**Chapter-1**

**INTRODUCTION**

The aim of this project is to build an e-commerce website to empower the economy of the small-scale stores in order to provide a digital platform near to big giants

**1.1 MVT architecture**

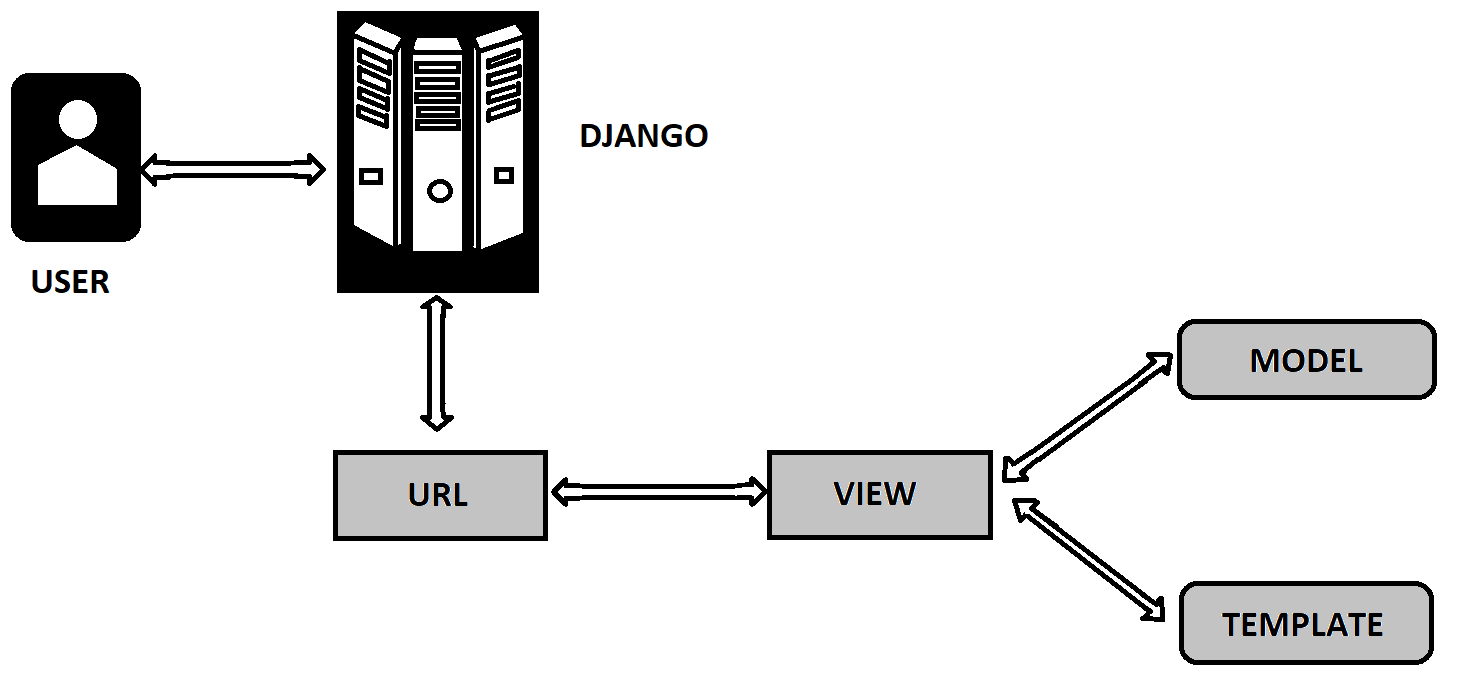
The MVT architecture stands for *Model View Template* it is the back-end design pattern for the Django web framework.

In Model, we have to write classes that create the table in the database each class creates a column in a database. It handles the database.

To execute the business logic the function or class must be written in the views.py file

The template handles the HTML files to render whenever the URL mapping occurs and it communicates with the model to store the data in the database.

Here the framework works as the controller hence it is similar to MVC architecture



**1.2 DJANGO**

Django is one of the most popular python server-side web frameworks, the latest version is Django 4.0.6 which follows MVT architecture.

Django is popular because of its speed, security, and scalability Django offers robust models for handling HTTP requests and sends templates as the HTTP response.

**1.3 FRONT-END**

The HTML and CSS are the client-side languages and with a bit of bootstrap the HTML files are presented as templates by Django as an HTTP response.

The static files are stored in the static folder of the Django application.

### 1.4 DATABASE

### Whatever the application we just need a database to store the data collected from the users via HTML forms as well to be rendered via HTML here we used a python inbuilt lightweight database that is an *SQLLITE* database we can perform

**Summary:** This chapter gives a brief introduction to the project and the technology stack used in this project

### 1.5 HTML

### Html is a client-side web application known as a hypertext markup language used to design the landing and its navigation pages for a particular web application, it consists of tags to define the different sections of the front page which inform the browser to display the sections in a well structured and formatted way.

### 1.5 CSS

### CSS is known as cascading stylesheets used for styling HTML pages, it consists

### of properties such as selectors, pseudo-classes, backgrounds, float, borders, colors, height/widths, and animations which makes designing the website layout and icons very impressive. It is a cornerstone framework of WWB which delivers the webpage presentable.

### 1.5 SQLLITE

### It is the most commonly used database engine in the backend web framework,

### It is the most reliable, stable, and cross-platform for linking the client-side

### Frameworks for serverside applications, This C-language library can be used across all the android and computer devices which provides the flexibility

**Chapter-2**

**LITERATURE REVIEW AND OBJECTIVES**

**2.2 Adding pins to MapBox**

Using Geocoder and MapBox we can add pins to geologically mark locations on the map which the user can interact with. Using libraries like ReactMapGL, we can get different functions like tempMarker and selected to make the pins interactive.

**2.3 MongoDB Mongoose**

Your MongoDB clusters or collections are connected to your Node.js application through Mongoose.js [4]. It gives you the ability to design schemas for your documents. When generating and interacting with schemas, Mongoose provides an abundance of functionalities. Express and Yarn is used to connect Mongoose with any project.

**2.4 Mongoose connection**

The process of connection between MongoDB and Node.js is handled with ease by Mongoose. Express plays an important part for the whole process. But just connecting to Mongoose is not enough as it passes the queries to MongoDB, hence to pass and catch these queries it is necessary to define the schema of the database along with the query format.

**2.5 Creating Interactive forms**

Interactive forms can be created in React using useState hooks provided in the same which is used to manage the state of the form. First a basic form using HTML tags is created and then functionalities like validation, etc., can be added. Functions like onChange() are used to define what needs to be done if the state hooks are triggered. Events like onBlur, onKeyUp, onFocus, etc., makes the form work and look better.

**2.6 Salting hashed passwords**

The user's password in plain text is used to generate this lengthy, intricate, and distinctive hash. You'll save the hash version of the user's password in the database as opposed to the plain-text version. In order to ensure a unique hash even when the inputs are same, salting hashes [5] involves adding random data to the hash function's input. As a result, the distinctive hash created by the salt addition can defend us from several attack methods, such as hash table attacks, while stifling dictionary and offline brute force attempts.

**2.7 React Weather Application**

This can be produced using a free OpenWeather API [7]. Make a weather app where the user enters the name of the state, and the app displays the temperature and the weather on the user interface as a result. Though you may also display additional information, such as the date, month, and year. The weather API, which automatically refreshes based on the weather, is where we'll acquire all the weather information.

**2.8 React Password Generation Application**

Every time we have to make a password for a website or a mobile application, it can be difficult for us to immediately come up with a secure password. Therefore, you can build a password generator [7] using React that allows you to generate secure passwords and copy them to your clipboard. The characters that go into a user's password can be customized, including small letters, capital letters, special characters, digits, etc.

**2.9 Objectives**

1. To create a MongoDB database to store all information like login credentials, reviews, and details of places like restaurants, hotels, etc. based on the locality.
2. To make a dynamic frontend that is crisp and easy to use.
3. To connect everything and ensure the proper functioning of the project.

**Summary:** This chapter talks about the different methodology and applications used to come up with the project implementation.

**Chapter-3**

**PROBLEM STATEMENT**

Aims to develop an e-commerce grocery website that helps to revive the small-scale economy and the daily needs of a customer at their doorsteps, In order to provide flexibility for both admin and end users of the product this idea came up to provide a digital interface that provides the 24/7 web service to the Shopvendors to sell their products online anytime throughout the year and the customers to place their orders on the web platform and delivered at their doorsteps, this project helps to lead the life of a customer and to prevent the financial instability of the small scale vendors beyond the pandemic situations and unforeseen circumstances of the present world. The financial transactions and delivery of the groceries through this digitalized platform prevent financial damage and avoid contracting deadly viral diseases which might be fatal and may succumb the lives to death.

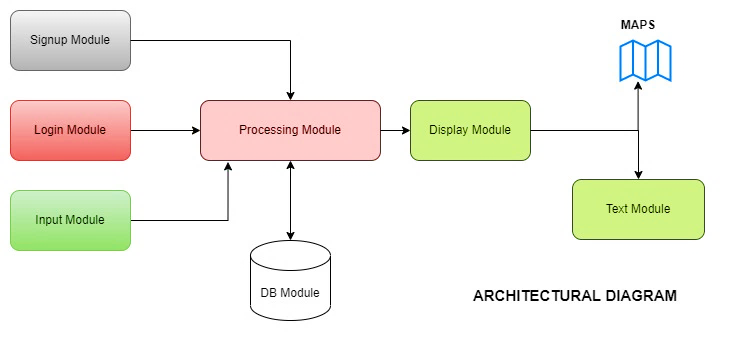
**Chapter-4**

**SYSTEM DESIGN**

System design talks about the process of designing the architecture, components, and interfaces for a system so that it meets the end-user requirements. Quality system design is essential in developing functional and lasting applications.

**4.1 Architecture Design**

A system's architecture outlines its main parts, their connections (structures), and how they work together. There are several contributing variables to software architecture and design, including business strategy, quality attributes, human dynamics, design, and IT environment.

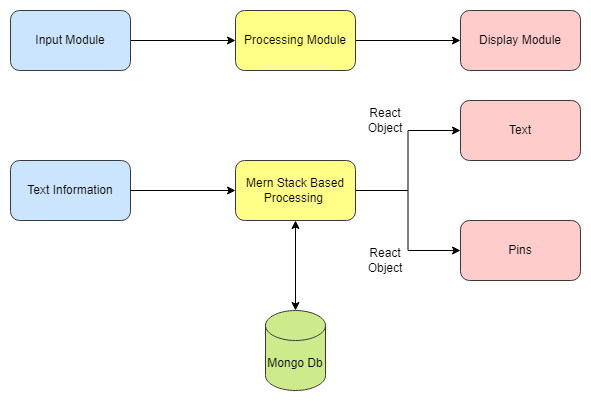
Figure 4.1 below shows the project’s architectural design.****

***Figure 4.1 Architectural Design***

1. **Signup Module**: User will be able to register self through this module by passing username, email and password.
2. **Login Module**: User will be able to login to the app through this module by passing username and password.
3. **Input Module**: User will be able to add pins to the map by passing information about the place.
4. **Processing Module:** This module processes the data given by user, performs the required operations and passes it on to the storage module or display module.
5. **DB Module:** Stores the data passed down from processing module and sends the data back whenever required.
6. **Display Module:** User will be able to see the outputs of the operations through this module in the form of text or map graphics.

**4.2 Software Architecture Block**

A visual depiction that depicts the actual physical implementation of a software system's components is called an architecture diagram. It displays the relationships, constraints, and boundaries between each piece as well as the overall structure of the software system.

Figure 4.2 shows the project’s software architecture block

***Figure 4.2 SAB***

1. **Input Module:** This module has components that takes input from the user. Text inputs with coordinates of the pin on the map, title, type, ratings.
2. **Processing Module**: MERN based processing is done where React framework handles the states for UI rendering. Express has multiple endpoints for handling different functionalities which in turn communicates with the database to perform the specified operation. Processing module communicates with the DB to retrieve and write information.
3. **Display Module:** This module communicates with the Processing module by monitoring states and changing specific components on the viewport as text and Pins on a map

**4.3 Flowchart**

Flowchart is one of the method for representation of the algorithm of the problem. An algorithm is represented graphically in the form of flowchart .The flowchart is constructed for the given algorithm by using some particular geometrical figures.The flowchart provides the control flow of the algorithm.

in the form of flowchart

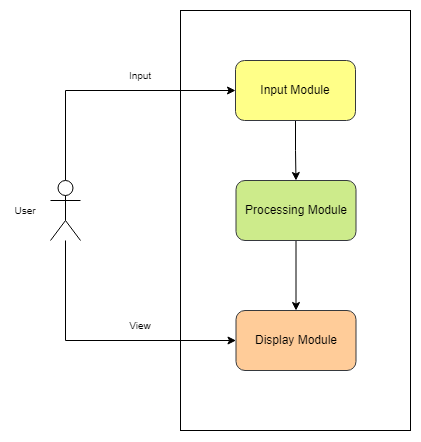
Figure 4.3 shows the project application’s flowchart.

***Figure 4.3 Flowchart***

1. Home page contains core modules like a map, login and registration. Registration and login go through a validation + store/retrieve process while writing user data to the database and generates an access token which could be used to authenticate users, As only authenticated users are allowed to make changes or add new reviews.
2. Maps module presents a visual map for the users to create new pins which has visual pin locations queried from the database.
3. Search field is used to filter reviews based on the type of that place, when a user enters a particular type, it gets fetched from the database and rendered on the map.

**4.4 Use case Diagram**

The scope and high-level functions of a system are described in use-case diagrams. The interactions between the system and its actors are also depicted in these diagrams. Utilize-case diagrams show what the system does and how the actors use it, but they do not show how the system works within.

Figure 4.4 shows the use case diagram of the project.

***Figure 4.4 Use Case Diagram***

1. **Input Module:** User will be able to register, login and add pins to the map by passing information about the place.
2. **Processing Module:** This module processes the data given by user, performs the required operations and passes it on to the storage module or display module.
3. **Display Module:** User will be able to see the outputs of the operations through this module in the form of text or map graphics.

**Chapter-5**

**IMPLEMENTATION**

The technical data package (TDP) for the software is converted into one or more fabricated, integrated, and tested software configuration items during the software implementation stage so that they are prepared for software acceptance testing.

**5.1 Algorithm**

An algorithm is a process used to carry out a computation or solve a problem. In either hardware-based or software-based routines, algorithms function as a detailed sequence of instructions that carry out predetermined operations sequentially. All aspects of information technology employ algorithms extensively.

The algorithm defined for our project is as follows-

Algorithm 8.1: ***Proposed work Algorithm***

***Input***: User Credentials (Register and Login), Location Type (Lat and Lng)

***Output***: Map with Locations Identified using Pins

***do***:

Step 1: User completes registration

Step 2: Input user Creds into login

user\_creds*-à login(username, password****)***

***if*** user\_cred is valid:

***do***:

*pin\_access(true)*

***else***

***return*** to *login(username, password)*

***end if***

Step 3:

*display\_map() // Loads the Mapbox API*

*display\_pins() // Loads all user-defined pins*

*display\_current\_user\_pins() // Loads current user pins*

Step 4:

***if*** current\_user wants to add pins:

***do***:

*add\_pins(Name,Description,Tags,Rating)*

Step 5: Display all the information such as Map, all other pins, current user defined pins

***goto*** Step 3

**5.2 Important Code Snippets**

**5.2.1 Mapbox component**

<ReactMapGL

{...viewport}

mapboxApiAccessToken=" "

width="100%"

height="100%"

transitionDuration="200"

onViewportChange={(viewport) => setViewport(viewport)}

onDblClick={currentUsername && handleAddClick}

>

{…otherComponents}

</ReactMapGL>

**5.2.2 Initial render loads all pins**

{pins.map((p,index) => (

<>

<Marker

latitude={p.lat}

longitude={p.long}

offsetLeft={-3.5 \* viewport.zoom}

offsetTop={-7 \* viewport.zoom}

>

<Room

style={{

fontSize: 7 \* viewport.zoom,

color:

currentUsername === p.username ? "purple" : "red",

cursor: "pointer",

}}

onClick={() => handleMarkerClick(p.\_id, p.lat, p.long)}

/>

</Marker>

**5.2.3 Each pin on click renders out a popup**

{p.\_id === currentPlaceId && (

<Popup

key={p.\_id}

latitude={p.lat}

longitude={p.long}

closeButton={true}

closeOnClick={false}

onClose={() => setCurrentPlaceId(null)}

anchor="left"

className="zindexed"

>

<button className="deletebutton" onClick={() => handleDelete(p.\_id) }><DeleteForeverIcon fontSize="small"/></button>

**5.2.4 To create a new pin**

const handleSubmit = async (e) => {

e.preventDefault();

const newPin = {

username: currentUsername,

title,

type: typeinp,

desc,

rating: star,

lat: newPlace.lat,

long: newPlace.long,

};

try {

const res = await axios.post("/pins", newPin);

setPins([...pins, res.data]);

setNewPlace(null);

} catch (err) {

console.log(err);

}

};

**5.2.5 Method that handles delete pins**

async function handleDelete(id){

console.log(id)

try{

const allPins = await axios.get("/pins/delete/" + id);

setPins(allPins.data);

}catch (err) {

console.log(err);

}

}

**5.2.6 Pins are filtered out as the state of the text field’s value changes**

useEffect(() => {

console.log(type)

const getPins = async () => {

try {

const allPins = await axios.get("/pins/" + type);

setPins(allPins.data)

}catch (err) {

console.log(err);

}

};

getPins();

},[type]);

**5.2.7 Connecting mongo to the Express server**

const mongoose = require("mongoose");

mongoose

.connect(process.env.MONGO\_URL, {

useNewUrlParser: true,

useUnifiedTopology: true,

useCreateIndex: true, })

.then(() => console.log("MongoDB connected!"))

.catch(err => console.log(err));

//dataset desc example of table entry

**5.2.8 Creating a new pin in MongoDB**

router.post("/", async (req, res) => {

const newPin = new Pin(req.body);

// console.log(newPin);

try {

const savedPin = await newPin.save();

console.log(savedPin);

res.status(200).json(savedPin);

} catch (err) {

res.status(500).json(err);

}

});

Record inserted as follows:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **\_id** | **Username** | **Title** | **Type** | **Desc** | **Rating** | **Lat** | **Long** |
| 1 | Aniket | Onesta | Cafe | Nicee | 4 | 13.070811 | 77.391234 |

**5.2.9 Using bcrypt to validate password**

const validPassword = await bcrypt.compare(

req.body.password,

user.password

);

!validPassword && res.status(400).json("Wrong username or password");

**Chapter-6**

**TESTING**

**Software testing** is a technique for determining whether the real software product complies with expectations and is error-free. It entails the human or automatic execution of software/system components in order to evaluate one or more interesting properties. Software testing's goal is to find flaws, gaps, or unmet requirements compared to the requirements as written. Software testing is crucial because it allows any faults or errors in the software to be found early and fixed before the software product is delivered.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case No** | **Test Case Description** | **Expected Input** | **Expected Output** | **Pass/Fail** |
| 1 | User Register | Wrong Email format | Fail | Pass |
| 2 | User Register | Correct Email format | Pass | Pass |
| 3 | User Login | Wrong Credentials | Fail | Pass |
| 4 | User Login | Correct Credentials | Pass | Pass |
| 5 | Add Pins | Pins added in Database | Fail | Pass |
| 6 | Add Pins | Pins added in Database | Pass | Pass |
| 7 | Delete Pins | Pins deleted from Database | Fail | Pass |
| 8 | Delete Pins | Pins deleted from Database | Pass | Pass |
| 9 | Find Pin Type | Pin Information fetched from Database | Fail | Pass |
| 10 | Find Pin Type | Pin Information fetched from Database | Pass | Pass |

The smallest testable components of an application, known as units, are separately and independently examined for appropriate operation as part of the **unit testing [6]** phase of software development. Software developers and occasionally QA employees use this testing methodology when the software is still in the development stage. Unit testing's primary goal is to separate written code for testing to see if it functions as intended. The table 6.1 bellow shows the unit testing done for this project.

**Unit Testing Done**

**Chapter-7**

**RESULTS AND SNAPHOTS**

**Snapshot 1**: The following figure 7.1 shows our landing page.

A map of the world

Description automatically generated with low confidence

***Figure 7.1 Landing Page***

**Snapshot 2**: The following figure 7.2 shows the records available in the database.

Graphical user interface, text, application

Description automatically generated

***Figure 7.2 Records in Database***

**Snapshot 3**: The following figure 7.3 shows registering as a test user.

Graphical user interface, application

Description automatically generated

***Figure 7.3 Registering as a user***

**Snapshot 4**: The following figure 7.4 shows test user inserted into database.

Graphical user interface, text, application, email

Description automatically generated

***Figure 7.4 User inserted into database***

**Snapshot 5**: The following figure 7.5 shows logging as a test user.

Graphical user interface, application

Description automatically generated

***Figure 7.5 Logging as test user***

**Snapshot 6**: The following figure 7.6 shows new pin being added by double clicking at any place on the map.

Graphical user interface, text, application

Description automatically generated Graphical user interface, text, application, chat or text message

Description automatically generated

***Figure 7.6 Adding a new pin***

**Snapshot 7**: The following figure 7.7 shows new pin added on the map and in the database.

Graphical user interface, text, application

Description automatically generatedGraphical user interface, application, map

Description automatically generated

***Figure 7.7 New pin added***

**Snapshot 8**: The following figure 7.8 shows pins getting filtered and shown on the map according to user input.

A map of the world

Description automatically generated with medium confidence

***Figure 7.8 Pins filtered***

**Chapter-7**

**CONCLUSION**

The application is able to show the map and all the data related to the places in real time. The user is able to register, login and create new pins and give reviews about a place which is seen by other users. The user can also delete the pins and also do selective searching based on the type of place. Through this project we learned how to work with MapBox and how to customize it according to our needs. We learned how to work with React and MongoDB and connect both. We also learned how to work with concepts like promises. In the future we can make our search filter better by adding options like data and time of the reviews and make the UI better.

**REFERENCES**

[1] *Guide book - Wikipedia*. (n.d.). Retrieved July 7, 2022, from https://en.wikipedia.org/wiki/Guide\_book

[2] *What Is the MERN Stack? Introduction & Examples | MongoDB*. (n.d.). Retrieved July 7, 2022, from https://www.mongodb.com/mern-stack

[3] *Maps, geocoding, and navigation APIs & SDKs | Mapbox*. (n.d.). Retrieved July 7, 2022, from https://www.mapbox.com/

[4] *Mongoose v6.4.3: Getting Started*. (n.d.). Retrieved July 7, 2022, from https://mongoosejs.com/docs/

[5] *password-hash-and-salt - npm*. (n.d.). Retrieved July 7, 2022, from https://www.npmjs.com/package/password-hash-and-salt

[6] *What is Unit Testing? Definition from WhatIs.com*. (n.d.). Retrieved July 7, 2022, from https://www.techtarget.com/searchsoftwarequality/definition/unit-testing

[7] *10 React Projects to Build (With Source Code) - InterviewBit*. (n.d.). Retrieved July 9, 2022, from https://www.interviewbit.com/blog/react-projects/