Asset Management System (AMS)

A PROJECT REPORT

Submitted by

Dharvakumar Mistry
(17CP301)

in partial fulfillment for the award of the degree of

B. TECH. (COMPUTER ENGINEERING)

Under the course of

CP446: FULL SEMESTER EXTERNAL PROJECT

BIRLA VISHVAKARMA MAHAVIDYALAYA
(ENGINEERING COLLEGE)

(An Autonomous Institution)

VALLABH VIDYANAGAR
Affiliated to

GUJARAT TECHNOLOGICAL UNIVERSITY, AHMEDABAD

Academic Year: 2019 – 2020

B.V.M. ENGINEERING COLLEGE, VALLABH VIDYANAGAR-388120

APPROVAL SHEET

The project work entitled “**Asset Management System(AMS)**" carried out by “Dharvakumar Mistry with ID No: 17CP301” (which was carried out at ***Crest Data Systems***) is approved for the submission in the course CP446, Full Semester External Project for the partial fulfillment for the award of the degree of B. Tech. (Computer Engineering).

SIGNATURE

(Name & Designation)

Date:

Place:

CERTIFICATE

This is to certify that Project Work embodied in this project report titled “**Asset Management System(AMS)**" was carried out by “Dharvakumar Mistry with ID No: 17CP301” which was carried out at ***Crest Data Systems*** under the course CP446, Full Semester External Project for the partial fulfillment for the award of the degree of B. Tech. (Computer Engineering). The followings are the supervisors at the institute.

Date:

Place:

(Prof. Kirti Sharma)

Project Guide

Assistant Professor of Computer Engineering

(Prof. Pranay Patel)

Project Guide

Assistant Professor of Computer Engineering

(Dr. Darshak G Thakore)

Prof. & Head

Computer Engineering Department, BVM

B.V.M. ENGINEERING COLLEGE, VALLABH VIDYANAGAR-388120

DECLARATION OF ORIGINALITY

I hereby certify that we are the sole authors of this report under the course CP446 Full Semester External Project and that neither any part of this report nor the whole of the report has been submitted for a degree to any other University or Institution.

I certify that, to the best of our knowledge, the current report does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations or any other material from the work of other people included in our report, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that we have included copyrighted material that surpasses the boundary of fair dealing within the meaning of the Indian Copyright (Amendment) Act 2012, we certify that we have obtained written permission from the copyright owner(s) to include such material(s) in the current report and have included copies of such copyright clearances to our appendix.

I declare that this is a true copy of the report, including any final revisions, as approved by the report review committee.

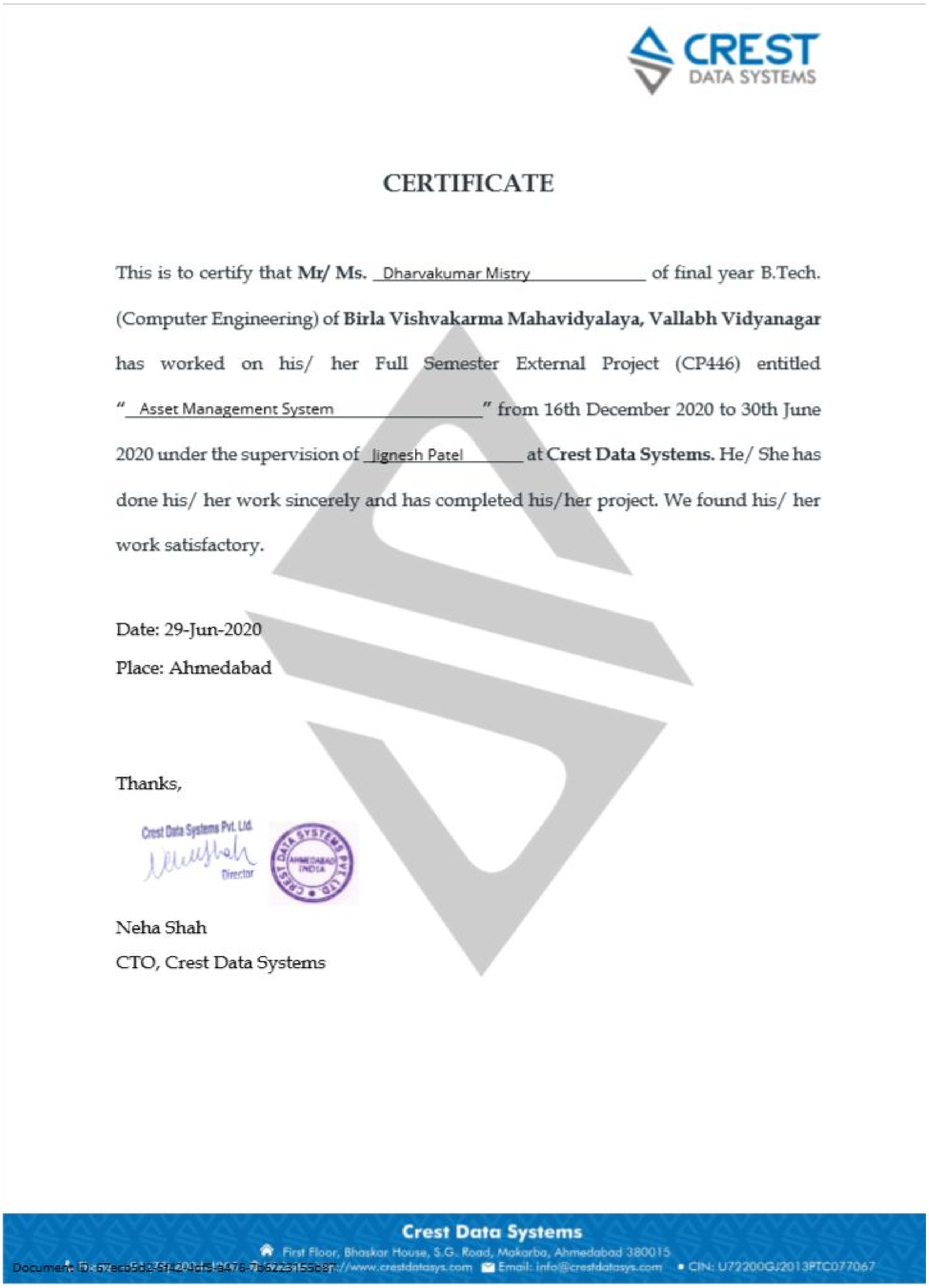
We have checked the write up of the present report using an anti-plagiarism database and it is in the allowable limit. Even though later on in case of any complaint pertaining to plagiarism, we are solely responsible for the same and we understand that as per UGC norms, University can even revoke the degree conferred to the student submitting this report.

Date:

Institute code: 007

Dharvakumar Mistry (17CP301)

CERTIFICATE



CREST DATA SYSTEMS

CERTIFICATE

This is to certify that **Mr/ Ms.** Dharvakumar Mistry of final year B.Tech. (Computer Engineering) of **Birla Vishvakarma Mahavidyalaya, Vallabh Vidyanagar** has worked on his/ her Full Semester External Project (CP446) entitled "Asset Management System" from 16th December 2020 to 30th June 2020 under the supervision of Jignesh Patel at **Crest Data Systems.** He/ She has done his/ her work sincerely and has completed his/her project. We found his/ her work satisfactory.

Date: 29-Jun-2020

Place: Ahmedabad

Thanks,

Crest Data Systems Pvt. Ltd.
[illegible]
Director

CREST DATA SYSTEMS PVT. LTD. AHMEDABAD INDIA

Neha Shah

CTO, Crest Data Systems

Crest Data Systems

First Floor, Bhaskar House, S.G. Road, Makarba, Ahmedabad 380015

Document ID: 67eca5d2-5f42-4df9-8476-7b6223155b87 ://www.crestdatasys.com Email: info@crestdatasys.com • CIN: U72200GJ2013PTC077067

ACKNOWLEDGMENT

I would sincerely like to thank **Prof. Kirti Sharma, Prof. Pranay Patel** (Department of Computer Engineering) for the unconditional support during the whole session of study and development and for guiding me throughout the whole internship period. They provided me with a favorable environment, without them, I would not have achieved my goal. They had always been there for me despite their busy schedule and were always a great source of inspiration. They had been easily approachable during and even after college hours. I sincerely thank them for that.

I would also like to thank **Mr**. **Jignesh Patel**, Technical Lead (Crest Data System Pvt. Ltd.) For supporting me during the whole internship period. Being with me all the time face to face and guiding me within a busy time schedule. I would like to thank him for that. I would also like to thank my other senior colleagues who also helped me whenever I required any kind of support.

A blend of gratitude, pleasure, and great satisfaction is what I feel to convey my indebtedness to all those who have directly and indirectly contributed to the successful completion of the project.

Dharvakumar Mistry

TABLE OF CONTENTS

Title **Page No**

ABSTRACT i
TABLES ii
LIST OF FIGURES ii
LIST OF TABLES iii
1. INTRODUCTION 1
1.1. Project Summary 1
1.2. Purpose 1
1.3. Objective 1
1.4. Scope 2
1.5. Tools and Technology Used 2
2. ABOUT THE SYSTEM 6
2.1. System Requirements study 6
2.1.1. User Characteristics 6
2.1.2. Hardware and Software Requirements 6
2.1.3. Assumptions 6
2.1.4. System Consideration 7
2.2. System Analysis 7
2.2.1. Study of current System 7
2.2.2. Problem and Weakness of current System 7
2.2.3. Requirements of System 7
3. ANALYSIS 10
3.1. Use Cases 10
3.1.1. Application Use Cases 10
3.2. Class Diagram 13
4. DESIGN 15
4.1. System Ui Architecture 15
4.2. System Core Architecture 16
4.2.1. Web Server 17
4.2.2. Angular 17
4.2.3. API Gateway 18
4.3. Data Models 20
4.4. WireFrames 25
5. IMPLEMENTATION 35
5.1. Implementation Environment 35
5.2. Key Implementation Objective 35
5.3. Implementation Modules 35
5.4. Implementation Steps for Modules 36
6. TESTING 37
6.1. Test Planning 37
6.2. Testing Strategy 37
6.2.1. Unit Testing 37
6.2.2. Integration Testing 37
6.3. Test Suits Design 38

6.4. API and Ui Validation Snapshots...40
6.5. Application Snapshots…………………..……………………………45
7. Conclusion and Future Extension..47
7.1. Conclusion ……..47
7.2. Future Extension…..47

REFERENCES...48

ABSTRACT

Every organization has its own Asset Management System in order to perform resource activities. Managing assets or utilizing assets or keeping track of assets is a significant task of the IT team. The main work of the IT team is to maintain all assets of the company like laptops and any other devices and also maintain the condition of that Asset. In order to support IT's, there are some electronic-based systems called Asset management systems (AMS). Many Organizations maintain asset details with excel sheets which are a very tedious job. But this application is a cost-effective one that allows them to manage their asset's data in a simple manner. This project belongs to a category of the web application that can be accessed through PC with an internet connection. This Asset Management System allows the IT team to provide various types of permission to other IT members and also manage Asset's data, status, category, location, etc. Asset Management systems are used to maintain assets of the company and require a significant amount of time if we do it without software.

TABLES

LIST OF FIGURES

Figure Number	Figure Name	Page Number
1.5.1	Swagger UI of Application	3
1.5.2	Docker container in Execution	5
3.2.1	Class diagram of the system	14
4.1.1	System UI architecture	15
4.2.1	System Core Architecture	16
4.4.1	Dashboard	25
4.4.2	Asset-list-with-filter	26
4.4.3	Add-asset	27
4.4.4	Add-Laptop	28
4.4.5	Asset-assign-to-employee	28
4.4.6	Assign-asset-to-room	29
4.4.7	Assign Activity	30
4.4.8	Report-asset-activity	31
4.4.9	Report-asset-details	32
4.4.10	Recent-activity	33
4.4.11	Permission	34
6.4.1	Asset-status-validation	40
6.4.2	Company-location-validation	41
6.4.3	Room-validation	42
6.4.4	Supplier-validation	43

6.4.5	Asset-category-validation	44
6.5.1	Dashboard	45
6.5.2	Asset-list	45
6.5.3	Asset-list-filter	46
6.5.4	Add-asset-into-system	46

LIST OF TABLES

Table Number	Table Name	Page Number
4.2.3	API Gateway	18
4.3.1	Data Models: Asset	20
4.3.2	Data Models: Status	21
4.3.3	Data Models: Rooms	21
4.3.4	Data Models: Activity Log	22
4.3.5	Data Models: Laptop	23
4.3.6	Data Models: Permission	24
4.3.7	Data Models: Supplier	24
6.3	Test Suites Design	38

Chapter 1
Introduction

1.1 PROJECT SUMMARY

Asset Management System is an application used by the company to automate and keep track of the equipment and inventory that are vital to the day-to-day operation of their businesses. Exactly how those assets are managed, though, is highly variable. A lot of organizations are managing their equipment and inventory through a manual process, including spreadsheets over which an employee or employees are tasked with the responsibility of maintaining. So, this application helps to keep track of these vital assets and manages efficiently and provides an easy way to use.

This project is intended to replace the Manually managed system and provide a platform that is more efficient, reliable, and robust.

1.2 PURPOSE

IT asset management ties the assets with the IT infrastructure of the organization. With a robust asset management system, management and IT professionals can review and monitor all types of assets within the organization. The information can be used to make detailed decisions about the purchase and other aspects of the asset's life cycle. So, the purpose of IT asset management is to :

1.2.1. Effectively help manage the assets.
1.2.2. Improve visibility of assets.
1.2.3. Ensure optimum utilization of assets.
1.2.4. Reduce IT and software costs.
1.2.5. Ensure compliance with regulatory requirements.

1.3 OBJECTIVE

The objective of the IT asset management system is to provide an organization with a deep knowledge of its information systems to use this information for the identification and rapid resolution of problems. And also complete information about the configuration of assets and their relationship among each other and also make sure that this data is available whenever it is needed.

The goal of this project is to reduce the effort required to maintain a variety of assets and automate the task of managing assets. Using the software can help in efficient resource planning. The main focus of this project is to provide management functionalities like Manage Asset, Manage Asset category, Manage Status, Manage Rooms, Employee Listing, Manage Permission, Generate Reports, and the Activity Logs.

1.4 SCOPE

Following features are there in the scope:

- Keep track of the company's hardware and software purchases. That determines the exact time to renew a license, update software, or replace a piece of equipment.
- Effective Resource Planning: Maximize utilization and reduce cost.
- Increase Security: Protection of data found in the company systems. Also, know the status of their security tools.
- Definition of service levels: Helps IT specialists define the level of service that can be offered to customers based on the number of existing assets.

1.5 TOOLS AND TECHNOLOGY USED

Following are the Technology used in the system:

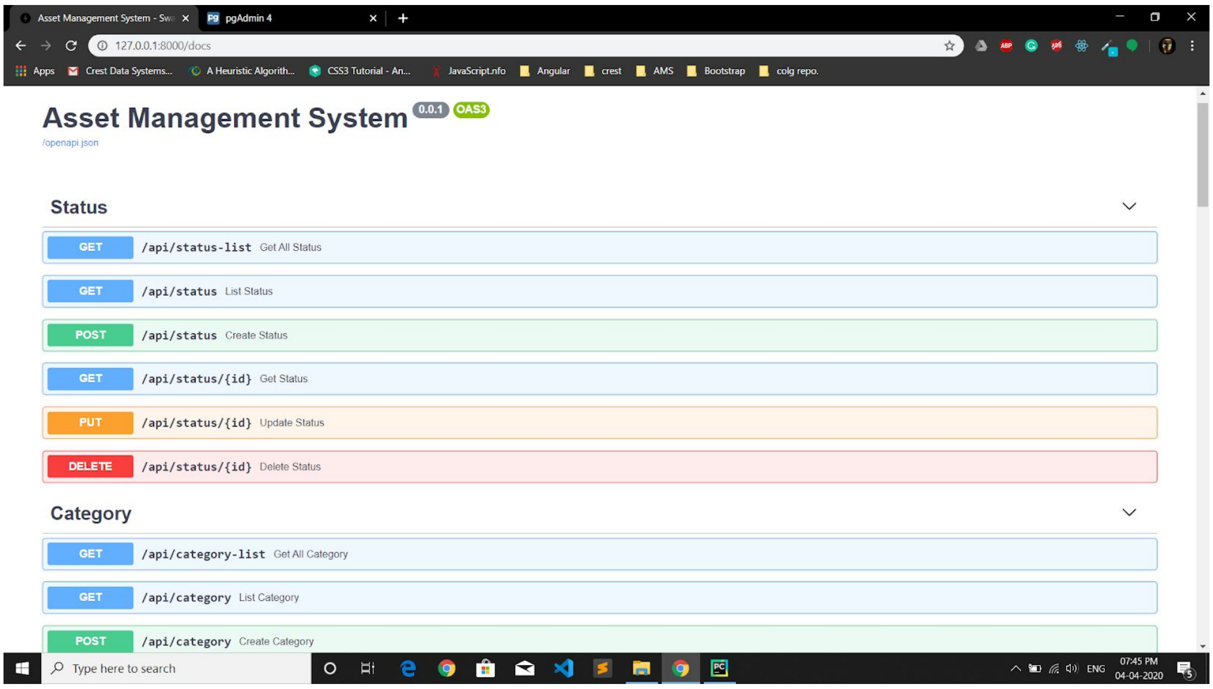
- **Backend:** Python 3.8, FastAPI, SQLAlchemy, Redis Cache
- **Frontend:** Angular 9, Angular Material
- **Database:** PostgreSQL

Distributed 3-tier Architecture. Each tier contains a separate Docker Container.

Overview of FastAPI [1]:

FastAPI is a modern, fast (high-performance), a web framework for building APIs with Python 3.6+. FastAPI provides the following:

- OpenAPI for API creation, including declarations of path operations, parameters, body requests, security, etc. It also generates automatic data model documentation with JSON Schema.
- Swagger UI for Interactive API documentation and exploration web user interfaces by exploration, call, and test your API directly from the browser.


Asset Management System 0.0.1 OAS3
/openapi.json

Status

GET /api/status-list Get All Status
GET /api/status List Status
POST /api/status Create Status
GET /api/status/{id} Get Status
PUT /api/status/{id} Update Status
DELETE /api/status/{id} Delete Status

Category

GET /api/category-list Get All Category
GET /api/category List Category
POST /api/category Create Category

Fig. 1.5.1 Swagger UI of Application

FastAPI provides validation for most python data types including JSON object, JSON array, String, Number with min and max values, etc. It also provides validation for exotic types like Email, URL, UUID, etc.

Security and authentication are also integrated into FastAPI. It gives security on HTTP Basics and authentication provides by OAuth2 (also with JWT tokens). It uses uvicorn for the server that loads and serves our application.

I preferred FastAPI for this project because of its high performance, robust, easy to use, and standard-based features.

Overview of SQLAlchemy [2]:

SQLAlchemy is the toolkit and Object Relational Mapper which is written in Python and provides the flexibility of SQL for application development.SQLAlchemy provides object-relational mapper (ORM), using which classes can be mapped to the database, thereby allowing the object model and database schema to develop in a cleanly decoupled way from the beginning.

I preferred this because it will be beneficial for the application as it provides good support to FastAPI. Here I use the Postgres ORM model for development.

Overview of PostgreSQL [5]:

PostgreSQL is a powerful, open-source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. PostgreSQL differs from other relational database management systems for its proven architecture, reliability, data integrity, robust feature set, extensibility, and the dedication of the open-source community behind the software to consistently deliver performant and innovative solutions. PostgreSQL runs on all major operating systems and has extensive support for ACID properties.

As the project requires a database having features like better support for Parallel merge joins and Parallel aggregation, So the PostgreSQL would be most suitable for this project. PostgreSQL is preferable for this project as a schema for the table is fixed of data models and because of PostgreSQL object-relational feature relation between data models would be well defined. Also, joins between data tables would easily be handled in the PostgreSQL database. PostgreSQL performance is utilized best in systems requiring the execution of complex queries. PostgreSQL has roles and inherited roles to set and maintain permissions. PostgreSQL has native SSL support for connections to encrypt client/server communications. It also has Row Level Security.

Overview of Angular 9 [6] and Angular Material [7]:

Angular is open-source frameworks for building web and mobile applications. A new version of Angular 9 is smaller, faster, and easier to use. With Angular 9, the community can benefit from smaller, high-performance applications, and better developer experience.

From Angular 9 the Ivy compiler is available for all apps. The main benefit of Ivy is that it is able to significantly reduce the size of large-sized applications.

Angular Material is a UI component library for Angular developers. Angular Material components help in constructing attractive, consistent, and functional web applications while adhering to modern web design principles like browser portability, device independence, and graceful degradation. It helps in creating faster, beautiful, and responsive websites.

Overview of Docker [8]:

Docker is an open platform for developing, shipping, and running applications. Docker enables us to separate our applications from our infrastructure so we can deliver software quickly. With Docker, we can manage infrastructure in the same ways we manage our applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, we can significantly reduce the delay between writing code and running it in production.

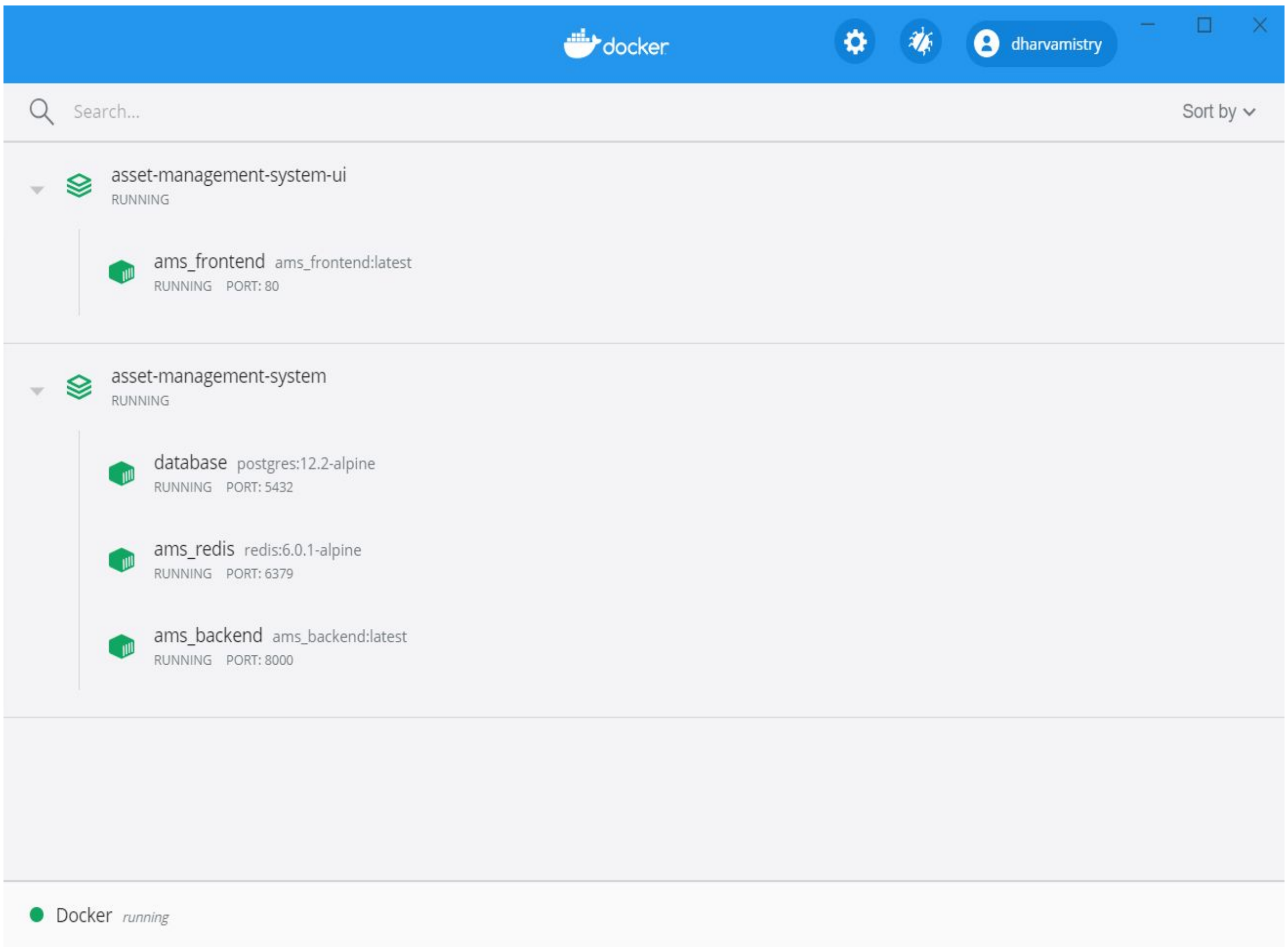
In this project, I have used four docker containers for each tier: Backend, Frontend, Redis, and Database, which will increase the production rate significantly.

All the Docker containers use Alpine Version Image as a base image, on top of that each container has its own separate volumes that are maintained and updated with the content of each docker container respectively.

We have two separate docker-compose files one for the Frontend and one for the Backend.

The Frontend docker-compose file includes AMS Frontend Image which internally uses Node's Alpine Image and NGINX's Alpine Image and also has instructions regarding creating the Production Build of the project and COPY the BUILD into the NGINX container volume.

The Backend docker-compose comprises the Postgres, and AMS Backend Images which internally uses Python v3.8.2 Alpine image. The environment variables are set for all the images using the environment property of each docker-compose service.



asset-management-system-ui
RUNNING

ams_frontend ams_frontend:latest
RUNNING PORT: 80

asset-management-system
RUNNING

database postgres:12.2-alpine
RUNNING PORT: 5432

ams_redis redis:6.0.1-alpine
RUNNING PORT: 6379

ams_backend ams_backend:latest
RUNNING PORT: 8000

Docker running

Fig. 1.5.2 Docker Containers in Execution

Chapter 2
About the system

2.1 SYSTEM REQUIREMENTS STUDY

2.1.1 USER CHARACTERISTICS

Application intended to use by mainly two users :

1. Super Admin
2. IT supervisor

Assets management will be managed by both users. The application would provide asset and user mapping. Users would manage assets like being able to add assets, update the asset, delete an asset, assign/unassign assets to users. SuperUser(Admin) will assign permission to the other IT supervisor. The application would also generate reports based on duration, asset details, employee details, and would also manage activity logs.

2.1.2 HARDWARE AND SOFTWARE REQUIREMENTS

Client Requirement

● Processor:	Dual Core 2.0 GHZ or latter CPU
● RAM:	4 GB
● Browser:	Chrome, Firefox

Server Requirement

● Processor:	4 Core CPUs
● RAM:	4 GB RAM minimum
● Hard Disk:	100 GB storage minimum

2.1.3 ASSUMPTIONS

It is assumed that:

- The Employee database will be accessible using the HRMS APIs.
  (HRMS is the company's internal system that manages the Employee Details.)
- Authorized users will be able to assign different permissions to the other users.
- The IT supervisor would be managing the Meeting/Conference/Training
- Rooms Assets (TV issues, WebCam Issues, Projector and its accessories, etc).

2.1.4 SECURITY CONSIDERATION

These are security considerations :

- The application components should not use any third-party modules, frameworks which have known vulnerabilities.
- In order to provide the best security, the web application would use JWT authentication tokens.
- To provide HTTPS support the web application would require to have its own dedicated HOST with its dedicated IP address and SSL Certificate.

2.2 SYSTEM ANALYSIS

2.2.1 STUDY OF CURRENT SYSTEM

Companies manage their equipment and inventory through a manual process, including spreadsheets over which an employee or employees are tasked with the responsibility of maintaining.

2.2.2 PROBLEMS AND WEAKNESSES OF CURRENT SYSTEM

Due to manual processes, IT supervisors suffer problems like forgotten items, broken or lost equipment, human error, hours spent searching for items that have mysteriously disappeared, inaccurate information needed for accounting and compliance, customer dissatisfaction.

2.2.3 REQUIREMENTS OF SYSTEM

2.2.3.1. Functional Requirements

Functional requirements involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish.

- **Functional requirement of API Endpoints**
  - **Manages Asset Details**

    By this functionality, the IT supervisor can maintain the asset detail by Adding new assets, editing existing asset details, and deleting assets not required from the system.

    Input: Asset Details

    Output: Record changes in the system

    Processing: Fire Query of PostgreSQL and perform changes in the database

- **Assignment or Unassignment of Asset to Employees or Rooms**

  IT supervisor can Assign/Unassign/Reassign the variant assets to the single employee or group of employees or maybe room.

  Input: Employee details, Room Details, Asset Details

  Output: Assign or Unassign Assets to employee or room

  Processing: Based on the Asset availability assignment would be performed

- **Set Notification and Get Notified on Events**

  IT supervisors would set reminders regarding assets like time to renew a license, update software, or replace a piece of equipment. Also, set a notification for the employee when the asset is assigned for a specific time duration. Get notified on events set in Notification when it occurs.

  Input: Time Interval for Asset or Employee

  Output: Acknowledgement on the successful set reminder

  Processing: Set Time period for the specified asset or employee and record in the database

- **Generate Reports of Assets and Employees**

  IT supervisors can generate the report based on filtering criteria and export it into formats like document or PDF.

  Input: Filtering criterion of assets and employee

  Output: Asset-Employee details based requirements.

  Processing: Perform Search Query on Employee data and/or Asset data then show data in well-formed UI.

2.2.3.2. Non-Functional Requirements

2.2.3.2.1. Security

The system uses HTTPS to transmit data so it passes through SSL. Authentication would be done by JWT token-passing mechanism.

2.2.3.2.2. Confidentiality

The system protects sensitive data of employees and allows only authorized access to the data.

2.2.3.2.3. Reliability

The application consistently performs the specified functions without failure by an efficient error handling mechanism.

2.2.3.2.4. Usability

The system is easy to use as Angular 9 is used as a frontend and due to Angular Material Component, controls look attractive and easy to use.

2.2.3.2.5. Maintainability

The ease with which faults in a software system can be found and fixed. The system would be extensible by adding new functionality.

2.2.3.2.6. Reusability

The API endpoints are reusable and also able to integrate with other systems efficiently.

Chapter 3
Analysis

3.1 USE CASES

3.1.1 APPLICATION USE CASES

This section describes the use cases of the system.

3.1.1.1. Ability to Manage Assets

Description:

Users would be able to manage the details for assets.

Acceptance criteria:

1. Users should be able to add a new asset.
2. Users should be able to update an existing asset.
3. Users should be able to delete an existing asset.
4. Users should be able to view assets based on filtering criteria or using search.
5. Users should be able to Import multiple assets from the file(.XLS or .CSV).
6. Users should be able to Export the assets(.PDF, .XLSX, .CSV).
7. Users should be able to assign/unassign assets to the user/room.
8. Users should be able to view the asset details.
9. Users should be able to send asset for maintenance
10. Users should be able to extend the warranty.
11. Users should be able to send notification (mail) to the owner of the asset.
12. Users should be able to add an image of an asset.

3.1.1.2. Ability to Manage Status

Description:

Users would be able to manage the asset status.

Acceptance criteria:

1. Users should be able to add an asset status.
2. Users should be able to update the existing asset status.
3. Users should be able to delete an existing asset status.
4. Users should be able to view status.
5. Users should be able to view different assets based on status filtering criteria or using a status search.

3.1.1.3. Ability to Manage Supplier

Description:

Users would be able to manage the asset supplier.

Acceptance criteria:

1. Users should be able to add suppliers.
2. Users should be able to update the existing suppliers.
3. Users should be able to delete existing suppliers.
4. Users should be able to view the supplier.
5. Users should be able to view different assets based on Supplier filtering criteria or using a Supplier search.

3.1.1.4. Ability to Manage Permission

Description:

Authorized Users would be managing the permission of the other users.

Acceptance criteria:

1. Authorized Users should be able to assign/unassign permissions to the other user.
2. Permission can be based on modules like assets, status, category, employee listing, and reports. For example, If only assets permission is assigned to the user then he can only be able to manage assets.

3.1.1.5. Ability to Manage log

Description:

The system would be managing the logs of every action made by the IT supervisor or Employee.

Acceptance criteria:

1. Users should be able to view the activity logs for any time duration, for any assets, for any employee.
2. Users should be able to export the activity logs.
3. Users should be able to perform auditing.

3.1.1.6. Ability to Generate Reports

Description:

Users would be able to generate reports based on criteria like status, duration, Asset Category.

Acceptance criteria:

1. Users should be able to generate asset reports.
2. Users should be able to generate employee reports.

3.1.1.7. Ability to Manage Rooms

Description:

Users can manage the room information and details regarding assets assigned to that room.

Acceptance criteria:

1. Users should be able to add a room.
2. Users should be able to update an existing room.
3. Users should be able to delete an existing room.
4. Users should be able to view rooms based on filtering criteria or using search.
5. Users should be able to see which asset is assigned to which room using the room details (id/name).

3.2 CLASS DIAGRAM

Asset

This class stores asset details(Id, name, model, arrival time, category, status, description). Users can manage and search assets in this class. Users can sort assets.

Asset-Category

This class stores asset categories like Mouse, Keyboard, Laptop, etc.

Asset-Status

This class stores asset status like assign, unassign, new, destroyed, etc.

Employee

This class lists all employees of the organization. Based on permission associated with the users, they can manage assets.

Location

It contains details regarding the locations of the organization.

Room

This class contains details regarding rooms in the organization and assets assigned to these rooms.

Supplier

It contains the supplier details and assets delivered by them.

Permission

This class provides permission to users. Based on permission, users can manage assets.

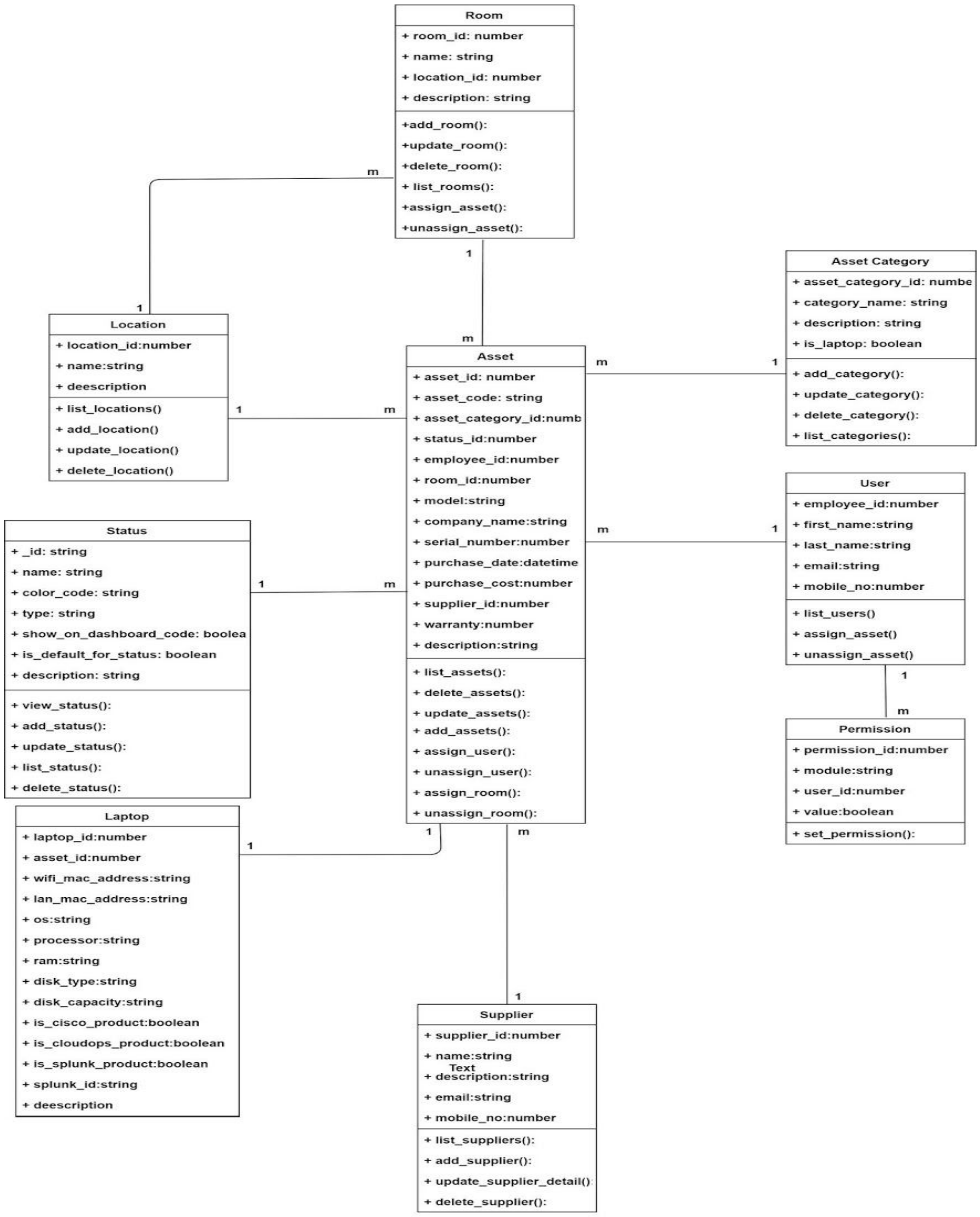


Fig. 3.2.1 Class diagram of the system

Chapter 4
Design

4.1 SYSTEM UI ARCHITECTURE

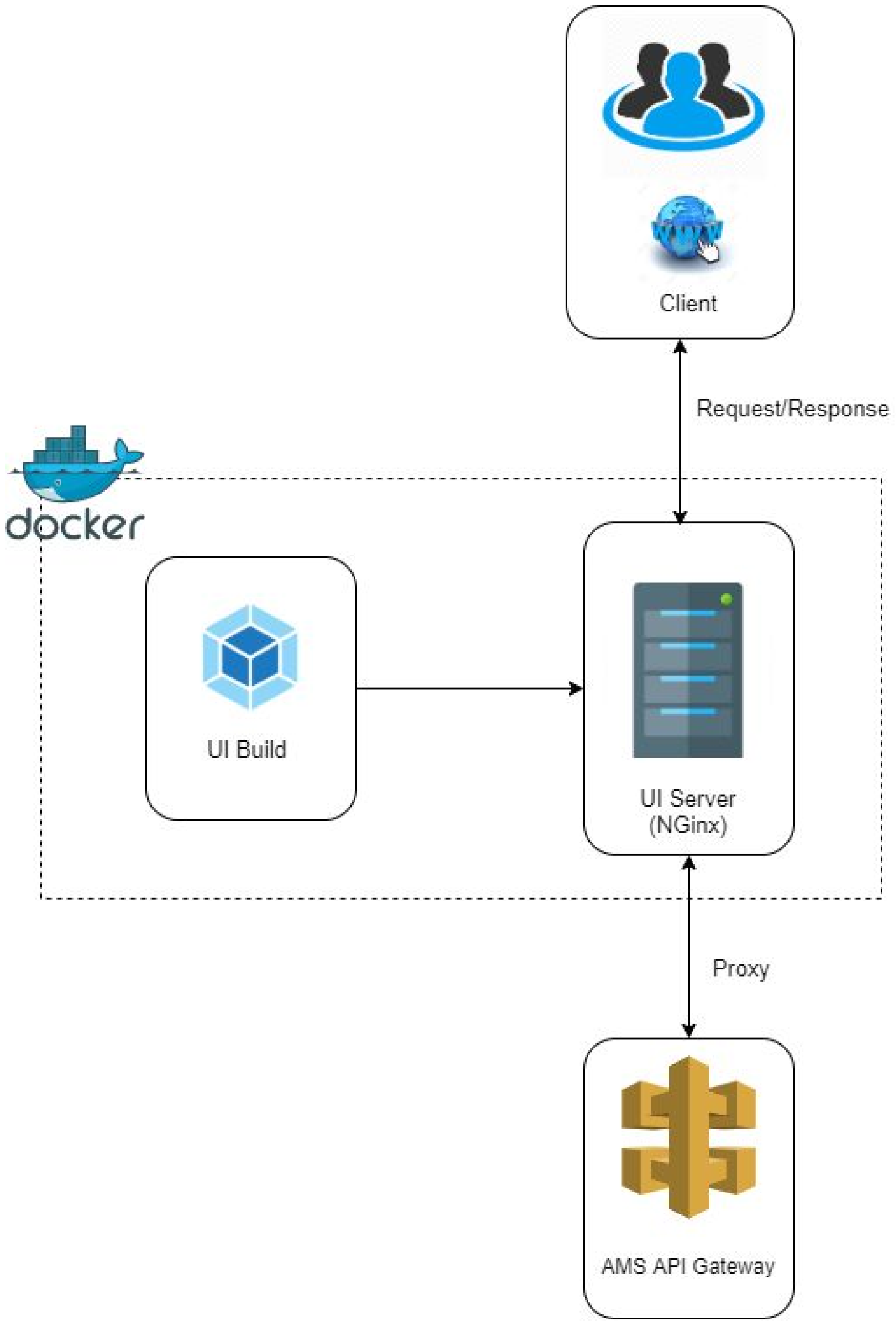


Fig. 4.1.1 System UI architecture

4.2 SYSTEM CORE ARCHITECTURE

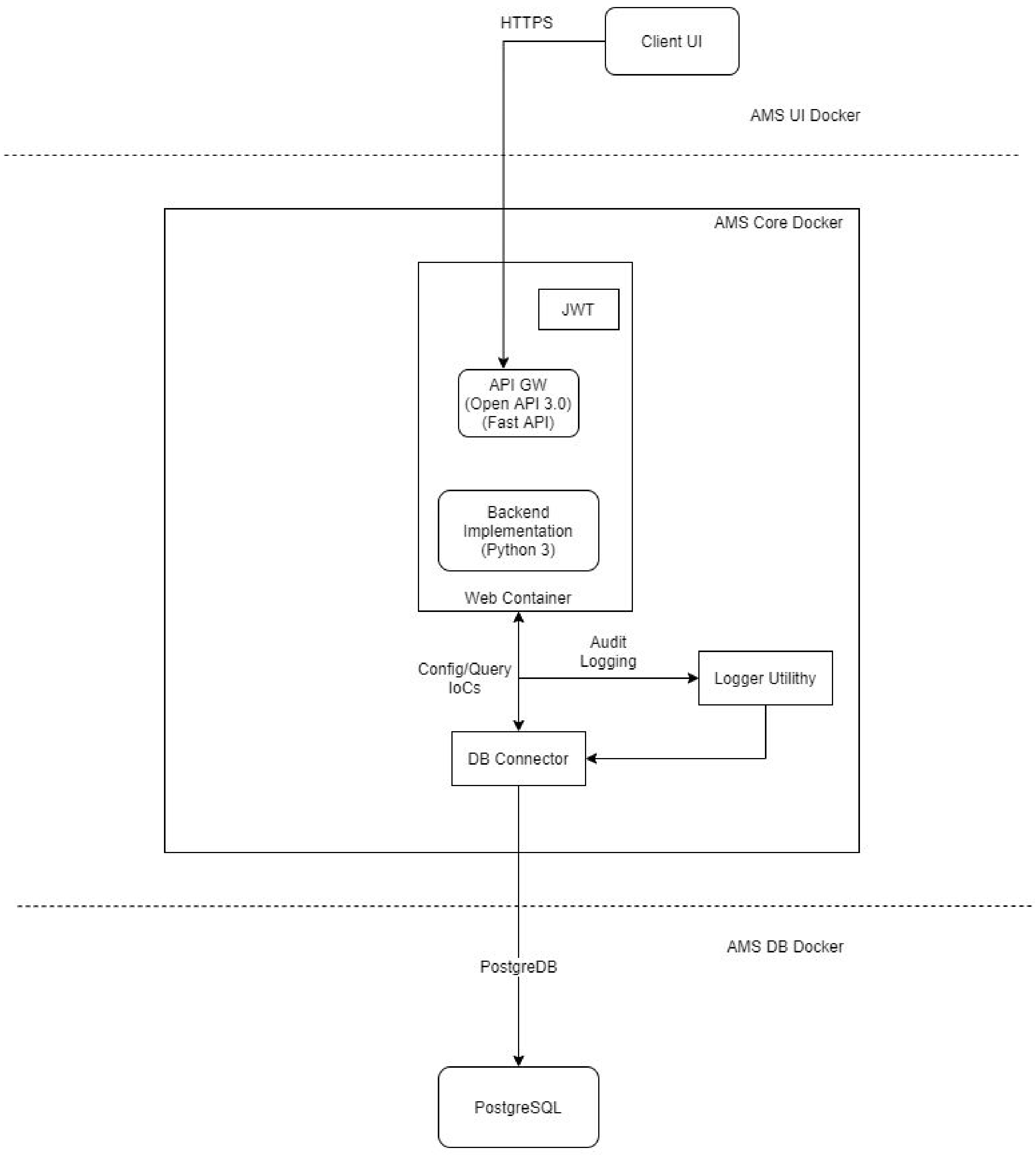


Fig. 4.2.1 System Core Architecture

Application components

This section lists the components required for the UI module.

4.2.1. Web Server

NGINX will be used as a webserver to serve HTML/CSS and JS files. NGINX became famous as the fastest web server, the scalable underlying architecture has proved ideal for many web tasks beyond serving content.

It can handle a high volume of connections, NGINX can be also used as a reverse proxy and load balancer to manage incoming traffic and distribute it to slower upstream servers.

4.2.2. Angular

Angular is a JavaScript library for building user interfaces. Angular can be used as a base in the development of a single-page or mobile application. Angular also provides Webpack and Babel tools. Webpack will be used to create a build of Angular application. It will create a build that can be served with the Web server. It is a module bundler primarily for JavaScript, it can transform front-end assets like HTML, CSS, and images if the corresponding loaders are included. Webpack takes modules with dependencies and generates static assets representing those modules. Babel is used to converting ECMAScript 2015+ code into a backward-compatible version of JavaScript that can be run by older browsers(JavaScript engines).

- **Angular Material [7]**

Angular Material is a UI component library for Angular developers. Angular Material components help in constructing attractive, consistent, and functional web pages and web applications while adhering to modern web design principles like browser portability, device independence, and graceful degradation. It helps in creating faster, beautiful, and responsive websites.

4.2.3. API Gateway

The solution will contain an API gateway that will be used to interact with the core services of the AMS. All the API calls are authenticated using the JWT authentication scheme.

The API gateway will use the FastAPI framework along with uvicorn to serve the requests. Rate limiting of the API will be a setup-time parameter. List of Endpoints that will be exposed by API gateway. The API documentation is auto-generated.

Context	Method	Functions
/api/auth	POST	Authentication endpoint to validate the user credentials and store an authentication token.
/api/status-list	GET	Provides list of all Status.
/api/status	GET	Provides range of status based on parameter.
/api/status	POST	Create new status
/api/status/{id}	GET	Get status using id
/api/status/{id}	PUT	Update status using id
/api/status/{id}	DELETE	Delete status using id
/api/room-list	GET	Get list of all room.
/api/room	GET	Get range of room based on parameter.
/api/room	POST	Create new Room.
/api/room/{id}	GET	Get room using id.
/api/room/{id}	PUT	Update room using id.

/api/room/{id}	DELETE	Delete room using id.
/api/supplier-list	GET	Get list of all suppliers.
/api/supplier	GET	Get range of suppliers using parameter.
/api/supplier	POST	Create new supplier.
/api/supplier/{id}	GET	Get supplier using id.
/api/supplier/{id}	PUT	Update supplier using id.
/api/supplier/{id}	DELETE	Delete supplier using id.
/api/activity-log	GET	Get all Activity log.
/api/permission-sync	POST	Sync Employee with HRMS.
/api/permission-list	GET	Get list of all permission.
/api/permission	GET	Get range of Permission using parameter.
/api/permission/{id}	GET	Get permission using id.
/api/permission/{id}	PUT	Update permission using id.
/api/permission/{id}	DELETE	Delete Permission using id.

Table 4.2.3. API Gateway

4.3 DATA MODELS

4.3.1. Asset :

Table Name: asset_master

Name	Type	Unique	Required	Description	Default
id	number	True	True	asset id	Auto Increment
asset_tag	varchar	True	True	asset tag of the company	
company_name	varchar		True	the asset's manufacturer company name.	
model	number		True	asset's model name	
description	varchar			asset's description	
warranty	varchar			asset warranty details	0 (Zero)
serial_number	varchar	True	True	asset's serial number	
purchase_date	datetime		True	asset's purchase date	
purchase_cost	number		True	asset's purchase cost	
category_id	number		True	asset's category	
status_id	number		True	asset's status	
employee_id	number			assigned employee id	
room_id	number			assigned room id	

Table 4.3.1. Asset

4.3.2. Status :

Table Name: status_master

Name	Type	Unique	Required	Description	Default
id	number	True	True	category id	Auto Increment
name	varchar	True	True	category name	
color_code	varchar			category's description	
show_on_dashboard	boolean		False	If it's true then the count will be shown on the dashboard	False
is_default_for_status	boolean		False	If it's true then the status value will be default selected while creating assets	False
description	varchar			status's description	

Table 4.3.2. Status

4.3.3. Rooms :

Table Name: room_master

Name	Type	Unique	Required	Description	Default
id	number	True	True	room id	Auto Increment
asset_id	number	True		Asset in room	
name	varchar		True	room name	
description	varchar			room's description	
location_id	varchar			room location name	

Table 4.3.3. Room

4.3.4. Activity Log :

Table Name: activity_log_master

Name	Type	Unique	Required	Description	Default
id	number	True	True	Activity log id	Auto Increment
activity_type	varchar		True	type of activity like created, updated, assigned, unassigned,	
old_value	text		True	The old value of activity type.	
new_value	text		True	The new value of activity type.	
field_type	varchar		True	type of file like the asset, room, asset_type, asset_category, supplier, status, permission	
field_id	varchar		True	Value of field name	
created_by	varchar		True	value of employee_id	

Table 4.3.4. Activity-log

4.3.5. Laptop [11]:

Table Name: laptop_master

Name	Type	Unique	Required	Description	Default
id	number	True	True	Laptop id	Auto Increment
asset_id	varchar			asset id	
wifi_mac_address	varchar			WiFi MAC description	
lan_mac_address	varchar			LAN MAC description	
OS	varchar			os name	
processor	number			name of processor	
RAM	varchar			RAM type	
disk_type	varchar			disk_type like SSD, HDD	
disk_capacity	varchar			disk_capacity description	
is_cisco_product	boolean				false
is_cloudops_product	boolean				false
splunk_id	varchar	true			

Table 4.3.5. Laptop

4.3.6. Permission

Table Name: asset_permission_master

Name	Type	Unique	Required	Description	Default
id	number	True	True	permission id	Auto Increment
module	varchar		True	module description	
user_id	varchar		True	user-id details	
value	boolean		True	permission value	

Table 4.3.6. Permission

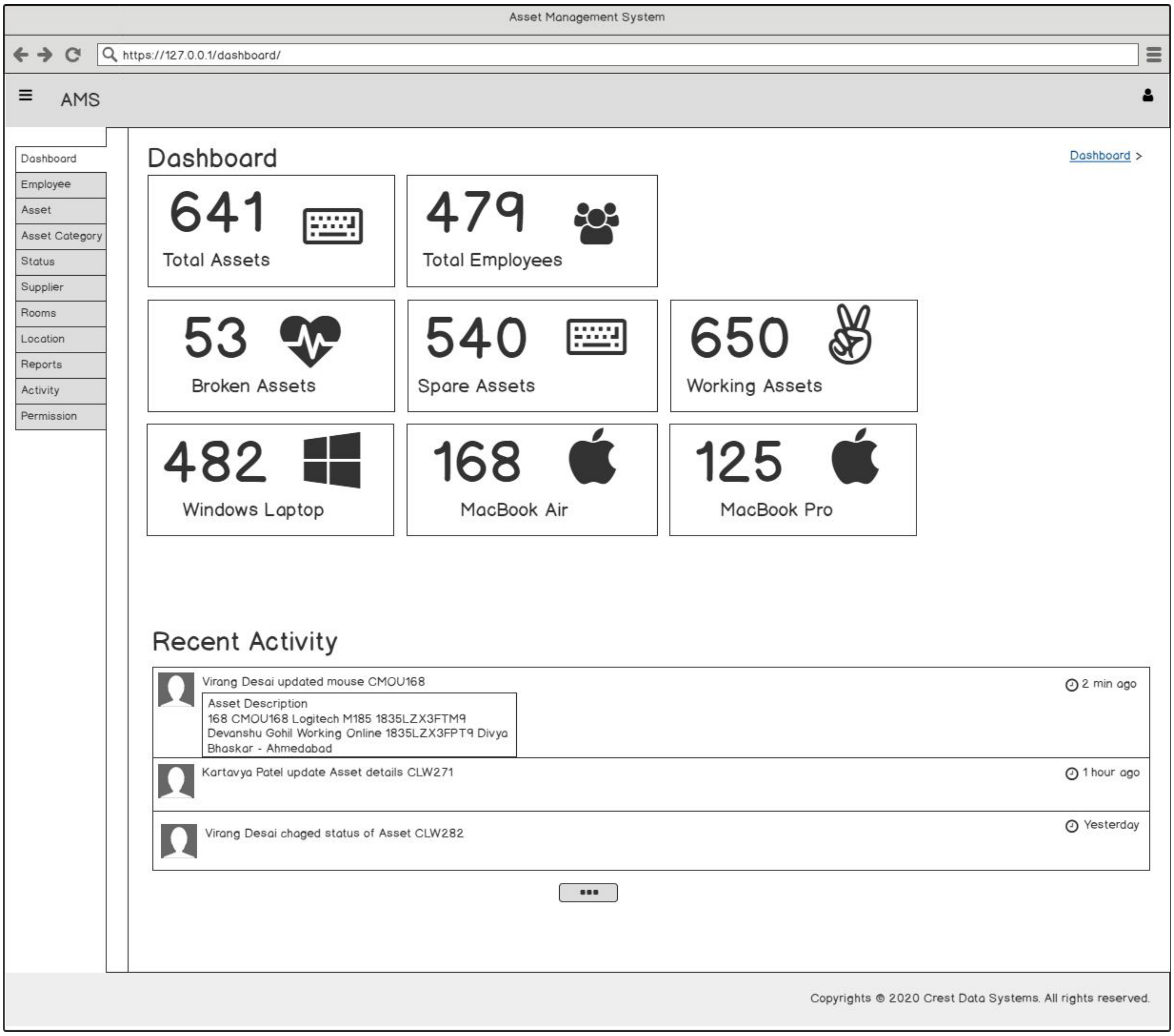
4.3.7. Supplier: Table Name: supplier_master

Name	Type	Unique	Required	Description	Default
id	number	True	True	Supplier id	
name	varchar		True	Supplier name	
description	varchar			Supplier description	
email	varchar		True	Supplier email	
mobile_no	text		True	Supplier mobile number. it can be multiple.	

Table 4.3.7. Supplier

4.4 WIREFRAMES [10]

4.4.1. DASHBOARD



Dashboard

641 Total Assets

479 Total Employees

53 Broken Assets

540 Spare Assets

650 Working Assets

482 Windows Laptop

168 MacBook Air

125 MacBook Pro

Recent Activity

Virang Desai updated mouse CMOU168 — 2 min ago
Asset Description
168 CMOU168 Logitech M185 1835LZX3FTM9 Devanshu Gohil Working Online 1835LZX3FPT9 Divya Bhaskar - Ahmedabad

Kartavya Patel update Asset details CLW271 — 1 hour ago

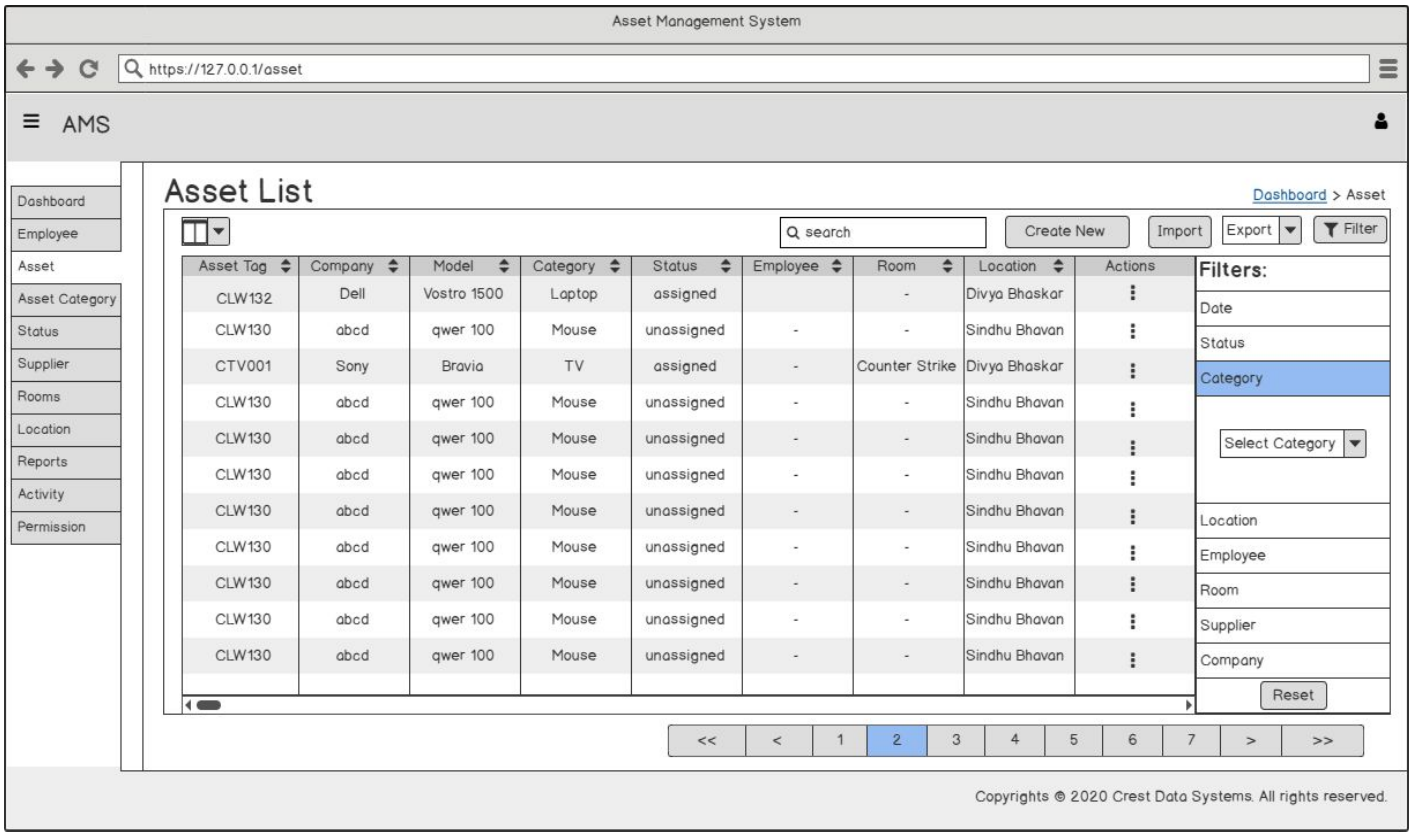
Virang Desai chaged status of Asset CLW282 — Yesterday

Copyrights © 2020 Crest Data Systems. All rights reserved.

Fig 4.4.1. Dashboard

Application Dashboard shows the various counts and Recent Activity log on screen.

4.4.2. ASSET LIST WITH FILTER



Asset Tag	Company	Model	Category	Status	Employee	Room	Location	Actions
CLW132	Dell	Vostro 1500	Laptop	assigned		-	Divya Bhaskar	⋮
CLW130	abcd	qwer 100	Mouse	unassigned	-	-	Sindhu Bhavan	⋮
CTV001	Sony	Bravia	TV	assigned	-	Counter Strike	Divya Bhaskar	⋮
CLW130	abcd	qwer 100	Mouse	unassigned	-	-	Sindhu Bhavan	⋮
CLW130	abcd	qwer 100	Mouse	unassigned	-	-	Sindhu Bhavan	⋮
CLW130	abcd	qwer 100	Mouse	unassigned	-	-	Sindhu Bhavan	⋮
CLW130	abcd	qwer 100	Mouse	unassigned	-	-	Sindhu Bhavan	⋮
CLW130	abcd	qwer 100	Mouse	unassigned	-	-	Sindhu Bhavan	⋮
CLW130	abcd	qwer 100	Mouse	unassigned	-	-	Sindhu Bhavan	⋮
CLW130	abcd	qwer 100	Mouse	unassigned	-	-	Sindhu Bhavan	⋮
CLW130	abcd	qwer 100	Mouse	unassigned	-	-	Sindhu Bhavan	⋮

Copyrights © 2020 Crest Data Systems. All rights reserved.

Fig 4.4.2. Asset-list-with-filter

Shows various filters applied to assets like asset status, category, location, Employees, or Rooms and also based on supplier or asset company details.

4.4.3. ADD ASSET

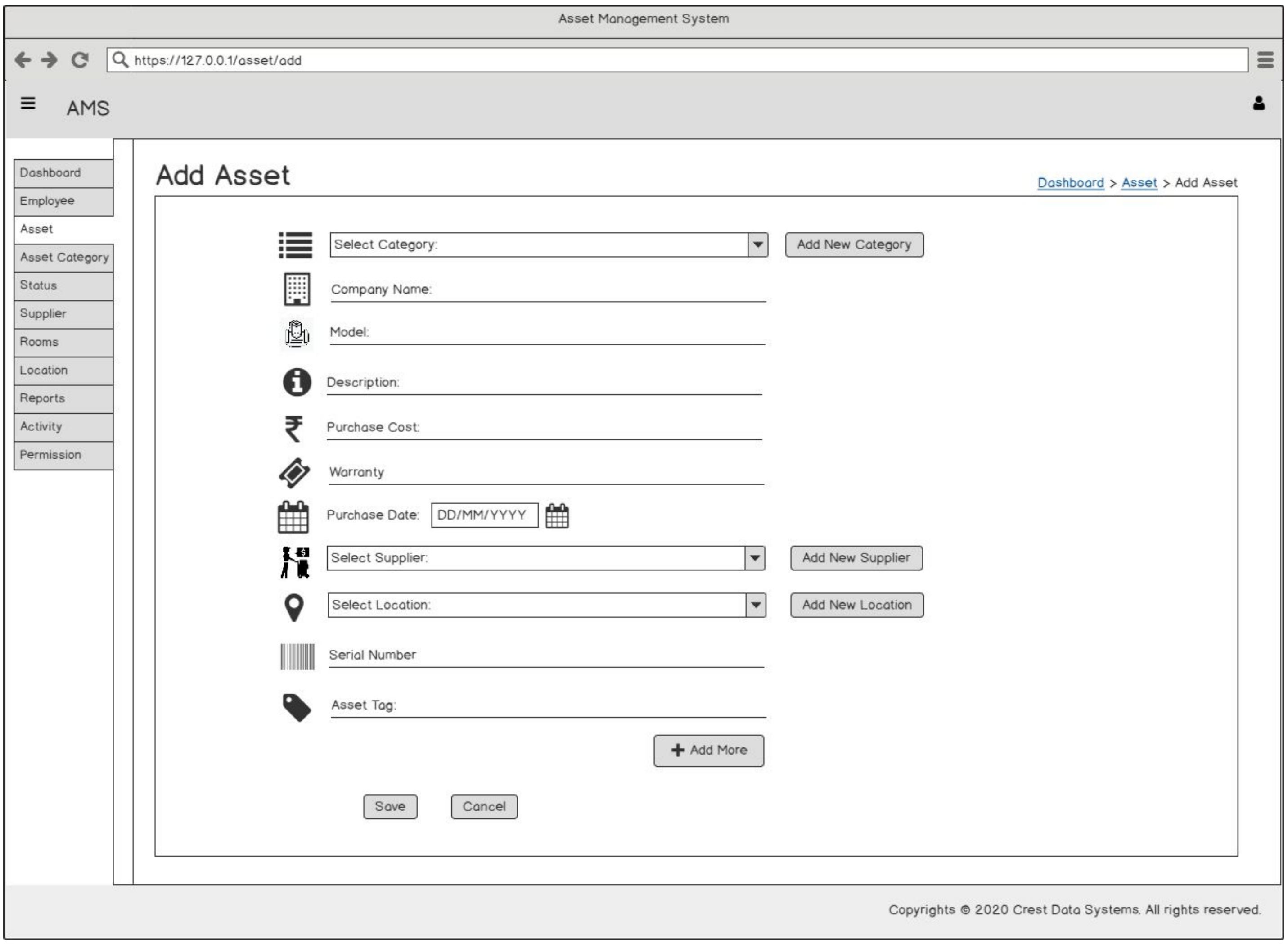


Fig 4.4.3. Add-asset

UI for Add a single Asset or multiple assets into the system. By clicking on 'Add More' Button users can add multiple assets and only specify a Serial number of assets also Asset Tag would dynamically suggest for the next count. Configuration and other details would be the same for these particular assets.

4.4.4. ADD LAPTOP

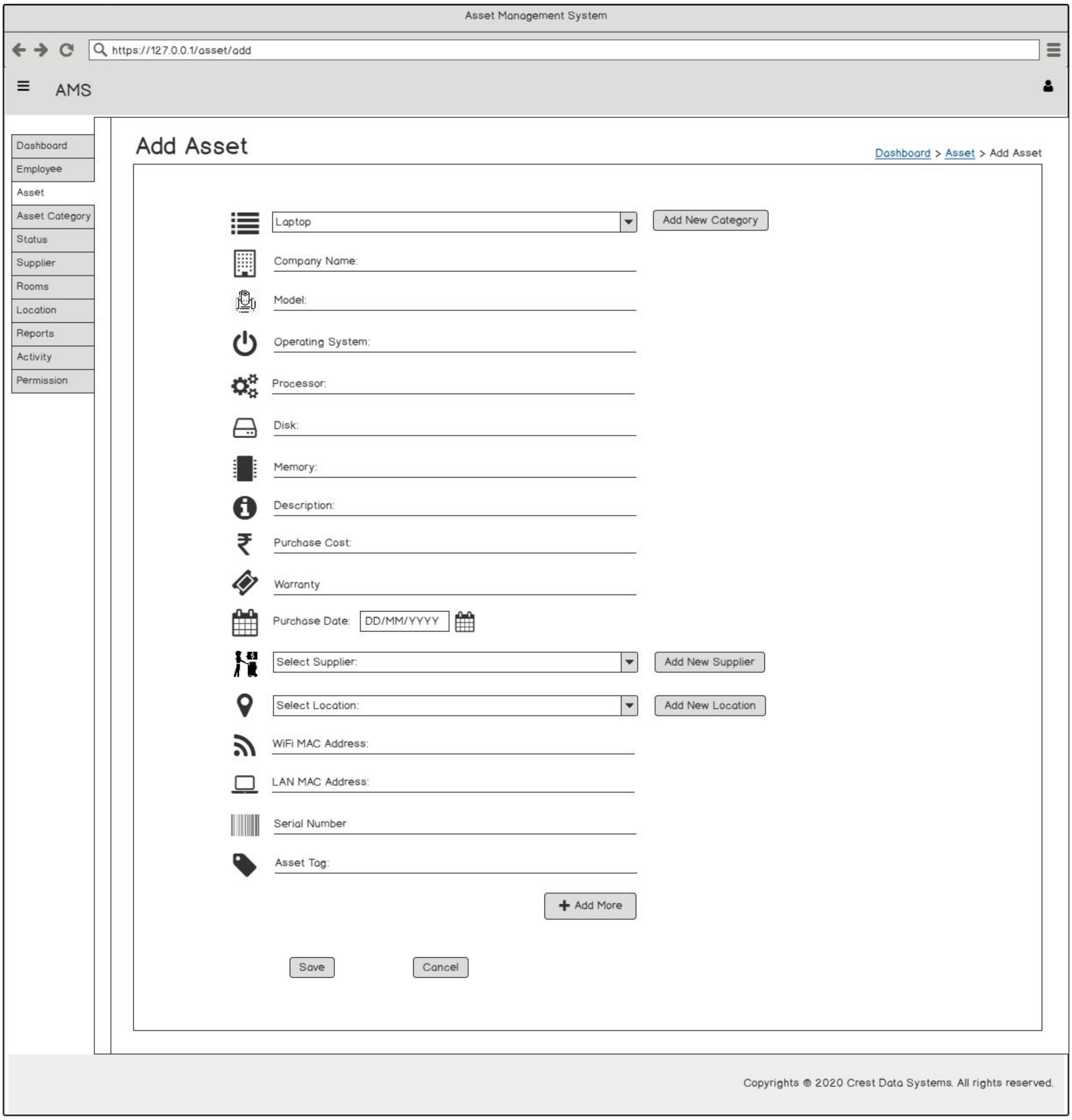
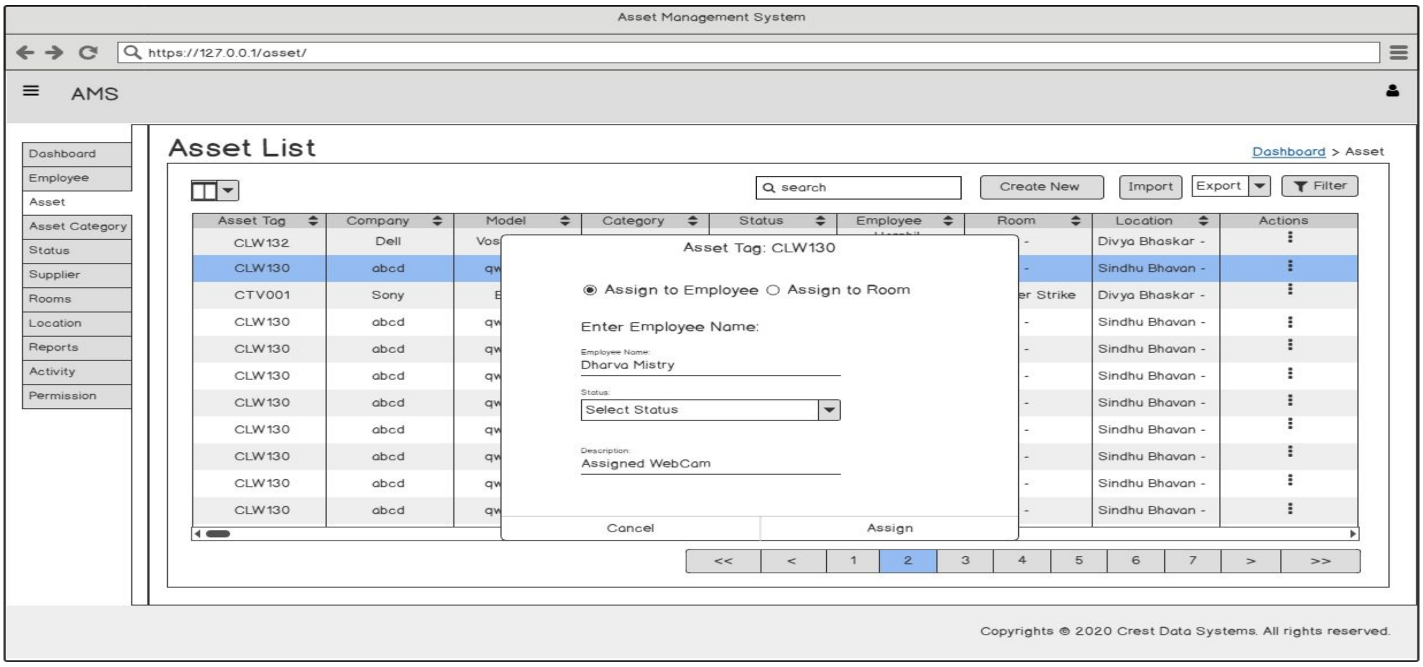


Fig 4.4.4. Add-Laptop

This UI shows when the Asset category would be selected as Laptop and dynamically add fields for adding a laptop in the system.

4.4.5. ASSIGN ASSET TO EMPLOYEE


Asset Management System

https://127.0.0.1/asset/

AMS

Dashboard | Employee | Asset | Asset Category | Status | Supplier | Rooms | Location | Reports | Activity | Permission

Asset List

Dashboard > Asset

search | Create New | Import | Export | Filter

Asset Tag	Company	Model	Category	Status	Employee	Room	Location	Actions
CLW132	Dell	Vos				-	Divya Bhaskar -	⋮
CLW130	abcd	qw				-	Sindhu Bhavan -	⋮
CTV001	Sony	E				er Strike	Divya Bhaskar -	⋮
CLW130	abcd	qw				-	Sindhu Bhavan -	⋮
CLW130	abcd	qw				-	Sindhu Bhavan -	⋮
CLW130	abcd	qw				-	Sindhu Bhavan -	⋮
CLW130	abcd	qw				-	Sindhu Bhavan -	⋮
CLW130	abcd	qw				-	Sindhu Bhavan -	⋮
CLW130	abcd	qw				-	Sindhu Bhavan -	⋮
CLW130	abcd	qw				-	Sindhu Bhavan -	⋮
CLW130	abcd	qw				-	Sindhu Bhavan -	⋮

Asset Tag: CLW130

[✓ Assign to Employee] Assign to Room

Enter Employee Name:

Employee Name: Dharva Mistry

Status: Select Status

Description: Assigned WebCam

Cancel | Assign

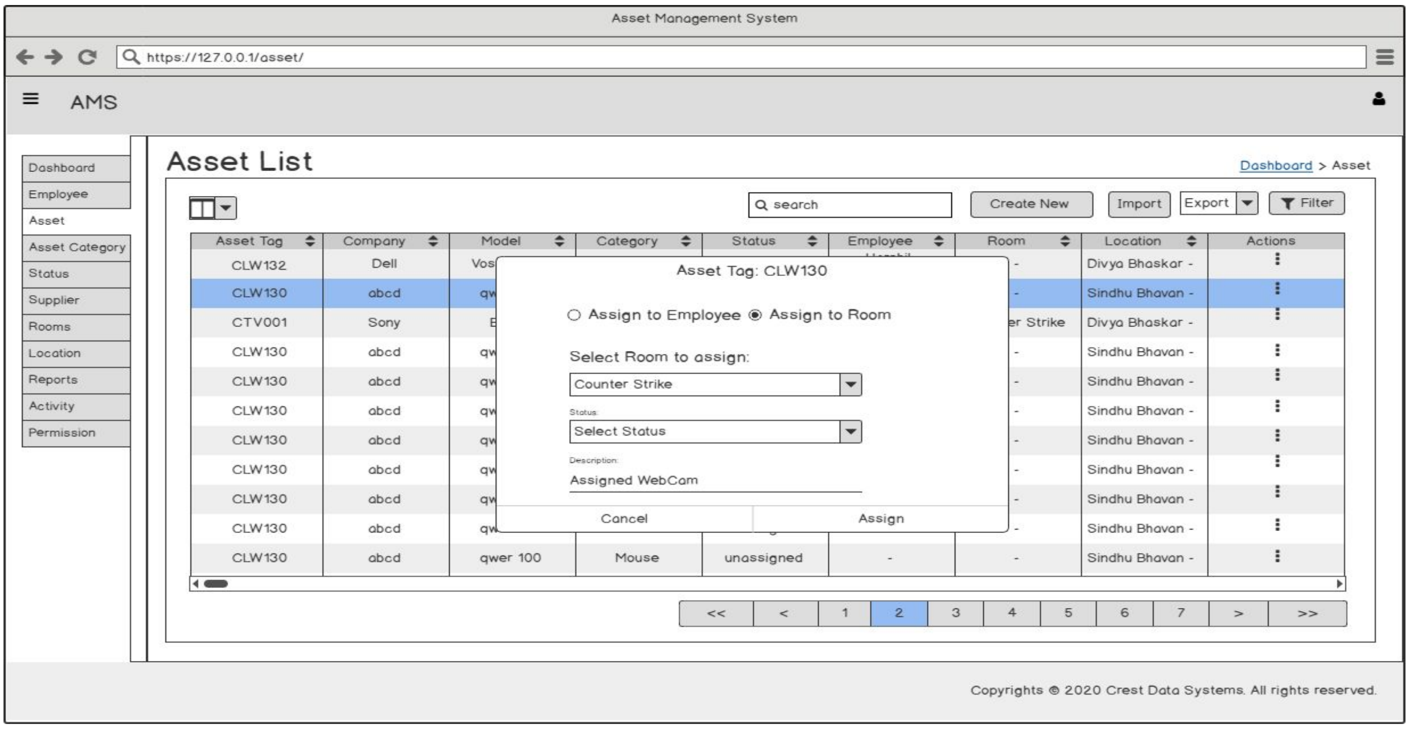
<< < 1 2 3 4 5 6 7 > >>

Copyrights © 2020 Crest Data Systems. All rights reserved.

Fig 4.4.5. Asset-assign-to-employee

Shows when the asset is assigned to Employee and changes status if required.

4.4.6. ASSIGN ASSET TO ROOM


Asset Management System

https://127.0.0.1/asset/

AMS

Dashboard | Employee | Asset | Asset Category | Status | Supplier | Rooms | Location | Reports | Activity | Permission

Asset List

Dashboard > Asset

search | Create New | Import | Export | Filter

Asset Tag	Company	Model	Category	Status	Employee	Room	Location	Actions
CLW132	Dell	Vos				-	Divya Bhaskar -	⋮
CLW130	abcd	qw				-	Sindhu Bhavan -	⋮
CTV001	Sony	E				er Strike	Divya Bhaskar -	⋮
CLW130	abcd	qw				-	Sindhu Bhavan -	⋮
CLW130	abcd	qw				-	Sindhu Bhavan -	⋮
CLW130	abcd	qw				-	Sindhu Bhavan -	⋮
CLW130	abcd	qw				-	Sindhu Bhavan -	⋮
CLW130	abcd	qw				-	Sindhu Bhavan -	⋮
CLW130	abcd	qw				-	Sindhu Bhavan -	⋮
CLW130	abcd	qw				-	Sindhu Bhavan -	⋮
CLW130	abcd	qwer 100	Mouse	unassigned	-	-	Sindhu Bhavan -	⋮

Asset Tag: CLW130

Assign to Employee [✓ Assign to Room]

Select Room to assign:

Counter Strike

Status: Select Status

Description: Assigned WebCam

Cancel | Assign

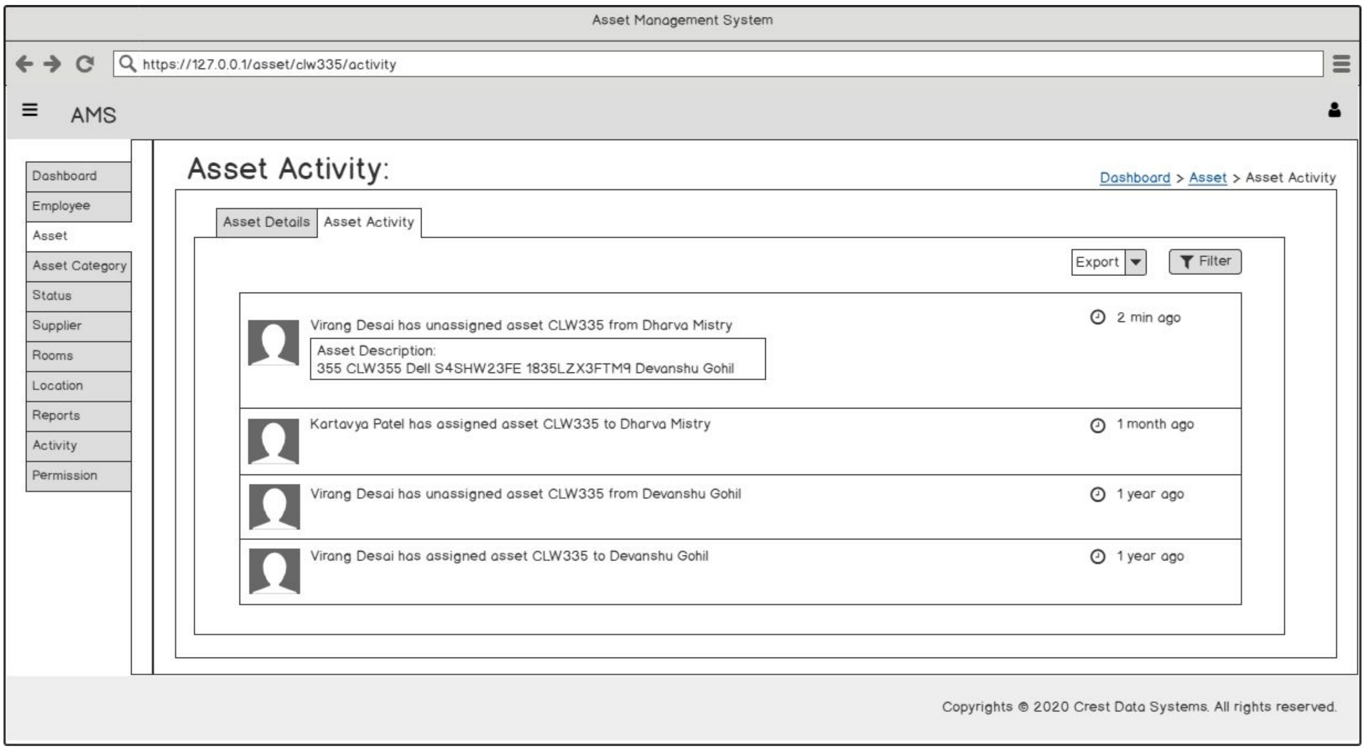
<< < 1 2 3 4 5 6 7 > >>

Copyrights © 2020 Crest Data Systems. All rights reserved.

Fig 4.4.6. Assign-asset-to-room

Shows when the asset is assigning to Room and changes status if required.

4.4.7. ASSET ACTIVITY



Virang Desai has unassigned asset CLW335 from Dharva Mistry — 2 min ago
Asset Description:
355 CLW355 Dell S4SHW23FE 1835LZX3FTM9 Devanshu Gohil

Kartavya Patel has assigned asset CLW335 to Dharva Mistry — 1 month ago

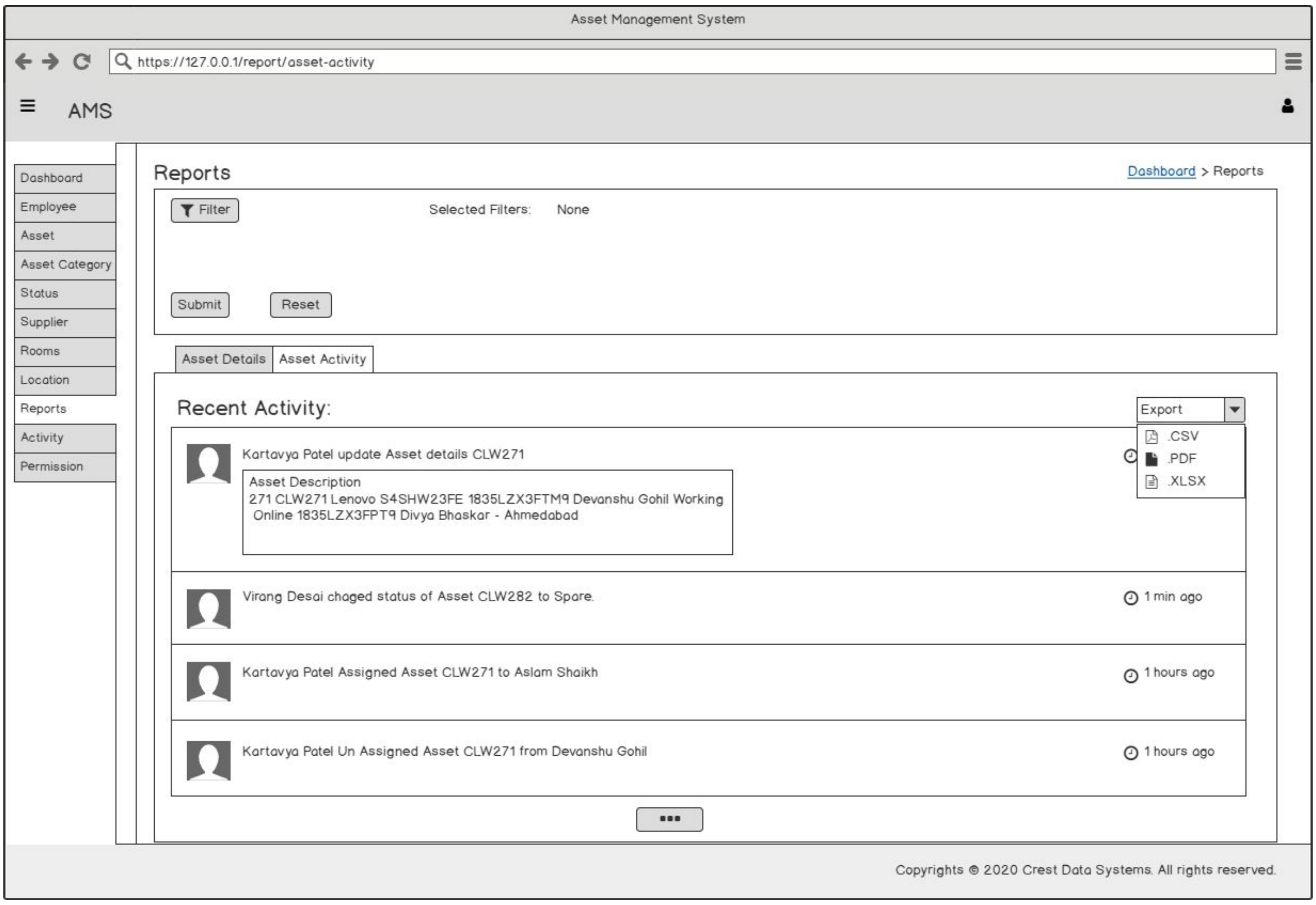
Virang Desai has unassigned asset CLW335 from Devanshu Gohil — 1 year ago

Virang Desai has assigned asset CLW335 to Devanshu Gohil — 1 year ago

Copyrights © 2020 Crest Data Systems. All rights reserved.

Fig 4.4.7. Asset-activity
Shows the Activity log that is performed on the selected Asset.

4.4.8. REPORT ASSET ACTIVITY


Asset Management System

https://127.0.0.1/report/asset-activity

AMS

Dashboard
Employee
Asset
Asset Category
Status
Supplier
Rooms
Location
Reports
Activity
Permission

Reports

Dashboard > Reports

Filter   Selected Filters: None

Submit   Reset

Asset Details | Asset Activity

Recent Activity:

Export: .CSV, .PDF, .XLSX

Kartavya Patel update Asset details CLW271

Asset Description
271 CLW271 Lenovo S4SHW23FE 1835LZX3FTM9 Devanshu Gohil Working Online 1835LZX3FPT9 Divya Bhaskar - Ahmedabad

Virang Desai chaged status of Asset CLW282 to Spare. — 1 min ago

Kartavya Patel Assigned Asset CLW271 to Aslam Shaikh — 1 hours ago

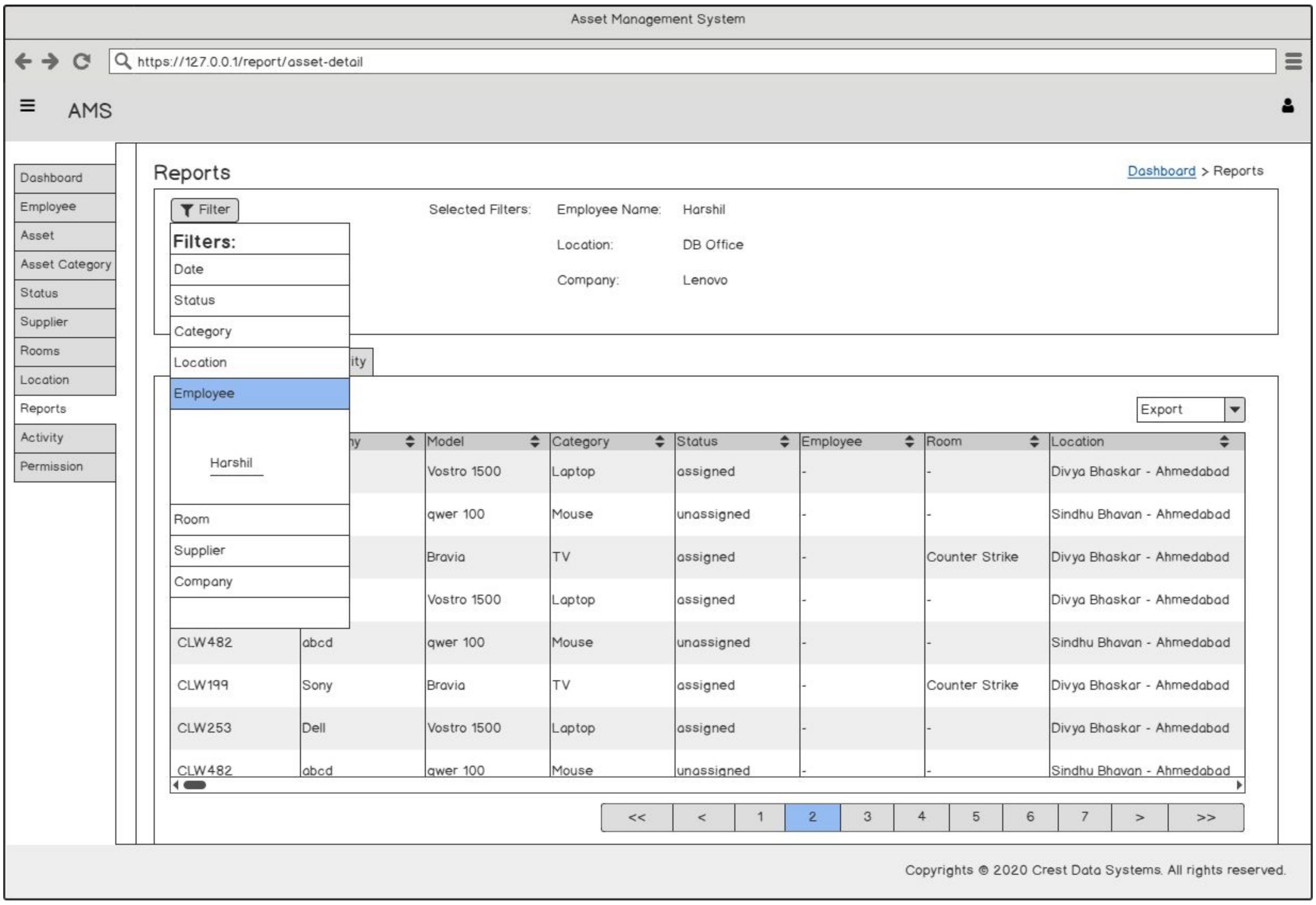
Kartavya Patel Un Assigned Asset CLW271 from Devanshu Gohil — 1 hours ago

Copyrights © 2020 Crest Data Systems. All rights reserved.

Fig 4.4.8. Report-asset-activity

Based on filtering criteria and requirements various Assets Activity would be sorted out and it would be exported based on user choices like CSV, PDF, or XLSX file format.

4.4.9. REPORT ASSET DETAILS



Selected Filters: Employee Name: Harshil
Location: DB Office
Company: Lenovo

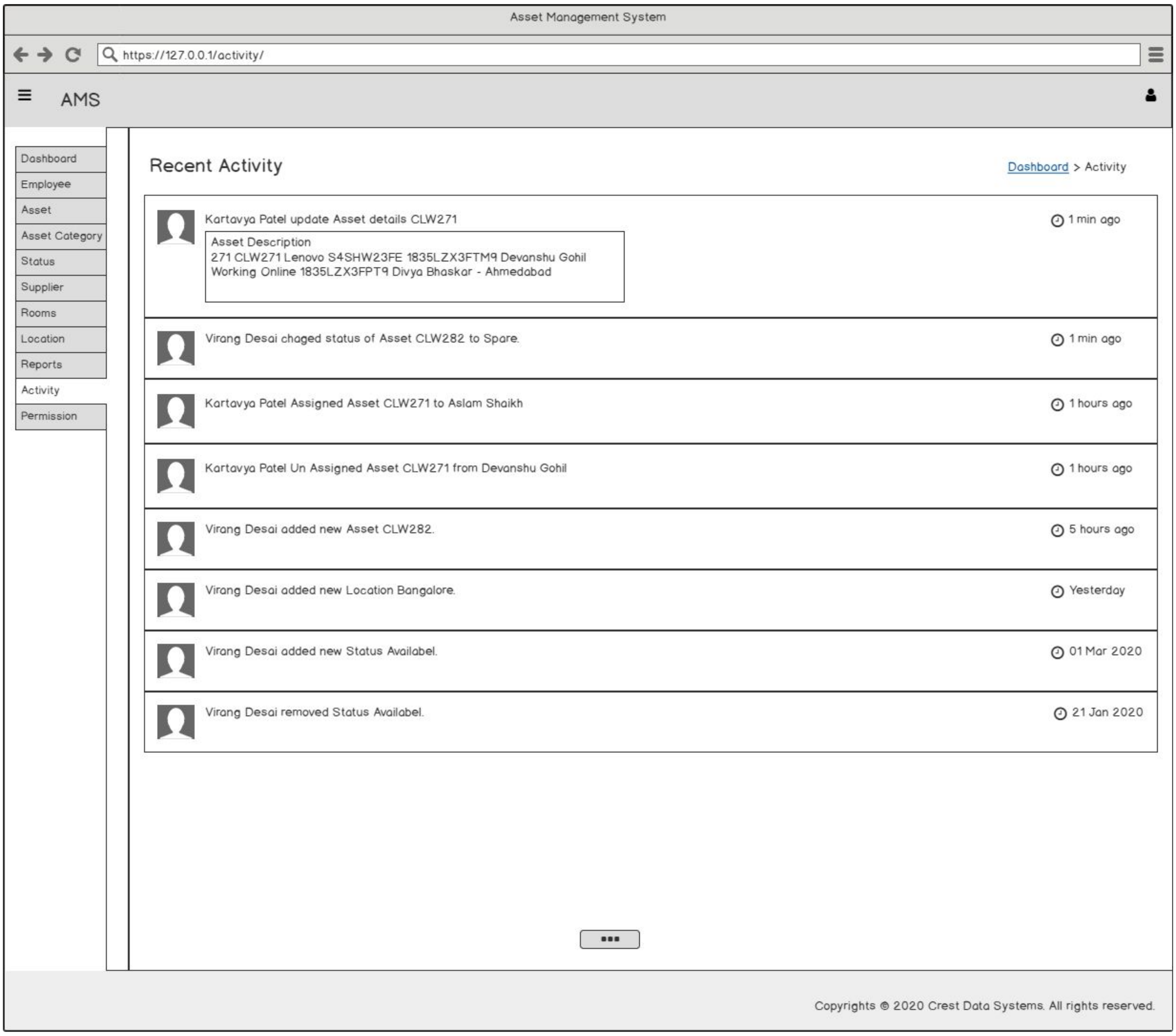
	ny	Model	Category	Status	Employee	Room	Location
		Vostro 1500	Laptop	assigned	-	-	Divya Bhaskar - Ahmedabad
		qwer 100	Mouse	unassigned	-	-	Sindhu Bhavan - Ahmedabad
		Bravia	TV	assigned	-	Counter Strike	Divya Bhaskar - Ahmedabad
		Vostro 1500	Laptop	assigned	-	-	Divya Bhaskar - Ahmedabad
CLW482	abcd	qwer 100	Mouse	unassigned	-	-	Sindhu Bhavan - Ahmedabad
CLW199	Sony	Bravia	TV	assigned	-	Counter Strike	Divya Bhaskar - Ahmedabad
CLW253	Dell	Vostro 1500	Laptop	assigned	-	-	Divya Bhaskar - Ahmedabad
CLW482	abcd	qwer 100	Mouse	unassigned	-	-	Sindhu Bhavan - Ahmedabad

Copyrights © 2020 Crest Data Systems. All rights reserved.

Fig 4.4.9. Report-asset-details

Based on filtering criteria and requirements various Assets details would be sorted out and it would be exported based on user choices like CSV, PDF, or XLSX file format.

4.4.10. RECENT ACTIVITY



Recent Activity

Kartavya Patel update Asset details CLW271 — 1 min ago
Asset Description
271 CLW271 Lenovo S4SHW23FE 1835LZX3FTM9 Devanshu Gohil
Working Online 1835LZX3FPT9 Divya Bhaskar - Ahmedabad

Virang Desai chaged status of Asset CLW282 to Spare. — 1 min ago

Kartavya Patel Assigned Asset CLW271 to Aslam Shaikh — 1 hours ago

Kartavya Patel Un Assigned Asset CLW271 from Devanshu Gohil — 1 hours ago

Virang Desai added new Asset CLW282. — 5 hours ago

Virang Desai added new Location Bangalore. — Yesterday

Virang Desai added new Status Availabel. — 01 Mar 2020

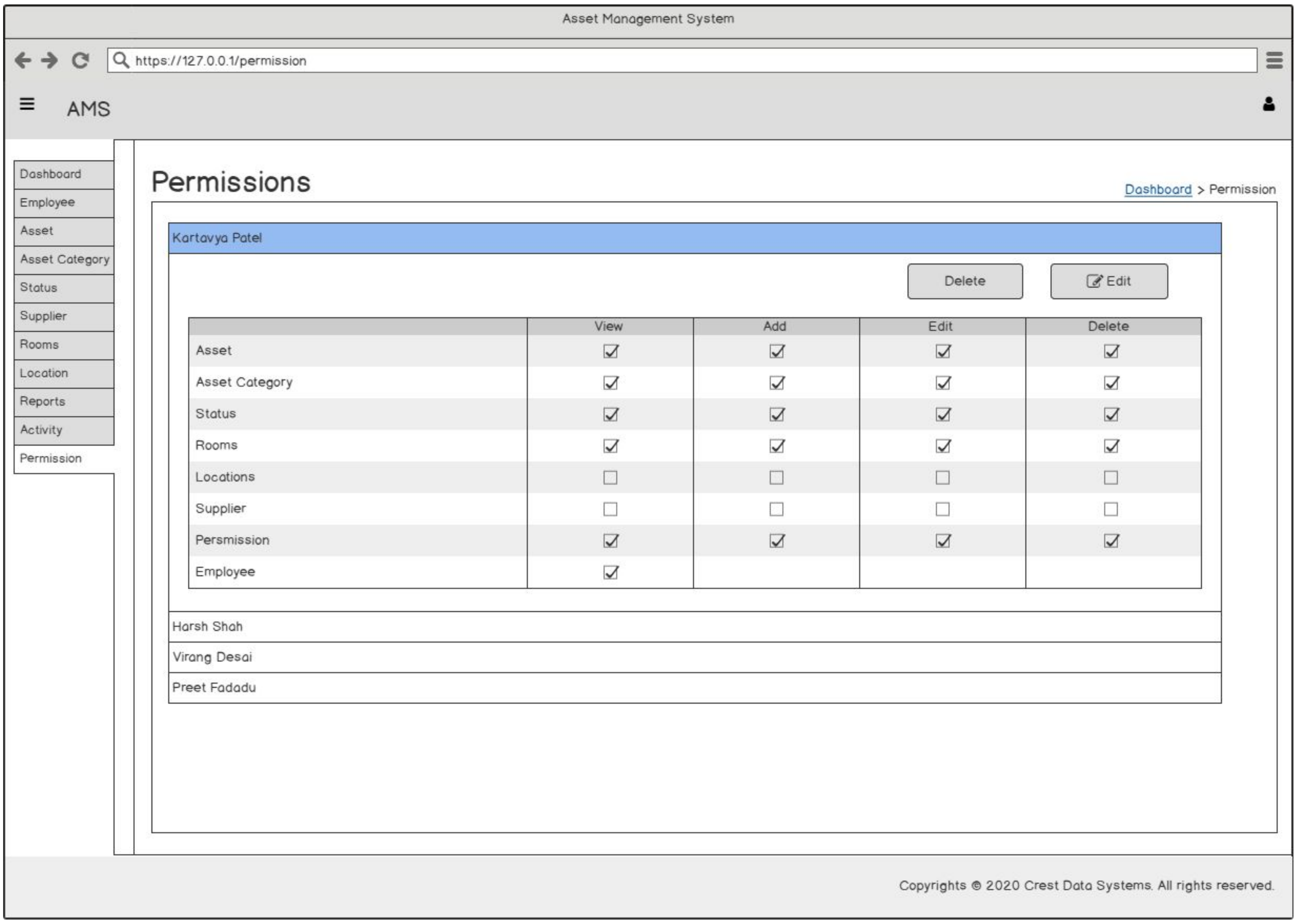
Virang Desai removed Status Availabel. — 21 Jan 2020

Copyrights © 2020 Crest Data Systems. All rights reserved.

Fig 4.4.10. Recent-activity

The screen shows the Recent Activity logs which represent tasks recently performed onto the system.

4.4.11. PERMISSION



	View	Add	Edit	Delete
Asset	✓	✓	✓	✓
Asset Category	✓	✓	✓	✓
Status	✓	✓	✓	✓
Rooms	✓	✓	✓	✓
Locations	☐	☐	☐	☐
Supplier	☐	☐	☐	☐
Persmission	✓	✓	✓	✓
Employee	✓			

Fig 4.4.11. Permission

Screen shows which permission is given to which employee. And can also change the permission of employees by admin or other authenticated person.

Chapter 5 Implementation

5.1 IMPLEMENTATION ENVIRONMENT

In this project, the implementation environment is made up of three components: FastAPI, Angular 9, and PostgreSQL. FastAPI is the library that is used to manage backend logic and provides APIs for each module like Asset, Employee, Room, Supplier, Asset Category, Status and Manage Permissions. Now, PostgreSQL is a Relational Database Management architecture used to record all occurrences of data. It helps in application to manage the relationship between data models efficiently by its Object-Oriented feature. And PostgreSQL has the Dashboard with real-time analysis, Servers connections, IO operation frequency, Transaction per Second and Efficient UI for data analysis. Angular 9 is used for UI design. Angular 9 is having a rich feature set with dynamic data and event behavior is useful for this project. Angular has a modular structure with Modules, Components, Services, and a lazy loading routing mechanism. Angular Material components help in constructing attractive, consistent, and functional web pages and web applications while adhering to modern web design principles like browser portability, device independence, and graceful degradation. It helps in creating faster, beautiful, and responsive websites. And all these components are integrated into Docker using a docker-compose file.

5.2 KEY IMPLEMENTATIONS OBJECTIVE

- Simple and Easy UI interactions should be developed.
- All API calls and functionality should be properly working.
- Consistent data should be there in the database.
- Activity Log should be recorded for specified operations.
- The report should be generated as per user requirements.

5.3 IMPLEMENTATION MODULES

Each module are Integrated into FastAPI, PostgreSQL, and Angular

- Asset: Manages Asset
- Employee: Assign, Reassign and Unassign Asset
- Room: Assign, Reassign and Unassign Asset
- Report: Generate Report based on specifications

5.4 IMPLEMENTATION STEPS FOR MODULES

1. First Install all required python [4] libraries for FastAPI and SQLAlchemy with PostgreSQL ORM.
2. Create a database connection code with PostgreSQL service by providing a connection server URL, port, and database name.
3. Develop database table-column Model by specifying the column name and data type store in that field.
4. Decide request time data schema and response time data schema.
5. Develop API that defines backend implementation includes database query.
6. Test APIs using SwaggerUI with realtime database server connection and also, Proper validation Error messages.
7. Integrate APIs into Angular UI design with proper user interaction.
8. Integrate API data into Angular Material components.
9. Develop Docker files for each component with specified configurations.
10. Combine all Docker services into a docker-compose file.
11. Upload docker images into Docker Hub and fetch that image into the real-time server then run a Docker container on the server.

Chapter 6
Testing

6.1 TEST PLANNING

Test planning is the phase to describe how Testing would be performed. It includes a plan to test before going to start making the test suite. The first step of testing is to test each system module that is once the module has been completed, we test the module. For this, white box testing and black-box testing is used. In white-box testing, structural testing is done so all the modules are tested one by one, and finally when the project is completed black box testing is used to test the whole system together.

The objective of the system testing is to ensure that all individual programs are working as expected, that the programs link together to meet the requirements specified, and ensure that the computer system and the associated clerical and other procedures work together.

6.2 TESTING STRATEGY

Following Testing Strategies are used in the Application:

6.2.1. UNIT TESTING

Unit testing focuses on the smallest unit of software design, like a module or system component. This testing strategy conducted on each module interface to access Boundary conditions are tested and all error handling paths are tested.

6.2.2. INTEGRATION TESTING

This testing strategy follows the testing of combined parts of an application to determine if they function correctly. The purpose of this level of testing is to expose faults in the interaction between integrated units. Testing performed to expose defects in the interfaces and interaction between integrated components.

6.3 TEST SUITES DESIGN

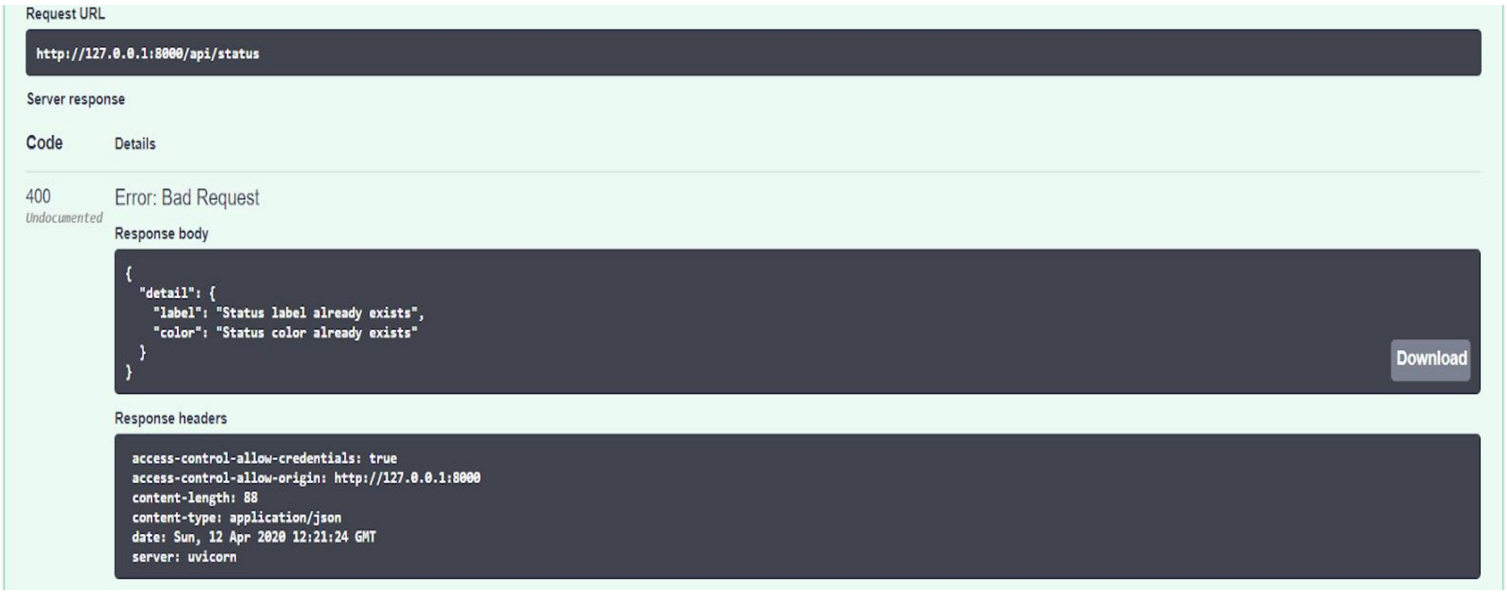
Sr. No.	Test Case	Expected Output	Actual Output	Test Case Result
1.	FastAPI connection with PostgreSQL database service	The connection success message is shown on the Server log	As Expected	Pass
2.	Add asset category with unique category name and asset tag	JSON Object with input values of the asset category	As Expected	Pass
3.	Add asset category with an existing name or asset tag	Error message to give other name or asset tag for a category as it is already exist	As Expected	Pass
4.	Add Status with a unique name and color code hex value	JSON Object with input values of the asset status	As Expected	Pass
5.	Add Status with an existing name or color code hex value	Error message to give other name or color hex value for status as it is already exist	As Expected	Pass
6.	Add company location with a unique name	JSON Object with input values of location	As Expected	Pass
7.	Add company location with an existing name	Error message to give other names for the company location	As Expected	Pass
8.	Add room with a unique name on the same location	JSON Object with input values of the room with the location object	As Expected	Pass
9.	Add room with an existing name on the same location	Error message ‘room name already exists on this location’	As Expected	Pass
10.	Add room with the same name on a different location	JSON Object with input values of the room with the location object	As Expected	Pass
11.	Add supplier with a unique email and contact number	JSON Object with input values of the Supplier	As Expected	Pass
12.	Add supplier with an	Error message to give other	As	Pass

	existing email or contact number	email or contact number as it already exists	Expected	
13.	Add Asset with auto-generated Asset tag and number as per selected asset category	JSON Object with input values of the Asset	As Expected	Pass
14.	Assign asset to an employee with unassigned asset status	Success message of assign asset to an employee	As Expected	Pass
15.	Assign asset to an employee with assigned asset status	Error message ‘Asset is already assigned to a specific employee or room’	As Expected	Pass
16.	Assign asset to room with unassigned asset status	Success message of assign asset to a room	As Expected	Pass
17.	Assign asset to room with assigned asset status	Error message ‘Asset is already assigned to a specific employee or room’	As Expected	Pass
18.	Generate Activity log for a specific task like add, edit, delete or assign, an unassign asset to room or employee	JSON object returned with asset details and as per task details	As Expected	Pass
19.	Filter asset based on category, Status, Location, Employee, Room, Supplier or Asset Company	Display asset details based on filtering criteria	As Expected	Pass
20.	Import Asset from .csv, .xls or .xlsx format file	JSON object list with asset details	As Expected	Pass
21.	Import Asset from other format files	Error message for the invalid file format	As Expected	Pass
22.	Export Assets to format like .csv, .xls, xlsx or PDF	Download file into the machine with the specified file format and requirement	As Expected	Pass

Table 6.3 Test Suites Design

6.4 API AND UI VALIDATIONS SNAPSHOTS

6.4.1. Asset Status Validation



Request URL

```
http://127.0.0.1:8000/api/status
```

Server response

Code Details

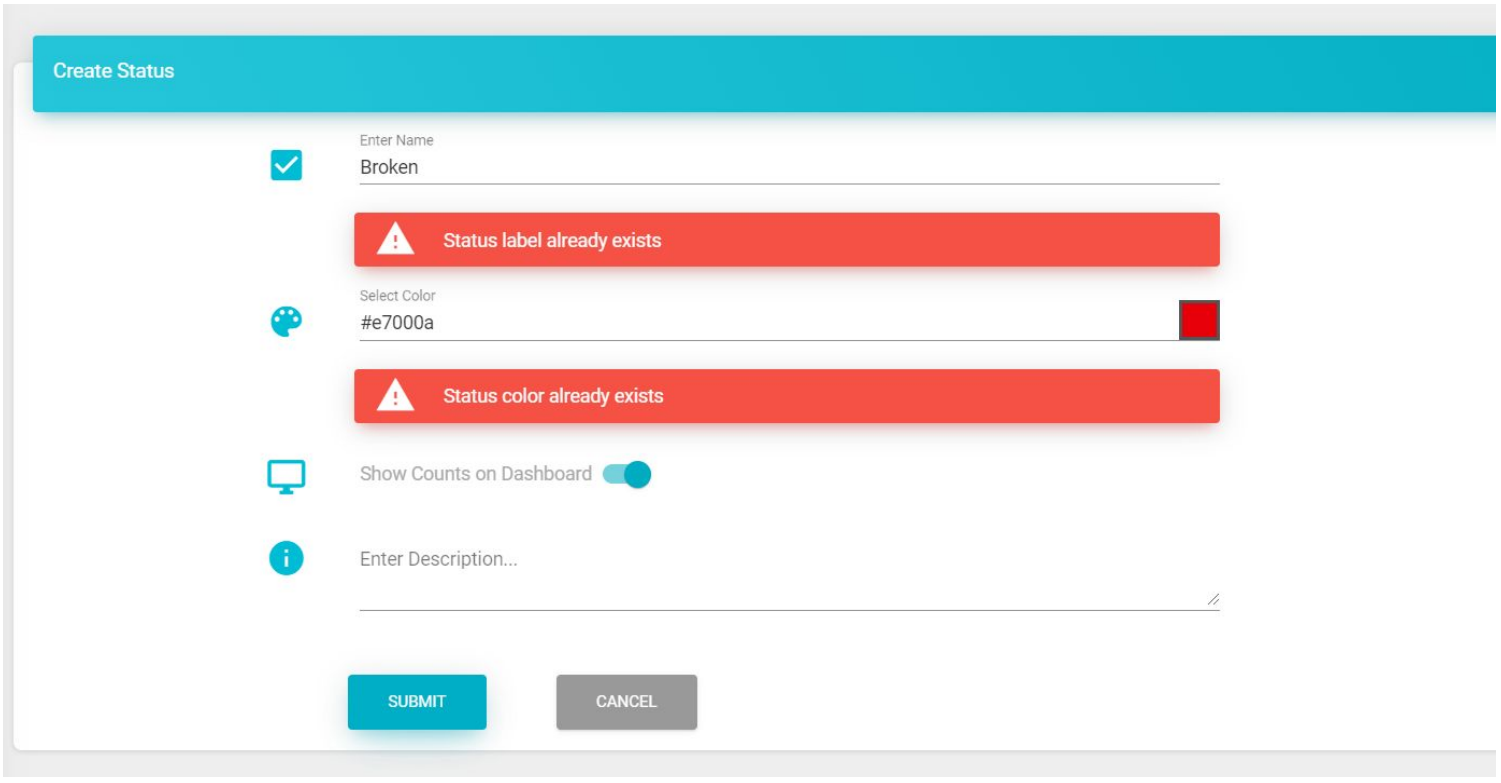
400 *Undocumented* Error: Bad Request

Response body

```
{
  "detail": {
    "label": "Status label already exists",
    "color": "Status color already exists"
  }
}
```

Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: http://127.0.0.1:8000
content-length: 88
content-type: application/json
date: Sun, 12 Apr 2020 12:21:24 GMT
server: uvicorn
```



Create Status

Enter Name
Broken

Status label already exists

Select Color
#e7000a

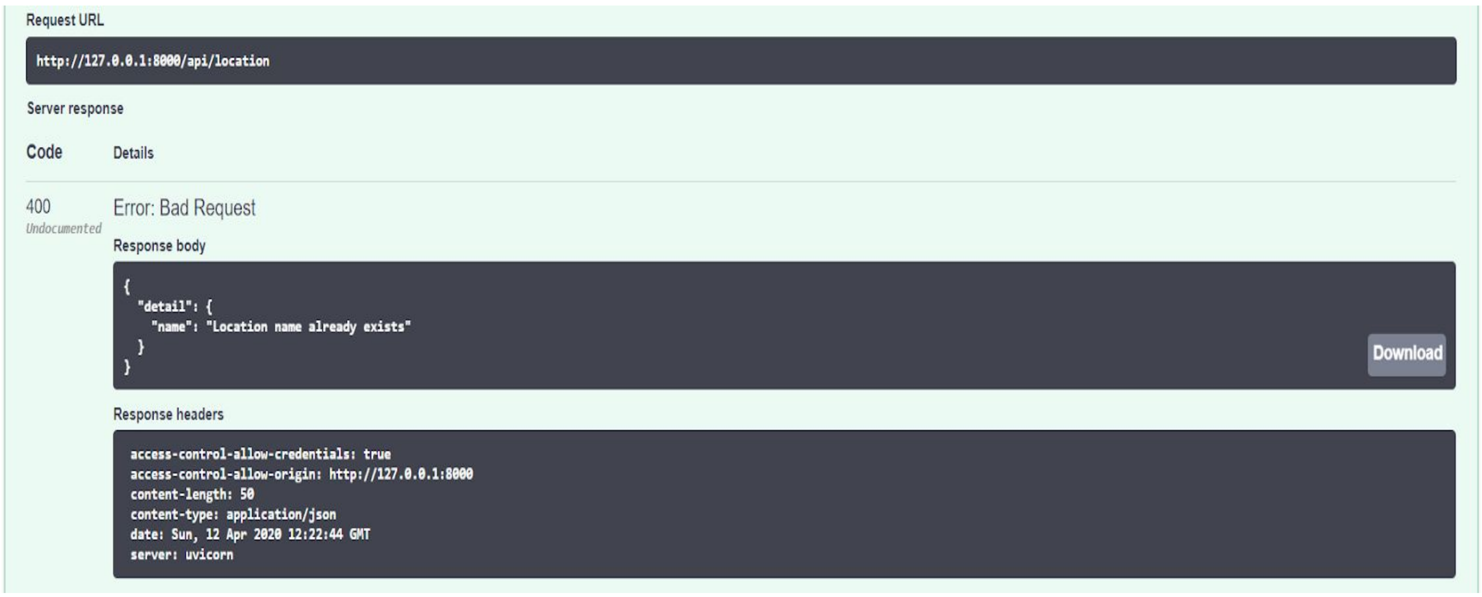
Status color already exists

Show Counts on Dashboard

Enter Description...

Fig 6.4.1. Asset-status-validation
Asset Status Label name and hex color code value should be unique.

6.4.2. Company Location Validation [11]


Request URL

```
http://127.0.0.1:8000/api/location
```

Server response

Code Details

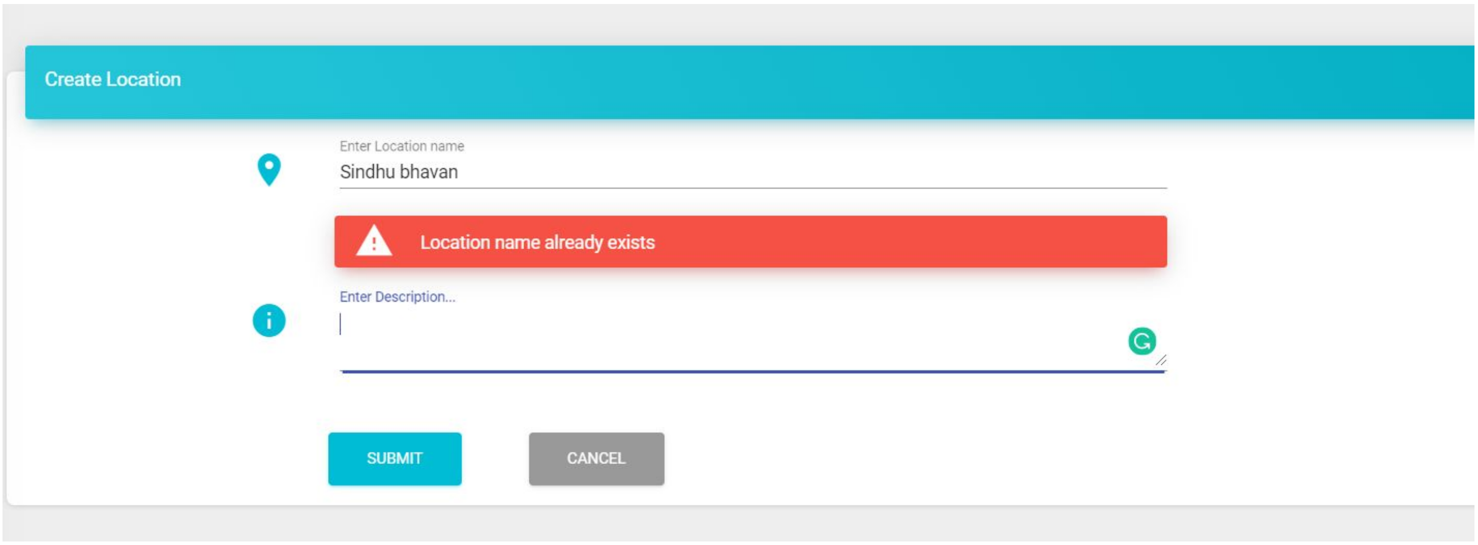
400 Undocumented — Error: Bad Request

Response body

```
{
  "detail": {
    "name": "Location name already exists"
  }
}
```

Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: http://127.0.0.1:8000
content-length: 50
content-type: application/json
date: Sun, 12 Apr 2020 12:22:44 GMT
server: uvicorn
```


Create Location

Enter Location name: Sindhu bhavan

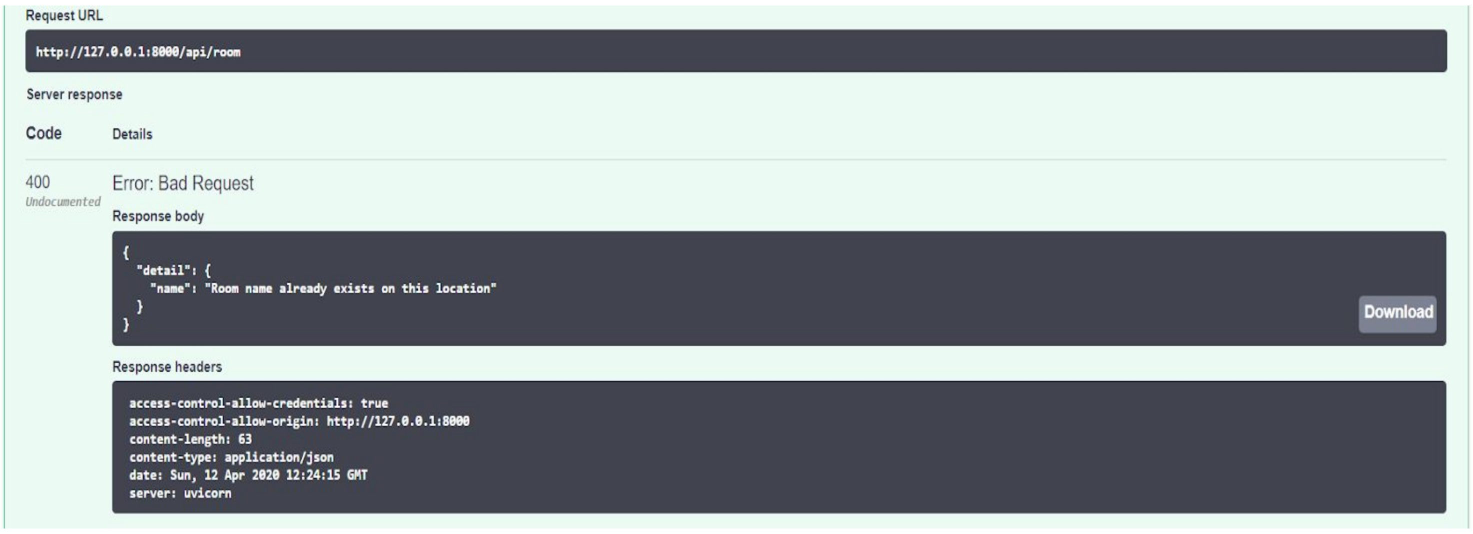
Location name already exists

Enter Description...

Fig 6.4.2. Company-location-validation [11]

The location name should be unique.

6.4.3. Room Validation


Request URL

```
http://127.0.0.1:8000/api/room
```

Server response

Code Details

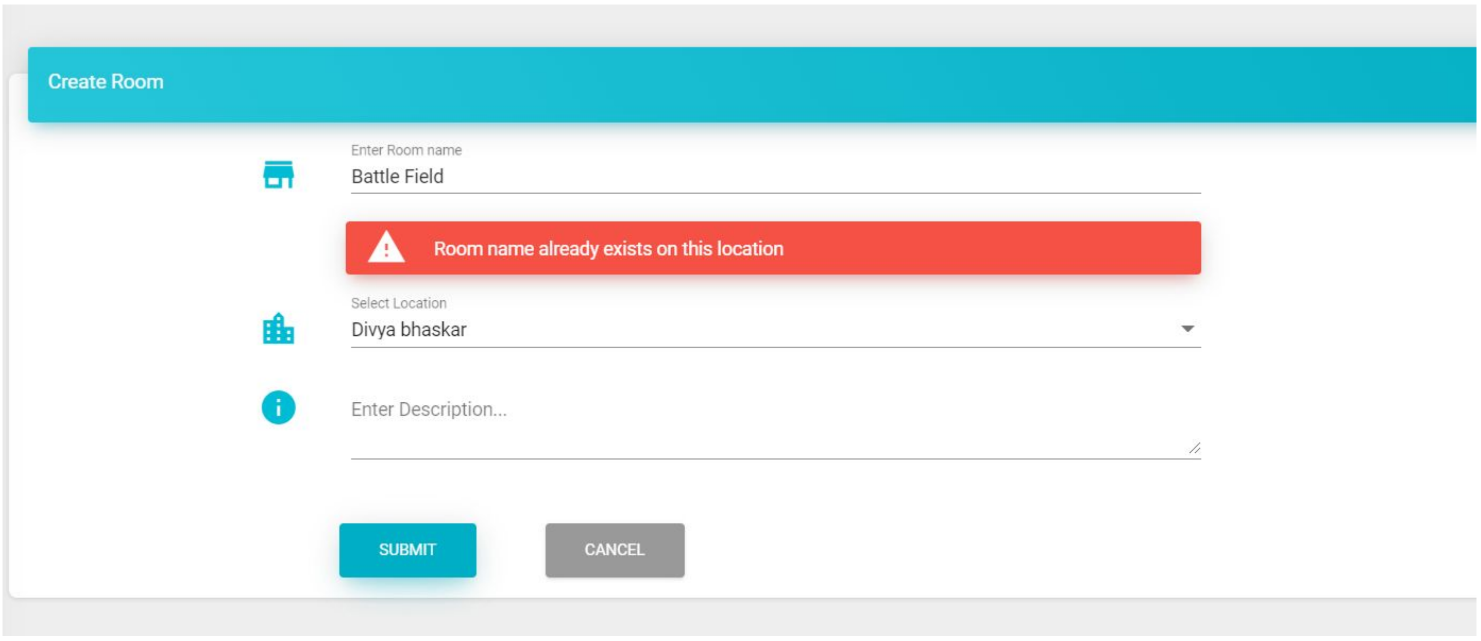
400 Undocumented Error: Bad Request

Response body

```
{
  "detail": {
    "name": "Room name already exists on this location"
  }
}
```

Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: http://127.0.0.1:8000
content-length: 63
content-type: application/json
date: Sun, 12 Apr 2020 12:24:15 GMT
server: uvicorn
```


Create Room

Enter Room name: Battle Field

Room name already exists on this location

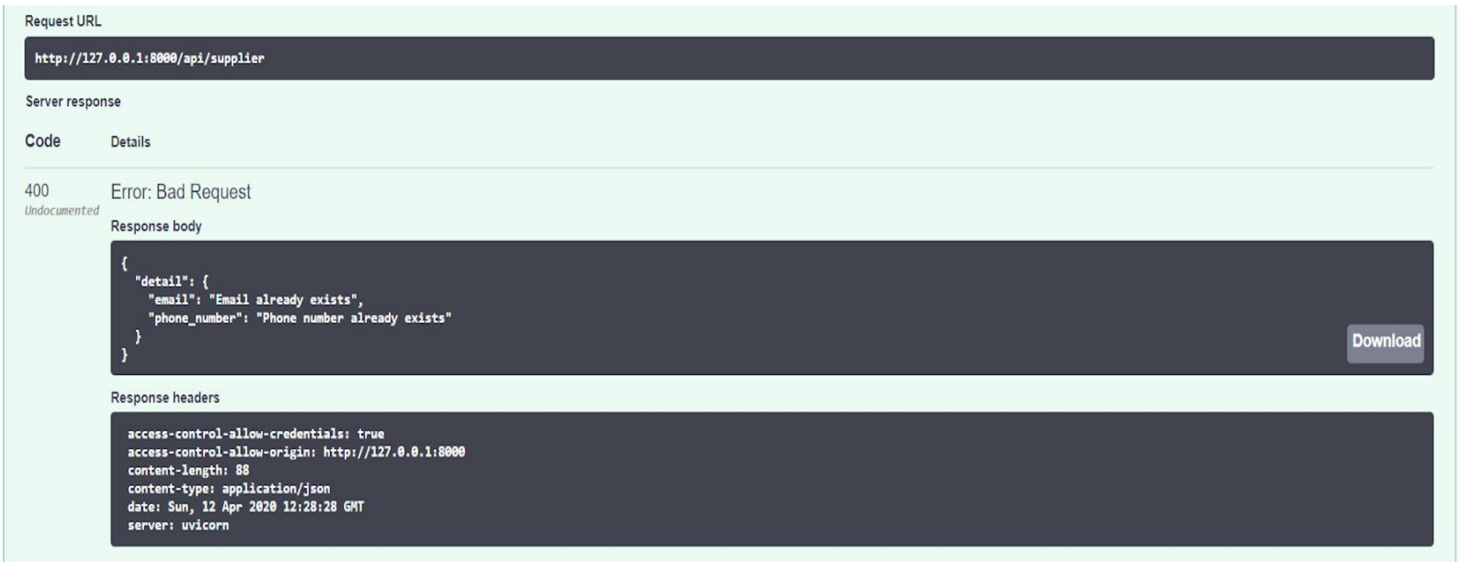
Select Location: Divya bhaskar

Enter Description...

Fig 6.4.3. Room-validation

The room name should be unique for the same location but the different locations can have the same name.

6.4.4. Supplier Validation


Request URL

```
http://127.0.0.1:8000/api/supplier
```

Server response

Code Details

400 *Undocumented* Error: Bad Request

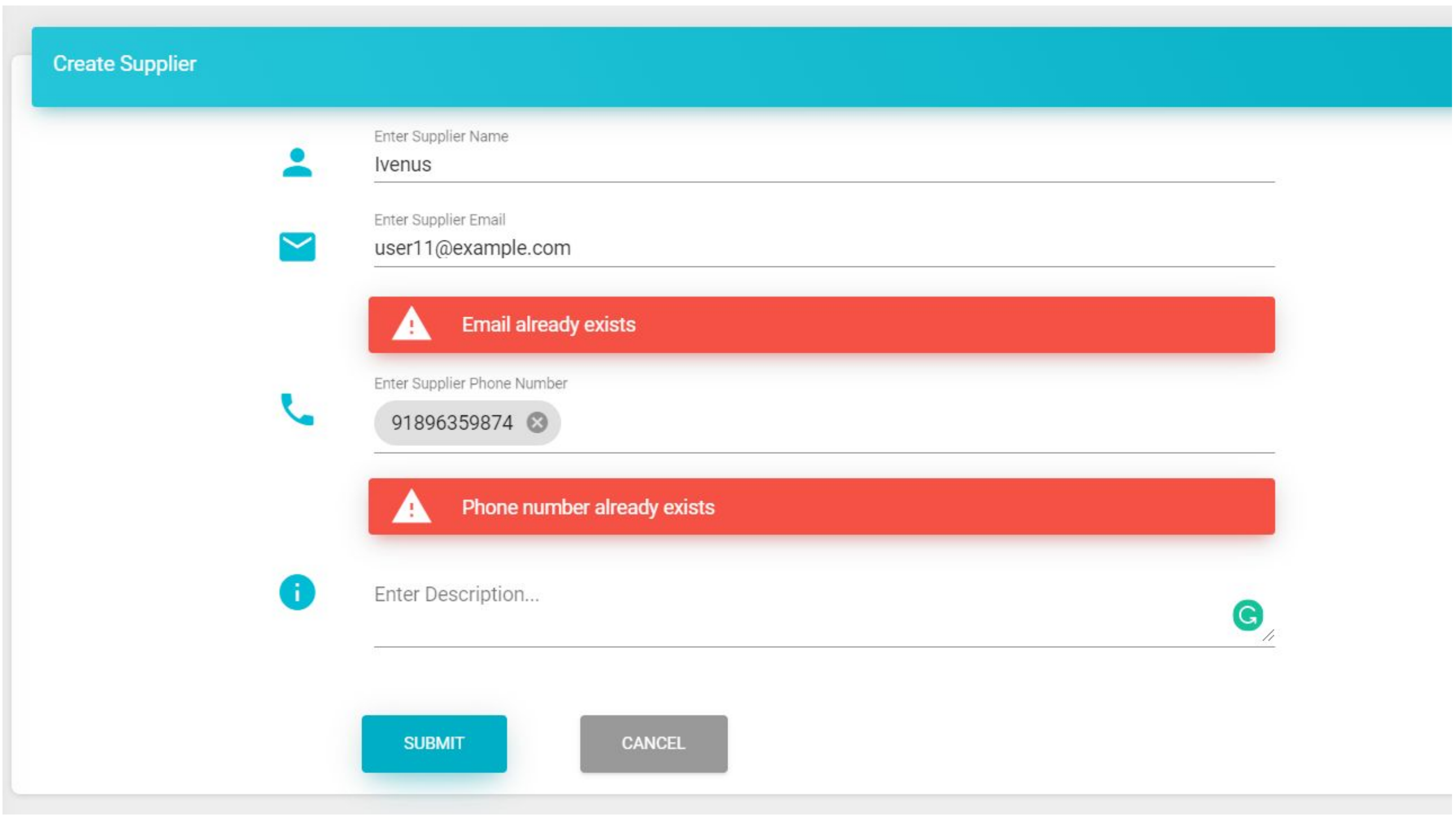
Response body

```
{
  "detail": {
    "email": "Email already exists",
    "phone_number": "Phone number already exists"
  }
}
```

Download

Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: http://127.0.0.1:8000
content-length: 88
content-type: application/json
date: Sun, 12 Apr 2020 12:28:28 GMT
server: uvicorn
```


Create Supplier

Enter Supplier Name: Ivenus

Enter Supplier Email: user11@example.com

Email already exists

Enter Supplier Phone Number: 91896359874

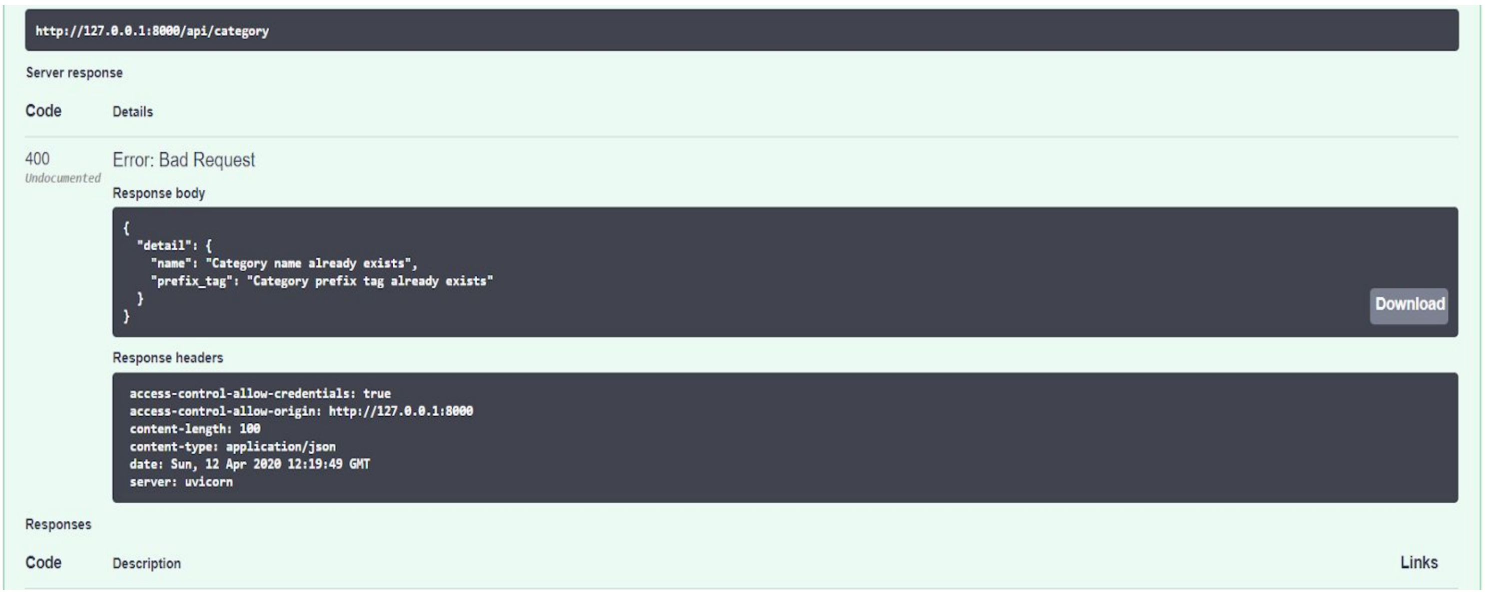
Phone number already exists

Enter Description...

Fig 6.4.4. Supplier-validation

The supplier email and Contact number should be unique.

6.4.5. Asset Category Validation [11]



```
http://127.0.0.1:8000/api/category
```

Server response

Code	Details
400 *Undocumented*	Error: Bad Request

Response body

```
{
  "detail": {
    "name": "Category name already exists",
    "prefix_tag": "Category prefix tag already exists"
  }
}
```

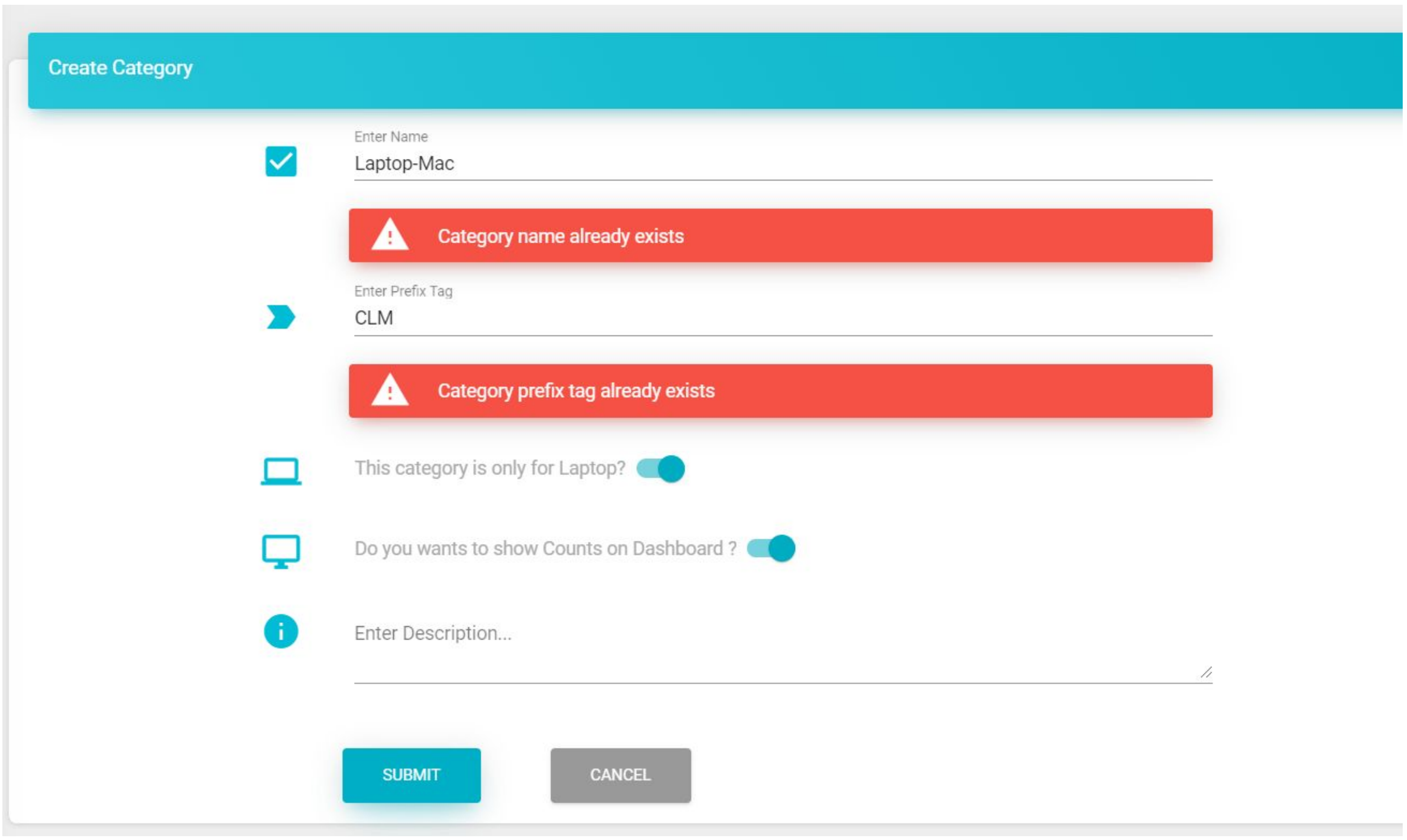
Download

Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: http://127.0.0.1:8000
content-length: 100
content-type: application/json
date: Sun, 12 Apr 2020 12:19:49 GMT
server: uvicorn
```

Responses

Code Description Links



Create Category

Enter Name: Laptop-Mac

Category name already exists

Enter Prefix Tag: CLM

Category prefix tag already exists

This category is only for Laptop? [✓]

Do you wants to show Counts on Dashboard ? [✓]

Enter Description...

Fig 6.4.5. Asset-category-validation [11]
Category name and Asset Prefix Tag should be unique.

6.5 APPLICATION SNAPSHOTS

6.5.1 Dashboard [11]

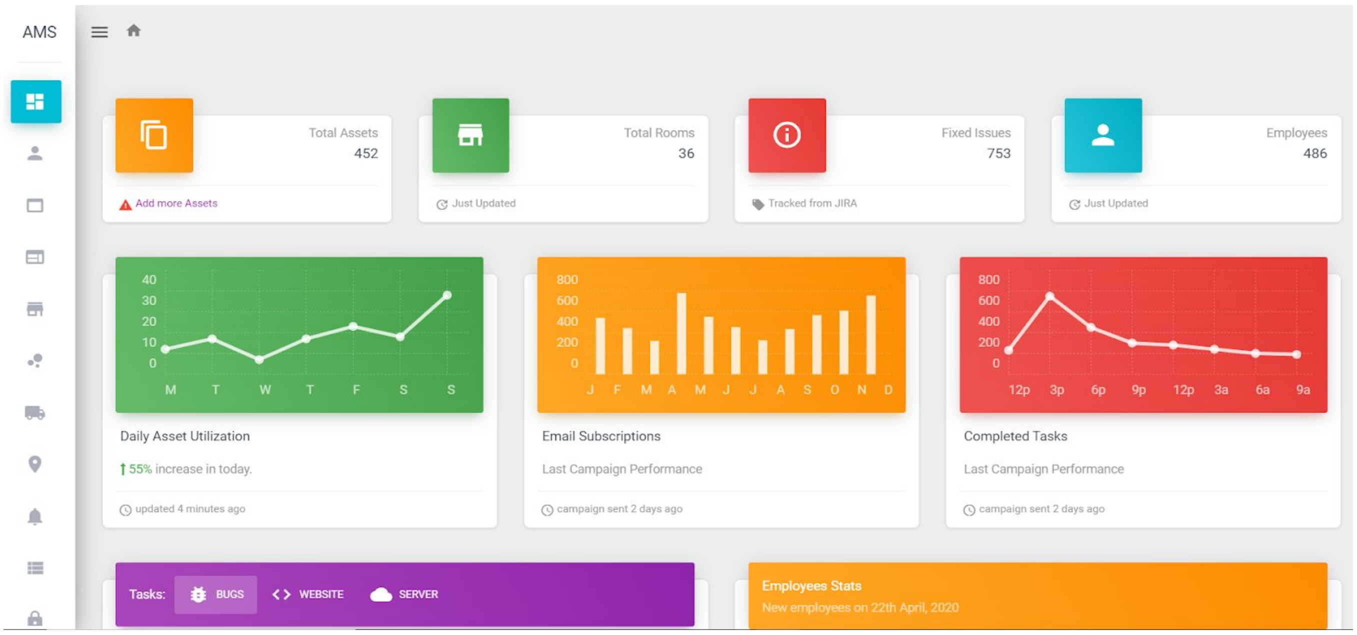
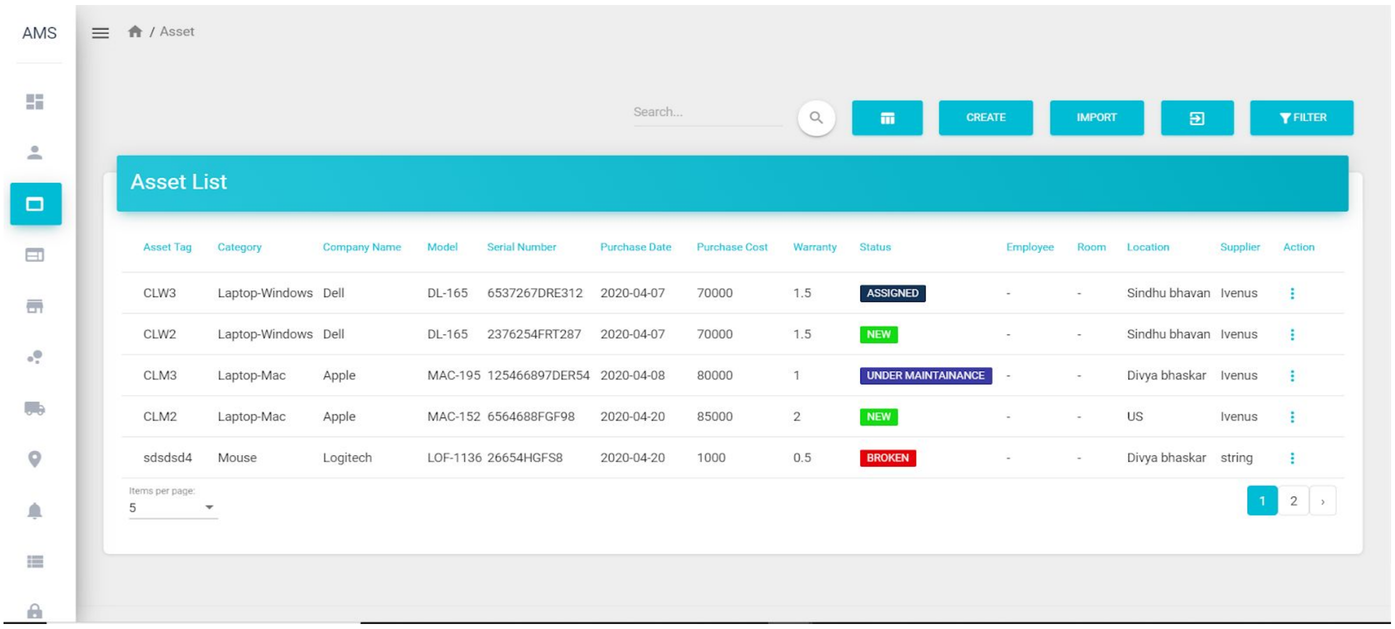


Fig 6.5.1. Dashboard [11]

The dashboard shows the various counts and Activity log of various Assets. Also, there are some statistical graphs and charts shown.

6.5.2 Asset List



Asset Tag	Category	Company Name	Model	Serial Number	Purchase Date	Purchase Cost	Warranty	Status	Employee	Room	Location	Supplier	Action
CLW3	Laptop-Windows	Dell	DL-165	6537267DRE312	2020-04-07	70000	1.5	ASSIGNED	-	-	Sindhu bhavan	Ivenus	⋮
CLW2	Laptop-Windows	Dell	DL-165	2376254FRT287	2020-04-07	70000	1.5	NEW	-	-	Sindhu bhavan	Ivenus	⋮
CLM3	Laptop-Mac	Apple	MAC-195	125466897DER54	2020-04-08	80000	1	UNDER MAINTAINANCE	-	-	Divya bhaskar	Ivenus	⋮
CLM2	Laptop-Mac	Apple	MAC-152	6564688FGF98	2020-04-20	85000	2	NEW	-	-	US	Ivenus	⋮
sdsdsd4	Mouse	Logitech	LOF-1136	26654HGFS8	2020-04-20	1000	0.5	BROKEN	-	-	Divya bhaskar	string	⋮

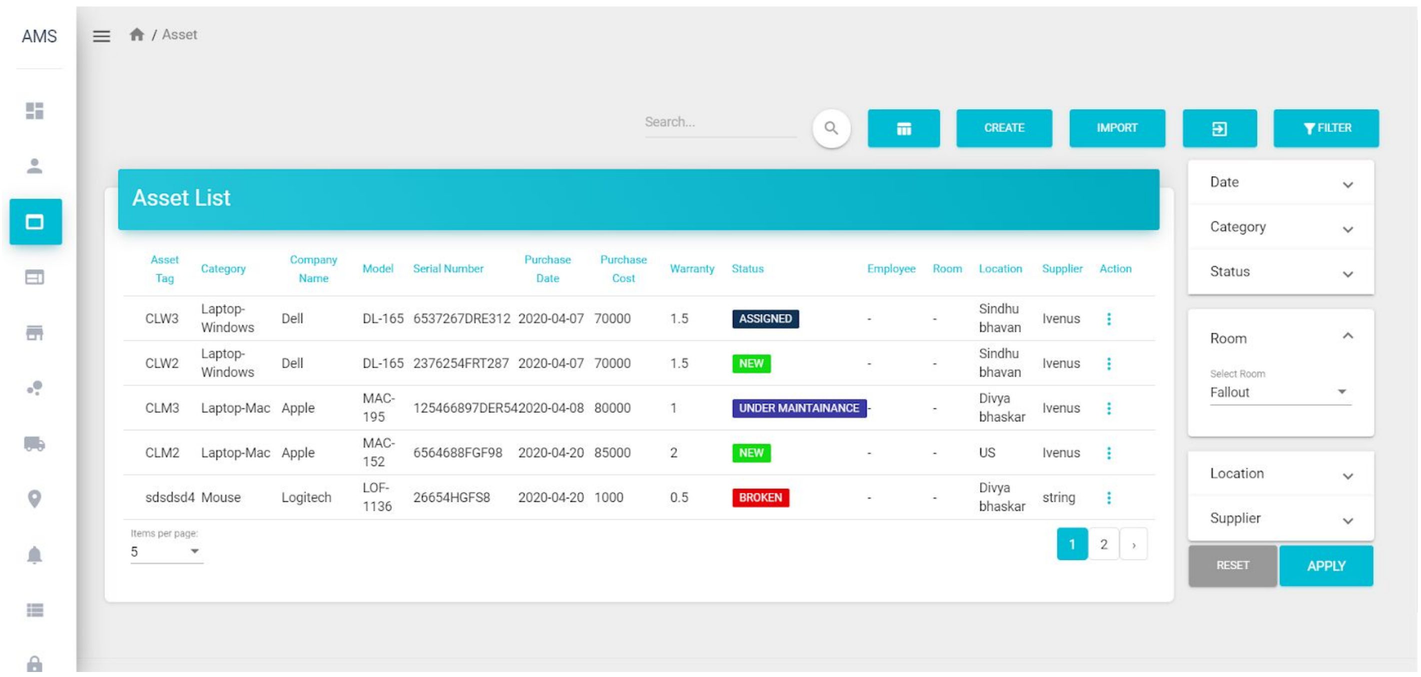
Items per page: 5

1 2 ›

Fig 6.5.2. Asset-list

List of All Assets with status code and relevant details.

6.5.3 Asset List Filters



Asset List

Asset Tag	Category	Company Name	Model	Serial Number	Purchase Date	Purchase Cost	Warranty	Status	Employee	Room	Location	Supplier	Action
CLW3	Laptop-Windows	Dell	DL-165	6537267DRE312	2020-04-07	70000	1.5	ASSIGNED	-	-	Sindhu bhavan	Ivenus	⋮
CLW2	Laptop-Windows	Dell	DL-165	2376254FRT287	2020-04-07	70000	1.5	NEW	-	-	Sindhu bhavan	Ivenus	⋮
CLM3	Laptop-Mac	Apple	MAC-195	125466897DER542	2020-04-08	80000	1	UNDER MAINTAINANCE	-	-	Divya bhaskar	Ivenus	⋮
CLM2	Laptop-Mac	Apple	MAC-152	6564688FGF98	2020-04-20	85000	2	NEW	-	-	US	Ivenus	⋮
sdsdsd4	Mouse	Logitech	LOF-1136	26654HGFS8	2020-04-20	1000	0.5	BROKEN	-	-	Divya bhaskar	string	⋮

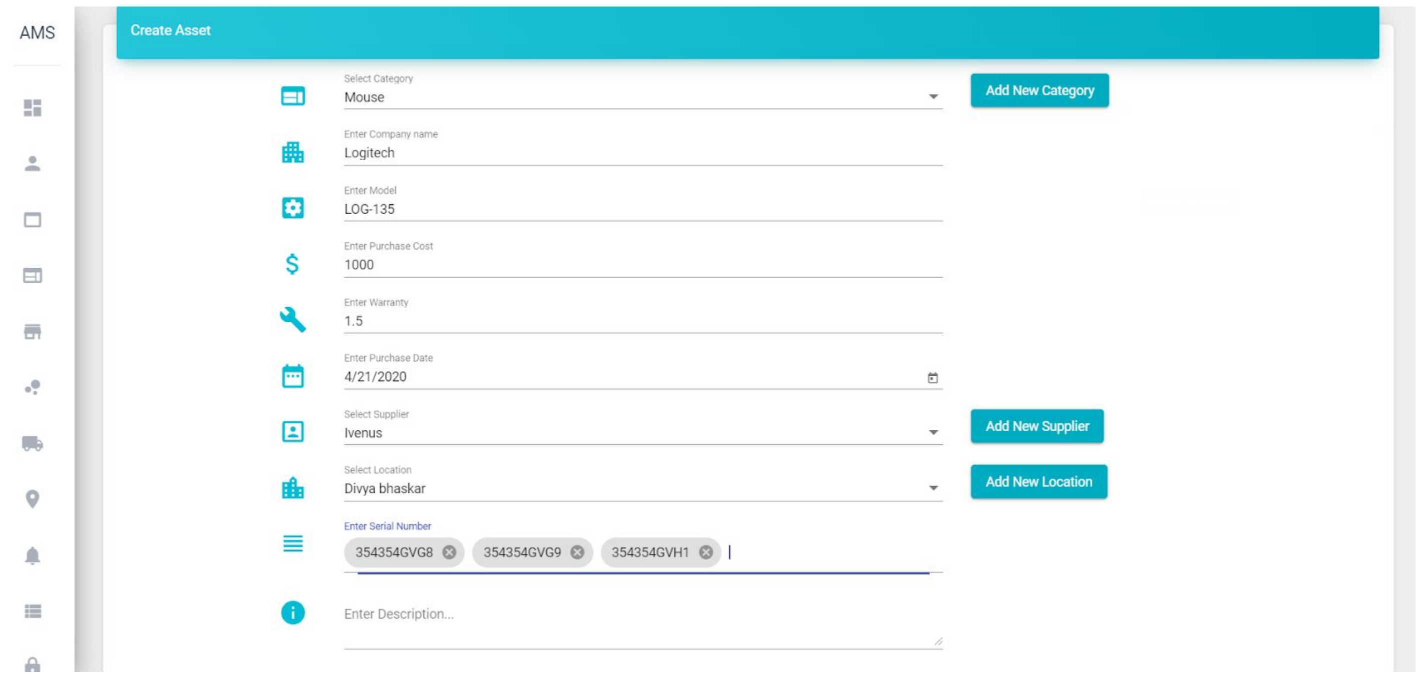
Items per page: 5

1 2 ›

Select Room: Fallout

Fig 6.5.3 Asset-list-filter
Asset list with various filters like Status, Category, Location, and Activity from the start date to end date. So, based on filtering criteria Assets will be sorted out.

6.5.4 Add Asset into System



Select Category	Mouse
Enter Company name	Logitech
Enter Model	LOG-135
Enter Purchase Cost	1000
Enter Warranty	1.5
Enter Purchase Date	4/21/2020
Select Supplier	Ivenus
Select Location	Divya bhaskar
Enter Serial Number	354354GVG8, 354354GVG9, 354354GVH1
Enter Description...	

Fig 6.5.4. Add-asset-into-system
Add multiple Asset within the system with the same configuration and different serial number

Chapter 7
Conclusion and Future Extension

7.1 CONCLUSION

As keeping a track of assets is important for every organization. This application ties the assets with the IT infrastructure of the organization. With a robust asset management system, management and IT professionals can review and monitor all types of assets within the organization. The information can be used to make detailed decisions about the purchase and other aspects of the asset’s life cycle.

The system helps in an accurate record of all types of assets. Using the software can help in efficient resource planning. It can also reduce the risk of theft of assets. An asset management system will help to monitor the assets located in different locations and departments. We will get to know where the assets are located. We can run reports to know about ownership, service details, and other insightful information.

The Project provides meaningful information regarding assets to optimize asset utilization and remove manual management tasks of maintaining a spreadsheet for various assets. The featured dashboard can help to get a quick decision by asset count and statistic charts or graphs information.

7.2 FUTURE EXTENSION

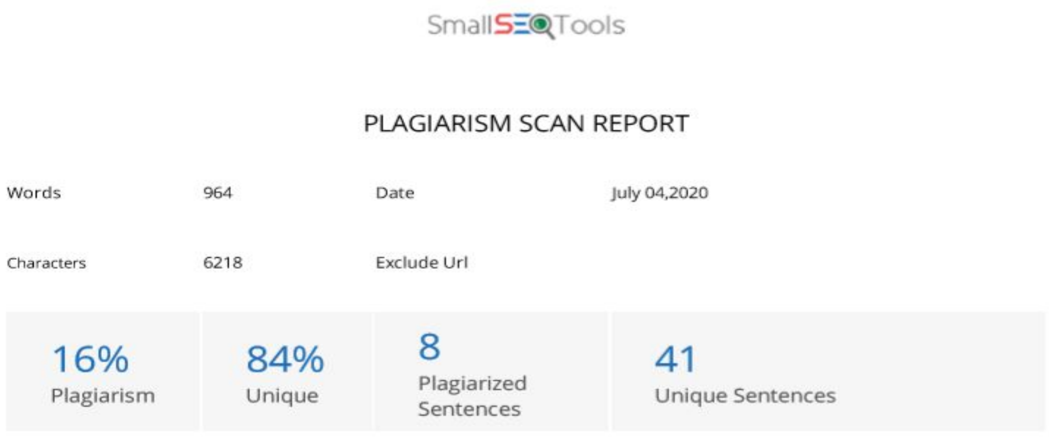
- Develop a request module by that Employee can request for a particular asset with a specific configuration and based on that assignment-reassignment of asset tasks would be automated using AI-based algorithms.
- Lost and Found: In the case of asset Lost, If someone lost his/her physical asset in the organization then put asset information (like Assert Tag) onto the system then a notification is sent to employees. In the case of asset Found, If someone found an asset then find the owner name of that asset by entering Asset Tag into the system and send an acknowledgment to the owner.

References

[1] FastAPI Inc. (2020,April). Documentation. https://fastapi.tiangolo.com/
[2] SQLAlchemy Inc. (2020,March). Documentation. https://docs.sqlalchemy.org/en/13/
[3] Psycopg2 Inc. (2020,March). Documentation. https://wiki.postgresql.org/wiki/Psycopg2_Tutorial
[4] Python Inc. (2019,Dec). Documentation. https://docs.python.org/3/
[5] PostgreSQL Inc. (2020, February). Documentation. https://www.postgresql.org/docs/
[6] Angular 9 Inc. (2020,February). Documentation. https://angular.io/docs
[7] Angular Material Inc. (2020,June). Documentation. https://material.angular.io/
[8] Docker Inc. (2020,May). Documentation. https://docs.docker.com/
[9] StackOverflow Inc. (2020,April). Community. https://stackoverflow.com/questions
[10] Balsamiq Wireframing. (2020,February).Documentation. https://balsamiq.cloud/
[11] Harshil Sadharakiya. (16CP052**),** BVM Engineering College, Vallabh Vidyanagar.
[12] PyCharm.(2020,January).Documentation. https://en.wikipedia.org/wiki/PyCharm
[13] VSCode.(2020,February).Documentation. https://code.visualstudio.com/docs

Plagiarism Report

- Average of All Plagiarism Reports - 14.67%



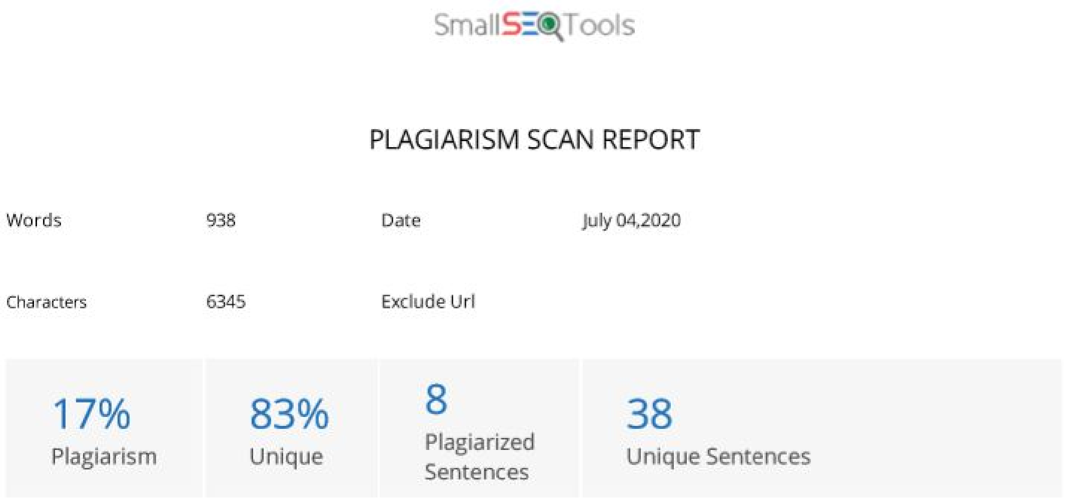
SmallSEOTools

PLAGIARISM SCAN REPORT

Words	964	Date	July 04,2020
Characters	6218	Exclude Url	

16% Plagiarism	84% Unique	8 Plagiarized Sentences	41 Unique Sentences

---



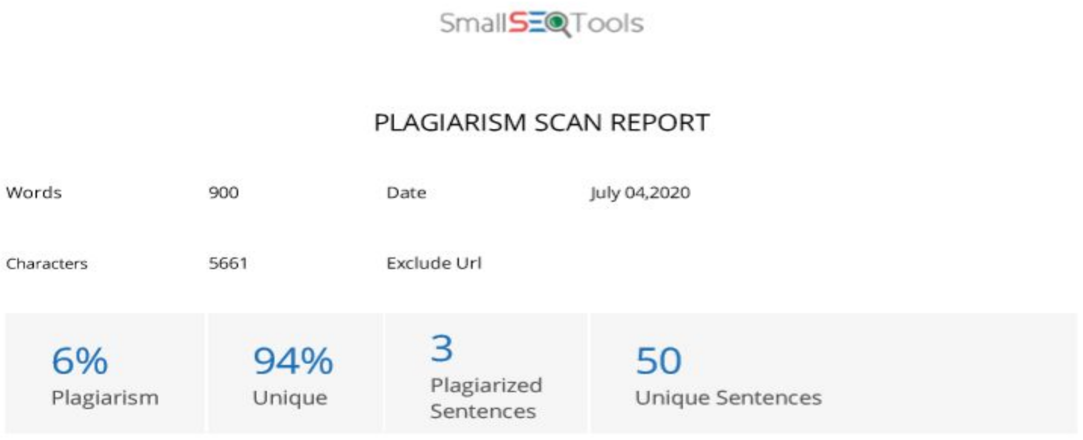
SmallSEOTools

PLAGIARISM SCAN REPORT

Words	938	Date	July 04,2020
Characters	6345	Exclude Url	

17% Plagiarism	83% Unique	8 Plagiarized Sentences	38 Unique Sentences

---



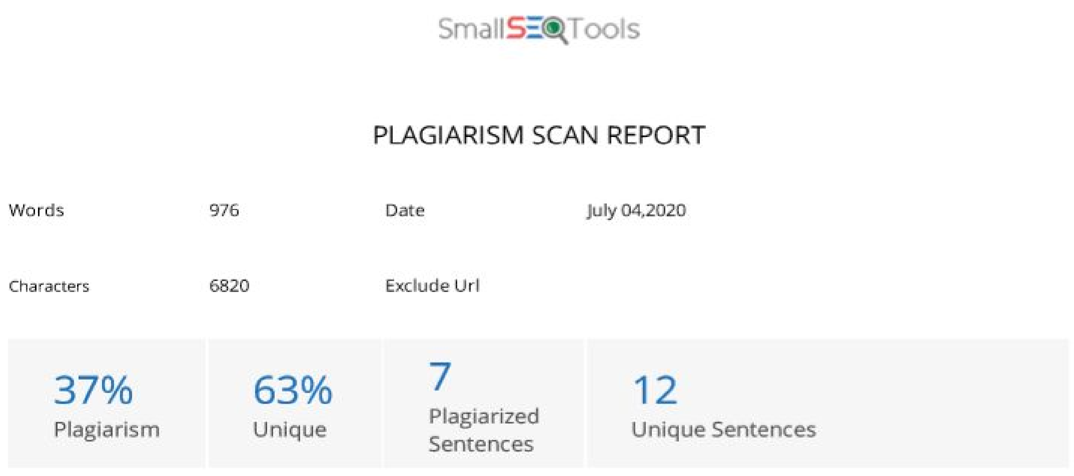
SmallSEOTools

PLAGIARISM SCAN REPORT

Words	900	Date	July 04,2020
Characters	5661	Exclude Url	

6% Plagiarism	94% Unique	3 Plagiarized Sentences	50 Unique Sentences

---

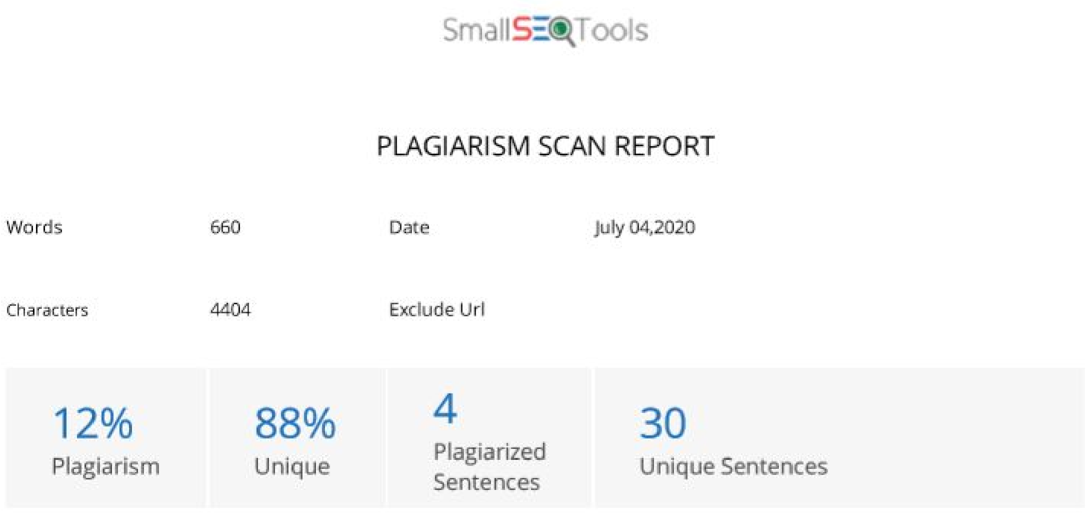


PLAGIARISM SCAN REPORT

Words	976	Date	July 04,2020
Characters	6820	Exclude Url	

37% Plagiarism	63% Unique	7 Plagiarized Sentences	12 Unique Sentences

---

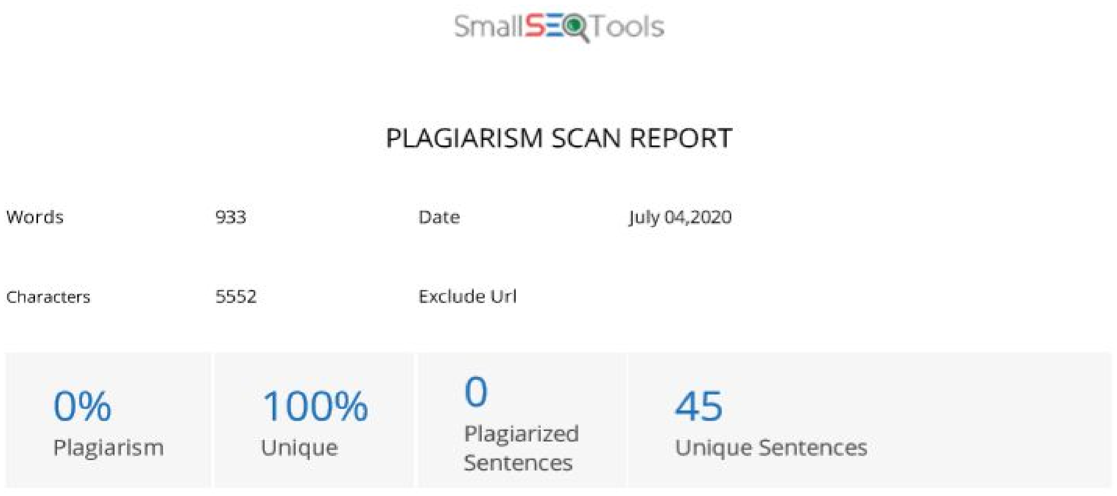


PLAGIARISM SCAN REPORT

Words	660	Date	July 04,2020
Characters	4404	Exclude Url	

12% Plagiarism	88% Unique	4 Plagiarized Sentences	30 Unique Sentences

---



PLAGIARISM SCAN REPORT

Words	933	Date	July 04,2020
Characters	5552	Exclude Url	

0% Plagiarism	100% Unique	0 Plagiarized Sentences	45 Unique Sentences