

# **PURCHASE AUTOMATION SYSTEM**

## **Main Project Report**

Submitted by

**SRUTHY E**

**Reg no:FIT20MCA-2108**

*Submitted in partial fulfillment of the requirements for the award of the*

*degree of*

***Master of Computer Applications***

***Of***

***A P J Abdul Kalam Technological University***



**FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)®**

**ANGAMALY-683577, ERNAKULAM(DIST)**

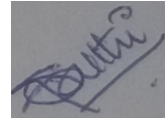
**JULY 2022**

# **DECLARATION**

I declare that the report of this project work, submitted to the Department of Computer Applications, Federal Institute of Science and Technology (**FISAT**), Angamaly in partial fulfillment of the award of the degree of Master of Computer Applications is an authentic record of my original work.

The report has not been submitted for the award of any degree of this university or any other university.

**Date :**



**Place: Angamaly**


**SRUTHY E**

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)®  
ANGAMALY, ERNAKULAM-683577




**CERTIFICATE**

This is to certify that the project report titled " **PURCHASE AUTOMATION SYSTEM** " submitted by **SRUTHY E, (Reg No: FIT20MCA-2108 )** towards partial fulfillment of the requirements for the award of the degree of Master of Computer Applications is a record of bonafide work carried out by her during the year 2022.

  
**Project Guide**  
**Ms.Senu Abi**



**Head of the Department**  
**Dr. Deepa Mary Mathews** 

*Submitted for the viva-voce held on ..... at .....*

**Examiner :**



# FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)<sup>®</sup>

(An ISO 9001:2015 Certified, NAAC ('A' Grade) Accredited Institution with NBA Accredited Programmes  
(Approved by AICTE – Affiliated to APJ Abdul Kalam Technological University, Kerala)

Owned and Managed by Federal Bank Officers' Association Educational Society

HORMIS NAGAR, MOOKKANNOOR P.O., ANGAMALY - 683 577, ERNAKULAM DT., KERALA, S. INDIA.

Tel: (O) 0484-2725272 Fax: 0484 – 2725250 E-mail: mail@fisat.ac.in Website: www.fisat.ac.in

11<sup>TH</sup> July 2022

## TO WHOMSOEVER IT MAY CONCERN

This is to certify that Mr./Ms. SRUTHY E (Reg. No. FIT20MCA-2108) has successfully completed his/her Main Project with the title "Purchase Automation System", in the Department of Computer Applications, FISAT, during the period from 30<sup>th</sup> March 2022 to 11<sup>th</sup> July 2022.

Dr DEEPA MARY MATHEWS  
HEAD OF THE DEPARTMENT



## ACKNOWLEDGEMENT

Gratitude is a feeling which is more eloquent than words, more silent than silence. To complete this project work I needed the direction, assistance and co-operation of various individuals, which is received in abundance with the grace of God.

I hereby express my deep sense of gratitude to **Dr. Manoj George** Principal, FISAT and **Dr. C Sheela** Vice principal, FISAT for allowing me to utilize all the facilities of the college.

My sincere thanks to **Dr. Deepa Mary Mathews** HOD, Department of Computer Applications FISAT ,who had been a source of inspiration. I express heartiest thanks to **Ms.Senu Abi** my internal guide for her encouragement and valuable suggestion. I express my heartiest gratitude to my scrum master **Ms. Joice T** and the faculty members in my Department for encouragement and constructive suggestions and comment during the project work.

Finally I wish to express my thanks to my parents, friends and well-wishers who extended their help in one way or other in preparation of my project. Besides all, I thank GOD for everything.

## **ABSTRACT**

The project Inventory Management System is a complete web based application. An inventory management system is the combination of technology (hardware and software), processes and procedures that oversee the monitoring and maintenance of stocked products, whether those products are assets, raw materials and supplies, or finished products ready to be sent to vendors or end consumers. Inventory Management System plays an important role because it reduces the stress, monitoring of products, making balance sheets and many more which was done manually. Simply Inventory Management System overtook the manual things and also it optimizes the cost and time constraint.

Purchase Automation System is an inventory management system which can be used for automating the purchase section in the college. It can help the overall purchase activity in the college. Inventory management software is a software system for tracking inventory levels, orders, sales and deliveries. It can also be used in the manufacturing industry to create a work order, bill of materials and other production-related documents. It helps to manage store products and keep track of all goods stock and also we can use it to check the purchases history of the store. Basically, I'm developing a system to manage/automate some processes of any retail store by using the computer. So by using this system, it can keep record data of products and even all purchases happening in-store. This system is mainly used for reducing the manual work in the college and make it as system process. It is a computerized process and all the transactions will be happened through this. It helps the authorities for balancing the total stock during the end of the year.

Now-a-days, lots of paper works are happening, it leads the works much more complicated. So, this proposed system helps to reduce the paper works and errors in it. There are so many advantages while making the system as an automated one. It can reduce the stress, increased production output, consistent and improved part production and quality etc.. Proposed system will definitely help the user and it is user friendly for the college.

# Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>8</b>
<b>2</b>	<b>PROOF OF CONCEPT</b>	<b>9</b>
2.1	Existing System . . . . .	9
2.2	Proposed System . . . . .	9
2.3	Objectives . . . . .	10
<b>3</b>	<b>IMPLEMENTATION</b>	<b>11</b>
3.1	Software Requirements: . . . . .	11
3.2	Technologies Used . . . . .	11
3.2.1	HTML . . . . .	11
3.2.2	Python . . . . .	12
3.2.3	JavaScript . . . . .	13
3.2.4	Flask . . . . .	13
3.2.5	PostgreSQL . . . . .	14
3.3	Database Design . . . . .	15
3.3.1	Database tables . . . . .	15
3.4	MODULES . . . . .	21
3.4.1	Administration Management . . . . .	21
3.4.2	Request Management Module . . . . .	21
3.4.3	Purchase Management . . . . .	22
<b>4</b>	<b>RESULT ANALYSIS</b>	<b>23</b>

---

<b>5</b>	<b>CONCLUSION AND FUTURE SCOPE</b>	<b>27</b>
5.1	Conclusion . . . . .	27
5.2	Future Scope . . . . .	27
<b>6</b>	<b>APPENDIX</b>	<b>28</b>
6.1	Source Code . . . . .	28
6.1.1	app.py . . . . .	28
6.1.2	connection.py . . . . .	38
6.1.3	purchase.js . . . . .	38
6.1.4	style.css . . . . .	53
6.1.5	purchase.py . . . . .	54
6.2	Screen Shots . . . . .	60
<b>7</b>	<b>REFERENCES</b>	<b>65</b>



# Chapter 1

## INTRODUCTION

The project “Purchase Automation System” is a live project used to help the college for the automating the system. Its mainly focused on the stock of the college and to manage store products and keep track of all goods stock and also it can use it to check the purchases history of the store. Basically, try to developing a system to manage/automate some processes of any retail store by using the computer. So by using this system it can keep record data of products and even all purchases happened in-store.

Now-a-days, all processes are done by manually. So, there is a lot of errors happening inside it. However, all the works are going to be automated and being computerized. So, it can be fetched the data easily from each and every field. The proposed system is computerized and has been developed using python language therefore it gives more facilities than present system. It provides quick access to any data.

This project has more than twenty tables. And there are sixteen common tables for the each tables. All are connected with each other. All tables consists of unique key that makes the table individual. In IMS, there are mainly four modules which are purchase management system, repair management system, fuel management system, food and beverage management system. I'm doing the purchase module, which can be used for collecting the stock details from each department, maintain a purchase level for knowing that, in which time the organisation need to purchase their stock, maintain a code for each department to track the required item for that department. All the purchasing section are happening under this module and its being an automated process.

# **Chapter 2**

## **PROOF OF CONCEPT**

### **2.1 Existing System**

In the existing system, when the required stock is not available in the warehouse. The officials might buy it from store outside and may not be included in the balance stock record. It is very difficult to manage the small small transaction for the officials. Also its a manual process and difficult for debugging the errors. In present non-computerized system all jobs are performed manually. All records of customer, and customers details are written and stored in paper format in a file. Authorities maintains separate files for each new information. So managing them is very complex and has become an impossible task to be performed manually systematic way. It is difficult to maintain important information in books. More manual hours need to generate required report. It is tedious to manage historical data which needs much space to keep all the previous years ledgers, books etc. Daily transactions are to be entering into different books immediately to avoid conflicts which are very difficult. To overcome these limitations, i introduced proposed system for my project

### **2.2 Proposed System**

The proposed system is computerized and has been developed using advance language therefore it gives more facilities than present system. It provides quick access to any

data.every thing that has been bought for use in departments and offices whether it be from the warehouse or outside shops will be recorded in the inventory. The stock should always be available whenever each department needs.Its an automated process. Proposed system is a software application which avoids more manual hours that need to spend in record keeping and generating reports. This application keeps the data in a centralized way which is available to all the users simultaneously. It is very easy to manage historical data in database. No specific training is required for the employees to use this application. They can easily use the tool that decreases manual hours spending for normal things and hence increases the performance.

## **2.3 Objectives**

The primary purpose of inventory management is to ensure there is enough goods or materials to meet demand without creating overstock, or excess inventory. This is a project for the college to maintain the stock in every department. Its main aim is to help the officers to handle the stock easily and they can report it to the nodal officer about the required stock that needed for the department.

# Chapter 3

## IMPLEMENTATION

### 3.1 Software Requirements:

- Database: PostgreSQL
- Front end: HTML, JavaScript
- Back end: Python

### 3.2 Technologies Used

#### 3.2.1 HTML

The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets and scripting languages such as JavaScript. HTML is an acronym which stands for Hyper Text Markup Language which is used for creating web pages and web applications. HyperText simply means "Text within Text." A text has a link within it, is a hypertext. Whenever you click on a link which brings me to a new webpage, you have clicked on a hypertext. HyperText is a way to link two or more web pages (HTML documents) with each other. A markup language is a computer language that is used to apply layout and formatting conventions to a text document. Markup lan-

guage makes text more interactive and dynamic. It can turn text into images, tables, links, etc.

A web page is a document which is commonly written in HTML and translated by a web browser. A web page can be identified by entering an URL. A Web page can be of the static or dynamic type. With the help of HTML only, can create static web pages. In this project, HTML is used for creating the user interface .It helps me to create the UI pages easily and user friendly. There are totally morethan five UI pages. Each and every page is different from another. CSS helps me to provide different styles for each web pages.

### **3.2.2 Python**

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.

Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace.

A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective

power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Python is more comfortable than another programming languages. It is easy understand and reduces the code complexity. So, I selected python as the programming language for my whole project.

### **3.2.3 JavaScript**

Script was originally developed to offer dynamic control over the various elements within an HTML document, and that is still its main use. But more and more, JavaScript is being used for Ajax. This is a term for the process of accessing the web server in the background. (It originally meant “Asynchronous JavaScript and XML,” but that phrase is already a bit outdated.)

Ajax is the main process behind what is now known as Web 2.0 in which web pages have started to resemble standalone programs, because they don’t have to be reloaded in their entirety. Instead, a quick Ajax call can pull in and update a single element on a web page, such as changing your photograph on a social networking site or replacing a button that you click with the answer to a question.

### **3.2.4 Flask**

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. Flask is a web development framework developed in Python. It is easy to learn and use. Flask is “beginner-friendly” because it does not have boilerplate code or dependencies, which can distract from the primary function of an application.

Flask originated in 2004 when a developer named Armin Ronacher created it as an

April Fool's joke. However, it quickly gained popularity in the open-source community anyway. Consequently, it developed into a popular open-source project and gained a massive following, which it maintains today. Flask provides a development server and a debugger.

### **3.2.5 PostgreSQL**

PostgreSQL is an advanced, enterprise class open source relational database that supports both SQL (relational) and JSON (non-relational) querying. It is a highly stable database management system, backed by more than 20 years of community development which has contributed to its high levels of resilience, integrity, and correctness. PostgreSQL is used as the primary data store or data warehouse for many web, mobile, geospatial, and analytics applications. The latest major version is PostgreSQL 12. PostgreSQL has a rich history for support of advanced data types, and supports a level of performance optimization that is common across its commercial database counterparts, like Oracle and SQL Server. AWS supports PostgreSQL through a fully managed database service with Amazon Relational Database Service (RDS). Amazon Aurora with PostgreSQL compatibility is also built using PostgreSQL.

Postgresql is ease for creating the tables,schemas etc by using pgadmin tool.Its very comfortable for the user.Pgadmin4 helps the user to use it as user friendly.pgAdmin4 is a popular application to manage Postgres databases. All types of PostgreSQL features are supported by this application. The three main parts of pgAdmin4 client are pgAdmin menu bar, pgAdmin tree control and tabbed browser control. Each part is used to do different types of management tasks.so I used this tool in my project for table creation.

### 3.3 Database Design

Database design can be generally defined as a collection of tasks or processes that enhance the designing, development, implementation, and maintenance of enterprise data management system. Designing a proper database reduces the maintenance cost thereby improving data consistency and the cost-effective measures are greatly influenced in terms of disk storage space. Therefore, there has to be a brilliant concept of designing a database. The designer should follow the constraints and decide how the elements correlate and what kind of data must be stored.

The main objectives behind database designing are to produce physical and logical design models of the proposed database system. To elaborate this, the logical model is primarily concentrated on the requirements of data and the considerations must be made in terms of monolithic considerations and hence the stored physical data must be stored independent of the physical conditions. On the other hand, the physical database design model includes a translation of the logical design model of the database by keep control of physical media using hardware resources and software systems such as Database Management System (DBMS).

#### 3.3.1 Database tables

A table is a collection of related data held in a table format within a database. It consists of columns and rows. In relational databases, and flat file databases, a table is a set of data elements using a model of vertical columns and horizontal rows, the cell being the unit where a row and column intersect. These are the tables used in this project :

- **stkpurchaserequisitionMaster** :This is the main table of this module.Purchase automation master uses a unique requisition id that is the primary key for that table and foreign key for the another table.It collects the details such as date of purchase,current stock,outstanding,cost etc.
- **stkpurchaserequisitiondetails** :It is used for collecting the details of the purchased



item with its order number. It uses a unique ID, named requisitiondetails ID, that is the unique key used in this table. By using this unique, data can be fetched easily.

- **stkpurchaserequisitionprocess** : This table is used for processing all the details from the master table and the details table. The data from this table is passed for further processing.
- **stkRequestProcessMaster** : In this table, the field requestprocessid is the primary key. It should contain the details such as the request name, the name of the process, the order no, action details. The field process name is used for checking the status whether the intent is approved/rejected/recommended etc.
- **stkRequestProcessMaster** : In this table, requestprocessid is the primary key. It should contain the details such as the request name, the name of the process, the order no, action details. The field process name is used for checking the status whether the intent is approved/rejected/recommended etc.
- **stkinstitution** : In this table there are two fields institutionid, institutionname. Institutionid is the unique id that is given to the table. The field Institutionname field is used for providing the name of the particular institution.
- **stkdepartment** : In this table there are four fields departmentid, departmentcode, departmentname, employeeid. The field departmentid is the unique id that is given to the table. departmentname and departmentcode field is used to identify the particular institution based on the information given. The attribute employeeid is the foreign key that is referred to the employee table.
- **stkuserlevel** : In this table there are four fields named userlevelid, userlevelcode, userlevel, employeeid. The field userlevelid is the primary key and employeeid is the foreign key which is referred to employee table.
- **stkemployee** : This table stores the employee information like employeename, employeeecode. employeeid is the unique id which is given to the table.

- **stkloginmaster** : Loginmaster table contains the login information like id,password.Attributes like Userlevelid, employeeid are the foreign keys in this table.
- **stkitemmaster** : This table contains ten fields, in that there is four fields which are referred to other tables. itemmasterid, itemmastercode, itemnameid, itemtypeid, brandid, modelid, unitid, subunitid, purchaseorderlevel, currentstock.The field such as itemmasterid is the unique id of this table.The attributes like brandid, modelid, unitid, subunitid are the four foreign keys.
- **stkitems** : Item table contains five fields to store the item details like itemcode, itemname. The field such as itemsid is the unique id of item table.Another fields such as itemtypeid, unitid is the two foreign keys in this table.
- **stkstkitemtype** : The field such as itemtype table includes three fields itemtypeid, itemtypecode, itemtype. The attribute like itemtypeid is the unique id of the table. Another attribute like item code and item type specifies the type and code of the corresponding item.
- **stkitemsubunitquantity** : In this table there are four fields named subunitid, subunitcode, subunitname, unitid.The field subunitid is the unique id and unitid is the foreign key.
- **stkmodel** : In this table there are four fields named modelid, modelcode, modelname, brandid. The field modelid,is the unique id and brandid is the foreign key.
- **stkbrand** : Brand table contains four fields brandid, brandcode, brandname, itemsid.The field brandid, is the primary key and itemsid is the foreign key.
- **stkunit** : Unit table contains three fields unitid, unitcode, unitname. Another field of this table is unitid, and it is the unique id of this table.

Column Name	Data Type
requisitionid	int(pk)
date	date
deptid	int
totalestimated <sub>amount</sub>	numeric(12,2)
budget amount	numeric(12,2)
outstanding	numeric(12,2)
current balance	numeric(12,2)
justification	varchar(500)
currentprocessstatus	varchar(50)
status date	date

Table 3.1: Purchase Requisition Master Table

Column Name	Data Type
requisitiondetails id	int(pk)
requisition id	int(fk)
item id	int(fk)
custodian id	int(fk)
quantity	numeric(12,2)
current status	varchar(500)
justification	varchar(500)
subsection id	int
subsection budget	numeric(12,2)
subsection outstanding	numeric(12,2)

Table 3.2: Purchase Requisition Details Table

Column Name	Data Type
requisitionprocess id	int(pk)
requisition id	int(fk)
requisitiondetails id	int(fk)
process <sub>serialno</sub>	int
transaction	varchar(50)
transactionfrom	int(fk)
transactionto	int(fk)
transactiontodesg	int(fk)
justification	varchar(500)
remarks	varchar(500)
current	int

Table 3.3: Purchase Requisition Process Table

Column Name	Data Type
requestprocess id	int(pk)
requestname	varchar(100)
processname	varchar(100)
orderno	int

Table 3.4: Request Process Master Table

Column Name	Data Type
requestprocessdetails id	int(pk)
requestprocess id	int(fk)
department id	int(fk)
staffid	int(fk)
assignmentid	int(fk)
staffassigned from	date
staffassigned to	date
whether current	varchar(100)

Table 3.5: Request Process Details Table

Column Name	Data Type
budget id	int(pk)
department id	int(fk)
financialyear	int
totalamount	numeric(12,2)
balance	numeric(12,2)
outstanding	numeric(12,2)

Table 3.6: Budget Table

Column Name	Data Type
budgetsection id	int(fk)
budget id	int(pk)
budgetsection name	varchar(500)
totalamount	numeric(12,2)
balance	numeric(12,2)
outstanding	numeric(12,2)

Table 3.7: Budget Section Table

## 3.4 MODULES

Inventory management system helps to determine the right amount of stock to keep on-hand to fill demand while avoiding spending too much on inventory storage. In this module there is a code for tracking the purchase level for each department. Each and every department must ensure the order level and the current stock in that department, if it reaches to the purchase order level it is urgent for keeping the stock in the warehouse. This is an automated system that's why the college can purchase before meeting the corresponding level. Mainly there are three tables for this module: purchaserequisitionmaster, purchaserequisitiondetails, purchaserequisitionprocess. There is an unique id for each table.

### 3.4.1 Administration Management

The authorities in the college like Administrative Officer, Principal etc can login and check whether the item is sanctioned or not. Admin can also check the current status of the purchased item.

### 3.4.2 Request Management Module

Data is to be collected from the each department and checks whether any department needs to purchase any item. Each and every department can request for the item that they

need to purchase. They have to mention the name of the item, date of the item, requirement, justification etc. Then it passes to HOD and verified it and forwarded to the higher authority.

### **3.4.3 Purchase Management**

After forwarding, the authorities can check whether the item is necessary to purchase or not. If it is necessary, they can recommend to their higher authorities. At last, the principal or administrative officer should sanction or reject the intent.

## **Chapter 4**

### **RESULT ANALYSIS**

The project was completed successfully in time. The project was completed successfully in time. This web application helps the college authorities for automating their process. All the process under the college is a manually process, now its an automated process. So, it can be accessed easily.





Figure 4.1: LOGIN PAGE

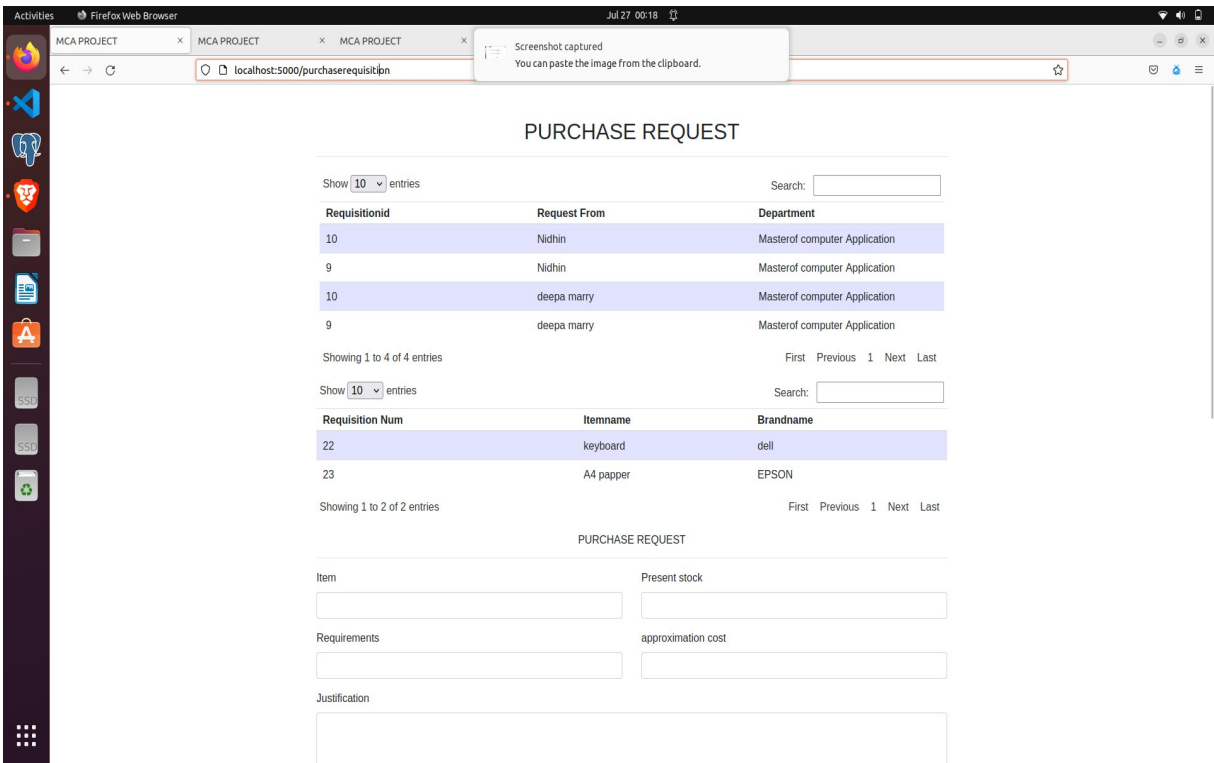


Figure 4.2: PURCHASE REQUISITION PAGE

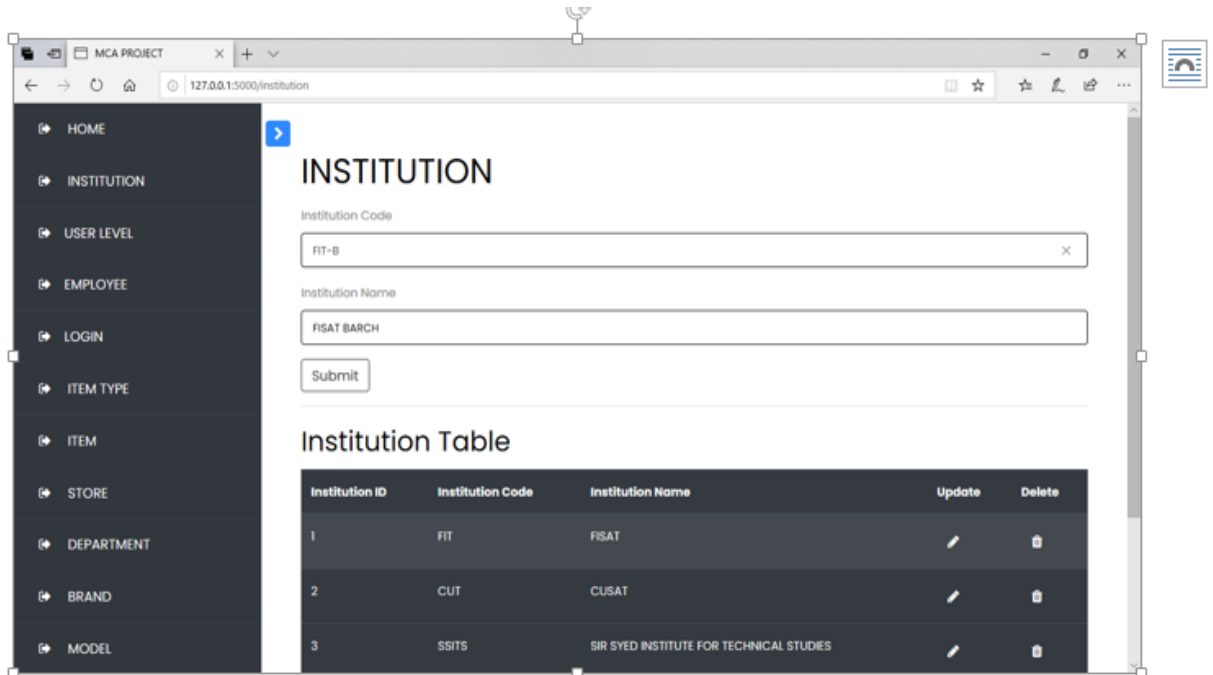


Figure 4.3: INSTITUTION PAGE

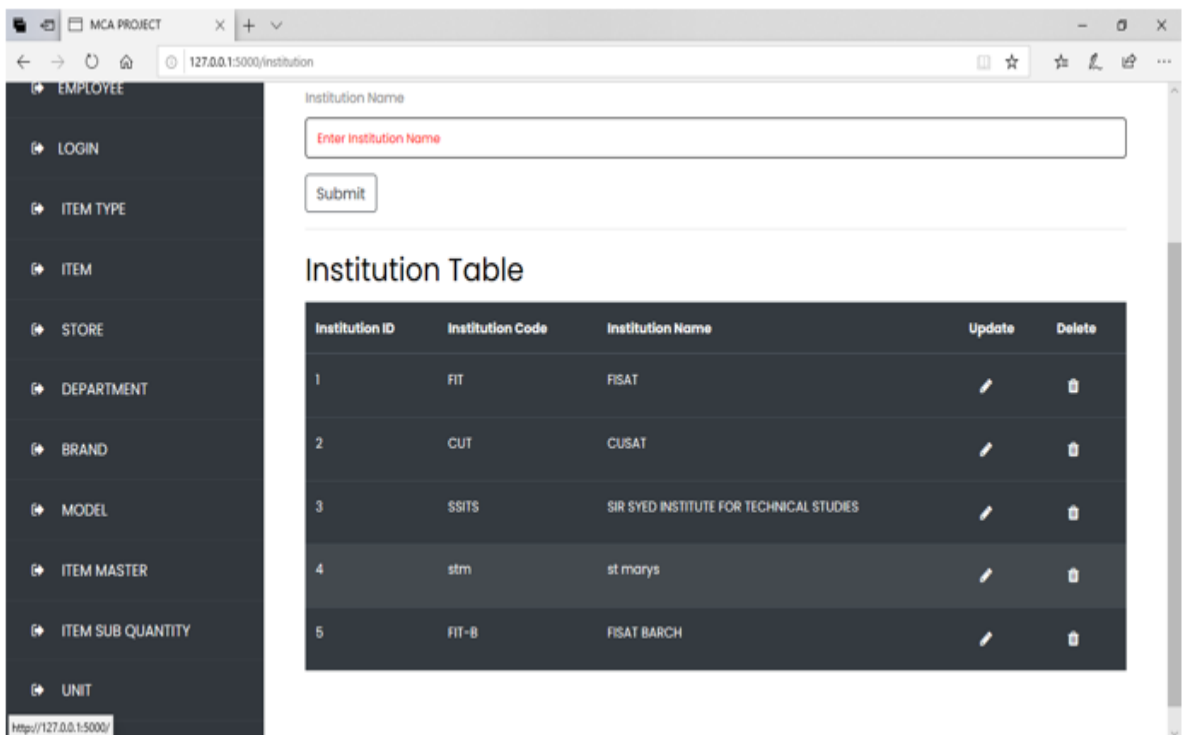
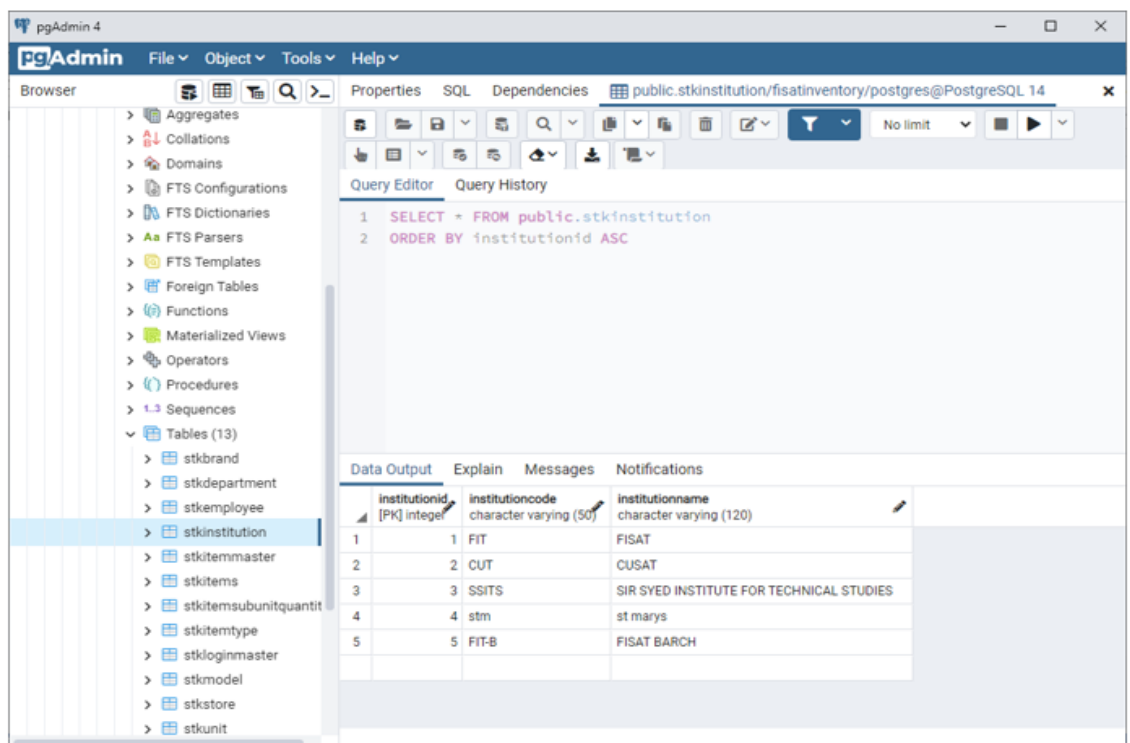


Figure 4.4: INSTITUTION PAGE1



pgAdmin 4

File Object Tools Help

Browser

- Aggregates
- Collations
- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (13)
  - stkbrand
  - stkdepartment
  - stkemployee
  - stkinstitution**
  - stkitemmaster
  - stkitems
  - stkitemsubunitquantit
  - stkitemtype
  - stkloginmaster
  - stkmodel
  - stkstore
  - stkunit

Properties SQL Dependencies public.stkinstitution/fisatinventory/postgres@PostgreSQL 14

Query Editor Query History

```
1 SELECT * FROM public.stkinstitution
2 ORDER BY institutionid ASC
```

Data Output Explain Messages Notifications

institutionid [PK] integer	institutioncode character varying (50)	institutionname character varying (120)
1	1 FIT	FISAT
2	2 CUT	CUSAT
3	3 SSITS	SIR SYED INSTITUTE FOR TECHNICAL STUDIES
4	4 stm	st marys
5	5 FIT-B	FISAT BARCH

Figure 4.5: INSTITUTION PAGE- TABLE DATA

## **Chapter 5**

# **CONCLUSION AND FUTURE SCOPE**

### **5.1 Conclusion**

An inventory Management system is a web-based application used for knowing the stocks that are needed for the college. Purchase Automation System enables all the purchase transactions within it. It helps the college to track the order level. It's a computerized process and an automated one, which can reduce the errors that happened manually. It reduces both paper and time wastage. Totally, this project will smoothen out the purchase activities for various departments of whole college

### **5.2 Future Scope**

The scope of an purchase automation system can cover many needs, including valuing the inventory or purchases, measuring the change in purchase level and planning for future purchased products. The value of the purchase level at the end of each period provides a basis for financial reporting on the balance sheet. Measuring the change in purchase level allows the college to determine the cost of inventory sold during the period. This allows the college to plan for future inventory needs. It become an automated and security web application for doing all the activities in it .The college activities become much more faster through this application.

# Chapter 6

## APPENDIX

### 6.1 Source Code

#### 6.1.1 app.py

```
# from crypt import methods
#from crypt import methods
from flask import Flask , render_template , request , redirect ,
url_for
from operator import getitem , methodcaller
from flask_sqlalchemy import SQLAlchemy

from api.institutionTable import getInstitution , insertInstitution
from api.unitTable import getUnit , insertUnit
from api.itemtypeTable import getitemtype , insertitemtype
from api.itemTable import getItem , insertItem
from api.employeeTable import getEmployee , insertEmployee
from api.userlevelTable import getUserlevel , insertUserlevel
from api.departmentTable import getDepartment , insertDepartment
```

```
from api.brandTable import getBrand,insertBrand
from api.modelTable import getModel,insertModel
from api.subunitTable import getSubunit,insertSubunit
from api.storeTable import getStore,insertStore
from api.itemmasterTable import getItemmaster,insertItemmaster
from api.loginTable import getLoginmaster,insertLoginmaster

app = Flask(__name__)

app.config['SQLALCHEMY_DATABASE_URI'] = 'postgresql:
//postgres:unnimaya@localhost/fisatinventory'
app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] = False
app.secret_key = 'secret string'
db = SQLAlchemy(app)

@app.route("/")
def home():
    return render_template('home.html')

# * INSTITUTION INSERTION AND FETCHING *
@app.route("/institution",methods=['GET','POST'])
def institution():
    if request.method=='GET':
        data=getInstitution()

    if request.method=='POST':
        institution_code=request.form['institution-code']
```

```
institution_name=request.form['institution -name']
print(insertInstitution([institution_code ,institution_name]))
data=getInstitution()

return render_template('institution.html',institutionData=data)

## USERLEVEL INSERTION AND FETCHING *
@app.route("/userlevel",methods=['GET','POST'])
def userlevel():
    if request.method=='GET':
        data=getUserlevel()
        employeeData=getEmployee()
        return render_template('user_level.html',employeeData=
employeeData ,userlevelData=data)

    if request.method=='POST':
        userlevel_code=request.form['userlevel -code']
        userlevel=request.form['user -level']
        employee=request.form['employee']
        employeeData=getEmployee()
        print(insertUserlevel([userlevel_code ,userlevel ,employee]))
        data=getUserlevel()
        print(data)
        return render_template('user_level.html',userlevelData=
data ,employeeData=employeeData)

## DEPARTMENT INSERTION AND FETCHING *
@app.route("/department",methods=['GET','POST'])
def department():
    if request.method=='GET':
```

```
employeeData=getEmployee()
data=getDepartment()
return render_template('department.html',employeeData=
employeeData,departmentData=data)

if request.method=='POST':

    department_code=request.form['department-code']
    department_name=request.form['department-name']
    employee=request.form['employee']
    print(insertDepartment([department_code,department_name,
employee]))
    data=getDepartment()
    print(data)
    return render_template('department.html',departmentData=data)

#*    EMPLOYEE INSERTION AND FETCHING    *
@app.route("/employee",methods=['GET','POST'])
def employee():
    if request.method=='GET':
        data=getEmployee()

    if request.method=='POST':
        employee_code=request.form['employee-code']
        employee_name=request.form['employee-name']
        print(insertEmployee([employee_code,employee_name]))
        data=getEmployee()

    return render_template('employee.html',employeeData=data)
```



```
    ## ITEMS INSERTION AND FETCHING *

@app.route("/items", methods=['GET', 'POST'])
def items():
    if request.method=='GET':
        itemTypeData=getitemtype()
        unitData=getUnit()
        data=getItem()
        return render_template('items.html', itemTypeData=
            itemTypeData, unitData=unitData, itemData=data)

    if request.method=='POST':

        item_code=request.form['item_code']
        item_name=request.form['item_name']
        item_type=request.form['item_type']
        unit=request.form['unit']
        print(insertItem([item_code, item_name, item_type, unit]))
        data=getItem()
        print(data)
        return render_template('items.html', itemData=data)

## ITEMTYPE INSERTION AND FETCHING *

@app.route("/itemtype", methods=['GET', 'POST'])
def itemtype():
    if request.method=='GET':
        data=getitemtype()
```

---

```
if request.method=='POST':
    itemtype_code=request.form['itemtype-code']
    itemtype_name=request.form['itemtype-name']
    print(insertitemtype([itemtype_code,itemtype_name]))
    data=getitemtype()

    return render_template('itemtype.html',itemtypeData=data)

## UNIT INSERTION AND FETCHING *
@app.route("/unit",methods=['GET','POST'])
def units():
    if request.method=='GET':
        data=getUnit()

    if request.method=='POST':
        unit_code=request.form['unit-code']
        unit_name=request.form['unit-name']
        print(insertUnit([unit_code,unit_name]))
        data=getUnit()

    return render_template('units.html',unitData=data)

## BRAND INSERTION AND FETCHING *
@app.route("/brand",methods=['GET','POST'])
def brand():
    if request.method=='GET':
        itemData=getItem()
        data=getBrand()
        return render_template('brand.html',itemData=itemData,
```

```
brandData=data)

if request.method=='POST':
    brand_code=request.form['brand-code']
    brand_name=request.form['brandname']
    item=request.form['items']
    print(insertBrand([brand_code,brand_name,item]))
    data=getBrand()
    print(data)
    return render_template('brand.html',brandData=data)

## MODEL INSERTION AND FETCHING *
@app.route("/model",methods=['GET','POST'])
def model():
    if request.method=='GET':
        brandData=getBrand()
        data=getModel()
        return render_template('model.html',brandData=
brandData,modelData=data)

    if request.method=='POST':

        model_code=request.form['model-code']
        model_name=request.form['model-name']
        brand=request.form['brand']
        print(insertModel([model_code,model_name,brand]))
        data=getModel()
        print(data)
        return render_template('model.html',modelData=data)
```

```
## STORE INSERTION AND FETCHING *
@app.route("/store", methods=['GET', 'POST'])
def store():
    if request.method=='GET':
        institutionData=getInstitution()
        data=getStore()
        return render_template('store.html', institutionData=
            institutionData ,storeData=data)

    if request.method=='POST':

        store_code=request.form['store-code']
        store_name=request.form['store-name']
        institution=request.form['institution']
        print(insertStore([store_code ,store_name ,institution]))
        data=getStore()
        print(data)
        return render_template('store.html',storeData=data)

## ITEM SUB QUANTITY INSERTION AND FETCHING *
@app.route("/subunit", methods=['GET', 'POST'])
def subunit():
    if request.method=='GET':
        unitData=getUnit()
        data=getSubunit()
        return render_template('subunit_quantity.html', unitData=
```

```
        unitData , subunitData=data )

if request.method=='POST':

    subunit_code=request.form['subunit-code']
    subunit_name=request.form['subunit-name']
    unit=request.form['unit']
    print(insertSubunit([subunit_code , subunit_name , unit]))
    data=getSubunit()
    print(data)
    return render_template('subunit_quantity.html',
        subunitData=data)

## ITEM MASTER INSERTION AND FETCHING *
@app.route("/itemmaster",methods=['GET','POST'])
def itemmaster():
    if request.method=='GET':
        itemData=getItem()
        itemtypeData=getitemtype()
        brandData=getBrand()
        modelData=getModel()
        unitData=getUnit()
        subunitData=getSubunit()
        data=getItemmaster()
        return render_template('itemmaster.html',itemData=
            itemData , itemtypeData=itemtypeData , brandData=brandData ,
            modelData=modelData , unitData=unitData , subunitData=subunitData ,
            itemmasterData=data)
```

---

```
if request.method=='POST':

    itemmaster_code=request.form['itemmaster_code']
    item=request.form['item-name']
    itemtype=request.form['itemtype']
    brand=request.form['brand']
    model=request.form['model']
    unit=request.form['unit']
    subunit=request.form['subunit']
    purchaseorderlevel=request.form['purchase-order']
    currentstock=request.form['current-stock']


    print(insertItemmaster([itemmaster_code,item,itemtype,brand,
    model,unit,subunit,purchaseorderlevel,currentstock]))
    data=getItemmaster()
    print(data)

    return render_template('itemmaster.html',itemmasterData=data)


#* LOGINMASTER INSERTION AND FETCHING *
@app.route("/loginmaster",methods=['GET','POST'])
def loginmaster():
    if request.method=='GET':
        userlevelData=getUserlevel()
        employeeData=getEmployee()
        data=getLoginmaster()
        return render_template('loginmaster.html',userlevelData=
        userlevelData,employeeData=employeeData,loginmasterData=data)
```

```
if request.method=='POST':

    user_level=request.form['user-level']
    employee=request.form['employee']
    password=request.form['password']
    print(insertLoginmaster([user_level,employee,password]))
    data=getLoginmaster()
    print(data)
    return render_template('loginmaster.html',
        loginmasterData=data)

if __name__ == '__main__':
    app.run(debug=True)
```

### 6.1.2 connection.py

```
import psycopg2
conn = psycopg2.connect(
    database="fisatinventory", user='postgres', password='unnimaya',
    host='127.0.0.1', port='5432')
conn.autocommit = True
cur = conn.cursor()
```

### 6.1.3 purchase.js

```
$(document).ready(function () {

    //var phi;
```

```
// location . reset ();  
// alert ( ' hai ' );  
  
/* datatable */  
  
phi = $( '#ptable ' ). dataTable ({  
    "sPaginationType": "full_numbers",  
    "bPaginate": true ,  
  
    "bDestroy": true ,  
    // "bJQueryUI": true ,  
    "sAjaxSource": "itemdetails",  
    "sAjaxDataProp": "",  
    "bDeferRender": false ,  
    "aaSorting": [],  
  
    "aLengthMenu": [  
        [10, 25, 50, 100, 200, -1],  
        [10, 25, 50, 100, 200, "All"]  
    ],  
  
    "aoColumns": [  
        { "mDataProp": "itemtype" },  
        // { 'visible' : false },  
        { "mDataProp": "itemcode" },  
        { "mDataProp": "itemname" },  
        { "mDataProp": "brandcode" },  
        { "mDataProp": "brandname" },
```



```
{ "mDataProp": "modelname" },

]

})

/* datatable with tbody*/

/*
$.ajax
({
    url: 'itemdetails',

    async: false,
    cache: false,
    success: function (table) {
        alert(table);
        table = JSON.decode(table);
        tstr ="<table id=ptable class=table table-striped>
        <thead>\
        <tr>\
            <th scope=col>Itemtype </th>\
            <th scope=col>Itemcode </th>\
            <th scope=col>Itemname </th>\
            <th scope=col>Brandcode </th>\
```

```
<th scope=col>Brandname</th>\
<th scope=col>Modelname</th>\
</tr>\
</thead> <tbody>";

for (x in table)
{

    tstr+="<tr><td>"+table[x]["itemtype"]+"</td>";

}

tstr+="</tbody></table>";
$('#purchasetableload').html(tstr);

phi = $('#ptable').dataTable({
    "sPaginationType": "full_numbers",
    "bPaginate": true,

    "bDestroy": true,
    //"bJQueryUI": true,
    "sAjaxSource": "itemdetails",
    "sAjaxDataProp": "",
    "bDeferRender": false,
    "aaSorting": [],

    "aLengthMenu": [
        [10, 25, 50, 100, 200, -1],
```

```
        [10, 25, 50, 100, 200, "All"]
    ],

    "aoColumns": [
        { "mDataProp": "itemtype" },
        { "mDataProp": "itemname" },
        { "mDataProp": "itemcode" },
        { "mDataProp": "brandname" },
        { "mDataProp": "brandcode" },
        { "mDataProp": "modelname" },

    ]

    })

    }

    })

    */

    /*end tbody*/

    $('#ptable tbody tr').live('click', function () {
        var iPos = phi.fnGetPosition(this);
        // alert(iPos);
    });
```

```
if (iPos != null) {  
    var aData = phi.fnGetData(iPos); // get data of the clicked  
  
    $('#item').val(aData["itemname"] + "/" + aData["brandname"] + "  
    valpurchaseitem = aData[2]; // value is taken from printe  
  
}  
  
/*AJAX call for unit loading when row click*/  
data =new Object();  
data.item=$('#item').val();  
$.ajax  
({  
    url:'unit',  
    type:"POST",  
    data:data,  
    async:false,  
    cache:false,  
    success:function(table1)  
    {  
  
        $('#unit').val(table1);  
  
    }  
}
```

```
        })
        /*unit loading when row click end */

    })

/*end of datatable*/

// purchase loading
$('#purchasedetails').hide();

$.ajax
(
    {
        url: 'employee',

        async: false,
        cache: false,
        success: function (table1) {
            // alert(table1);
            table1 = JSON.decode(table1);
            locationData = "";
            locationData += "<option selected value='None'>None</option>";
            for (x in table1) {

                locationData += "<option value=" + table1[x]['emp
```

```
        }

        locationData += "<option selected value='employee" + employeeName + ">" + employeeName + "</option>";

        $('#employee').html(locationData);

    }

    })

// for department

$.ajax
(
    {
        url: 'department',

        async: false,
        cache: false,
        success: function (table1) {
            // alert(table1);
            table1 = JSON.decode(table1);
            locationData = "";
            locationData += "<option selected value='None'>None</option>";
            for (x in table1) {

                locationData += "<option value=" + table1[x]['deptname'] + ">" + table1[x]['deptname'] + "</option>";
            }

            locationData += "<option selected value='departmentname'>departmentname</option>";
        }
    }
);
```

```
$( '#department' ).html( locationData );

    }
})

$.ajax
({
    url: 'custodian',

    async: false,
    cache: false,
    success: function ( table1 ) {
        // alert( table1 );
        table1 = JSON.decode( table1 );
        locationData = "";
        locationData += "<option selected value='None'>None</option>";
        for ( x in table1 ) {

            locationData += "<option value=" + table1[x]['uselevel'] + ">" + table1[x]['name'] + "</option>";
        }

        // locationData += "<option selected value='userlevel'>userlevel</option>";

        $( '#custodian' ).html( locationData );

    }
})
```

```
// adding items in the table button function
```

```
$('#purchaseadd').click(function () {
```

```
    // alert("adding");
```

```
    var data = $('#purchaseform').serializeArray();
```

```
    // alert($('#purchaseform').serialize());
```

```
    data = serializeToJson(data);
```

```
    var obj = $.parseJSON(data);
```

```
    obj = JSON.decode(obj);
```

```
    // alert(obj['item']);
```

```
    // alert(data[2]);
```

```
    $('#purchasedetails').show();
```

```
    table = "<tr id= onclick=btndelfunc($(this))><td></td><td>" +
```

```
    $('#purchasetable').append(table);
```

```
    addSerialNumber();
```

```
})
```

```
// edit purchase item button
```

```
$('#purchaseedit').click(function () {
```

```
    if ($('#purchasetable').find('input[type=checkbox]:checked').
```

```
        alert("Please select one record to edit");
```

```
    }
```



```
// alert("delete");

$("#purchasetable tbody").find('input[name="checkbox"]').each(
    if ($(this).is(":checked")) {

        dt = $('#purchasetable');
        $(this).parents("tr").find('td:eq(1)').text()
        $(this).parents("tr").find('td:eq(2)').text()
        $(this).parents("tr").find('td:eq(3)').text()
        $(this).parents("tr").find('td:eq(4)').text()
        $(this).parents("tr").find('td:eq(5)').text()
        // dt[0].rows[1].cells[2].innerHTML = 'New Content';
        // alert($(this).parents("tr").find('td:eq(2)').text()
        $('#item').val($(this).parents("tr").find('td:eq(1)')
        $('#quantity2').val($(this).parents("tr").find('td:eq(2)')
        $('#unit').val($(this).parents("tr").find('td:eq(3)')
        $('#custodian').val($(this).parents("tr").find('td:eq(4)')
        $('#justification').val($(this).parents("tr").find('td:eq(5)')
        $('#purchaseadd').attr('value', 'update data');

    }

})

// delete purchase item button
$('#purchasedetele').click(function () {
```

```
        if (!confirm("Do you want to delete")) {
            return false;
        }
        $("#purchasetable tbody").find('input[name="checkbox"]').each(

            if ($(this).is(":checked")) {

                $(this).parents("tr").remove();

            }

        });
    })

//purchase submit button

$('#purchasesubmit').click(function () {
    // alert('HAI');
    data = new Object();
    var item = new Array();
    var quantity = new Array();
    var unit = new Array();
    var custodian = new Array();
    var justification = new Array();
    /* code for fetcing data from table */
    $('#purchasetable tr').each(function (row, tr) {
        // alert($(tr).find('td:eq(3)').text());
        // alert($(tr).find('td:eq(2)').text());
        item.push("" + $(tr).find('td:eq(1)').text() + "");
```

```
quantity.push("" + $(tr).find('td:eq(2)').text() + "");
unit.push("" + $(tr).find('td:eq(3)').text() + "");
custodian.push("" + $(tr).find('td:eq(4)').text() + "");
justification.push("" + $(tr).find('td:eq(5)').text() + ""
}))
```

```
data.item = item;
data.quantity = quantity;
data.unit = unit;
data.custodian = custodian;
data.justification = justification;
// alert("quantity" + data.quantity);
```

```
$.ajax
({
    url: 'purchaseinsert',
    type: "POST",
    data: data,
    // contentType: 'application/json; charset=UTF-8',
    async: false,
    cache: false,
    success: function (message) {
        alert(message);
    }
})
```

```
        })

    })

    }) // main close of combo box

// functions
function serializeToJson(serializer) {
    var _string = '{';
    for (var ix in serializer) {
        var row = serializer[ix];
        _string += '"' + row.name + '":' + row.value + '",';
    }
    var end = _string.length - 1;
    _string = _string.substr(0, end);
    _string += '}';
    console.log('_string: ', _string);
    return JSON.encode(_string);
}

// button delete function
```

---

```
// function btndelfunc(data1) {  
//     alert(data1.attr('id'));  
//     $("#repairtable tr:eq(1)").remove();  
  
// }  
// serial number adding table function  
var addSerialNumber = function () {  
    $('table tr').each(function (index) {  
        $(this).find('td:nth-child(1)').html(index + 1 - 2);  
    });  
};
```

### **6.1.4 style.css**

```
background-color: lightblue;
}

h1 {
    color: navy;
    margin-left: 20px;
}
title {
    color: red;
    text-align: center;
}
h2 {
    text-align: center;
    color: red;
}

.center {
    margin: 0;
    position: absolute;
    top: 50%;
    left: 50%;
    -ms-transform: translate(-50%, -50%);
    transform: translate(-50%, -50%);
}
.grid-container {
    display: grid;
    grid-template-rows: auto auto auto auto;
    gap: 20px;
```

```
background-color: #2196F3;
padding: 10px;
}

.grid-container > div {
background-color: rgba(255, 255, 255, 0.8);
border: 1px solid black;
text-align: center;
font-size: 20px;
}
```

### 6.1.5 purchase.py

```
$(document).ready(function () {
    alert('hai ');

    $('#purchasedetails').hide();

    $.ajax
    ({
        url: 'employee',

        async: false,
        cache: false,
        success: function (table1) {
            alert(table1);
            table1 = JSON.decode(table1);
            locationData = "";
            locationData += "<option selected value='None'>
```

```
        None</option>";
        for (x in table1) {

            locationData += "<option value=" + table1[x]
            ['employee'] + ">" + table1[x]['employee'] +
            "</option>";
        }

        locationData += "<option selected value=
        'employee'>employee</option>";

        $('#employee').html(locationData);

    }

})

// for department

$.ajax
(
    {
        url: 'department',

        async: false,
        cache: false,
        success: function (table1) {
            alert(table1);
            table1 = JSON.decode(table1);
            locationData = "";
            locationData += "<option selected value='None'>
```



```
None</option>";
    for (x in table1) {
        locationData += "<option value=" + table1[x]
        ['departmentname'] + ">" + table1[x]['departmentname']
        + "</option>";
    }
```

```
locationData += "<option selected value='departmentname'>departmen
```

```
$('#department').html(locationData);
```

```
    }
    })
```

```
// adding items in the table button function
```

```
$('#purchaseadd').click(function () {
    alert("adding");
    var data = $('#purchaseform').serializeArray();
    alert($('#purchaseform').serialize());
    data = serializeToJson(data);
    var obj = $.parseJSON(data);
    obj = JSON.decode(obj);
    alert(obj['item']);
    // alert(data[2]);
    $('#purchasedetails').show();
    table = "<tr id= onclick=btndelfunc($(this))><td></td>
```

```
<td>" + obj[ 'item ' ] + "</td><td>" + obj[ 'quantity ' ] +
"<td>" + obj[ 'unit ' ] + "</td><td>" + obj[ 'custodian ' ] +
"</td><td>" + obj[ 'justification ' ] + "</td><td><input
type=checkbox name=checkbox value=edit></td><td><input
type=checkbox name=checkbox value=delete></td></tr>";
$( '#purchasetable' ).append( table );
addSerialNumber();

})

// edit purchase item button
$( '#purchaseedit' ).click( function () {
    alert( "edit" );
})

// delete purchase item button
$( '#purchaseddelete' ).click( function () {
    alert( "delete" );
    $( "#purchasetable tbody" ).find( 'input[ name="checkbox" ]' ).each

        if ( $( this ).is( ":checked" ) ) {

            $( this ).parents( "tr" ).remove();

        }

    });
})

// purchase submit button
```

---

```
$('#purchasesubmit').click(function ()

    alert('hai ');
    var data = $('#purchaseform').serializeArray();
    alert($('#purchaseform').serialize());
    data = serializeToJson(data);
    $.ajax
        ({
            url: 'purchaseinsertion ',
            type: "POST",
            data: data ,
            contentType: 'application/json;charset=UTF-8',
            async: false ,
            cache: false ,
            success: function (message)
            {
                alert(message);
            }
        })
    })

}) // main close of combo box

// functions
function serializeToJson(serializer) {
    var _string = '{}';
    for (var ix in serializer) {
```

```
        var row = serializer[ix];
        _string += ''' + row.name + '":"' + row.value + '",'';
    }
    var end = _string.length - 1;
    _string = _string.substr(0, end);
    _string += '}';
    console.log(' _string: ', _string);
    return JSON.encode(_string);
}

// button delete function
function btndelfunc(data1) {
    alert(data1.attr('id'));
    $("#repairtable tr:eq(1)").remove();

}

// serial number adding table function
var addSerialNumber = function () {
    $('table tr').each(function (index) {
        $(this).find('td:nth-child(1)').html(index + 1 - 1);
    });
};
```

## 6.2 Screen Shots

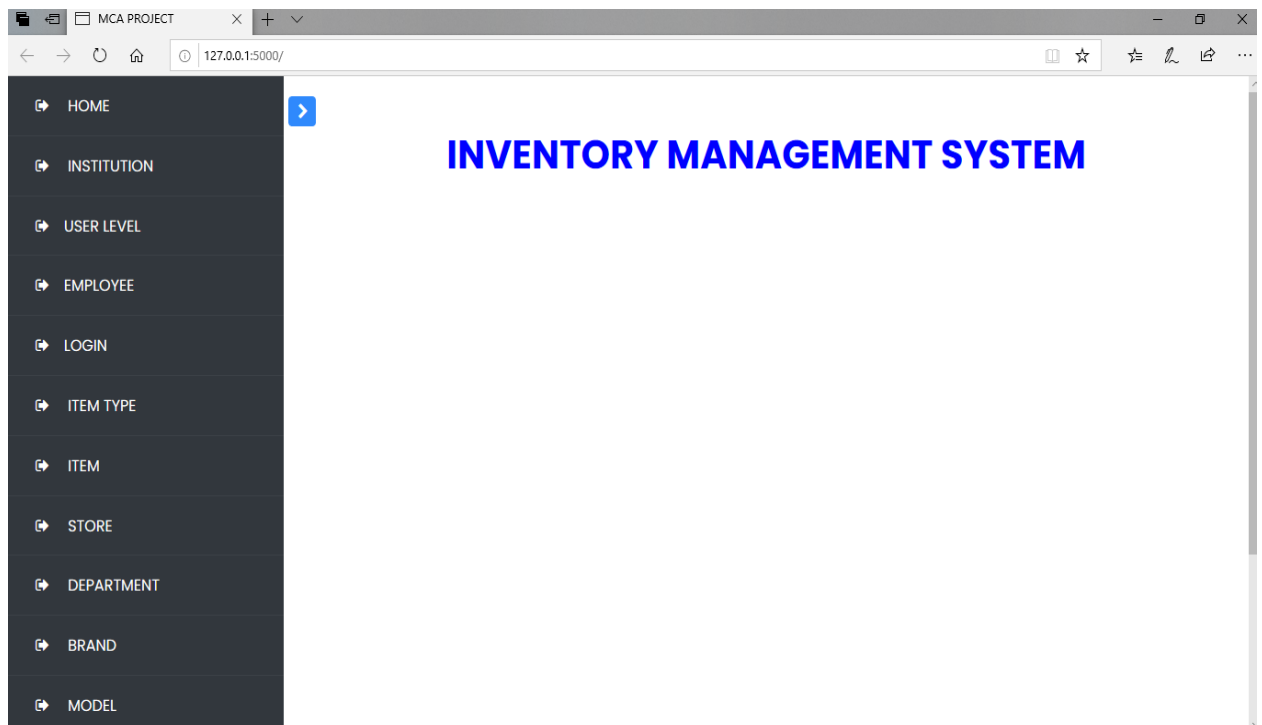


Figure 6.1: HOME PAGE

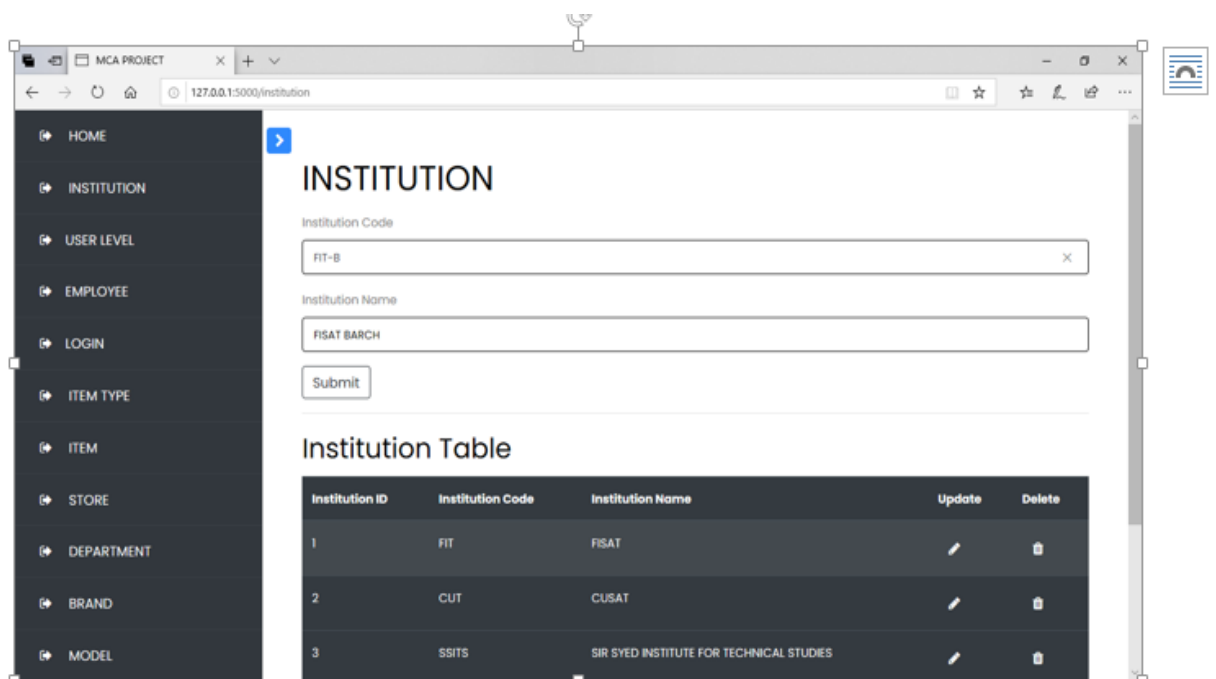


Figure 6.2: INSTITUTION PAGE

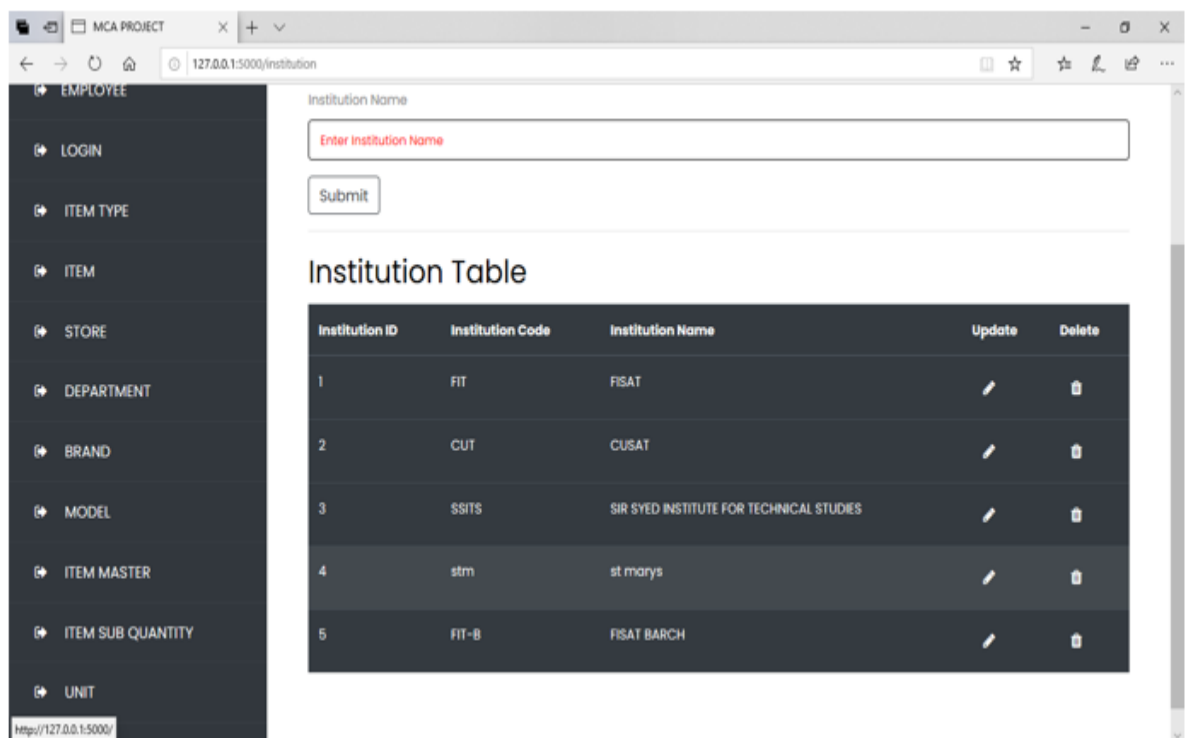


Figure 6.3: INSTITUTION PAGE

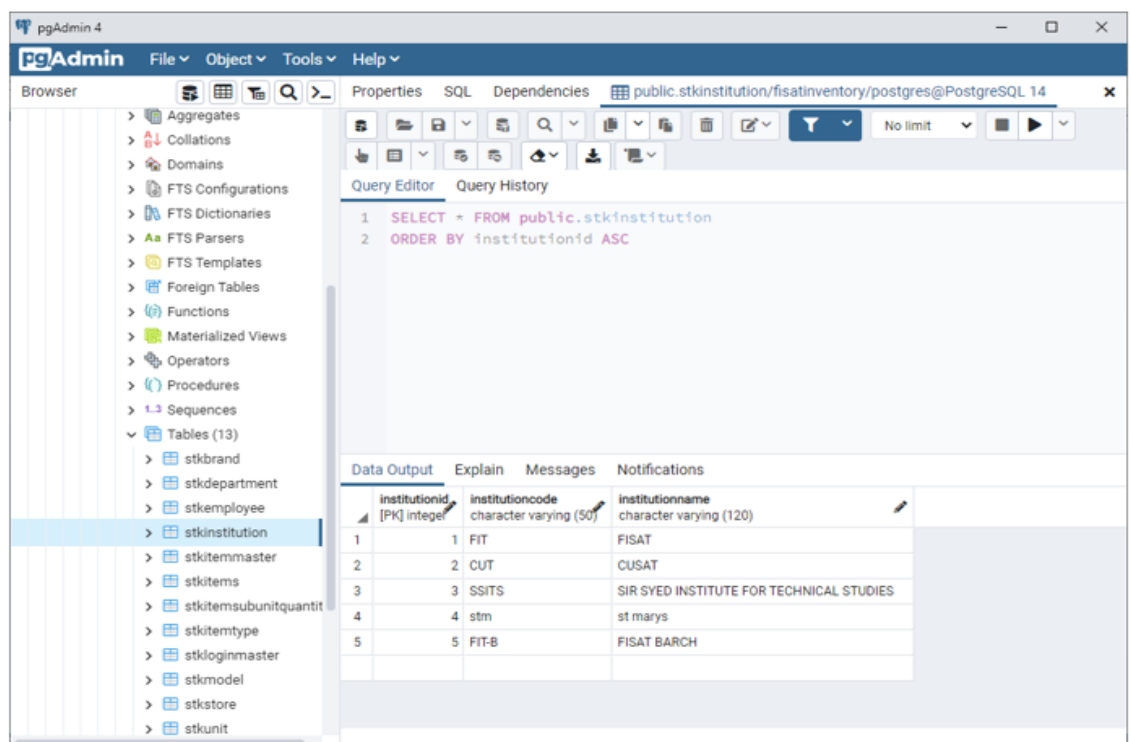


Figure 6.4: INSTITUTION PAGE- TABLE DATA

The screenshot shows a web application interface for managing item types. On the left is a dark sidebar with a menu containing: HOME, INSTITUTION, USER LEVEL, EMPLOYEE, LOGIN, ITEM TYPE (highlighted), ITEM, STORE, DEPARTMENT, BRAND, and MODEL. The main content area is titled 'ITEM TYPE' and contains a form with two input fields: 'Item Type Code' with the value 'CN' and 'Item Type Name' with the value 'Non-consumable'. Below the inputs is a 'Submit' button. Under the form is a section titled 'Item Type Table' containing a table with the following data:

Item Type ID	Item Type Code	Item Type Name	Update	Delete
1				
2	aaaa	bbbb		
3	STA	CONSUMABLE		

Figure 6.5: ITEM TYPE PAGE

The screenshot shows the same web application interface, but the 'Item Type Table' now displays 8 rows of data. The sidebar menu is also visible, with 'ITEM' highlighted. The table data is as follows:

Item Type ID	Item Type Code	Item Type Name	Update	Delete
1				
2	aaaa	bbbb		
3	STA	CONSUMABLE		
4	IT001	consumable		
5				
6	IT002	Non-consumable		
7	IT003	Non-consumable		
8	CN	Non-consumable		

Figure 6.6: ITEM TYPE PAGE

**Brand's**

Brand code  
BR-pen

Brand Name  
LEX

Select items  
IT-PN

Submit

**Brand Table**

Brand Id	Brand Code	Brand Name	Item Name	Update	Delete
1	BR-pen	CELLO	1		

Figure 6.7: BRAND PAGE

Enter Brand Name

Select items

Submit

**Brand Table**

Brand Id	Brand Code	Brand Name	Item Name	Update	Delete
1	BR-pen	CELLO	1		
2	BR-pen	LEX	1		

Figure 6.8: BRAND DISPLAY PAGE



## PURCHASE REQUEST

Purchase Requested By

--Please choose purchase request--

Purchase Request for department

--Please choose dept--

Enter Code

Enter code

Advanced Search:

Item type

--Please choose Item Typ--

Brand

--Please choose Brand--

Model

--Please choose Model--

---

Item

item name

Request Quantity

--Please choose Quantity --

Unit

--Please choose Unit--

Custodian

--Please choose Custodiz --

Justification

submit

items id	Required quantity	unit	custodian	justification	update	Delete
----------	-------------------	------	-----------	---------------	--------	--------

Figure 6.9: PURCHASE PAGE

# Chapter 7

## REFERENCES

- [1] <https://ieeexplore.ieee.org/document/5478077>
- [2] <https://www.youtube.com/watch?v=eR3rcalj06Q>
- [3] <https://www.wikipedia.com>
- [4] <https://ieeexplore.ieee.org/document/5478077>
- [4] <https://ieeexplore.ieee.org/abstract/document/9447350>