**A PROJECT REPORT**

**ON**

**CANTEEN AUTOMATION ANDROID APPLICATION**

***Submitted by***

**PARTH DADWAL (1/16/FET/BCG/1/007)**

**JATIN VASHISHT (1/16/FET/BCG/1/003)**

**AKSHAY GUPTA (1/16/FET/BCG/1/018)**

**SARTHAK AGARWAL (1/16/FET/BCF/1/054)**

***Under the Guidance of***

**Mrs. SHWETA SHARMA**

**ASSOCIATE PROFESSOR**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**



**Faculty of Engineering & Technology**

**Manav Rachna International Institute of Research and Studies, Faridabad**,

**MAY, 2020**

**ACKNOWLEDGEMENT**

The successful realization of project is an outgrowth of a consolidated effort of people from desperate fronts. We are thankful to **Mrs. SHWETA SHARMA** (Associate Professor) for her variable advice and support extended to us without which we could not be able to complete our project for a success.

We are thankful to **Dr. Poonam Tanwar,** Project Coordinator, Associate Professor, CSE department for her guidance and support.

We express our deep gratitude to Dr. **Brijesh** Kumar Head of Department (CSE, IBM Program) for his endless support and affection towards us. His constant encouragement has helped to widen the horizon of our knowledge and inculcate the spirit of dedication to the purpose.

We would like to express our sincere gratitude to Prof. (Dr.) **Harish C. Rai**, Dean, FET for providing us the facilities in the Institute for completion of our work.

Words cannot express our gratitude for all those people who helped us directly or indirectly in our Endeavour. We take this opportunity to express our sincere thanks to all staff members of CSE department for the valuable suggestion and also to our family and friends for their support.

Parth Dadwal (1/16/FET/BCG/1/007) Sign.

Jatin Vashisht (1/16/FET/BCG/1/003) Sign.

Akshay Gupta (1/16/FET/BCG/1/018) Sign.

Sarthak Agarwal (1/16/FET/BCF/1/054) Sign.

**APPENDIX 3**

**Declaration**

We hereby declare that this project report entitled “**CANTEEN AUTOMATION ANDROID APPLIATION**” *by* **PARTH DADWAL (1/16/FET/BCG/1/007); JATIN VASHISHT (1/16/FET/BCG/1/003); AKSHAY GUPTA (1/16/FET/BCG/1/018) and** **SARTHAK AGARWAL (1/16/FET/BCF/1/054),** being submitted in partial fulfillment of the requirements for the degreeof Bachelor of Technology in **Computer Science & Engineering** under Faculty of Engineering & Technology of Manav Rachna International Institute of Research and Studies, Faridabad, during the academic year 2019-2020, is a bonafide record of our original work carried out under the guidance of **Mrs. SHWETA SHARMA, ASSOCIATE PROFESSOR , FET.**

We further declare that we have not submitted the matter presented in this Project for the award of any other Degree/Diploma of this University or any other University/Institute.

1. Parth Dadwal (1/16/FET/BCG/1/007)
2. Jatin Vashisht (1/16/FET/BCG/1/003)
3. Akshay Gupta (1/16/FET/BCG/1/018)
4. Sarthak Agarwal (1/16/FET/BCF/1/054)

**Appendix 4**



**Manav Rachna International Institute of Research and Studies, Faridabad**

**Faculty of Engineering & Technology**

**Department of Computer Science and Engineering**

May, 2020

**Certificate**

This is to certify that this project report entitled “**CANTEEN AUTOMATION ANDROID APPLICATION**” *by* **PARTH DADWAL (1/16/FET/BCG/1/007); JATIN VASHISHT (1/16/FET/BCG/1/005); AKSHAY GUPTA (1/16/FET/BCG/1/018) and** **SARTHAK AGARWAL (1/16/FET/BCF/1/054),** submitted in partial fulfillment of the requirements for the degreeof Bachelor of Technology in **Computer Science & Engineering** under Faculty of Engineering & Technology of Manav Rachna International Institute of Research and Studies, Faridabad, during the academic year 2019-2020, is a bonafide record of work carried out under my guidance and supervision. I hereby declare that the work has been carried out under my supervision and has not been submitted elsewhere for any other purpose.

**(Signature of Project Guide) (Signature of HoD)**

Mrs. Shweta Sharma Dr. Brijesh Kumar

DESIGNATION Head of Department

Department of Computer Science and Engineering Department. of Computer Science and Engineering

FET, MRIIRS, Faridabad FET, MRIIRS, Faridabad

**Appendix 5**

**TABLE OF CONTENTS**

**Declaration ii**

**Certificate iii**

**Acknowledgement iv**

**Abstract v**

**List of Tables vi**

**List of Figures vii**

**Chapter 1. Introduction 1**

* 1. Purpose of Project
  2. Project and Product Overviews
     1. How it’s benefits a Canteen
     2. How it benefits a Student
  3. Team Architecture
  4. Overall Description
     1. Product Perspective
     2. Problem Statement
  5. Purpose of Project
  6. Feasibility
  7. Objective

**Chapter 2. Requirements**

* 1. Specific Requirements
     1. External Interface Requirements
  2. Hardware Requirements
  3. Software Requirements
  4. Functional Requirements
     1. Menu management system
     2. Order Retrieval System
  5. Non-Functional Requirements

**Chapter 3. Design Analysis**

1. Designing techniques
   1. Java
   2. XML
   3. C++
   4. Tier Architecture
   5. Software Process Model
      1. Why not evolutionary models?
      2. Why not Waterfall Model?
      3. Why Incremental RAD Model?
   6. Observation
   7. Feasibility Analysis
2. **Chapter 4. Design**
   1. Login
      1. Use-Case diagram for login page
      2. Use-Case diagram for Registration
      3. Android Code for Login Page & Registration
      4. Class Diagram for Login
      5. Class Diagram for Registration
      6. Sequence Diagram for Login
      7. Sequence Diagram for Registration
      8. Output for Login
      9. Output for Registration Page
   2. Menu Screen and Home Screen
      1. Code for Menu Screen
      2. Menu Screen
      3. Code for Home Screen
      4. Home Screen
   3. Order update and payment
      1. Use-Case Diagram for order Updation and payment
      2. Class Diagram for order Updation and payment
      3. Sequence Diagram for payment
      4. Code for payment and order receipt
      5. Order Updation and payment Screen
   4. User Profile
      1. User Profile Screen
   5. Code Token Generation
3. **Chapter 5. ER Diagrams**
   1. Data Flow Diagram
4. **Chapter 6. Database Tables**
   1. Customer
   2. Admin
   3. Canteen Owner
   4. Menu
   5. Order
   6. Payment

**References/Bibliography**

**Appendices**

* Project Code
* Progress Report by Guide
* Plagiarism Report (not more than 5% from single source)

**Appendix 6**

**LIST OF TABLES**

**Table**  **Page No**

Table 6.1 Customer Data Table

Table 6.2 Admin Data Table

Table 6.3 Canteen Owner Data Table

Table 6.4 Menu Data Table

Table 6.5 Order Data Table

Table 6.6 Payment Data Table

**Appendix 7**

**LIST OF FIGURES**

<Font Style Times New Roman, size -16 Bold>

**Figure Page No.**

Figure 4 Use-Case for Login Page

Figure 4.1 Use-Case Diagram for Registration

Figure 4.2 Class Diagram for Login Page

Figure 4.3 Class Diagram for Registration

Figure 4.4 Sequence Diagram for Registration

Figure 4.5 Sequence Diagram for Registration

Figure 4.6 Login Screen

Figure 4.7 Registration Screen

Figure 4.8 Menu Screen

Figure 4.9 Home Screen

Figure 4.10 Use-Case for Order, payment & update

Figure 4.11 Class Diagram for Order and Update

Figure 4.12 Class Diagram for payment

Figure 4.13 Sequence diagram for Payment

Figure 4.14 Sequence Diagram of Placing order

Figure 4.15 Sequence Diagram for Updation

Figure 4.16 Order Placement

Figure 4.17 Payment Screen

Figure 4.18 User Profile Screen

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_X\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Appendix 8**

**ABSTRACT**

The Project "Canteen Automation System" empowers the end clients to enlist on the web, read and select the food from e-menu card and request food online by simply choosing the food that the client need to have utilizing android application. The outcomes subsequent to choosing the food from the E-menu card will legitimately show up on the screen close to the Chef who is going to prepare the nourishment for you. The framework is the blend of Android just as android Application. By utilizing this application, crafted by the server is decreased and we can likewise say that the work is invalidated. The advantage of this is on the off chance that there is a surge in the Canteen, at that point there will be chances that the servers will be inaccessible and the clients can straightforwardly arrange the food to the culinary expert online by utilizing this application. The client will have a username and a secret key, by utilizing which they can login into the framework. This suggests the client is the customary client of the Canteen.

The manual framework includes administrative work through keeping up different documents and manuals. Keeping up basic data in the documents and manuals is loaded with chance and a dreary procedure. Counting a system telling the best way to apply Internet innovation dynamically as aptitudes and certainty develop, the task exhibits the course from adjusting materials to building up an online situation.

These days individuals don't have a lot of time to spend in canteen by just there and trusting that the server will take their request. Numerous clients visit the flask in their mid-day break and break so they have constrained time to eat and come back to their particular office and universities. So this product causes them to spare time and request food at whatever point they need without calling the server over and over.

**CHAPTER 1: INTRODUCTION**

**1. INTRODUCTION**

In the present period of quick canteen mechanization in the canteen, numerous canteens have decided to concentrate on fast arrangement and expedient conveyance of requests. Until as of late, the entirety of this conveyance of requests were put via telephone, yet there are numerous disservices to this framework, including the burden of the client expecting to have a physical duplicate of the menu, absence of a visual affirmation that the request was put accurately, and the need for the canteen to have a representative picking up the telephone and taking requests.

The primary bit of leeway of a web based requesting framework is that it extraordinarily improves the requesting procedure for both the client and the canteen. At the point when the client visits the requesting page of the application, they are given an intelligent and state-of-the-art menu, complete with every single accessible alternative and modifying costs dependent on the chosen choices. Subsequent to making a determination, the thing is then added to their request, which the client can audit the subtleties whenever before looking at. This gives moment visual affirmation of what was chosen.

This framework additionally incredibly alleviates the burden on the canteen's end, as the whole procedure of taking requests is mechanized. When a request is put on the application, it is gone into the database and afterward recovered, in basically constant, by an android put together application with respect to the canteen's end. Inside this application, all things in the request are shown, alongside their relating choices and conveyance subtleties, in a compact and simple to understand way. This permits canteen workers to rapidly experience the requests as they are put and produce the essential things with insignificant deferral and disarray.

* 1. **PURPOSE OF PROJECT**

Canteen Automation System is where clients request their food and get food in the canteen immediately as they can legitimately proceed to gather what they requested without hanging tight for a turn or holding up time. This framework expects to quicken client requests and client request framework utilized by workers to acknowledge client request.

The motivation behind the framework is to build up a basic Canteen Automation System and actualize it, which later will be utilized for an online application.

The proposed "Canteen Automation System" is financially achievable on the grounds that:

The framework requires exceptionally less time factors when contrasted with manual framework

The framework will give quick and productive computerized condition rather than moderate and blunder inclined manual framework, along these lines decreasing both time and labor spent in running the framework.

The framework will have GUI interface and less client preparing is required to learn it.

* 1. **PROJECT & PRODUCT OVERVIEWS**

A total canteen robotization framework makes canteen laborers screen by and large everyday business examination all the more precisely no sweat. It expands the nearness of canteen captivates center clients towards your food business prompting expanded deals.

By getting all necessities one spot canteen mechanization framework benefits both the client just as the canteen proprietor sagaciously. Eventually all business activity will be frantic simpler and solitary with the necessary inbuilt highlights.

The working of canteen robotization framework is like a web based business site. At whatever point your clients are occupied with their work, rather than going to your canteen they can simply open your site, pick the menu they like and can just request food.

When request gets set, you will be advised promptly, with the goal that you can begin request handling. One of the primary points of interest of canteen robotization framework is the request subtleties are exact when contrasted with the telephone requesting framework or manual framework.

* + - 1. **HOW ITS BENEFITS A CANTEEN**

Canteen Mechanization framework spares the canteen's time by keeping away from the food arranges via telephone that must be done physically. By making the requesting procedure completely mechanized, it expands the cost adequacy and efficiency of the canteen with a less labor. What's more, it keeps you one stride in front of your rivals who don't serve on the web.

Canteen mechanization framework even helps an easygoing eating canteen and gives an extra income source. It permits a canteen proprietor to effectively refresh the online menu, food things and so on and assists with keeping in contact with the clients by offering limits and focused on advancements.

With an improved administration process, web based requesting framework makes it simple to deal with different canteens from brought together application

**HOW IT BENEFITS A STUDENT**

Presently a day's break time for understudies isn't sufficient to arrange, pause, eat, and have their time. This canteen the board application spares understudies time without a doubt. Permit them to request and line food online by making an adaptable requesting stage and serve them in time. The understudy can choose the food things from the online menu on the canteen application and can request and put their request in a line appropriately according to their advantage. Truth be told, they can put in request the food requests inside their financial plan by including or evacuating the food things according to the cost varieties

This canteen robotization application is valuable for the canteen also client since canteen mechanization framework spares the hour of the canteen's laborers too of the clients and as recoveries the labor.

* 1. **TEAM ARCHITECTURE**

There were four individuals in our undertaking. The total task is separated into two groups.

Group 1:

Necessity GATHERING and PLANNING

Arranging recognizes every single deliverable assistance, depicts the offices, and characterizes the attempting to give insights regarding lab subtleties to the resources. This arranging issue begins with a determination of client request that will be met by the creation plan. For this specific circumstance, the offices gave to the client resources are the significant worry to be concentrated. This incorporates the least demanding and proficient approach to make them accessible an abandoned lab. Therefore, all the fields must be kept while creating. For the most part, an arranging issue exists on the grounds that there are constrained creation assets that can't be put away from period to period. Decisions must be made with regards to which assets to incorporate and how to display their ability and conduct, and their expenses. Likewise, there might be vulnerability related with the creation work, for example, questionable yields or lead times. One may just remember the most basic or restricting asset for the arranging issue. Here comes the job of legitimate necessity gathering. On the off chance that the necessities of the client are clear, at that point it is very simpler for the engineer to satisfy his all needs. As he can investigate all the necessary assets, and with appropriate arranging and cost estimation, he can accomplish his product.

Along these lines, a legitimate arranging and necessity gathering prompts a proficient programming framework

Group 2:

Structuring AND BACKEND

Structuring is the most significant and the most effective capacity while programming advancement. Without a legitimate plan, it is hard to build up a suitable programming that satisfies about all client requests. In this way, overseeing planning part in an association is a basic movement. A creator needs to guarantee that the plan made by him can be effortlessly comprehended by the all the individuals from building up a group. A legitimate structure will permit the coder to actualize the framework improvement arranging appropriately. So also, the backend part that incorporates the database the board assumes a key job in any of the frameworks. Along these lines, the group taking a shot at this field must know appropriate taking care of and the board of database and its apparatuses.

* 1. **OVERALL DESCRIPTION**

Efficient: Our canteen computerization framework is created with an essential point of 'Sparing Time. The client can arrange the food and it is likewise proficient for canteen laborers since this framework takes lesser time when contrasted with telephone based or physically based framework.

No Complication: Major confusion part for canteen robotization framework is including an item or overseeing items area. In our canteen computerization framework, no confused part is associated with overseeing areas; we have taken monstrous consideration in this segment and invalidated all entanglements which make this framework exact and special.

Practical: It's less expensive. You don't need to buy various duplicates of programming to introduce on different PCs. Different duplicates frequently expect you to pay various permitting charges, however since you aren't really buying any product with an online framework, that is not a worry.

Security: Online frameworks are similarly as secure. Most online canteen robotization framework programs permit you to make different client accounts with different degrees of access. Your information is put away on secure, ensured servers that highlight firewalls and other online security programs.

* + 1. **PRODUCT PERSPECTIVE**
* The framework will be created utilizing JAVA, C++, XML and other android advancement instruments.
* Item Functions
* The item evolved will give a simple access to the client to oversee and modify the canteen computerization as indicated by the need.
* General Constraints
* The cost imperative incorporates requesting extra equipment to run the new framework
* No present site
* Security must be updated for on-line looking.
* Suspicions
* The framework will bolster all programs.
* The processor must be at any rate Dual cored and can be some other most recent ones and so on.

* + 1. **PROBLEM STATEMENT**

The Challenges experienced by the manual framework in canteens is productivity and consumer loyalty. The experience of requesting in most cheap food canteens isn't wonderful for clients. Clients need to make long lines before putting in the request and when the request is set, they need to hold up close to the counter until the request is readied. The another issue is proficiency that food canteen ought to keep up in their standard tasks and keep with the nature of their item and administrations regardless of how much group is available in canteen yet they need to keep up effectiveness just as nature of item .in any case, we believe that there are some issue concerning the customary method to arrange food in canteens.

The significant issues are as per the following

Verbal correspondence among clerk and client or we can say telephonic correspondence: The verbal correspondence between two gatherings for putting in a request and the data about bill ought to likewise bring about blunder implies mistake additionally happens in understanding what the individual needs to state and particularly in occupied hours in canteens. At the point when the spot is extremely packed and uproarious, miscommunications are normal. The issue is even Worsen if the clerk or potentially the clients don't communicate in the local language.

Food customization: The capacity to modify food has been a critical business system in the food business since McDonald's trademark "I'm Lovin' it" for the crusade. Food customization permits some adaptability for clients who have a unique solicitation (less ice for pop, no pickles and so on.) and is particularly essential for clients who are hypersensitive to certain fixings .as of now, the clerk needs to remember these solicitations since the current framework doesn't bolster any methods for recording such solicitation. The outcome is that the solicitation are overlooked or miscommunication to the cook.

Menu show: Today's opposition between food canteens persuades every canteen to dispatch new things on their menus on a progressively visit premise. Nonetheless, the menu in more canteens normally connected to a divider behind the counter and the client don't know about that new thing on the grounds that the menu isn't cutting-edge. Moreover, not all thing on the menu list has a graphical delineation to assist clients with taking Decision effectively what they need. The print is here and there little that it's not obvious to the clients.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_X\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CHAPTER 2: REQUIREMENTS**

1. **SPECIFIC REQUIREMENTS** 
   * 1. **EXTERNAL INTERFACE REQUIREMENTS**

The outer framework is to accept full accountability for capacity works just as distribution center administration and stockroom control for a whole stockroom. The interfaces in this area are determined by reporting: the name and depiction of each plan, source or information, goal or yield, reaches, exactness and resiliences, units of measure, timing, show arrangements, and association, and information positions.

The UI required to be created for the framework ought to be easy to understand and appealing. The interface between the client and the framework will be WIMP (Windows, Icons, Menu, Pointers) remembering that the framework is to be gone through an internet browser. All activities will be off point and snap nature with all routes performed through windows of the framework explicitly fastens and menus:

Catches: The catch is actuated when the client will tap on the left snap of the mouse inside the limits of the catch. Furthermore, in this way, the activity related with it will be done.

Menu: All the activities will be organized.

* 1. **HARDWARE REQUIREMENT**

To be able to use the canteen Management App for online logs:

* Standard Internal memory on any device
* 2GB RAM
* Any standard mobile device Processor
* Input Devices: Touch pad
* Output Devices: Mobile device screen

* 1. **SOFTWARE REQUIREMENT**

* Operating System: Android Oreo or higher

* 1. **FUNCTIONAL REQUIREMENTS**

Clients of the canteen computerization framework, in particular canteen clients, must be given the accompanying usefulness:

1. Make a record.
2. Deal with their record.
3. Sign into the framework.
4. Explore the canteen's menu.
5. Select a thing from the menu.
6. Add a thing to their present request.
7. Survey their present request.
8. Create a token for the request on client's enlisted versatile number.
9. Expel a thing/expel all things from their present request.
10. After the endorsement of undertaking for additional advancement including financing:
11. Give installment subtleties.
12. Put in a request.
13. Get affirmation as a request number.
    * 1. **Menu Management System**

The menu the executives framework will be accessible just to canteen representatives and will, as the name proposes, permit them to deal with the menu that is shown to clients of the web requesting framework. The capacities managed by the menu the executives framework give the client the capacity to, utilizing a graphical interface:

1)Add another/update/erase food classification to/from the menu.

2)Add another/update/erase food thing to/from the menu.

3)Add another/update/erase choice for a given food thing.

4)Update default choices for a given food thing.

5)Update extra data (depiction, photograph, and so forth.) for a given food thing.

* + 1. **Order Retrieval System**

Of the three parts, the request recovery framework is practically the least complex. Like the menu the executives framework, it is intended to be utilized distinctly by canteen workers, and gives the accompanying capacities:

1. Recover new requests from the database.
2. Show the requests in an effectively intelligible, graphical way.
3. Imprint a request as having been handled and expel it from the rundown of dynamic requests.
4. Produce a token for the recovered request.

UI Specifications:

1. Every one of the framework segments will have their own remarkable interface. These are portrayed underneath.
2. Application Ordering System

1. Clients of the Application requesting framework will cooperate with the application through a progression of straightforward structures. Every class of food has its own structure related with it which presents a drop-down menu for picking which explicit thing from the classification ought to be added to the request.

Adding a thing to the request is cultivated by a solitary catch contact. Clients select which classification of food they might want to arrange, and in this manner which structure ought to be shown, by exploring a menu bar, a methodology which ought to be recognizable to most clients.

Entering conveyance and installment bargains is done along these lines. The client is given a structure and should finish the necessary fields, which incorporate the two drops down and message boxes, before looking at and getting an affirmation number.

* 1. **Non-functional Requirements:**

**Execution Criteria:**

**Time:**

The slipped by time between the accommodation of request process between the client and clerk in a canteen ought to be as least as could be expected under the circumstances.

**User-accommodating:**

Our canteen the board application ought to be more easy to understand. The UI ought to be kept basic and uncluttered. Since the distinctive kind of individuals will communicate with this procedure so our venture ought to be anything but difficult to them to comprehend.

**Adaptability:**

Our undertaking should be adaptable to such an extent that at whatever point we need to make changes in it effectively it very well may be finished.

**Extensibility:**

It ought to have the option to oblige the varieties like:

The diverse request ought to be dealt with without any problem.

It ought to be a possibility for money down, pay through card among client and canteen.

**Versatile:**

Our undertaking should be compact on any stage and accessible on sites effectively and at a quicker speed than others.

**Reusable:**

All the client site pages that are being utilized for client data ought to be effectively get prepared such a large number of clients can connect with us effectively and extremely quick with no data obliterate.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_X\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CHAPTER 3: DESIGN ANALYSIS**

1. **DESIGN TECHNIQUES**

The design of the site has been done using the following technologies:-

* JAVA
* Json
* C++
* XML
* Other necessary languages for android development.

* 1. **JAVA**

Java is a programming language and registering stage initially discharged by Sun Microsystems in 1995. There are bunches of uses and sites that won't work except if you have Java introduced, and more are made each day. Java is quick, secure, and dependable. From workstations to datacenters, game consoles to logical supercomputers, mobile phones to the Internet, Java is all over the place! Therefore, the source code behind the present Web pages is regularly a revolting blend of labels and scripting.

* 1. **XML**

Extensible Markup Language (XML) is a markup language that characterizes a lot of rules for encoding reports in a configuration that is both comprehensible and machine-intelligible. The World Wide Web Consortium's XML 1.0 Specification of 1998 and a few other related determinations—every one of them free open gauges—characterize XML.

The plan objectives of XML underline straightforwardness, all inclusive statement, and ease of use over the Internet It is a printed information design with solid help by means of Unicode for various human dialects. In spite of the fact that the plan of XML centers around reports, the language is broadly utilized for the portrayal of self-assertive information structures, for example, those utilized in web administrations.

A few diagram frameworks exist to help in the meaning of XML-based dialects, while software engineers have created numerous application programming interfaces (APIs) to help the preparing of XML information.

* 1. **C++**

C++ is an elevated level, broadly useful programming language made by Bjarne Stroustrup as an augmentation of the C programming language, or "C with Classes". The language has extended altogether after some time, and present day C++ has object-arranged, nonexclusive, and useful highlights notwithstanding offices for low-level memory control. It is quite often executed as an aggregated language, and numerous merchants give C++ compilers, including the Free Software Foundation, LLVM, Microsoft, Intel, Oracle, and IBM, so it is accessible on numerous stages.

C++ was structured with an inclination toward framework programming and inserted, asset obliged programming and enormous frameworks, with execution, proficiency, and adaptability of utilization as its plan features. C++ has likewise been discovered helpful in numerous different settings, with key qualities being programming framework and asset obliged applications, including work area applications, computer games, servers (for example internet business, Web search, or SQL servers), and execution basic applications (for example phone switches or space tests).

C++ is normalized by the International Organization for Standardization (ISO), with the most recent standard form sanctioned and distributed by ISO in December 2017 (casually known as C++17). The C++ programming language was at first normalized in 1998 as ISO/IEC 14882:1998, which was then corrected by the C++03, C++11 and C++14 guidelines. The current C++17 standard supplants these with new highlights and a developed standard library. Prior to the underlying normalization in 1998, C++ was created by Danish PC researcher Bjarne Stroustrup at Bell Labs since 1979 as an augmentation of the C language; he needed a proficient and adaptable language like C that additionally given significant level highlights to program association. C++20 is the following arranged norm, keeping with the present pattern of another form at regular intervals.

* 1. **TIER ARCHITECTURE.**

The different classes as acquired from the business class graph is classified as follows-

Form of the project Class Class

Application or

Presentation

Layer

Business Layer

or Logical Layer

Data Layer or

Data Access

Layer

The 3-level engineering comprises of three layers:

Introduction Layer - The site or windows structures application is known as the introduction layer. The introduction layer is the most significant layer just on the

grounds that the one everybody sees and employments. Indeed, even with a very much organized business and information layer, if the introduction layer is planned ineffectively, this gives the clients a poor perspective on the framework. The introduction layer is where we configuration utilizing the controls like textbox, names, order catches and so on.

Business Layer - Though a site could converse with the information get to layer legitimately, it generally experiences another layer called the business layer.

This layer is a class which we use to compose the capacity which fills in as a go between to move the information from Application or introduction layer information layer. In the three-level design, we never let the information get to layer to interface with the introduction layer.

This layer is likewise a class where we announce the variable relating to the fields of the database which can be required for the application and cause the properties with the goal that we to can get or set the information utilizing these properties into the factors. These properties are open with the goal that we can get to its qualities.

Probably the best explanation behind reusing rationale is that applications that start off little as a rule develop in usefulness. For example, an organization starts to build up a site, and as they understand their business needs, they later choose to include a shrewd customer application and windows administration to enhance the site. The business layer causes move rationale to a focal layer for "greatest reusability."

Business layer have been introduced having two jobs

1. customer application
2. server segment

A case of Business Layer-The Business layer has elements of which takes the parameters from the model given in the introduction layer .As the client inputs the information esteems, comparing capacities are brought in the business layer which is additionally gone on through the information layer where relating techniques are called and the information is been refreshed.

The business layer is where we compose the capacities which get the information from the application layer and goes through the information get to layer.

Information layer - The key segment to most applications is the information. The information must be served to the introduction layer some way or another. The information layer is a different part whose sole reason for existing is to present the information from the database and return it to the guest. This layer is additionally a class which we use to get or interfere with the information to the database and forward. This layer just connects with the database. We compose the database inquiries or use put away methods to get to the information from the database or to play out any activity to the database.

**Bit of leeway OF 3 TIER ARCHITECTURE**

1. Customer Server engineering is 2-Tier design in light of the fact that the customer doesn't recognize Presentation layer and business layer.
2. The expanding requests on GUI controls made trouble deal with the blend of source code from GUI and Business Logic.
3. Further, Client Server Architecture doesn't bolster enough the Change Management. Let guess that the administration builds the Entertainment charge rate from 4% to 8 %, at that point in the Client-Server case, we need to send an update to every customer and they should refresh simultaneously on a particular time else, we may store invalid or wrong data.
4. The Client-Server Architecture is additionally a weight to arrange traffic and assets. Let us accept that around 500 customers are dealing with an information server then we will have 500 ODBC associations and a few miscreant record sets, which must be shipped from the server to the customers.
5. This arrangement of the application makes the capacity increasingly reusable effectively and it turns out to be too simple to even consider finding the capacities which have been composed beforehand. On the off chance that the software engineer needs to make a further update in the application, at that point he effectively can comprehend the recently composed code and can refresh without any problem.

**HINDRANCES**

1. Increment multifaceted nature/exertion
2. Increasingly hard to assemble 3 level design instead of a 2 level.
3. Purposes of correspondence are multiplied
4. Support devices are as of now lacking for keeping up server libraries.

**3.1 Programming PROCESS MODEL**

**3.1.1 Why not Evolutionary models?**

These models are most appropriate where necessities are fluffy. These models are most appropriate for the frameworks where prerequisites continue evolving. Be that as it may, for our framework prerequisites are completely clear so it isn't achievable to embrace any of the transformative models.

**3.1.2 Why not Waterfall model?**

Cascade model can be embraced in light of the fact that for our situation since prerequisites are known ahead of time yet there are a few confinements of cascade model because of which it isn't achievable to receive:

No parallelism of work.

Tedious

**3.1.3 Why the Incremental RAD model?**

Steady model is fitting where necessities are clear and the advancement time is less. The striking element of the gradual model is that every module can be finished and discharged as and when the prerequisite emerges due to absence of time.

As in our framework, a considerable lot of the modules are not between related so can be discharged in confinement. The client would thus be able to get a vibe of these modules and give his input which can be used for making the product more easy to use and in accordance with the client necessities.

Not just that the cutoff time set for this task is 3 months and we need a high adjustment model and again will focus on parallelism on the grounds that our group will be taking a shot at the distinctive module simultaneously. Also, we will utilize most recent devices, for example, Visio, Project Manager because of which we can work a lot quicker. So investigating every one of these necessities we find Incremental RAD model is most appropriate for our framework since it empowers the advancement group to make a completely useful framework inside a brief timeframe.

**3.2 OBSERVATION**

We have seen that our framework that is Canteen Automation System would be of huge assistance to the customer as presently everything is done physically, which brings about a great deal of time utilization, is blunder inclined and furthermore increments financial weight as installment not paid by the clients. In addition, such a manual arrangement of overseeing canteen is very unstructured. Our framework would be proficient, exact and simple to utilize.

**3.3 POSSIBILITY ANALYSIS**

The possibility study is certainly not an out and out frameworks study. Or maybe, the plausibility study is utilized to accumulate expansive information to settle on a choice on whether to continue with framework study.

Framework venture attainability is surveyed in three head ways:

Monetarily

In fact

Operationally

Monetary Feasibility

The association has assessed the expense of programming and equipment required for the framework including the capacity of information. The advantages anticipated from the framework are concentrated to evaluate the decreased expense because of the new framework.

Specialized Feasibility:

Association has demonstrated an ability to buy all equipment and programming apparatuses which we prescribe to effectively actualize the framework. Subsequently in fact there are no confinements for the improvement of the framework. Therefore, the undertaking is in fact achievable.

Operational Feasibility:

Operational possibility is subject to the people who will utilize the product once it's prepared and introduced for use. The product will have an easy to understand interface which will be a lot of helpful when contrasted with the present manual methodology. In this manner, the venture is operationally doable

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_X\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CHAPTER 4: DESIGN**

* 1. **LOGIN**
     1. **USE-CASE DIAGRAM FOR LOGIN PAGE**

An utilization case is a portrayal of how end-clients will utilize a product code. It depicts an undertaking or a progression of errands that clients will achieve utilizing the product and incorporates the reactions of the product to client activities.

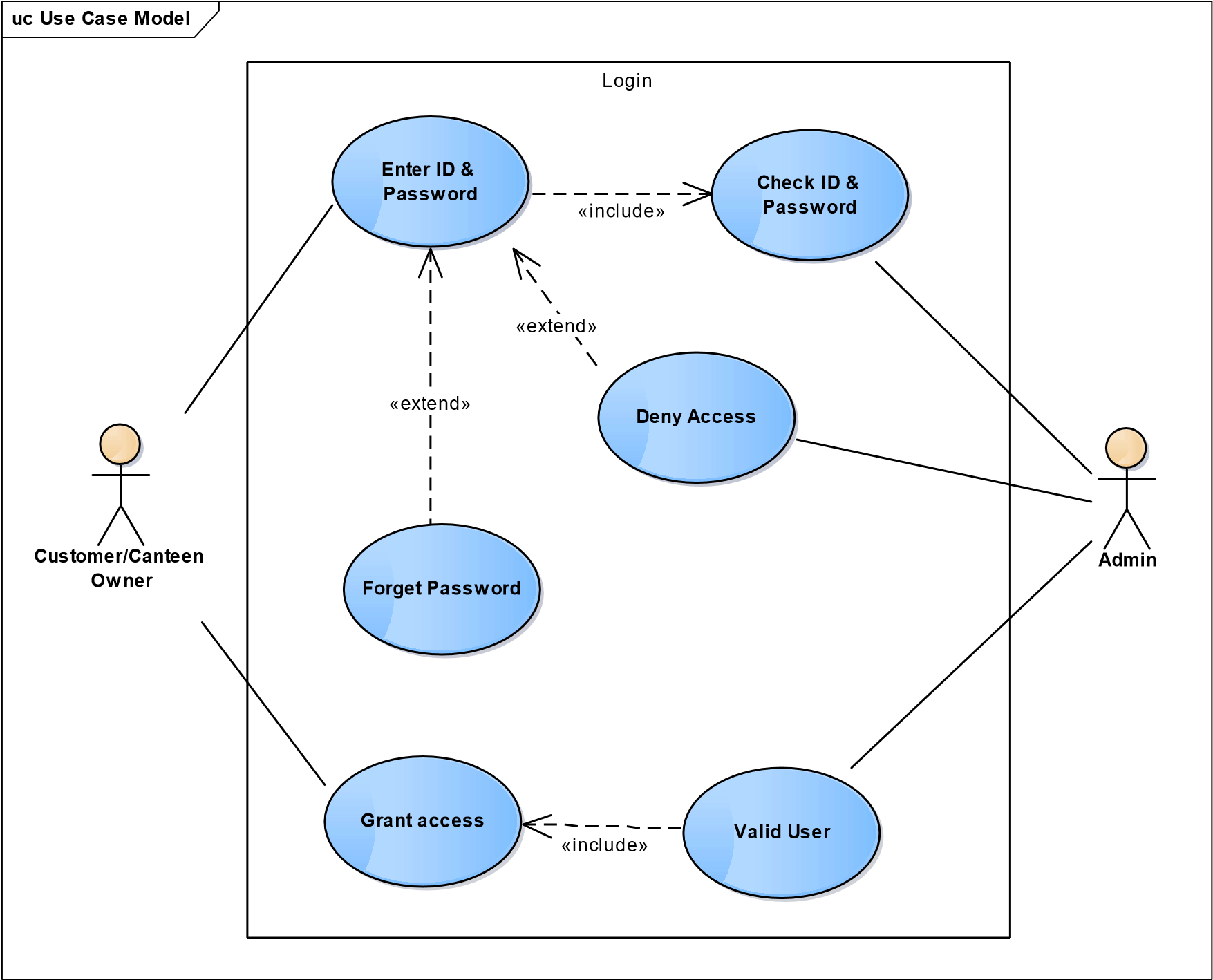


Fig 4. Use case for Login Page

* + 1. **USE CASE DIAGRAM FOR REGISTRATION**

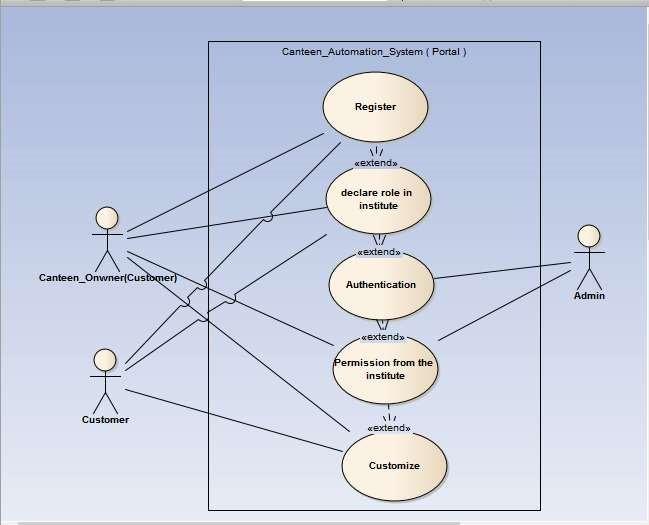


Fig.4.1 Use Case Diagram for Registration

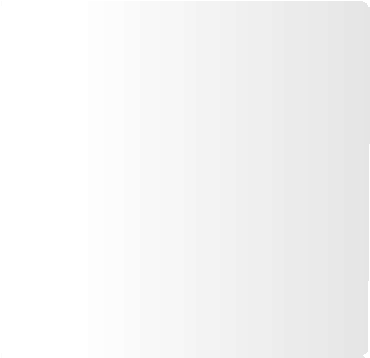
* + 1. **ANDROID CODE FOR LOGIN PAGE & REGISTRATION**

public class LoginActivity extends AppCompatActivity implements View.OnClickListener{  
  
 private LoginActivityDataBinding loginActivityDataBinding;  
 private DatabaseOperations databaseOperations;  
 private static String registeredUserEmailAddress;  
 private static String registeredUserPassword;  
 private final static int GOOGLE\_SIGN\_IN\_CLIENT\_INTENT\_CODE = 234;  
 private ProgressDialog progressDialog;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 loginActivityDataBinding = DataBindingUtil.setContentView(LoginActivity.this, R.layout.activity\_login);  
  
 if (!hasPermission(Manifest.permission.READ\_EXTERNAL\_STORAGE)  
 && !hasPermission(Manifest.permission.VIBRATE)  
 && !hasPermission(Manifest.permission.ACCESS\_FINE\_LOCATION)  
 && !hasPermission(Manifest.permission.INTERNET))  
 ActivityCompat.requestPermissions(this, new String[] {Manifest.permission.READ\_EXTERNAL\_STORAGE,  
 Manifest.permission.RECORD\_AUDIO, Manifest.permission.READ\_EXTERNAL\_STORAGE, Manifest.permission.INTERNET}, 1);  
 else if (!hasPermission(Manifest.permission.READ\_EXTERNAL\_STORAGE))  
 ActivityCompat.requestPermissions(this, new String[] {Manifest.permission.READ\_EXTERNAL\_STORAGE}, 1);  
 else if (!hasPermission(Manifest.permission.VIBRATE))  
 ActivityCompat.requestPermissions(this, new String[] {Manifest.permission.VIBRATE}, 1);  
 else if (!hasPermission(Manifest.permission.ACCESS\_FINE\_LOCATION))  
 ActivityCompat.requestPermissions(this, new String[] {Manifest.permission.ACCESS\_FINE\_LOCATION}, 1);  
 else if (!hasPermission(Manifest.permission.INTERNET))  
 ActivityCompat.requestPermissions(this, new String[] {Manifest.permission.INTERNET}, 1);  
  
 databaseOperations = new DatabaseOperations(LoginActivity.this);  
  
 loginActivityDataBinding.textViewNewUserSignUp.setOnClickListener(this);  
 loginActivityDataBinding.buttonSignIn.setOnClickListener(this);  
 loginActivityDataBinding.googleSignInImageView.setOnClickListener(this);  
 loginActivityDataBinding.textViewResetPassword.setOnClickListener(this);  
  
 loginActivityDataBinding.getRoot();  
 }  
  
 private boolean hasPermission(String permission) {  
 return ActivityCompat.checkSelfPermission(this, permission) == PackageManager.PERMISSION\_GRANTED;  
 }  
  
 private void userSignIn() {  
 registeredUserEmailAddress= loginActivityDataBinding.editTextEmailAddress.getText().toString().trim();  
 registeredUserPassword = loginActivityDataBinding.editTextPassword.getText().toString();  
 Call<LoginModel> call = RetrofitServicesHandler  
 .getInstance()  
 .getApi()  
 .loginUser(registeredUserEmailAddress, registeredUserPassword);  
  
 call.enqueue(new Callback<LoginModel>() {  
 @Override  
 public void onResponse(Call<LoginModel> call, Response<LoginModel> response) {  
 LoginModel userLogin = response.body();  
 if (response.body() != null) {  
 if (userLogin.getMESSAGE().equals("Login Successful")) {  
 UserDetails.id = userLogin.getId();  
 UserDetails.app\_id = userLogin.getApp\_id();  
 UserDetails.token = userLogin.getToken();  
 UserDetails.fullName = userLogin.getFullname();  
 UserDetails.emailAddress = userLogin.getEmail();  
 UserDetails.mobileNUmber = userLogin.getMobile();  
 progressDialog.dismiss();  
 Toast.makeText(LoginActivity.this, "Welcome " + userLogin.getFullname(), Toast.LENGTH\_SHORT).show();  
 Intent intent = new Intent(LoginActivity.this, HomeActivity.class);  
 startActivity(intent);  
 } else {  
 progressDialog.dismiss();  
 Toast.makeText(LoginActivity.this, "No such user exists", Toast.LENGTH\_SHORT).show();  
 }  
 }  
 }  
  
 @Override  
 public void onFailure(Call<LoginModel> call, Throwable t) {  
 Toast.makeText(LoginActivity.this, "Error logging in", Toast.LENGTH\_SHORT).show();  
 }  
 });  
 progressDialog = new ProgressDialog(LoginActivity.this);  
 progressDialog.show();  
 progressDialog.setContentView(R.layout.log\_in\_custom\_progress\_dialog);  
 progressDialog.getWindow().setBackgroundDrawableResource(android.R.color.transparent);  
 }  
  
 private void userSignInWithGoogle() {  
 GoogleSignInClient googleSignInClient;  
 GoogleSignInOptions googleSignInOptions = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT\_SIGN\_IN)  
 .requestIdToken(getString(R.string.default\_web\_client\_id))  
 .requestEmail()  
 .build();  
 googleSignInClient = GoogleSignIn.getClient(LoginActivity.this, googleSignInOptions);  
 Intent intent = googleSignInClient.getSignInIntent();  
 startActivityForResult(intent, GOOGLE\_SIGN\_IN\_CLIENT\_INTENT\_CODE);  
 }  
  
 private void userSignInWithFacebook() {  
  
 }  
  
 private void userSignInWithTwitter() {  
  
 }  
  
 @Override  
 protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);  
 if (requestCode == GOOGLE\_SIGN\_IN\_CLIENT\_INTENT\_CODE) {  
 Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);  
 try {  
 GoogleSignInAccount googleSignInAccount = task.getResult(ApiException.class);  
 databaseOperations.registerUser(null, null, null, googleSignInAccount, null, 2);  
 } catch (ApiException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 private void resetPassword() {  
 ViewGroup viewGroup = findViewById(android.R.id.content);  
 final View dialogView = LayoutInflater.from(LoginActivity.this).inflate(R.layout.reset\_password\_custom\_alert\_dialog\_box, viewGroup, false);  
 AlertDialog.Builder builder = new AlertDialog.Builder(LoginActivity.this);  
 builder.setView(dialogView);  
 final EditText resetRegisteredEmailAddressEditText = dialogView.findViewById(R.id.resetRegisteredEmailAddressEditText);  
 Button resetPasswordButton = dialogView.findViewById(R.id.resetPasswordButton);  
  
 resetPasswordButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 String registeredEmailAddress = resetRegisteredEmailAddressEditText.getText().toString().trim();  
 databaseOperations.passwordResetEmail(registeredEmailAddress);  
 finish();  
 }  
 });  
  
 AlertDialog alertDialog = builder.create();  
 alertDialog.show();  
 }  
  
 @Override  
 public void onClick(View v) {  
 switch (v.getId()) {  
 case R.id.textViewNewUserSignUp:  
 Intent intent = new Intent(LoginActivity.this, MainActivity.class);  
 startActivity(intent);  
 break;  
 case R.id.buttonSignIn:  
 userSignIn();  
 break;  
 case R.id.googleSignInImageView:  
 userSignInWithGoogle();  
 break;  
 case R.id.facebookSignInImageView:  
 userSignInWithFacebook();  
 break;  
 case R.id.twitterSignInImageView:  
 userSignInWithTwitter();  
 break;  
 case R.id.textViewResetPassword:  
 resetPassword();  
 break;  
 }  
 }  
}

* + 1. **CLASS DIAGRAM FOR LOGIN**

In the Unified Modeling Language (UML), a class chart is a kind of static structure graph that portrays the structure of a framework by demonstrating the framework's classes, their traits, and the connections between the classes.

class Class Model



Login UI

User id: char

-

-

Password: char

+

Get\_id\_Password(): boolean

Authenticate(): boolean

+

Enter\_id\_Password(): boolean

+



Customer/Canteen Owner

User id : char

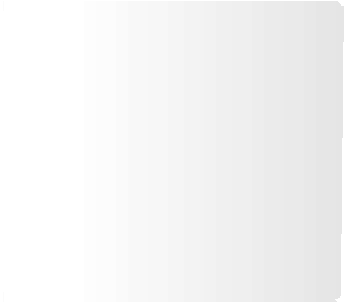
-

-

Password : char

+

Enter\_id\_password(): boolean



Administrator

-

Admin\_id: char

-

Password: char

+

Authenticate(): boolean

+

Get\_id\_password(): boolean

Fig. 4.2 Class diagram of Login Page

* + 1. **CLASS DIAGRAM FOR REGISTRATION**

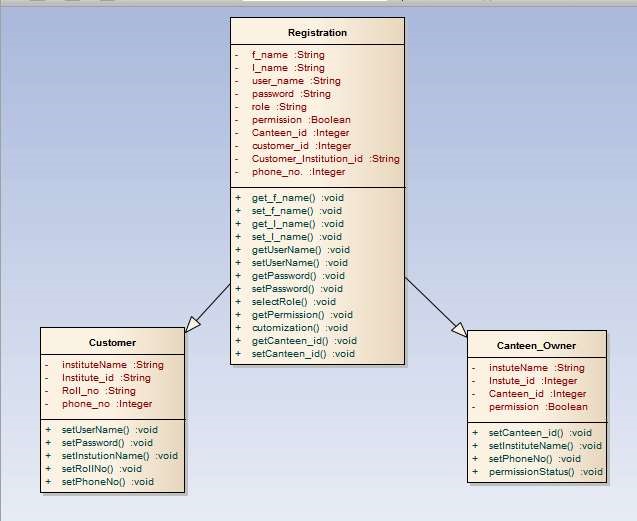


Fig. 4.3 Class Diagram for Registration

* + 1. **SEQUENCE DIAGRAM FOR LOGIN**

An arrangement graph in Unified Modeling Language (UML) is a sort of collaboration chart that shows how procedures work with each other and in what request. It is a build of a Message Sequence Chart. Succession charts are now and then called occasion outlines.

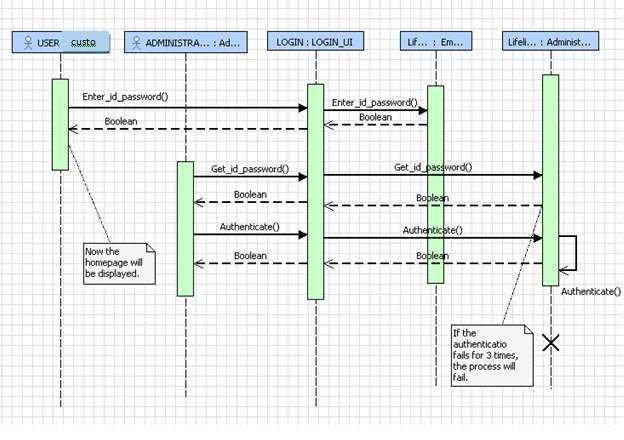


Fig. 4.4 Sequence of Login

* + 1. **SEQUENCE DIAGRAM FOR REGISTRATION**

An arrangement graph in Unified Modeling Language (UML) is a sort of cooperation outline that shows how procedures work with each other and in what request. It is a develop of a Message Sequence Chart. Grouping outlines are some of the time called occasion charts, occasion situations, and timing graphs.

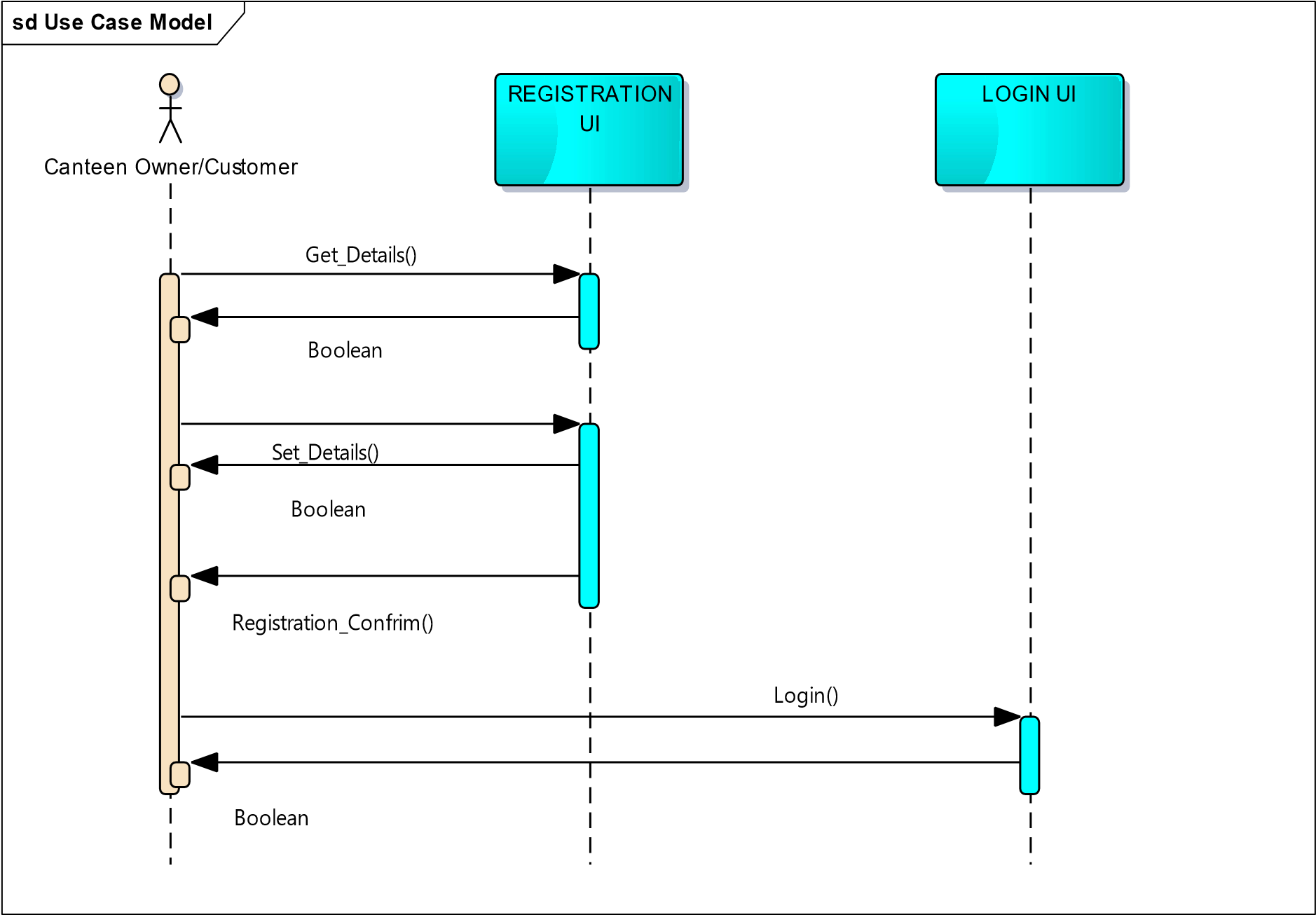


Fig. 4.5 Sequence Diagram for Registration

* + 1. **OUTPUT FOR LOGIN**

Below is the login screen as can be seen on the app

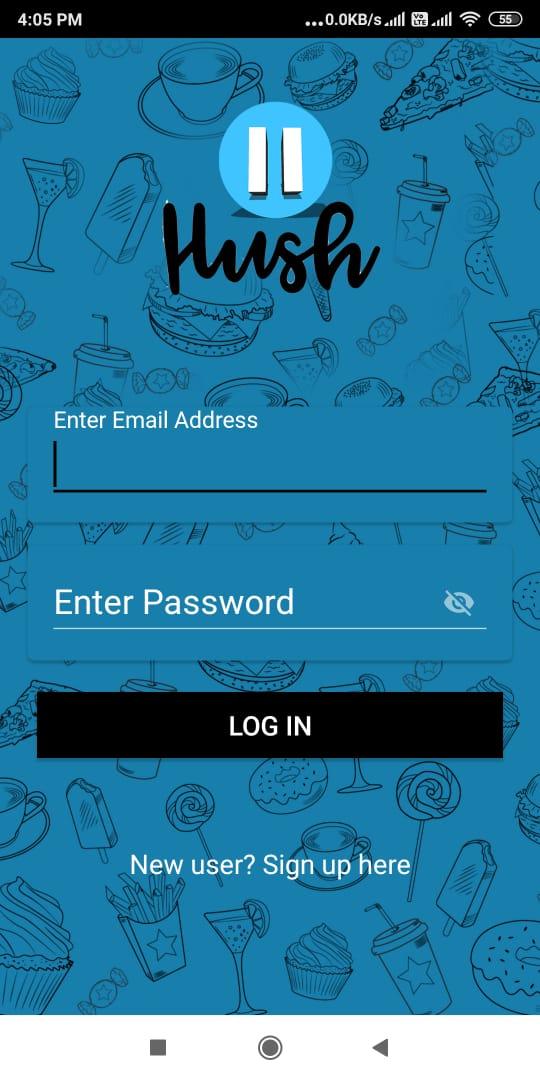
****

Fig. 4.6 Login Screen

* + 1. **OUTPUT FOR REGISTRATION PAGE**

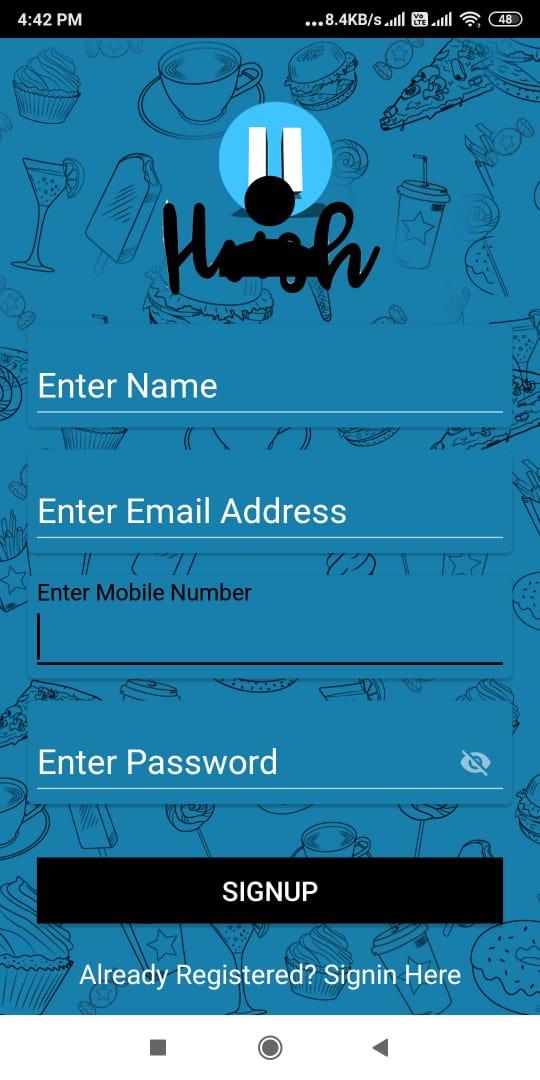
****

Fig. 4.7 Registration Screen

* 1. **MENU SCREEN AND HOME SCREEN**

For the order surfing intuitive and descriptive menu and home screens

* + 1. **CODE FOR MENU SCREEN**

package com.example.hush;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.databinding.DataBindingUtil;  
import androidx.recyclerview.widget.LinearLayoutManager;  
  
import android.content.Context;  
import android.content.Intent;  
import android.net.ConnectivityManager;  
import android.net.NetworkInfo;  
import android.net.Uri;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.View;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.example.hush.databinding.MenuActivityDataBinding;  
import com.google.android.material.bottomsheet.BottomSheetDialog;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.ncorti.slidetoact.SlideToActView;  
  
import org.jetbrains.annotations.NotNull;  
  
import java.text.DecimalFormat;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
  
public class MenuActivity extends AppCompatActivity {  
  
 private MenuActivityDataBinding menuActivityDataBinding;  
  
 final int UPI\_PAYMENT = 0;  
 private LinearLayoutManager linearLayoutManager;  
 private MenuAdapter menuAdapter;  
 private List<MenuItem> menuItemList;  
 private DatabaseReference databaseReference;  
 private static int total = 0;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 menuActivityDataBinding = DataBindingUtil.setContentView(MenuActivity.this, R.layout.activity\_menu);  
 menuActivityDataBinding.getRoot();  
  
 menuItemList = new ArrayList<>();  
  
 int[] ints = {R.drawable.chicken\_image,  
 R.drawable.chicken\_fried\_rice,  
 R.drawable.chicken\_rolls,  
 R.drawable.lemon\_chicken,  
 R.drawable.chicken\_image,  
 R.drawable.chicken\_fried\_rice,  
 R.drawable.chicken\_rolls,  
 R.drawable.lemon\_chicken,  
 R.drawable.chicken\_image,  
 R.drawable.chicken\_fried\_rice,  
 R.drawable.chicken\_rolls,  
 R.drawable.lemon\_chicken};  
  
 String[] strings = {"Chicken Fried Rice",  
 "Chicken Dimsums",  
 "Chicken Lollipop",  
 "Afghani Chicken",  
 "Chicken Korma",  
 "Mutton Kebabs",  
 "Chicken Tikka",  
 "Prawns Fried Rice",  
 "Grilled Fish",  
 "Steak",  
 "Mutton Korma",  
 "Radda Chicken"  
 };  
  
 String[] prices = {"299",  
 "400",  
 "388",  
 "722",  
 "299",  
 "122",  
 "344",  
 "800",  
 "299",  
 "220",  
 "300",  
 "599"  
 };  
  
 for (int i = 0; i < ints.length; i++) {  
 menuItemList.add(new MenuItem(ints[i], strings[i], prices[i]));  
 }  
  
 menuAdapter = new MenuAdapter(MenuActivity.this, menuItemList);  
 menuActivityDataBinding.restaurantMenuRecyclerView.setLayoutManager(new LinearLayoutManager(MenuActivity.this));  
 menuActivityDataBinding.restaurantMenuRecyclerView.setAdapter(menuAdapter);  
  
 menuActivityDataBinding.orderFoodFloatingActionButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 DecimalFormat df2 = new DecimalFormat("#.##");  
 double tax = (0.08 \* DatabaseOperations.totalAmount);  
 double grandTotal = tax + DatabaseOperations.totalAmount;  
 BottomSheetDialog bottomSheetDialog = new BottomSheetDialog(MenuActivity.this);  
 bottomSheetDialog.setContentView(R.layout.order\_payment\_bottom\_sheet\_dialog);  
 TextView itemAmountTextView = bottomSheetDialog.findViewById(R.id.foodOrderItemTextView);  
 TextView taxTextView = bottomSheetDialog.findViewById(R.id.taxTextView);  
 TextView grandTotalTextView = bottomSheetDialog.findViewById(R.id.grandTotalTextView);  
 TextView customerNameTextView = bottomSheetDialog.findViewById(R.id.customerNameTextView);  
 SlideToActView payForOrderSlideToActView = bottomSheetDialog.findViewById(R.id.payForOrderSlideToActView);  
 itemAmountTextView.setText(String.format("₹"+ DatabaseOperations.totalAmount));  
 taxTextView.setText(String.format("₹"+ df2.format(tax)));  
 grandTotalTextView.setText(String.format("₹%s", df2.format(grandTotal)));  
 customerNameTextView.setText(UserDetails.mobileNUmber);  
 bottomSheetDialog.show();  
  
 String currentDateTimeString = java.text.DateFormat.getDateTimeInstance().format(new Date());  
  
 databaseReference = FirebaseDatabase.getInstance().getReference(Configuration.ORDER\_RECEIPT\_DETAILS);  
 Order order = new Order(UserDetails.fullName, UserDetails.emailAddress, UserDetails.mobileNUmber, DatabaseOperations.orderReceiptDetailsList);  
 databaseReference.child(currentDateTimeString).setValue(order);  
 Toast.makeText(MenuActivity.this, "Order Successfully Submitted", Toast.LENGTH\_SHORT).show();  
 menuAdapter = new MenuAdapter(MenuActivity.this, menuItemList);  
 menuActivityDataBinding.restaurantMenuRecyclerView.setLayoutManager(new LinearLayoutManager(MenuActivity.this));  
 menuActivityDataBinding.restaurantMenuRecyclerView.setAdapter(menuAdapter);  
  
 payForOrderSlideToActView.setOnSlideCompleteListener(new SlideToActView.OnSlideCompleteListener() {  
 @Override  
 public void onSlideComplete(@NotNull SlideToActView slideToActView) {  
 makeGooglePayPayment(String.valueOf(grandTotal));  
 }  
 });  
 DatabaseOperations.totalAmount = 0;  
 }  
 });  
 }  
  
 private void makeGooglePayPayment(String total) {  
 Uri uri = Uri.parse("upi://pay").buildUpon()  
 .appendQueryParameter("pa", "parthdadwal92-2@okaxis")  
 .appendQueryParameter("pn", "Parth Dadwal")  
 .appendQueryParameter("tn", "Food Order Payment")  
 .appendQueryParameter("am", total)  
 .appendQueryParameter("cu", "INR")  
 .build();  
 Intent upiPayIntent = new Intent(Intent.ACTION\_VIEW);  
 upiPayIntent.setData(uri);  
 Intent chooser = Intent.createChooser(upiPayIntent, "Pay with");  
 if (null != chooser.resolveActivity(getPackageManager())) {  
 startActivityForResult(chooser, UPI\_PAYMENT);  
 } else {  
 Toast.makeText(MenuActivity.this, "No UPI app found, please install one to continue", Toast.LENGTH\_SHORT).show();  
 }  
 }  
  
 @Override  
 public void onActivityResult(int requestCode, int resultCode, Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);  
 Log.e("main ", "response " + resultCode);  
 switch (requestCode) {  
 case UPI\_PAYMENT:  
 if ((RESULT\_OK == resultCode) || (resultCode == 11)) {  
 if (data != null) {  
 String trxt = data.getStringExtra("response");  
 Log.e("UPI", "onActivityResult: " + trxt);  
 ArrayList<String> dataList = new ArrayList<>();  
 dataList.add(trxt);  
 upiPaymentDataOperation(dataList);  
 } else {  
 Log.e("UPI", "onActivityResult: " + "Return data is null");  
 ArrayList<String> dataList = new ArrayList<>();  
 dataList.add("nothing");  
 upiPaymentDataOperation(dataList);  
 }  
 } else {  
 Log.e("UPI", "onActivityResult: " + "Return data is null");  
 ArrayList<String> dataList = new ArrayList<>();  
 dataList.add("nothing");  
 upiPaymentDataOperation(dataList);  
 }  
 break;  
 }  
 }  
  
 private void upiPaymentDataOperation(ArrayList<String> data) {  
 if (isConnectionAvailable(MenuActivity.this)) {  
 String str = data.get(0);  
 Log.e("UPIPAY", "upiPaymentDataOperation: " + str);  
 String paymentCancel = "";  
 if (str == null) str = "discard";  
 String status = "";  
 String approvalRefNo = "";  
 String response[] = str.split("&");  
 for (int i = 0; i < response.length; i++) {  
 String equalStr[] = response[i].split("=");  
 if (equalStr.length >= 2) {  
 if (equalStr[0].toLowerCase().equals("Status".toLowerCase())) {  
 status = equalStr[1].toLowerCase();  
 } else if (equalStr[0].toLowerCase().equals("ApprovalRefNo".toLowerCase()) || equalStr[0].toLowerCase().equals("txnRef".toLowerCase())) {  
 approvalRefNo = equalStr[1];  
 }  
 } else {  
 paymentCancel = "Payment cancelled by user.";  
 }  
 }  
 if (status.equals("success")) {  
 Toast.makeText(MenuActivity.this, "Transaction successful.", Toast.LENGTH\_SHORT).show();  
 Log.e("UPI", "payment successfull: " + approvalRefNo);  
 } else if ("Payment cancelled by user.".equals(paymentCancel)) {  
 Toast.makeText(MenuActivity.this, "Payment cancelled by user.", Toast.LENGTH\_SHORT).show();  
 Log.e("UPI", "Cancelled by user: " + approvalRefNo);  
 } else {  
 Toast.makeText(MenuActivity.this, "Transaction failed.Please try again", Toast.LENGTH\_SHORT).show();  
 Log.e("UPI", "failed payment: " + approvalRefNo);  
 }  
 } else {  
 Log.e("UPI", "Internet issue: ");  
 Toast.makeText(MenuActivity.this, "Internet connection is not available. Please check and try again", Toast.LENGTH\_SHORT).show();  
 }  
 }  
  
 public static boolean isConnectionAvailable(Context context) {  
 ConnectivityManager connectivityManager = (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY\_SERVICE);  
 if (connectivityManager != null) {  
 NetworkInfo netInfo = connectivityManager.getActiveNetworkInfo();  
 if (netInfo != null && netInfo.isConnected()  
 && netInfo.isConnectedOrConnecting()  
 && netInfo.isAvailable()) {  
 return true;  
 }  
 }  
 return false;  
 }  
}

* + 1. **MENU SCREEN**

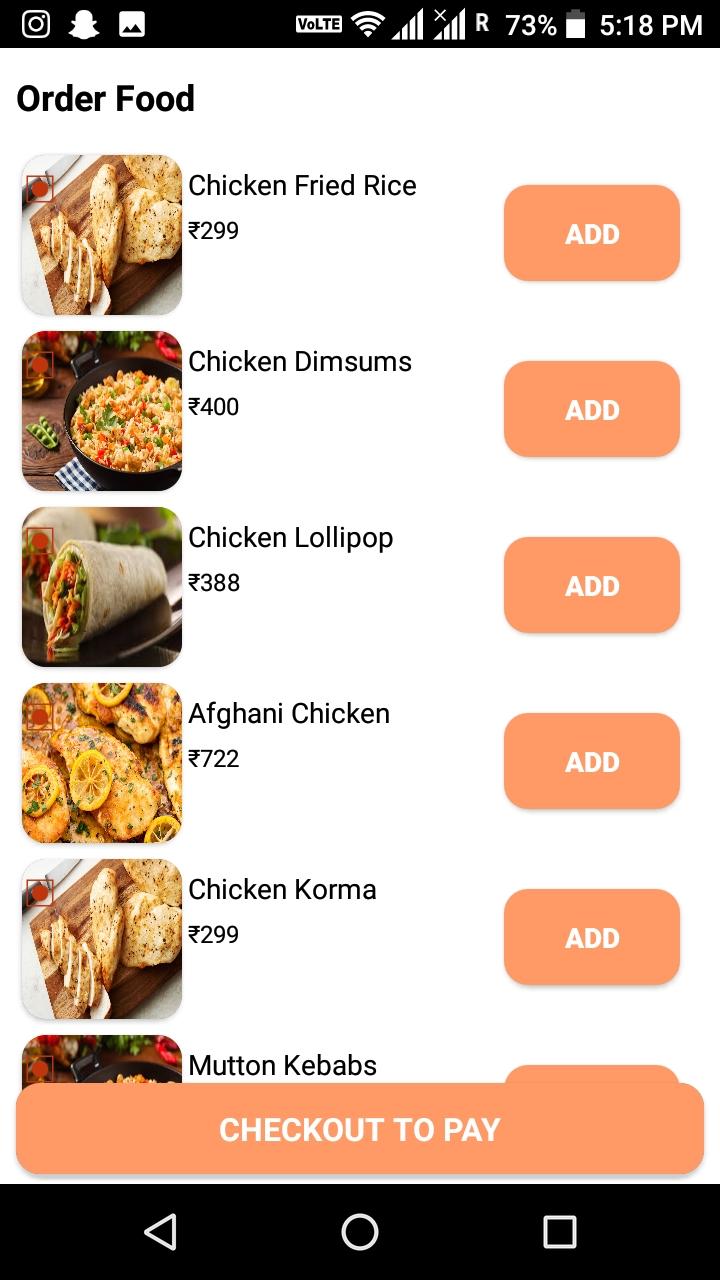
****

Fig. 4.8 Menu Screen

* + 1. **CODE FOR HOME SCREEN**

package com.example.hush;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.databinding.DataBindingUtil;  
  
import android.graphics.Color;  
import android.os.Bundle;  
import android.view.View;  
import android.view.ViewGroup;  
  
import com.example.hush.databinding.HomeActivityDataBinding;  
import com.google.android.material.tabs.TabLayout;  
  
public class HomeActivity extends AppCompatActivity implements TabLayout.OnTabSelectedListener {  
  
 private HomeActivityDataBinding homeActivityDataBinding;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
  
 super.onCreate(savedInstanceState);  
 homeActivityDataBinding = DataBindingUtil.setContentView(HomeActivity.this, R.layout.activity\_home);  
 homeActivityDataBinding.getRoot();  
  
 homeActivityDataBinding.tabLayout.addTab(homeActivityDataBinding.tabLayout.newTab().setText("Hush Menu"));  
 homeActivityDataBinding.tabLayout.addTab(homeActivityDataBinding.tabLayout.newTab().setText("User Profile"));  
  
 homeActivityDataBinding.tabLayout.setTabGravity(TabLayout.GRAVITY\_FILL);  
 homeActivityDataBinding.tabLayout.setTabMode(TabLayout.MODE\_SCROLLABLE);  
 homeActivityDataBinding.tabLayout.setSelectedTabIndicatorColor(Color.parseColor("#ff9966"));  
  
 UserViewPager userViewPager = new UserViewPager(getSupportFragmentManager(), homeActivityDataBinding.tabLayout.getTabCount(), HomeActivity.this);  
  
 for(int i=0; i < homeActivityDataBinding.tabLayout.getTabCount(); i++) {  
 View tab = ((ViewGroup) homeActivityDataBinding.tabLayout.getChildAt(0)).getChildAt(i);  
 ViewGroup.MarginLayoutParams p = (ViewGroup.MarginLayoutParams) tab.getLayoutParams();  
 p.setMargins(0, 0, 20, 0);  
 tab.requestLayout();  
 }  
  
 homeActivityDataBinding.userProfileViewPager.addOnPageChangeListener(new TabLayout.TabLayoutOnPageChangeListener(homeActivityDataBinding.tabLayout));  
  
 homeActivityDataBinding.userProfileViewPager.setAdapter(userViewPager);  
 homeActivityDataBinding.tabLayout.setOnTabSelectedListener(this);  
 }  
  
 @Override  
 public void onTabSelected(TabLayout.Tab tab) {  
 homeActivityDataBinding.userProfileViewPager.setCurrentItem(tab.getPosition());  
 }  
  
 @Override  
 public void onTabUnselected(TabLayout.Tab tab) {  
  
 }  
  
 @Override  
 public void onTabReselected(TabLayout.Tab tab) {  
  
 }  
}

* + 1. **HOME SCREEN**

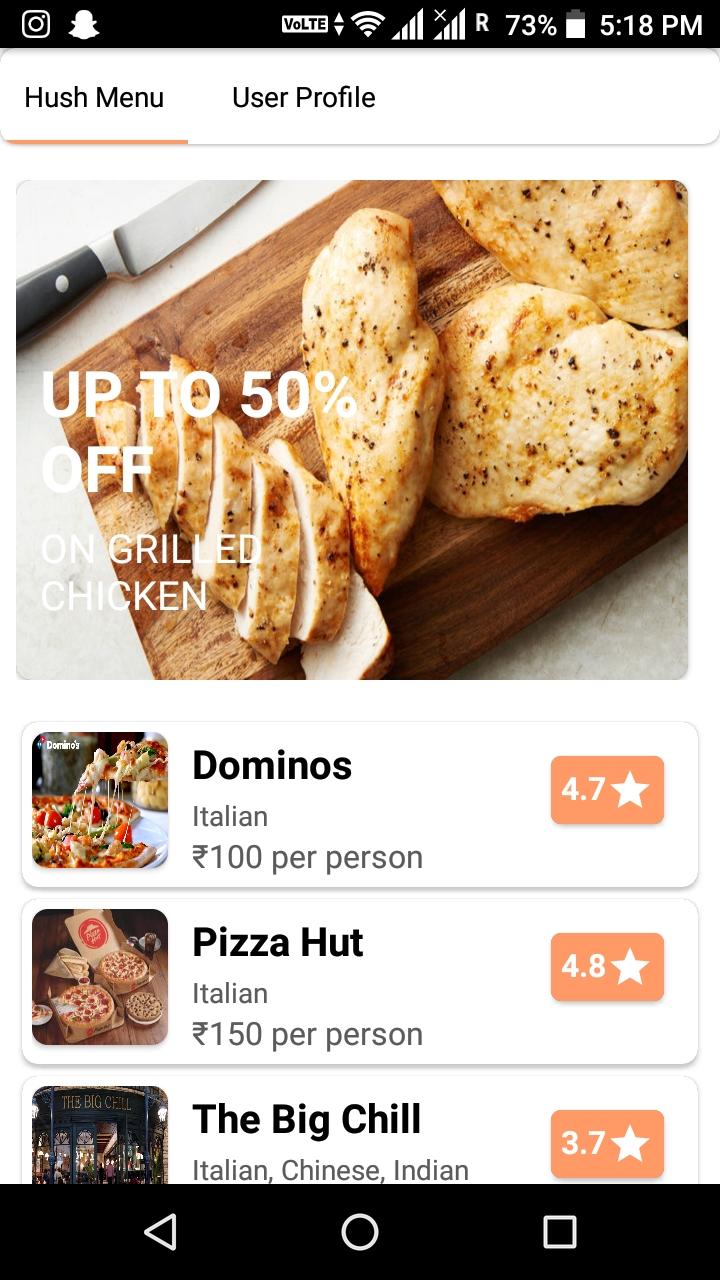
****

Fig. 4.9 Home Screen

* 1. **ORDER UPDATE AND PAYMENT**

For ordering and updating following is the designing.

* + 1. **USE CASE DIAGRAM FOR ORDER UPDATION AND PAYMENT**

uc Use Case Model

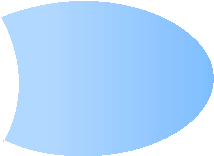
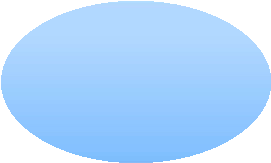
Order & Payment



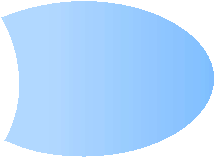
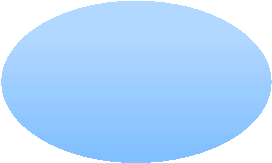
Canteen Owner



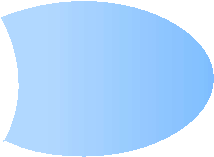
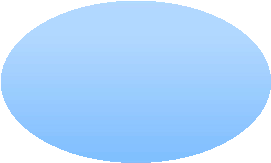
Customer



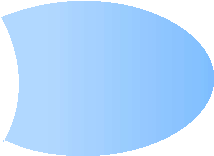
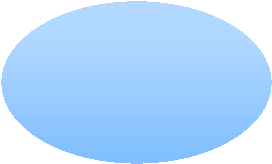
Update Menu



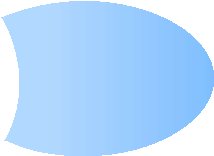
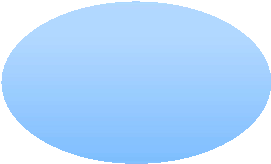
Monitor Canteen



Place Order



Read Order



Make Payment

«include»

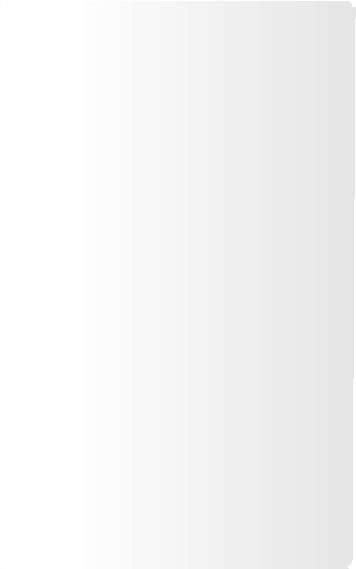
«extend»

«extend»

Fig. 4.10 Use case for Order, Payment & Update

* + 1. **CLASS DIAGRAM FOR ORDER UPDATION AND PAYMENT**

class Class Model



Order UI

Food items: char

-

cold drinks: char

-

order\_id: int

-

Tea and Coffee: char

-

Cust\_name: char

-

choose\_order(): char

+

make\_order(): char

+

order\_again(): char

+

confirm\_order(): char

+

bill(): char

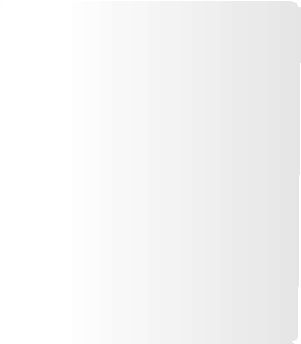
+

+

update\_menu(): char

+

update to\_do\_list(): char



Customer

-

Food items: char

-

cold drinks: char

Tea and Coffee: char

-

choose\_order(): char

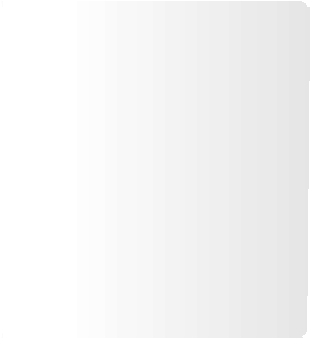
+

+

make\_order(): char

+

order\_again(): char



Canteen manager

-

Cust\_name: char

-

order\_id: int

confirm\_order(): char

+

+

bill(): char

+

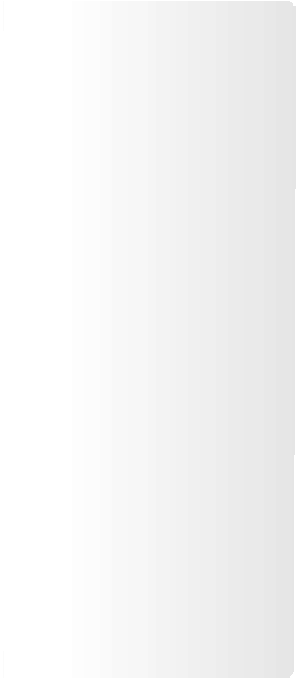
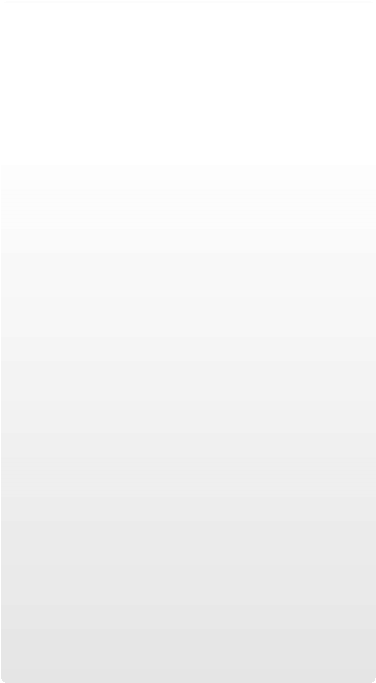
update menu(): char

+

update to\_do\_list(): char

Fig. 4.11 Class diagram of Order and Update

class Class Model



Payment UI

-

cash: int

card name: char

-

card type: char

-

-

payment id: char

-

Payment\_id: char

-

cust\_name: char

-

order\_id: int

select\_mode(): char

+

cash\_mode(): char

+

pay\_cash(): float

+

card\_mode(): char

+

+

card\_payment(): int

+

fee\_mode(): char

+

card\_error(): boolean

print\_receipt(): char

+

+

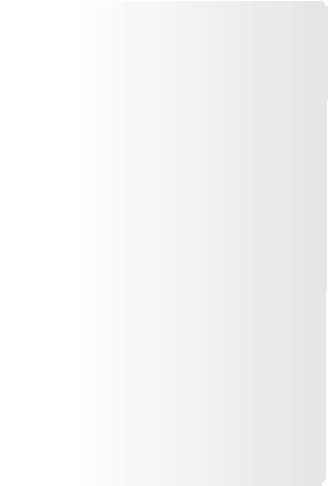
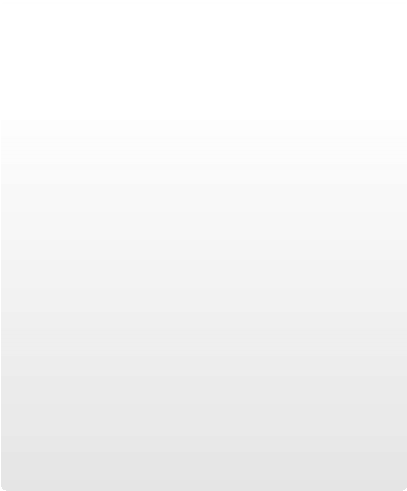
changes(): int

select\_fee\_mode(): char

+

+

proceed\_payment(): float



Customer

-

cash: int

-

card name: char

-

card type: char

-

payment id: char

select\_mode(): char

+

+

cash\_mode(): char

+

pay\_cash(): float

+

card\_mode(): char

+

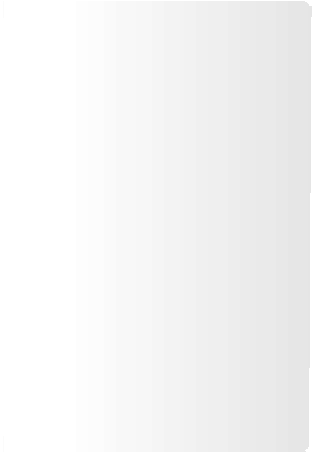
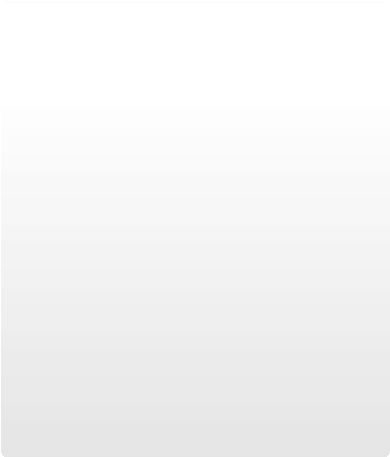
card\_payment(): int

+

fee\_mode(): char

+

card\_error(): boolean



Canteen manager

-

Payment\_id: char

-

cust\_name: char

order\_id: int

-

+

print\_receipt(): char

+

changes(): int

+

select\_fee\_mode(): char

+

proceed\_payment(): float

Fig. 4.12 Class diagram for Payment

* + 1. **SEQUENCE DIAGRAM FOR PAYMENT**

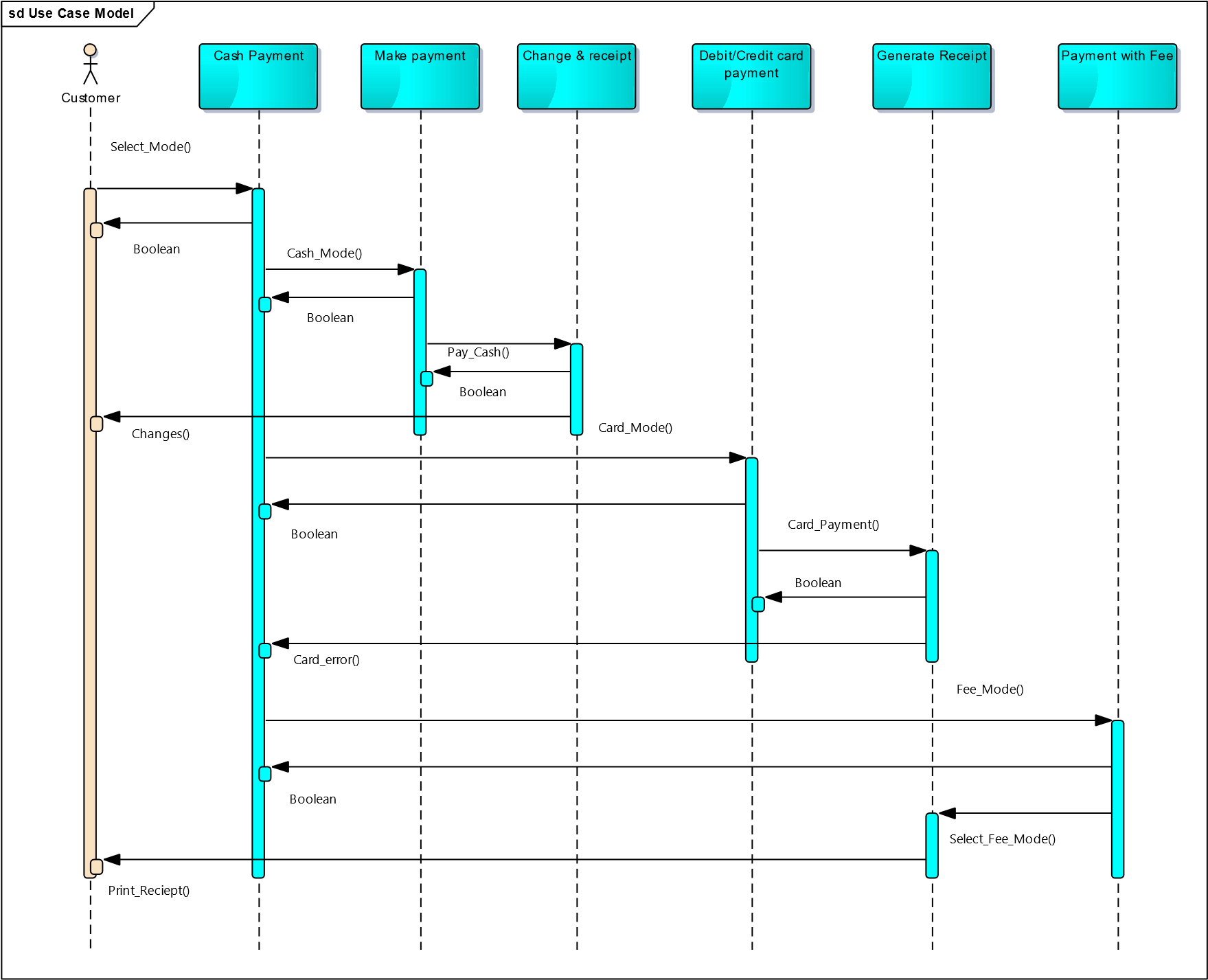


Fig. 4.13 Sequence diagram for payment

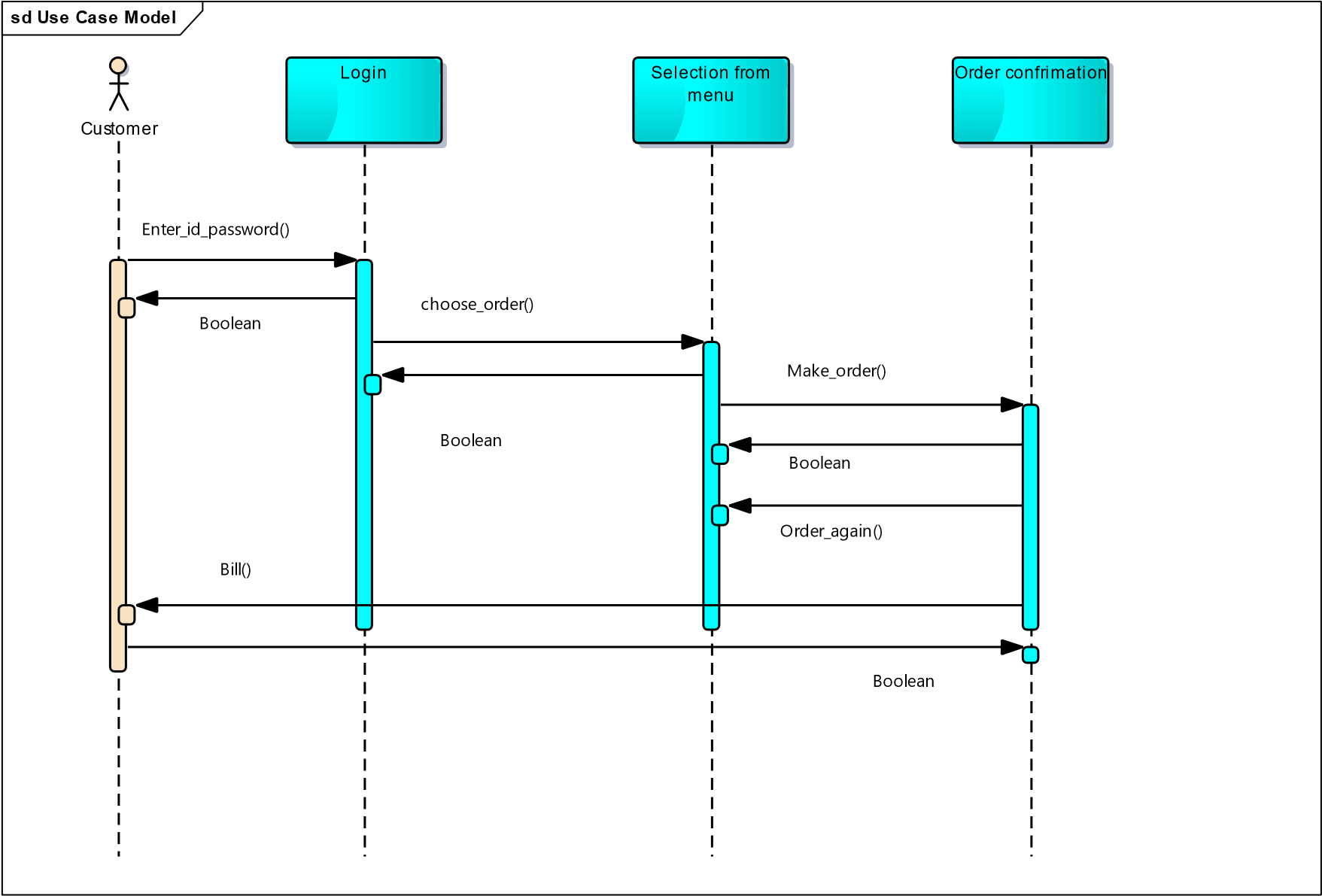
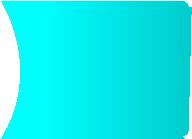


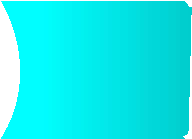
Fig. 4.14 Sequence diagram of placing order

sd Use Case Model

Canteen Owner



Menu



To Do List

Update\_Menu()

Boolean

Update\_todolist()

Boolean

Fig. 4.15 Sequence Diagram for Updating Menu

* + 1. **CODE FOR PAYMENT & ORDER RECIEPT**

private void makeGooglePayPayment(String total) {  
 Uri uri = Uri.parse("upi://pay").buildUpon()  
 .appendQueryParameter("pa", "parthdadwal92-2@okaxis")  
 .appendQueryParameter("pn", "Parth Dadwal")  
 .appendQueryParameter("tn", "Food Order Payment")  
 .appendQueryParameter("am", total)  
 .appendQueryParameter("cu", "INR")  
 .build();  
 Intent upiPayIntent = new Intent(Intent.ACTION\_VIEW);  
 upiPayIntent.setData(uri);  
 Intent chooser = Intent.createChooser(upiPayIntent, "Pay with");  
 if (null != chooser.resolveActivity(getPackageManager())) {  
 startActivityForResult(chooser, UPI\_PAYMENT);  
 } else {  
 Toast.makeText(MenuActivity.this, "No UPI app found, please install one to continue", Toast.LENGTH\_SHORT).show();  
 }  
}  
  
@Override  
public void onActivityResult(int requestCode, int resultCode, Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);  
 Log.e("main ", "response " + resultCode);  
 switch (requestCode) {  
 case UPI\_PAYMENT:  
 if ((RESULT\_OK == resultCode) || (resultCode == 11)) {  
 if (data != null) {  
 String trxt = data.getStringExtra("response");  
 Log.e("UPI", "onActivityResult: " + trxt);  
 ArrayList<String> dataList = new ArrayList<>();  
 dataList.add(trxt);  
 upiPaymentDataOperation(dataList);  
 } else {  
 Log.e("UPI", "onActivityResult: " + "Return data is null");  
 ArrayList<String> dataList = new ArrayList<>();  
 dataList.add("nothing");  
 upiPaymentDataOperation(dataList);  
 }  
 } else {  
 Log.e("UPI", "onActivityResult: " + "Return data is null");  
 ArrayList<String> dataList = new ArrayList<>();  
 dataList.add("nothing");  
 upiPaymentDataOperation(dataList);  
 }  
 break;  
 }  
}  
  
private void upiPaymentDataOperation(ArrayList<String> data) {  
 if (isConnectionAvailable(MenuActivity.this)) {  
 String str = data.get(0);  
 Log.e("UPIPAY", "upiPaymentDataOperation: " + str);  
 String paymentCancel = "";  
 if (str == null) str = "discard";  
 String status = "";  
 String approvalRefNo = "";  
 String response[] = str.split("&");  
 for (int i = 0; i < response.length; i++) {  
 String equalStr[] = response[i].split("=");  
 if (equalStr.length >= 2) {  
 if (equalStr[0].toLowerCase().equals("Status".toLowerCase())) {  
 status = equalStr[1].toLowerCase();  
 } else if (equalStr[0].toLowerCase().equals("ApprovalRefNo".toLowerCase()) || equalStr[0].toLowerCase().equals("txnRef".toLowerCase())) {  
 approvalRefNo = equalStr[1];  
 }  
 } else {  
 paymentCancel = "Payment cancelled by user.";  
 }  
 }  
 if (status.equals("success")) {  
 Toast.makeText(MenuActivity.this, "Transaction successful.", Toast.LENGTH\_SHORT).show();  
 Log.e("UPI", "payment successfull: " + approvalRefNo);  
 } else if ("Payment cancelled by user.".equals(paymentCancel)) {  
 Toast.makeText(MenuActivity.this, "Payment cancelled by user.", Toast.LENGTH\_SHORT).show();  
 Log.e("UPI", "Cancelled by user: " + approvalRefNo);  
 } else {  
 Toast.makeText(MenuActivity.this, "Transaction failed.Please try again", Toast.LENGTH\_SHORT).show();  
 Log.e("UPI", "failed payment: " + approvalRefNo);  
 }  
 } else {  
 Log.e("UPI", "Internet issue: ");  
 Toast.makeText(MenuActivity.this, "Internet connection is not available. Please check and try again", Toast.LENGTH\_SHORT).show();  
 }  
}

public Order(String name, String emailAddress, String mobileNUmber, List<OrderReceiptDetails> orderReceiptDetailsList) {  
 this.name = name;  
 this.emailAddress = emailAddress;  
 this.mobileNUmber = mobileNUmber;  
 this.orderReceiptDetailsList = orderReceiptDetailsList;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getEmailAddress() {  
 return emailAddress;  
 }  
  
 public void setEmailAddress(String emailAddress) {  
 this.emailAddress = emailAddress;  
 }  
  
 public String getMobileNUmber() {  
 return mobileNUmber;  
 }  
  
 public void setMobileNUmber(String mobileNUmber) {  
 this.mobileNUmber = mobileNUmber;  
 }  
  
 public List<OrderReceiptDetails> getOrderReceiptDetailsList() {  
 return orderReceiptDetailsList;  
 }  
  
 public void setOrderReceiptDetailsList(List<OrderReceiptDetails> orderReceiptDetailsList) {  
 this.orderReceiptDetailsList = orderReceiptDetailsList;  
 }  
}

package com.example.hush;  
  
public class OrderReceiptDetails {  
 String id;  
 private String foodItemName;  
 private String foodItemPrice;  
  
 public OrderReceiptDetails() {  
  
 }  
  
 public OrderReceiptDetails(String id, String foodItemName, String foodItemPrice) {  
 this.id = id;  
 this.foodItemName = foodItemName;  
 this.foodItemPrice = foodItemPrice;  
 }  
  
 public String getId() {  
 return id;  
 }  
  
 public void setId(String id) {  
 this.id = id;  
 }  
  
 public String getFoodItemName() {  
 return foodItemName;  
 }  
  
 public void setFoodItemName(String foodItemName) {  
 this.foodItemName = foodItemName;  
 }  
  
 public String getFoodItemPrice() {  
 return foodItemPrice;  
 }  
  
 public void setFoodItemPrice(String foodItemPrice) {  
 this.foodItemPrice = foodItemPrice;  
 }  
}

* + 1. **ORDER UPDATION AND PAYMENT SCREEN**

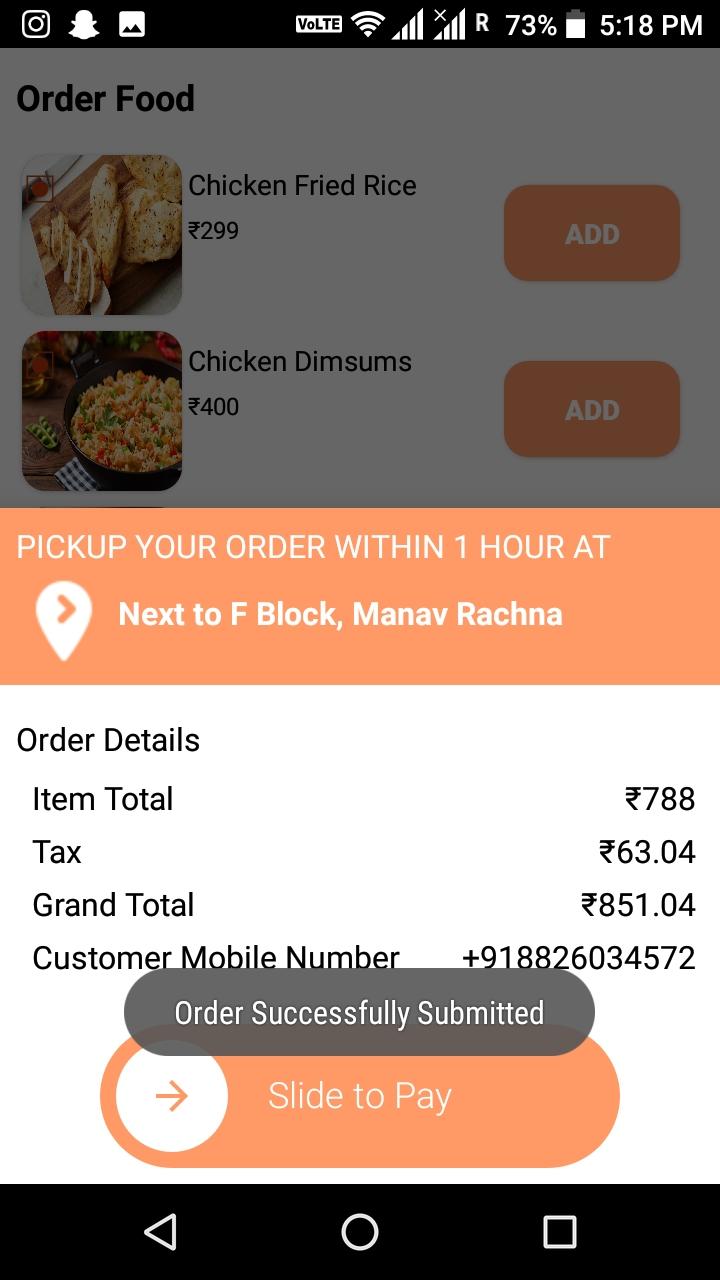
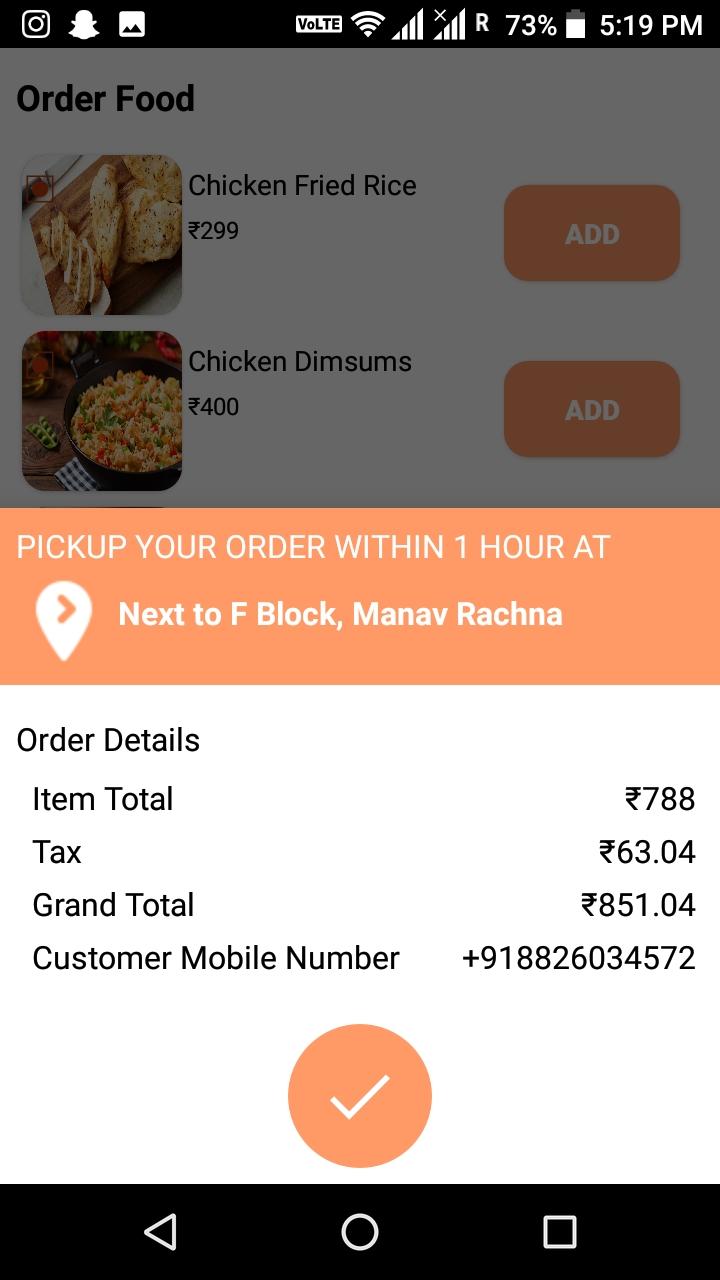
** **

Fig. 4.16 Order placement Fig. 4.17 Payment Screen

* 1. **USER PROFILE**

package com.example.hush;  
  
import android.os.Bundle;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.databinding.DataBindingUtil;  
import androidx.fragment.app.Fragment;  
  
import com.example.hush.apiservices.RetrofitServicesHandler;  
import com.mikhaellopez.circularimageview.CircularImageView;  
import com.squareup.picasso.Picasso;  
  
import retrofit2.Call;  
import retrofit2.Callback;  
import retrofit2.Response;  
  
  
public class UserProfile extends Fragment {  
  
 private CircularImageView circularImageView;  
 private TextView nameTextView;  
 private TextView userNameTextView;  
 private TextView emailAddressTextView;  
 private TextView mobileNumberTextView;  
  
 @Nullable  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  
  
 View view = inflater.inflate(R.layout.user\_profile\_fragment, container, false);  
  
 circularImageView = view.findViewById(R.id.userProfilePhotoCircularImageView);  
 nameTextView = view.findViewById(R.id.nameTextView);  
 userNameTextView = view.findViewById(R.id.userNameTextView);  
 emailAddressTextView = view.findViewById(R.id.emailAddressTextView);  
 mobileNumberTextView = view.findViewById(R.id.mobileNumberTextView);  
  
 Call<GetUserProfileDetails> getUserProfileDetailsCall = RetrofitServicesHandler  
 .getInstance()  
 .getApi()  
 .getUserProfileDetails(UserDetails.id);  
  
 getUserProfileDetailsCall.enqueue(new Callback<GetUserProfileDetails>() {  
 @Override  
 public void onResponse(Call<GetUserProfileDetails> call, Response<GetUserProfileDetails> response) {  
 GetUserProfileDetails getUserProfileDetails = response.body();  
 if (response.body() != null) {  
 if (!getUserProfileDetails.getImage().equals("")) {  
 Picasso.get().load("https://acubeapps.com/sanskariApi/post/Api/uploads/" + getUserProfileDetails.getImage()).into(circularImageView);  
 } else {  
 Picasso.get().load("https://encrypted-tbn0.gstatic.com/images?q=tbn%3AANd9GcS3GOTKsjMYfeu4L5B-A9LQHPU\_W4KnM2HmLH8nbhYMzNKJiA66&usqp=CAU").into(circularImageView);  
 }  
 nameTextView.setText(getUserProfileDetails.getFullname());  
 userNameTextView.setText(String.format("@%s", getUserProfileDetails.getFullname()));  
 emailAddressTextView.setText(getUserProfileDetails.getEmail());  
 mobileNumberTextView.setText(getUserProfileDetails.getMobile());  
 } else {  
  
 }  
 }  
  
 @Override  
 public void onFailure(Call<GetUserProfileDetails> call, Throwable t) {  
  
 }  
 });  
  
 return view;  
 }  
}

package com.example.hush;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.databinding.DataBindingUtil;  
  
import android.os.Bundle;  
  
public class UserProfileCoverPhotosAcitivity extends AppCompatActivity {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_user\_profile\_cover\_photos\_acitivity);  
 }  
}

package com.example.hush;  
  
public class UserProfileDetails {  
 private String name;  
 private String emailID;  
 private String imageURL;  
 private String userID;  
 private String phoneNumber;  
 private int gender;  
 private String password;  
  
 public UserProfileDetails() {  
  
 }  
  
 public UserProfileDetails(String name, String emailID, String imageURL, String userID, String phoneNumber, int gender, String password) {  
 this.name = name;  
 this.emailID = emailID;  
 this.imageURL = imageURL;  
 this.userID = userID;  
 this.phoneNumber = phoneNumber;  
 this.gender = gender;  
 this.password = password;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getEmailID() {  
 return emailID;  
 }  
  
 public void setEmailID(String emailID) {  
 this.emailID = emailID;  
 }  
  
 public String getImageURL() {  
 return imageURL;  
 }  
  
 public void setImageURL(String imageURL) {  
 this.imageURL = imageURL;  
 }  
  
 public String getUserID() {  
 return userID;  
 }  
  
 public void setUserID(String userID) {  
 this.userID = userID;  
 }  
  
 public String getPhoneNumber() {  
 return phoneNumber;  
 }  
  
 public void setPhoneNumber(String phoneNumber) {  
 this.phoneNumber = phoneNumber;  
 }  
  
 public int getGender() {  
 return gender;  
 }  
  
 public void setGender(int gender) {  
 this.gender = gender;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
}

* + 1. **USER PROFILE SCREEN**

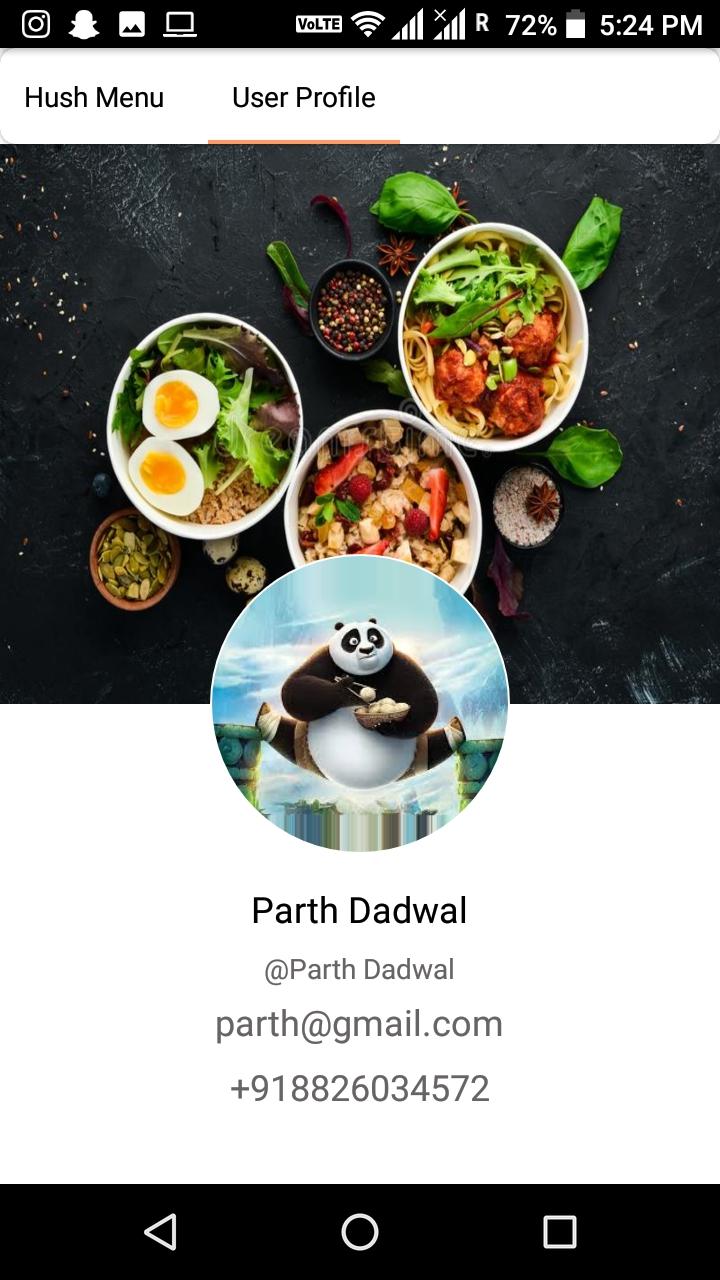
****

Fig. 4.18 User profile screen

* 1. **CODE TOKEN GENERATION**

package com.example.hush;  
  
import com.google.gson.annotations.SerializedName;  
  
public class UserRegistrationResponse {  
 @SerializedName("token")  
 private String token;  
  
 public UserRegistrationResponse(String token) {  
 this.token = token;  
 }  
  
 public String getToken() {  
 return token;  
 }  
  
 public void setToken(String token) {  
 this.token = token;  
 }  
}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_X\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CHAPTER 5: ER DIAGRAMS**

1. **ER DIAGRAM**

An entity-relationship (ER) diagram is a specialized graphic that illustrates the interrelationships between entities in a database. ER diagrams often use symbols to represent three different types of information. Boxes are commonly used to represent entities. Diamonds are normally used to represent relationships and ovals are used to represent attributes.

Symbols used in Entity-Relationship Diagram are as follows:

• Represent Data Entity.

* Represent connection Administrator
* Connect two Entities or One to One relation.

• Represent the relationship.

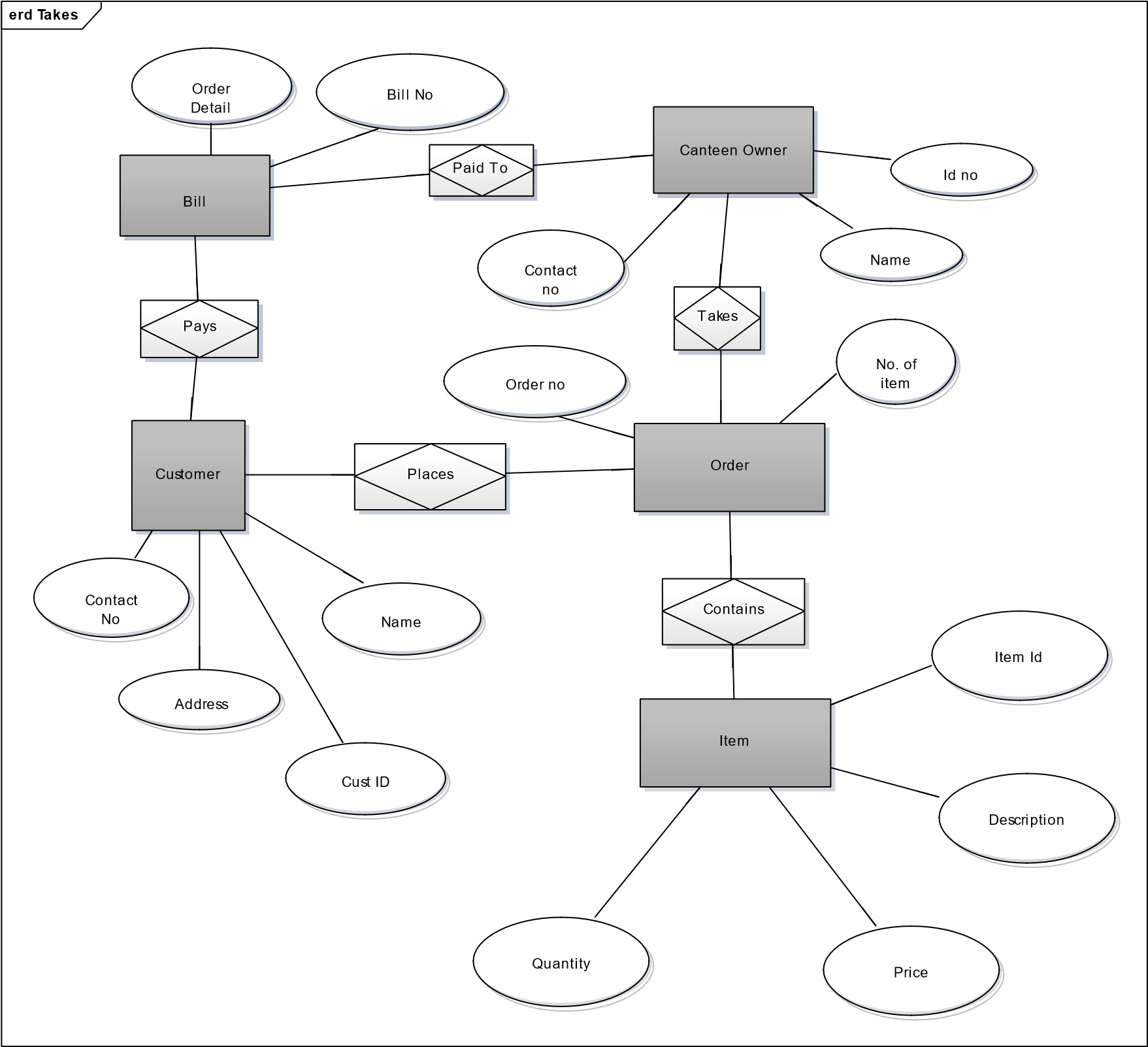


Fig. 5.1 ER Diagram for Canteen Automation Android application

* 1. **DATA FLOW DIAGRAM**

Admin

Verification of College

Canteen

Automation

I/P

O/P

|  |
| --- |
| Canteen Owner |

Canteen Maintaining, I/P Automation

|  |
| --- |
| Customer |

Canteen Placing Order

Fig. 5.2 Level-0 DFD

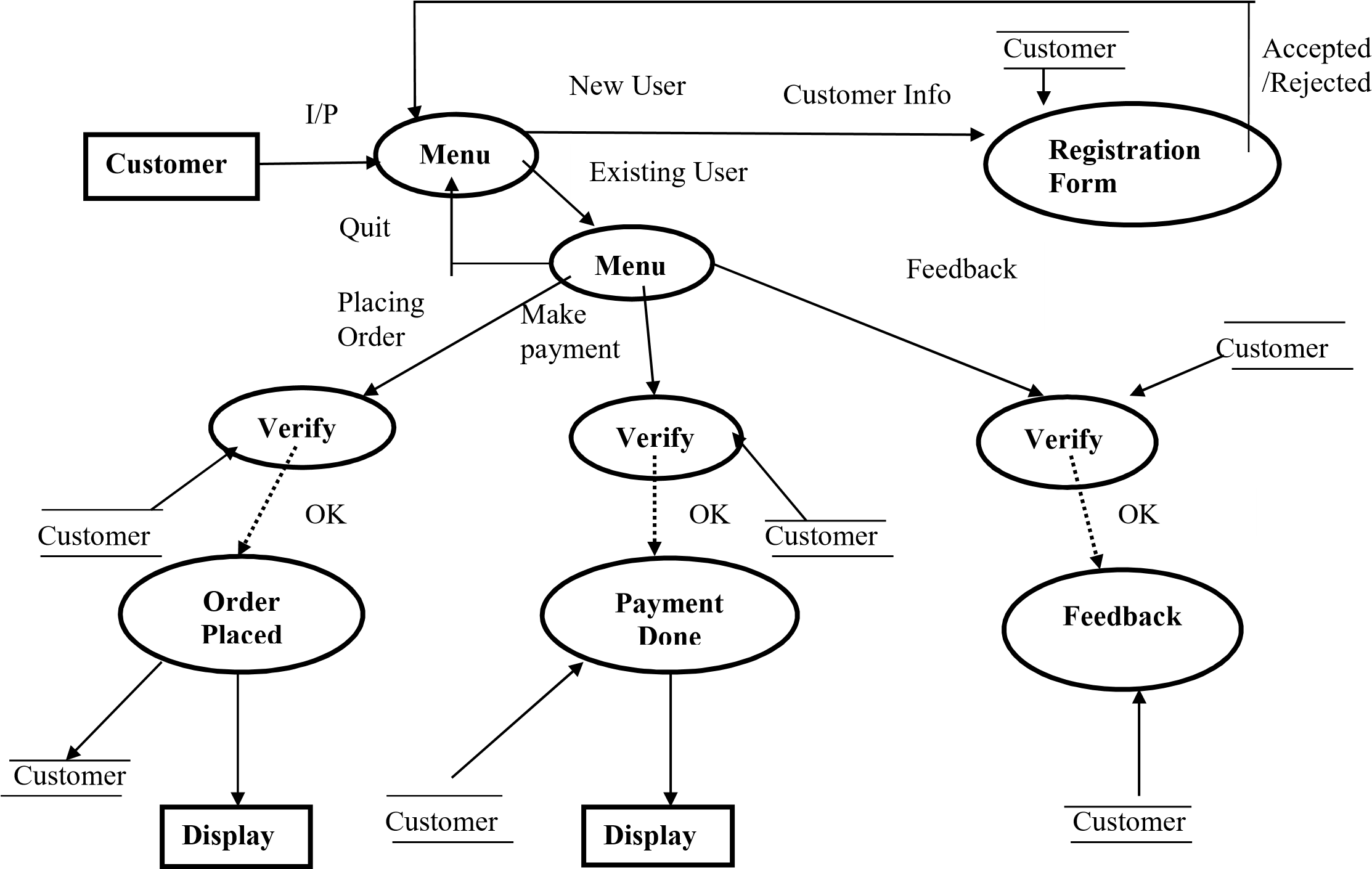
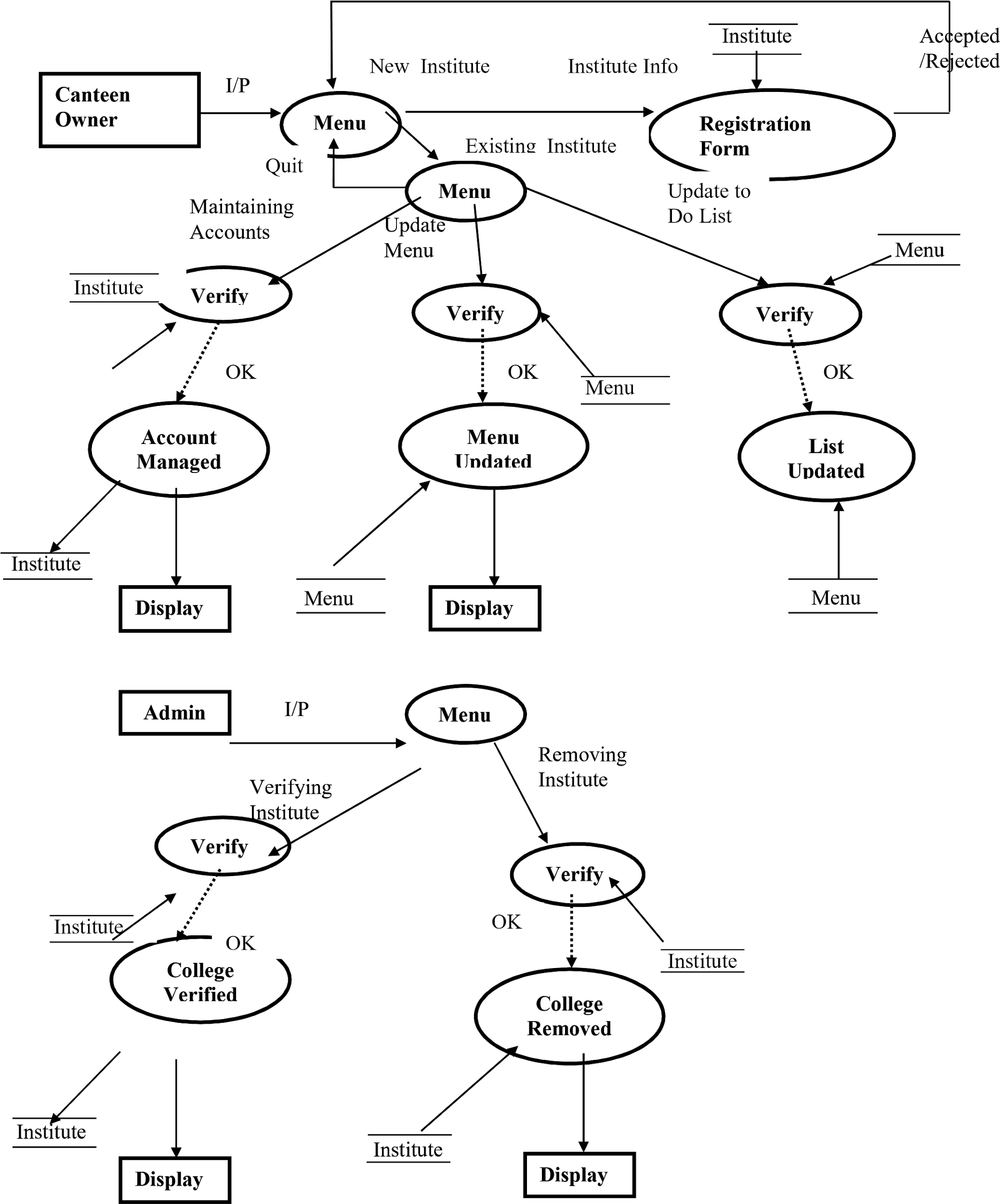


Fig. 5.3 LEVEL 1 DFD



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_X\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CHAPTER 6: DATABASE TABLES**

1. DATABASE TABLES

A few truncations recorded beneath -

PK – Primary Key

FK – Foreign Key

NN – Not Null (Required) UC – Unique Constraint

SLT – Single Line of Text

MLT – Multiple Lines of Text

PG – Person or Group

Indeed/No (check box) – Yes/No

* 1. **CUSTOMER**

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Customer | | |
| Description | This table will contain the personal information of the customer who places an order in the canteen. | | |
| Primary Keys | Cust\_Id | | |
|  |  | | |
| Field Name | Data type | Constraints | Comments |
| Cust\_Id | Int | PK |  |
| FirstName | varchar(50) | NN |  |
| LastName | varchar(50) | NN |  |
| EmailId | varchar(50) | NN |  |
| UserName | varchar(50) | NN |  |
| Password | varchar(25) |  | Must contain 8 character |
| Mobile No. | varchar(10) |  | It could be home or cell, should take more than one and allow numbers + text like Home-408888-3333, cell-408-888-9999 |
| Address | varchar(100) |  |  |
| City | varchar(20) |  |  |
| State | varchar(20) |  |  |
| Country | varchar(50) |  |  |
| Zipcode | varchar(8) |  |  |

Table 6.1

* 1. **ADMIN**

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Admin | | |
| Description | This table will contain the personal information of the admin that verify the college registration. | | |
| Primary Keys | Admin\_Id | | |
| Foreign Keys | College\_Id | | |
|  |  | | |
| Field Name | Data type | Constraints | Comments |
| Admin\_Id | Int | PK |  |
| College\_Id | Int | FK |  |
| FirstName | varchar(50) | NN |  |
| LastName | varchar(50) | NN |  |
| EmailId | varchar(50) | NN |  |
| UserName | varchar(50) | NN |  |
| Password | varchar(25) |  | Must contain 8 character |
| Mobile No. | varchar(10) |  | It could be home or cell |
| Address | varchar(100) |  |  |
| City | varchar(20) |  |  |
| State | varchar(20) |  |  |
| Country | varchar(50) |  |  |
| Zipcode | varchar(8) |  |  |

Table 6.2

* 1. **CANTEEN OWNER**

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Canteen Owner | | |
| Description | This table will contain the personal information of the canteen owner. | | |
| Primary Keys | College\_Id | | |
| Foreign Keys | Cust\_Id,Order\_Id,Item\_No | | |
|  |  | | |
| Field Name | Data type | Constraints | Comments |
| College\_Id | Int | PK |  |
| Cust\_Id | Int | FK |  |
| Order\_Id | Int | FK |  |
| Item\_Id | Int | FK |  |
| FirstName | varchar(50) | NN |  |
| LastName | varchar(50) | NN |  |
| EmailId | varchar(50) | NN |  |
| UserName | varchar(50) | NN |  |
| Password | varchar(25) |  | Must contain 8 character |
| Mobile No. | varchar(10) |  | It could be home or cell |
| Address | varchar(100) |  |  |
| City | varchar(20) |  |  |
| State | varchar(20) |  |  |
| Country | varchar(50) |  |  |
| Zipcode | varchar(8) |  |  |

Table 6.3

* 1. **MENU**

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Menu | | |
| Description | This table will contain all the details of the menu card. | | |
| Primary Keys | Item\_Id | | |
| Foreign Keys | College\_Id,Cust\_Id,Order\_Id | | |
|  |  | | |
| Field Name | Data type | Constraints | Comments |
| Item\_Id | Int | PK |  |
| College\_Id | Int | FK |  |
| Order\_Id | Int | FK |  |
| Item\_Id | Int | FK |  |
| Cust\_Id | Int | FK |  |
| ItemName | varchar(50) | NN |  |
| Quantity | Int | NN |  |

Table 6.4

* 1. **ORDER**

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Order | | |
| Description | This table will contain the details of Order that customer does. | | |
| Primary Keys | Order\_Id | | |
| Foreign Keys | College\_Id,Cust\_Id,Bill\_Id,Item\_Id | | |
|  |  | | |
| Field Name | Data type | Constraints | Comments |
| Order\_Id | Int | PK |  |
| Item\_Id | Int | FK |  |
| College\_Id | Int | FK |  |
| Bill\_Id | Int | FK |  |
| Item\_Id | Int | FK |  |
| Cust\_Id | Int | FK |  |

Table 6.5

* 1. **PAYMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Payment | | |
| Description | This table will contain the details of payment done by the customer. | | |
| Primary Keys | Bill\_Id | | |
| Foreign Keys | College\_Id,Cust\_Id,Order\_Id,Item\_Id | | |
| Field Name | Data type | Constraints | Comments |
| Bill\_Id | Int | PK |  |
| College\_Id | Int | FK |  |
| Order\_Id | Int | FK |  |
| Item\_Id | Int | FK |  |
| Cust\_Id | Int | FK |  |
| Payment\_mode | varchar(50) | NN |  |
| Amount | Int | NN |  |
| Credit/Debit\_card\_no | Int | NN |  |

Table 6.6

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_X\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**SUMMARY & CONCLUSION**

Summarizing the project, this android application lets the student place orders of their favorite and desired dishes from any canteen present in the college campus and premises.

Our team came up with this idea because it saves a lot of time of the students limited short lunch break among the whole day’s lectures

It saves their time to go at the canteen, decide amongst each other if there’s a group, may be fight for who’ll pay, place order and wait for the order to come up and when it arrives, the break is over by the time

Simply open the app, select the specific canteen, select from menu, place order, pay, receive token,

When the break starts get to campus canteen and order is already waiting to be eaten.

Well this app surely doesn’t help the fight for who’s going to pay, may be in future we will get to that too.

**REFERENCES**

WEBSITE REFERENCE

www.google.com www.tutorialpoint.com

www.w3schools.com

www. Scribd.com