



# Canteen System Automation App

---

Done by:

Arshit Babariya (18bce018)

Aditi Bhatt (18bce004)

## Introduction

Our product is a food ordering app that helps college students and faculties order food from the canteen effortlessly. This online app brings together the students and faculties across all institutes and departments of the college and connects them directly to the canteen staff. The user can efficiently place an order from the menu, directly to the canteen staff from any location within college premises, and upon receiving this order, the staff can prepare the meal and inform the user that it has been prepared, and is ready to be taken from the canteen.

## Problems with the existing system

The existing system consists of an order placement based on the coupon system, which involves waiting in line to buy the coupon, standing in line to place an order with the purchased coupon and waiting yet again for the order to be processed.

This is very time consuming and unpleasant especially during peak hours due to congestion. The overcrowdedness and lack of organisation reduces the efficiency of the canteen staff, and hinders it from generating the maximum possible revenue.

Our app helps in distributing the load and structurizing the food ordering and purchasing process.

## Purpose

The purpose of this product is to ease the food buying and selling process especially during peak hours. This proves to be remunerative for the canteen, as it prompts the users to purchase eatables without having to wait in line, hand in hand, it also ameliorates the purchasing process and saves time for the user. Hence it benefits both, the buyers and the sellers.

## Product scope

This product is an online app that can be downloaded from Playstore, it can be used by all the members of a college to place orders to the canteen staff. All placed orders can be monitored by the assigned Admin.

- Technologies Used:
  1. User Interface : XML in Android Studio
  2. Client-Side scripting : Java in Android Studio
  3. Operating System : Any OS that supports Google Chrome
  4. Database : Firebase
  
- Assumptions
  1. The user will have an active internet connection while using this application.

## 2 General Description

### 2.1 Stakeholders and Users of system

Agile Modeling (AM)'s definition of a project stakeholder is anyone who is materially impacted by the outcome of the solution. A stakeholder could be a direct user, indirect user, manager of users, senior manager, operations staff member, the "gold owner" who funds the project, support (help desk) staff member, auditors, your program/portfolio manager, developers working on other systems that integrate or interact with the one under development, or maintenance professionals potentially affected by the development and/or deployment of a software project.

By looking into this definition, we can divide them into two major categories:

#### 1. End users (External Users):

These are the people who will use the system, often to fulfill their requirements.

For example,

- Buyers (Students/Faculties/College staff members)
- Sellers(Canteen staff members)
- Assigned Admin

#### 2. Insiders (InternalUsers):

1. Database Administrator

2. Verification and technical team
3. Developer

## 2.2 Features

We have build this application with the following features:

1. View History : The users can view previously ordered items.
2. Placing Orders: The users can place an order of multiple food items.
3. View Orders: The canteen's staff members can view the current pending orders of all customers
4. Deleting Orders: The canteen's staff members can delete the received orders on their completion.
5. View All Orders: The admin can view all the orders that have been placed.
6. View Users: The admin can view all the users profiles.

## 2.3 Requirements

### 2.3.1 Functional Requirements:

1. Register :  
The user can register himself/herself into the system by giving the required information.  
Only the admin can register a staff member.
2. Management:  
The canteen staff can add/edit/delete the menu.  
The admin can view the activity of all the users.
3. Login:  
The admin/user/staff member will be able to login using their email id and password.
4. Logout:  
The admin/staff/user will be able to logout of the system.
5. Add order:  
The buyer can add orders to their cart.
6. Menu database:  
The canteen staff will be able to edit, add and delete items from the existing menu.
7. Representation of the food items:

The food items will be viewed in a card format, with their name, price and image.

### 3. Classification of Sprints

#### 3.1 Sprint 1

- Sign Up
- Login
- Select preferred eatable (can view previous history)
- Place the order

##### 3.1.1 User Stories

###### 3.1.1.1 Must have

- As a user I should be able to login in to the app:
  - User selects the login option
  - User enters the account email and password.
  - Email id and password gets verified
  - If everything is fine, the user is taken to his or her profile.
- User should be able to view my previous orders and select food from the menu :
  - A menu can be viewed from where orders and their quantities can be selected and added to cart.
  - On viewing the cart, changes can be made and the order can be placed.


###### 3.1.1.2 Could have

- The user can view his previous orders:
  - A page with previous orders can be viewed.
  - All the previous orders of that particular individual user are stored and maintained in a database.

Implementation of the major features:

User's side:

1. Sign in /Login



Users can register a new account by providing their details. These details are added in the authentication list of the app, in the firebase. An existing user can login using their registered email and password.

2. Selection of items and adding them to the user's cart:

This is done using the concept of activities and classes in Java. The user interface is maintained using a recyclerview which is implemented using XML.

3. Confirming an order:

The order in the cart is stored in the user's temporary database which is maintained via a firebase database. On confirming the order, the data is deleted from the user's temporary database and then added to the database that is accessible by the canteen's staff members.

4. Placing an order:

The placed order is stored in a firebase database that can be viewed by the staff members. On placing an order a unique order id is generated. This is a combination of the username, date and time. This serves as a unique primary key in the database. Using this orderid, the user can retrieve his/her order after it has been completed.

On placing this order, it is added to 2 databases, one which is accessible by the staff and one, from which each ordered item can be seen by the user and admin. From the latter, data cannot be deleted.

5. Viewing personal order history:

Each user has access to his/her previously ordered eatables, which is maintained in a permanent database using the firebase.

### 3.2 Sprint 2:

- Receiving orders
- Completing the orders
- Notifying the customer upon completion
- Storing the completed items
- View past orders

#### 3.2.1. Staff and Admin Stories

##### 3.2.1.1 Must have:

- Record of the orders which can be viewed by the staff
  - The staff member goes to pending orders, and views them.
- Completing the orders
  - The staff member should be able to select the completed order and delete it.
- Completed items should be stored to maintain the record
  - Admin should be able to see and monitor the placed and completed orders.

##### 3.2.1.2 Could have:

- Editing the menu
  - The staff can be allowed to add/edit/delete items from the existing menu.
- View user info
  - The admin can be allowed to view the users details.

#### Staff's side:

##### 1. Changes in the menu:

The staff can edit/add/delete items from the menu which is maintained in the firebase database. This is possible since the staff has add and delete authorities for the menu database while the user can only read.

##### 2. Viewing and deleting pending orders:

The firebase database which contains the customer's order list, can be viewed and deleted from. Each order has a unique order number and on completion of the order this id number can be deleted along with the order, notifying the customer. This is implemented in the backend using java and creating instances and references of the database in use.

#### Admin's side:



View all past orders:

1. The admin can view all the orders placed by all the users that are stored in the permanent firebase database.

View user's/staff's information:

2. The admin can view every registered member's information from the firebase database.