Watchguard Status and Task Management Tool

Project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering

By

Siddhant Tyagi (191349)

Under the Supervision of

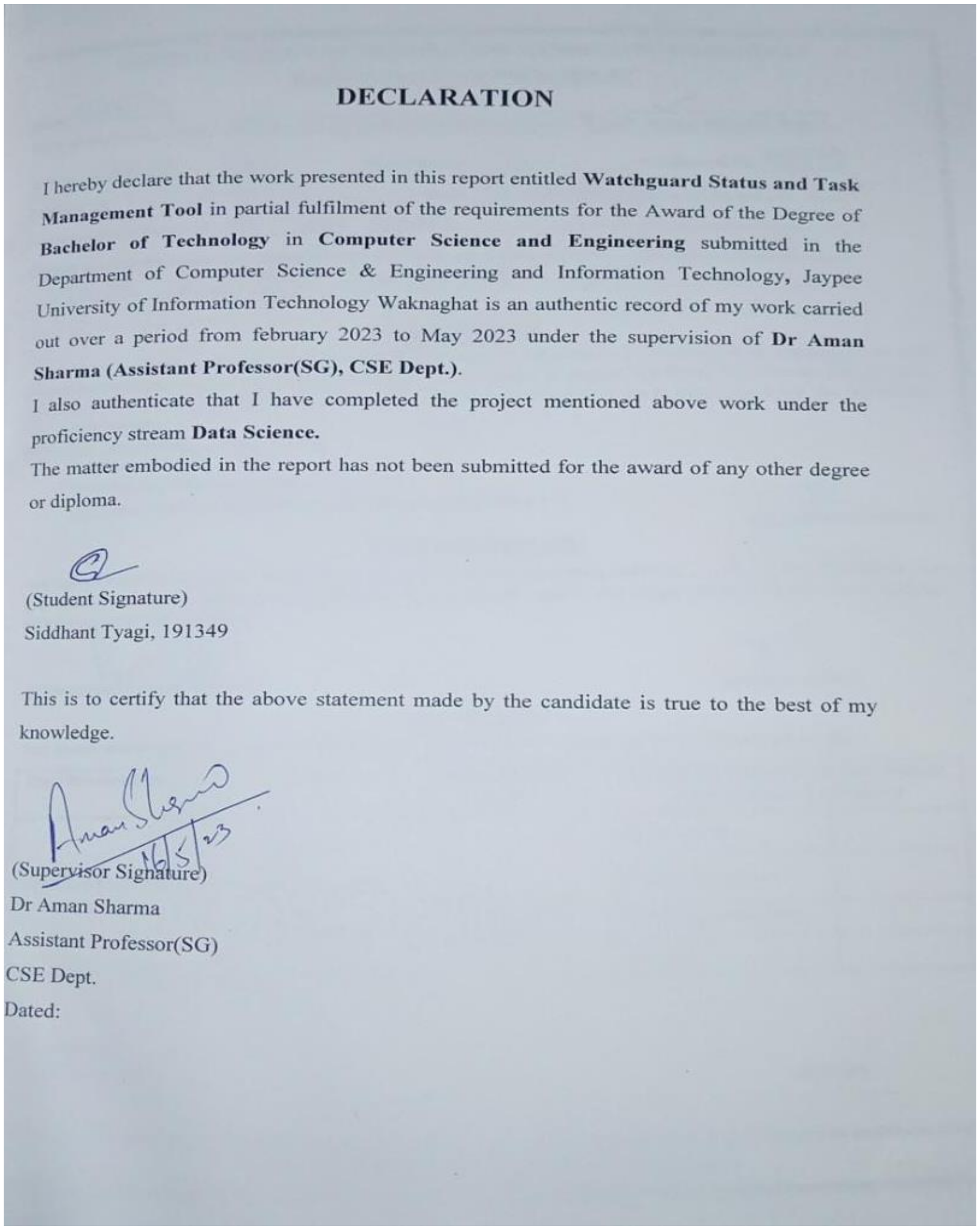
Dr Aman Sharma

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

DECLARATION



DECLARATION

I hereby declare that the work presented in this report entitled **Watchguard Status and Task Management Tool** in partial fulfilment of the requirements for the Award of the Degree of **Bachelor of Technology** in **Computer Science and Engineering** submitted in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my work carried out over a period from february 2023 to May 2023 under the supervision of **Dr Aman Sharma (Assistant Professor(SG), CSE Dept.)**.

I also authenticate that I have completed the project mentioned above work under the proficiency stream **Data Science.**

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

[illegible]

(Student Signature)

Siddhant Tyagi, 191349

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

[illegible]
16/5/23

(Supervisor Signature)

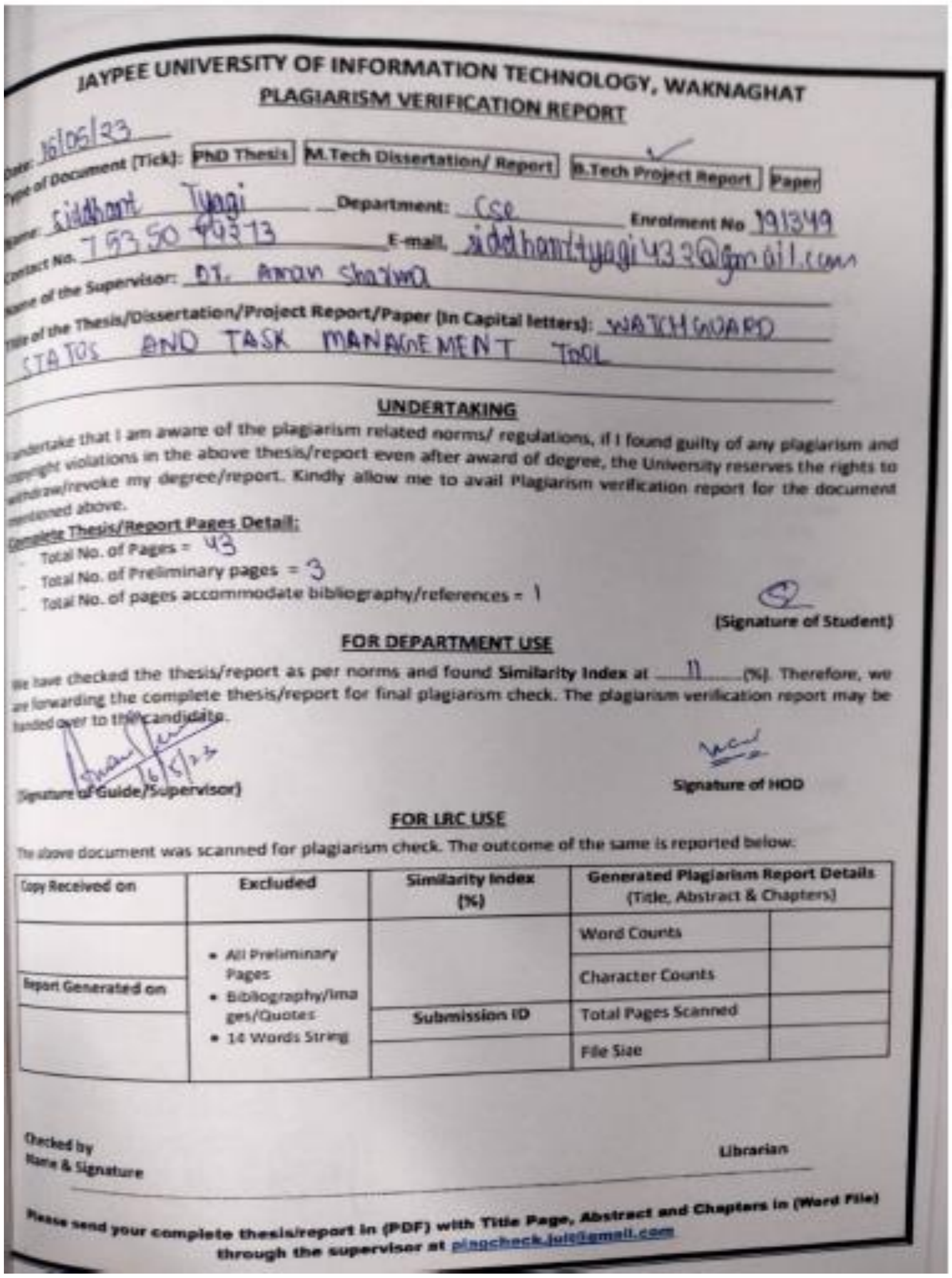
Dr Aman Sharma

Assistant Professor(SG)

CSE Dept.

Dated:

Plagiarism Certificate



JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 16/05/23

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | [✓ B.Tech Project Report] | Paper

Name: Siddhant Tyagi Department: CSE Enrolment No 191349

Contact No. 7 53 50 49313 E-mail. siddhanttyagi433@gmail.com

Name of the Supervisor: Dr. Aman Sharma

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): WATCHGUARD STATUS AND TASK MANAGEMENT TOOL

UNDERTAKING

ndertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and pyright violations in the above thesis/report even after award of degree, the University reserves the rights to thdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document ntioned above.

plete Thesis/Report Pages Detail:

- Total No. of Pages = 43
- Total No. of Preliminary pages = 3
- Total No. of pages accommodate bibliography/references = 1

[illegible]
(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at 11 (%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

[illegible] 16/5/23
(Signature of Guide/Supervisor)

[illegible]
Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

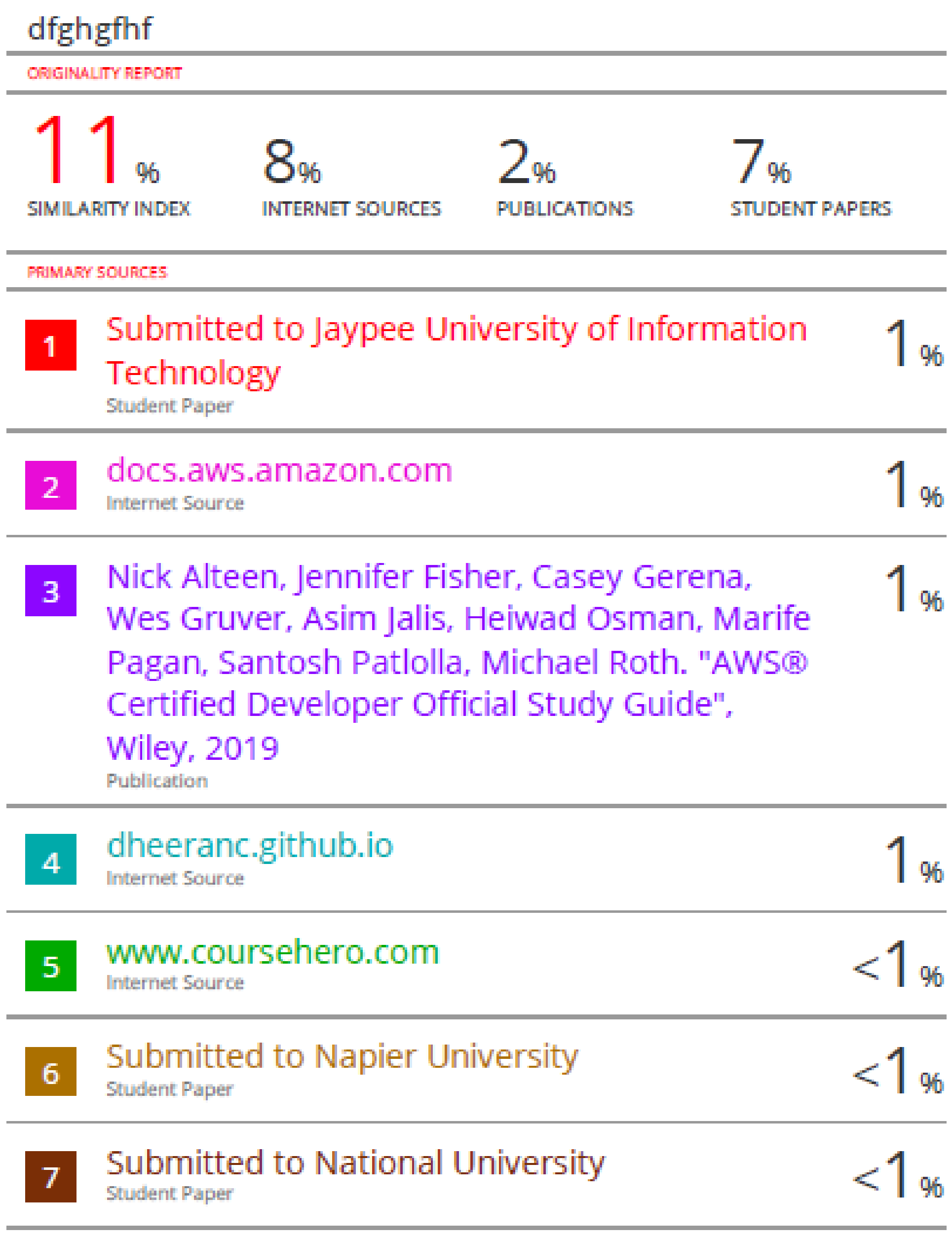
Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

Plagiarism Report



dfghgfhf

ORIGINALITY REPORT

11%	8%	2%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Jaypee University of Information Technology Student Paper	1%
2	docs.aws.amazon.com Internet Source	1%
3	Nick Alteen, Jennifer Fisher, Casey Gerena, Wes Gruver, Asim Jalis, Heiwad Osman, Marife Pagan, Santosh Patlolla, Michael Roth. "AWS® Certified Developer Official Study Guide", Wiley, 2019 Publication	1%
4	dheeranc.github.io Internet Source	1%
5	www.coursehero.com Internet Source	<1%
6	Submitted to Napier University Student Paper	<1%
7	Submitted to National University Student Paper	<1%

TABLE OF CONTENT

ABSTRACT

Employee Performance Monitoring is a web application designed to provide a framework for vie
wing and monitoring the performance status of employees in an organization. The app aims to so
lve the problem of granular status updates by providing a unified system where team members ca
n post their status and administrators can easily access and monitor groups.

The main features of the application include the ability to create and save the status of their w
ork, which can be posted daily or weekly by everyone on the team. Status updates are shared by t
he team for efficiency and management. Managers and senior officers gain transparency and acc
ountability by accessing status updates from their partners.
To increase productivity and timeliness, the app includes a notification system that notifies admi
nistrators when status updates are due or delayed. Also implemented a reward for timely support
and positive events. The app also facilitates the creation of comprehensive reports that provide in
sight into team performance and productivity.
The front end of the
application is built using the Angular framework, which provides a user-
friendly and intuitive interface for employees and managers. The backend is built in Python and l
everages AWS serverless technology and DynamoDB for efficient data management and storage
.
In general, Workforce Management is designed to simplify the process of monitoring and manag
ing work by promoting better collaboration, productivity and accountability in an organization.

INTRODUCTION

1.1 Introduction-

The development of a web application to monitor and track employee job status is the goal of this project. The current approach is ineffective since it is challenging to track and gather data because employees report their condition using various mediums. It is getting more crucial to have a centralised method to track employee work status as the Noida team expands.

1.2 Problem Statement:

The lack of a suitable system in place to see and monitor employee work status is the issue. Because employees communicate their status through various channels, it is challenging to track and compile data. As a result, there are inefficiencies and low production. It is getting more crucial to have a centralised method to track employee work status as the Noida team expands.

1.3 Objectives:

These are the project's goals:
=>Create a web application to track and manage employee work status.
=>to control employee job status using a centralised system.
=>to boost productivity through the use of a more effective system for monitoring and viewing employee job status.
=>to give managers with a simple interface via which they may see the status of their team members and provide their own updates.
=>to create a system of rewards to encourage employees to submit their statuses on time and with high-quality work.
=>to adhere to OWASP 10 recommendations in order to guarantee the security of the application and the data.

1.4 Methodology:

The project will follow an Agile methodology, with a focus on iterative development and continuous testing. The project will be divided into small sprints, with each sprint focusing on a specific set of features. The team will hold daily stand-up meetings to discuss progress and identify any roadblocks. The team will also conduct user testing and feedback sessions to ensure that the application is meeting the needs of its users.

1.5 Organization:

The project will be organized into several groups, each with its own specific tasks and responsibilities. This team will be responsible for:

Database Design and Creation: This team will design and build the database for the application.
API and Website Architecture and Technology: This group will design and build the API and website architecture and select the appropriate technology.
UI Technology Research and Development: This team will be responsible for creating the user interface of the application.
Team Coordination: This team will be responsible for coordinating the work of the different teams to ensure the success of the project.
API Coding: This team will be responsible for writing the API code.
Portal Code Writing: This team will be responsible for writing the portal code.
Testing: This team will be responsible for testing the application and making sure it meets the requirements.
Tech Write: This team will be responsible for writing the API and database structure.

Literature Survey:

2.1 Literature Survey

In the context of this project, survey data will include research and analysis of employees current jobs to find and manage solutions. This will include exploring the features, functionality and usability of the various devices, apps and software available in the market.

Case studies focusing on collecting data on the following topics:

Employee Monitoring and Management Tools: Learn about the various software applications available in the market, allowing employees to report their job status and for employees to monitor and monitor it.

Features and Functions: Features and functions of the diagnostic tool, including event generation and storage, sending frequency, management, alerts and alerts, reporting and analysis, and integration with other tools.

User Interface and User Experience: Examine the device's user interface and user experience, including ease of use, navigation, accessibility, and responsiveness across products.

Technologies: Learn about the different technologies used to build similar applications, including front-end systems, back-end systems, and cloud services.

Security and Data Privacy: Security and data privacy scanning tools, including access control, access, data backup, and compliance with data protection laws.

A literature review will help identify best practices, trends, and potential challenges in recruiting and managing practices. It will also help to choose the right technology and create a strong and secure design for implementation.

System Development

3.1 Analysis, Design, Development and Algorithm :

During the analysis, we identify the problem statement, write down the requirements and create a solution concept. Based on these, we create the architecture of the system. The proposed system follows a client-server architecture where the server side is developed by AWS and the client side is developed using Angular.

During the design phase, we consider the architecture and identify many of the things that need to be done to implement the process. We also identified the technologies and tools needed to build the system, such as AWS Lambda, API Gateway, DynamoDB, and Angular.

During development, we followed a process in which tools and techniques were used. We use AWS CloudFormation to build the necessary infrastructure on the AWS platform. We created a Lambda function to handle the backend logic and integrated it with API Gateway to expose the API. We use DynamoDB for data storage and Angular for client-side development.

We follow the agile process during development, which helps us break things down into small sprints and prioritize tasks according to their importance and urgency.

This helps us to deliver the process on time and within budget. At the

Algorithm level, we have determined various algorithms that should be used in the system, such as warning algorithms, report generation algorithms, access control algorithms. These algorithms are used in the Lambda function to manage the backend logic of the system.

High-level Features of the project is as follow:

1)Each team member can create and save a status, once completed he can submit daily or weekly as needed.

2)List users by their team and we will have each Term Owner (can view the status of team members).

3)Once user status is ready user will submit and it will be visible to the manager by clicking the user name from the list.

4)Employee status can only be viewed by their manager and upper manager and who is allowed to view.

5)Feature to create consolidated the status.

6)After Submit the Reminder system send notification mail to the manager.

7)Compile Report.

8)Reminder mail to submit the status. If not submitted on a weekend day. Reminder before 1 day.

9)Each team member can create and save a status, once completed he can submit daily or weekly as needed.

10)User management.

11)Role management.

12)We can have a Reward system.

13)Give Reward on submission on-time Auto.

14)Reward on good Status by Manager.

List of required Item:

The interface is divided into two View.

My View (User View).

Managerial View ((Can say Check Status).

User View:

Dashboard - Common for all View, This is the Home page of Each user login

The following information will be shown here:

1)Top 5 users who submitted Status on time

2)Who has not submitted Status

3)More TBD

4)My Status - User can submit thire status and View

5)User Info

6)Add Status and Status only Edit by user

7)Status List

8)Status View

Managerial View

1. **Employee Lists -** Landing Page of Managerial View
    1. List of users by team
    2. Filter by Team
    3. Each user has a list of Status which can be open from here.
2. **Employee Status List Page** - Individual employee Status list from where Manager can View Status
    1. On click View, Status Status open on Popup
3. Team Management -
    1. List of Team
    2. Add/Edit Team
4. We can have one more Feature Add task - User itself can assign task himself and manager also (We have some work which usually not assign by Jira)
5. Report
    1. Generate Report (Monthly, Weekly, Yearly)
    2. check productivity - need more brainstorming

Considerations In Design:

1)Use Role-Based Access Control (RBAC)

2)Required Responsive Design (For 3 Device Desktop, Tab, Mobile)

3)User password must be encrypted by irreversible encryption SHA256 or Secure SALT

4)You have to design your DB

5)Discuss and find and design best Architecture

6)Architecture must be API based

7)For each record entry and modification need to keep - Created by, Modify by, Created Date, Modify Date

8)Consider - At the Tutorial page (A1:2017-Injection | OWASP) there are a section References

9)Your customer app does not have any SQL vulnerability.

10)Follow all OWASP 10 Guideline

11)Do Brainstorm

12)Do not Hard Code any credential in the Code

13)Divide your task into teams, Following possible task of planning

14)break up all tasks into small tasks

15)DB design and DB creation

16)API and website Architecture and Technology

17)UI Technology finding and Development

18)Team coordination

19)API Code writing

20)API access must be secure

21)Web Portal code writing

22)Code merge planning by using Version control

23)Hosting Planning

24)Testing planning

Also technical writing in API and DB model Confluence page

Testing - You also have to do Application testing

a)Functional testing

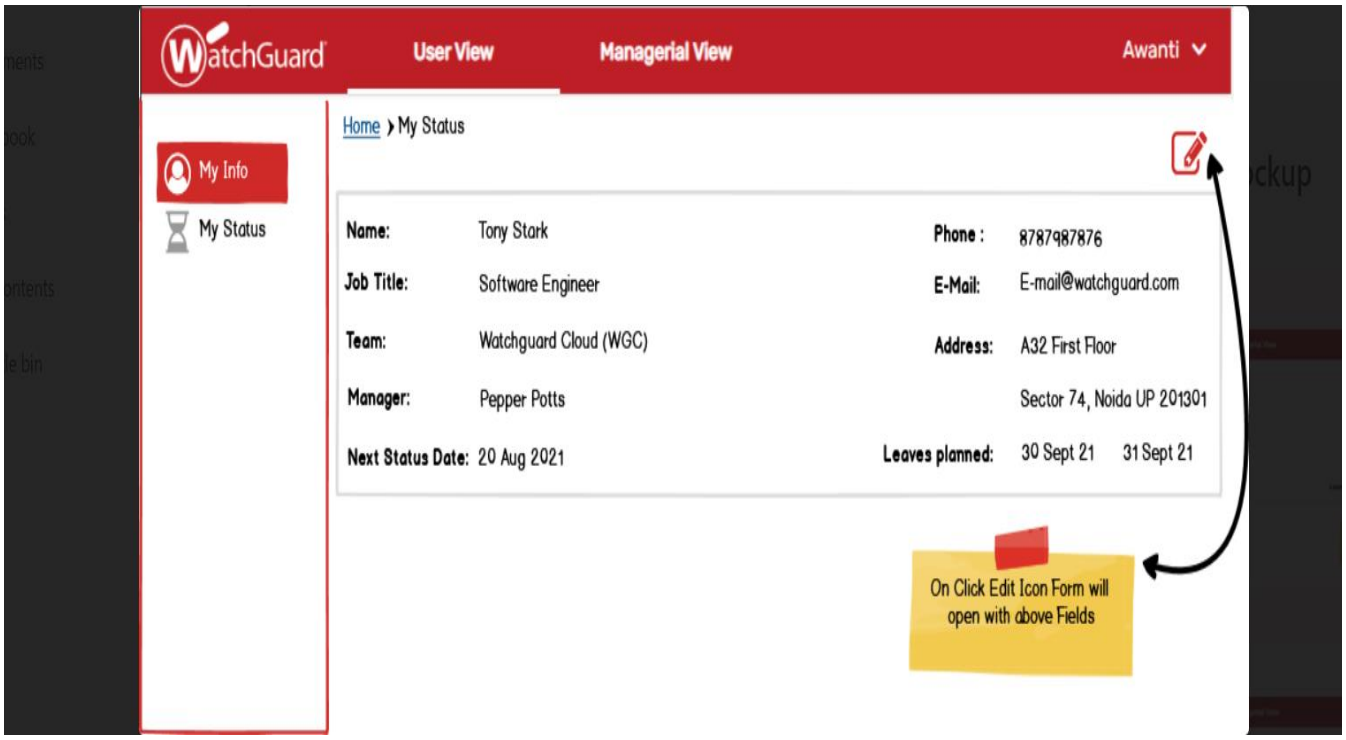
b)Unit testing

c)Application Security Testing

UI Screen:

Login Page -Allow users to login into the system by using Valid "Username" and "Password". Designing login page it should look like the user login into the system where he/she can submit the status.

My View - Screen 1 - My Info - Landing page for each user After login. Here you will show the user Information.



User View | Managerial View | Awanti

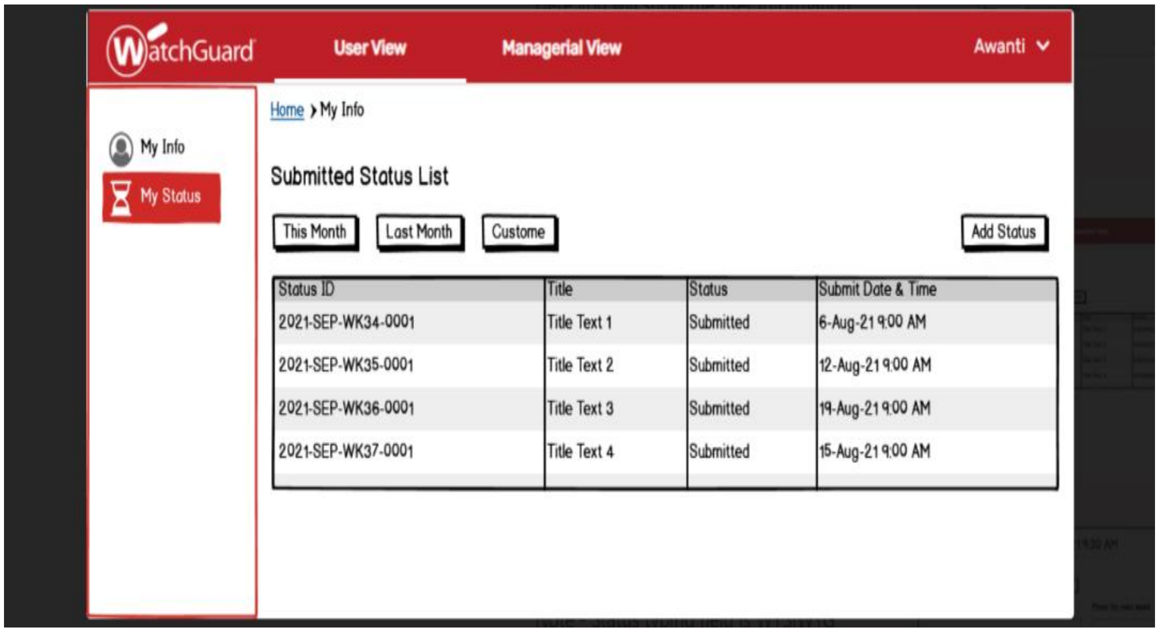
My Info
My Status

Home › My Status

Name:	Tony Stark	Phone :	8787987876
Job Title:	Software Engineer	E-Mail:	E-mail@watchguard.com
Team:	Watchguard Cloud (WGC)	Address:	A32 First Floor
Manager:	Pepper Potts		Sector 74, Noida UP 201301
Next Status Date:	20 Aug 2021	Leaves planned:	30 Sept 21 31 Sept 21

On Click Edit Icon Form will open with above Fields

My View - Screen 2 - My Status - Here we need to show all submitted status lists of logged-in user with filter.



User View | Managerial View | Awanti

My Info
My Status

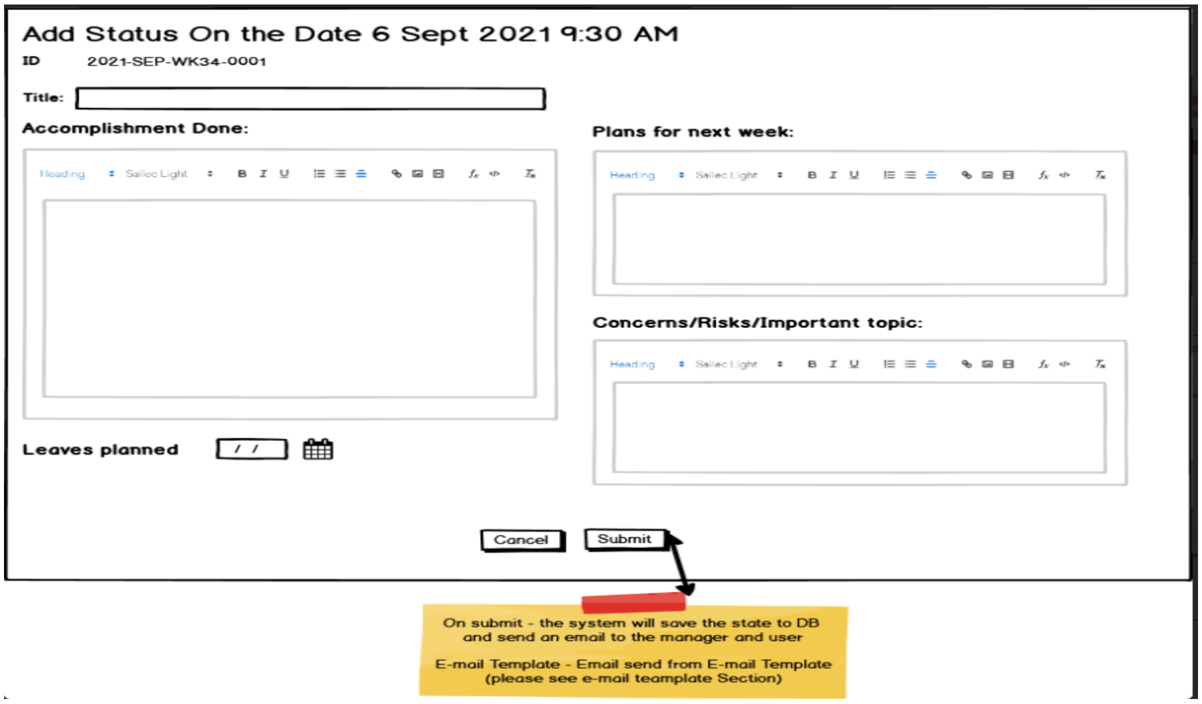
Home › My Info

Submitted Status List

This Month | Last Month | Custome | Add Status

Status ID	Title	Status	Submit Date & Time
2021-SEP-WK34-0001	Title Text 1	Submitted	6-Aug-21 9:00 AM
2021-SEP-WK35-0001	Title Text 2	Submitted	12-Aug-21 9:00 AM
2021-SEP-WK36-0001	Title Text 3	Submitted	19-Aug-21 9:00 AM
2021-SEP-WK37-0001	Title Text 4	Submitted	15-Aug-21 9:00 AM

My View - Screen 3 - Add Status – this is used to add status to the watchguard team manager assigned work.



Add Status On the Date 6 Sept 2021 9:30 AM

ID 2021-SEP-WK34-0001

Title:

Accomplishment Done:

Plans for next week:

Concerns/Risks/Important topic:

Leaves planned / /

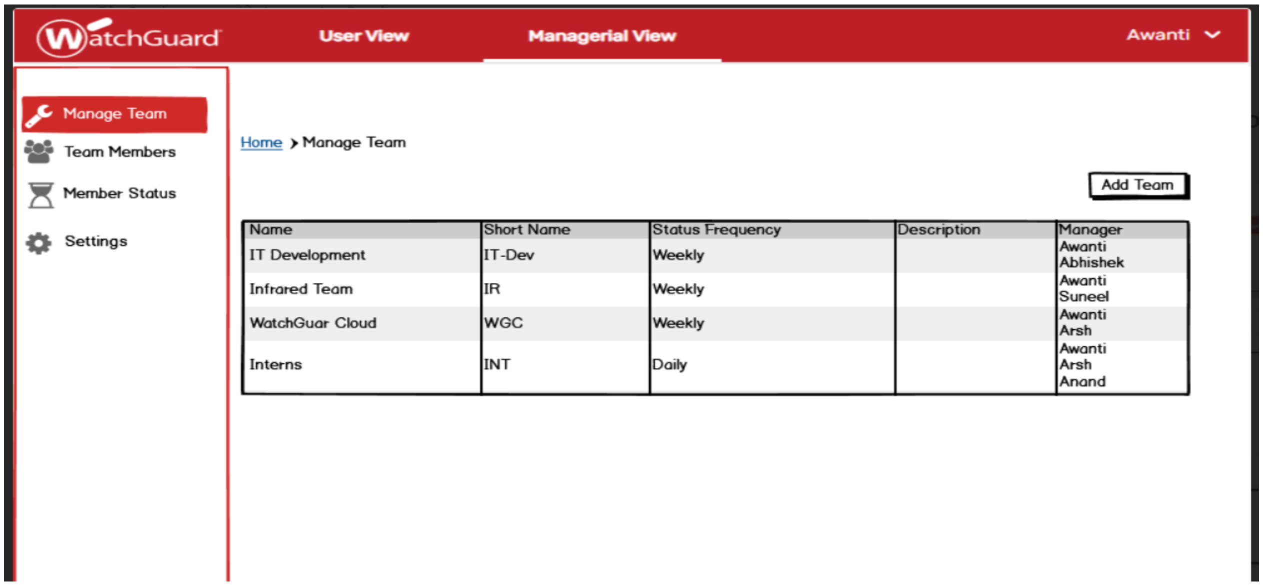
Cancel Submit

On submit - the system will save the state to DB and send an email to the manager and user

E-mail Template - Email send from E-mail Template (please see e-mail teamplate Section)

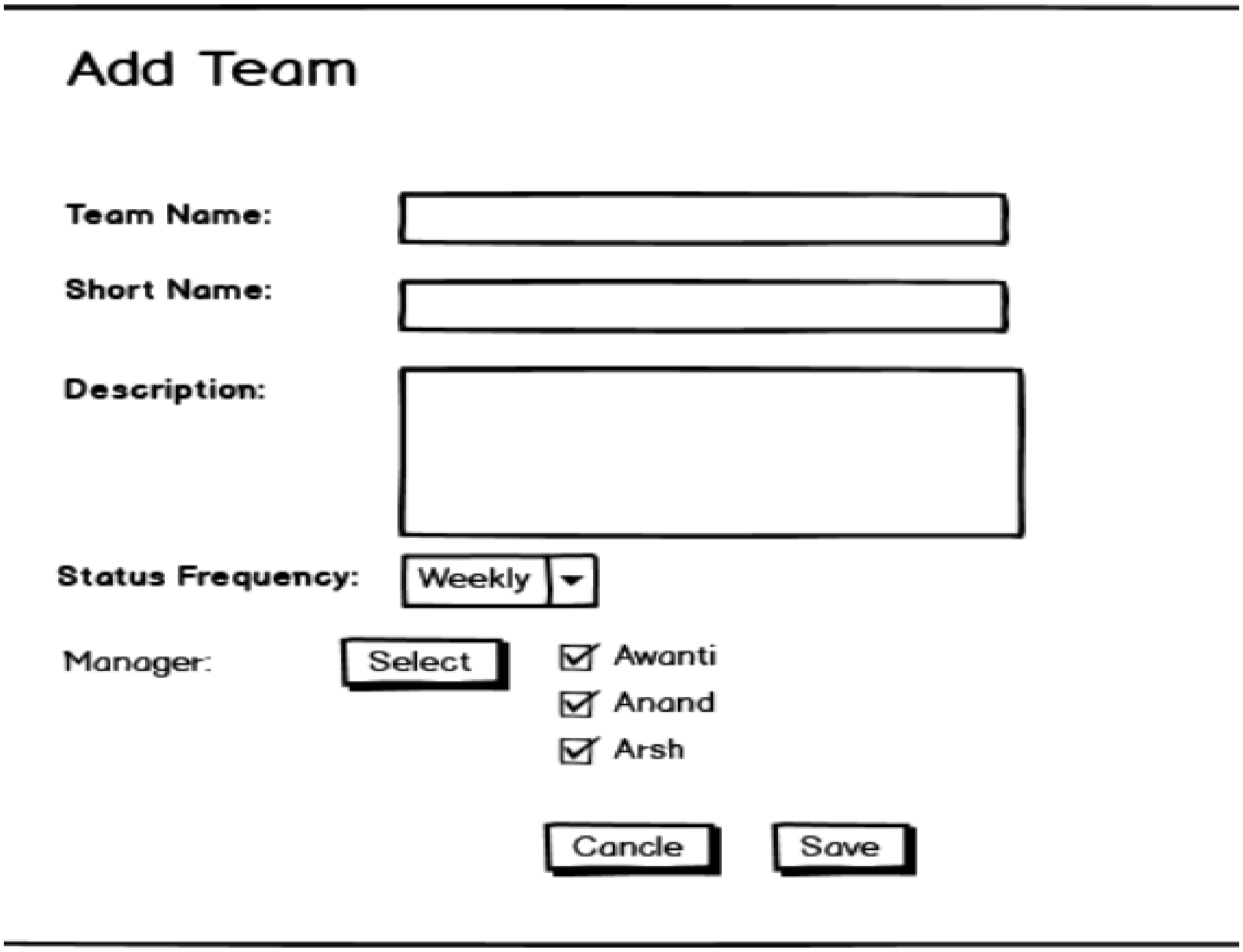
Team Management :

List of Team: Will show all existing Team list.



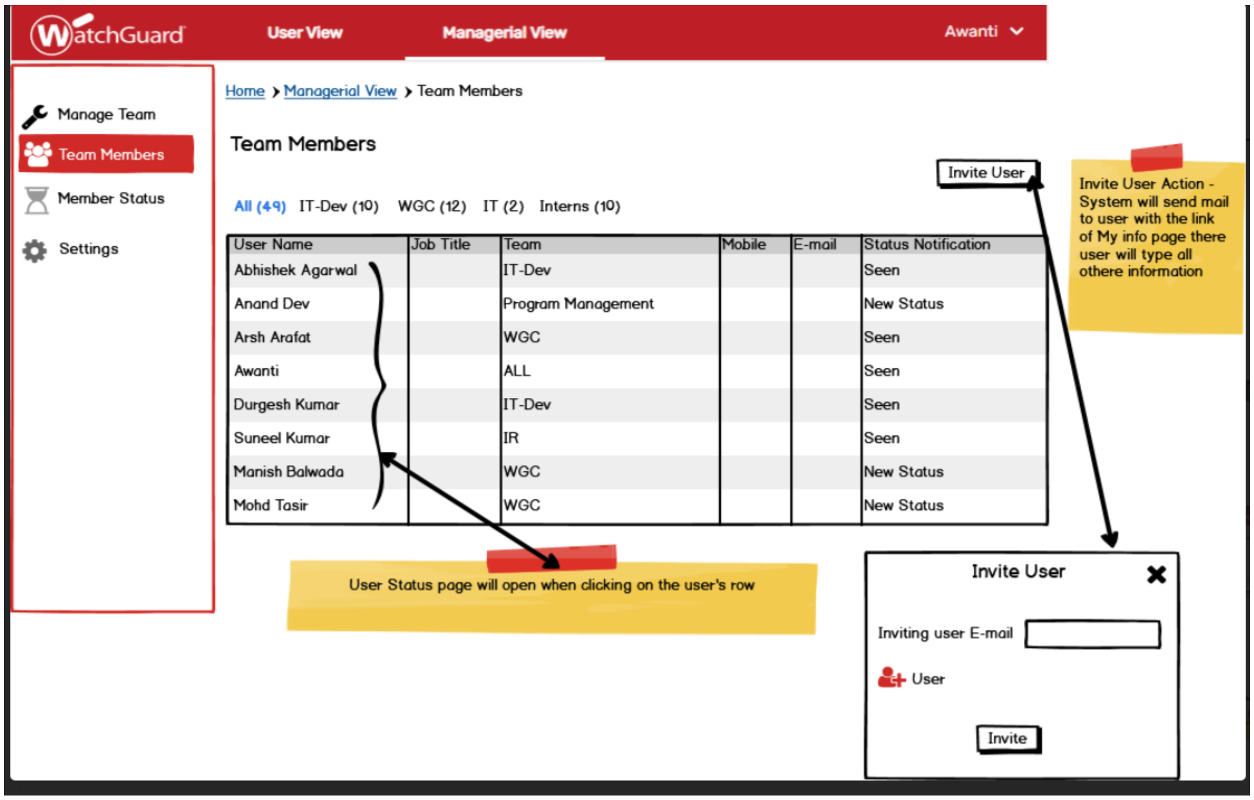
Name	Short Name	Status Frequency	Description	Manager
IT Development	IT-Dev	Weekly		Awanti Abhishek
Infrared Team	IR	Weekly		Awanti Suneel
WatchGuar Cloud	WGC	Weekly		Awanti Arsh
Interns	INT	Daily		Awanti Arsh Anand

Add/Edit Team - Team Add Edit Form.



Managerial View :

Managerial View - Screen - 1 - Team Members - The managerial View landing page is Employee Lists .Will show list of All users and filter by their Team.On Click on user That user status list Page will open.



Home > Managerial View > Team Members

Team Members

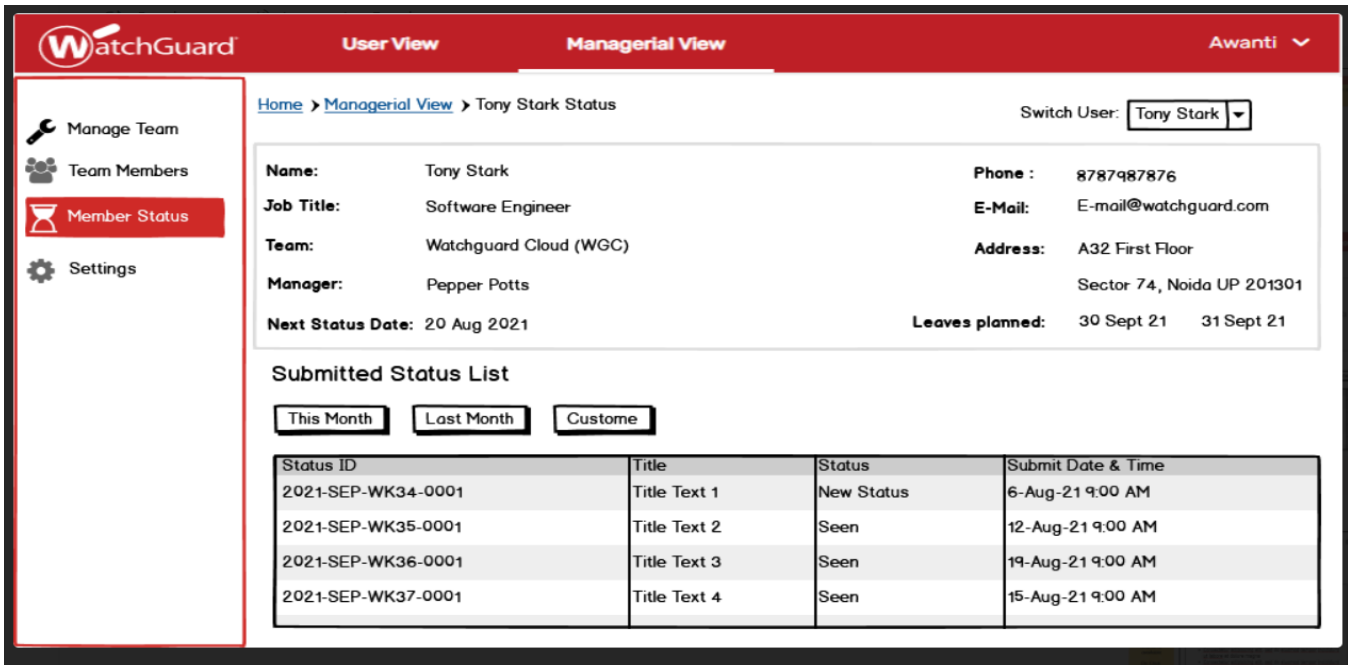
All (49) IT-Dev (10) WGC (12) IT (2) Interns (10)

User Name	Job Title	Team	Mobile	E-mail	Status Notification
Abhishek Agarwal		IT-Dev			Seen
Anand Dev		Program Management			New Status
Arsh Arafat		WGC			Seen
Awanti		ALL			Seen
Durgesh Kumar		IT-Dev			Seen
Suneel Kumar		IR			Seen
Manish Balwada		WGC			New Status
Mohd Tasir		WGC			New Status

Invite User Action - System will send mail to user with the link of My info page there user will type all othere information

User Status page will open when clicking on the user's row

Managerial View - Screen - 2 - Employee Status List Page - Here we will show some information and his/her weekly status list.



Home > Managerial View > Tony Stark Status

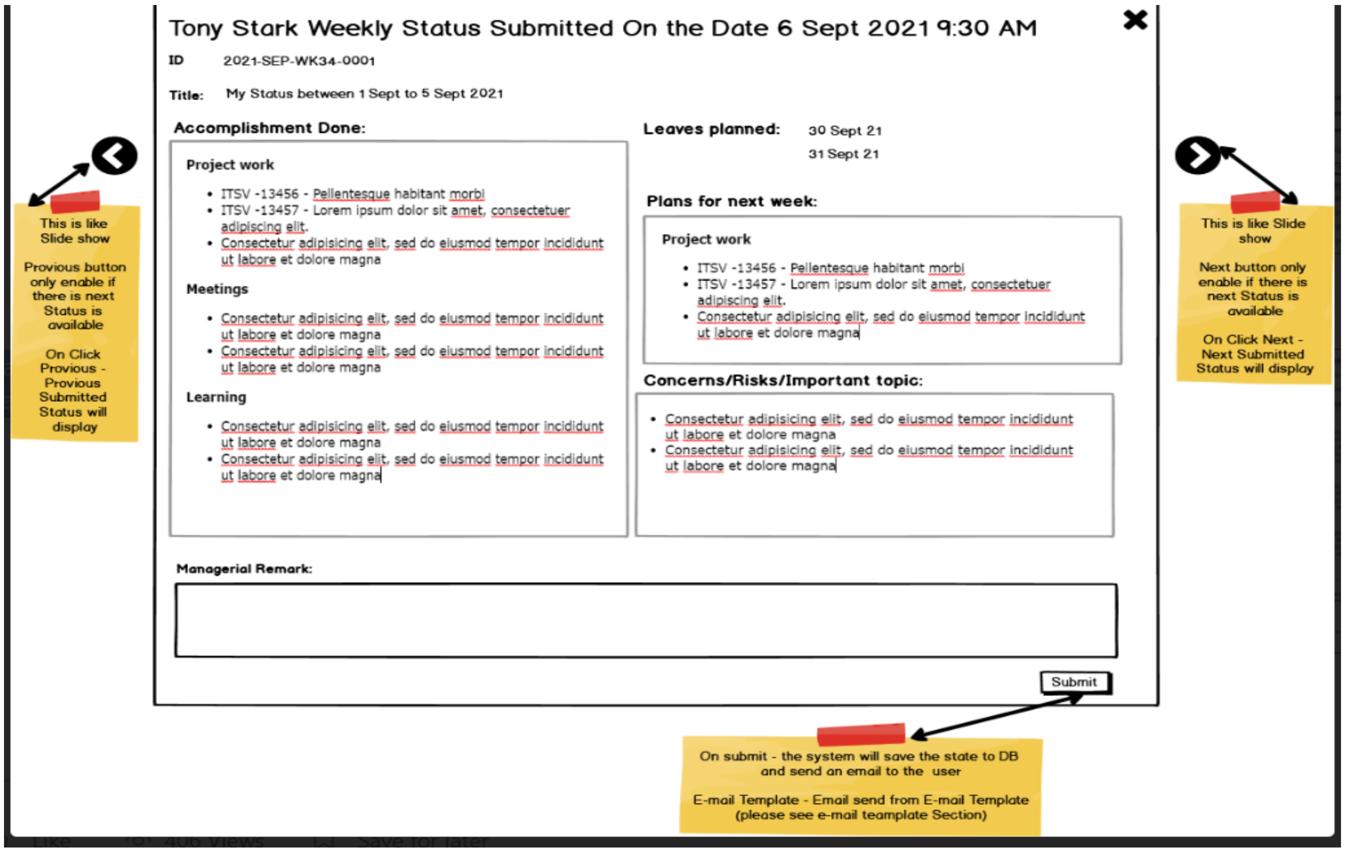
Switch User: Tony Stark

Name:	Tony Stark	**Phone :**	8787987876
Job Title:	Software Engineer	**E-Mail:**	E-mail@watchguard.com
Team:	Watchguard Cloud (WGC)	**Address:**	A32 First Floor
Manager:	Pepper Potts		Sector 74, Noida UP 201301
Next Status Date:	20 Aug 2021	**Leaves planned:**	30 Sept 21 31 Sept 21

Submitted Status List

Status ID	Title	Status	Submit Date & Time
2021-SEP-WK34-0001	Title Text 1	New Status	6-Aug-21 9:00 AM
2021-SEP-WK35-0001	Title Text 2	Seen	12-Aug-21 9:00 AM
2021-SEP-WK36-0001	Title Text 3	Seen	19-Aug-21 9:00 AM
2021-SEP-WK37-0001	Title Text 4	Seen	15-Aug-21 9:00 AM

Managerial View - Screen - 3 - Employee Status View Popup – can view the status of the individual team member.


Tony Stark Weekly Status Submitted On the Date 6 Sept 2021 9:30 AM

ID 2021-SEP-WK34-0001

Title: My Status between 1 Sept to 5 Sept 2021

Accomplishment Done:

Project work

- ITSV -13456 - Pellentesque habitant morbi
- ITSV -13457 - Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
- Consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna

Meetings

- Consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna
- Consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna

Learning

- Consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna
- Consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna

Leaves planned: 30 Sept 21
31 Sept 21

Plans for next week:

Project work

- ITSV -13456 - Pellentesque habitant morbi
- ITSV -13457 - Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
- Consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna

Concerns/Risks/Important topic:

- Consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna
- Consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna

Managerial Remark:

This is like Slide show

Provious button only enable if there is next Status is available

On Click Provious - Provious Submitted Status will display

This is like Slide show

Next button only enable if there is next Status is available

On Click Next - Next Submitted Status will display

On submit - the system will save the state to DB and send an email to the user

E-mail Template - Email send from E-mail Template (please see e-mail teamplate Section)

We use REST API principles for making the API end points:

1) **Client-Server :** The client server REST API (Representational State Change Application Service Interface) protocol is a standard framework for building web-based applications that communicate between clients and servers via HTTP requests and responses. RESTful APIs include GET, POST, PUT, DELETE, etc., usually expressed in JSON or XML format. It performs CRUD (create, read, update, delete) operations on resources using methods.

   In a client-server REST API, a client sends a request to the server, and the server processes the request and sends a response to the client. A client can be a device or application that communicates with a server that stores and manages data and resources.

For example, let's consider a simple REST API for a bookstore.
The client can be a web app or a mobile app that allows users to search and purchase bo oks. The server returns the final result as:

GET /books: get a list of books
GET /books/{id}: get a specific book by id
POST /books : add new book to database
PUT /books/ {id} }: Update current book with id
DELETE /books/{id}: Delete book with id
When the client wants to save the book, it sends a GET request to the server at the end of /book /{id}, {id} is the id of the book. The server will process the request, retrieve the data from the database, and send the response back to the client in JSON or XML format . Similarly, users can use other methods to add, edit or delete books in the database.

2) **Statelessness -** Statelessness is a fundamental principle of RESTful API design; this means that the server is not preserving any user state of the request. Every request sent from the client to the server must contain all the information needed to complete the request, including credentials. The server processes each request independently and the response it returns is not optional from the previous one.

For example, let's say a customer wants to purchase a book in the book API. The client sends a POST request to the server to create a new order.
The application contains all the information needed to complete the order, including the identity of the book, the amount, and the customer's payment information. The server processes the request and creates a new order in the database.

The server does not store information about the client or the request queue. If the customer wishes to update the order, he must submit another request with all the updated information required to complete the update.

Statelessness is useful because it makes servers bigger, more reliable, and easier to manage.
A stateless API can handle many requests without overloading the server and can easily load balance across multiple servers.

In essence, statelessness in a RESTful API means that each request contains all the information needed to complete the request and the server does not store any user state on request. For the

Bookstore API, the error to append a book to a file looks like this:

```
POST /books
Content-Type: application/json

{
"title" : "The Alchemist",
author": "Paulo Coelho",
"published": "1988",
"isbn": "9780061122415",
"price": 9.99
}
```

title, author, publication year, ISBN and price Server processes the request and new The design gives a response with the resulting book ID.

3) Cacheable - Caching is another important principle of RESTful API design and is designed to improve performance and reduce network traffic by storing responses to frequent requests. When the client makes a request, the server can tell if the response is cacheable. If it is cacheable, the client can store the response in its cache and reuse it for subsequent requests, reducing the need to get resources from the server.

For the

Booklist API, let's say the client wants to keep the list of books available in the bookstore. A server can specify that a response is cacheable by adding an appropriate HTTP header such as "Cache-Control: max-age=3600" (meaning that the response can be cached for one hour).

As long as the response is valid, the client can cache the response and reuse it for subsequent requests.

If the user needs to retrieve the same list again in the next hour, they can reuse the cached response without sending another request to the server. This reduces network traffic and improves API performance.

Here is an example of a cacheable request to get a list of books:

```
GET /books HTTP/1.1

Host: Example.
```

com

Cache-Control: max-age=3600

The server can respond with a directory listing in JSON format with the appropriate "Cache-Control" header:

```
HTTP/1.1 200 OK

Content-Type: application / json

Cache -Control : max-age=3600


[

{

"id": 1,

"title": "The Alchemist",

"author": "Paulo Coelho",

"1988": " " ,

"isbn" : "9780061122415",

"price": 9.99

},

{

"id": 2,

"title": "To Kill a Mockingbird" ,

"Harperor": Lee ,

"published": "1960 ",

"isbn": "9780061122415",

"price": 12,99

},
// more books..
.
]
```

As long as the "Cache-Control" header is valid, the client can store this response in its cache and reuse it for subsequent requests. If the user needs to retrieve the list of books again after the cache is exhausted, they must send another request to the server to retrieve the new list.

4)Layered System :

For example, in a simple three-layer RESTful API architecture, the layers include:

Presentation layer: this layer is responsible for interacting with people who use products and provide resources to them. It usually includes API endpoints and handles HTTP requests and responses. The submission process is responsible for receiving and processing requests and returning responses.

Application Layer: This layer manages the business logic of the application. It is responsible for processing requests from the presentation layer and generating appropriate responses. It also interacts with the data layer to provide or update data.

Data layer: This layer is responsible for storing and managing the data used by the application.

It includes databases and other data storage. The database layer is responsible for performing CRUD (create, read, update, delete) operations on the database and returning the results to the application layer.

An example of how the protocol works in a RESTful API for an e-commerce application:

The serving protocol receives a request for a product list based on a customer request.

The serving process sends the request to the request layer.

5) Uniform Interface - Uniform interface is an important component of RESTful API design. It refers to the design of connections between clients and servers that support separation of concerns and make APIs extensible and reusable. In other words, Unified Interface is the framework that controls how the RESTful API behaves. The four restrictions imposed by

Unified Interface are:

1. Resource Identification: Each resource in the API must be uniquely identified using a URI.

This URI should be used to access, manage and delete resources.

2. Operations: The API should provide appropriate operation methods such as GET (get resources), POST (create new resources), PUT (replace existing resources) and DELETE (delete resources).

3. Self-describing messages: Messages exchanged between client and server must be self-describing, that is, they contain all the information necessary to describe the content of the message, including metadata and data types.

4. Hypermedia as an Application State Engine (HATEOAS): APIs should use hypermedia connections to provide users with information about the resources and operations they can perform on that resource. This allows users to access the API without prior knowledge of its structure.

Resource identification limits ensure that each resource in the API has a unique URI that can be used to identify and access it. This is important for scalability as it allows resources to be cached and distributed across multiple servers.

Resource performance constraints define performance criteria for resource management. These functions are based on commonly used and understood HTTP methods. By using a process to manage resources, the API will be more understandable and easy to use.

Self-explanatory restrictions ensure that messages exchanged between client and server contain all the information necessary to describe the content of the message. This allows users to understand the data type and metadata of the message without prior knowledge of the API.

The application layer requests and receives products from the data layer.

The application layer generates a response containing the object name and returns it to the presentation layer.

The submission process returns the response to the requester.

In this example, the presentation layer interacts only with the application layer, and the application layer only interacts with the data layer. Each layer has its own functions and changes made in one layer do not affect the other layers.

Layered system architecture is a good way to design a RESTful API as it provides simplicity, scalability, and separation of concerns.

It allows different teams to work on different layers of the API without affecting other layers, making it easy to switch or replace layers as needed.

Code On Demand - Code on Demand constraints are optional restrictions in RESTful API design that allow the server to continue processing the client by occasionally sending executable code (such as JavaScript) to the client. This principle is sometimes referred to as "active resources" because it allows the server to process code on behalf of the client, making the server more collaborative in application form.

This restriction is optional as not all RESTful APIs are required to be able to send full code to the client. It is important in some cases, such as when the server wants to provide complex algorithms for the user to perform on the client, or the server wants to tweak the client's behavior based on its ability.

Here is an example of how to use the optional code constraint in a RESTful API:

Let's say we have a weather API that provides weather information about different locations.

The API provides a number of methods, including the ability to store the current weather for a location. In addition to these capabilities, the API provides capabilities that allow users to download executables that can be used to display weather data in a custom way.

When a user requests weather information for a particular location, the server can send the user a script that updates the weather information according to the user's preferences. This script can be written in JavaScript or any other executable language and sent to the requestor as part of the response.

After the user receives the script, he can process it in his own environment and use special methods to display weather information to the user.

The user doesn't need to understand the contents of the script, they just need to understand how to do it.

Optional Code restrictions are useful when the server wants to delegate some functionality to the client, or when the server wants to provide client specific code. However, since the server sends executable code to the client, it can also pose a security risk that could harm the client. Therefore, it is important to use this limitation to determine and apply appropriate protection.

3.2 Model Development :

A conceptual system is usually a data-driven system that relies on data stored in a database for its operation. Therefore, we created a database model using DynamoDB to represent various entities in the system such as users, groups, events, and rewards.

The standard database contains the following fields:

User: Contains user details such as name, email, password, and role.

Group: Contains group details such as name and description.

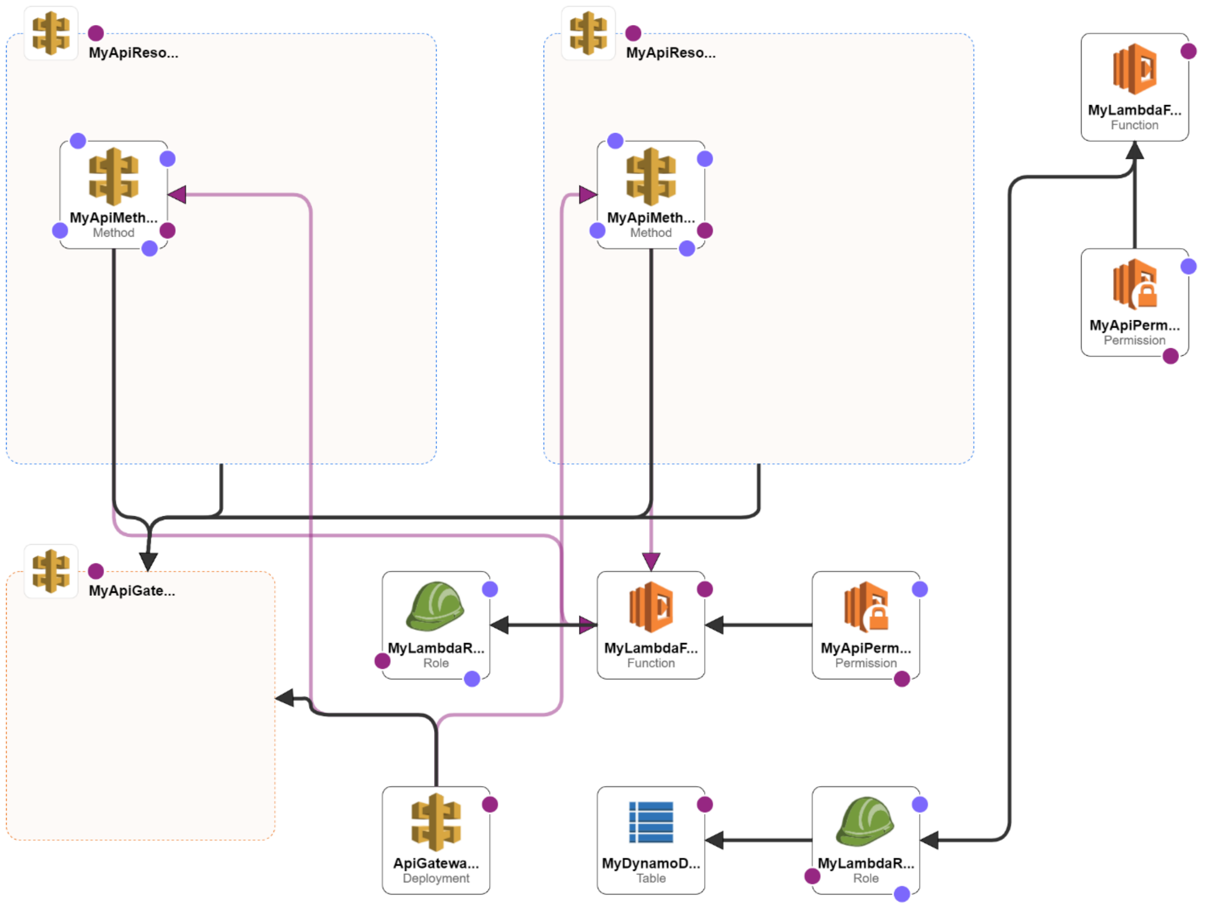
Status: Status type, description, date, etc. Contains details of the user-submitted status.

Gift: Contains details such as the type, description and date of the gift given to users.

The standard database also includes relationships between these entities, such as relationships between users and groups, and relationships between users and states.

We use AWS CloudFormation to create data on AWS platform and we use AWS SDK to interact with data via a Lambda function**.**

The below figure describe how API gateway is connected to my lambda functions and lambda function is using the boto library in python to perform the read and write operations.



We created it all in the cloud formation template. We have created the template in yaml.

1)Lambda function –

Lambda functionality in AWS is a serverless service that allows you to run code in response to events such as changes to data in an S3 bucket or updates to database tables.

When creating a Lambda function on AWS, you can submit the code in one of several languages, including Python, Node.js, Java, C#, and Go. You can then configure the task to trigger on specific events, such as an Amazon S3 bucket or Amazon API Gateway endpoint.

When an event triggers a Lambda function, AWS automatically allocates the required computing resources to run the job and then executes your code.You only pay for the duration of your event, with no upfront or administrative fees.

Lambda functions can be used for a variety of applications, including data manipulation, real-time programming, and building serverless web applications. They can also be integrated with other AWS services such as Amazon S3, Amazon DynamoDB, and Amazon Kinesis to build complex and scalable applications.

3)Lambda Role: In AWS (Amazon Web Services), Lambda functionality is a serverless service that allows you to run code without managing or managing servers. Lambda functions can be triggered by a variety of events, such as changes to an S3 bucket, API Gateway requests, or scheduled events.

When creating a Lambda function, you must specify the role of the function. This role determines the permissions a task has to access AWS services and resources. A function function is basically a set of rules that define what Lambda functions are allowed to perform.

The role is created using AWS Identity and Access Management (IAM) and can be customized to the specific needs of your Lambda role. For example, you may need to grant your Lambda function access to read and write to an S3 bucket or access a particular site or API Gateway.

By default, when you create a new Lambda function, AWS creates a new IAM role for the function with basic permissions. However, you can create your own IAM roles and assign them to your Lambda roles as needed. This is useful if you need special permissions for your Lambda function that are not available in the original function.

In summary, the Lambda role is an AWS IAM role that defines the Lambda role's permissions and access to AWS services and resources. This is an important security feature that helps ensure that your Lambda function can only access the programs it needs and nothing else.

4)API Gateway - Amazon API Gateway is a fully managed service that makes it easy to create, deploy, and manage APIs of any size. It acts as a "gate" to your application, allowing you to define what should be the request for the backend service and what should be done in response. API Gateway supports RESTful APIs and WebSocket APIs, making it suitable for a variety of applications including mobile and web applications, IoT devices, and serverless applications. Here are some key features and benefits of

1. Integration with AWS Services: API Gateway can be easily integrated with other AWS services such as AWS Lambda, AWS Step Functions, and Amazon S3.

This allows you to build scalable and robust serverless architectures to handle large traffic.

2. Scalability and High Availability: API Gateways are designed to handle large traffic and can scale as needed. It is versatile and can withstand the failure of individual components.

3. Security: API Gateway provides many security features such as authentication, authorization, and encryption. Integrates with AWS Identity and Access Management (IAM) to manage access to your API and manage security.

4. Monitoring and Analysis: API Gateway provides real-time monitoring and analysis of API usage, errors, and performance. Integrates with AWS CloudWatch to provide detailed information and alerts that can be used to monitor and troubleshoot APIs.

AWS Lambda supports many types of integrations such as HTTP and AWS Step Functions. You can use API Gateway to add features such as caching, throttling, and request and response handling.

When you define your API, you can refer to it as a stage that represents a point-in-time snapshot of your API. You can then use the API Gateway console, CLI, or SDK to manage your API, including creating new levels, managing domains and SSL/TLS certificates, and monitoring API usage and operation.

In a nutshell, Amazon API Gateway is a powerful and flexible service that makes it easy to create, deploy, and manage APIs at scale.

Whether you're building mobile apps, web apps or serverless architectures, API Gateway can help you build secure, scalable and versatile APIs to handle any task.

Dynamo DB - Amazon DynamoDB is a fast and scalable NoSQL service from AWS. DynamoDB is designed to provide high performance and scalability with low latency and high availability. It's a full-service management system that lets you store as much data as possible and is the foundation for some of AWS' most popular services, including Amazon Alexa, Amazon Prime, and AWS IoT.

Here are some key features and benefits of DynamoDB:

1. Scalability: DynamoDB is designed to be highly scalable, handling millions of requests per second.

It provides automatic measurement without the need for manual intervention to meet the needs of applications.

2. Performance: DynamoDB provides fast and consistent performance with low latency. It uses SSD storage and memory caching to provide sub-millisecond response for read and write operations.

3. Flexibility: DynamoDB is a simple database service that allows you to store and store any type of data, including structured, semi-structured and unstructured data.

4. Security: DynamoDB has security features including authentication at rest and in transit, active access control using AWS Identity and Access Management (IAM), and AWS Key Management Service (KMS).

5. Global distribution: DynamoDB supports global distribution, allowing you to replicate data across multiple regions to provide cost-effective data for anywhere in the world.

To use DynamoDB, you first create a table, which is a large container for your data. You define the schema for your table, including the primary key and other objects you want to store. DynamoDB supports two types of primary keys: partitioned values and partitioned values and keys. Shared keys allow you to store and store information as a single character, while shared keys and keys enable further querying and analysis.

After you create a table, you can start adding data to the table using the DynamoDB API or the AWS Management Console.

DynamoDB provides a flexible data model that allows you to store any type of data, including files, images and timestamps.

DynamoDB provides many features to improve performance, including caching, indexing and query optimization. You can use DynamoDB Streams to capture data changes in real time and integrate with other AWS services such as AWS Lambda, AWS Glue, and Amazon Kinesis to create real-time data pipelines.

In summary, DynamoDB is a scalable, flexible and fast NoSQL database service that provides low latency and versatility. With its ability to store large amounts of data and support global distribution, DynamoDB is ideal for building high-performance and scalable databases.

We make our project by taking into **OWASP** vulnerabilities will not take place :

OWASP-Open Web Application Security Project OWASP is a non-profit organization focused on improving the security of software applications. The organization provides advice, tools, and resources to help developers, security professionals, and organizations improve web security.

OWASP was founded in 2001 and has since grown into a leader in web application security. Its purpose is to make software security visible so individuals and organizations can make informed decisions about software security risks. The organization is funded by individual and corporate donations and run by volunteers.

OWASP has created several programs for the software security community, including:

1. OWASP Top Ten: List of Worst Web Application Security Risks. The top ten threats are updated every few years to reflect changes in the threat landscape.

2. OWASP ZAP (Zed Attack Proxy): A web application security scanner that helps identify vulnerabilities in web applications.

3. OWASP Application Security Verification Standard (ASVS): Methods and techniques for verifying the security of software applications.

4. OWASP Web Security Test Guide: A comprehensive web security testing guide.

444 5.

OWASP Code Review Guide: A guide to reviewing source code for security.

6. OWASP Mobile Security Project: It is a project to increase the security of mobile applications.

OWASP also hosts international conferences and workshops to facilitate software security awareness and knowledge sharing.

In a nutshell, OWASP is a non-profit organization focused on improving the security of software applications.

Provides resources, tools, and guidance to help developers, security professionals, and organizations identify and mitigate security risks in web applications. The organization is highly respected in the software security community and plays an important role in providing the best software security.

1.Broken authentication -

There are several ways to make sure you won't be hacked:

a) Weak passwords: If an application allows weak passwords or doesn't manage complex passwords, outsiders can easily guess passwords or brute force access.

b) Session management flaws: If the application cannot properly manage user sessions, such as using weak session IDs or not expiring after a certain time, an attacker will hijack user sessions to gain access to the application.

Not trusting credentials: If the application insecurely stores user credentials such as passwords or authentication tokens, attackers can steal those credentials and use them to enter the application.

c) Broken Multi-Factor Authentication: If an application uses two-factor or multi-factor authentication, but does so poorly, attackers can bypass additional steps to gain access login.

d)Deletion of erroneous information: Attackers can continue to use credentials to gain access if the application does not correctly delete the user's credentials, such as when the user leaves the organization or the role changes.

To prevent authentication breaches, application developers and administrators should follow best practices for authentication procedures, such as making password management strong, using secure session management, securely storing certificates, and using multiple authentication methods. Regular evaluation and monitoring of the accreditation process can also help identify and address vulnerabilities.

SQL INJECTION –

SQL injection is a vulnerability that occurs when an attacker can insert malicious SQL code into an application's database queries. SQL injection can allow an attacker to access, modify or delete sensitive data and even control the entire application.

SQL injection vulnerabilities often arise when an application fails to use the appropriate user input, such as when retrieving data from a userform and then using that input directly.

For example, consider an application that allows users to search for items by entering a keyword in the search field. The application takes the user's input and uses it in SQL queries to retrieve related items from the database.

If the application fails to validate the user's input and does not allow them to enter the SQL code along with the search terms, attackers can insert malicious SQL code into the query.

Example of a simple SQL injection block:

Suppose the search query field accepts the following words:

``

apple ' OR 1 = 1 -

``

This entry indicates that the SQL of the application query is similar to:

```

SELECT * FROM items WHERE name = 'apple' OR 1=1 -

``
```

Quit queries with double (--) which effectively disables raw queries and allows an attacker to run arbitrary SQL code.

The "OR 1 = 1" part of the

SQL code will always be true, so this query returns all items in the database, not just those that match the search term.

To prevent malicious SQL injection, application developers should use invalid queries or prepared statements, separate user input from SQL code, and use valid input before using the question. Additionally, regular testing and analysis of application code can help identify and resolve SQL injection issues.

3.Broken Access Control –

An access control violation is a security breach that occurs when an application's access control is breached, allowing access to sensitive data or unauthorized processes. Broken access controls are the worst on the OWASP Top 10 Web Application Security Risks list.

Access control is the process of allowing or denying access to resources based on user identity, role, or other characteristics. An attacker could exploit this vulnerability to gain unauthorized access to sensitive data or processes if controls are not properly implemented.

There are several ways to break the check:

1.Insufficient authorization checks: An attacker could gain access if the application does not properly check whether the user is authorized to access a resource or function.

2. Insecure direct object references: If an application exposes internal IDs or other direct object references in its URLs or APIs, attackers can manipulate them to gain access to the disallowed resources layer.

3. Broken Authentication: If the application's authentication process is incorrect, an attacker can use it to gain unauthorized access to the application and its resources.

4. Compliance: If the application does not limit the rights of users, the attacker can support their rights and access resources that they should not enter.

5. Incompatible access control APIs: If an application exposes APIs that do not control access, attackers can access sensitive data or operate on APIs.

To prevent unauthorized access violations, application developers and administrators should implement access control best practices, such as using access control functions, providing appropriate permissions, restricting privilege escalation, and regularly monitoring and analyzing access controls.

In addition, access policies should be continually reviewed and updated to ensure they remain relevant and appropriate for changes in practices.

Cross Site Scripting-

Cross-site scripting (XSS) is a vulnerability that allows attackers to insert malicious code into web pages viewed by other users. The malware can steal sensitive information such as access credentials or personal information, or perform other malicious actions such as destroying websites or redirecting users to other malicious websites.

There are two types of XSS attacks:

1. Reflective XSS: Reflective XSS means that the attacker injects malicious code into the website and then sends that code back to the user. This happens when the application does not properly handle the user's input and reflects this in an error message or search page.
2. Stored XSS: Stored XSS occurs when an attacker injects malicious code into a web page, and then that code is stored on the server and served to all users viewing the page. This happens when the application does not properly process user input before saving it to the database.

XSS attacks can be particularly dangerous as they can be used to attack other security systems such as new browsers and security software. In addition, they can be difficult to detect and prevent, especially if attackers use advanced techniques to avoid detection.

To prevent XSS attacks, developers need to fix and clean up user input, for example using a library to use data or clearing user data eight before it is stored in file. Additionally, a web application firewall can be used to block malicious requests and prevent successful XSS attacks.

Finally, regular vulnerability scanning and penetration testing can help detect and address XSS vulnerabilities before they are exploited**.**

DDOS attack- A Distributed Denial of Service (DDoS) attack is a type of cyberattack in which multiple viruses or malicious computers (also known as "botnets") are used to flood a target or network with traffic, making it inaccessible. for legitimate users.

In a DDoS attack, an attacker first takes control of multiple computers, usually through malware or social engineering techniques. These infected computers are used to flood the target or network with traffic, bypassing their resources and making them unusable for legitimate users.

DDoS attacks can have serious consequences such as lost revenue, damaged reputation and potential liability. They can also be used as a barrier against interference from other, more negative factors such as data theft or network intrusion.

To prevent DDoS attacks, organizations can take a number of measures, including:

1. Network monitoring and analysis: Organizations can use network monitoring tools to identify and analyze traffic patterns and identify unusual activity that may indicate a DDoS attack.

2. Network Segmentation: Organizations can limit the impact of a DDoS attack by segmenting a network, isolating the impact from other networks.

3.Cloud-based mitigation services: Cloud-based DDoS mitigation services can help organizations reduce the impact of attacks by filtering traffic and blocking malicious traffic.

4. Security patches and new software: Make sure all software and systems are up to date and up to date, this helps minimize the risk of DDoS attacks.

5. Capacity Planning and Scalability: Ensuring that an organization's systems have enough capacity to handle traffic can help mitigate the impact of a DDoS attack.

In general, preventing DDoS attacks requires a combination of solutions and management processes and procedures, including security audits and employee training.

Security Misconfiguration –

A security misconfiguration is a security breach that occurs when security settings or configurations are not used or protected, resulting in systems and applications being compromised by attackers. This can occur at multiple technology layers, including web servers, databases, operating systems, and application processes.

Some examples of incorrect security configurations:

1. Default Passwords: Many systems and applications come with default usernames and passwords that can be easily used by the opposition.

2.Open ports and services: Insecure ports and services can allow attackers to access sensitive data or control systems.

3. Outdated software: Running outdated software can expose system vulnerabilities to know vulnerabilities that can be easily exploited by attackers.

4. Improper access control: Improper access control can allow unauthorized users to access sensitive information or perform actions for which they are unqualified.

5. Inappropriate permissions: Inappropriate permissions can allow an attacker to read, modify or delete incorrect data or information.

6. Unsecured Communication: Unencrypted or unsecured communication between systems or applications can allow data to be intercepted and manipulated.

Incorrect security configurations can have serious consequences such as data breaches, downtime and reputation.

Organizations can take a number of measures to prevent vulnerabilities, including:

1. Vulnerability testing: Regular vulnerability testing helps identify set bugs and other security issues before they are used.

2. Best Practices: Implementing security best practices, such as keeping systems up-to-date, restricting access, and following occupational safety procedures can help reduce the risk of misconfiguration.

3.System hardening: Strengthening systems and applications help mitigate the attack by not using services and ports, using security procedures and communication, and preserving security settings.

4. Regular Audits: Regular security audits can help identify and fix bugs and other vulnerabilities.

In general, security misconfigurations can pose a significant risk to an organization if not handled appropriately. Following best practices, regular audits and safety procedures can help reduce the risk of security breaches and other criminal issues.

INSECURE DESERIALIZATION – Insecure deserialization is a vulnerability that occurs when untrusted or corrupt data is deserialized without authentication and cleaning, leading to remote control crime, denial of service attacks, and other crimes.

Serialization is the process of converting an object into a stream of bytes that can be stored, transmitted, or reproduced later. Serialization is the reverse process of converting serialized data back into objects. Insecure deserialization vulnerabilities occur when untrusted data is deserialized without validation and cleaning.

An attacker could exploit this deserialization vulnerability by processing a malicious payload that, when deserialized, will execute arbitrary code, bypass authentication and authorization mechanisms, or unexpectedly change the data in the process.

Some examples of unsafe attacks are:

1. Remote Control Execution (RCE): An attacker can exploit a malicious deserialization vulnerability to breach the victim by sending a malicious program when serialized; application or system authorization.

2. Authentication and Authorization Bypass: An attacker can manipulate serialized data to bypass authentication and authorization mechanisms, access sensitive information, or perform unauthorized operations.

3.Denial of Service (DoS): An attacker can cause known denial of service (DoS) by sending large or illegal data, causing an application or system to use too many resources.

Organizations can implement a variety of measures to prevent security attacks, including:

1. Input Validation: Ensure all input data is valid, and the processed cleanup before deserialization helps prevent attackers from injecting malicious code.

2. Ensure safe deserialization: Implementing safe deserialization practices, such as deserializing only data from trusted sources and using secure deserialization libraries, can help reduce the risk of bad deserialization vulnerabilities.

3. Access controls: The use of access controls such as isolation policies, data whitelisting, and sandboxing can help reduce the impact of insecurity attacks.

4. Regular security checks: Regular security checks can help identify and fix security deserialization vulnerabilities.

In general, insecure deserialization vulnerabilities can pose a significant risk to organizations if not properly addressed.

Following best practices, regular audits, and safety protocols can help reduce the risk of unsafe serialization and other issues.

Sensitive Data Exposure –

A sensitive data breach is a security breach that occurs when sensitive or confidential information is exposed to unauthorized users or systems. This can take many forms, including insecure storage, transmission or data processing, weak encryption, and other related factors.

Sensitive information can include a variety of information, such as personal information, financial information, health information, property and trade secrets. When sensitive information is exposed, it can lead to theft, financial loss, reputation and other negative consequences.

Some examples of sensitive data breach include:

1.Unsecured storage: Storing sensitive information in plain text or weakly encrypted form can facilitate theft or abuse.

2. Weak transmission: Sending sensitive data over unencrypted channels (such as HTTP) can allow attackers to intercept and steal data.

3. Weak encryption: Attackers can use weak encryption techniques or abuse encryption to decrypt sensitive data.

4. Inappropriate controls: Inappropriate controls may allow unauthorized users to access sensitive information or perform unauthorized actions.

5. Vulnerable applications: Weak security applications such as SQL injection or cross-site scripting can be used by attackers to access and steal sensitive data.

Organizations can take various measures to prevent data breaches, including:

1.

Encryption: Encrypting sensitive information in transit and at rest helps prevent unauthorized access and theft.

2. Security Management: To help prevent unauthorized access and misuse, use access controls that restrict who can access sensitive information and how it can be accessed.

3. Regular security testing: Regular testing of systems and applications for vulnerabilities through penetration testing or vulnerability scanning can help identify and fix security issues.

4. Hiding information: Hiding sensitive information, such as a credit card or social security number, helps prevent disclosure or misuse.

In general, the breach of sensitive data can pose a significant risk to organizations if not handled appropriately. Following best practices, regular audits and safety procedures can help reduce the risk of data leaks and other security issues.

XML External entities –

XML External Entities (XXE) is a vulnerability that occurs when an XML parser performs external operations on an XML document without validation and sanity. An external field is any data that is set outside the XML document but referenced within.

An attacker could exploit the XXE vulnerability by sending a custom XML file that, when analyzed, contains external malicious domains that, when analyzed, could lead to attacks that can read sensitive information, enforce unauthorized rights, or cause denial of service attacks.

Outdoor areas are of two types: indoor and outdoor. Internal fields are defined in the XML document itself, while external fields are defined outside the file, for example, in a separate file or remote control.

The vulnerabilities arise when the XML parser parses and executes external transactions without validation and sanity, leading to various attacks. Some examples of XXE attacks include:

1. Remote Inclusion of Information (RFI): Attackers can cause RFI to attack by using XXE vulnerabilities to include remote information and breaches.

2. Local File Inclusion (LFI): An attacker can exploit the XXE vulnerability to read sensitive data such as configuration files or password files from an LFI attack.

3. Denial of Service (DoS): An attacker can cause a DoS attack by sending malicious XML files containing excessive resource-consuming external malicious sites.

Organizations can implement a variety of measures to prevent XXE attacks, including:

1. Input Verification: Ensure all input data is valid and cleared before serving to protect attackers from outside shots.

2.Disable External Entities: Disabling external entities in XML parsers helps prevent XXE vulnerabilities.

3. Use a secure parser: Using an XML parser designed to protect against XXE vulnerabilities, such as the latest version of a popular XML parser, will help reduce the risk of XXE attacks.

4. Access Control: Use access controls such as data whitelisting to help prevent unauthorized access to sensitive data or information.

Insufficient logging and monitoring –

Inaccessibility and lack of monitoring are different aspects of security that occur when organizations lack access and monitoring to detect and respond to security incidents.

Logs are records of events that have occurred in a system or application and may contain information such as user actions, system events, and security-related events. Monitoring includes checking logs and other data sources for signs of security or vulnerability.

Inadequate decision making and monitoring can make it difficult for organizations to identify security incidents, investigate incidents that have occurred, and take appropriate corrective action. This can result in delayed incident response, risk of data breaches and damage to reputation.

Some examples of poor access and surveillance include:

1. Not logging enough: If organizations don't log enough, they can miss important security-related information, making them harder to search and investigate.

2. Not checking the logs: If the logs are not monitored and checked, the security situation will not be detected and the response time will be extended.

3.Not keeping logs long enough: If logs are not kept long enough, it will be difficult to investigate past events or identify attack patterns.

To address logging and maintenance deficiencies, organizations can take a number of measures, including:

1. Collect and retain logs: Collect and retain logs long enough to help ensure that all information is relevant to the investigation of the incident.

2. Implement automated monitoring: Using monitoring tools that can detect and alert security teams to incidents will help reduce response time.
3. Regular log analysis: Regular log analysis can help identify security incidents and reduce incident response time.

4. Definitive Incident Response Processes: A clear definition of processes and procedures will help ensure that security incidents are resolved in a timely and timely manner.

5.Regular safety training: Providing safety training to employees can help raise awareness of the importance of hacking and monitoring and how to respond to a safety issue.

In general, enabling access and monitoring is critical for organizations to detect and respond to security events. By taking steps to address a lack of discretion and oversight, organizations can improve their security and reduce the risk of data breaches and other security incidents.

Injection -

Injection is a security breach that occurs when an attacker can inject malicious code or data into an application or system. An attacker could exploit this vulnerability to perform unauthorized actions, such as accessing or modifying sensitive data, bypassing authentication checks, or executing malicious code on a target.

There are several injection vulnerabilities, including the following:

1. SQL Injection (SQLi): SQL injection occurs when an attacker can inject SQL commands into the application's input field. This could allow an attacker to take unauthorized actions, such as accessing or modifying sensitive information stored in the database.

2. Cross-site scripting (XSS): Cross-site scripting happens when an attacker can inject malicious code (usually JavaScript) into web pages used by other users. This could allow an attacker to steal sensitive information such as login credentials or credit card numbers.

3. Command injection: Injection occurs when an attacker can inject commands into the application's input fields or parameters.

This could allow an attacker to perform unauthorized actions, such as running commands on the target.

4. LDAP Injection: LDAP injection happens when an attacker is able to inject malicious code or data into an LDAP query. This could allow an attacker to access or modify sensitive information stored in the LDAP directory.

To prevent bad shots, organizations can take a variety of measures, including:

1.

Input Validation: To help prevent attackers from injecting malicious data or information, ensure that all input data is valid and cleaned before processing.

2. Parameterized Queries: Using parameterized queries in database applications will help prevent SQL injection vulnerabilities.

3. Output Encoding: Encoding all output data so that it appears as plain text helps prevent corruption of the written source.

Following best practices, regular safety assessments and industry standard safety procedures can help reduce injection failure, quality and other safety risks.

3.2.1 Analytical Model Development :

As the system is primarily a data-driven system, there was no need for an analytical model in this project.

3.2.2 Computational Model Development :

Since the system relies on the use of various AWS services, we use AWS CloudFormation to create the necessary processes on the AWS platform. This helps us deliver systems quickly and easily without human intervention.

3.2.5 Statistical Model Development :

Because the system is based on data, we use statistical models to analyze and generate data. We use an AWS Lambda function to analyze the data stored in the DynamoDB database and generate reports based on the analysis results.

Experiments and Result Analysis

4.1 Experiments and Result Analysis:

Various tests were conducted to evaluate the effectiveness of the system and the results were analyzed. Testing is done with a benchmark data size of 50 users who each post their activity daily for 30 days.

Test 1: Performance Evaluation

The first test is to measure the performance of the system in terms of the time it takes to send the state and the time it takes to store it. Testing was done on an Intel i7 processor with 16GB of RAM running Windows 10.

To simulate user behavior, a script was created to send an event every 24 hours and another script to store all users. situations every 24 hours.

The test was run for 30 days, recording the time it took for each user to submit and log events.

The test results show that the average time to submit a status is 3 seconds and the average time between states is 2 seconds. The system performed well under heavy loads and there were no lags during testing.

Experiment 2: User Feedback Evaluation

The second experiment aims to evaluate the user experience of the system by collecting user feedback. Create and distribute a survey to 50 users, ask them to rate the system on a scale of 1 to 10, 10 being the highest score.

The survey includes various aspects of the system such as ease of use, user interface and overall user experience. Evaluation results showed that the system scored an average of 8 out of 10 and users were satisfied with the system.

Third test: Accuracy Test

The third test is to test how accurately the system monitors the activity status of each user. To do this, each user's status is analyzed based on the actual job they completed within the 30-day period.

The test results show that the accuracy of the system has reached 95%, which shows that the system can track the work status of each user.

In general, the test results show that the central employee tracking system created with AWS and Angular can monitor the work of all users and provide sufficient pressure on customers. Future improvements may increase the efficiency and power of the system, as well as add additional features such as reports and analytics.

Conclusions:

5.1 Conclusions:

As a result, the project has solved the problem of the company's declaration that it does not have a framework to view and track the job status of its employees. The built-in web application provides a great way for members to post their daily or weekly activities that can be viewed by their admins. The system also includes functionality to notify users of delivery status and make timely adjustments. With the access control role, only authorized employees can view events, keeping the information confidential.

Using AWS on the backend and Angular on the front-end makes the website scalable, powerful and efficient.

The application is secure and follows the OWASP Top 10 Guidelines to avoid SQL vulnerabilities. The cloud-based architecture allows the system to scale easily as the team grows.

5.2 Future Scope:

The system can be further enhanced by integrating with existing management tools such as JIRA. This will allow users to directly submit and track work in the system. The application can also be developed using machine learning algorithms to predict the probability of on-time delivery or to identify patterns in employee performance.

5.3 Applications contributions:

Web application design can be used for any organization that needs a framework to view and monitor employee work status. The system can be customized to meet a specific organization and help improve team productivity and accountability.

References:

1. AWS CloudFormation documentation: https://aws.amazon.com/cloudformation/

2. AWS API Gateway documentation: https://docs.aws.amazon.com/apigateway/

3. AWS Lambda Documentation: https://docs.aws.amazon.com/lambda/

4. Angular Documentation: https://angular.io/docs

5. OWASP Top Ten Projects: https://owasp.org/www-project-top-ten/

6. Rest API features, principles and challenges:
https://www.knowledgehut.com/blog/programming/rest-api

7. A study on task management system :
https://www.researchgate.net/publication/311461505_A_study_on_task_management_system