Automatic Report Generation

PROJECT REPORT submitted in partial fulfillment of the requirements

Submitted by

Aravapalli Lakshmi Keerthana 178W1A1201

Under the Guidance of

Dr.N.Neelima, Ph.D

Asst. Professor

For the award of the degree

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

V R SIDDHARTHA ENGINEERING COLLEGE

(AUTONOMOUS - AFFILIATED TO JNTU-K, KAKINADA)

Approved by AICTE &Accreted by NBA

KANURU, VIJAYAWADA-7

ACADEMIC YEAR

(2020-21)

V.R.SIDDHARTHA ENGINEERING COLLEGE

(Affiliated to JNTUK: Kakinada, Approved by AICTE, Autonomous)

(An ISO certified and NBA accredited institution)

Kanuru, Vijayawada – 520007

CERTIFICATE

This is to certify that this project report titled **"Automatic Report Generation For Security Audit"** is a bonafide record of work done by **Aravapalli Lakshmi Keerthana (178W1A1201)** under my guidance and supervision is submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology, **V.R. Siddhartha Engineering College** (Autonomous under JNTUK) during the year 2020-21.

Dr. N. Neelima	**Dr. M. Suneetha**
Asst. Prof	Professor &Head
Dept. of Information Technology	Dept. of Information Technology

ACKNOWLEDGEMENT

First and foremost, I sincerely salute our esteemed institution **V.R SIDDHARTHA ENGINEERING COLLEGE** for giving me this opportunity for fulfilling my project. I am grateful to our principal **Dr. A.V.RATNA PRASAD**, for his encouragement and support all through the way of my project.

On the submission of this Project report, I would like to extend my honor to **Dr. M.Suneetha**, Head of the Department, IT for her constant motivation and support during the course of my work.

I feel glad to express my deep sense of gratitude to my project guide Dr.N.Neelima, Asst. Prof**,** for her guidance and assistance in completing this project successfully.

I would also like to convey my sincere indebtedness to all faculty members, including supporting staff of the Department, friends and family members who bestowed their great effort and guidance at appropriate times without which it would have been very difficult on my part to finish the project work.

TABLE OF CONTENTS

ACKNOWLEDGEMENT iii

LIST OF FIGURES v

ABSTRACT vi

CHAPTER-1 Introduction 1

1.1Origion of the Problem 1

1.2 Basic definitions and Background 1

1.3 Problem Statement 2

CHAPTER-2 Review of Literature 4

2.1Description of Existing Systems 4

2.3Summary of Literature Study 4

CHAPTER-3 Proposed Method 5

3.1Design Methodology (UML representation) 5

3.2System Architecture Diagram 6

3.3Description of Algorithms 7

3.4 Description of datasets and Tools 15

CHAPTER-4 Results and Observations 19

4.1Stepwise description of Results 19

4.2Test case results 28

4.3Observations from the work 38

CHAPTER-5 Conclusion and Future study 39

5.1 Conclusion 39

References 40

Appendix 41

LIST OF FIGURES

Figure	Page
Figure3.1: UML Diagram	5
Figure3.2: Architecture Diagram	6
Figure 3.3.1: Stored Procedure to insert project details	8
Figure 3.3.2: SQL Query to modify project details	9
Figure 3.3.3: Stored Procedure to insert vulnerability details	10
Figure 3.3.4: SQL Query to modify Vulnerability details	10
Figure 3.3.5: SQL Query to display all project details	11
Figure 3.3.6: SQL Query to add vulnerability details and project details for report	12
Figure 3.3.7: SQL Query to add observation steps and image details	14
Figure 3.3.8: SQL Query to insert retest details	15
Figure 3.4.1: Table for Project Details	16
Figure 3.4.2: Table for Vulnerability Details	16
Figure 3.4.3: Table for Project Vulnerabilities	17
Figure 3.4.4: Table for Retest Details	17
Figure 3.4.5: Table for Observation Details	18
Figure 4.1.1: Home Page	20
Figure 4.1.2: Adding New vulnerability Details	21
Figure 4.1.3: Entering Vulnerability details in the form	22
Figure 4.1.4: Editing Vulnerability Details	23
Figure 4.1.5: Updating the vulnerability details	24
Figure 4.1.6: Creating a New Project form	25
Figure 4.1.7: Entering project details in form	25
Figure 4.1.8: Editing Project details	26
Figure 4.1.9: List of all Project Details	26

Figure 4.1.10: Selecting Vulnerabilities present in a project 27

Figure 4.1.11: Edit Retest Details 28
Figure 4.2.1: Entering project details in form 29
Figure.4.2.2: Entering vulnerability details in the form 30
Figure.4.2.3: Updating the vulnerability details 31
Figure 4.2.4: Updating Project details before adding vulnerabilities 31
Figure 4.2.5: Generated Report 32
Figure 4.2.6: Generated Report 33
Figure 4.2.7: Generated Report 34
Figure 4.2.8: Generated Report 35
Figure 4.2.9: Generated Report 36
Figure 4.2.10: Project Details 37
Figure 4.2.11: Vulnerability Details 37
Figure 4.2.12: Project Vulnerability Details 37
Figure 4.2.13: Observation steps Details 37
Figure 4.2.14: Retest Details 37

ABSTRACT

Now-a-days most of the companies started automating the tasks which are mundane and consumes a lot of time. Automation is the creation and application of technologies to produce and deliver goods and services with minimal human intervention. The implementation of automation technologies, techniques and processes improve the efficiency, reliability, and speed of many tasks that were previously performed by humans. Andhra Pradesh Technology Services performs security audits for government applications. After performing the audit, a report needs to be created in which all the Vulnerability details and the steps followed to resolve the vulnerabilities are listed. Currently, APTS is using Microsoft Word to generate the report. The aim of this project is to automate the process of report generation. Instead of storing a bulk amount of files, they want to maintain a website that can store the details of applications received for audit. For this .net framework is used to develop a web application that can allow users to enter the details. After taking the details, they are stored in the database and retrieved at the time of report generation. SQL Server is used as a backend database. The users should be able to modify the application audit details and generate reports at any time. This results in decreased human efforts and saves the time consumed.

Keywords: .Net Framework, Audit report, Automatic report generation, SQL Server, Vulnerability details.

CHAPTER-1
INTRODUCTION

1.1 Origin of the Problem

Andhra Pradesh Technology Services Limited (APTS) is a wholly-owned Government Company incorporated in the year 1986. Under the administrative control of the Information Technology, Electronics & Communication Department (ITE&C) Department, APTS is a self-sustained company, offering wide-ranging of services including implementation of IT and Infrastructure Projects, IT Consultancy Services, Information Security Assurance Services & Procurement of Infrastructure for Departments. APTS is the nodal agency for Implementation of Cyber Security Policy (APCSP) promulgated for realizing the vision of "Building a Secure and resilient cyberspace for Citizens, Business and Government". Besides offering holistic and competent services in office automation, APTS focuses on signing MOUs, striking joint ventures and Special Purpose Vehicles (SPVs) with technology partners including start-ups with the objective of introducing new and innovative technologies – to be used by the Government – to improve efficiency, transparency and cost effectiveness.

APTS performs security audits for profit and non-profitable organizations. They perform security assessments and identify vulnerabilities present in the applications. After identifying the vulnerabilities, they need to generate a detailed report with recommendation steps to be followed to resolve the vulnerability. Also they need a structured application that can help them to organize their project details.

2. Basic Definitions

a) .Net Framework - .Net Framework is a software development platform developed by Microsoft for building and running Windows applications. The .Net framework consists of developer tools, programming languages, and

libraries to build desktop and web applications. It is also used to build websites, web services, and games.

b) SQL Server-SQL Server Management Studio (SSMS) is an IDE that provides a graphical interface for connecting and working with MS SQL server. It was launched with Microsoft SQL Server 2005 and is used for configuring, managing, and administering all components within Microsoft SQL Server.

c) Vulnerability-a vulnerability is a weakness that can be exploited by cybercriminals to gain unauthorized access to a computer system.

d) Security Assessment- It is a process of identifying vulnerabilities present in an application and also finding the risk level based on the number of threats.

e) Visual Studio-Microsoft Visual Studio is an integrated development environment from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps.

f) AJAX-Ajax is a set of web development techniques using many web technologies on the client-side to create asynchronous web applications. With Ajax, web applications can send and retrieve data from a server asynchronously without interfering with the display and behaviour of the existing page.

g) Bootstrap-Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

3. Problem Statement

For generating reports on security audit APTS is currently using Microsoft Word and manually creating the report. The report includes detailed information about the vulnerability details that are identified in an application. Manually generating a report usually takes at least 2 days. Inspecting an application, identifying and listing all the vulnerabilities consumes time. Also by manually generating a report, there is a chance of encountering grammatical mistakes or sentence errors. By automating it can easily reduce human effort and time. We can customize the font and structure of the report while building a report using .Net Framework and bootstrap. So that we can avoid all possible mistakes.

CHAPTER-2
REVIEW OF LITERATURE

2.1 Description of Existing Systems

The organization is currently using Microsoft Word to generate reports after a security audit. In this process they need to type everything manually which requires a lot of human effort and time. It is also causing grammatical and sentence formation errors while generating the report. For that the organization needs to check the report again and again which consumes their time. In some cases they need to generate reports again for their client needs. All these activities take a lot of time and human effort.

2.2 Summary of Literature Study

Automating the report generation process can save the time spent on the task and the efforts needed to be put in. A Web Application is developed to automate the process of generating the report. In that application we can automatically generate everything related to the report. We can add the projects and vulnerability details to the database from the application. We can modify or edit the existing projects and vulnerability details. Since all the details will be present in the database it will be easy to generate a report as it just requires to get the details from the database and display them. This Application helps to generate reports in a productive and effective manner and saves the time and helps to

generate more and more reports in less time without stress. This can help the employees to concentrate on more creative works.

CHAPTER-3
PROPOSED METHOD

3.1 Design Methodology

The members of the organization can have access to all the webpages. They can create and modify project details and vulnerability details. From the list of all projects they can access any project and directly add vulnerability details to the project. If new vulnerabilities were found in the process of testing they can add that vulnerability to the master table by creating a new vulnerability. Also the user can retest the project again if it comes for a security audit again. The details will be modified and updated in the project table as a new record. Finally users can generate reports with vulnerability details , recommendations, and pie charts that describes the level of risk and other briefings.

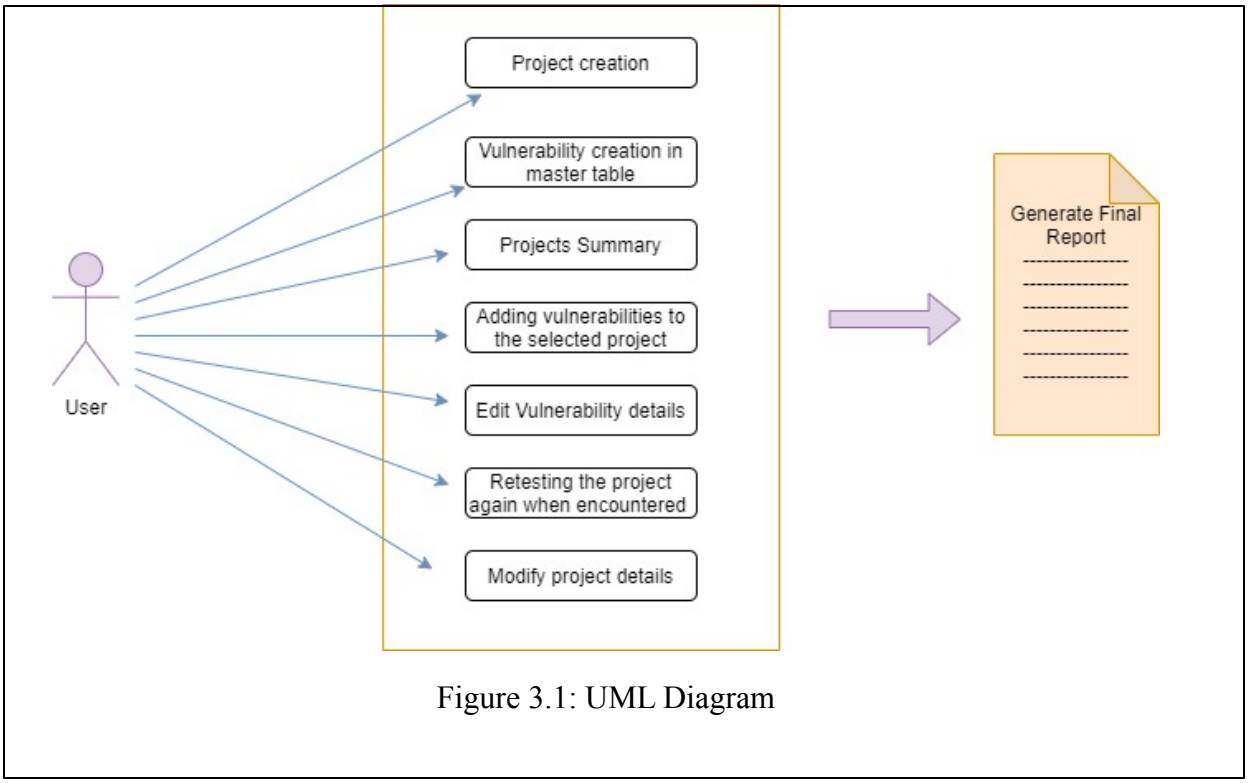


Figure 3.1: UML Diagram

3.2 System Architecture Diagram

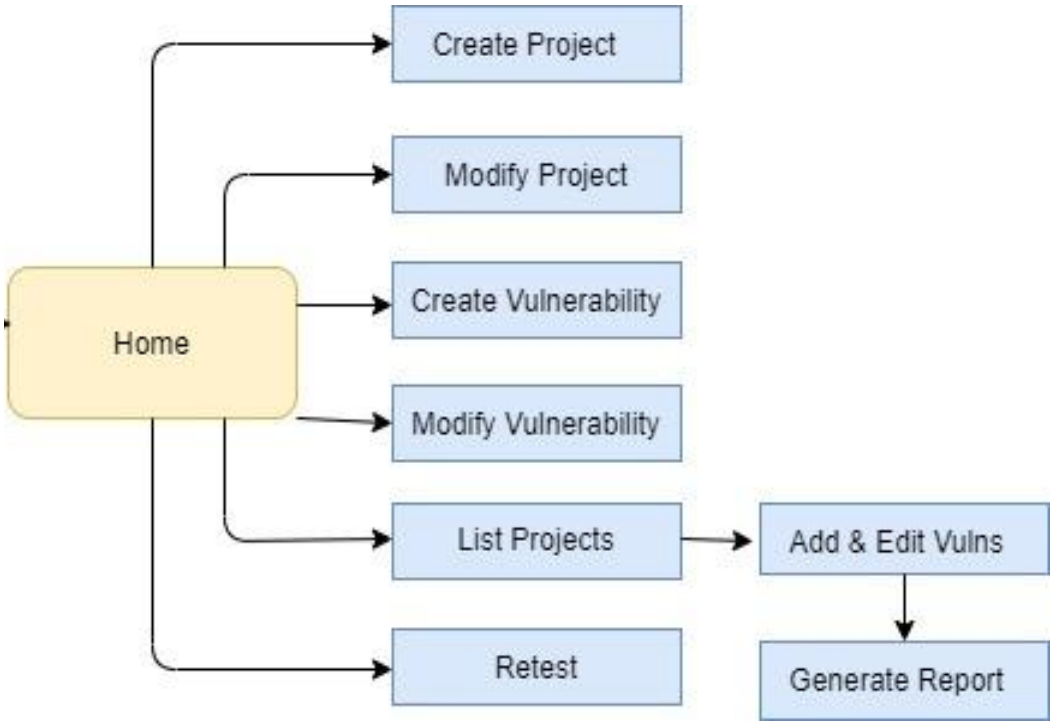


Figure 3.2: Architecture Diagram

- Users will be presented with the home page. The home page consists of Buttons which redirect to the following pages Create Project, Modify Project, Create Vulnerability, Modify Vulnerability, List Projects (In List Projects we have another two buttons “Add & Edit Vulnerability” and “Generate Report” which redirect to the two pages respectively) and Retest.

- On clicking the Create Project Button it redirects to the Create Project page where the user creates the new project on which a security audit will be performed.

- On clicking the Modify Project Button it redirects to the Modify Project page where the user modifies or edits the details of the existing projects.

- ➔ On clicking the Create Vulnerability Button it redirects to the Create Vulnerability page. It is the Master Vulnerability page where the user enters the vulnerability details.

- ➔ On clicking the Modify Vulnerability Button it redirects to the Modify Vulnerability details page. It modifies the vulnerability details present in the masterlist.

- ➔ On clicking the List Projects Button it redirects to the List Projects page where we find all projects that completed the security audit and the projects on which the user is performing security audit. In the List Project there are two buttons "Add & Edit Vulnerability" and "Generate Report" which redirect to the two pages respectively.

- ➔ On clicking the Add & Edit Vulnerability Button it redirects to the Add & Edit Vulnerability page where the user can select the vulnerabilities that are detected in the project and modify or edit the details of the Vulnerabilities according to the project.

- ➔ On clicking the Generate Report Button it redirects to the Generated Report page with the above mentioned details.

- ➔ Finally we have a Retest Button on clicking that button it redirects to the Retest page where the user can perform the retest on the given application. It lists the details of the project that comes for retest.

3.3 Description of Procedure

We created a website using .Net Framework with visual studio as the platform. The Database is maintained using Microsoft SQL Server. The creation of the website started with creating a Home Page. It contains a header with details of the organization for whom we are creating a website. Home Page contains all the navigation buttons for the rest of the website forms. The page is designed using Bootstrap to make it more responsive.

Home Page

1. Home page consists of buttons which can redirect the user to other pages.
2. It consists of buttons that can be used to navigate to create project, Modify project, create vulnerability, modify vulnerability, projects list and report pages.

Create Project Form

1. The Create Project web page contains all the information about the applications the organization receives for security audit.
2. The project details include the department or organization name from which the project came for security audit.
3. Also it includes client company details,start and end assessment dates,auditor information and audit status, application information like type of os used,ip address,port number and web server.
4. Textboxes are used to collect the data from the user and SQL insert query used to store the data in the database.
5. All these details of the project are gathered and structured in the project table at the backend for report generation.

```
CREATE procedure [dbo].[create_project](@Application_Name varchar(100),@Department_Name varchar(50),@Category_of_organization varchar(50),
@Sector_of_organization varchar(50),@Type_of_Audit varchar(50),@Reason_for_Audit varchar(max),@State varchar(50),@URL varchar(max),
@TestedAppVersion varchar(max),@SPOC_Details varchar(max),@StartDate date,@EndDate date,@AuditStatus varchar(50),@AuditorName varchar(max),
@IPAddress varchar(50),@OS varchar(50), @AppRun varchar(50),@WebServer varchar(50),@Portno int )
as
begin
insert into dbo.Project(Application_Name,Department_Name,Category_of_organization,Sector_of_organization,Type_of_Audit,Reason_for_Audit,
State,URL,TestedAppVersion,SPOC_Details,StartDate,EndDate,AuditStatus,AuditorName,IP_Address,OS,AppRunning,WebServer,PortScanned)
values(@Application_Name,@Department_Name,@Category_of_organization,@Sector_of_organization,@Type_of_Audit,@Reason_for_Audit,@State,@URL,
@TestedAppVersion,@SPOC_Details,@StartDate,@EndDate,@AuditStatus,@AuditorName,@IPAddress,@OS,@AppRun,@WebServer,@Portno)
end
GO
```

Figure 3.3.1: Stored Procedure to insert project details

Modify Project

1. The Modify Project page is used to update details of existing projects. For this a dropdown is added to list all the names of the existing projects.
2. From this user will select a project and details of that project are displayed using form view's item template.
3. Form view's edit item template is used to allow users to update the details.
4. SQL Select command command is used to display the details and Update Command is used to update the data in the database.

```
 <asp:LinkButton ID="UpdateButton" runat="server" class="btn btn-dark" CausesValidation="True" CommandName="Update" Text="Update" />
p;<asp:LinkButton ID="UpdateCancelButton" class="btn btn-dark" runat="server" CausesValidation="False" CommandName="Cancel" Text="Can
/div> </EditItemTemplate>

p:FormView>
:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:ReportGenerator %>" SelectCommand="SELECT [
epartment_Name=@Department_Name, Category_of_organization=@Category_of_organization, Sector_of_organization=@Sector_of_organization,
<SelectParameters>
    <asp:ControlParameter ControlID="Name" Name="Application_Name" PropertyName="SelectedValue" Type="String" />
</SelectParameters>
<UpdateParameters>
    <asp:ControlParameter ControlID="Name" Name="Application_Name" PropertyName="SelectedValue" Type="String" />
<asp:Parameter Name="Department_Name" Type="string" />
       <asp:Parameter Name="Category_of_organization" Type="string" />
       <asp:Parameter Name="Sector_of_organization" Type="string" />
       <asp:Parameter Name="Type_of_Audit" Type="string" />
    <asp:Parameter Name="Reason_for_Audit" Type="string" />
    <asp:Parameter Name="URL" Type="string" />
    <asp:Parameter Name="TestedAppVersion" Type="string" />
    <asp:Parameter Name="SPOC_Details" Type="string" />
    <asp:Parameter Name="StartDate" Type="DateTime" />
    <asp:Parameter Name="EndDate" Type="DateTime" />
    <asp:Parameter Name="Audit_Status" Type="string" />
    <asp:Parameter Name="Auditor_Name" Type="string" />
    <asp:Parameter Name="State" Type="string" />
</UpdateParameters>
```

Figure 3.3.2: SQL Query to modify project details

Add Vulnerability to database

1. Create vulnerability page is used to collect the details of the vulnerabilities which are identified in the project.
2. Form input fields are used to collect the details from the user.

3. The database has the following information about the vulnerabilities App type, vulnerability title, severity, owasp category, cve id, description ,impact, recommendation and attributing factor.
4. All this information is stored in the vulnerability table in the database.
5. This serves as a master list. When a new application comes for audit this master list can be used for selecting Vulnerabilities that are present in the application.
6. Then the stored vulnerability details are retrieved directly in the report page.

```
CREATE procedure [dbo].[create_vulnerability](@Name varchar(50),@AppType varchar(10),@Severity varchar(20),@CWE_CVE_id varchar(20),
@OWASP varchar(50),@Description varchar(max),@Impact varchar(max),@Recommendation varchar(max),@Attributing_Factor varchar(50))
as
begin
insert into Vulnerability(Name,AppType,Severity,CWE_CVE_id,OWASP,Description,Impact,Recommendation,Attributing_Factor)
values(@Name,@AppType,@Severity,@CWE_CVE_id,@OWASP,@Description,@Impact,@Recommendation,@Attributing_Factor)
end
GO
```

Figure 3.3.3: Stored Procedure to insert vulnerability details

Modify Vulnerability Details

1. Modify Vulnerability page is used to update details of Vulnerabilities added to master list.
2. For this a dropdown is added to list all the names of the Vulnerabilities.
3. From this user will select a Vulnerability and details of that Vulnerability are displayed using form view's item template.
4. Form view's edit item template is used to allow users to update the details.
5. SQL Select command command is used to display the details and Update Command is used to update the data in the database.

```
<asp:LinkButton ID="UpdateButton" class="btn btn-dark" runat="server" CausesValidation="True" CommandName="Update" Text="Update" />
|  <asp:LinkButton ID="UpdateCancelButton" class="btn btn-dark" runat="server" CausesValidation="False" CommandName="Cancel" Text
        </div>
                </EditItemTemplate>


    </asp:FormView>
    <asp:SqlDataSource ID="SqlDataSource2" runat="server" ConnectionString="<%$ ConnectionStrings:ReportGenerator %>" SelectCommand="
        <SelectParameters>
            <asp:ControlParameter ControlID="Name" Name="Name" PropertyName="SelectedValue" Type="String" />
        </SelectParameters>
        <UpdateParameters>
        <asp:ControlParameter ControlID="Name" Name="Name" PropertyName="SelectedValue" Type="String" />
                <asp:Parameter Name="AppType" Type="string" />
                <asp:Parameter Name="Severity" Type="string" />
                <asp:Parameter Name="CWE_CVE_id" Type="string" />
                <asp:Parameter Name="OWASP" Type="string" />
            <asp:Parameter Name="Description" Type="string" />
            <asp:Parameter Name="Impact" Type="string" />
            <asp:Parameter Name="Recommendation" Type="string" />
            <asp:Parameter Name="Attributing_Factor" Type="string" />
        </UpdateParameters>
    </asp:SqlDataSource>
    <br />
```

Figure 3.3.4: SQL Query to modify Vulnerability details

Project Summary

1. The project summary page displays all the list of projects/applications they received till date.
2. For ease of search, a search is provided at the top of the page with a search button.
3. To display all the projects we used gridview and datatable in visual studio that retrieves and displays the data in a table format.
4. To display the data in the gridview the web application should connect with the database.
5. We imported necessary libraries in c# to connect the web form with the database.
6. To connect with database the following code need to be used in web-config file-

   <connectionStrings>

   <add name="ReportGenerator"connectionString="Data Source=ITC51_172\SQLEXPRESS02;Initial Catalog=ReportGenerator;Integrated

Security=True;MultipleActiveResultSets = True"/>

</connectionStrings>

7. If a user wants to edit the details before proceeding with adding vulnerabilities, an edit button is provided in the gridview table for every project detail to update the details.
8. To add the vulnerabilities the user needs to click on the 'Add Vulnerability' button to add the vulnerabilities to the project.

```
cs.Open();
SqlCommand cmd = new SqlCommand("select * from project", cs);
SqlDataReader dr = cmd.ExecuteReader();
if (dr.HasRows == true)
{
    GridView1.DataSourceID = null;
    GridView1.DataSource = dr;
    GridView1.DataBind();
}
```

Figure 3.3.5: SQL Query to display all project details

Add Vulnerability Details to Project

1. This page is used to select the vulnerabilities that are identified in the application which are to be added to the report for that project.
2. First the user selects the project and its status for which the report has to be generated.
3. All the Vulnerabilities present in the master list are displayed in this page.
4. Gridview is used to display the vulnerability name, apptype, severity, cve id, owasp, description, impact, recommendation.
5. A column with checkboxes is added to allow users to select the vulnerabilities to be added to the project report.
6. The selected Vulnerabilities are added to a data table and stored in a session variable to access them in the next pages.
7. SQL Select statement is used to display the Vulnerability details in grid view.

8. After Selecting, the user clicks on the add button which will then be redirected to the next page.

```
CREATE PROCEDURE [dbo].[StoreProjVulner](@ProjName varchar(Max),@Name varchar(max),@ProjStatus varchar(max),@Severity varchar(max),
@Description varchar(max),@Impact varchar(max),@Recommendation varchar(max),@AppType varchar(max),@CWE_CVE_id varchar(max),
@OWASP varchar(max),@Attributing_Factor varchar(max))
as
Begin
insert into project_Vulnerabilities(Project_Name,VulnerabilityName,Audit_Status,Severity,Description,Impact,Recommendation,AppType,
CWE_CVE_id,OWASP,Attributing_Factor) values(@ProjName,@Name,@ProjStatus,@Severity,@Description,@Impact,@Recommendation,@AppType,
@CWE_CVE_id,@OWASP,@Attributing_Factor)
end

GO
```

Figure 3.3.6: SQL Query to add vulnerability details and project details for report

Edit Vulnerabilities

1. The Vulnerabilities selected in the previous page are displayed here by accessing the datatable stored in the session variable.
2. These are displayed using a grid view.
3. A column with a hyperlink “select” is added to the gridview.
4. It is used to redirect to the edit project vulnerabilities page which allows the user to edit the respective vulnerability details.

Edit Project Vulnerabilities

1. This page allows users to edit the details of vulnerabilities added to the project.
2. It also allows users to add the observation steps and upload the images corresponding to each step.
3. The details are displayed as a form.
4. A text box and file upload is provided to add observation steps.
5. Users can enter the step in the textbox and upload the corresponding image and click on the add step to store it in the database.
6. After adding all steps the user can click on the update button.

```
Create procedure [dbo].[uploadimage](@ProjName varchar(50),@VulnName varchar(50), @obs varchar(MAX),
@Name varchar(MAX),@ImageData varbinary(max))
as
begin
insert into Observation
values(@ProjName,@VulnName,@obs,@Name,@ImageData)

end
GO
```

Figure 3.3.7: SQL Query to add observation steps and image details

Report

1. Report consists of Security Audit Details, Disclaimer, Table of Contents, Summary, Risk Level, Threats summary, Executive Summary, Vulnerabilities in Detail, annexure.
2. Security Audit Details consists of resource name and testing url. Summary consists of IP address of the server, Operating System, Application running, Web Server, Port Scanned.
3. Risk Level indicates the level of risk present in the application.
4. If one or more High severe vulnerabilities are discovered in the website then overall threat level is rated as High.
5. If one or more Low severe vulnerabilities are discovered in the website then overall threat level is rated as Low.
6. If one or more Medium severe vulnerabilities are discovered in the website then overall threat level is rated as Medium.
7. If one or more Informative vulnerabilities are discovered in the website then overall threat level is rated as Informative.
8. Threats Summary gives the information about the total threats found in the application.
9. It also shows the information in the form of a pie chart based on the risk level of the threat.

10. Executive Summary describes on which application the security assessment is conducted and the start and end date of the assessment.
11. It consists of a table providing the details like url at which the vulnerability is discovered, vulnerability name, recommendation, severity.
12. Vulnerabilities in Detail lists all the vulnerabilities in the application.
13. Each vulnerability consists of the following details:Owasp, Affected url, Description, proof of concept and recommendation.
14. Proof of concept consists of Observation steps and pictures corresponding to each step.
15. All these details are retrieved from the database using sql select query.
16. Ajax Toolkit is used to create pie charts from the data extracted from the database.
17. Annexure contains all the urls that are crawled during security assessment of the application.

Retest Form

1. Retest page is used when the same application is encountered for security audit.
2. Search for the application name in the search bar provided and click on the retest button to edit the assessment start and end date details and auditor name and audit status.
3. After editing click on the insert button, then a new record will be stored in the project table in the database.
4. Then the newly inserted record will go for security testing again.

```
create procedure [dbo].[RetestInsert](@Name varchar(50),@StartDate date, @EndDate date,@AuditStatus varchar(20), @Auditor varchar(50))
as
Begin
insert into Retest(Application_Name,Audit_Status,Auditor_Name,Start_Date,End_Date) values (@Name,@AuditStatus , @Auditor, @StartDate,
@EndDate)
end
GO
```

Figure 3.3.8: SQL Query to insert retest details

Change Vulnerability Status

1. After adding a new record and submitting the user will be redirected to this page.
2. In this page all the vulnerabilities added to the project in its initial stage are displayed.
3. Here users can change the status of the Vulnerabilities displayed to open, closed or not tested.
4. The Item Template of GridView is used to display the details.
5. Edit item template is used to allow users to update the status of the vulnerabilities.

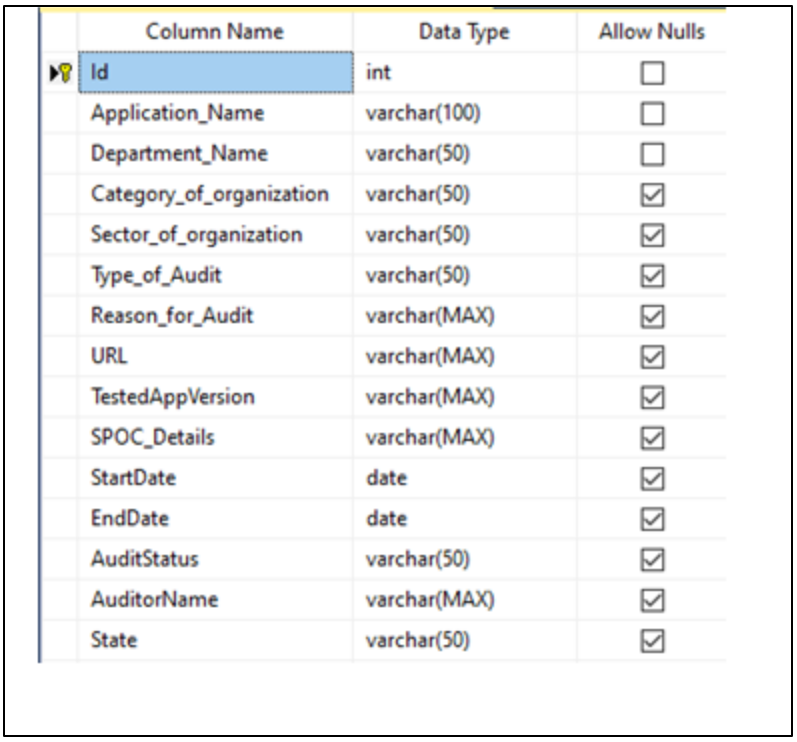
3.4 Description of Datasets and Tools

For this project we use “SQL Server Management Studio(SSMS)”database SSMS is an integrated environment for managing any SQL infrastructure, from SQL Server to Azure SQL Database. SSMS provides tools to configure, monitor, and administer instances of SQL Server and databases. Use SSMS to access, configure, manage, administer, and develop all components of SQL server.

The database consists of following tables:

1. Project
2. Vulnerabilities
3. Project Vulnerabilities
4. Retest
5. Observation

Project Table:

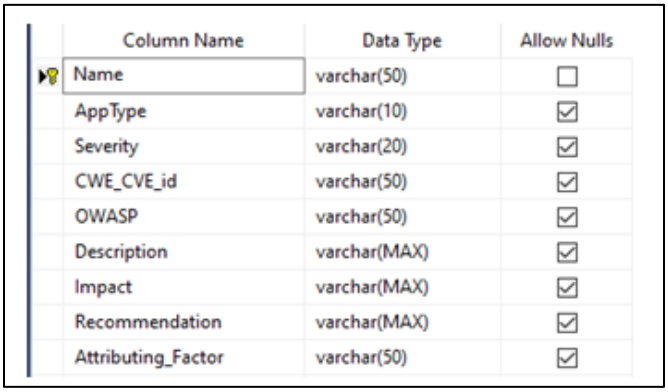


Column Name	Data Type	Allow Nulls
Id	int	☐
Application_Name	varchar(100)	☐
Department_Name	varchar(50)	☐
Category_of_organization	varchar(50)	☑
Sector_of_organization	varchar(50)	☑
Type_of_Audit	varchar(50)	☑
Reason_for_Audit	varchar(MAX)	☑
URL	varchar(MAX)	☑
TestedAppVersion	varchar(MAX)	☑
SPOC_Details	varchar(MAX)	☑
StartDate	date	☑
EndDate	date	☑
AuditStatus	varchar(50)	☑
AuditorName	varchar(MAX)	☑
State	varchar(50)	☑

Figure 3.4.1: Table for project Details

The project table contains all the details of the project on which the security audit is performed. The user also creates the new project on which the security audit will be performed.

Vulnerabilities Table:



Column Name	Data Type	Allow Nulls
Name	varchar(50)	☐
AppType	varchar(10)	☑
Severity	varchar(20)	☑
CWE_CVE_id	varchar(50)	☑
OWASP	varchar(50)	☑
Description	varchar(MAX)	☑
Impact	varchar(MAX)	☑
Recommendation	varchar(MAX)	☑
Attributing_Factor	varchar(50)	☑

Figure 3.4.2: Table for Vulnerability Details

It is the master vulnerabilities table. The table displays the details of all the vulnerabilities listed.

Project Vulnerabilities Table:

Column Name	Data Type	Allow Nulls
Project_Name	varchar(MAX)	☑
VulnerabilityName	varchar(MAX)	☑
Audit_Status	varchar(MAX)	☑
Severity	varchar(MAX)	☑
Description	varchar(MAX)	☑
Impact	varchar(MAX)	☑
Recommendation	varchar(MAX)	☑
AppType	varchar(MAX)	☑
CWE_CVE_id	varchar(MAX)	☑
OWASP	varchar(MAX)	☑
Attributing_Factor	varchar(MAX)	☑
Vulnerability_Status	varchar(50)	☑

Figure 3.4.3: Table for Vulnerabilities added to a specific Project

The project vulnerabilities table displays the details of the vulnerabilities that are detected from the project after a security audit is performed. We can add or edit the new vulnerabilities here.

Retest Table:

Column Name	Data Type	Allow Nulls
Application_Name	varchar(50)	☑
Audit_Status	varchar(20)	☑
Auditor_Name	varchar(50)	☑
Start_Date	date	☑
End_Date	date	☑
Retest	int	☑

Figure 3.4.4: Table for Retest Details

The Retest table lists the details about the project that comes for the retest. It shows the dates of when retesting started and ended.

Observation Table:

Column Name	Data Type	Allow Nulls
ProjectName	varchar(50)	☑
VulnerabilityName	varchar(50)	☑
ObservationStep	varchar(MAX)	☑
ImageName	varchar(MAX)	☑
Image	varbinary(MAX)	☑

Figure 3.4.5: Table for Observation Details

The team needs to store the data about the vulnerabilities that are detected and their related images for the report generation purpose. This table lists observations of detected vulnerabilities and its images. At the time report generation the information stored in this table gets the report.

CHAPTER-4
RESULTS AND OBSERVATIONS

4.1.Stepwise Description of Results

The web application consists of the following modules/Pages

1. Create Project
2. Modify Project Details
3. Create Vulnerability(To add vulnerabilities for master list)
4. Modify Vulnerability Details(of Master List)
5. List all Projects
6. Select Vulnerabilities present in project
7. Modify Vulnerability Details according to Project
8. Generate Report
9. Retest

Figure 4.1.1 shows the home page that will be shown to the user upon accessing the application. It consists of buttons which help to navigate to other pages.

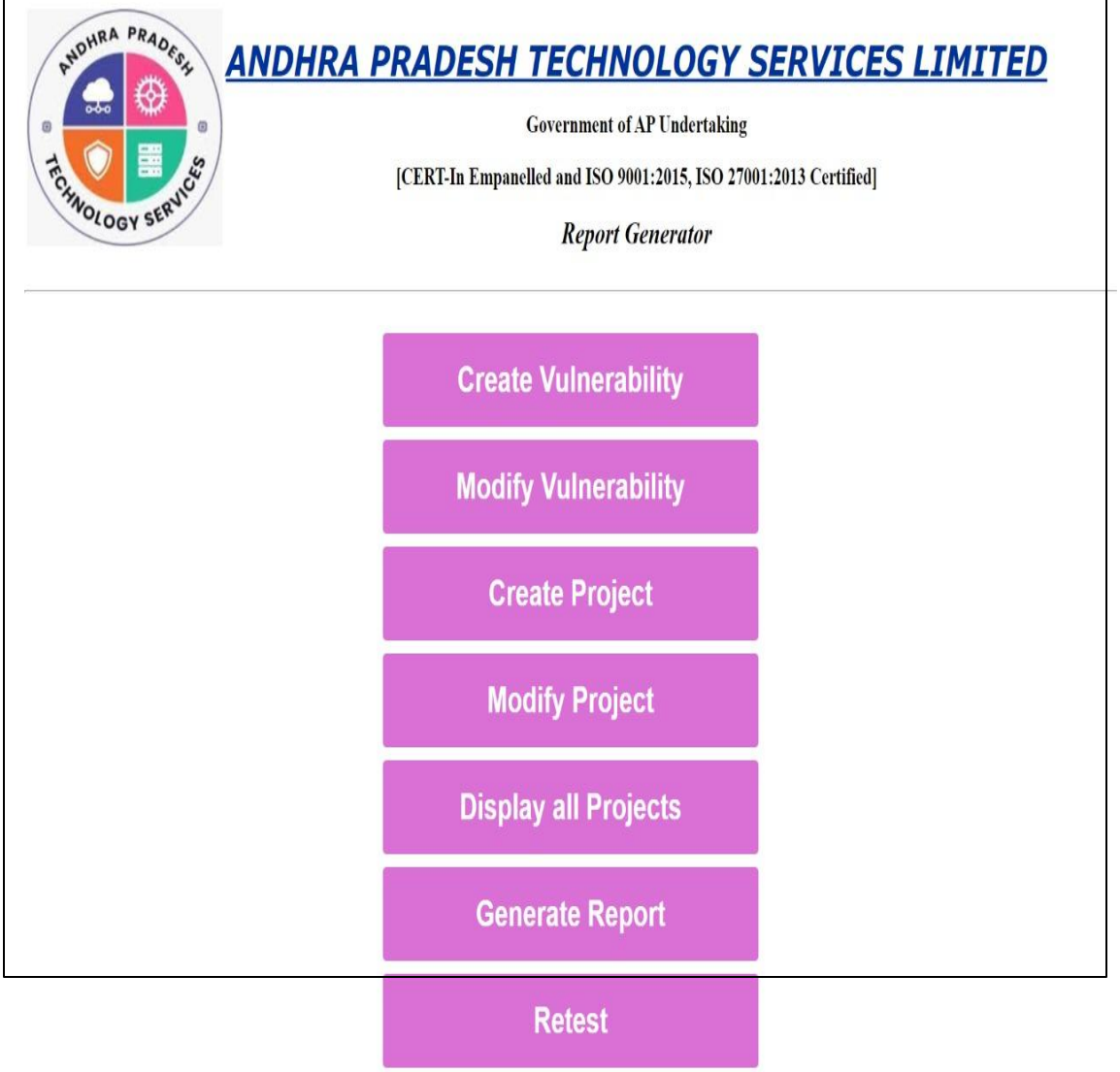


Figure 4.1.1: Home Page

The figure 4.1.2 shows the Create Vulnerability Screen. This page allows users to create a master list of vulnerabilities by entering the details of each vulnerability.

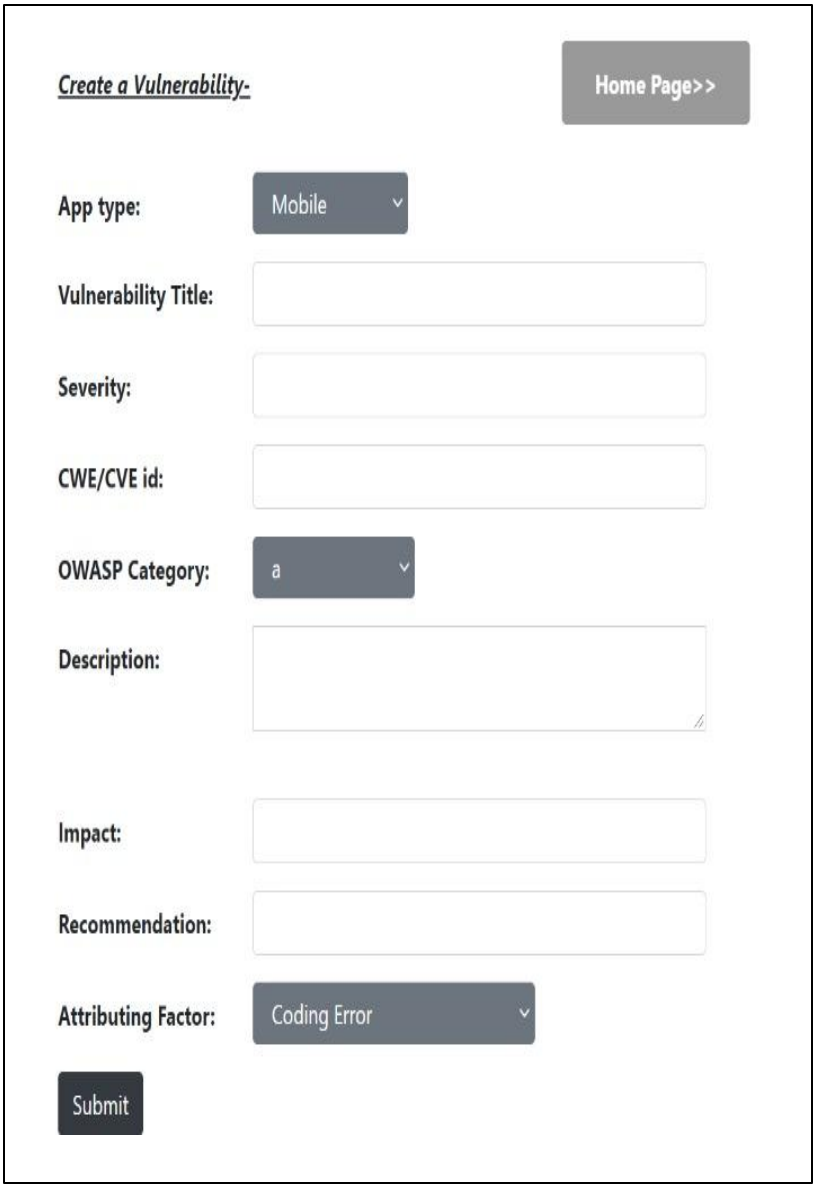


Figure 4.1.2: Adding New Vulnerability Details

The figure 4.1.4 shows the Edit Vulnerability Details Screen. This page allows users to edit the details of vulnerabilities present in the master list. When the user selects the Vulnerability name in the dropdown, all the details related to that Vulnerability. Users can click on the edit button to update the details. After editing, they can click update to update these details in the database.

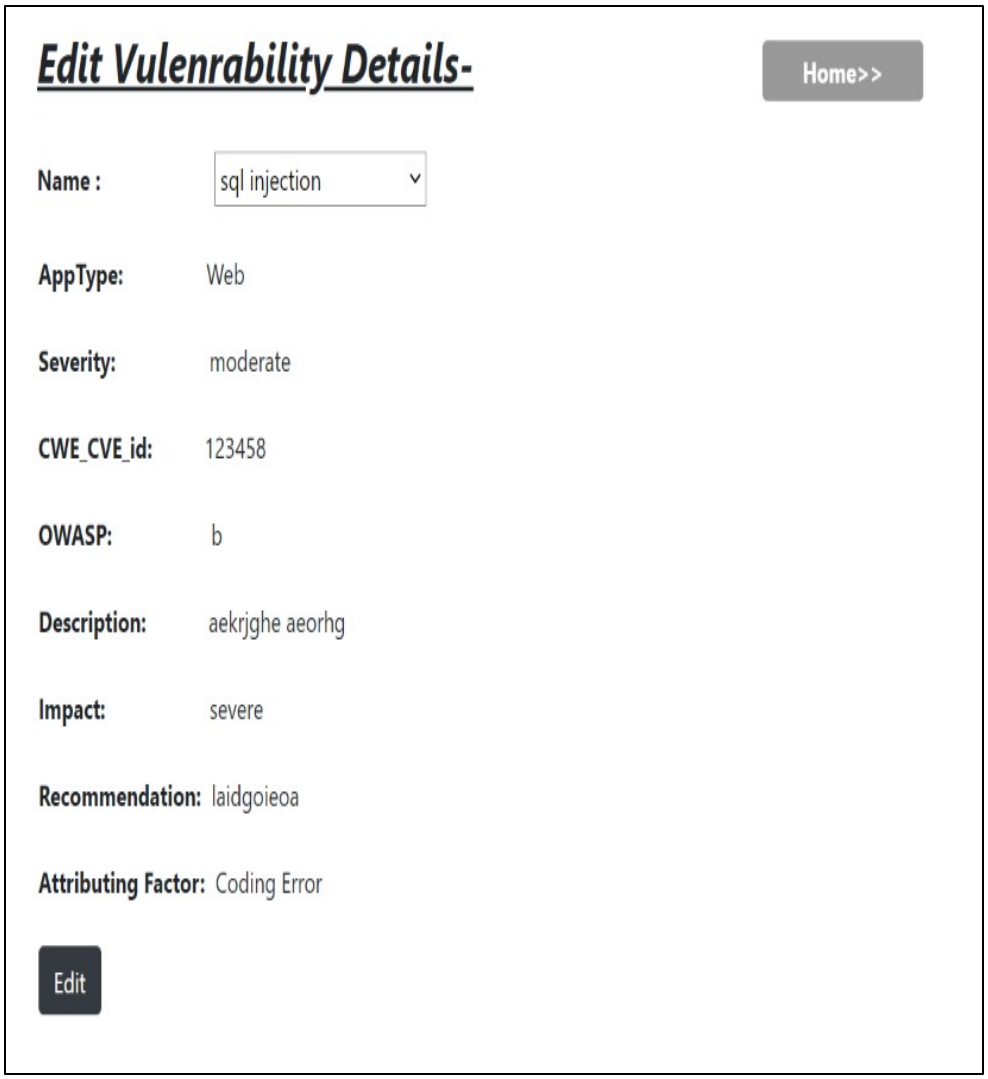


Figure 4.1.3: Editing Vulnerability Details

Figure 4.1.6 shows the create project screen. It allows users to add the details of the application that comes for audit as a new project. User enters all the required details and clicks on submit. The details will be added to the database as a new record

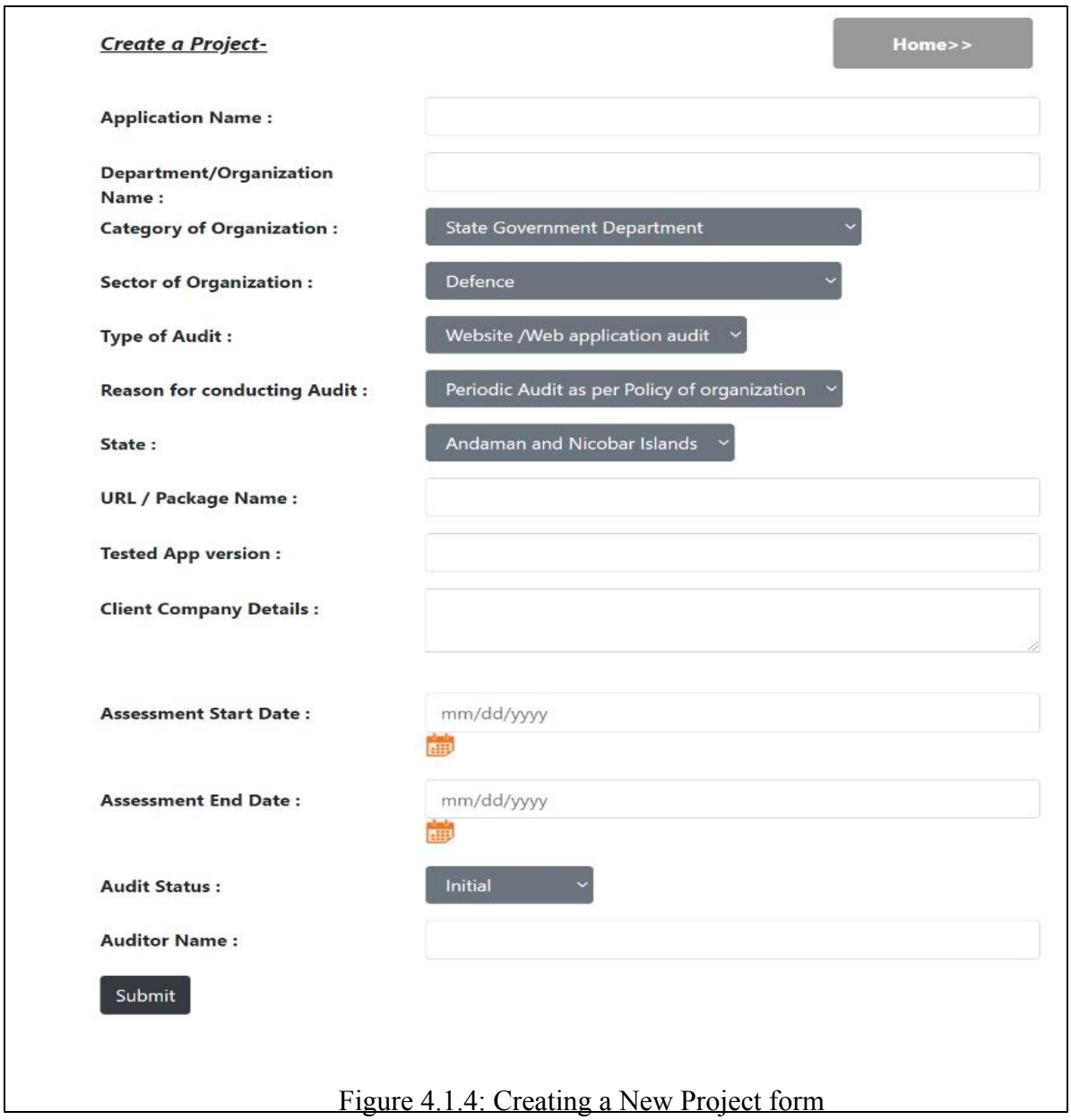


Figure 4.1.4: Creating a New Project form

Figure 4.1.8 shows the modified project screen. It allows users to modify the details of the already existing applications. When the user selects the application name in the dropdown, all the details related to that project. Users can click on the edit button to update the details. After editing, they can click update to update these details in the database.

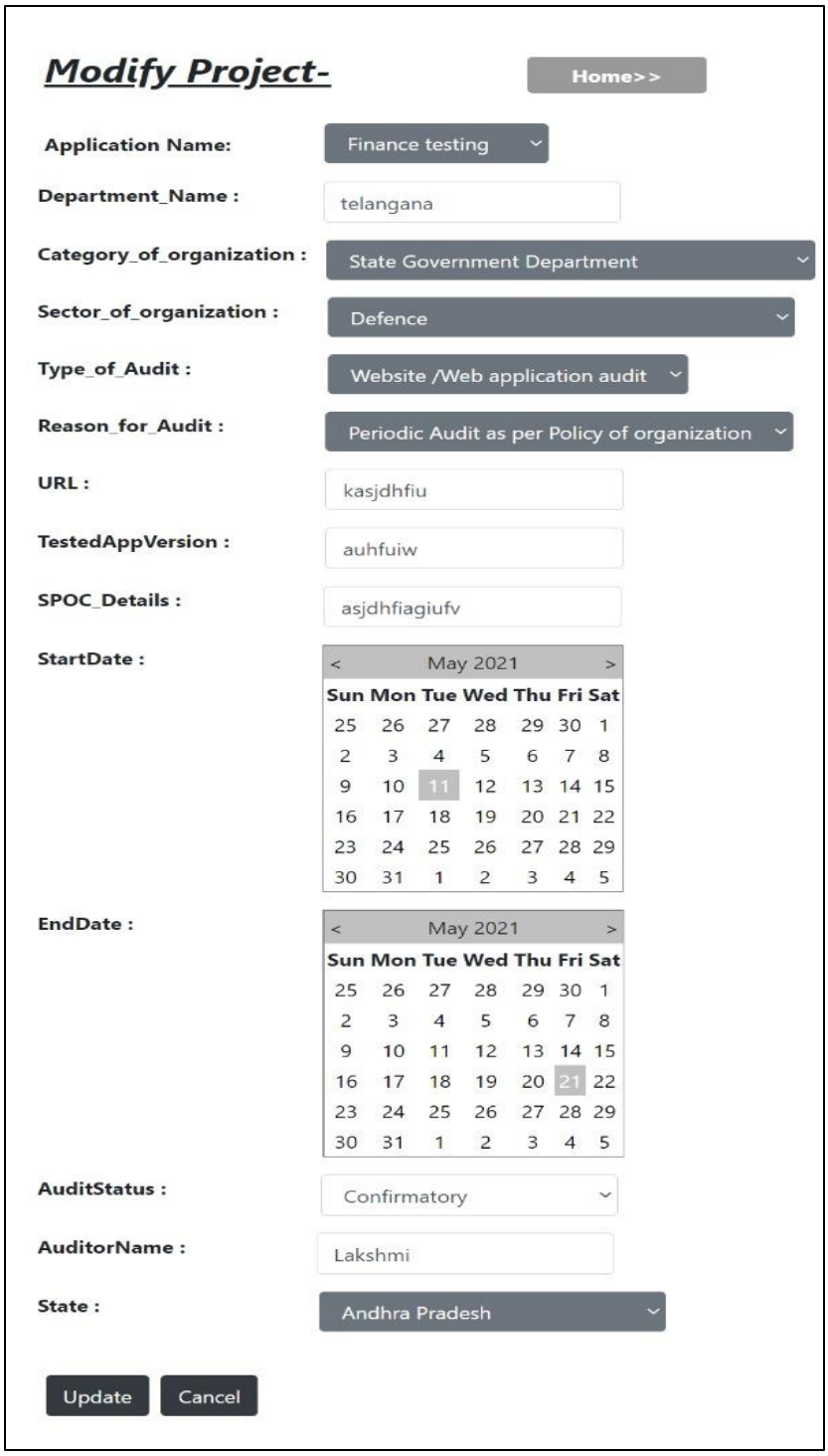


Figure 4.1.5: Editing Project Details

Figure 4.1.9 displays the list of all projects present in the database. From here users can select a project and click on add vulnerabilities to add the details of vulnerabilities

present in that project. Users can also search the projects based on project name and department names. Here users can also update the status of the project.

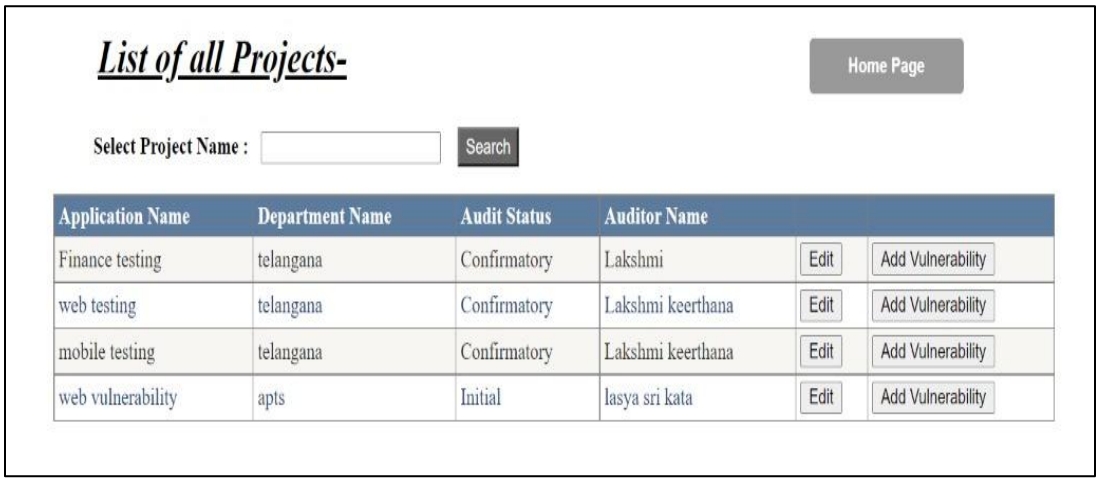


Figure 4.1.6: List of all Projects

Figure 4.1.11 shows the screen that will be displayed when the Add Vulnerability button is clicked. Here all the vulnerabilities present in the master list are displayed. Users can select the vulnerabilities that are present in the project which are to be added to the report and click add.

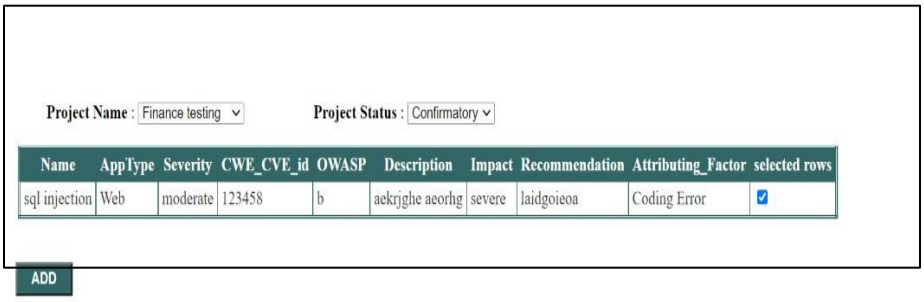


Figure 4.1.7: Selecting Vulnerabilities present in a Project

Figure 4.1.12 shows the screen that will be displayed after clicking add in the previous page. In this page the user can select any vulnerability and click on select to modify the details according to the project. Add crawled URLs helps to navigate to a

page where all the URL's that are crawled during the audit can be added. After this the user clicks on Generate report to generate report for the project.

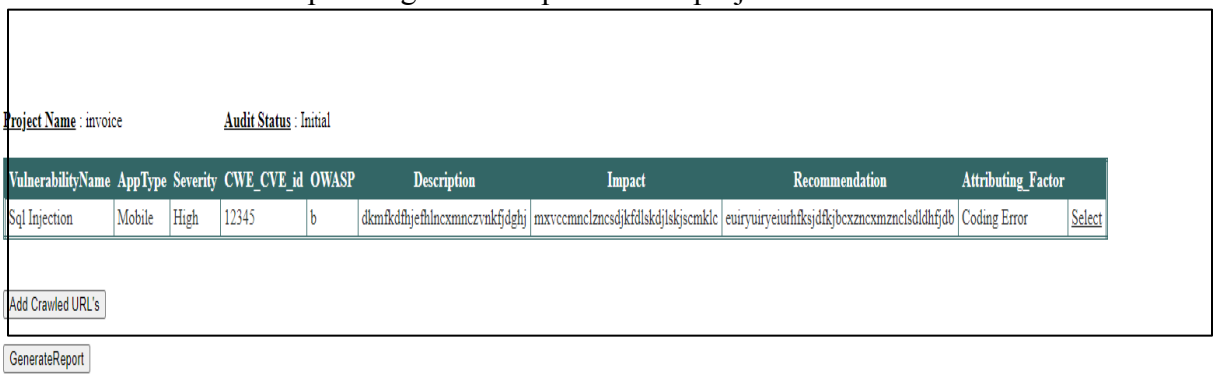


Figure 4.1.8: Display Selected Vulnerabilities

Figure 4.1.13 shows the page in which the user can edit the vulnerability details according to the project.

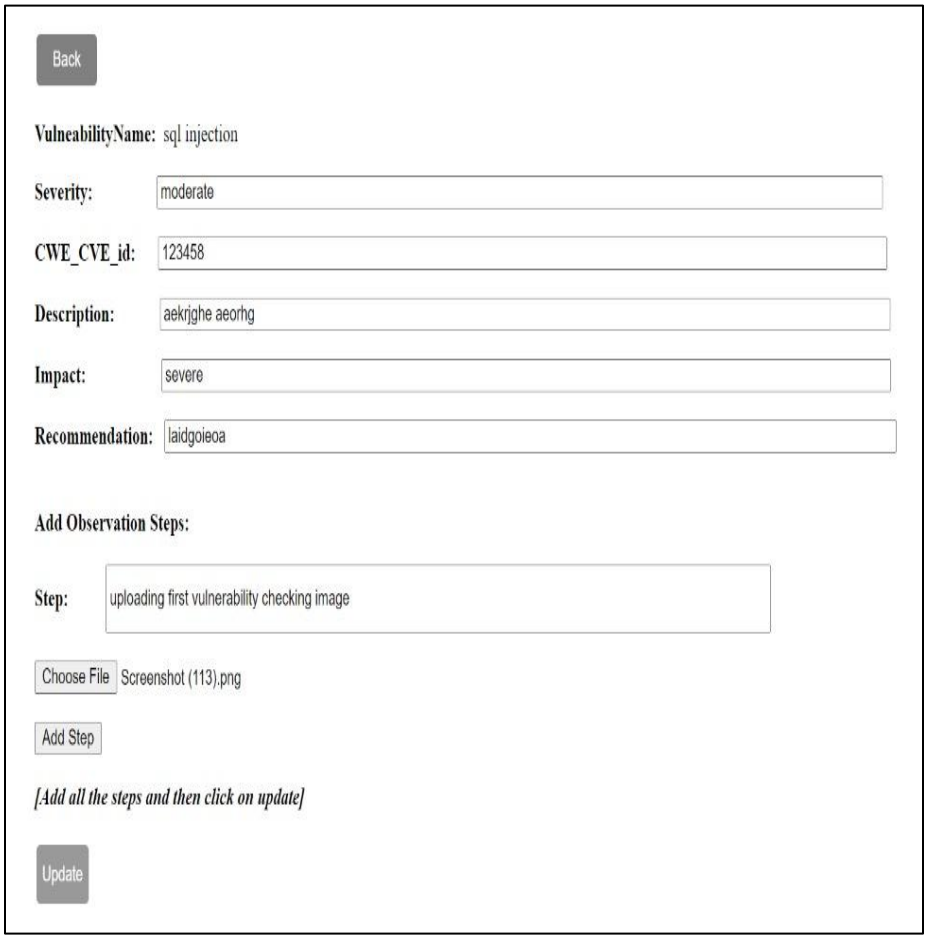


Figure 4.1.9: Edit Vulnerability Details for Project

Figure 4.1.14 shows the page that will be displayed upon clicking Retest in the Home page. This module is used when an application comes for retest. Users can select the project from the dropdown and click on retest. All the details related to the project will be displayed. Users can click on a new record to edit the details and store them as a new record in the database. Upon submitting the user will be directed to a page where all the list of vulnerabilities added in the initial stage of the project will be displayed. They can change the status of those Vulnerabilities.

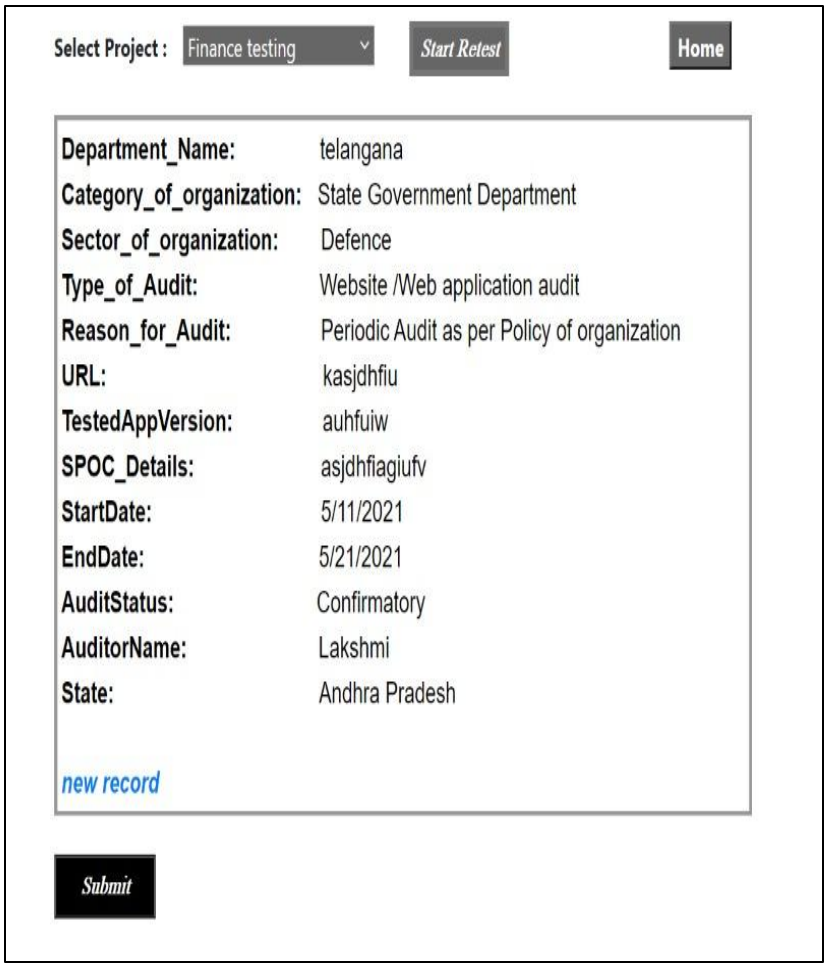


Figure 4.1.10: Retest Page

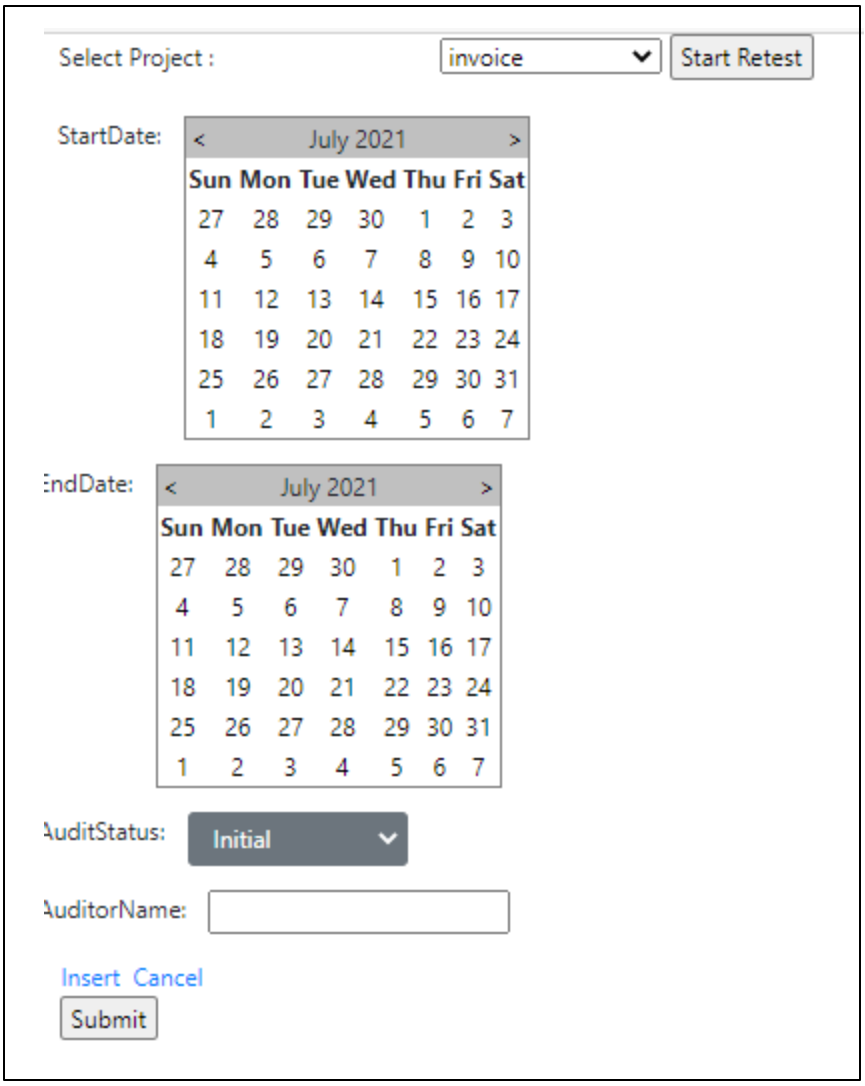


Figure 4.1.11: Edit Retest Details

4.2 Test case results

1. We tested the application by adding a project and a few vulnerabilities.
2. A project named APCSOC is added to the database using the create project module.

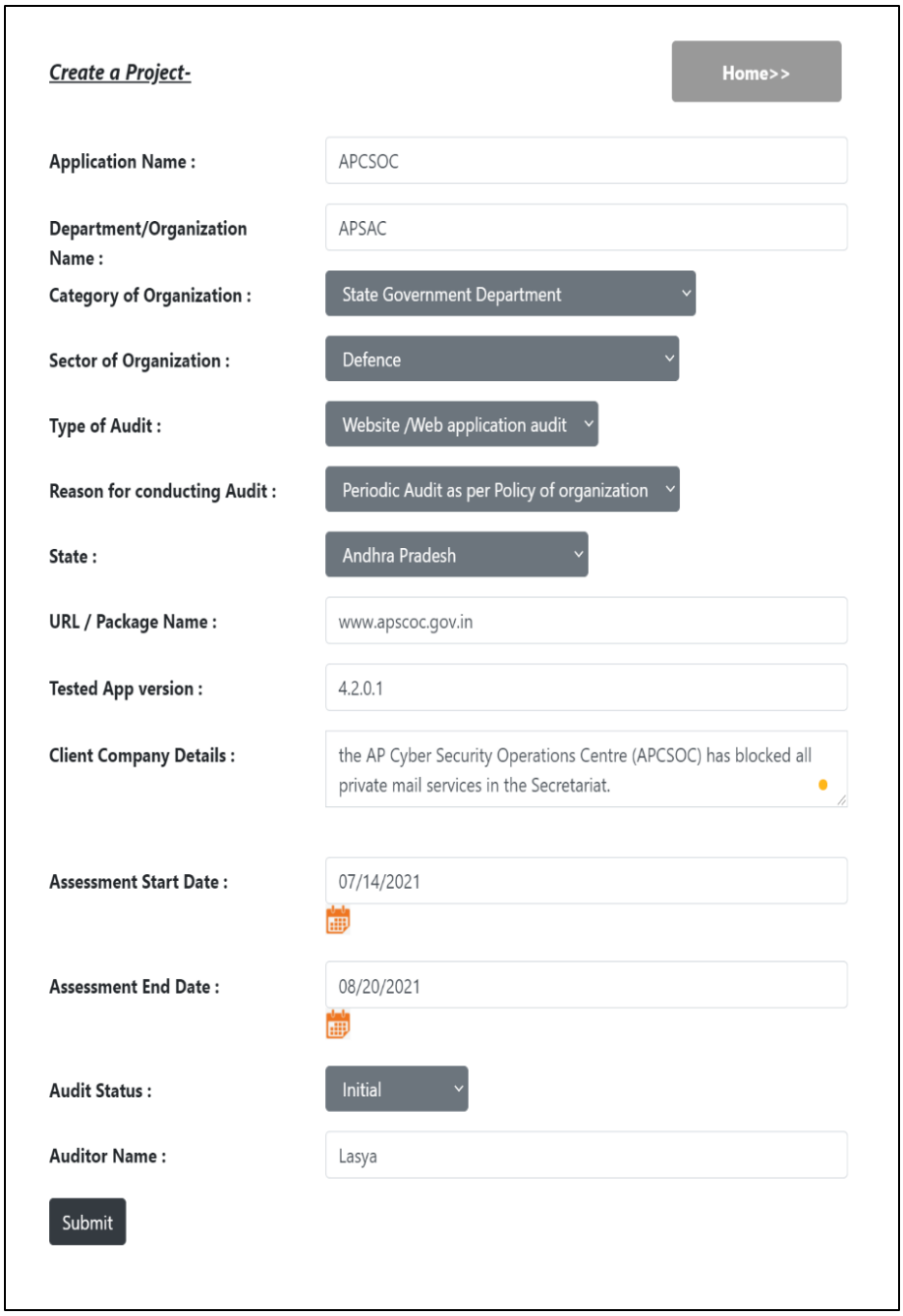


Figure 4.2.1: Entering project details in form

3. The Modify Project module is tested by changing the auditor name and updating it in the database.
4. The Create Vulnerability Module is tested by adding the details of vulnerabilities to the database.

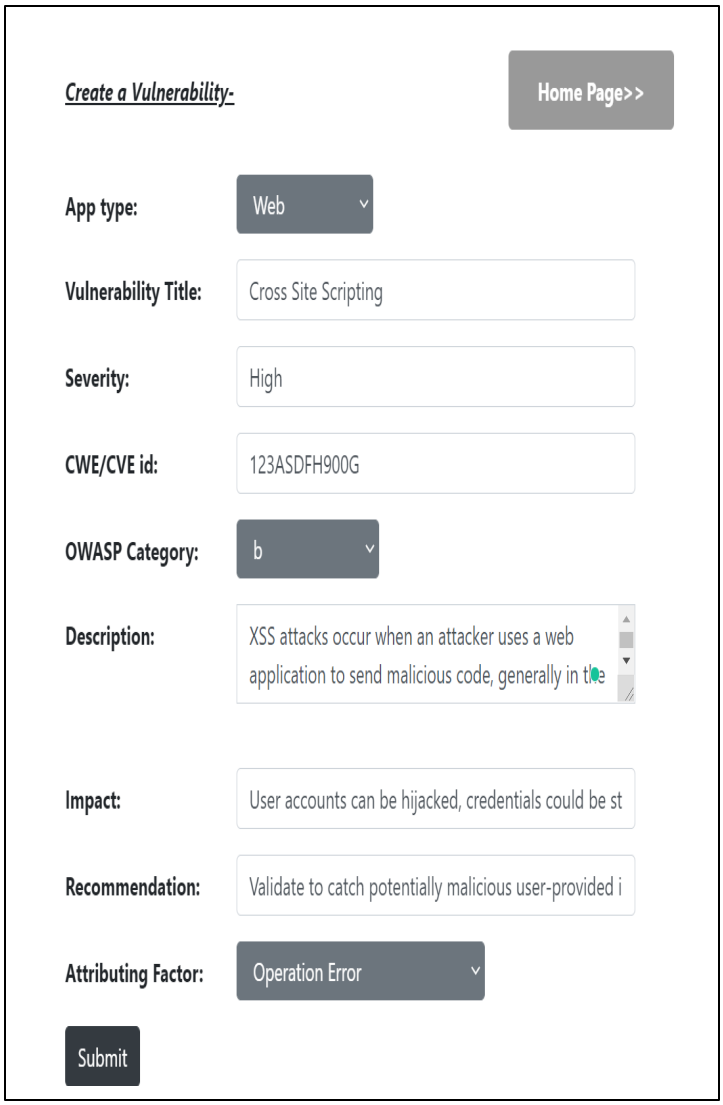


Figure.4.2.2: Entering vulnerability details in the form

5. The Modify Vulnerability page is tested by updating the owasp id.

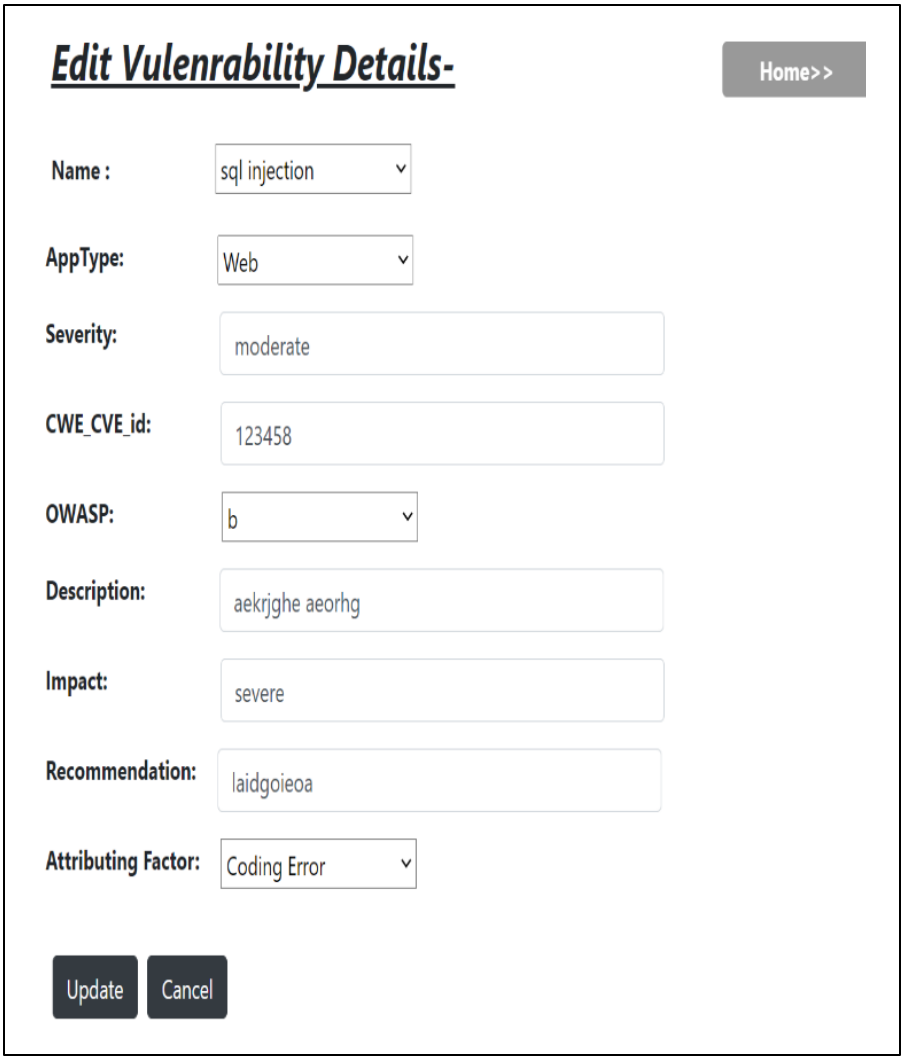


Figure.4.2.3: Updating the vulnerability details

6. List of added projects are displayed in the projects list page.

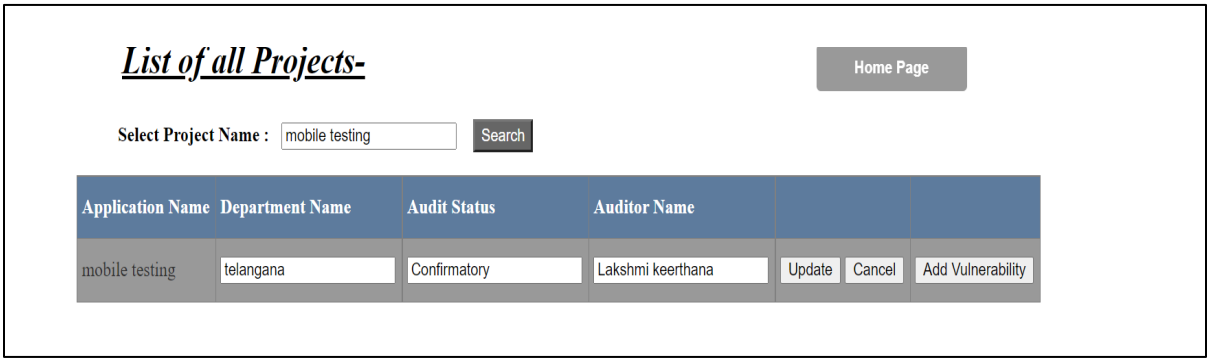
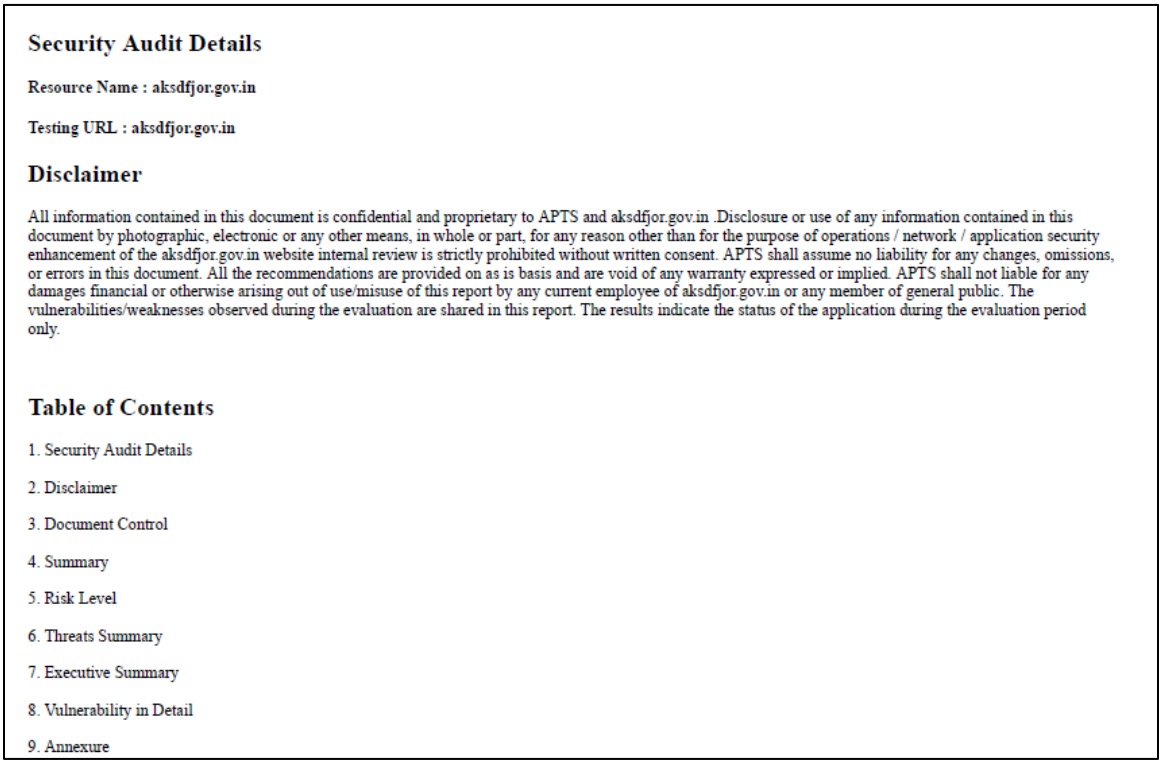


Figure 4.2.4: Updating Project details before adding vulnerabilities

7. From the project list page invoice project is selected to add vulnerabilities.
8. SQL Injection and one more vulnerability is selected to add into the project report.
9. Some details are modified in the selected vulnerabilities and added crawled urls
10. After all this Generate Report is clicked to Generate the Report.
11. Report also consists of a pie chart that shows the Vulnerabilities based on their risk level and overall threat level is displayed according to the pie chart.



Security Audit Details

Resource Name : aksdfjor.gov.in

Testing URL : aksdfjor.gov.in

Disclaimer

All information contained in this document is confidential and proprietary to APTS and aksdfjor.gov.in .Disclosure or use of any information contained in this document by photographic, electronic or any other means, in whole or part, for any reason other than for the purpose of operations / network / application security enhancement of the aksdfjor.gov.in website internal review is strictly prohibited without written consent. APTS shall assume no liability for any changes, omissions, or errors in this document. All the recommendations are provided on as is basis and are void of any warranty expressed or implied. APTS shall not liable for any damages financial or otherwise arising out of use/misuse of this report by any current employee of aksdfjor.gov.in or any member of general public. The vulnerabilities/weaknesses observed during the evaluation are shared in this report. The results indicate the status of the application during the evaluation period only.

Table of Contents

1. Security Audit Details
2. Disclaimer
3. Document Control
4. Summary
5. Risk Level
6. Threats Summary
7. Executive Summary
8. Vulnerability in Detail
9. Annexure

Figure 4.2.5: Generated Report

Summary

IP Address of the Server : 10.0.9.200

Operating System : windows

Application Running : j2ee

Web Server : tomcat

Port Scanned : 8000

Risk Level

High

One or more High severe vulnerabilities are discovered in the website and thus the overall threat level is rated as High.

Threats Summary

Total Threats Found : 4

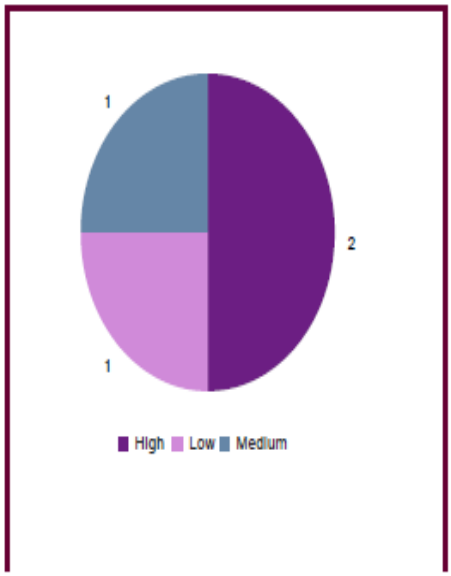


Figure 4.2.6: Generated Report

Executive Summary

AndhraPradesh Technology Services(APTS) has conducted application security assessment on aksdfjor.gov.in application from 15-06-2021 to 30-06-2021. The vulnerabilities/weaknesses observed during the evaluation are given below. The results indicate the status of the application during the evaluation period only.

Overall Threat Level High

SNO	Vulnerability Discovered At	VulnerabilityName	Recommendation	Severity
1	www.dgfdhgf.com	Sql Injection	euiryuiryeiurhfksjdfkjbcxzncxmznclsdldhfjdb	High
2	www.dgfdhgf.com	jghjhgjh	jegekdjgbmncbmchdgiuysdjh	Low
3	www.jkhfkdjgfj.com	Vuln3	vnbvcbjcvbfdjgbfdghfdjghfkjgf	Medium
4	www.kdhfskfh.com	Sql Injection	euiryuiryeiurhfksjdfkjbcxzncxmznclsdldhfjdb	High

Figure 4.2.7: Generated Report

Vulnerabilities in Detail

1 Sql Injection High

Affected URL : www.dgfdhgf.com

OWASP : b

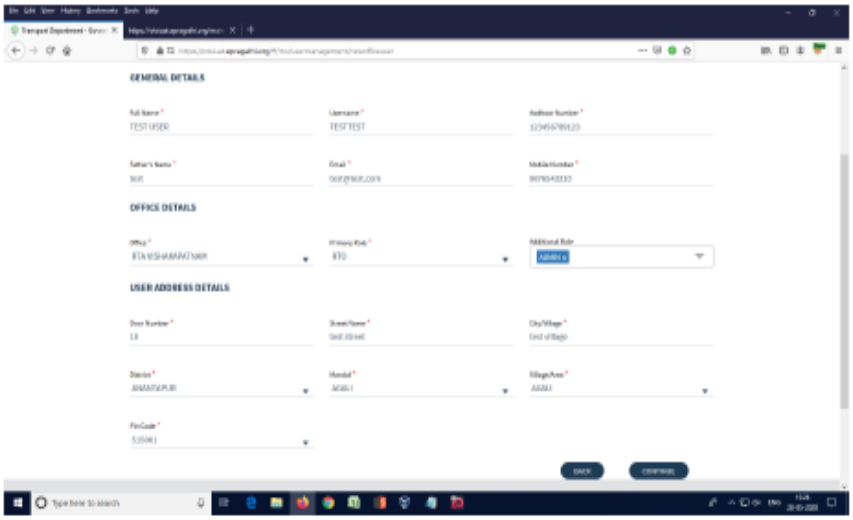
1.1 Description :

dkmfkdfhjefhlncxmncznkfjdghj

1.2 Proof of Concept :

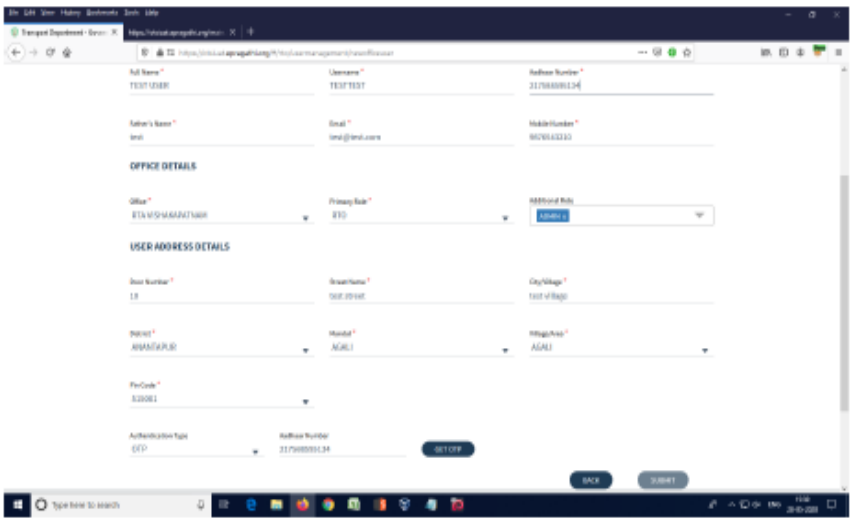
1.2.1 Step 1

gfghfgdrtereytfhgkjhkjhuyugygjhgjhb



1.2.2 Step 2

tryuwertwyertiweyreoituwerwioeyrowuiry



1.3 Recommendation

euiryuiryeiurhfksjdfkjbcxznczxmznclsdldhfjdb

Figure 4.2.8: Generated Report

2 jghjhgjh Low

Affected URL : www.dgfdhgf.com

OWASP : a

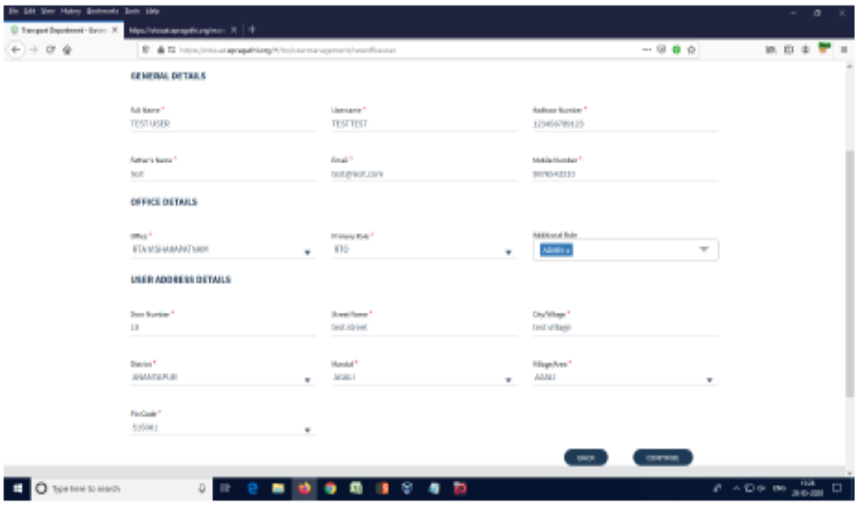
2.1 Description :

kajhdskafsufkzjvbxmvbxdzgu

2.2 Proof of Concept :

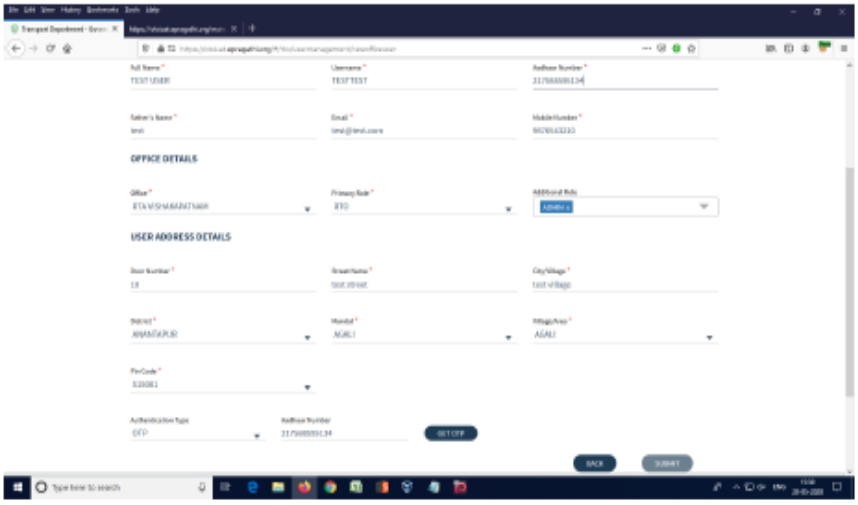
2.2.1 Step 1

djfhdkjghkdjghf



2.2.2 Step 2

yjsiutuiydkghlfkjhgklm



2.3 Recommendation

jegekdjgbmncbmchdgiuysdjh

Figure 4.2.9: Generated Report

Data in database

Id	Application_Name	Department_Name	Category_of_organization	Sector_of_organization	Type_of_Audit
1	report generator	APTS	Central Ministry/ Department	Defence	Website /Web application audit
2	invoice	apts	State Government Department	Information & Broadcasting	Website /Web application audit
3	invoice	apts	State Government Department	Defence	Website /Web application audit

Reason_for_Audit	URL	TestedAppVersion	SPOC_Details
Periodic Audit as per Policy of organization	www.report.in	12.0.1	rdfghujklsdfghjk
Regulatory directions	www.invoice.com	1.2.3	hithusiskfdgfvhrthanaiamfine
Periodic Audit as per Policy of organization	www.invoice.com	1.2.3	sdfghjkl;ll,m,nbvgcfdsawsdetfgyhuji

StartDate	EndDate	AuditStatus	AuditorName	State
2021-05-04	2021-05-19	Initial	keerthana	Andhra Pradesh
2021-05-05	2021-05-12	Initial	lasya	Andhra Pradesh
2021-05-05	2021-05-12	Initial	lasya	Andhra Pradesh

Figure 4.2.10: Project Details

	Name	AppType	Severity	CWE_CVE_id	OWASP	Description	Impact	Recommendation	Attributing_Factor
1	SQLInjection	Mobile	High	123a45	b	fgvhjnmk	xcvbn	dcfghjn	Coding Error

Figure 4.2.11: Vulnerability Details

1	report generator	SQLInjection	Initial	High	fgvhjnmk	xcvbn	dcfghjn	Mobile	123a45	b	Coding Error
2	report generator	SQLInjection	Initial	High	fgvhjnmk	xcvbn	dcfghjn	Mobile	123a45	b	Coding Error

Figure 4.2.12: Project Vulnerability Details

	ProjectName	VulnerabilityName	ObservationStep	ImageName	Image
1	invoice	SQLInjection	that jhggj jvjhvjh jhbkbk jbbknlkkl	Love-What-You-Do-wallpaper-wpt1006784.jpg	0xFFD8FFE1001845786966000049492A0008000000000000...

Figure 4.2.13: Observations Details

	Application_Name	Audit_Status	Auditor_Name	Start_Date	End_Date	Retest
1	invoice	Initial	lasya	2021-05-05	2021-05-12	0
2	NULL	Confirmatory	lasya	2021-05-11	2021-05-19	1
3	invoice	Confirmatory	dhruthi	2021-05-12	2021-05-19	1

Figure 4.2.14: Retest Details

4.3 Observations from work

Developing a Report Generation application aids the process of Generating Reports. It makes the process easy and consumes less time. It can also help the organization get rid of grammatical errors that occur during report generation. It helps to organize the data related to the projects in a systematic format. It also helps to get a summary of the projects done and those that are yet to be completed.

CHAPTER-5
CONCLUSION AND FUTURE STUDY

5.1.Conclusion

The idea of this project is to create a web application that automates the process of report generation after security audit. The web application is developed using .Net framework for adding the details the application and its vulnerabilities and in the backend, SQL Server is used to store and fetch the details of the vulnerabilities in the application .Finally a detailed report with the vulnerabilities present in the application is generated .This application helps in reducing human effort and time taken to generate reports. It also helps in reducing grammatical errors in the report.

References

[1] c-sharpcorner.com/UploadFile/225740/introduction-of-session-in-Asp-Net/

[2] https://www.codeproject.com/Articles/8055/Transferring-page-values-to-another-page

[3] http://asp.net-informations.com/gridview/gridview-operations.html

[4] https://www.infoworld.com/article/3605276/how-to-create-pdf-documents-in-aspnet-core-5.html

[5] https://www.thebestcsharpprogrammerintheworld.com/2018/08/20/how-to-create-a-pie-chart-using-asp-net-and-c/

[6] https://www.codemag.com/article/0603051/Using-the-Ajax.NET-Framework

Appendix

HomePage.html

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Homepage.aspx.cs" Inherits="WebApplicationActivity.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

</head><body>

<h1>   Report Generator</h1>

<page size="A4"></page>

<page size="A4" layout="portrait"></page>

<form id="form1" runat="server">

<asp:Button ID="Button1" runat="server" Height="50px" Text="Create Vulnerability" Width="200px" OnClick="Button1_Click" BackColor="#003366" ForeColor="Red" Font-Bold="true"/>

<asp:Button ID="Button2" runat="server" Height="50px" Text="Modify Vulnerability" Width="200px" OnClick="Button2_Click" BackColor="#003366" ForeColor="Red" Font-Bold="true"/>

 <asp:Button ID="Button3" runat="server" Height="50px" Text="Create Project" Width="200px" OnClick="Button3_Click" BackColor="#003366" ForeColor="Red" Font-Bold="true"/>

<asp:Button ID="Button4" runat="server" Height="50px" Text="Modify Project" Width="200px" BackColor="#003366" ForeColor="Red" OnClick="Button4_Click" Font-Bold="true"/>

<asp:Button ID="Button5" runat="server" Height="50px" Text="All Projects" Width="200px" BackColor="#003366" ForeColor="Red" Font-Bold="true" OnClick="Button5_Click"/>

<asp:Button ID="Button6" runat="server" Height="50px" Text="Generate Report" Width="200px" BackColor="#003366" ForeColor="Red" Font-Bold="true" OnClick="Button6_Click" />

 </form></body></html>

HomePage.aspx.cs

```
using System;
namespace WebApplicationActivity
{
   public partial class WebForm1 : System.Web.UI.Page
   {
      protected void Button1_Click(object sender, EventArgs e)
      {
         Response.Redirect("createvulnerability.aspx");
      }
```

```
    protected void Button2_Click(object sender, EventArgs e)
    {
        Response.Redirect("modifyvulnerability.aspx");
    }
    protected void Button3_Click(object sender, EventArgs e)
    {
        Response.Redirect("createproject.aspx");
    }

    protected void Button4_Click(object sender, EventArgs e)
    {
        Response.Redirect("modifyproject.aspx");
    }
    protected void Button5_Click(object sender, EventArgs e)
    {
        Response.Redirect("ProjectSummary.aspx");
    }
    protected void Button6_Click(object sender, EventArgs e)
    {
        Response.Redirect("add_vulnerabilities.aspx");
    }
  }
}
```

Report.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="ReportPDF.aspx.cs"
Inherits="WebApplicationActivity.ReportPDF" %>

<%@ Register assembly="AjaxControlToolkit" namespace="AjaxControlToolkit"
tagprefix="ajaxToolkit" %>
```

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

   <title></title>

</head>

<body>

   <form id="form1" runat="server">

            <asp:Label ID="ProjName" runat="server" Text="Label" Visible="False"></
asp:Label>

      <h2 style="background-color:beige; width:1500px">Security Audit Details</h2>

      <h4>Resource Name : <TextBox ID="URL1" runat="server"></TextBox></h4>

      <h4>Testing URL : <TextBox ID="URL2" runat="server"></TextBox></h4>


      <h2 style="background-color:beige; width:1500px">Disclaimer</h2>

        <p>All information contained in this document is confidential and proprietary to
APTS and  <TextBox ID="URL3" runat="server"></TextBox> .Disclosure or use of any
information contained in this document by photographic,

electronic or any other means, in whole or part, for any reason other than for the purpose
of operations

/ network / application security enhancement of the  <TextBox ID="URL4"
runat="server"></TextBox>  website internal

review is strictly prohibited without written consent.


APTS shall assume no liability for any changes, omissions, or errors in this document.
All the recommendations are provided on as is basis and are void of any warranty
expressed or implied. APTS


shall not liable for any damages financial or otherwise arising out of use/misuse of this
report by any
```

current employee of <TextBox ID="URL5" runat="server"></TextBox> or any member of general public. The vulnerabilities/weaknesses observed during the evaluation are shared in this report. The results indicate the status

of the application during the evaluation period only.

<h2 style="background-color:beige; width:1500px">Table of Contents</h2>

1. Security Audit Details

2. Disclaimer

3. Document Control

4. Summary

5. Risk Level

6. Threats Summary

7. Executive Summary

8. Vulnerability in Detail

9. Annexure

<h2 style="background-color:beige; width:1500px">Summary</h2>

<h4>IP Address of the Server : <TextBox ID="IP" runat="server"></TextBox></h4>

<h4>Operating System : <TextBox ID="OS" runat="server"></TextBox></h4>

<h4>Application Running : <TextBox ID="AppRun" runat="server"></TextBox></h4>

<h4>Web Server : <TextBox ID="ServerName" runat="server"></TextBox></h4>

<h4>Port Scanned : <TextBox ID="Port" runat="server"></TextBox></h4>

<h2 style="background-color:beige; width:1500px">Risk Level</h2>

<TextBox ID="Risk" runat="server" style="color:red"></TextBox>

One or more <TextBox ID="Risk1" runat="server"></TextBox> severe vulnerabilities are discovered in the website and thus the overall threat level is

rated as <TextBox ID="Risk2" runat="server"></TextBox>.

```
<h2 style="background-color:beige; width:1500px">Threats Summary</h2>
<p>Total Threats Found : <textbox id="cnt" runat="server"></textbox></p>
<div style="width: 337px; height: 379px; align-content:center">
<ajaxToolkit:PieChart ID="PieChart1" runat="server" bordercolor="#660033"
borderstyle="Solid" forecolor="White" height="349px" width="330px"
charttitlecolor="Black">
</ajaxToolkit:PieChart>
</div>
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
<div>
<h2 style="background-color:beige; width:1500px">Executive Summary</h2>
<p>AndhraPradesh Technology Services(APTS) has conducted application security assessment on <TextBox ID="URL" runat="server"></TextBox> application from <TextBox ID="startDate" runat="server"></TextBox> to <TextBox ID="EndDate" runat="server"></TextBox>. The vulnerabilities/weaknesses observed during the evaluation are given below. The results indicate the status of the application during the evaluation period only.</p>
</div>
<table style="width: 100%;">
<tr style="background-color:red">
<td style="width:1500px;">Overall Threat Level</td>
```

```
            <td style="width:100px;"><TextBox ID="threat" runat="server"></TextBox>
</td>

        </tr>


    </table>

    <br />


    <div>

        <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
DataSourceID="SqlDataSource1">


         <Columns>

            <asp:TemplateField HeaderText="SNO">

               <ItemTemplate>  <%#Container.DataItemIndex+1 %> </ItemTemplate>

            </asp:TemplateField>

              <asp:BoundField DataField="Affected_URL" HeaderText="Vulnerability
Discovered At" SortExpression="Vulnerability Discovered At" />

                                    <asp:BoundField  DataField="VulnerabilityName"
HeaderText="VulnerabilityName" SortExpression="VulnerabilityName" />

                                     <asp:BoundField  DataField="Recommendation"
HeaderText="Recommendation" SortExpression="Recommendation" />

                          <asp:BoundField DataField="Severity" HeaderText="Severity"
SortExpression="Severity" />


         </Columns>


       </asp:GridView>

                          <asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$ ConnectionStrings:ReportGenerator %>"
SelectCommand="SELECT [VulnerabilityName], [Recommendation], [Severity],
```

```
[Affected_URL] FROM [project_Vulnerabilities] WHERE ([Project_Name] = @Project_Name)">
            <SelectParameters>
                    <asp:ControlParameter ControlID="ProjName" Name="Project_Name" PropertyName="Text" Type="String" />
            </SelectParameters>
        </asp:SqlDataSource>
        <br />
        <h2>Vulnerabilities in Detail</h2>
      </div>
      <br />

   </form>



</body>
</html>
```

Report.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Text;
using System.Web.UI.WebControls;
```

```
namespace WebApplicationActivity
{
   public partial class ReportPDF : System.Web.UI.Page
   {
      public object Status { get; private set; }

      protected void Page_Load(object sender, EventArgs e)
      {
         ProjName.Text = (string)Session["Projectname"];

         //ProjName.Text = "invoice";
                                                    string strcon0 =
ConfigurationManager.ConnectionStrings["con"].ConnectionString;

         SqlConnection con3 = new SqlConnection(strcon0);
         con3.Open();
            SqlCommand cmd3 = new SqlCommand("select Severity,count(Severity) from
project_Vulnerabilities where Project_Name=@Name group by Severity",con3);
         cmd3.Parameters.AddWithValue("@Name", ProjName.Text);
         DataTable dt2 = new DataTable();
         dt2.Load(cmd3.ExecuteReader());
         int c = 0;
         int i = 0;
         string risk="";
         foreach (DataRow dr1 in dt2.Rows)

         {  if(i<Convert.ToDecimal(dr1[1]))
            {
```

```
            i = Convert.ToInt32(dr1[1]);
            risk = (string)dr1[0];
        }
        c = c +Convert.ToInt32(dr1[1]);

        PieChart1.PieChartValues.Add(new AjaxControlToolkit.PieChartValue

        {

            Category = dr1[0].ToString(),
            Data = Convert.ToDecimal(dr1[1])



        }) ;


    }
    cnt.InnerText = Convert.ToString(c);
    Risk.InnerText = risk;
    Risk1.InnerText = risk;
    Risk2.InnerText = risk;
    threat.InnerText = risk;

    con3.Close();

    SqlConnection con0 = new SqlConnection(strcon0);
    con0.Open();
            SqlCommand cmd2 = new SqlCommand("select * from Project where
Application_Name=@Name", con0);
```

```
cmd2.Parameters.AddWithValue("@Name", ProjName.Text);
SqlDataReader DR1 = cmd2.ExecuteReader();
if(DR1.Read())
{
   URL.InnerText=DR1.GetString(7);
   URL1.InnerText = DR1.GetString(7);
   URL2.InnerText = DR1.GetString(7);
   URL3.InnerText = DR1.GetString(7);
   URL4.InnerText = DR1.GetString(7);
   URL5.InnerText = DR1.GetString(7);
   IP.InnerText = DR1.GetString(15);
   OS.InnerText = DR1.GetString(16);
   AppRun.InnerText = DR1.GetString(17);
   ServerName.InnerText = DR1.GetString(18);
   Port.InnerText = Convert.ToString(DR1.GetInt32(19));
   var from= Convert.ToString(DR1.GetDateTime(10)).Split(' ');
   startDate.InnerText = from[0];
   var to= Convert.ToString(DR1.GetDateTime(11)).Split(' ');
   EndDate.InnerText = to[0];

}
   con0.Close();
                                                    string strcon =
ConfigurationManager.ConnectionStrings["con"].ConnectionString;
SqlConnection con = new SqlConnection(strcon);
con.Open();
```

```
                                                    string strcon1 =
ConfigurationManager.ConnectionStrings["con"].ConnectionString;
        SqlConnection con1 = new SqlConnection(strcon1);
        con1.Open();



        //Status.Text = (string)Session["status"];

          SqlCommand cmd1 = new SqlCommand("select * from project_Vulnerabilities
where Project_Name=@Name", con);
        cmd1.Parameters.AddWithValue("@Name", ProjName.Text);
        //cmd1.Parameters.AddWithValue("@Status", Status.Text);

        DataTable dt = new DataTable();
        dt.Load(cmd1.ExecuteReader());
        con.Close();
        int VulNo = 0;
        foreach (DataRow dr in dt.Rows)
        {

          VulNo = VulNo + 1;
          string VulnName = (string)dr[1];

          Literal l1 = new Literal();
          DataTable dt1 = new DataTable();
          StringBuilder sb = new StringBuilder();

                dt1.Columns.AddRange(new DataColumn[3] { new DataColumn("Id",
typeof(int)),
```

```
            new DataColumn("Name", typeof(string)),
            new DataColumn("Severity",typeof(string)) });

        dt1.Rows.Add(VulNo, (string)dr[1], (string)dr[3]);

        sb.Append("<table cellpadding='5' cellspacing='0' >");

            sb.Append("<tr style='background-color:red'>");

              sb.Append("<td style='width:30px;'>" + dt1.Rows[0]["Id"].ToString() + "</
td>") ;
             sb.Append("<td style='width:1500px;'>" + dt1.Rows[0]["Name"].ToString() +
"</td>");
            sb.Append("<td style='width:100px;'>" + dt1.Rows[0]["Severity"].ToString() +
"</td>");

        sb.Append("</tr>");


        //Table end.
        sb.Append("</table>");
        l1.Text = sb.ToString();

        Label lbl = new Label();
        lbl.Width = 150;
        lbl.Height = 25;
        lbl.Text = "<br>" + "Affected URL :";

        Label lbl1 = new Label();
        lbl1.Text = (string)dr[12] + "<br>";
```

```
Label lbl2 = new Label();
lbl2.Width = 150;
lbl2.Height = 25;
lbl2.Text = "<br>" + "OWASP :";

Label lbl3 = new Label();
lbl3.Text = (string)dr[9] ;

string DNO = Convert.ToString(VulNo) + ".1";
Label lbl4 = new Label();
lbl4.Width = 1800;
lbl4.Height = 20;
lbl4.BackColor = System.Drawing.Color.Beige;
lbl4.Text =  DNO + " " + "Description :";

Label lbl5 = new Label();
lbl5.Text = "<br>" + (string)dr[4] + "<br>";

string PNO = Convert.ToString(VulNo) + ".2";
Label lbl6 = new Label();
lbl6.Width = 1800;
lbl6.Height = 25;
lbl6.BackColor = System.Drawing.Color.Beige;
lbl6.Text = PNO + " " + "Proof of Concept :" + "<br>";

form1.Controls.Add(l1);
form1.Controls.Add(lbl);
```

```
form1.Controls.Add(lbl1);
form1.Controls.Add(lbl2);
form1.Controls.Add(lbl3);
form1.Controls.Add(lbl4);
form1.Controls.Add(lbl5);
form1.Controls.Add(lbl6);

    SqlCommand cmd = new SqlCommand("select * from Observation where ProjectName=@Proj and VulnerabilityName=@Vuln", con1);
form1.Controls.Add(lbl1);
cmd.Parameters.AddWithValue("@Proj", ProjName.Text);
cmd.Parameters.AddWithValue("@Vuln", VulnName);
SqlDataReader DR = cmd.ExecuteReader();
int count = 0;
while (DR.Read())
{
   count = count + 1;
   string SNO = PNO + "." + Convert.ToString(count);
   Label lbl7 = new Label();
   lbl7.Text = "<br>" + SNO + " Step " + count + "<br>";

   Label lbl0 = new Label();
   lbl0.Text = "<br>" + (string)DR.GetValue(2) + "<br>";

   Label lbl10 = new Label();
   lbl10.Text = "<br>" ;

   Image img = new Image();
   byte[] bytes = (byte[])DR.GetValue(4);
```

```
    string strBase64 = Convert.ToBase64String(bytes);
    img.ImageUrl = "data:Image/png;base64," + strBase64;
    img.Width = 800;
    img.Height = 400;


    form1.Controls.Add(lbl7);
    form1.Controls.Add(lbl0);
    form1.Controls.Add(lbl10);
    form1.Controls.Add(img);

}

Label l = new Label();
l.Text = "<br>";

string RNO = Convert.ToString(VulNo) + ".3";
Label lbl8 = new Label();
lbl8.Width = 1800;
lbl8.Height = 25;
lbl8.BackColor = System.Drawing.Color.Beige;
lbl8.Text = RNO + " " + "Recommendation" + "<br>";

Label lbl9 = new Label();
lbl9.Text = "<br>" + (string)dr[6] + "<br>";

form1.Controls.Add(l);
form1.Controls.Add(lbl8);
```

```
            form1.Controls.Add(lbl9);
        }
        Label l2 = new Label();
        l2.Text = "Annexure";

        Label l3 = new Label();
        l3.Text = "<br>" + "URL's Crawled during Audit";

        Label l4 = new Label();
        l4.Text = (string)Session["URL"];



        form1.Controls.Add(l2);
        form1.Controls.Add(l3);
        form1.Controls.Add(l4);


    }
  }
}
```

SQL Queries:

Project Table

```
CREATE TABLE [dbo].[Project](
        [Id] [int] IDENTITY(1,1) NOT NULL,
        [Application_Name] [varchar](100) NOT NULL,
        [Department_Name] [varchar](50) NOT NULL,
        [Category_of_organization] [varchar](50) NULL,
```

```
	[Sector_of_organization] [varchar](50) NULL,
	[Type_of_Audit] [varchar](50) NULL,
	[Reason_for_Audit] [varchar](max) NULL,
	[URL] [varchar](max) NULL,
	[TestedAppVersion] [varchar](max) NULL,
	[SPOC_Details] [varchar](max) NULL,
	[StartDate] [date] NULL,
	[EndDate] [date] NULL,
	[AuditStatus] [varchar](50) NULL,
	[AuditorName] [varchar](max) NULL,
	[State] [varchar](50) NULL,
	[IP_Address] [varchar](50) NULL,
	[OS] [varchar](50) NULL,
	[AppRunning] [varchar](50) NULL,
	[WebServer] [varchar](50) NULL,
	[PortScanned] [int] NULL,
 CONSTRAINT [PK_Project] PRIMARY KEY CLUSTERED
(
	[Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

Vulnerability Table

```
CREATE TABLE [dbo].[Vulnerability](
```

```
	[Name] [varchar](50) NOT NULL,
	[AppType] [varchar](10) NULL,
	[Severity] [varchar](20) NULL,
	[CWE_CVE_id] [varchar](50) NULL,
	[OWASP] [varchar](50) NULL,
	[Description] [varchar](max) NULL,
	[Impact] [varchar](max) NULL,
	[Recommendation] [varchar](max) NULL,
	[Attributing_Factor] [varchar](50) NULL,
 CONSTRAINT [PK_Vulnerability] PRIMARY KEY CLUSTERED
(
	[Name] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

Project Vulnerabilities Table

```
CREATE TABLE [dbo].[project_Vulnerabilities](
	[Project_Name] [varchar](max) NULL,
	[VulnerabilityName] [varchar](max) NULL,
	[Audit_Status] [varchar](max) NULL,
	[Severity] [varchar](max) NULL,
	[Description] [varchar](max) NULL,
	[Impact] [varchar](max) NULL,
	[Recommendation] [varchar](max) NULL,
```

```
    [AppType] [varchar](max) NULL,
    [CWE_CVE_id] [varchar](max) NULL,
    [OWASP] [varchar](max) NULL,
    [Attributing_Factor] [varchar](max) NULL,
    [Vulnerability_Status] [varchar](50) NULL,
    [Affected_URL] [varchar](100) NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

Observation Table

```
CREATE TABLE [dbo].[Observation](
    [ProjectName] [varchar](50) NULL,
    [VulnerabilityName] [varchar](50) NULL,
    [ObservationStep] [varchar](max) NULL,
    [ImageName] [varchar](max) NULL,
    [Image] [varbinary](max) NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

Retest Table

```
CREATE TABLE [dbo].[Retest](
    [Application_Name] [varchar](50) NULL,
    [Audit_Status] [varchar](20) NULL,
    [Auditor_Name] [varchar](50) NULL,
    [Start_Date] [date] NULL,
    [End_Date] [date] NULL,
```

```
    [Retest] [int] NULL
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Retest] ADD CONSTRAINT [DF_Retest_Retest] DEFAULT ((0)) FOR [Retest]
GO
```

Insert Project Details Stored Procedure

```
CREATE procedure [dbo].[create_project](@Application_Name varchar(100),@Department_Name varchar(50),@Category_of_organization varchar(50),@Sector_of_organization varchar(50),@Type_of_Audit varchar(50),@Reason_for_Audit varchar(max),@State varchar(50),@URL varchar(max),@TestedAppVersion varchar(max),@SPOC_Details varchar(max),@StartDate date,@EndDate date,@AuditStatus varchar(50),@AuditorName varchar(max), @IPAddress varchar(50),@OS varchar(50), @AppRun varchar(50),@WebServer varchar(50),@Portno int )
as
begin
insert into dbo.Project(Application_Name,Department_Name,Category_of_organization,Sector_of_organization,Type_of_Audit,Reason_for_Audit,State,URL,TestedAppVersion,SPOC_Details,StartDate,EndDate,AuditStatus,AuditorName,IP_Address,OS,AppRunning,WebServer,PortScanned)values(@Application_Name,@Department_Name,@Category_of_organization,@Sector_of_organization,@Type_of_Audit,@Reason_for_Audit,@State,@URL,@TestedAppVersion,@SPOC_Details,@StartDate,@EndDate,@AuditStatus,@AuditorName,@IPAddress,@OS,@AppRun,@WebServer,@Portno)
end
GO
```

Insert Vulnerability Details Stored Procedure

```
CREATE procedure [dbo].[create_vulnerability](@Name varchar(50),@AppType varchar(10),@Severity varchar(20),@CWE_CVE_id varchar(20),@OWASP
```

varchar(50),@Description varchar(max),@Impact varchar(max),@Recommendation varchar(max),@Attributing_Factor varchar(50))

as

begin

insert into Vulnerability(Name,AppType,Severity,CWE_CVE_id,OWASP,Description,Impact,Recommendation,Attributing_Factor) values(@Name,@AppType,@Severity,@CWE_CVE_id,@OWASP,@Description,@Impact,@Recommendation,@Attributing_Factor)

end

GO

Store Project Vulnerability Details Stored Procedure

CREATE PROCEDURE [dbo].[StoreProjVulner](@ProjName varchar(Max),@Name varchar(max),@ProjStatus varchar(max),@Severity varchar(max),@Description varchar(max),@Impact varchar(max),@Recommendation varchar(max),@AppType varchar(max),@CWE_CVE_id varchar(max),@OWASP varchar(max),@Attributing_Factor varchar(max))

as

Begin

insert into project_Vulnerabilities(Project_Name,VulnerabilityName,Audit_Status,Severity,Description,Impact,Recommendation,AppType,CWE_CVE_id,OWASP,Attributing_Factor) values(@ProjName,@Name,@ProjStatus,@Severity,@Description,@Impact,@Recommendation,@AppType,@CWE_CVE_id,@OWASP,@Attributing_Factor)

end

GO

Insert Uploaded Image Details Stored Procedure

Create procedure [dbo].[uploadimage](@ProjName varchar(50),@VulnName varchar(50), @obs varchar(MAX),

@Name varchar(MAX),@ImageData varbinary(max))

as

```
begin

insert into Observation

values(@ProjName,@VulnName,@obs,@Name,@ImageData)

end

GO
```

Insert Retest Details Stored Procedure

```
create procedure [dbo].[RetestInsert](@Name varchar(50),@StartDate date, @EndDate date,@AuditStatus varchar(20), @Auditor varchar(50))

as

Begin

insert into Retest(Application_Name,Audit_Status,Auditor_Name,Start_Date,End_Date) values (@Name,@AuditStatus , @Auditor, @StartDate, @EndDate)

end

GO
```