Swathi Selvakumaran

Introduction

Multi-Insurance Management System is a one-point platform for users in need to manage all their insurance policies, such as vehicles, buildings, health, and many more. The multi-insurance management system helps to manage and keep track of various insurance policies the user has taken by maintaining their data securely. This software provides the ability to manage insurances, track the status of insurance, notifying insurance expiry and many more.

Application Scope

Multi-Insurance Management System’s main target audience is the users oof the application. The Scope of Multi Insurance Management System is:

1. Allow the user to register in the system.
2. Allow the user to add and remove new insurance availed.
3. Keep track of the insurance status and the insurance policy availed.
4. Renew/Terminate the insurance currently availed.
5. The system should verify the authenticity of the user by verifying the user email.
6. The system should notify the user about insurance expiry dates and user creation details.

Design Overview

System level architecture

Diagram

Description automatically generated

The diagram above demonstrates the high-level system architecture of the Multi Insurance Management System. The following process takes place in the system:

1. The User client initiates any request to the server which is added to the request queue present in the server.
2. The Server then operates on each request queue based on the request requirement.
3. If the request is to create a user/login to the application, then the server directs the request to Authentication service where both the process takes place.
4. If the request is to track insurance status/view insurance policy/ register insurance, the server executes the request and sends or receives the required data to the database.
5. If the server has to notify the user regarding any request, then it is passed through the Notification Service.

UML Diagram:

The UML Diagram is used to demonstrate the various class and the methods that will be defined in the system. Here it is evident that there will four classes present namely: InsuranceDetails, User, System and SendEmail. The relationship between the classes is defined below:

Insurance Details and User: These classes are closely coupled as the user will continuously fetch details of the insurance availed.

Insurance Details and System: These classes are closely coupled as the system keeps track of the insurance expiry everyday to track the status of the insurance and notify the user about insurance expiry date.

User and System: These classes are closely coupled as the system keeps track of all the users and the sessions in use. The system also ensures the sessions that are active are secure. Notifications are sent to the user through the system.

There is no dependency between the SendEmail class and InsuranceDetails/User class as there is no communication required.

Diagram

Description automatically generated

Interface Design:

For each subsystem , its interface is designed and documented. This interfaces will help in the communication between the subsystems and helps retrieve data from database as well.

The User can:

* registerUser()
* validateUser()
* changePassword()
* deleteUser()
* updateUserDetails()
* getUserPriveledges()

The System can:

* validateUser()
* notifyUserCreation()
* notifyInsuranceExpiry()

The user can obtain insurance details through:

* registerInsurance()
* modifyInsurance()
* deleteInsurance()
* insuranceExpiryDuration()
* renewOrTerminateInsurance()
* searchInsurance()
* insuranceTypes()
* viewInsurance()

The notification System will sendEmail().

Data Flow Design:

1. Registering a user as part of the web application:

Diagram

Description automatically generated

* User selects register user details and the application presents the web screen with a form asking personal information of the user
* The User enters the valid details and submits the form details.
* The system invokes the User’s email id to send a validation email to the user, this will be used to verify the user authenticity.
* The User receives a notification through which the user is validated.
* After validation the user details are stored in the User Detail Repository.
* The user will be successfully registered into the system and login page is displayed.

1. Login/change password of the user:

Diagram

Description automatically generated

* In the Login page of the application, the user can either login to the system or the user can change the user account password.
* For login: The user enters the user name and password and selects the login option. The user details are verified after comparing with the credentials stored in the User Detail Repository. Once verified, the valid users are logged into the application.
* For change password: The user selects the change password option. The user enter old password and new password. The server verifies the old password with the User Detail Repository and once validated the user password is updated in the database.

1. View the details of availed insurance:

Diagram

Description automatically generated

* Once the user is logged in the user will be able to view the home page with the privileges the user has been assigned.
* The user can navigate to the availed insurance tab to view all the insurance that the user has registered already.
* To view the insurance policy of a specific insurance, the user can select the insurance they would want to view and click on the link that redirects them to the insurance policy page.
* To track the status of a specific insurance, the user can select the insurance they want to view and view the status of the insurance. The user can also terminate/renew the insurance for the next period by selecting the required option.

1. System data flow:

Diagram

Description automatically generated

* The system will verify the new user created and add the user as part of the User Detail repository. Once the user is added to the datatbase a notification is sent to the user intimating the suer creation.
* The system keeps track of the insurance expiry date by regularly checking the database of Insurance Repository. Once the system realizes that the insurance will expire in three days then a notification about the insurance expiry is sent to the user.

**ER Diagram of Multi-Insurance Management System**

The Entity Relationship Diagram represents the model of the Multi-Insurance Management System Entity. The entity-relationship diagram shows all the visual instrument database tables and the relation between the entities. The main entity of Multi-Insurance Management System are User, Insurance, Login and System.

Diagram

Description automatically generated

* A Rectangle is used to represent an Entity.
* An Oval is used to represent either an Entity or a Relationship Attribute.
* A diamond is used to represent a relationship between Entities.

**Session Management:**

Session management manages the session created for each operation during the communication between web application and the user. The communication between the web browser and the user takes place through HTTP or HTTPS. Multiple sessions will be creted at a time and each session has to be handled with efficiency and security. Once a session is created a Session ID, consisting of a token is attached to the request and response sent across. Session cookies contains data that is stored in the cache to increase the efficiency of the web application. For Secure session management the following process should be followed:

1. Set secure HTTP flags on the session cookies.
2. New session cookie should be generated for every interaction.
3. The session ID should be encrypted during transmission.

The session initiation, session process and post session is provided along with the requirements.

**System Requirements:**

**Logging and Error Handling:**

Error handling should be handled carefully by suppressing the exception that has occurred and y having a clean exit from the system. Once an error occurs the error should be notified to the user based on its priority. The error should be logged to provide the administrator with the information on the error scenario that will help the administrator to over come it.

The application should always handle logging without the interference of the server. The developers should make sure that no sensitive data (such as passwords, session information, system details(like IP), etc.) or debugging or stack trace information is included in the logs. Furthermore, logging ought to include information on both successful and unsuccessful security events, with a focus on crucial log event data. The crucial cases that has to be logged are:

* Input validation failures.
* Authentication attempts.
* Access control failures.
* Change in data state.
* Illegal session access attempts.
* System exceptions.
* Administrative functions.

**Encryption/Decryption:**

There are some sensitive information such as user account password, insurance claimed, insurance policy, user personal information stored in the database. These information once available to the hacker the hacker can easily exploit the user information and the web application as the attacker will gain full access to the system by gaining these sensitive information. To prevent this from occurring these sensitive information should be hashed or encrypted before storing in the database. Strong encryption should be used and the attacker should not have access to any keys that can exploit the information.

**Threat Modeling:**

Diagram

Description automatically generated

**Interaction: API Request**

Diagram

Description automatically generated

**1. Spoofing of the SMTP Server External Destination Entity  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Spoofing |
| **Description:** | SMTP Server may be spoofed by an attacker and this may lead to data being sent to the attacker's target instead of SMTP Server. Consider using a standard authentication mechanism to identify the external entity. |
| **Justification:** | Authenticate users and systems and Use Encrypted and Authenticated Protocols |

**2. External Entity SMTP Server Potentially Denies Receiving Data  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Repudiation |
| **Description:** | SMTP Server claims that it did not receive data from a process on the other side of the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data. |
| **Justification:** | Ensure SMTP server is running and log the record of the source, time, and summary of data. |

**3. Data Flow API Request Is Potentially Interrupted  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Denial Of Service |
| **Description:** | An external agent interrupts data flowing across a trust boundary in either direction. |
| **Justification:** | Document the Response Process and log the record of the source, time, and summary of data. |

**Interaction: API Response**

Diagram

Description automatically generated

**4. Spoofing of the SMTP Server External Destination Entity  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Spoofing |
| **Description:** | SMTP Server may be spoofed by an attacker and this may lead to data being sent to the attacker's target instead of SMTP Server. Consider using a standard authentication mechanism to identify the external entity. |
| **Justification:** | Use Encrypted and Authenticated Protocols during communication. |

**5. External Entity SMTP Server Potentially Denies Receiving Data  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Repudiation |
| **Description:** | SMTP Server claims that it did not receive data from a process on the other side of the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data. |
| **Justification:** | Ensure SMTP server is running and log the record of the source, time, and summary of data. |

**6. Data Flow API Response Is Potentially Interrupted  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Denial Of Service |
| **Description:** | An external agent interrupts data flowing across a trust boundary in either direction. |
| **Justification:** | Document the Response Process and log the record of the source, time, and summary of data. |

**Interaction: Cache Search**

Diagram

Description automatically generated

**7. Potential Excessive Resource Consumption for Web Server or HTML5 Local Cache  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Denial Of Service |
| **Description:** | Does Web Server or Local Cache take explicit steps to control resource consumption? Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job. Be careful that your resource requests don't deadlock, and that they do timeout. |
| **Justification:** | Add a limit on the resource used by cache and clear cache at specific intervals. |

**8. Spoofing of Destination Data Store HTML5 Local Cache  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Spoofing |
| **Description:** | Local Cache may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of Local Cache. Consider using a standard authentication mechanism to identify the destination data store. |
| **Justification:** | Use Encrypted and Authenticated Protocols during communication |

**Interaction: JSON Service Request**

Diagram

Description automatically generated

**9. Elevation Using Impersonation  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | Notification Service may be able to impersonate the context of Web Server in order to gain additional privilege. |
| **Justification:** | Assign minimum privileges to the users |

**Interaction: JSON Service Request**

Diagram

Description automatically generated

**10. Elevation Using Impersonation  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | Backend Services - Authentication/Verification/List view may be able to impersonate the context of Web Server in order to gain additional privilege. |
| **Justification:** | Assign minimum privileges to the users |

**Interaction: JSON Service Response**

Diagram

Description automatically generated

**11. Elevation Using Impersonation  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | Web Server may be able to impersonate the context of Notification Service in order to gain additional privilege. |
| **Justification:** | Assign minimum privileges to the users |

**12. Cross Site Scripting  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Tampering |
| **Description:** | The web server 'Web Server' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input. |
| **Justification:** | Sanitize all input and output |

**Interaction: JSON Service Response**

Diagram

Description automatically generated

**13. Cross Site Scripting  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Tampering |
| **Description:** | The web server 'Web Server' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input. |
| **Justification:** | Sanitize all input and output |

**14. Elevation Using Impersonation  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | Web Server may be able to impersonate the context of Backend Services - Authentication/Verification/List view in order to gain additional privilege. |
| **Justification:** | Assign minimum privileges to the users |

**Interaction: Request**

Diagram

Description automatically generated

**15. Elevation Using Impersonation  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | Web Server may be able to impersonate the context of Browser/GUI in order to gain additional privilege. |
| **Justification:** | Assign minimum privileges to the users |

**16. Data Flow Request Is Potentially Interrupted  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Denial Of Service |
| **Description:** | An external agent interrupts data flowing across a trust boundary in either direction. |
| **Justification:** | Document the Response Process and log the record of the source, time, and summary of data. |

**17. Potential Process Crash or Stop for Web Server  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Denial Of Service |
| **Description:** | Web Server crashes, halts, stops or runs slowly; in all cases violating an availability metric. |
| **Justification:** | Crash testing is used to evaluate the crashworthiness of these devices. |

**18. Data Flow Sniffing  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Information Disclosure |
| **Description:** | Data flowing across Request may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations. Consider encrypting the data flow. |
| **Justification:** | Use of encryption protocols |

**19. Potential Data Repudiation by Web Server  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Repudiation |
| **Description:** | Web Server claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data. |
| **Justification:** | Add logs for every transaction taking place |

**20. Cross Site Scripting  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Tampering |
| **Description:** | The web server 'Web Server' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input. |
| **Justification:** | Sanitize all input and output |

**21. Potential Lack of Input Validation for Web Server  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Tampering |
| **Description:** | Data flowing across Request may be tampered with by an attacker. This may lead to a denial of service attack against Web Server or an elevation of privilege attack against Web Server or an information disclosure by Web Server. Failure to verify that input is as expected is a root cause of a very large number of exploitable issues. Consider all paths and the way they handle data. Verify that all input is verified for correctness using an approved list input validation approach. |
| **Justification:** | Validate all the input entered |

**25. Elevation by Changing the Execution Flow in Web Server  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | An attacker may pass data into Web Server in order to change the flow of program execution within Web Server to the attacker's choosing. |
| **Justification:** | Provide minimum privilege to user |

**Interaction: Response**

Diagram

Description automatically generated

**26. External Entity Browser Potentially Denies Receiving Data  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Repudiation |
| **Description:** | Browser/GUI claims that it did not receive data from a process on the other side of the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data. |
| **Justification:** | Log all transaction taking place |

**27. Spoofing of the Browser External Destination Entity  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Spoofing |
| **Description:** | Browser/GUI may be spoofed by an attacker and this may lead to data being sent to the attacker's target instead of Browser/GUI. Consider using a standard authentication mechanism to identify the external entity. |
| **Justification:** | Authenticate users and systems and Use Encrypted and Authenticated Protocols |

**Interaction: SQL Query**

Diagram

Description automatically generated

**29. Spoofing of Destination Data Store SQL Database  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Spoofing |
| **Description:** | Database may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of Database. Consider using a standard authentication mechanism to identify the destination data store. |
| **Justification:** | Authenticate users and systems and Use Encrypted and Authenticated Protocols |

**31. Potential Excessive Resource Consumption for Backend Services or SQL Database  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Denial Of Service |
| **Description:** | Does Backend Services - Authentication/Verification/List view or Database take explicit steps to control resource consumption? Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job. Be careful that your resource requests don't deadlock, and that they do timeout. |
| **Justification:** | Limit the time the resource can be used when the web is unresponsive. |

**Interaction: SQL Response**

Diagram

Description automatically generated

**32. Spoofing of Source Data Store SQL Database  [State: Not Started]  [Priority: High]**

|  |  |
| --- | --- |
| **Category:** | Spoofing |
| **Description:** | Database may be spoofed by an attacker and this may lead to incorrect data delivered to Backend Services - Authentication/Verification/List view. Consider using a standard authentication mechanism to identify the source data store. |
| **Justification:** | Authenticate users and systems and Use Encrypted and Authenticated Protocols |