



## **Online Bakery Project**

Done by:

**Sara Limooee 9632421**

**Negin Khalifat 9632461**

**Fateme Masoudi 9632440**

**Melika Zare 9632456**

Teacher:

**Dr. Alipour**

Fall 1399-1400

## Table of Contents

Introduction .....	4
Complete application .....	4
Running back-end .....	4
Test .....	4
Running front-end .....	5
Back-end .....	6
DBMS .....	6
Account .....	7
Admin .....	7
Customer .....	8
Employee .....	8
Confectioner .....	9
Sweets .....	10
Multi-Tiered Cake .....	10
Birthday items .....	10
Gift card .....	11
Discount .....	11
Payment .....	11
Cost .....	12
Order status .....	12
Delivery information .....	13
Order .....	13
Vehicle .....	14
Delivery system .....	14
Complaint .....	15
Post .....	15
Comment .....	15

Front-end .....	16
Splash activity .....	16
Login activity .....	17
Register activity .....	18
Main activity .....	19
Change password .....	20
Profile activity .....	21

## Introduction

This project is going to be an android application for ordering sweets and cakes easily from any confectionary you want. This is going to be an android application.

### ❖ **Back-end:**

In the below link, the back-end of this application and relation between different classes in implemented in Java. The link to the back-end sources code is:

[Online-Bakery: Application for Bakers and Bakeries to work online](#)

### ❖ **Front-end:**

The front-end of this project, as mentioned above, is going to be an android application. The link to the front-end sources code is:

[OnlineBakeryUI: Online bakery android application for bakers & bakeries](#)

### ❖ **Design**

Also, you can check the design of the project from the below link:

[OnlineBakeryDesign](#)

## Complete application

Complete android application will have 3 sides from ordering until delivering the order.

1. First of all, customer request an order through the app and will wait for the confectioner to accept it.
2. Confectioner will accept or discard the order through his account and this will notify customer of the status of his order.
3. If the order is accepted by confectioner, it will be sent to delivery system so that an employee or employees be assigned to ship the order to the customer. After delivering it to customer, the employee must press “Delivered” button in his account through the android app.

## Running Back side project

To run the back-end side of the project implemented in Java, you need to download the complete project in the back-end link mentioned in introduction.

In order to run the project, you can create a new project on an IDE e.g., NetBeans or IntelliJ and then run the “main.java” file. In main, an instance of Test class is called that tests almost all the features implemented in our project.

You can check the complete film of running back-end of the project in the below link:

[back-end film](#)

## Test

In this class, the below steps are tested sequentially:

- Delivery Employees Sign Up
- Add Vehicles
- Bakers & Bakeries Sign Up
- Show All Bakeries with Details
- Show All Bakers with Details
- Show Result from Query Search by Name of Bakeries
- Show Result from Query Search by Name of Bakers
- Customers Sign Up
- Forgot Password Scenario
- Employees Sign Up
- Buy a Gift Card
- Order Multi-tiered Cake
- Make Sweets by Confectioner
- Add Discount by Confectioner
- Add Birthday Items to Bakery
- Create Post by Bakery
- Comment on Post by Customer
- Like Post by Customer
- Login Scenario
- Order Scenario
- Create a Compliment

## Running front side project

To run the front-end side of the project, first you have to install android studio. Then, you can install the application on android studio virtual device or you can enable debugging mode in your android phone and then choose your phone to install the application on.

- ✓ To enable debugging mode of your android phone:

**Open settings/ System window/ Developer options/ check USB debugging**

You can also check the complete film of running front-end android application in the below link:

[front-end film](#)

## Back-end

### DBMS

DBMS class contains all the lists that must be saved in a database for the application to work. It is implemented using **Singleton pattern**. DBMS has the following lists:

- Username to hashed password for all users
- Security questions: for registering in the application
- Admins
- Bakeries
- Bakers
- Customers
- Employees
- Vehicles
- Orders
- Order-Employee map: to show which employees ship the order
- Discounts
- Posts
- Comments
- Complaints

All the necessary functions to access lists in database are implemented. For connecting the app to a permanent database, this class functions must be replaced by functions of connecting a database.

## Account

In this application we have 4 major roles:

- Customer
- Employee
- Confectioner
  - ❖ Bakery
  - ❖ Baker (individual bakers working from home)
- Admin

Similar functionality of all users is implemented in Account. Account has below fields:

- Unique ID (assigned to users when signing up)
- Username (chosen when signing up, must be unique)
- Password
- First Name
- Last Name
- Address
- Contact Number
- Wallet (for saving money in application & pay from wallet later)

We store hash of the password in the database for security. Whenever a user signs up in our application, he/she should enter necessary information of the above fields. In case of forgetting password, user must answer security questions again and set a new password.

“Ask Permission” method is implemented so that in special cases, admin be able to give access for changing password without answering questions.

Users can remove their account in the application by deactivating it. But, his/her information in the database will not be deleted and can be recovered later.

Also, a first signed up date field and last login date to application are stored for each user. User can log out from the application by calling “LogOut” method.

## Admin

Admin is implemented using a **Singleton pattern**. The main purpose of admin is to communicate with database which can be accessed only if the user role is admin.

Admin is also a user. So, it has a wallet which is used as application fund. It also controls information about discounts paying money to employees and confectioner. This is completely discussed in Order and Customer sections.

## Customer

Initially customer can order sweets and birthday items to bakeries. But actually, before this event customer can design its own sweet. So, we have list of designs in customer class.

Customer can create order by calling “createNewOrder()” method with a list of sweets and birthday items. After that he can add and remove another sweet to the order with methods available from Order.

Every customer can buy gift card for other customers(target). By calling method “BuyGiftCardTo()”, customer specified details of gift card and at the end pay the cost with wallet or online. Afterward this money sent to admin wallet whom can manage and show this amount of money to target customer. The target customer while purchasing can use this amount of money.

## Employee

Employees are another type of user which extends Account class. One more difference of employee with other users is in its registering that does not need any security question since admin will register them in this app and database in order to check their qualifications.

Employee has the following fields besides Account fields:

- Score
- Number-of-people-scored (to update its score each time someone gives a score)
- Is-busy (shows if the employee is currently free to be assigned for shipping an order or is on the way)

Employee has 2 important functions:

- Deliver-order: called when employee says that the order is delivered to the customer and it calls the finish-Order function of Order class that changes the order-Status of order to DELIVERED
- Ruin-order: called when employee can't deliver the order to the customer intact and it calls the ruin-By-Employee functions of Order class that changes the order-Status of order to RUINED\_BY\_EMPLOYEE



## Confectioner:

Confectioner interface is implemented using **Factory pattern**. It can be a baker or a bakery.

Confectioners are also a type of user, so besides Account fields, they have below attributes:

- Name
- Description

Bakery has a list of order sweets, ready sweets and birthday items and baker has a list of order sweets, that the customer can order.

The confectioner has following abilities:

- Create sweets, add this to the list of sweets, remove a sweet from the list
- a score that we can get this with “getScore()” function. The score value in DBMS is obtained by averaging the score of confectioner sweets.
- Add discount
- Create post
- Accept order (the confectioner sees the customer's order and specifies the confectioner status for each part of the order to say if accepted or not)
- “getProfit(Date start , Date end)” function shows orders from the start date to the end date and the profit that the confectioner has made from

The bakery can create birthday item by calling “CreateBirthdayItem()” method.

## Sweets

We use a **Decorator pattern** for implementing the “Sweets” class. This class has the following fields:

- Id (automatically incremented)
- Description (about details of ingredients of sweets)
- Total cost
- Total germs
- score
- name
- NumCustomerForScore (Customers that rated to this sweet until now)
- Fee (fee for the baker or bakery that made this sweet)
- Images
- TypeOfSweets (that illustrate which kind of sweet is it. for example:CAKE, TART, COOKIE, UNKNOWN)

When we want to make some kind of sweets for example cake, the cake class has a builder to create a cake with appending decorators. The actual cost of the sweet is OrderCost that get\_OderCost() method returns it.

Customer can also rate the sweets that he/she ordered.

## Multi-Tiered Cake

Customers can make a multi-tiered cake with their own design. First, they choose the tire, and then after choosing the cake, they can design it by some designer that suggested and set the color of each tire. This class has the following fields:

- tire
- CakeWithDesign (ArrayList<Design>)

A new multi-tiered cake is created by calling function “MakeMultitieredCake()” method.

## Birthday Items

We use **Factory pattern** for implementing the birthday items e.g., candle, snow spray, balloon, etc.

**“ Confectioner profit from the sale of birthday item = cost – purchase price ”**

You can get information of birthday Item by “getDescription()” method.

## **Gift Card**

Customers can buy a gift card for another customer and it can be used to reduce orders costs. (This is a kind of discount for the customer who has it).

This class has the following fields:

- From (Customer)
- Owner (Customer)
- Price
- Note
- Remaining cost

A new gift card is created by calling function “BuyGiftCardTo()” method.

## **Discount**

All admin, baker and bakery can create a discount that can be applied to the customer's order. This class has the following fields:

- Id (automatically incremented)
- Name (reason for this discount)
- Percent
- Start & End Date
- Creator ID
- MaxUsed (max number of customers who can use this discount)
- NumberUser: (number of customers who used this discount)

## **Payment**

All payments are handles using Payment class. Payment fields are as following:

- Tracking code (a random decimal strings generated by Tracking-Code class)
- Date
- Amount
- Discription
- Payment type (from-wallet, pay-online)
- Payment status (failed, successful)

Wallet will be charged through creating an online payment. Then, it can be used to pay for orders online or buy a gift card.

## Cost

All discounts are considered to be used only on sweets and items costs, not the delivery cost. All customer has 20% discount for their first order which is handled by admin and is removed from offering application discounts list after the first time.

For considering discounts in payment of orders, we search for discounts if the confectioner settled any discount for a special event or our application has special discount for customers. Each order can have at most two discount one from application one from confectioner. Only the maximum discount from confectioner discounts list is selected and become the victorious discount.

It is also the same for application discounts list and we choose the highest discount. There is only one exception for “first ordering discount” which always has more priority than others.

Gift cards are also counted in calculating the final cost. Customer can choose a gift card to use if he has any, and the cost is reduced as equal as the amount of gift card. If There is still some money in the gift card, it can be used for reducing delivery cost.

## Order Status

An order can have the following status from the first time a customer creates a new order until the last level of being delivered or etc. The below list, shows the complete FSM of our program:

- **ORDERING\_BY\_CUSTOMER**: customer can edit order
- **FINALIZED\_BY\_CUSTOMER**: no more edit in the order is allowed
- **CANCELED\_BY\_CUSTOMER**: canceled by the customer in FINALIZED mode
- **PENDING\_CHOSEN\_BAKER**: choose one baker from the list and ask for confirm
- **CANCELED\_BY\_BAKER**: baker can cancel an order
- **ACCEPTED**: order accepted by baker
- **SET\_DELIVERY**: assign an employee to the order
- **PAYED**: pay cost of order
- **IN\_PROGRESS**: it is becoming ready
- **DONE**: it is ready but not delivered
- **RUINED\_BY\_EMPLOYEE**: the order was ruined by employee
- **WAITING**: it must wait for the delivery
- **ON\_THE\_WAY**: Order is now on the way to customer
- **CANCELED\_BY\_DELIVERY**: there is shipping available to deliver the order
- **DELIVERED**: it is delivered to customer

## **Delivery Information**

By the time the customer is ordering anything he wants, he must fill the delivery information fields that where and when he wants to receive the order. He can also add some notes on the order that confectioner or employee need to know.

In delivery information class, information such as below are mentioned:

- Id (automatically incremented)
- Delivery time
- Delivery address
- Actual delivery time
- Transfer price: which will be calculated by the source & destination addresses

## **Order**

Customer creates an object of order in its methods. Order has below fields:

- ID (automatically incremented)
- OrderStatus
- Customer
- Confectioner
- Delivery Information
- Expected Date for delivery
- List of Sweets
- List of birthday items
- Payment
- List of Discount (not in all orders)
- Note (can be set by customer or confectioner or employee)

Before customer finalizing the order, he can edit what he ordered. He can also cancel the order before confectioner accepting the order.

Confectioner sends a list of acceptance on every sweet and item of order. If order is completely accepted, delivery information is set and the price of transferring is calculated considering the destination address the use gives and calls delivery system to add the order to its queue.

Then a payment object is created to pay the cost. This money will be initially sent to admin's wallet and after confectioner gives the order to the employee, confectioner is paid. Also, after the employee shipped the order safely to customer, he will be paid the fee.

At the end, customer can give score for both sweets and employee service as well.

## Vehicle

Some vehicles are registered in our list of vehicles and deliveries are done using these vehicles. Vehicles have below fields:

- ID (automatically incremented)
- Plaque
- Type: Motor, Car, Truck
- Is-busy: shows if the vehicle is currently free to be used or is on the way

## Delivery System

In this class, orders are assigned to employees to be delivered by.

All orders are sent to this part after being confirmed and stored in a queue of orders which is always sorted by order-priority by calling sort-Order-Queue-Based-On-Priority function. All orders can have a priority that will be set by admin, default for all is 0.

In function assign-Employees-To-Orders, as it is written in the comment in DeliverySystem.java file, it must assign an employee or employees who are free for delivering an order.

Also, it must check if more than 1 order in order-Queue are in a close geographic span and are not too far, assign both to the same employee to improve the performance of delivering.

Besides, it checks if the order is not too heavy or its total grams is not more than a range and it can be delivered by a motor rather than a car; or if It is a big wedding cake that must be delivered by a truck.

For now, it just gets the first free employee and the first free vehicle to be assigned to order and is updated in order-Employee-Map in DBMS.

order-Employee-Map has a structure as below:

**Map<Pair<Integer, Integer>, Pair<List<Employee>, Vehicle>>**

2 scenarios can happen in this function:

1. Currently, we are capable of delivering the order which notifies the order that it is now ready to be sent.
2. Currently, we don't have any employee or vehicles, so we notify the order that it will be sent later.

## **Complaint**

In this class, anyone who wants to make a complaint can write something to admin and admin can later check out the complaints and answer them.

This class has the following fields:

- Complainant: This is an object of type of class Account which can be a customer, a confectioner or an employee
- Created date
- Responded date
- Title
- Text
- Responded: a Boolean that shows if the complaint is answered or not.

A new complaint is created by calling function create-Complaint.

## **Post**

All admin, baker and bakery can create a Post. This class has the following fields:

- Id (automatically incremented)
- Images
- Caption
- Likes
- Author Id

Each Account can do the following actions on posts:

- like Post (“addLike()” method )
- Delete like (“deleteLike()” method)
- create comment for the Post (“addComment()” method)

## **Comment**

Each Account can create a comment for a post. This class has the following fields:

- Id (automatically incremented)
- Text
- Post Id
- Author Id

## Front-End

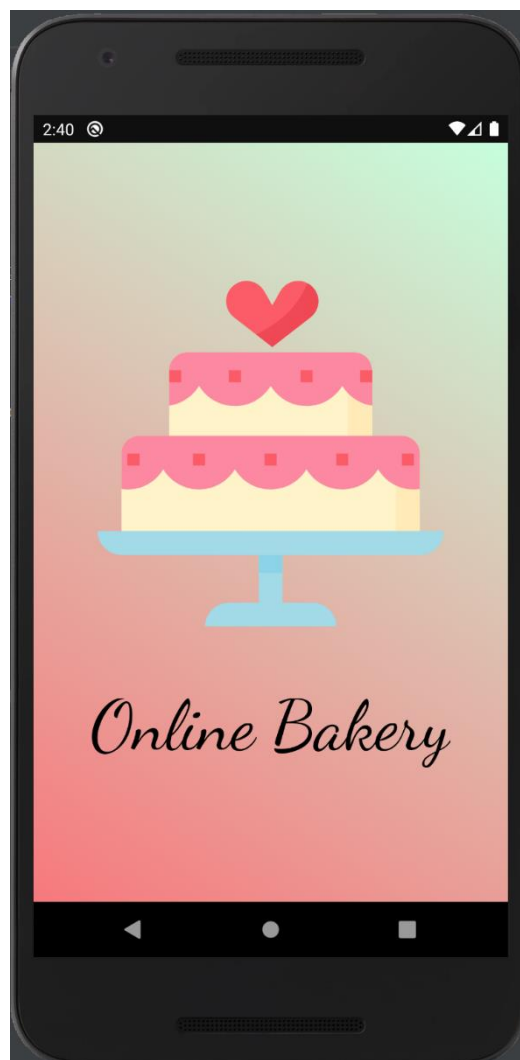
Classes described above are all used classes in the online bakery project. As mentioned before, the project is going to be an android application which gives easy and fast access to all users.

For writing the android app, all these classes must be moved to the android studio project and make instance of them in related activities codes.

For now, the following activities for all users have been implemented:

## Splash Activity

Splash activity is the splash screen with an animation running for 4 seconds when the app starts.

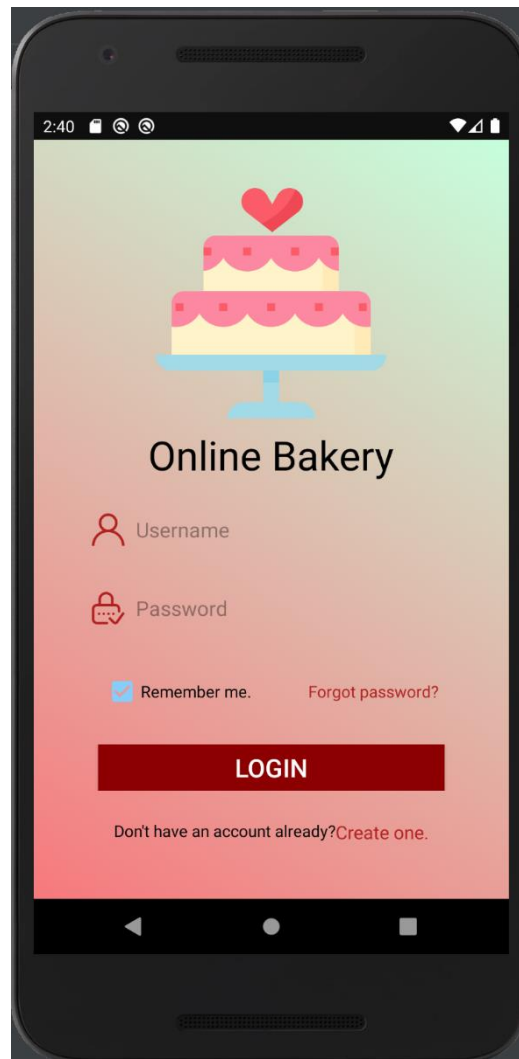




## Login Activity

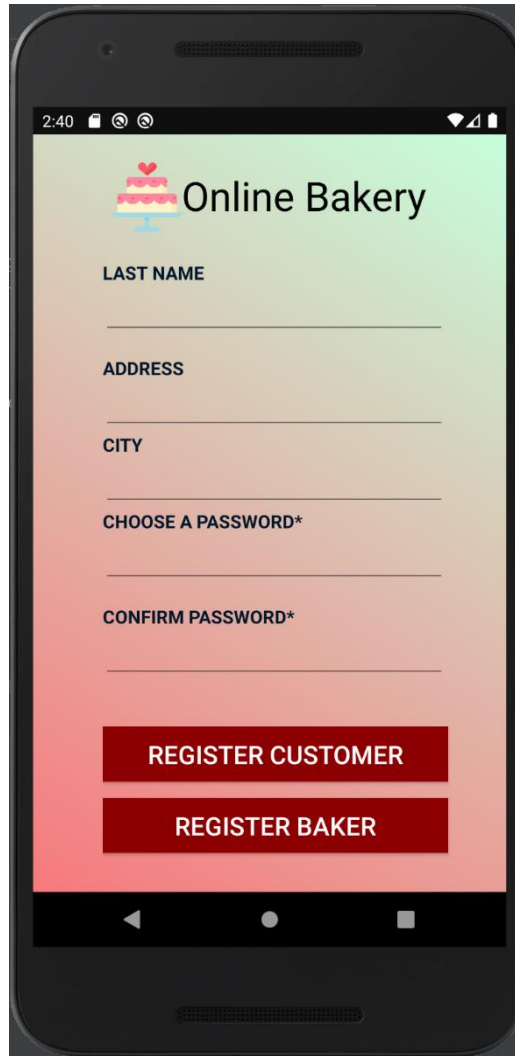
Users can enter their username and password to enter the application. They can also check the “Remember me” check box to be logged in in the app the next time they start the application.

If they do not have an account already, they can create a new account using register activity.




## Register Activity

Users must enter the necessary and can fill the optional fields for registering in the application. They can then press registering as a new baker or a new customer.



2:40

 Online Bakery

LAST NAME

ADDRESS

CITY

CHOOSE A PASSWORD\*

CONFIRM PASSWORD\*

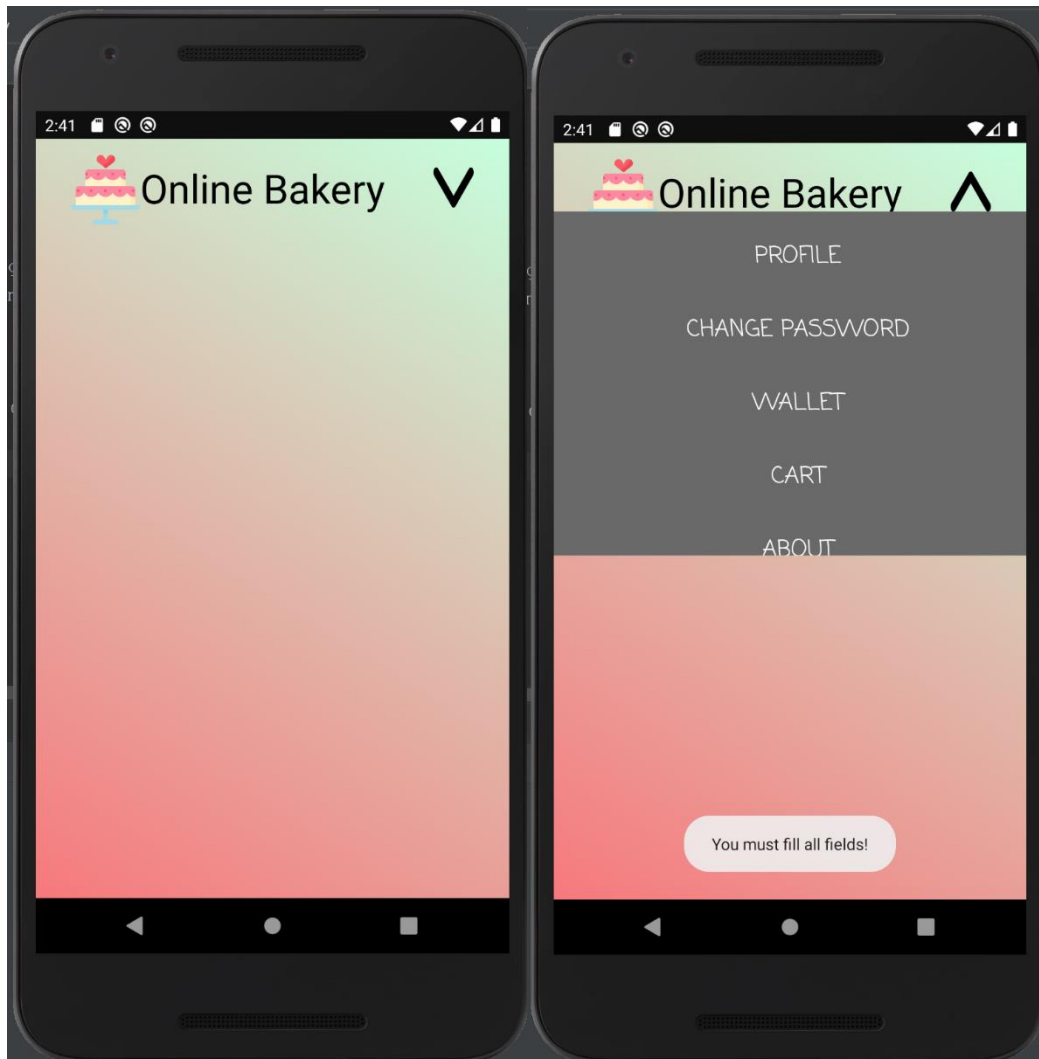
REGISTER CUSTOMER

REGISTER BAKER

## Main Activity

There will be 3 different main activities for different users, while the activities implemented currently in the front-end android app link mentioned before, are the same for all users.

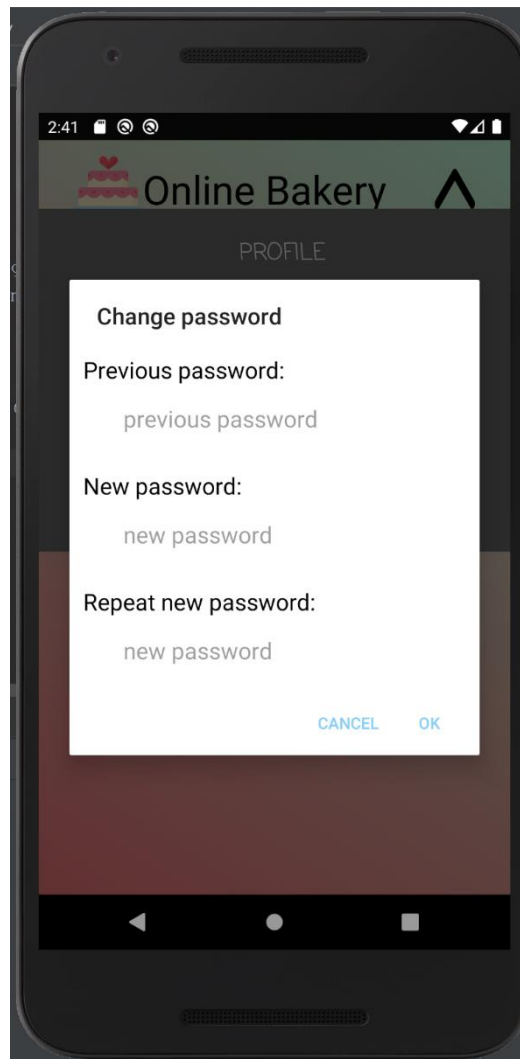
In future, admin may also be able to do his actions through an app; since it will be much easier.



## Change Password

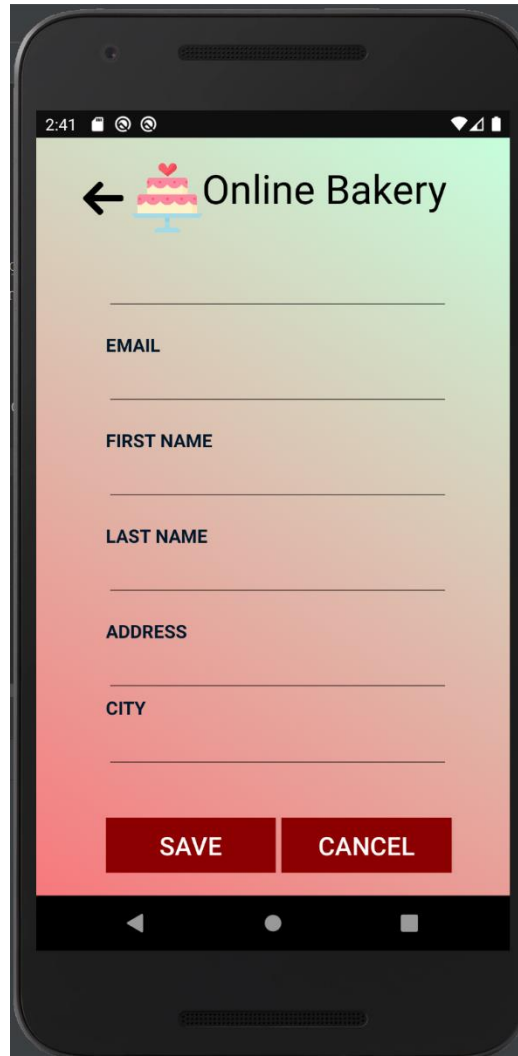
They can also change their password after writing their previous and new password.

Remember that this process differs from forgetting password; in case of forgetting password, they must answer security questions in order to redirect to entering new password activity.



## Profile Activity

Users can view or edit their profile in this activity and save or cancel their editing.



The screenshot shows a mobile application interface for editing a profile. At the top, there is a status bar with the time 2:41 and various icons. Below the status bar, there is a header area with a back arrow, a cake icon, and the text "Online Bakery". The main content area contains several input fields for profile information: EMAIL, FIRST NAME, LAST NAME, ADDRESS, and CITY. At the bottom, there are two buttons: "SAVE" and "CANCEL". The background of the form has a light green to light red gradient.

2:41

← 🍰 Online Bakery

EMAIL

FIRST NAME

LAST NAME

ADDRESS

CITY

SAVE CANCEL