

PROJECT WORK PHASE -1 REPORT ASSET MANAGEMENT SYSTEM

SUBMITTED BY

VENKATA SURYA .V

11209A020

ARAVIND .S

11209M001

GUIDED BY

MR V BALU

ASSISTANT PROFESSOR



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

**SRI CHANDRASEKHARENDRASARASWATHI VISWA
MAHAVIDYALAYA**

[NOV 2023]

Sri Chandrasekharendra Saraswathi Viswa Mahavidyalaya

Enathur, Kanchipuram – 631501



BONAFIDE CERTIFICATE

This is to certify that the Project work phase-1 Report entitled **ASSET MANAGEMENT SYSTEM** is the bonafide work carried out by Mr. VEMPARALA VENKATA SURYA, ARAVIND.S Reg.No 11209A020, 11209M001 during the academic year 2023-2024.

MR V.BALU
Assistant Professor,
Department of CSE,
SCSVMV.

DR .M. SENTHIL KUMARAN
Associate Professor & Head,
Department of CSE,
SCSVMV.

Submitted for the project work phase -1 viva - voce examination held on _____

Place : Kanchipuram.

Date :

Examiner 1

Examiner 2

DECLARATION

It is certified that the project work phase -1 titled **Asset Management System** is originally implemented by me. No ideas, processes, results or words of others have been presented as my own work. Due acknowledgement is given wherever others' work or ideas are utilized.

- a. There is no fabrication of data or results which have been compiled /analyzed.
- b. There is no falsification by manipulating data or processes, or changing or omitting data or results.

I understand that the project is liable to be rejected at any stage (even at a later date) if it is discovered that the project has been plagiarized, or significant code has been copied. I understand that if such malpractices are found, the project will be disqualified, and the Degree awarded itself will become invalid.

Signature of the student with date

TABLE OF CONTENTS

Chapter No.	Title	Page No.
	ABSTRACT	6
1.	Introduction	
1.1	Introduction	7
1.2	Scope of the project	8
1.3	Existing System	8
2.	Literature Survey	
2.1	Literature Survey	9
2.2	Problem Statement	9
3.	Proposed Method / Architecture / Process /Methodology / Project Description	
3.1	Proposed method	10
3.2	Architecture	11
3.3	project Description	13
3.4	Module Description	15
3.5	Methodology	16
4.	Implementation Work	
4.1	Implementation process	17
4.2	Code	18
4.3	Testing and methodologies	37
4.4	Result Analysis / Result comparison	38

5.	Conclusion	42
6.	Future Enhancement	43
7.	References	44

Abstract

In this project, we developed a system for effective management of asset for organizations to optimize their resources and streamline operations. This abstract introduces an innovative Asset Management System built on the MERN (MySQL, Express, React, Node) stack, designed to efficiently manage product data. This web-based solution offers a comprehensive approach. The system allows users to record, categorize, and monitor a wide array of products within an organization. Once the product is searched into the database it fetch the variants available for the product. Before fetching master has accesses to add products, variants, values etc . In react it offence more scalable dynamically

Keywords: Asset management, MERN stack, scalability, categorization, streamline operations, variants

CHAPTER 1

Introduction

1. Introduction

The Asset Management System is a web-based application designed to streamline asset tracking and management within organizations. Developed using React.js for the front end and Node.js for the back end, the system ensures secure user authentication and authorization with role-based permissions. The intuitive dashboard provides a quick overview of asset status and distribution through visualizations and charts. Key features include comprehensive asset management and search capabilities. The technology stack incorporates React.js for its interactive UI and Node.js for its scalability on the backend. A relational database ensures data integrity. Overall, the system aims to enhance organizational efficiency by providing a user-friendly platform for managing assets, contributing to improved productivity and resource optimization.

Functionality: The Application works by compiling information about various Asset data from organization. This information can include details about the materials used and current location. The Application also includes features for managing Asset within a collection. For example, users can track the conservation needs of each Asset, as well as its exhibition history and provenance. The Application may also include tools as well as for tracking the movement of Assets between different locations.

User Interface: The user interface of the Asset management Application is designed to be intuitive and user-friendly. Users can easily search for Assets by product name it fetch automatically the data from the provided database.

Security: As Assets are often unique and valuable works of an organization. security is a critical concern for any Asset management Application. The platform should include robust security measures to prevent unauthorized access and to protect against other users . As of now we have prepared regarding admin dashboard The Application may include features as managing access to sensitive information about each Asset, such as its provenance or conservation needs.

1.1 Scope of the project

The Application for finding Asset details within the organization involve -in developing a software system that provides a user-friendly interface for searching products . Here are some possible features and functionalities that could be included:

Database Management: The system should have a database to store information about product Assets, such as product name, variant, value of the product.

Application: If the parameters match, then the relevant information about the Asset should be displayed to the user.

Reporting: The system should generate reports on the search results, such as the number of records, records with respective product

1.2 Existing system

The existing systems for asset management in many organizations often suffer from several drawbacks, hindering their effectiveness in optimizing resources and streamlining operations. we develop this with dynamic languages such as react JS If we want to search some product Asset in the web ,we can't select our own options available in the web . So this system is very useful to organizations to track the Asset from which product and description about the Asset.

Chapter 2

Literature survey

2.1 Literature survey

1. Investigation: A Productive Asset Management Web Application Computer Systems Science & Engineering DOI:10.32604/csse.2021.015314
2. M Wang, J Tan, Y Li - 2015 IEEE international conference on ..., 2015 - ieeexplore.ieee.org
3. Markus Keinänen Creation of a web service using the MERN stack
4. D Sarkar, H Patel, B Dave - ... Journal of Construction Management, 2022 - Taylor & Francis
5. L Turnip, A Triayudi , ID Solihati - Jurnal Mantik , 2020 - iocscience.org
Asset Management Information System Using the Waterfall Method (Case Study: National University)

2.2 Problem Statement

In today's rapidly evolving business landscape, the efficient management of asset is more importance to organizations seeking to optimize resource allocation and streamline their operations. Despite the significance of effective asset management, many organizations face challenges in cataloging, monitoring, and categorizing their products. This often results in inefficient resource utilization and hinders decision-making processes.

To address this issue, we propose the development of an innovative Asset Management System built on the MERN (MySQL, Express, React, Node) stack. The primary problem that this system aims to solve is the lack of a robust and comprehensive solution for managing product data effectively.

Chapter 3

3.1 Proposed method

In this we proposed Asset management using the MERN (MYSQL, Express.js, React.js, Node.js) stack for web development involves creating a system to efficiently track and manage assets within an organization

1) Database Design (MY SQL):

Asset Collection:

Create a MY SQL collection to store asset information. Each document in this collection represents a unique asset and includes fields such as asset id, name, variants, purchase value data, etc.

2) Backend Development (Node.js and Express.js):

API Endpoints:

Implement RESTful API endpoints using Express.js to perform CRUD operations on assets.

Middleware:

Implement middleware for user authentication and authorization to ensure that only authorized users can access certain endpoints.

3) Frontend Development (React.js):

User Interface:

Develop a user-friendly interface using React.js to interact with the asset management system.

Create components for listing assets, adding new assets, updating existing assets, etc.

State Management:

Utilize state management libraries like Redux to manage the application's state, especially if the application grows in complexity.

4) Integration & User Authentication and Authorization:

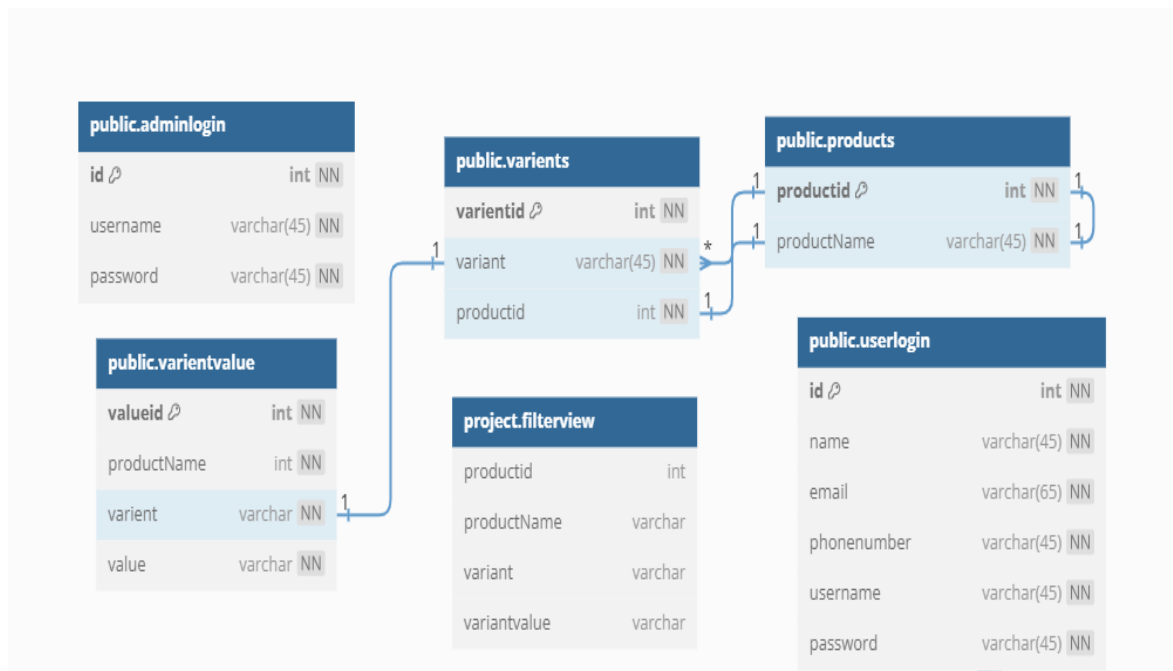
Connect the React.js frontend with the Node.js backend through API calls.

Implement Authentication:

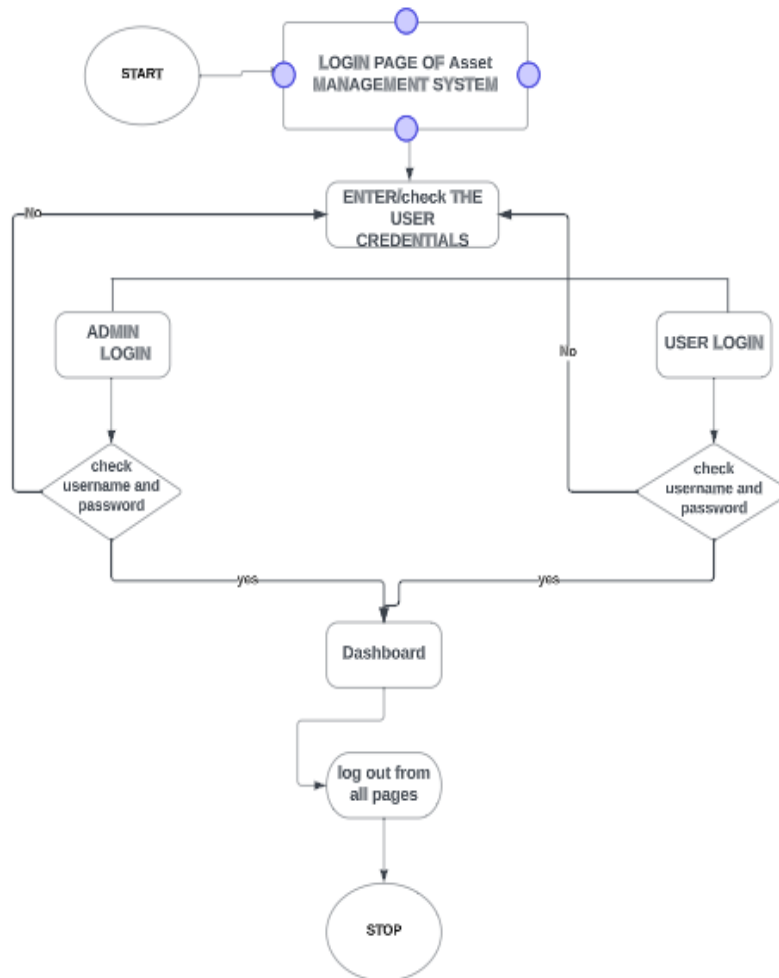
Use a secure authentication method (e.g., JWT) to authenticate users.

3.2 Architecture

Database Design



Architecture diagram



3.3 Project Description

The Asset Management System is a web-based application designed to streamline and optimize the management of assets within an organization. Developed using React.js for the front end, Tailwind for designing, Node.js and Express for the backend, and MySQL for the database. This system provides a user-friendly interface for efficient asset tracking, variant management, and purchase transactions.

Key Features: The system offers a secure and intuitive login mechanism, allowing authorized users to access and manage assets seamlessly. Users can add new assets with different variants, and assign specific values to each variant. The ability to delete variants provides flexibility in adapting to changing organizational needs. The search functionality lets users quickly locate specific assets, enhancing overall accessibility.

Purchase Functionality: One of the core functionalities is the purchase feature, where users can select and purchase products with their respective variants by simply checking the desired options. This streamlined process not only simplifies procurement but also ensures accuracy in the selection of products and their variants.

Technological Stack: The project utilizes a modern and efficient technological stack. React.js provides a dynamic and responsive user interface, while Node.js and Express establish a robust backend connection. MySQL serves as the database, ensuring data integrity and reliability. The integration of these technologies creates a scalable and maintainable system capable of meeting the evolving asset management needs of the organization.

In the Asset Management System enhances organizational efficiency by offering a centralized platform for managing assets, variants, and purchase transactions. Its user-friendly interface, coupled with the power of React.js, Node.js, Express, and

MySQL, makes it a versatile and indispensable tool for any organization seeking to optimize its asset management processes.

The system stores details about products, including the product ID, name. For each product, the system can store multiple variants, with details including the variant ID, variant name, value, and product name to help with searching and filtering the variants in the console.

In addition, the system provides reporting functionality, allowing to generate reports based on the information in the database. Reports can be generated based on various criteria, such as by product or variant.

3.4 Module Description

The Asset Management System is an application designed to manage the details of Assets in a collection or exhibition. This system contains two modules describe as follows

- Administrator module
- User Authentication module
- Variant management module
- Purchase module

Administrator Module: Administrator has a full authority to maintain the system with a database. The Admin can log in through admin login. The main functions of the administrator

- **Add product:** Admin can add details about the product and assign one ID for each product. This contains productid and ProductName.
- **Add variant:** Admin has access to add Asset Variant. It contains variant id, variant, and productid.
- **Add Variant value:** Admin has accesses to add Value for each variant of a particular product. In this contains id, variant id, and value.

User Authentication Module: The User Authentication module forms the foundation of the system, ensuring secure access to the Asset Management System. This module includes user registration, login, and authentication mechanisms. User credentials are securely stored and managed, allowing only authorized personnel to interact with the system.

Variant Management Module: This module focuses on the creation, modification, and deletion of variants associated with each asset. Users can define specific attributes and values for each variant, tailoring the system to accommodate diverse asset specifications. The ability to delete variants ensures

adaptability to changing organizational requirements.

Purchase Module: The Purchase Transaction module streamlines the purchase process by allowing users to select and purchase products along with their specified variants. Users can easily navigate through available assets, select desired variants via checkboxes, and proceed with the purchase. This module ensures accuracy and efficiency in the procurement of assets

3.5 Methodology

An Asset management system is a software system designed to help manage and organize Assets in a collection or exhibition. The methodology for developing an Asset management system.

1. **Requirements gathering:** Identify key search criteria to efficiently locate and manage assets within the system (e.g., source and product variant). Determine the user interface elements and functionalities needed for optimal asset search and management.
2. **Design:** Based on the requirements gathered, design the Application module's architecture and user interface. The system should include modules for managing the asset.
3. **Implementation:** Develop the Application module according to the design. This may involve selecting and configuring an Application platform, creating the user interface, and developing the necessary backend components.
4. **Integration:** Integrate the Application module with the Asset management system to ensure that it can access the necessary data. Test the module to ensure that it meets the requirements and that the search results are accurate and relevant.
5. **Maintenance:** Provide ongoing maintenance and support for the Application module. This may involve fixing bugs, optimizing search performance, and updating the module to meet changing requirements.

Chapter 4

Implementation work

4.1 Implementation Process

- 1. Design:** We design the architecture and user interface for the Asset management console. This may involve creating wireframes and mockups to visualize the user interface and flow of the console. Identify the necessary components and technologies needed to implement the console.
- 2. Database design:** We design and implement the database schema for the Asset, including tables relationships, and views.
- 3. User interface development:** We developed the user interface for the Asset Management Console using appropriate front-end technologies, such as React.js. Tailwind CSS was employed to design the web pages, creating views and templates for various pages of the console.
- 4. Back-end development:** We develop the back-end functionality of the console using Node.js and REST API server-side technologies. This includes implementing the data entry, search, and detailed display of Asset information.
- 5. User authentication and authorization:** We implement a user authentication and authorization system to ensure that only authorized users can access the Asset management console.

4.2 Sample Code

```
const express = require('express');
const mysql = require('mysql2');
const bodyParser = require('body-parser');
const cors = require('cors');
const cookieParser = require('cookie-parser');
const jwt = require('jsonwebtoken');
const svgCaptcha = require('svg-captcha');
const session = require('express-session');

const app = express();
const port = 3000; // You can change this to your desired port

app.use(cors(
  {
    origin:["http://localhost:3001"],
    methods:["POST","GET","PUT","DELETE"],
    credentials:true
  }
));
app.use(bodyParser.json());
app.use(cookieParser());
app.use(
  session({
    secret: "",
    resave: true,
    saveUninitialized: true,
  })
);
```

```

const connection = mysql.createConnection({
  host: 'localhost',
  user: '',
  password: '',
  database: ''
});

connection.connect();

app.get('/api/products/count', (req, res) => {
  const query = 'SELECT COUNT(*) as count FROM products'; // Replace 'products'
  with your actual table name

  connection.query(query, (err, results) => {
    if (err) {
      console.error('Error fetching product count:', err);
      res.status(500).json({ error: 'Internal server error' });
    } else {
      if (results && results.length > 0) {
        const productCount = results[0].count;
        res.json({ count: productCount });
      } else {
        console.error('No results found in the "products" table.');
```

res.status(500).json({ error: 'Internal server error' });

```

      }
    }
  });
});

app.get('/api/products', (req, res) => {
  const query = 'SELECT * FROM products'; // Replace 'products' with your actual table
  name

```

```

connection.query(query, (err, results) => {
  if (err) {
    console.error('Error fetching products:', err);
    res.status(500).json({ error: 'Internal server error' });
  } else {
    res.json({ products: results });
  }
});
});

app.get('/api/varient/count', (req, res) => {
  const query = 'SELECT COUNT(*) as count FROM varients'; // Replace 'products'
  with your actual table name

  connection.query(query, (err, results) => {
    if (err) {
      console.error('Error fetching product count:', err);
      res.status(500).json({ error: 'Internal server error' });
    } else {
      if (results && results.length > 0) {
        const varientCount = results[0].count;
        res.json({ count: varientCount });
      } else {
        console.error('No results found in the "products" table.');
```

with your actual table name

```

        res.status(500).json({ error: 'Internal server error' });
      }
    }
  });
});

app.get('/api/user/count', (req, res) => {
  const query = 'SELECT COUNT(*) as count FROM userlogin';
  connection.query(query, (err, results) => {

```

```

    if (err) {
      console.error('Error fetching user login count:', err);
      res.status(500).json({ error: 'Internal server error' });
    } else {
      if (results && results.length > 0) {
        const userLoginCount = results[0].count;
        res.json({ count: userLoginCount });
      } else {
        console.error('No results found in the "userlogins" table.');
        res.status(500).json({ error: 'Internal server error' });
      }
    }
  });
});

const generateRandomAlphabets = () => {
  const alphabets =
    'ABCDEFGFG234abcdefghHIJKLMNOPQRSijklmnopTU#$$%^&*_VWXYZ01qrstuvwxyz56789!@';

  const captchaLength = 5; // You can adjust the length as needed
  let captcha = '';
  for (let i = 0; i < captchaLength; i++) {
    const randomIndex = Math.floor(Math.random() * alphabets.length);
    captcha += alphabets[randomIndex];
  }
  return captcha;
};

app.get('/captcha', (req, res) => {
  const captchaText = generateRandomAlphabets();
  req.session.captcha = captchaText;
  res.json({ captcha: captchaText });
});

```

```

app.post('/loginform', (req, res) => {
  const { username, password, usertype, userEnteredCaptcha } = req.body;

  // Verify the user-entered CAPTCHA text
  if (userEnteredCaptcha !== req.session.captcha) {
    return res.status(400).json({ message: 'CAPTCHA verification failed' });
  }

  let query = '';
  if (usertype === 'user') {
    query = `SELECT * FROM userlogin WHERE username = ? AND password = ?`;

  } else if (usertype === 'admin') {
    query = `SELECT * FROM adminlogin WHERE username = ? AND password = ?`;
  } else {
    return res.status(400).json({ message: 'Invalid user type' });
  }

  connection.query(query, [username, password], (error, results) => {
    if (error) {
      console.error('Error querying the database:', error);
      return res.status(500).json({ error: 'Internal server error' });
    }

    if (results.length > 0) {
      const user = results[0];
      res.json({ message: 'Login successful', userType: usertype, userId: user.id
        /* Include any other user-related data you need */ });
    } else {
      console.error('Invalid credentials:', { username, usertype });
      res.status(401).json({ message: 'Invalid credentials' });
    }
  });
});

```

```

    }
  });
});

app.get('/dashboard', (req, res) => {
  return res.json({ Status: "Success" })
})

app.post('/register', (req, res) => {
  const { name, email, phoneNumber, username, password } = req.body;

  // Define the SQL query to insert user data into the database
  const sql = 'INSERT INTO userlogin (name, email, phonenumber, username, password) VALUES (?, ?, ?, ?, ?)';
  const values = [name, email, phoneNumber, username, password];

  // Execute the insert query
  connection.query(sql, values, (err, result) => {
    if (err) {
      console.error('MySQL error:', err);
      res.status(500).json({ message: 'Registration failed' });
    } else {
      console.log('User registered:', result.insertId);
      res.status(200).json({ message: 'Registration successful' });
    }
  });
});

app.get('/assesst', (req, res) => {
  const sql = 'SELECT * FROM products';
  connection.query(sql, (err, data) => {
    if (err) return res.json(err);
    return res.json(data);
  });
});

```

```

    });
  });

  app.post('/addproduct', (req, res) => {
    const { id, productName } = req.body;

    // Check if a product with the same 'productid' already exists
    const checkProductQuery = 'SELECT COUNT(*) as count FROM products WHERE
productid = ?';

    connection.query(checkProductQuery, [id], (checkError, checkResults) => {
      if (checkError) {
        console.error('Error checking product existence:', checkError);
        return res.status(500).json({ error: 'Failed to check product existence.' });
      }

      const productExists = checkResults[0].count > 0;

      if (productExists) {
        // Product with the same 'productid' already exists, handle accordingly
        return res.status(400).json({ error: 'Product with the same ID already exists.' });
      }

      // Product with 'productid' doesn't exist, proceed with insertion
      const insertProductQuery = 'INSERT INTO products (productid, productName)
VALUES (?, ?)';

      connection.query(insertProductQuery, [id, productName], (insertError,
insertResults) => {
        if (insertError) {
          console.error('Error adding product:', insertError);
          return res.status(500).json({ error: 'Failed to add product.' });
        }
      }
    }
  });

```



```

    res.json({ message: 'Product added successfully' });
  });
});
});

app.post('/adddescription', (req, res) => {
  const { descriptions } = req.body;

  if (!descriptions || !Array.isArray(descriptions) || descriptions.length === 0) {
    return res.status(400).json({ error: 'Invalid descriptions data.' });
  }

  // Start a transaction
  connection.beginTransaction(function (err) {
    if (err) {
      console.error('Error beginning transaction:', err);
      return res.status(500).json({ error: 'Failed to begin transaction.' });
    }

    const sql = 'INSERT INTO varients (variant) VALUES ?'; // Note the change in
    SQL query

    // Prepare the data for insertion
    const valuesToInsert = descriptions.map((description) => [description]);

    // Insert descriptions within the transaction
    connection.query(sql, [valuesToInsert], (err, result) => {
      if (err) {
        console.error('Error adding descriptions:', err);

        // Rollback the transaction in case of an error

```

```

        connection.rollback(function () {
            console.error('Transaction rolled back.');
```

```

            res.status(500).json({ error: 'Failed to add descriptions.' });
        });
    } else {
        // Commit the transaction if there are no errors
        connection.commit(function (err) {
            if (err) {
                console.error('Error committing transaction:', err);
                connection.rollback(function () {
                    console.error('Transaction rolled back.');
```

```

                    res.status(500).json({ error: 'Failed to commit transaction.' });
                });
            } else {
                console.log('Descriptions added successfully.');
```

```

                res.status(201).json({ message: 'Descriptions added successfully.' });
            }
        });
    }
});
});
});

```

```

app.get('/product/variants/:productName', (req, res) => {
    const { productName } = req.params;
    const sql = 'SELECT variant FROM productview WHERE productName = ?';

    connection.query(sql, [productName], (err, results) => {
        if (err) {
            console.error('Error fetching variants:', err);
            res.status(500).json({ error: 'Failed to fetch variants.' });
        } else {

```

```

        const variants = results.map((result) => result.variant);
        res.status(200).json({ variants });
    }
});
});

//product list info
import React, { useEffect, useState } from 'react';
import { FaTrash, FaPlus, FaShoppingCart } from 'react-icons/fa';
import AddProductForm from './AddProductForm';
import PurchaseForm from './PurchaseForm';
import AddVarientForm from './AddVarientForm';
import AddVarientValue from './AddVarientValue';
import sucessimage from '../images/sucess2.jpeg';
import './Style.css';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { faPlus } from '@fortawesome/free-solid-svg-icons';
import CustomAlert from './CustomAlert';

function ProductTable() {
    const [data, setData] = useState([]);
    const [showForm, setShowForm] = useState(false);
    const [showPurchaseForm, setShowPurchaseForm] = useState(false);
    const [purchasedProduct, setPurchasedProduct] = useState("");
    const [addvarientvalue, sethandleAddVarientValue] = useState("");
    const [variantsData, setVariantsData] = useState([]);
    const [selectedProductId, setSelectedProductId] = useState(null);
    const [showAddVarientValueForm, setShowAddVarientValueForm] = useState(false);
    const [selectedVariants, setSelectedVariants] = useState([]);
    const [deleteConfirmation, setDeleteConfirmation] = useState(false);
    const [showCustomAlert, setShowCustomAlert] = useState(false);
    const [userConfirmedDeletion, setUserConfirmedDeletion] = useState(false);

```

```
useEffect(() => {  
  fetch('http://localhost:3000/assesst')  
    .then((res) => res.json())  
    .then((data) => setData(data))  
    .catch((err) => console.log(err));  
}, []);  
  
const handleToggleForm = () => {  
  setShowForm(!showForm);  
};  
  
const handlePurchaseClick = () => {  
  setShowPurchaseForm(!showPurchaseForm);  
};  
  
const handlePurchaseSubmit = (productName) => {  
  setPurchasedProduct(productName);  
  setShowPurchaseForm(false);  
};  
  
const handleAddVariant = (product) => {  
  setSelectedProductId(product.productid);  
};  
  
const handleAddVariantValueClick = () => {  
  setShowAddVariantValueForm(!showAddVariantValueForm);
```

```

    }

    const handleAddVariantValueSubmit = (productName) => {
      sethandleAddVariantValue(productName);
      setShowAddVariantValueForm(false);
    };

    const handleDeleteVariant = (productId) => {
      // Fetch variants for the selected product
      if (productId !== null) {
        fetch(`http://localhost:3000/variants1/${productId}`)
          .then((res) => res.json())
          .then((data) => {
            setVariantsData(data);
            setSelectedVariants(data.map((variant) => variant.id));
            setDeleteConfirmation(true);
            console.log('Fetched variants:', data);
            console.log('Variant IDs:', selectedVariants);
          })
          .catch((err) => console.log(err));
      }
    };

    const handleVariantAdded = () => {
      // Fetch updated variants data for the selected product
      if (selectedProductId !== null) {
        fetch(`http://localhost:3000/variants/${selectedProductId}`)
          .then((res) => res.json())
          .then((data) => {
            setVariantsData(data);
          })
          .catch((err) => console.log(err));
      }
    }
  }

```

```

    // Variant added, clear the selection
    setSelectedProductId(null);
};

const handleCheckboxChange = (variantId) => {
  if (selectedVariants.includes(variantId)) {
    // Deselect the variant
    setSelectedVariants(selectedVariants.filter((id) => id !== variantId));
  } else {
    // Select the variant
    setSelectedVariants([...selectedVariants, variantId]);
  }
};

const handleDeleteConfirmation = () => {
  if (selectedVariants.length === 0) {
    return; // No variants selected for deletion
  }

  setShowCustomAlert(true);
};

const handleDeleteOperation = () => {
  // Log the selected variants before sending the request
  console.log('Selected Variants:', selectedVariants);

  // Send a request to delete the selected variants
  fetch('http://localhost:3000/variants/delete', {
    method: 'DELETE',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({ variantIds: selectedVariants }),
  })

```

```

.then((res) => {
  if (res.status === 200) {
    // Successfully deleted, update the UI
    // You can choose to refresh the data or remove the deleted variants from the UI
    setVariantsData(variantsData.filter((variant) =>
!selectedVariants.includes(variant.id)));
    setSelectedVariants([]);
    setDeleteConfirmation(false);
    alert("Varients deleted sucessfully")
  } else {
    // Handle any errors, e.g., display an error message
    console.error('Error deleting variants');
  }
})
.catch((error) => {
  console.error('Error deleting variants:', error);
});

const handleCustomAlert = (confirmed) => {
  if (confirmed) {
    // User confirmed deletion
    setUserConfirmedDeletion(true);
    setShowCustomAlert(false);
    handleDeleteOperation(); // Call your deletion operation here
  } else {
    // User canceled deletion
    setShowCustomAlert(false);
  }
};

//use effect cases

```

```

useEffect(() => {
  if (showPurchaseForm) {
    // Scroll to the element with id "purchase-form" when showPurchaseForm changes
    const purchaseFormElement = document.getElementById('purchase-form');
    if (purchaseFormElement) {
      purchaseFormElement.scrollIntoView({ behavior: 'smooth' });
    }
  }
}, [showPurchaseForm]);

useEffect(() => {
  if (showForm) {
    // Scroll to the element with id "add-product-form" when showForm changes
    const addProductFormElement = document.getElementById('add-product-form');
    if (addProductFormElement) {
      addProductFormElement.scrollIntoView({ behavior: 'smooth' });
    }
  }
}, [showForm]);

useEffect(() => {
  if (showAddVarientValueForm) {
    // Scroll to the element with id "add-varient-value-form" when
    showAddVarientValueForm changes
    const addVarientValueFormElement = document.getElementById('add-varient-value-
      form');
    if (addVarientValueFormElement) {
      addVarientValueFormElement.scrollIntoView({ behavior: 'smooth' });
    }
  }
}, [showAddVarientValueForm]);

return (

```



```

<div className="p-4 bg-gray-100"
style={{
  backgroundImage: `url(${sucessimage})`,
  backgroundSize: 'cover',
}}
>
<div className="flex justify-between items-center mb-4">
  <button
    className="bg-green-500 hover:bg-green-700 text-white font-bold py-2 px-4
rounded"
    onClick={handlePurchaseClick}
  >
    <FaShoppingCart className="mr-2" /> Purchase
  </button>
  <h2 className="text-2xl font-bold text-center">Product List</h2>
  <button
    className="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4
rounded"
    onClick={handleToggleForm}
  >
    <FaPlus className="mr-2" /> Add New Product
  </button>
</div>
<table className="border-collapse border w-full bg-black mt-7">
  <thead>
    <tr className="bg-yellow-100 text-red-500 text-lg uppercase">
      <th className="border p-2">ID</th>
      <th className="border p-2">PRODUCT</th>
      <th className="border p-2">ADD Variants</th>
      <th className="border p-2">DELETE Variants</th>
    </tr>
  </thead>

```

```

<tbody>
  { data.map((d, i) => (
    <tr key={i} className="border text-black uppercase text-center">
      <td className="border p-2 text-white">{ d.productid}</td>
      <td className="border p-2 text-white">{ d.productName}</td>
      <td className="border p-2">
        <button
          className="bg-green-500 hover:bg-blue-700 text-white font-bold py-2 px-4
rounded"
          onClick={() => handleAddVariant(d)}
        >
          <FontAwesomeIcon icon={faPlus} className="mr-2" />
          Add New variant
        </button>

      </td>
      <td className="border p-2 flex justify-center items-center">
        <button className="bg-red-500 hover:bg-red-700 text-white font-bold py-2
px-4 rounded flex items-center align-center"
          onClick={() => handleDeleteVariant(d.productid)}>
          <FaTrash className="mr-2" />
          Delete Variants
        </button>
      </td>

    </tr>
  )})
</tbody>
</table>
<br />

```


 Click "Add Variants" for the suitable product you want.


```
{ showForm && <AddProductForm /> }
{ showPurchaseForm && <PurchaseForm onSubmit={handlePurchaseSubmit} /> }
{ showAddVariantValueForm && <AddVariantValue
onSubmit={handleAddVariantValueSubmit} /> }
```

```
{ selectedProductId !== null && (
  <AddVariantForm
    productId={selectedProductId}
    onVariantAdded={handleVariantAdded}
  />
)}
```

```
))
```

```
{ deleteConfirmation && (
```

```
  <div className="mt-4 bg-gradient-to-r from-yellow-600 via-blue-500 to-green-400
p-4 rounded shadow">
```

```
    <h2 className="text-xl text-black font-semibold mb-4 uppercase font-bold">
```

```
      Delete Variants
```

```
    </h2>
```

```
    { variantsData.length > 0 ? (
```

```
      <div>
```

```
        { variantsData.map((variant) => (
```

```
<label key={variant} className="block mt-2 text-black">
```

```
  <input
```

```
    type="checkbox"
```

```
    checked={selectedVariants.includes(variant)}
```

```
    onChange={() => handleCheckboxChange(variant)}
```

```
    className="mr-2 text-black-500"
```

```
  />
```

```
    { variant}
```

```
</label>
```

```
)))
```

```
  <button
```

```

        className="bg-red-500 hover:bg-red-700 text-white font-bold py-2 px-4
rounded"
        onClick={handleDeleteConfirmation}
      >
        Delete Selected Variants
      </button>
    </div>
  ) : (
    <p className="text-red-700">No variants found for deletion.</p>
  )}
</div>
)}
{showCustomAlert && (
<CustomAlert
  message="Do you want to proceed with the deletion?"
  onOK={() => handleCustomAlert(true)}
  onCancel={() => handleCustomAlert(false)}
/>
)}

<div id="purchase-form"></div>
<div id="add-product-form"></div>
<div id="add-varient-value-form"></div>
</div>
);
}
export default ProductTable;

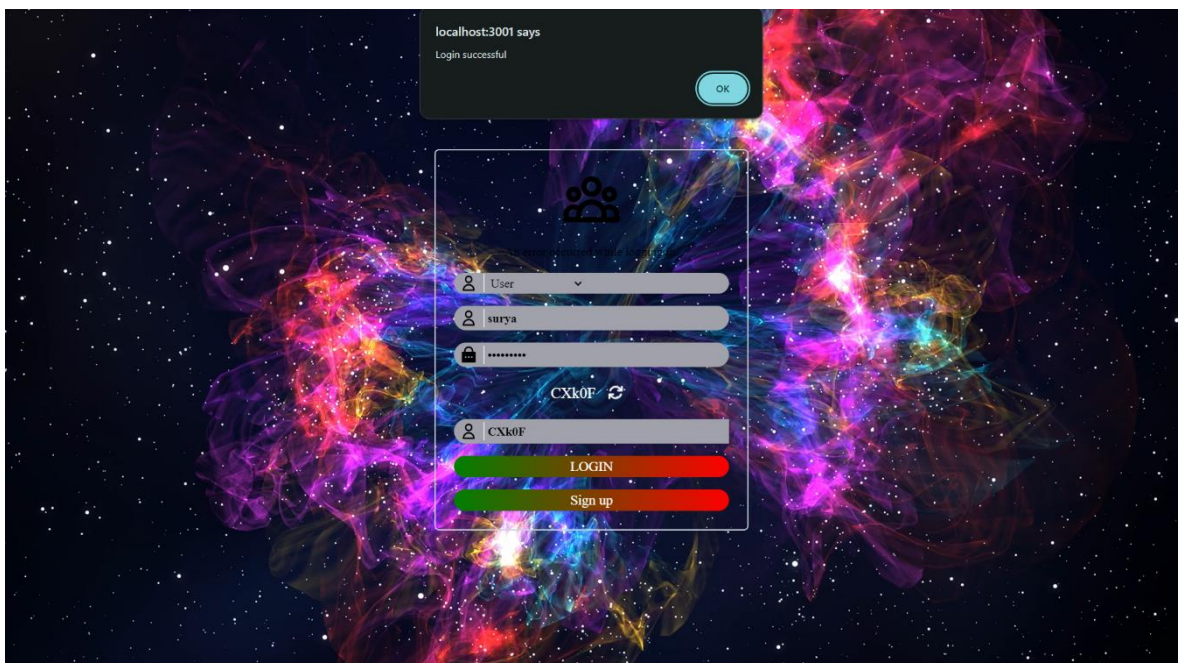
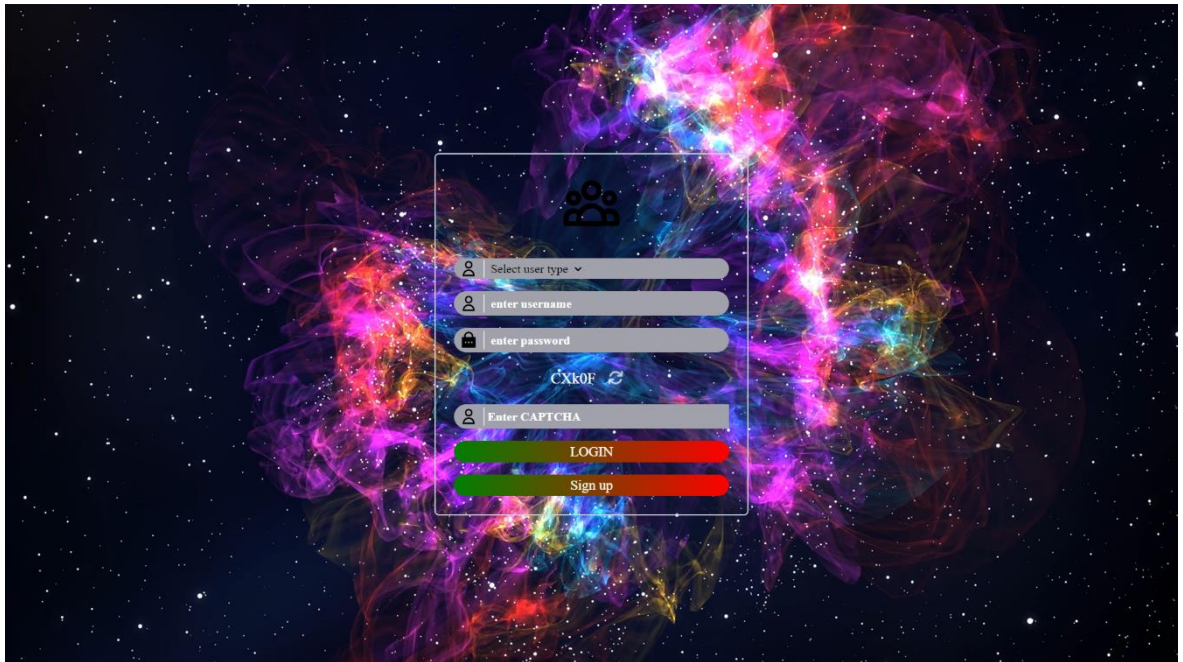
```

4.3 Testing And Methodology

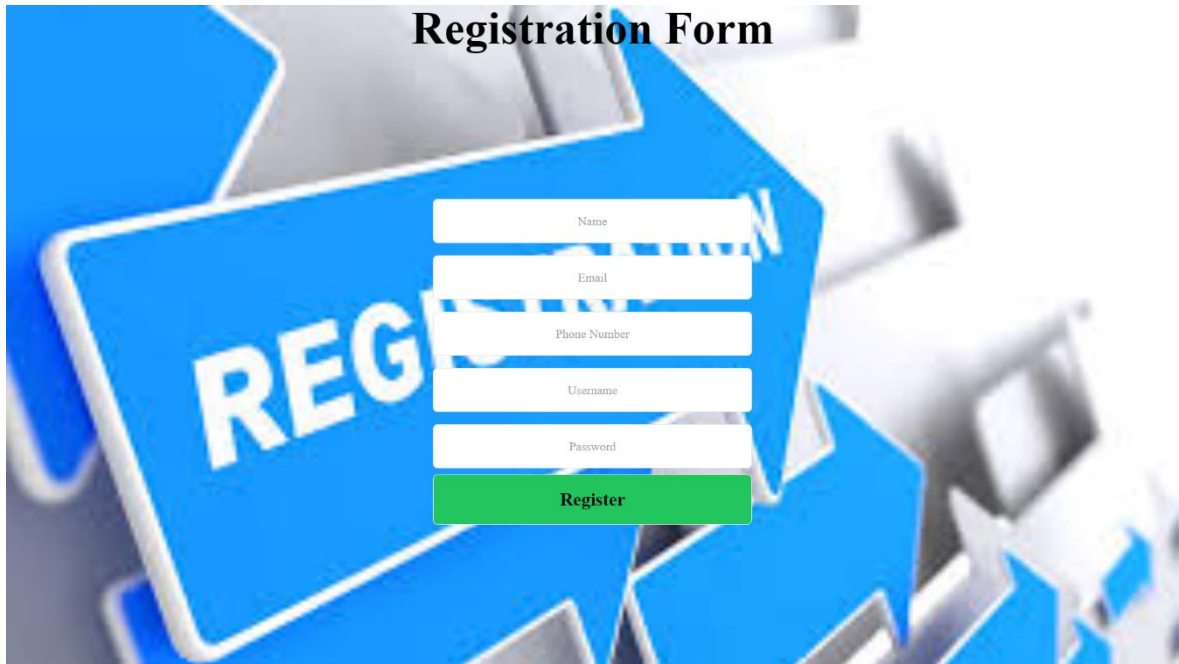
Testing is an important part of the development process for an Asset management system, as it ensures that the system is working correctly and meets the requirements of its users.

1. **Unit testing:** First we start testing separate components to ensure that they work as expected.
2. **Integration testing:** Later we start testing different components of the system to work together as a whole. In this, we include testing how the user interface interacts with the database
3. **Overall testing:** finally we deploy the total project in the localhost server and we solve all the vulnerabilities and bugs

4.4 Result

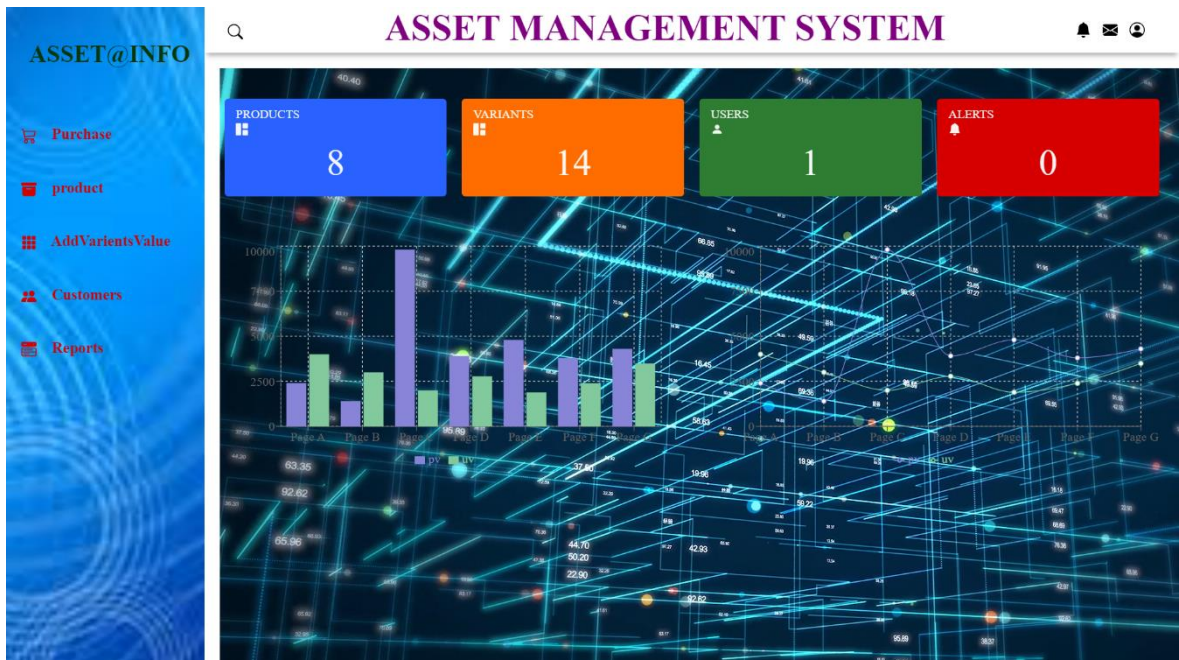


Registration Form



A registration form overlaid on a background of blue 3D arrows pointing right. The form consists of five white input fields stacked vertically, followed by a green 'Register' button. The input fields are labeled: Name, Email, Phone Number, Username, and Password. The word 'REGISTER' is partially visible on one of the background arrows.

Name
Email
Phone Number
Username
Password
Register



Purchase

Product List

Add New Product

ID	PRODUCT	ADD VARIANTS	DELETE VARIANTS
2	LAPTOP	+ Add New variant	Delete Variants
3	PROJECTOR	+ Add New variant	Delete Variants
4	AC	+ Add New variant	Delete Variants
5	BULB	+ Add New variant	Delete Variants
12	UPS	+ Add New variant	Delete Variants
18	FAN	+ Add New variant	Delete Variants
19	CHAIR	+ Add New variant	Delete Variants
20	CPU	+ Add New variant	Delete Variants

CLICK "ADD VARIANTS" FOR THE SUITABLE PRODUCT YOU WANT.

Purchase

Product List

Add New Product

ID	PRODUCT	ADD VARIANTS	DELETE VARIANTS
2	LAPTOP	+ Add New variant	Delete Variants
3	PROJECTOR	+ Add New variant	Delete Variants
4	AC	+ Add New variant	Delete Variants
5	BULB	+ Add New variant	Delete Variants
12	UPS	+ Add New variant	Delete Variants
18	FAN	+ Add New variant	Delete Variants
19	CHAIR	+ Add New variant	Delete Variants
20	CPU	+ Add New variant	Delete Variants

PRODUCT YOU WANT.

SEARCH

Product Name:

PURCHASE PRODUCT

PRODUCT NAME:

PRODUCT VARIANTS

BULB WATT:

BULB TYPE:

ADD NEW PRODUCT

Product Name

MANAGEMENT REPORT

ID	PRODUCT NAME	VARIANT	VALUE
2	LAPTOP	CPU	INTEL I7, INTEL I5, INTEL I3
		ram	16, 8, 4
		storage	512, 1 TB, 256 GB
		Model	pavallion, 2018, inspiron 7071
		Brand	hp, Dell
3	PROJECTOR	LENS TYPE	MICRO
		lens brand	ultra
		projector Brand	xerrhon
5	BULB	BULB WAITT	5, 5
		bulb type	led, led
12	UPS	MAKE	COPPER
		capacity	230volt
19	CHAIR	CHAIRTYPE	WOOD, IRON, PLASTIC ROUNDED
		chair wieght	1.5kg, 3kg, 5kg

Press **F11** to exit full screen

PRODUCT REPORT

ID	PRODUCT NAME
2	LAPTOP
3	PROJECTOR
4	AC
5	BULB
12	UPS
18	FAN
19	CHAIR
20	CPU

Chapter 5

5.1 Conclusion

The Asset Management System project has successfully implemented a user-friendly solution using React.js, Node.js, Express, and MySQL. With modules for user authentication, asset and variant management, and efficient search capabilities, the system provides a comprehensive platform for seamless asset tracking. The deployment-ready application showcases a modular design, data accuracy, and scalability. As we move towards implementation, the Asset Management System is poised to enhance operational efficiency, offering a robust solution to meet the evolving asset management needs of our organization.

CHAPTER 6

6.1 FUTURE ENHANCEMENT

In our roadmap for future enhancements, we prioritize the application with advanced search filters. Another significant upgrade includes the capability to add images to product profiles, providing a visual dimension to asset management. Recognizing the increasing demand for on-the-go access, we are exploring the development of a mobile compatibility, offering users the flexibility to manage assets from their smartphones.

Chapter 7

7.1 References

- [1] Investigation: A Productive Asset Management Web Application Computer Systems Science & Engineering DOI:10.32604/csse.2021.015314
- [2] M Wang, J Tan, Y Li - 2015 IEEE international conference on ..., 2015 - ieeexplore.ieee.org
- [3] Markus Keinänen Creation of a web service using the MERN stack
- [4] D Sarkar, H Patel, B Dave - ... Journal of Construction Management, 2022 - Taylor & Francis
- [5] L Turnip, A Triayudi , ID Solihati - Jurnal Mantik , 2020 - iocscience.org
Asset Management Information System Using the Waterfall Method (Case Study: National University)

