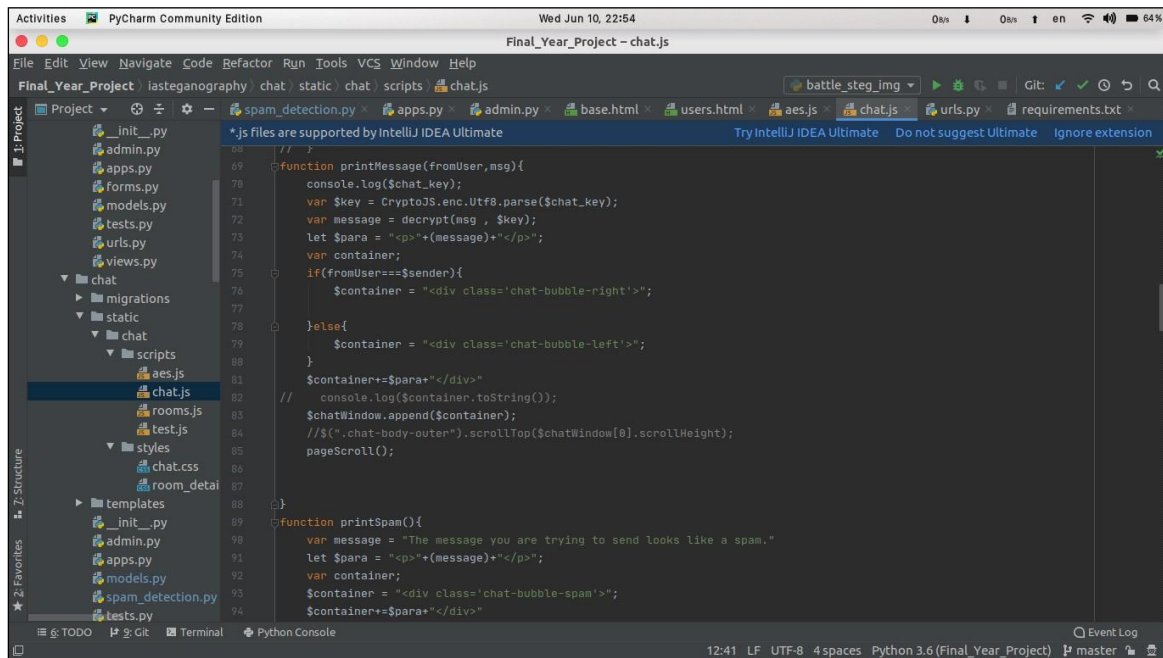


Figure 5.5.17 Chat and Spam Bubble at Client Side

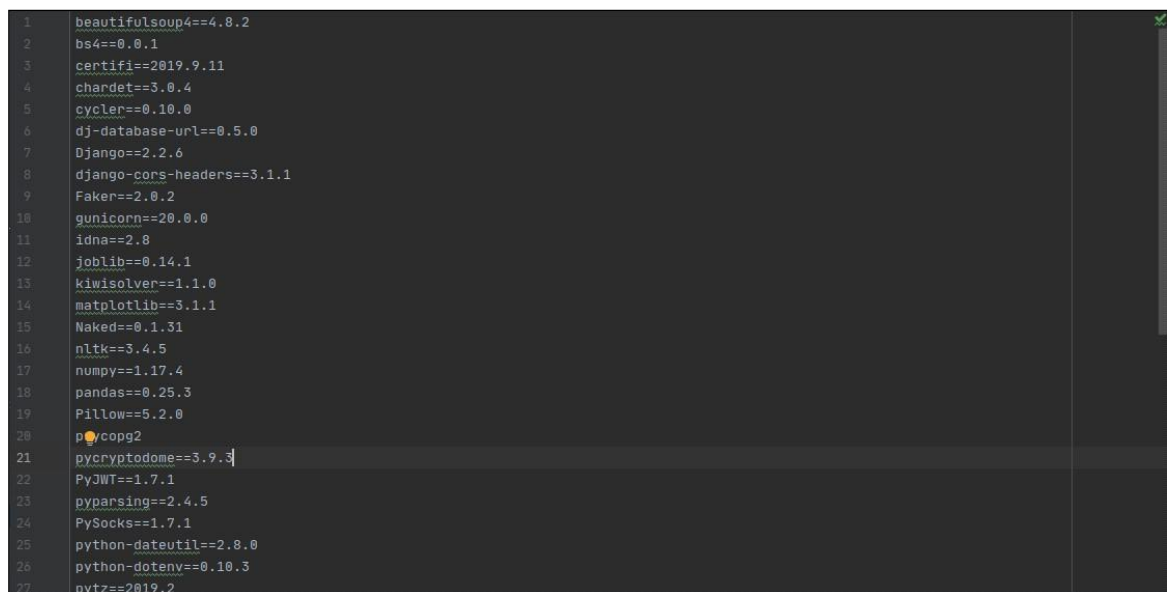


Figure 5.5.18 Requirements Configuration

CHAPTER 6

TESTING

The quality of any web service or software depends on testing i.e. whether it is tested properly or not. As testing is a process of executing an application or program of finding the software bugs (errors or other defects). It can also be started as the process of validating and verifying that an application meets the business and technical requirements that guided its design and development.

Software testing involves the execution of a software component or system component to one or more properties of interest. In general, these properties indicate the extent which the component or system under test:

- Meets the requirements that guided its design and development,
 - Responds correctly to all kinds of inputs,
 - Performs its functions within an acceptable time,
 - Is sufficiently usable

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding software bugs (errors or other defects).

The job of testing is an iterative process as when one bug is fixed, it can illuminate other, deeper bugs, or can even create new ones.

Software testing can provide objective, independent information about the quality of software and risk of its failure to users or sponsors.

A fundamental problem with software testing is that testing under *all* combinations of inputs and preconditions (initial state) is not feasible, even with a simple product. This means that the number of defects in a software product can be very large and defects that occur infrequently are difficult to find in testing. More significantly, non-functional dimensions of quality (how it is supposed to *be* versus what it is supposed to *do*)—

usability, scalability, performance, compatibility, reliability—can be highly subjective; something that constitutes sufficient value to one person may be intolerable to another.

Software developers can't test everything, but they can use combinatorial test design to identify the minimum number of tests needed to get the coverage they want. Combinatorial test design enables users to get greater test coverage with fewer tests. Whether they are

looking for speed or test depth, they can use combinatorial test design methods to build structured variation into their test cases.

Description of Agile Model

Agile software development refers to software development methodologies centered round the idea of iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. The ultimate value in Agile development is that it enables teams to deliver value faster, with greater quality and predictability, and greater aptitude to respond to change.

Diagram of Agile Model



Fig. 6.2.1 Agile

- By delivering working, tested, deployable software on an incremental basis, agile development delivers increased value, visibility, and adaptability much earlier in the life cycle, significantly reducing project risk.
- While the team needs to stay focused on delivering an agreed-to subset of the product's features during each iteration, there is an opportunity to constantly refine and reprioritize the overall product backlog.
- New or changed backlog items can be planned for the next iteration, providing the opportunity to introduce changes within a few weeks.

Types of Testing

There are mainly two types of testing:

Black Box Testing

Black Box Testing, also known as Behavioral Testing, is a software testing method in which the internal structure of the item being tested is not known to the tester.

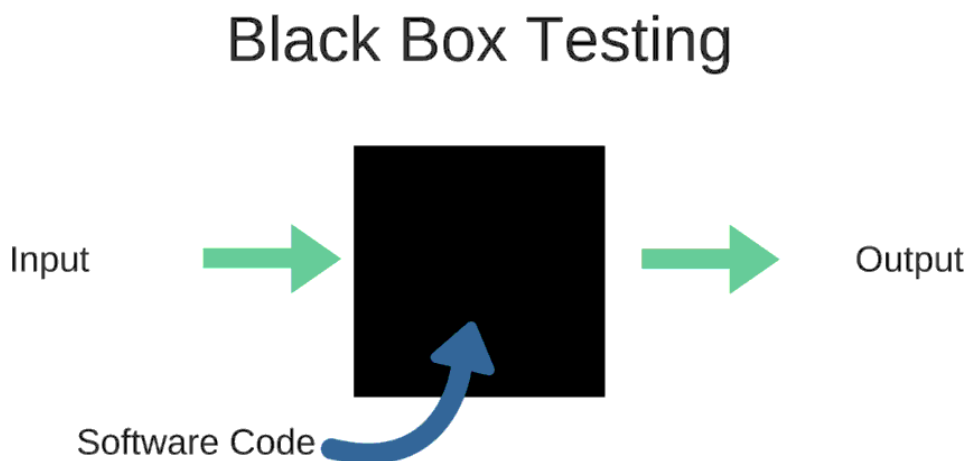


Fig 6.3.1.1 Blackbox Testing

This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories:

- Incorrect or missing functions
 - Interface errors
- Errors in data structures or external database access

One advantage of the black box technique is that no programming knowledge is required.

Whatever biases the programmers may have had, the tester likely has a different set and may emphasize different areas of functionality.

On the other hand, black-box testing has been said to be "like a walk in a dark labyrinth without a flashlight. Because they do not examine the source code, there are situations when a tester writes many test cases to check something could have been tested by only one test case, or leaves some part of the program untested.

White Box Testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing).

In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs.

White box testing is a method of testing the application at the level of the source code. These test cases are derived through the use of the design techniques mentioned above: control flow testing, data flow testing, branch testing, path testing, statement coverage and decision coverage as well as modified decision coverage.

White-box testing is the use of these techniques as guidelines to create an error free environment by examining any fragile code.

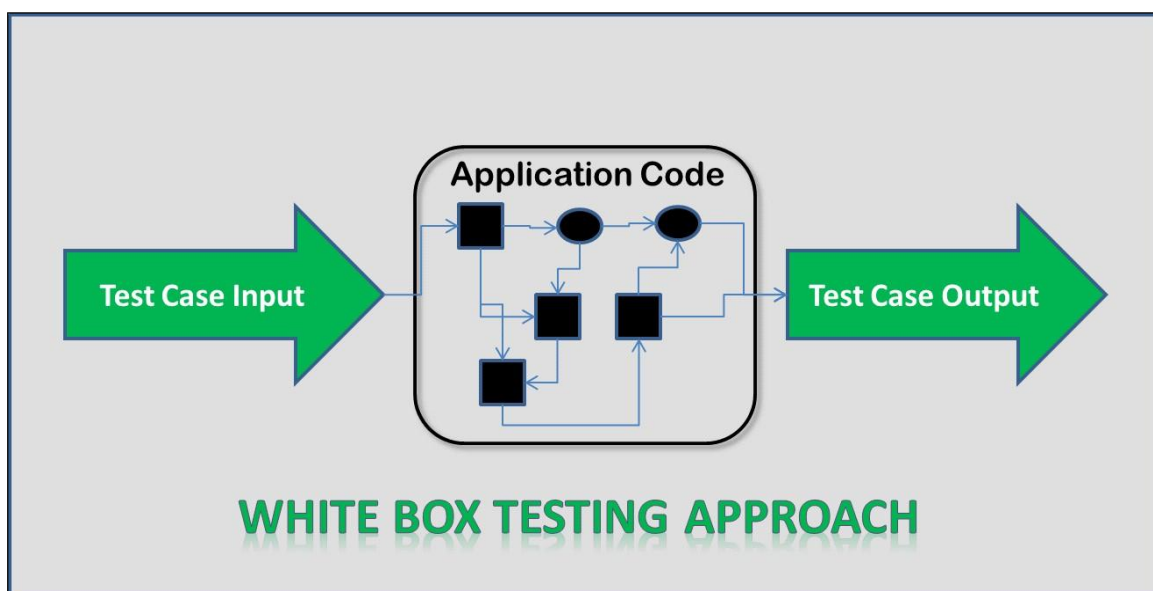


Fig. 6.3.2.1 White Box Testing

Steps to perform White box testing:

Step 1 – Understand the functionality of the application through its source code. Having said that, it simply means that the tester must be well versed with the programming language and other tools and techniques used to develop the software.

Step 2– Create the tests and execute them. When we discuss about testing, “coverage” is the most important factor. Here I will explain how to have maximum coverage in the context of White box testing.

White-box testing's basic procedures involves the tester having a deep level of understanding of the

source code being tested. The programmer must have a deep understanding of the application to know what kinds of test cases to create so that every visible path is exercised for testing. Once the source code is understood then the source code can be analyzed for test cases to be created.

These are the three basic steps that white-box testing takes in order to create test cases:

Input involves different types of requirements, functional specifications, detailed designing of documents, proper source code, security specifications. This is the preparation stage of white-box testing to layout all of the basic information.

Processing involves performing risk analysis to guide whole testing process, proper test plan, execute test cases and communicate results. This is the phase of building test cases to make sure they thoroughly test the application the given results are recorded accordingly.

Output involves preparing final report that encompasses all of the above preparations and results.

CHAPTER 7

CONCLUSION

Sending the messages to people and having conversations over text messages is very easy and free these days. Using internet for conversations has always issue of privacy. Even nearly all the messaging applications have their own way of providing their messaging services. With the help of your devices which have functionality to access the internet and have a browser, you can access this web application for free and from any part of the world.

Although messaging services have been there from a long time but we came with exclusive features that any other platform do not have. There is use of cover to hide every message (which is encrypted before hiding it on the cover). We have also acknowledged the fact that anything connected to the internet is vulnerable to hacking. So even if someone hacked and tried to read messages, they would not be able to because:

1. They will get the cover as it is sent over the internet.
2. Cover is made secure using steganography and cryptography algorithm.

We have used the cryptography and steganography to provide the security for the user's messages. This project is designed to block the spammy materials so this web application is suitable for your kids as well. It blocks all the spammy messages using machine learning algorithm.

The web application is useful for the kids, parents and for anyone who don't want spam in their messages and wants security of their contents. The web application is free and it does not take any additional devices to access, just your everyday devices like phones, laptops, tablets can be used to access the web application.

This web application can be used in everyday messaging and it is simple in a way that everyone can use it in just few seconds.

Future Scope

Future scope for this web application consists of making the service even smarter to block spams and other messages that consist of internet hate, bullying messages etc. Also making the

web application more secure and adding various innovative features and keeping up with changing technologies and time. This way the service will get better and better with the time.

Uses & Targets

Since the concept of web applications and messaging services is so broad, it is almost impossible to come up with a detailed specification of what a system for what the service will be able to do. As has been stated before, the main purpose is to provide secured messaging service that can block the spam messages so the user gets quality with the service.

There are every other messaging services that use various methods of securing the user's chat but most of them are being compromised with the hackings.

We are not denying the fact that any of the services on the internet can be compromised and to overcome with that we are using steganography concepts and algorithm in addition to the cryptography concepts and algorithms to make messaging even more secure. There will always be a cover with encrypted messages. The message is placed in cover in a way that it is hard to identify where it is placed as the cover does not get distorted so much and looks very similar to cover without anything hidden into it.

Anyone can use this web service and with growing time, this web application will become even more user friendly.

Applications

- This web application is used to send and receive messages.
- The message with spam content is detected and blocked.
 - Accessible from anywhere and all the time

Result & Analysis

This thesis set out to analyze the background and necessary conditions for success in the area of internet messaging services. Furthermore, an analysis of the current state of the market and development was to be made, and future trends to be extrapolated.

Finally, a prototype was to be created to show the possibilities in the area. The background concludes that both the technology of the market and the potential customers should be ready to use the service as a natural next step in the development. In the current development, two groups could be discerned. The first one is younger users preferably kids, to make service better while considering them. Surely, we need better understanding of kids, their psychology and the type of services that help them grow and do not harm their mental health.

The other group have people other than the first group to make service better by focusing on user

friendliness, security and not letting spams bother them while using the service. Based on the analysis, this thesis focused on a simple, yet powerful solution that would allow the users to have a secured service with providing quality in conversations.

Limitations

- The analysis of background and current and future development is quite shallow due to the time limitations. In order to get a better analysis, more information gathering, surveys and interviews would have to be conducted.
- The application is only available as a website and can only be accessed with a web browser. In future we might consider a mobile application as well.
- This, of course, also affects our design choices for the service, but the most fundamental assessments in that area should be correct. We are always prepared to come up with better solutions and to know what the customer actually wants irrespective of what we expected that a customer wants.
- This web application needs internet access every time and will not work without internet.
- In order to use the service, one must be registered with the website. It is for security purpose which cannot be compromised.
- The ability to use your custom covers is not given as of now but you can always choose our default covers., also we will be adding more covers in the upcoming time.

REFERENCES

- [1] Holovaty, A., & Kaplan-Moss, J. (2009). The definitive guide to Django: Webdevelopment done right. Apress.
- [2] Rish, I. (2001, August). An empirical study of the naive Bayes classifier. In IJCAI 2001 workshop on empirical methods in artificial intelligence (Vol. 3, No. 22, pp. 41-46).
- [3] Chandramouli, R., & Memon, N. (2001, October). Analysis of LSB based image steganography techniques. In Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205) (Vol. 3, pp. 1019-1022). IEEE.
- [4] Hussain, M., & Hussain, M. (2013). A survey of image steganography techniques.
- [5] Agrawal, A., & An, A. (2012, December). Unsupervised emotion detection from text using semantic and syntactic relations. In 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (Vol. 1, pp. 346-353). IEEE.
- [6] Johnson, N. F., & Jajodia, S. (1998). Exploring steganography: Seeing the unseen. *Computer*, 31(2), 26-34.
- [7] Cox, I., Miller, M., Bloom, J., Fridrich, J., & Kalker, T. (2007). Digital watermarking and steganography. Morgan kaufmann.
- [8] Low, S., & Wilson, G. (2003). U.S. Patent Application No. 10/136,022.
- [9] Gordon, A. T. (1997). U.S. Patent No. 5,608,786. Washington, DC: U.S. Patent and Trademark Office.
- [10] Safavian, S. R., & Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3), 660-674.
- [11] Kleinbaum, D. G., Dietz, K., Gail, M., Klein, M., & Klein, M. (2002). Logistic regression. New York: Springer-Verlag.