INVENTORY MANAGEMENT SYSTEM

## A Project Report Submitted

**In Partial Fulfillment of the Requirements for the Degree of**

MASTERS OF COMPUTER APPLICATION

## By

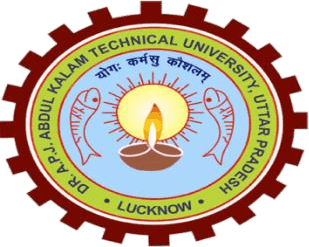
**Shrish Shukla**

**(1900290149094)**

**Under the Supervision of**

**Mr. Ankit Verma**

**KIET Group of Institutions, Ghaziabad**

****

**To the**

**FACULTY OF MCA**

**DR. APJ ABDUL KALAM TECHNICAL UNIVERSITY**

**(Formerly Uttar Pradesh Technical University) LUCKNOW**

**July 2021**

**DECLARATION**

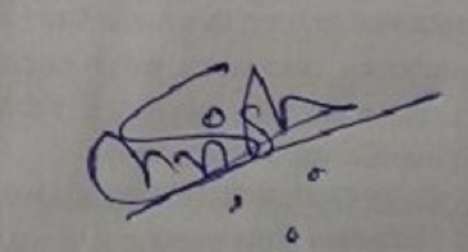
I hereby declare that the work presented in this report entitled “Inventory Management System", was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.

I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. I have used quotation marks to identify verb a time sentences and given credit to the original authors/sources.

I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

Name : Shrish Shukla Roll No. : 1900290149094

Branch : Master of Computer Application



## CERTIFICATE

Certified that **Shrish Shukla** (**1900290149094**) has carried out the project work presented in this report entitled “**Inventory Management System**” for the award of **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University, Lucknow under my supervision. The report embodies result of original work, and studies are carried out by the student himself and the content soft here port do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University.

**Mr. Ankit Verma External Examiner**

Assistant Professor

Department of Computer Applications KIET Group of Institutions, Ghaziabad

**Dr. Ajay Kumar Srivastava**

Professor & Head

Department of Computer Applications

KIET Group of Institutions, Ghaziabad

Date:

# ABSTRACT

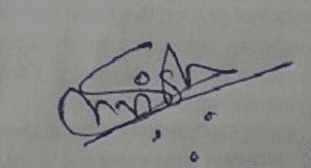
Inventory Management System is important to ensure quality control in businesses that handle transactions revolving around consumer goods. Without proper inventory control, a large retail store may run out of stock on an important item. A good Inventory Management System will alert the retailer when it is time to reorder. Inventory Management System is also an important means of automatically tracking large shipments. For example, if a business orders ten pairs of socks for retail resale, but only receives nine pairs, this will be obvious upon inspecting the contents of the package, and error is not likely. On the other hand, say a wholesaler orders 100,000 pairs of socks and 10,000 are missing. Manually counting each pair of socks is likely to result in error. An automated Inventory Management System helps to minimize the risk of error. In retail stores, an Inventory Management System also helps track theft of retail merchandise, providing valuable information about store profits and the need for theft-prevention systems. Automated Inventory Management System work by scanning a barcode either on the item. A barcode scanner is used to read the barcode, and the information encoded by the barcode is read by the machine. This information is then tracked by a central computer system. For example, a purchase order may contain a list of items to be pulled for packing and shipping. The Inventory Management System can serve a variety of functions in this case. It can help a worker locate the items on the order list in the warehouse, it can encode shipping information like tracking numbers and delivery addresses, and it can remove these purchased items from the inventory tally to keep an accurate count of in-stock items.

## ACKNOWLEDGEMENT

Success in life is never attained single handedly. My deepest gratitude goes to my Project supervisor, **Mr. Ankit Verma, Professor, Department of Computer Applications** for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions. Words are not enough to express my gratitude for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.



**Shrish Shukla 1900290149094**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
|  | | Page No. |
| Declaration | | i |
| Certificate | | ii |
| Abstract | | iii |
| Acknowledgement  List of Figure  List of Tables | | iv  v  vi |
| **CHAPTER 1: INTRODUCTION** | | **4-6** |
| 1.1 Project Description | | 4-5 |
| 1.2 Project Purpose | | 5 |
| 1.3 Tools/Environment used | | 6 |
| **CHAPTER 2: LITERATURE REVIEW** | | **7-13** |
| 2.1 System analysis | | 7-8 |
| 2.2 | Feasibility study | 9 |
| 2.3 | Fact finding technique | 10-11 |
| 2.4 | Scheduling | 12-13 |
| **CHAPTER 3: REQUIREMENT SPECIFICATIONS** | | **14-18** |
| 3.1 Hardware Requirements | | 14 |
| 3.2 Software Requirements | | 14-15 |
| 3.3 SRS of the project | | 16-18 |
| 3.3.1 Functional | | 18 |
| 3.3.2 Non-Functional | | 19 |
|  | |  |

|  |  |
| --- | --- |
| **CHAPTER 4: DESIGN** | **20-28** |
| * 1. Entity Relationship Diagram   2. Data Flow Diagram      1. 0 level Data Flow Diagram | 21  22-24  22 |
| * + 1. 1st level Data Flow Diagram     2. 2nd level Data Flow Diagram | 23  24 |
| * 1. Data Dictionary      1. Table used | 26-30  25-28 |
| **CHAPTER 5: USER INTERFACES / CODING**   * 1. User Interface   2. Coding | **26-68**  26-33  34-68 |
|  |  |
| **CHAPTER 6: LIMITATION AND FUTURE SCOPE** | **69-71** |
| 6.1 Limitation | 69 |
| 6.2 Future Scope  6.2 References | 70  71 |

**LIST OF FIGURES**

1 Fig. 2 Process Design of System Analysis 5

2 Fig. Gantt Chart 9

3 Fig. ERD 13-14

1. Fig. 0 LEVEL DFD 15
2. Fig. 1STLEVEL DFD 16
3. Fig. 2NDLEVEL DFD 17

**LIST OF TABLES**

TABLE 4.5.1 CUSTOMER DETAILS 22

TABLE 4.5.2 DELIVERY DETAILS 22

TABLE 4.5.3 ADMIN DETAILS 23

TABLE 4.5.4 STAFF DETAILS 23

TABLE 4.5.5 FEEDBACKDETAILS 24

TABLE 4.5.6 ORDER DETAILS 24

TABLE 4.5.7 LOGIN DETAILS 25

# CHAPTER 1

## INTRODUCTION

* 1. **PROJECTDESCRIPTION**

Inventory management system is the backbone of any business operations. With the development and availability of process driven software applications, inventory management system has gone under revolutionary changes. In any organisations or business

all functions are interlinked or connected to each other and are often overlapping. Some key aspects are supply chain management, logistic and inventory management system form the backbone of business delivery function. Therefore these functions are very-very important to marketing managers as well as finance controllers.

Inventory Management System is very important function that determines the health of supply chain as well as the financial health of balance sheet. Every organisations strives to maintain optimum inventory to be able to meet its requirements and avoid over or under inventory that can impact the financial figures. Inventory is always dynamic it requires constant and careful evaluation of internal factors and external factors. Most of the department have separate department or job functions called inventory planners who continuously monitors, control and review inventory and interface with production, procurement and financial departments.

## PROJECT PURPOSE

This Case study looks provides information to efficiently flow the manage the flow of materials, effectively utilize people and equipment, coordinate internal activities and communicate with customers.

Inventory Management System must tie together the following objectives:

* Company’s strategic goals.
* Sales forecasting.
* Sales and operations planning.
* Production and materials requirement planning.

The main advantage of my system is that it reduces risk of overselling it will simplify inventory management, there will be improved business negotiations etc.

Anticipated Benefits are:

* + No more manual work.
  + The system will help to reduce labour cost involved.
  + Cut cost and increase profit.
  + Enhanced productivity in operations.

## Tools/Environment Used: HARDWARE USED

### PROCESSOR-I56thgeneration RAM- 8 GB

Monitor – HCL LCD Hard disk – 256 GB Key board – HP **SOFTWARE USED**

Visual studio Code - 2020

Operating System – Mac OS

**CHAPTER 2**

**LITERATURE REVIEW**

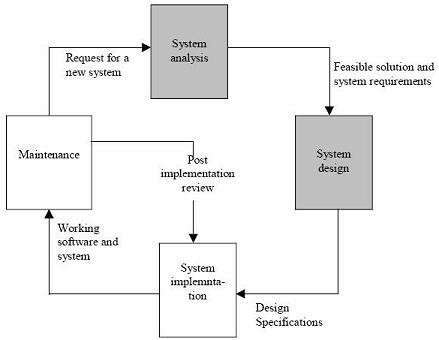
* 1. **System Analysis**

System Analysis by definition is a process of systematic investigation for the purpose of gathering data, interpreting the facts, diagnosing the problem and using this information to either build a completely new system or to recommend the improvements to the existing system.

A satisfactory system analysis involves the process of examining a business situation with the intent of improving it through better methods and procedures. In its core sense, theanalysisphasedefinestherequirementsofthesystemandtheproblemswhichuseris trying to solve irrespective of how the requirements would be accomplished.

System development can generally be thought of having two major components: systems analysis and systems design. In System analysis more emphasis is given to understanding the details of an existing system or a proposed one and then deciding whether the proposedsystemisdesirableornotandwhethertheexistingsystemneedsimprovements. Thus, system analysis is the process of investigating a system, identifying problems, and using the information to recommend improvements to the system.

## Process Design of System Analysis



**Fig. 1**

* 1. **Feasibility Study**

Feasibility is the determination of whether or not a project is worth doing. The process followed in making this determination is called a feasibility study.

**Types of feasibility**

* + 1. **Technical Feasibility: -**It is concerned with the availability of hardware and software required for the development of the system. The technical needs of the system may vary considerable, but might include:
       - The facility to produce outputs in a given time.
       - Response time under certain condition.
       - Ability to process a certain volume of transaction at a particular speed.
       - Facility to communication data to distinct location.
    2. **Operational Feasibility: -** Operational feasibility is all about problems that may arise during operations. There are two aspects related with this issue:-
       - What is the probability that the solution developed may not be put to use or may not work?
       - What is the inclination of the management and end users towards the solution?
    3. **EconomicFeasibility:-**Itisthemeasureofcosteffectivenessoftheproject.The economicfeasibilityisnothingbutjudgingwhetherthepossiblebenefitofsolving the problem is worth right or not.
    4. **Social Feasibility: -** Social feasibility is determined a proposed project will be acceptable to the people or not.
    5. **ManagementFeasibility:-**Thistypeoffeasibilitydeterminesaproposedproject will be acceptable to management. If Managements does not support or gives a negligible support to it. The analyst will tend to view the project as anon-feasible one.
    6. **Legal Feasibility: -** Legal feasibility studies issues arising out of the need to the development of the system. The possible consideration might include copyright law, labor law, antitrust legislation, foreign trade, regulation etc.

## [Fact Finding Techniques](http://systemanalysisanddesign.blogspot.in/2008/11/fact-finding-techniques.html)

To Study any system the analyst needs to do collect facts and all relevant information the facts when expressed in quantitative form are termed as data. The success of any project is depended upon the accuracy of available data. Accurate information can be collected with help of certain methods/ techniques. These specific methods for finding information of the system are termed as fact finding techniques. Interview, Questionnaire, Record View and Observations are the different fact finding techniques used by the analyst. The analyst may use more than one technique for investigation.

**Interview**

This method is used to collect the information from groups or individuals. Analyst selects the people who are related with the system for the interview. In this method the analyst sits face to face with the people and records their responses. The interviewer must plan in advance the type of questions he/ she is going to ask and should be ready to answer any type of question. He should also choose a suitable place and time which will be comfortable for the respondent.

The information collected is quite accurate and reliable as the interviewer can clear and cross check the doubts there itself. This method also helps gap the areas of misunderstandings and help to discuss about the future problems. Structured and unstructured are the two sub categories of Interview. Structured interview is more formal interview where fixed questions are asked and specific information is collected whereas unstructured interview is more or less like a casual conversation where in- depth areas topics are covered and other information apart from the topic may also be obtained.

**Questionnaire**

It is the technique used to extract information from number of people. This method can be adopted and used only by a skillful analyst. The Questionnaire consists of series of questions framed together in logical manner. The questions are simple, clear and to the point. This method is very useful for attaining information from people who are concerned with the usage of the system and who are living in different countries. The questionnaire can be mailed or send to people by post. This is the cheapest source of fact finding.

**Record view**

The information related to the system is published in the sources like newspapers, magazines, journals, documents etc. This record review helps the analyst to get valuable information about the system and the organization.

**On-Site Observation**

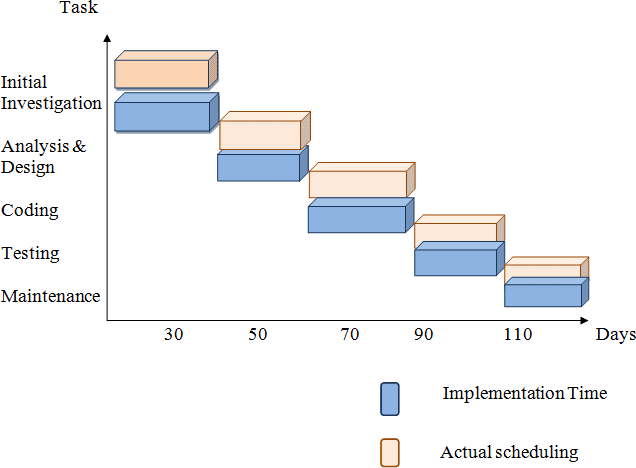
Unlike the other fact finding techniques, in this method the analyst himself visits the organization and observes and understand the flow of documents, working of the existing system, the users of the system etc. For this method to be adopted it takes an analyst to perform this job as he knows which points should be noticed and highlighted. In analyst may observe the unwanted things as well and simply cause delay in the development of the new system.

## Scheduling

* + 1. **Gantt chart:**

Gantt chart is a project control technique that can be used for several purposes including scheduling and planning. Gantt chart is also known bar chart with each box representing an activity.

We estimated the no. of weeks required for each as follows:



**Fig. 2**

# Program Evaluation and Review Technique (PERT)

Like the Gantt chart, PERT makes use of tasks. Like milestone charts, it shows achievements .These achievements however are not task achievements. They are terminal achievement, called events. Arrows are used to represent tasks and circles represent the beginning or completion of a task. The PERT chart uses these paths and events to show the interrelationships of project activities.

**The events in my project can be categorised as:**

1. Meeting to the Employees of company to understand the project.
2. Table Designing
3. Form Designing
4. Writing Codes
5. Designing Reports
6. Testing the project
7. Implementation of project

Each task is limited by an identifiable event. An event has no duration; it simply tells you that the activity has ended or begun. Each task must have a beginning and an ending event. A task can start only after the tasks depends on have been completed.

PERT does not allow “looping back” because a routing that goes back to a task does not end.

# CHAPTER 3

## REQUIREMENT SPECIFICATIONS

* 1. **HARDWAREREQUIREMENTS**
     + RAM: 2GB
     + Operating system: Android or iOS

## SOFTWAREREQUIREMENTS

# SRS of the Project

**FUNCTIONAL & NON-FUNCTIONAL REQUIREMENTS**

**Functional Requirements:**

The System aims at providing an efficient interface to the user for managing of inventory, it shall also provide the user varied options for managing the inventory through various functions at hand. The ingredient levels are continuously monitored based on their usage and are checked for the threshold levels in the inventory and accordingly the user is alerted about low levels of certain ingredients. The design is such that the user does not have to manually update the inventory every time, the System does if for the user. The System calculates and predicts the amount of usage for specific set days that are pre-set by the user (admin) , it also alerts the user of an impending action to order ingredients before the specific day set by the user. Therefore the user never has to worry about manually calculating the estimated usage of the ingredients as the System does it for the user. The simple interface of the System has functions like adding a recipe, removing or updating the recipe. It also extends to functions such as adding a vendor for an ingredient, removing the vendor, checking threshold levels, processing orders, altering processed orders etc. System can make changings in menu like adding or removing food items which are not available.

**Non-Functional Requirements:**

**Usability**

The system must be easy to use by both managers and chefs such that they do not need to read an extensive amount of manuals. The system must be quickly accessible by both managers and chefs. The system must be intuitive and simple in the way it displays all relevant data and relationships. The menus of the system must be easily navigable by the users with buttons that are easy to understand.

**Reliability**

The System must give accurate inventory status to the user continuously. Any inaccuracies are taken care by the regular confirming of the actual levels with the levels displayed in the system. The System must successfully add any recipe, ingredients, vendors or special occasions given by the user and provide estimations and inventory status in relevance with the newly updated entities. The system must provide a password enabled login to the user to avoid any foreign entity changing the data in the system. The system should provide the user updates on completion of requested processes and if the requested processes fail, it should provide the user the reason for the failure. The system should not update the data in any database for any failed processes.

**Performance**

The system must not lag, because the workers using it don’t have down-time to wait for it to complete an action. The system must complete updating the databases, adding of recipe, ingredient, vendor and occasions successfully every time the user requests such a process. All the functions of the system must be available to the user every time the system is turned on. The calculations performed by the system must comply according to the norms set by the user and should not vary unless explicitly changed by the user.

**Supportability**

The software is designed such that it works even on systems having the minimum configuration. The system is adaptable even if additional plugins or modules are added at a later point. The data can be exported to the manager so as to make the system more portable.

**Packaging**

The system must be able to run on the Windows operating systems beginning with Windows XP, and must be able to run on future releases such as the upcoming Windows 8 The software must incorporate a license key authentication process. The packaging must come with a manual that details the use of the system, and also the instructions on how to use the program. This manual may be included either in a booklet that comes with the software, or on the disc that the software itself is on. Implementation The System User Interface is built on Microsoft Visual Studio 2010.

The Database is implemented on the Microsoft Access 2010. The connection between the Database and the System is achieved using ODBC connection available at hand in Visual Studio 2010.

**Interfacing**

The system must offer an easy and simple way of viewing the current inventory. The system must be able to display the relationships between vendors, ingredients, and recipes in an intuitive manner.

**Legal**

The software must be licensed on an individual basis for smaller companies, as well as through a multi-license deal for larger corporations. The client should agree to EULA before using our software.

# CHAPTER 4

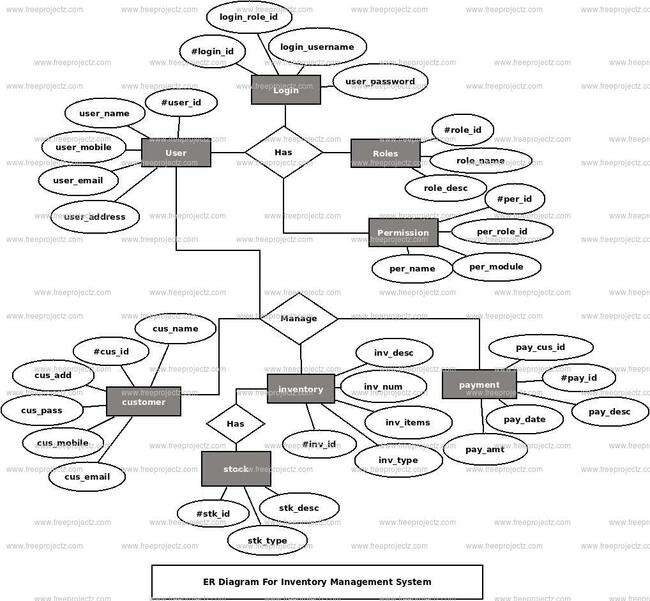
## DESIGN

* 1. **ENITITY RELATIONSHIPDIAGRAM**

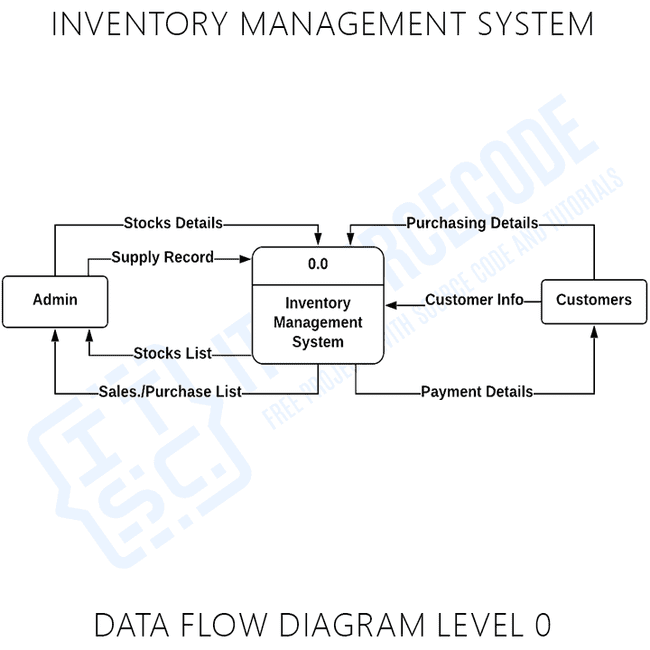
An ER-Diagram is the model that identifies the concept or entities that exist in a system and relationships between those entities. An ERD is often used as a way to visualize a relational database: each entity represents a database table and the relationship lines represents the key in one table that point to specific records in related tables.

Advantages of ER diagram

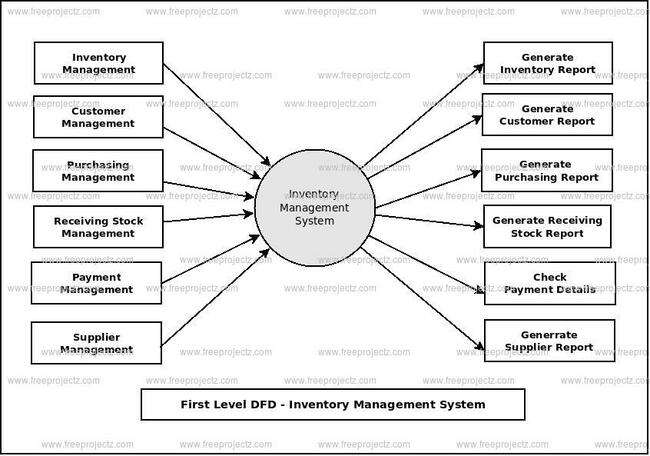
* + - Professional and faster Development.
    - Productivity Improvement.
    - Fewer Faults in Development.
    - Maintenance becomes easy.

 **Fig. 3**

* 1. 0 Level DFD

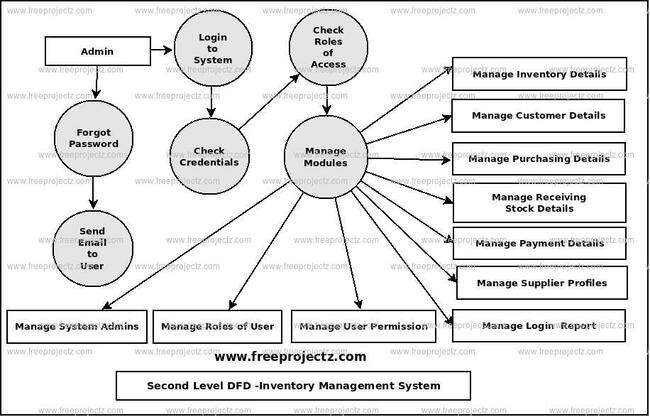


* 1. 1- Level DFD



**Fig. 5**

* 1. 2nd levelDFD



**Fig. 6**

## DATADICTIONARY

* + - **Tables Used**

**1) Customer Details**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Attribute** | **Data Type** | **Size** | **Description** |
| 1. | userID | varchar | 20 | This is the primary key  which stores unique ID of the users |
| 2. | First Name | varchar | 50 | This stores the first name of  the users. |
| 3. | LastName | Varchar | 50 | This stores the last name of the users. |
| 4. | Address | Varchar | MAX | This stores the address of the  users. |
| 5. | EmailId | varchar | 50 | This stores the email address of the users. |
| 6. | MobileNo | integar |  | This stores mobile number of  the users. |
| 9. | Password | Varchar | 50 | This stores password of the  users. |

**TABLE 4.5.1**

1. **Delivery Details**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Attribute** | **Data Type** | **Size** | **Description** |
| 1. | PlacePINCode | Varchar | 20 | This is primary key which stores the unique ID for places. |
| 2. | NameOfPlace | Varchar | 50 | This field stores the name of the place. |
| 3. | Address | Varchar | MAX | This filed stores the address of  the places. |
| 4. | City | varchar | 20 | This field stores the city of the place. |

**TABLE 4.5.2**

1. **Admin Details**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Attribute** | **Data Type** | **Size** | **Description** |
| 1. | AdminID | varchar | 20 | This is the primary key which stores unique ID of the staff/admin. |
| 2. | AdminName | varchar | 50 | This stores the name of the  admin/staff. |
| 3. | Password | varchar | 50 | This stores password of the admin/staff. |
| 4. | Age | int | 10 | This will store age of the  admin/staff. |
| 5. | Address | varchar | MAX | This will store address of the admin/staff. |
| 6. | DOJ | date |  | This will store date of joining  of the admin/staff. |
| 7. | Gender | varchar | 20 | This stores gender of the admin/staff |

**TABLE 4.5.3**

1. **Staff Details**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Attribute** | **Data Type** | **Size** | **Description** |
| 1. | StaffID | varchar | 20 | This is the primary key which store unique ID of guide. |
| 2. | StaffName | varchar | 50 | This will store name of the  guide. |
| 3. | Mobile No. | integer | 10 | This stores mobile number of  the staff. |
| 5. | Address | varchar | MAX | This stores the address of the guide. |
| 6. | Qualification | varchar | 50 | This stores the qualification of the guide. |
| 7. | Age | int | 10 | This stores the age of the guide. |
| 8. | DOJ | varchar |  | This stores gender of the admin/staff |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 9. | Gender | varchar | 20 | This stores gender of the guide. |

**TABLE 4.5.4**

1. **Feedback Details**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Attribute** | **Data Type** | **Size** | **Description** |
| 1. | FeedbackNo. | Varchar | 20 | This is primary key and store unique number for  feedback. |
| 2. | FeedbackMessage | Varchar | MAX | This field store feedback messages from the users. |
| 3. | FeedbackDate | Date |  | This store the date of the  message. |
| 4. | FeedbackTime | Time |  | This store the time of the message. |
| 6. | userID | Varchar | 20 | This is foreign key. |

**TABLE 4.5.5**

1. **Order Details**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Attribute** | **Data Type** | **Size** | **Description** |
| 1. | OrderNo. | varchar | 20 | This is primary key which  stores unique number for queries. |
| 2. | OrderDate | date |  | This stores date of the query. |
| 3. | OrderTime | time |  | This stores time of the query. |
| 5. | userID | varchar | 20 |  |

**TABLE 4.5.6**

1. **Login Details**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Attribute** | **Data Type** | **Size** | **Description** |
| 1. | LoginID | varchar | 20 | This stores the Login ID of users and staff |
| 2. | Password | varchar | 50 | This stores the password of  users and staff. |
| 3. | Role | int |  | This stores Role. |

G©

**TABLE 4.5.7**

# CHAPTER 5

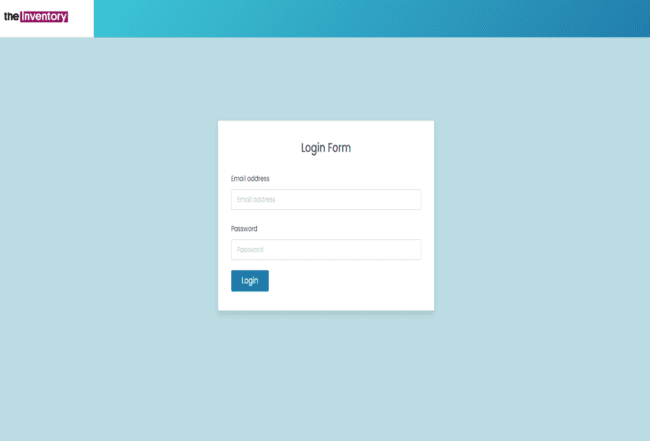
## USER INTERFACES / CODING

* 1. **User Interfaces**

**5.1.1 Splash Screen**

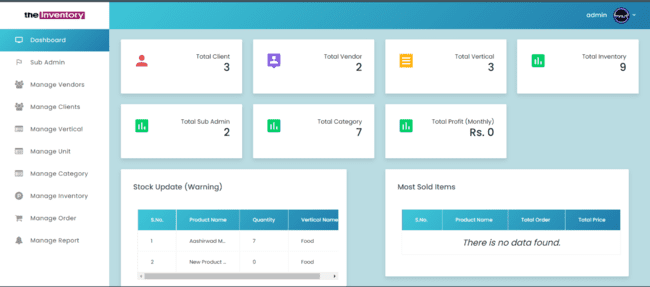
This is our Splash Screen. User get started here.

**5.1.2 Sign In Screen**

**** This is our Sign In Screen. User login our id.

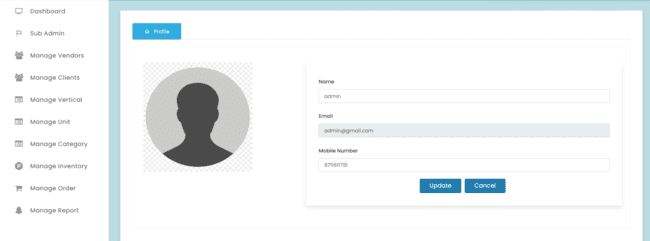
**5.1.3 Home Screen**

This is our Home Screen. User get the list of explore places here.

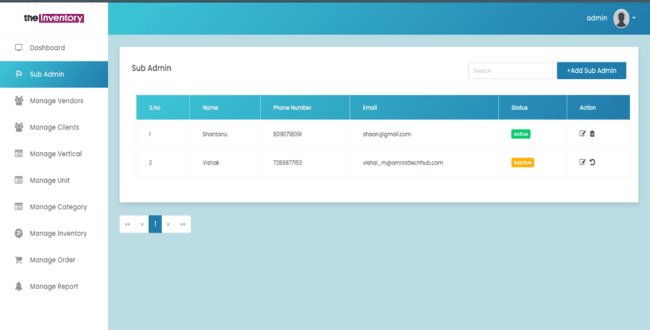


**5.1.4 Profile Screen**

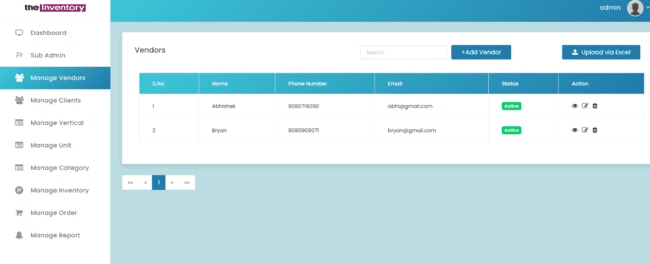
This is our Profile Screen. User check profile here.



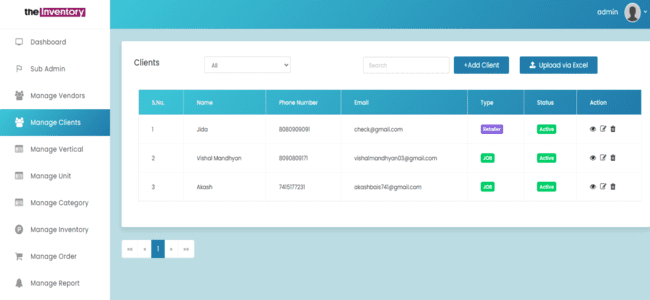
**5.1.5 Sub admin screen**



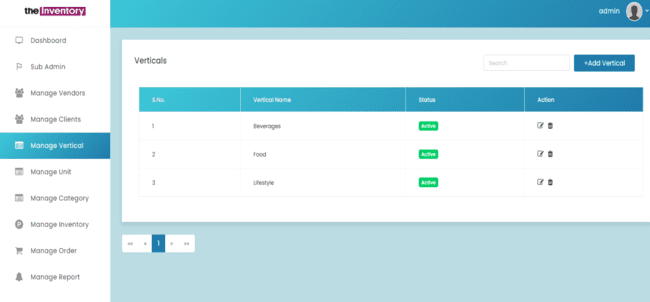
**5.1.6 Manage vendors screen**



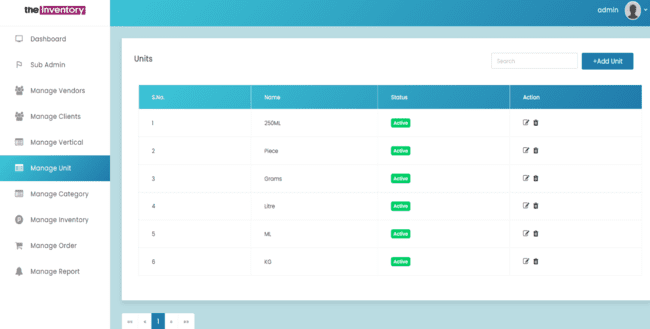
**5.1.7 Manage clients screen**



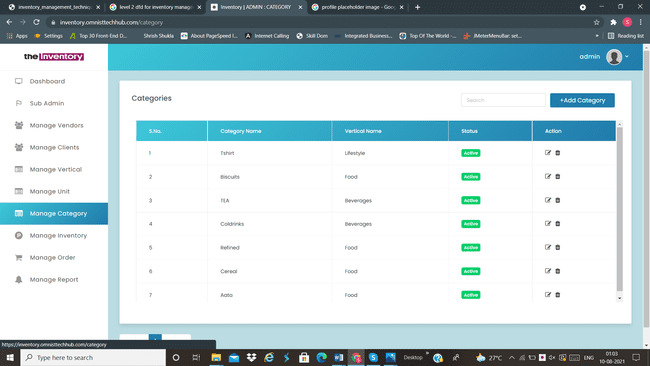
**5.1.8 Manage vertical screen**

.

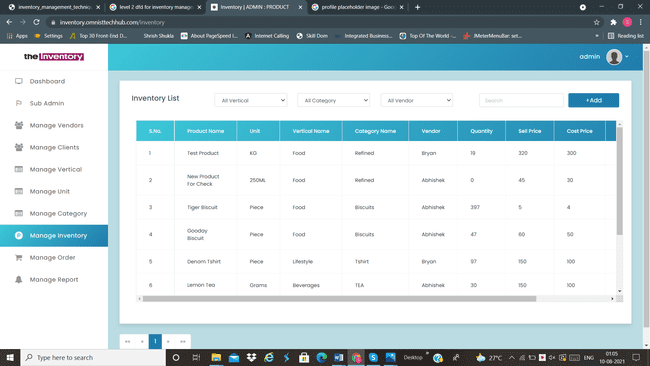
**5.1.9 Manage unit screen**

****

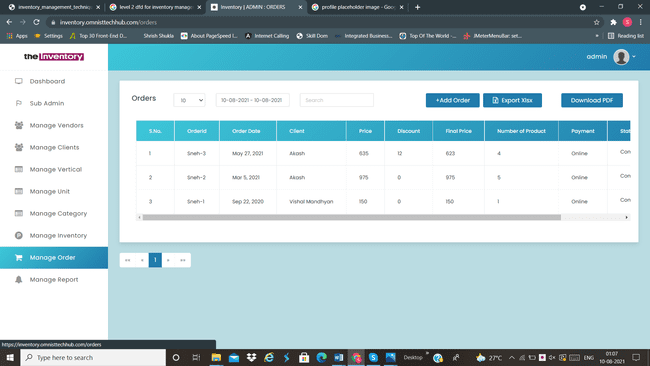
**5.1.10 Manage category screen**



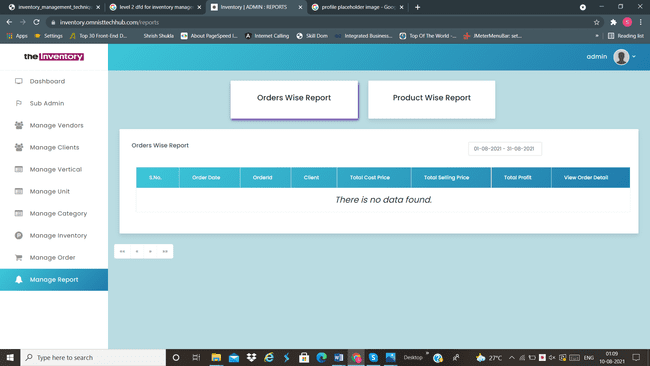
**5.1.11 Manage inventory screen**



**5.1.11 Manage order screen**



**5.1.12 Manage report screen**



* 1. **CodingNode.js**

1. import { StatusBar as ExpoStatusBar } from 'expo-status-bar';
2. import AppLoading from 'expo-app-loading';
3. import React from 'react';
4. import { ThemeProvider } from 'styled-components/native';
5. import \* as firebase from 'firebase';
6. import { useFonts, Oswald\_400Regular } from '@expo-google-fonts/oswald';
7. import { Lato\_400Regular } from '@expo-google-fonts/lato';
8. import { theme } from './src/infrastructure/theme';
9. import { Navigation } from './src/infrastructure/navigation';
10. import { AuthenticationContextProvider } from './src/services/authentication/AuthenticationContext';
11. const firebaseConfig = {
12. apiKey: 'AIzaSyAMOQP9M0--a-l6JDPF5wIXxSHtTd58BLw',
13. authDomain: 'yummeals-e8d3d.firebaseapp.com',
14. projectId: 'yummeals-e8d3d',
15. storageBucket: 'yummeals-e8d3d.appspot.com',
16. messagingSenderId: '572941448591',
17. appId: '1:572941448591:web:9993d5c64e60e32232c7fa',
18. };
19. if (!firebase.apps.length) {
20. firebase.initializeApp(firebaseConfig);
21. }
22. export default function App() {
23. const [fontsLoaded] = useFonts({
24. Oswald\_400Regular,
25. Lato\_400Regular,
26. });
27. if (!fontsLoaded) {
28. return <AppLoading />;
29. } else {
30. return (
31. <>
32. <ThemeProvider theme={theme}>
33. <AuthenticationContextProvider>
34. <Navigation />
35. </AuthenticationContextProvider>
36. </ThemeProvider>
37. <ExpoStatusBar style="auto" />
38. </>
39. );
40. }
41. }

**CompactRestaurantInfo.js**

import React from 'react';

import { Image, View, Platform } from 'react-native';

import { WebView } from 'react-native-webview';

import styled from 'styled-components/native';

import { CustomText as Text } from '../CustomText/CustomText';

const CompactImage = styled(Image)`

border-radius: 10px;

width: 120px;

height: 100px;

`;

const CompactWebview = styled(WebView)`

border-radius: 10px;

width: 120px;

height: 100px;

`;

const Item = styled(View)`

padding: 10px;

max-width: 120px;

align-items: center;

`;

const isAndroid = Platform.OS === 'android';

export const CompactRestaurantInfo = ({ restaurant, isMap }) => {

const CustomImage = isAndroid && isMap ? CompactWebview : CompactImage;

return (

<Item>

<CustomImage source={{ uri: restaurant.photos[0] }} />

<Text center variant="caption">

{restaurant.name}

</Text>

</Item>

);

};

1. {

import React, { useState, useContext } from 'react';

import { ActivityIndicator, Colors } from 'react-native-paper';

import {

AccountBackground,

AccountCover,

AccountContainer,

AuthButton,

AuthInput,

Title,

ErrorContainer,

} from '../components/Account.styles';

import { Spacer } from '../../../components/Spacer/Spacer';

import { CustomText as Text } from '../../../components/CustomText/CustomText';

import { AuthenticationContext } from '../../../services/authentication/AuthenticationContext';

export const LoginScreen = ({ navigation }) => {

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

const { onLogin, isLoading, error } = useContext(AuthenticationContext);

return (

<AccountBackground>

<AccountCover />

<Title variant="caption">YumMeals</Title>

<AccountContainer>

<AuthInput

label="E-mail"

value={email}

textContentType="emailAddress"

keyboardType="email-address"

autoCapitalize="none"

onChangeText={(userEmail) => setEmail(userEmail)}

/>

<Spacer size="large">

<AuthInput

label="Password"

value={password}

textContentType="password"

secureTextEntry

autoCapitalize="none"

onChangeText={(userPass) => setPassword(userPass)}

/>

</Spacer>

{error && (

<ErrorContainer size="large">

<Text variant="error">{error}</Text>

</ErrorContainer>

)}

<Spacer size="large">

{!isLoading ? (

<AuthButton

icon="lock-open-outline"

mode="contained"

onPress={() => onLogin(email, password)}

>

Login

</AuthButton>

) : (

<ActivityIndicator animating={true} color={Colors.blue300} />

)}

</Spacer>

</AccountContainer>

<Spacer size="large">

<AuthButton mode="contained" onPress={() => navigation.goBack()}>

Back

</AuthButton>

</Spacer>

</AccountBackground>

);

};

**CustomText.js**

import { Text } from 'react-native';

import styled from 'styled-components';

const defaultTextStyles = (theme) => `

font-family: ${theme.fonts.body};

font-weight: ${theme.fontWeights.regular};

color: ${theme.colors.text.primary};

flex-wrap: wrap;

margin-top: 0px;

margin-bottom: 0px;

`;

const body = (theme) => `

font-size: ${theme.fontSizes.body};

`;

const hint = (theme) => `

font-size: ${theme.fontSizes.body};

`;

const error = (theme) => `

color: ${theme.colors.text.error};

`;

const caption = (theme) => `

font-size: ${theme.fontSizes.caption};

font-weight: ${theme.fontWeights.bold};

`;

const label = (theme) => `

font-family: ${theme.fonts.heading};

font-size: ${theme.fontSizes.body};

font-weight: ${theme.fontWeights.medium};

`;

const variants = {

body,

hint,

error,

label,

caption,

};

export const CustomText = styled(Text)`

${({ theme }) => defaultTextStyles(theme)};

${({ variant, theme }) => variants[variant](theme)};

`;

Text.defaultProps = {

variant: 'body',

};

**LoginScreen.js**

import React, { useState, useContext } from 'react';

import { ActivityIndicator, Colors } from 'react-native-paper';

import {

AccountBackground,

AccountCover,

AccountContainer,

AuthButton,

AuthInput,

Title,

ErrorContainer,

} from '../components/Account.styles';

import { Spacer } from '../../../components/Spacer/Spacer';

import { CustomText as Text } from '../../../components/CustomText/CustomText';

import { AuthenticationContext } from '../../../services/authentication/AuthenticationContext';

export const LoginScreen = ({ navigation }) => {

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

const { onLogin, isLoading, error } = useContext(AuthenticationContext);

return (

<AccountBackground>

<AccountCover />

<Title variant="caption">YumMeals</Title>

<AccountContainer>

<AuthInput

label="E-mail"

value={email}

textContentType="emailAddress"

keyboardType="email-address"

autoCapitalize="none"

onChangeText={(userEmail) => setEmail(userEmail)}

/>

<Spacer size="large">

<AuthInput

label="Password"

value={password}

textContentType="password"

secureTextEntry

autoCapitalize="none"

onChangeText={(userPass) => setPassword(userPass)}

/>

</Spacer>

{error && (

<ErrorContainer size="large">

<Text variant="error">{error}</Text>

</ErrorContainer>

)}

<Spacer size="large">

{!isLoading ? (

<AuthButton

icon="lock-open-outline"

mode="contained"

onPress={() => onLogin(email, password)}

>

Login

</AuthButton>

) : (

<ActivityIndicator animating={true} color={Colors.blue300} />

)}

</Spacer>

</AccountContainer>

<Spacer size="large">

<AuthButton mode="contained" onPress={() => navigation.goBack()}>

Back

</AuthButton>

</Spacer>

</AccountBackground>

);

};

## AccountScreen.js

import React from 'react';

import LottieView from 'lottie-react-native';

import {

AccountBackground,

AccountCover,

AccountContainer,

AuthButton,

Title,

} from '../components/Account.styles';

import { Spacer } from '../../../components/Spacer/Spacer';

import { LottieAnimationWrapper } from '../../../components/LottieAnimationWrapper/LottieAnimationWrapper';

export const AccountScreen = ({ navigation }) => {

return (

<AccountBackground>

<AccountCover />

<LottieAnimationWrapper>

<LottieView

key="animation"

resizeMode="cover"

autoPlay

loop

source={require('../../../../assets/animations/watermelon-lottie.json')}

/>

</LottieAnimationWrapper>

<Title variant="caption">YumMeals</Title>

<AccountContainer>

<AuthButton

icon="lock-open-outline"

mode="contained"

onPress={() => navigation.navigate('Login')}

>

Login

</AuthButton>

<Spacer size="large">

<AuthButton

icon="email"

mode="contained"

onPress={() => navigation.navigate('Register')}

>

Register

</AuthButton>

</Spacer>

</AccountContainer>

</AccountBackground>

);

};

**Sub adminScreen.js**

import React, { useState, useContext } from 'react';

import { ActivityIndicator, Colors } from 'react-native-paper';

import {

AccountBackground,

AccountCover,

AccountContainer,

AuthButton,

AuthInput,

Title,

ErrorContainer,

} from '../components/Account.styles';

import { Spacer } from '../../../components/Spacer/Spacer';

import { CustomText as Text } from '../../../components/CustomText/CustomText';

import { AuthenticationContext } from '../../../services/authentication/AuthenticationContext';

export const RegisterScreen = ({ navigation }) => {

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

const [repeatedPassword, setRepeatedPassword] = useState('');

const { onRegister, isLoading, error } = useContext(AuthenticationContext);

return (

<AccountBackground>

<AccountCover />

<Title variant="caption">YumMeals</Title>

<AccountContainer>

<AuthInput

label="E-mail"

value={email}

textContentType="emailAddress"

keyboardType="email-address"

autoCapitalize="none"

onChangeText={(userEmail) => setEmail(userEmail)}

/>

<Spacer size="large">

<AuthInput

label="Password"

value={password}

textContentType="password"

secureTextEntry

autoCapitalize="none"

onChangeText={(userPass) => setPassword(userPass)}

/>

</Spacer>

<Spacer size="large">

<AuthInput

label="Repeat Password"

value={repeatedPassword}

textContentType="password"

secureTextEntry

autoCapitalize="none"

onChangeText={(userPass) => setRepeatedPassword(userPass)}

/>

</Spacer>

{error && (

<ErrorContainer size="large">

<Text variant="error">{error}</Text>

</ErrorContainer>

)}

<Spacer size="large">

{!isLoading ? (

<AuthButton

icon="email"

mode="contained"

onPress={() => onRegister(email, password, repeatedPassword)}

>

Register

</AuthButton>

) : (

<ActivityIndicator animating={true} color={Colors.blue300} />

)}

</Spacer>

</AccountContainer>

<Spacer size="large">

<AuthButton mode="contained" onPress={() => navigation.goBack()}>

Back

</AuthButton>

</Spacer>

</AccountBackground>

);

};

**styles.js**

import { ImageBackground, View } from 'react-native';

import { Button, TextInput } from 'react-native-paper';

import styled from 'styled-components/native';

import { colors } from '../../../infrastructure/theme/colors';

import { CustomText as Text } from '../../../components/CustomText/CustomText';

export const AccountBackground = styled(ImageBackground).attrs({

source: require('../../../../assets/images/homepage\_bg.jpg'),

})`

flex: 1;

align-items: center;

justify-content: center;

`;

export const AccountCover = styled(View)`

position: absolute;

width: 100%;

height: 100%;

background-color: rgba(255, 255, 255, 0.3);

`;

export const AccountContainer = styled(View)`

background-color: rgba(255, 255, 255, 0.5);

padding: ${(props) => props.theme.space[4]};

margin-top: ${(props) => props.theme.space[2]};

`;

export const AuthButton = styled(Button).attrs({

color: colors.brand.primary,

})`

padding: ${(props) => props.theme.space[2]};

`;

export const AuthInput = styled(TextInput)`

width: 300px;

`;

export const Title = styled(Text)`

font-size: 30px;

`;

export const ErrorContainer = styled(View)`

max-width: 300px;

align-items: center;

align-self: center;

margin-top: ${(props) => props.theme.space[2]};

margin-bottom: ${(props) => props.theme.space[2]};

`;

**styles.js**

import styled from 'styled-components/native';

import {

Avatar,

ActivityIndicator,

Colors,

TextInput,

Button,

} from 'react-native-paper';

import { colors } from '../../../infrastructure/theme/colors';

export const CartIconContainer = styled.View`

align-items: center;

justify-content: center;

flex: 1;

`;

export const CartIcon = styled(Avatar.Icon).attrs({

size: 128,

})`

background-color: ${(props) => props.bg || props.theme.colors.brand.primary};

`;

export const PaymentProcessing = styled(ActivityIndicator).attrs({

size: 128,

animating: true,

color: Colors.blue300,

})`

position: absolute;

top: 50%;

left: 35%;

z-index: 999;

`;

export const NameInput = styled(TextInput)`

margin: ${(props) => props.theme.space[3]};

`;

export const PayButton = styled(Button).attrs({

color: colors.brand.primary,

})`

width: 80%;

align-self: center;

padding: ${(props) => props.theme.space[2]};

`;

export const ClearButton = styled(Button).attrs({

color: colors.ui.error,

})`

width: 80%;

align-self: center;

padding: ${(props) => props.theme.space[2]};

`;

**CreditCardInput.js**

import React from 'react';

import { LiteCreditCardInput } from 'react-native-credit-card-input';

import { cardTokenRequest } from '../../../services/checkout/checkout.service';

export const CreditCardInput = ({ name, onSuccess, onError }) => {

const onChange = async (formData) => {

const { values, status } = formData;

const isIncomplete = Object.values(status).includes('incomplete');

const expiry = values.expiry.split('/');

const card = {

number: values.number,

exp\_month: expiry[0],

exp\_year: expiry[1],

cvc: values.cvc,

name: name,

};

if (!isIncomplete) {

try {

const info = await cardTokenRequest(card);

onSuccess(info);

} catch (err) {

onError();

}

}

};

return <LiteCreditCardInput onChange={onChange} />;

};

**MapCallout.js**

import React from 'react';

import { CompactRestaurantInfo } from '../../../components/CompactRestaurantInfo/CompactRestaurantInfo';

export const MapCallout = ({ restaurant }) => (

<CompactRestaurantInfo restaurant={restaurant} isMap />

);

**Search.js**

import React, { useContext, useState, useEffect } from 'react';

import { View } from 'react-native';

import { Searchbar } from 'react-native-paper';

import styled from 'styled-components/native';

import { LocationContext } from '../../../services/location/LocationContext';

const SearchContainer = styled(View)`

padding: ${(props) => props.theme.space[3]};

position: absolute;

z-index: 999;

top: 40px;

width: 100%;

`;

export const Search = () => {

const { keyword, search } = useContext(LocationContext);

const [searchKeyword, setSearchKeyword] = useState(keyword);

useEffect(() => {

setSearchKeyword(keyword);

}, [keyword]);

return (

<SearchContainer>

<Searchbar

placeholder="Search for a location"

icon="map"

value={searchKeyword}

onSubmitEditing={() => search(searchKeyword)}

onChangeText={(text) => {

setSearchKeyword(text);

}}

/>

</SearchContainer>

);

};

**MapScreen.js**

import React, { useContext, useState, useEffect } from 'react';

import MapView from 'react-native-maps';

import styled from 'styled-components/native';

import { LocationContext } from '../../../services/location/LocationContext';

import { RestaurantsContext } from '../../../services/restaurants/RestaurantsContext';

import { Search } from '../components/Search';

import { MapCallout } from '../components/MapCallout';

const Map = styled(MapView)`

height: 100%;

width: 100%;

`;

const RestaurantMap = ({ navigation }) => {

const { location } = useContext(LocationContext);

const { restaurants = [] } = useContext(RestaurantsContext);

const [latDelta, setLatDelta] = useState(0);

const { lat, lng, viewport } = location;

useEffect(() => {

const northeastLat = viewport.northeast.lat;

const southwestLat = viewport.southwest.lat;

setLatDelta(northeastLat - southwestLat);

}, [location, viewport]);

return (

<>

<Search />

<Map

region={{

latitude: lat,

longitude: lng,

latitudeDelta: latDelta,

longitudeDelta: 0.01,

}}

>

{restaurants.map((restaurant) => {

return (

<MapView.Marker

key={restaurant.name}

title={restaurant.name}

coordinate={{

latitude: restaurant.geometry.location.lat,

longitude: restaurant.geometry.location.lng,

}}

>

<MapView.Callout

onPress={() =>

navigation.navigate('RestaurantDetail', { restaurant })

}

>

<MapCallout restaurant={restaurant} />

</MapView.Callout>

</MapView.Marker>

);

})}

</Map>

</>

);

};

export const MapScreen = ({ navigation }) => {

const { location } = useContext(LocationContext);

if (!location) {

return (

<Map

region={{

latitude: 0,

longitude: 0,

}}

/>

);

}

return <RestaurantMap navigation={navigation} />;

};

import React from 'react';

import { View } from 'react-native';

import { SvgXml } from 'react-native-svg';

import { Spacer } from '../../../components/Spacer/Spacer';

import { CustomText as Text } from '../../../components/CustomText/CustomText';

import { Favourite } from '../../../components/Favourite/Favourite';

import star from '../../../../assets/icons/star';

import open from '../../../../assets/icons/open';

import {

RestaurantCard,

RestaurantCardCover,

Address,

Info,

Section,

Rating,

OperationStatus,

Icon,

} from './RestaurantInfoCard.styles';

export const RestaurantInfoCard = ({ restaurant = {} }) => {

const {

name = 'Sick Eats',

icon = 'https://maps.gstatic.com/mapfiles/place\_api/icons/v1/png\_71/lodging-71.png',

photos = [

'https://images.unsplash.com/photo-1504674900247-0877df9cc836?ixid=MnwxMjA3fDB8MHxzZWFyY2h8MXx8Zm9vZHxlbnwwfHwwfHw%3D&ixlib=rb-1.2.1&auto=format&fit=crop&w=500&q=60',

],

address = '100 random sick avenue',

isOpenNow = true,

rating = 4,

isClosedTemporarily = true,

placeId,

} = restaurant;

const ratingArray = Array.from(new Array(Math.floor(rating)));

return (

<RestaurantCard elevation={2}>

<View>

<Favourite restaurant={restaurant} />

<RestaurantCardCover key={name} source={{ uri: photos[0] }} />

</View>

<Info>

<Text variant="label">{name}</Text>

<Section>

<Rating>

{ratingArray.map((\_, index) => (

<SvgXml

key={`star-${placeId}-${index}`}

xml={star}

width={20}

height={20}

/>

))}

</Rating>

<OperationStatus>

{isClosedTemporarily && (

<Text variant="error">CLOSED TEMPORARILY</Text>

)}

<Spacer position="left" size="large">

{isOpenNow &&<SvgXml xml={open} width={20} height={20} />}

</Spacer>

<Spacer position="left" size="large">

<Icon source={{ uri: icon }} />

</Spacer>

</OperationStatus>

</Section>

<Address>{address}</Address>

</Info>

</RestaurantCard>

);

};

**styles.js**

import { Image, View, Text } from 'react-native';

import { Card } from 'react-native-paper';

import styled from 'styled-components/native';

export const RestaurantCard = styled(Card)`

background-color: ${(props) => props.theme.colors.bg.primary};

width: 95%;

align-self: center;

`;

export const RestaurantCardCover = styled(Card.Cover)`

padding: ${(props) => props.theme.space[3]};

background-color: ${(props) => props.theme.colors.bg.primary};

`;

export const Address = styled(Text)`

font-family: ${(props) => props.theme.fonts.body};

font-size: ${(props) => props.theme.fontSizes.caption};

`;

export const Info = styled(View)`

padding: ${(props) => props.theme.space[3]};

`;

export const Section = styled(View)`

flex-direction: row;

align-items: center;

`;

export const Rating = styled(View)`

flex-direction: row;

padding-top: ${(props) => props.theme.space[2]};

padding-bottom: ${(props) => props.theme.space[2]};

`;

export const OperationStatus = styled(View)`

flex: 1;

flex-direction: row;

justify-content: flex-end;

`;

export const Icon = styled(Image)`

width: 15px;

height: 15px;

`;

**List.styles.js**

import { FlatList } from 'react-native';

import { Button } from 'react-native-paper';

import { colors } from '../../../infrastructure/theme/colors';

import styled from 'styled-components/native';

export const RestaurantList = styled(FlatList).attrs({

contentContainerStyle: {

padding: 16,

},

})``;

export const OrderButton = styled(Button).attrs({

color: colors.brand.primary,

})`

padding: ${(props) => props.theme.space[2]};

width: 80%;

align-self: center;

`;

import React, { useContext, useState, useEffect } from 'react';

import { View } from 'react-native';

import { Searchbar } from 'react-native-paper';

import styled from 'styled-components/native';

import { LocationContext } from '../../../services/location/LocationContext';

const SearchContainer = styled(View)`

padding: ${(props) => props.theme.space[3]};

background-color: ${(props) => props.theme.colors.brand.primary};

`;

export const Search = ({ isFavouritesToggled, onFavouritesToggle }) => {

const { keyword, search } = useContext(LocationContext);

const [searchKeyword, setSearchKeyword] = useState(keyword);

useEffect(() => {

setSearchKeyword(keyword);

}, [keyword]);

return (

<SearchContainer>

<Searchbar

placeholder="Search for a location"

value={searchKeyword}

icon={isFavouritesToggled ? 'heart' : 'heart-outline'}

onIconPress={onFavouritesToggle}

onSubmitEditing={() => search(searchKeyword)}

onChangeText={(text) => {

setSearchKeyword(text);

}}

/>

</SearchContainer>

);

};

import React, { useContext, useRef, useState, useEffect } from 'react';

import { View, TouchableOpacity } from 'react-native';

import { Camera } from 'expo-camera';

import AsyncStorage from '@react-native-async-storage/async-storage';

import styled from 'styled-components/native';

import { CustomText as Text } from '../../../components/CustomText/CustomText';

import { AuthenticationContext } from '../../../services/authentication/AuthenticationContext';

const ProfileCamera = styled(Camera)`

width: 100%;

height: 100%;

flex: 1;

`;

const InnerSnap = styled(View)`

width: 100%;

height: 100%;

z-index: 999;

`;

export const CameraScreen = ({ navigation }) => {

const { user } = useContext(AuthenticationContext);

const [hasPermission, setHasPermission] = useState(null);

const cameraRef = useRef();

const snap = async () => {

if (cameraRef) {

const photo = await cameraRef.current.takePictureAsync();

AsyncStorage.setItem(`${user.uid}-photo`, photo.uri);

navigation.goBack();

}

};

useEffect(() => {

(async () => {

const { status } = await Camera.requestPermissionsAsync();

setHasPermission(status === 'granted');

})();

}, []);

if (hasPermission === null) {

return <View />;

}

if (hasPermission === false) {

return <Text>No access to camera</Text>;

}

return (

<ProfileCamera

ref={(camera) => (cameraRef.current = camera)}

type={Camera.Constants.Type.front}

>

<TouchableOpacity onPress={snap}>

<InnerSnap />

</TouchableOpacity>

</ProfileCamera>

);

};

**FavouritesScreen.js**

import React, { useContext } from 'react';

import { TouchableOpacity } from 'react-native';

import styled from 'styled-components/native';

import LottieView from 'lottie-react-native';

import { SafeArea } from '../../../components/SafeArea/SafeArea';

import { Spacer } from '../../../components/Spacer/Spacer';

import { CustomText as Text } from '../../../components/CustomText/CustomText';

import { RestaurantList } from '../../restaurants/components/RestaurantList.styles';

import { RestaurantInfoCard } from '../../restaurants/components/RestaurantInfoCard';

import { LottieAnimationWrapper } from '../../../components/LottieAnimationWrapper/LottieAnimationWrapper';

import { FavouritesContext } from '../../../services/favourites/FavouritesContext';

const NoFavouritesArea = styled(SafeArea)`

align-items: center;

justify-content: center;

`;

export const FavouritesScreen = ({ navigation }) => {

const { favourites } = useContext(FavouritesContext);

return favourites.length ? (

<SafeArea>

<RestaurantList

data={favourites}

renderItem={({ item }) => {

return (

<TouchableOpacity

onPress={() =>

navigation.navigate('RestaurantDetail', { restaurant: item })

}

>

<Spacer position="bottom" size="large">

<RestaurantInfoCard restaurant={item} />

</Spacer>

</TouchableOpacity>

);

}}

keyExtractor={(item) => item.name}

/>

</SafeArea>

) : (

<NoFavouritesArea>

<LottieAnimationWrapper>

<LottieView

key="animation"

resizeMode="cover"

autoPlay

loop

source={require('../../../../assets/animations/heartbreak-lottie.json')}

/>

</LottieAnimationWrapper>

<Text variant="label">No favourites yet</Text>

</NoFavouritesArea>

);

};

**SettingsScreen.js**

import React, { useContext, useState, useCallback } from 'react';

import { ImageBackground, TouchableOpacity, View } from 'react-native';

import { List, Avatar } from 'react-native-paper';

import { useFocusEffect } from '@react-navigation/native';

import AsyncStorage from '@react-native-async-storage/async-storage';

import styled from 'styled-components/native';

import { SafeArea } from '../../../components/SafeArea/SafeArea';

import { Spacer } from '../../../components/Spacer/Spacer';

import { CustomText as Text } from '../../../components/CustomText/CustomText';

import { colors } from '../../../infrastructure/theme/colors';

import { AuthenticationContext } from '../../../services/authentication/AuthenticationContext';

const SettingsBackground = styled(ImageBackground).attrs({

source: require('../../../../assets/images/homepage\_bg.jpg'),

})`

position: absolute;

height: 100%;

width: 100%;

`;

const TransparentSafeArea = styled(SafeArea)`

background-color: transparent;

`;

const SettingsItem = styled(List.Item)`

padding: ${(props) => props.theme.space[3]};

background-color: rgba(255, 255, 255, 0.4);

`;

const AvatarContainer = styled(View)`

align-items: center;

`;

export const SettingsScreen = ({ navigation }) => {

const { onLogout, user } = useContext(AuthenticationContext);

const [photo, setPhoto] = useState(null);

const getProfilePicture = async (currentUser) => {

const photoUri = await AsyncStorage.getItem(`${currentUser.uid}-photo`);

setPhoto(photoUri);

};

useFocusEffect(

useCallback(() => {

getProfilePicture(user);

}, [user])

);

return (

<SettingsBackground>

<TransparentSafeArea>

<AvatarContainer>

<TouchableOpacity onPress={() => navigation.navigate('Camera')}>

{!photo && (

<Avatar.Icon

size={180}

icon="account-circle"

backgroundColor={colors.brand.primary}

/>

)}

{photo && (

<Avatar.Image

size={180}

source={{ uri: photo }}

backgroundColor={colors.brand.primary}

/>

)}

</TouchableOpacity>

<Spacer position="top" size="large">

<Text variant="label">{user.email}</Text>

</Spacer>

</AvatarContainer>

<List.Section>

<SettingsItem

title="Favourites"

description="View your favourites"

onPress={() => navigation.navigate('Favourites')}

left={(props) => (

<List.Icon {...props} color={colors.ui.error} icon="heart" />

)}

/>

<Spacer />

<SettingsItem

title="Logout"

onPress={onLogout}

left={(props) => (

<List.Icon {...props} color={colors.ui.secondary} icon="logout" />

)}

/>

</List.Section>

</TransparentSafeArea>

</SettingsBackground>

);

};

**CHAPTER 7**

**LIMITATION AND FUTURE SCOPE**

* 1. **Limitation**
     1. High cost implementation.
     2. Greater complexity.

# FUTURE SCOPE

1. We can add new features as and when require.
2. Reusability of this system is also possible.
3. We can give cab and hotel facilities to the users.
4. We can give weather conditions of every place so as to make sure that the user will be comfortable to visit the desired place.
5. We can charge from user for providing guide.

We can give location of the explore places on map also.

## REFERENCES:-

* 1. [Svetlin Nakov,](https://www.amazon.in/s/ref%3Ddp_byline_sr_ebooks_1?ie=UTF8&field-author=Svetlin%2BNakov&text=Svetlin%2BNakov&sort=relevancerank&search-alias=digital-text) [Veselin Kolev,](https://www.amazon.in/s/ref%3Ddp_byline_sr_ebooks_2?ie=UTF8&field-author=Veselin%2BKolev&text=Veselin%2BKolev&sort=relevancerank&search-alias=digital-text)[Nakov's Team,](https://www.amazon.in/s/ref%3Ddp_byline_sr_ebooks_3?ie=UTF8&field-author=Nakov%27s%2BTeam&text=Nakov%27s%2BTeam&sort=relevancerank&search-alias=digital-text) “*Fundamentals of Computer Programming with C#: Programming Principles, Object- Oriented Programming, Data Structures” ,* Faber Publishing (19 March2019).
  2. Fundamentals of Java Programming “<https://econsortium.aktu.ac.in/?p=47417>”
  3. Start Programming Using HTML, CSS, and JavaScript“<https://econsortium.aktu.ac.in/?p=44914>”
  4. HTML5 “<https://econsortium.aktu.ac.in/?p=46277>”
  5. HTML5 Mobile Websites“<https://econsortium.aktu.ac.in/?p=50847>”
  6. Mastering C++“<https://econsortium.aktu.ac.in/?p=53548>”
  7. Learning Java with Games “<https://econsortium.aktu.ac.in/?p=47927>”
  8. Java in Two Semesters “<https://econsortium.aktu.ac.in/?p=48102>”
  9. Introduction to Software Design with Java “<https://econsortium.aktu.ac.in/?p=48665>”
  10. A Functional Start to Computing with Python“<https://econsortium.aktu.ac.in/?p=43948>”
  11. A Beginners Guide to Python 3 Programming “<https://econsortium.aktu.ac.in/?p=48751>”
  12. Introduction to Python for Science and Engineering “https://econsortium.aktu.ac.in/?p=44568”
  13. [Sarah Martin](https://www.researchgate.net/scientific-contributions/Sarah-Martin-2135151873) , ”Programming using HTML ,CSS and Javascript” in<https://www.researchgate.net/publication/321043238_HTML_CSS_and_JavaScript>, November 2017 ,DOI: [10.1007/978-1-4842-2937- 8\_5](http://dx.doi.org/10.1007/978-1-4842-2937-8_5)
  14. [Dick Oliver](https://www.amazon.in/Dick-Oliver/e/B001IXOKC2/ref%3Ddp_byline_cont_book_1) ,[Michael Morrison](https://www.amazon.in/Michael-Morrison/e/B000AQ2H3C/ref%3Ddp_byline_cont_book_2) , “HTML5” in ”*Sams Teach Yourself HTML and CSS in 24 Hours” ,* Sams; 7th edition (14 December 2005).
  15. .Microsoft Visual studio 2015 Unleashed Publication(ThirdEdition)
  16. Kanchan Naik , “Introduction to software design in c#” , [https://www.c-sharpcorner.com/UploadFile/bd5be5/design-patterns- in-net/](https://www.c-sharpcorner.com/UploadFile/bd5be5/design-patterns-in-net/) ,Dec 12,2020
  17. [Kalman Toth](https://www.amazon.in/Kalman-Toth/e/B00A7TKLQC/ref%3Ddp_byline_cont_ebooks_1) , “Database design with SQL” , in SQL Server 2012, Database Design , 31 December2012
  18. PANKAJ JALOTE , “Concept of software engineering”, in An Integrated Approach to Software Engineering , DOI:10.1007/978-1- 4684-9312-2 , EditionNumber:2

**Thank you**