King Saud University
College of Computer and Information Sciences
Department of Information Technology

IT 320 Course Project
Term-2, 1445H

Sprint-1
< The Blissful Touch >

Prepared by

<Shahad AlHaddad, 443203031>
<Sarah AlZuman, 443200744>
<Yara AlSubhi, 443201013>
<Reem AlNasser, 443200497>
<Shooq AlDossary, 443200601>

Supervised by
< Ghaida AlFayez >

Table of Contents

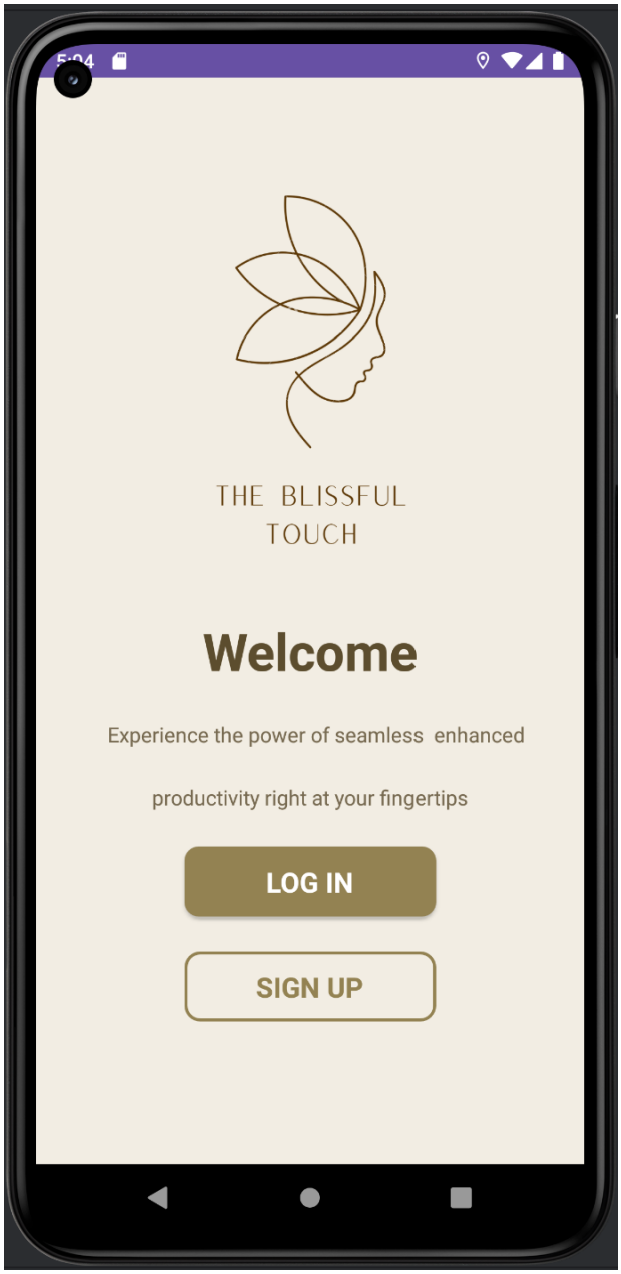
Contents

1 GitHub Link

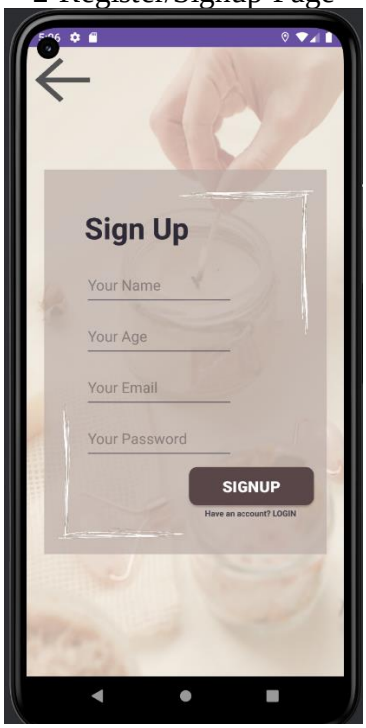
https://github.com/shahad789/BlissfullTouch

2 Interfaces

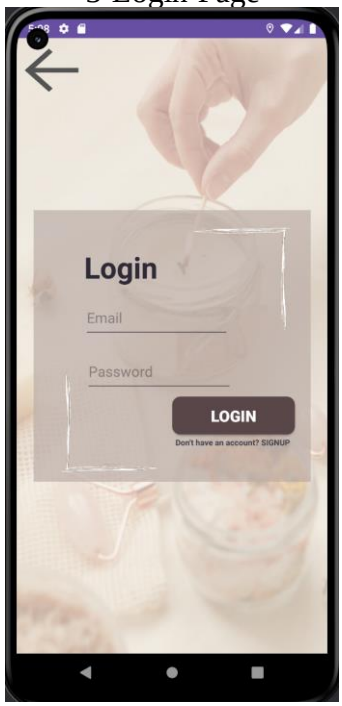
1-Main Display Page:



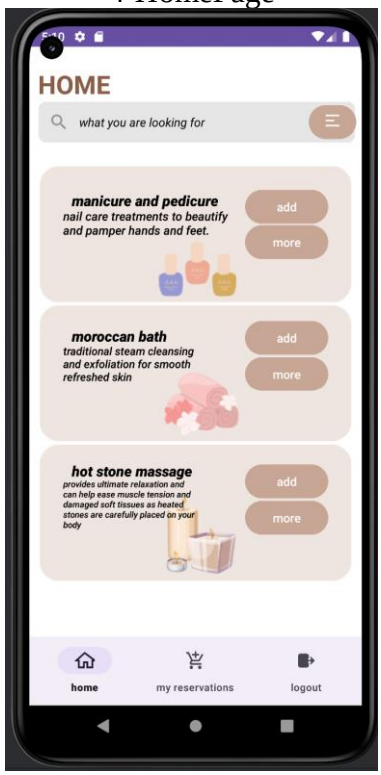
2-Register/Signup Page



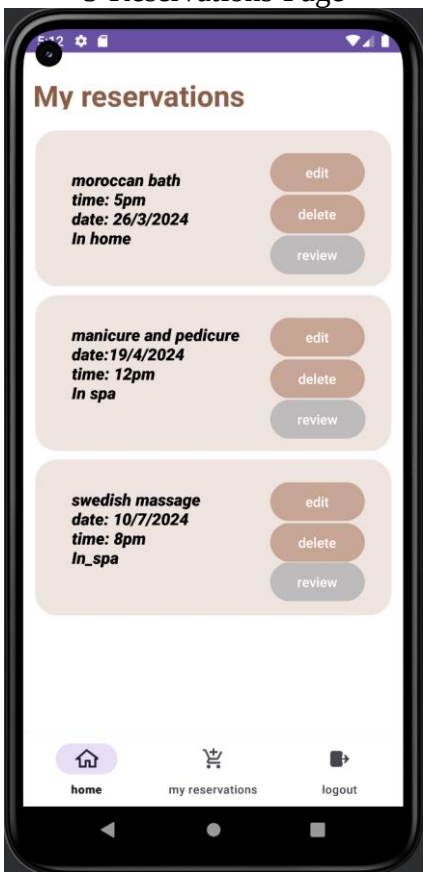
3-Login Page



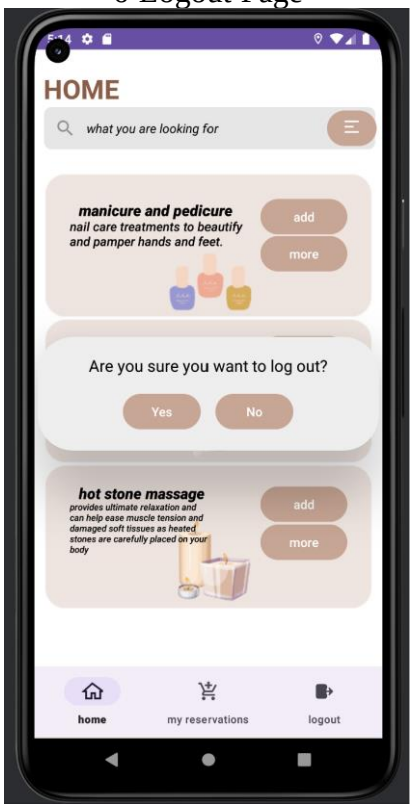
4-HomePage



5-Reservations Page

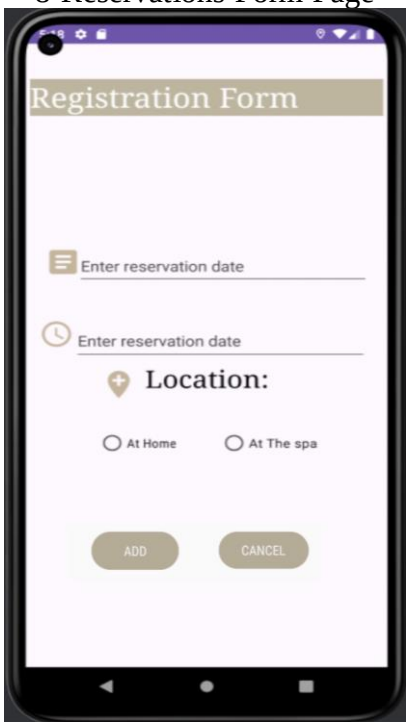


6-Logout Page

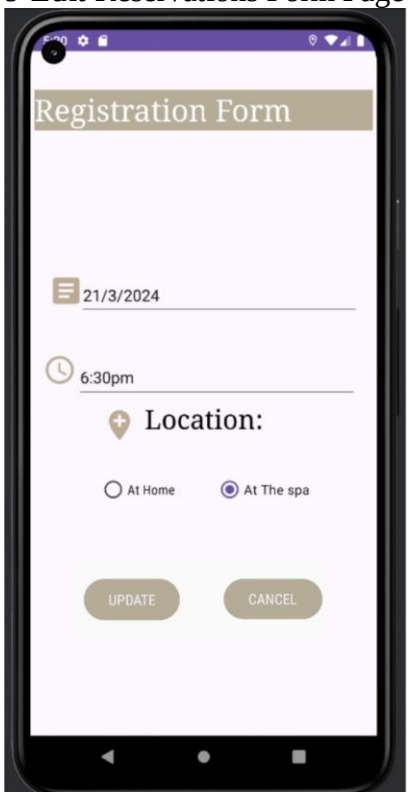


7-Delete Reservation Page

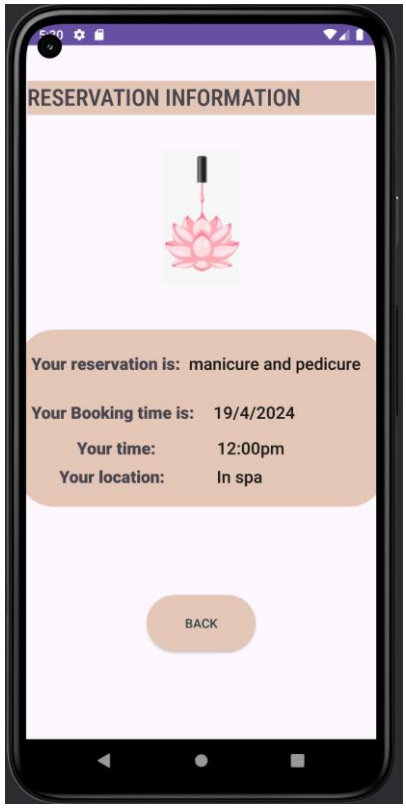
8-Reservations Form Page



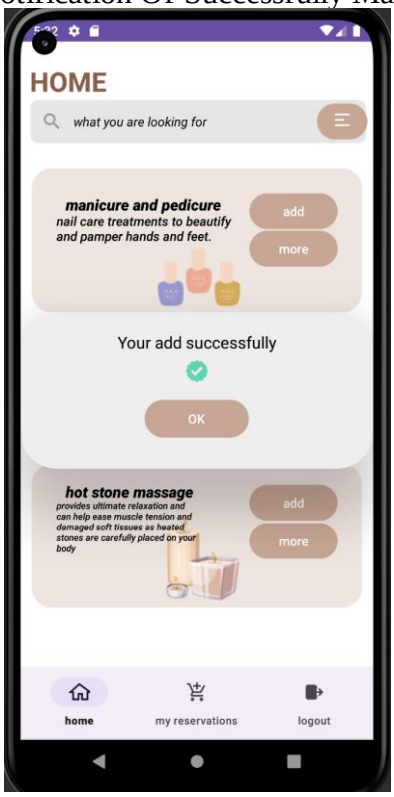
9-Edit Reservations Form Page



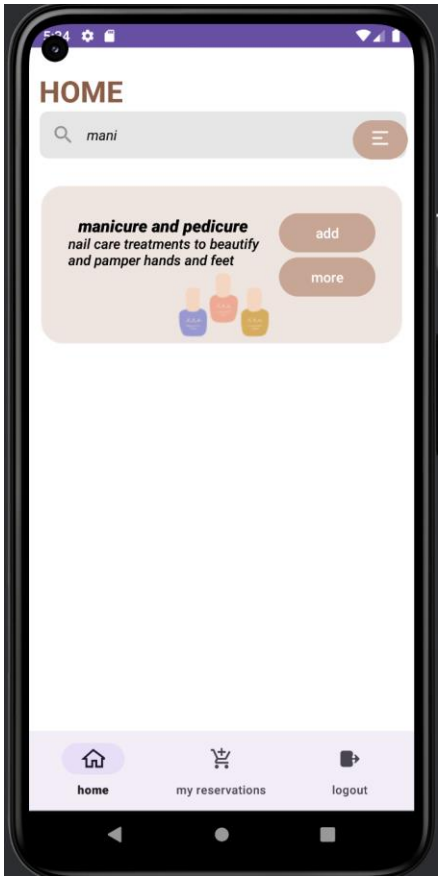
10-View Reservations Page



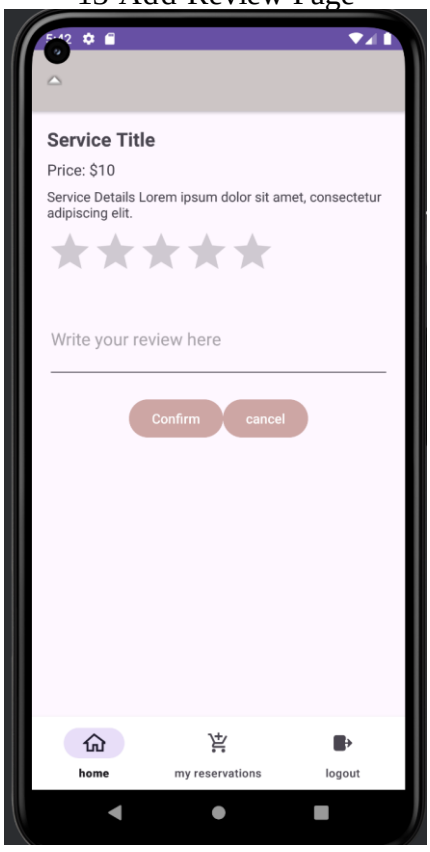
11-Confirmation Notification Of Successfully Making A Reservation



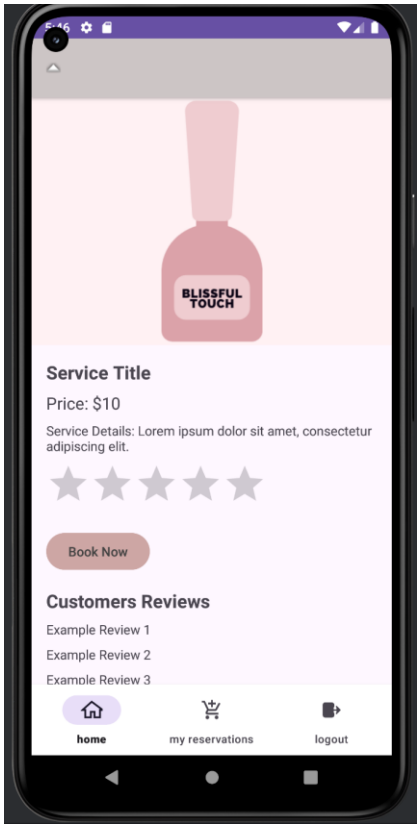
12-search



13-Add Review Page



14-Product Details Page


BLISSFUL TOUCH
Service Title
Price: $10
Service Details: Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Book Now
Customers Reviews
Example Review 1
Example Review 2
Example Review 3
home
my reservations
logout

3 DATABASE

```
package com.example.blissfulltouch;


import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class MyDatabaseHelper extends SQLiteOpenHelper {

    private static final String DATABASE_NAME = "BlissfulTouch.db";
    private static final int DATABASE_VERSION = 1;

    // Table names and column names
    private static final String TABLE_USERS = "users";
    private static final String COLUMN_EMAIL = "email";
    private static final String COLUMN_NAME = "clientname";
    private static final String COLUMN_PASSWORD = "pass";
    private static final String COLUMN_AGE = "age";

    private static final String TABLE_SERVICES = "services";
    private static final String COLUMN_SERVICE_NAME = "servicenm";
    private static final String COLUMN_DESCRIPTION = "description";
    private static final String COLUMN_REVIEW = "review";

    private static final String TABLE_RESERVATION = "reservation";
    private static final String COLUMN_RESERVATION_ID = "id";
    private static final String COLUMN_LOCATION = "location";
    private static final String COLUMN_TIME = "time";
    private static final String COLUMN_DATE = "date";
    private static final String COLUMN_SERVICE_NAME_FK =
"serviceName";

    public MyDatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // Create users table
        String createUserTableQuery = "CREATE TABLE " + TABLE_USERS +
                "(" + COLUMN_EMAIL + " TEXT PRIMARY KEY, " +
                COLUMN_NAME + " TEXT, " +
                COLUMN_AGE + " TEXT, " +
                COLUMN_PASSWORD + " TEXT)";
        db.execSQL(createUserTableQuery);
```

```
        // Create services table
        String createServiceTableQuery = "CREATE TABLE " +
TABLE_SERVICES +
                "(" + COLUMN_SERVICE_NAME + " TEXT PRIMARY KEY, " +
                COLUMN_DESCRIPTION + " TEXT, " +
                COLUMN_REVIEW + " TEXT)";
        db.execSQL(createServiceTableQuery);

        // Create reservation table
        String createReservationTableQuery = "CREATE TABLE " +
TABLE_RESERVATION +
                "(" + COLUMN_RESERVATION_ID + " INTEGER PRIMARY KEY
AUTOINCREMENT, " +
                COLUMN_LOCATION + " TEXT, " +
                COLUMN_TIME + " TEXT, " +
                COLUMN_DATE + " TEXT, " +
                COLUMN_SERVICE_NAME_FK + " TEXT, " +
                "FOREIGN KEY(" + COLUMN_SERVICE_NAME_FK + ")
REFERENCES " + TABLE_SERVICES + "(" + COLUMN_SERVICE_NAME + "))";
        db.execSQL(createReservationTableQuery);
    }

    @Override
    //to update info in database like edit form
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_RESERVATION);
        onCreate(db);
    }

    // Method to add a user to the database
    public boolean addUser(String email, String name, String age,
String password) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(COLUMN_EMAIL, email);
        contentValues.put(COLUMN_NAME, name);
        contentValues.put(COLUMN_AGE, age);
        contentValues.put(COLUMN_PASSWORD, password);
        long result = db.insert(TABLE_USERS, null, contentValues);
        return result != -1;
    }

    // Method to add a new service to the database
    public boolean addService(String serviceName, String description,
String review) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(COLUMN_SERVICE_NAME, serviceName);
        contentValues.put(COLUMN_DESCRIPTION, description);
        contentValues.put(COLUMN_REVIEW, review);
```

```
        long result = db.insert(TABLE_SERVICES, null, contentValues);
        return result != -1;
    }



    // Method to retrieve user information by email
    public Cursor getUserInfo(String email) {
        SQLiteDatabase db = this.getWritableDatabase();
        return db.rawQuery("SELECT * FROM " + TABLE_USERS + " WHERE "
+ COLUMN_EMAIL + "=?", new String[]{email});
    }

    // Method to retrieve service information by service name
    public Cursor getServiceInfo(String serviceName) {
        SQLiteDatabase db = this.getWritableDatabase();
        return db.rawQuery("SELECT * FROM " + TABLE_SERVICES + " WHERE
" + COLUMN_SERVICE_NAME + "=?", new String[]{serviceName});
    }

    // Method to retrieve reservation information by service name
    public Cursor getReservationInfo(String serviceName) {
        SQLiteDatabase db = this.getWritableDatabase();
        return db.rawQuery("SELECT * FROM " + TABLE_RESERVATION + "
WHERE " + COLUMN_SERVICE_NAME_FK + "=?", new String[]{serviceName});
    }



    // Method to update reservation information
    public boolean updateReservation(int id, String location, String
time, String date) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(COLUMN_LOCATION, location);
        contentValues.put(COLUMN_TIME, time);
        contentValues.put(COLUMN_DATE, date);
        int result = db.update(TABLE_RESERVATION, contentValues,
COLUMN_RESERVATION_ID + "=?", new String[]{String.valueOf(id)});
        return result > 0;
    }

    // Method to delete a reservation from the database
    public boolean deleteReservation(int reservationId) {
        SQLiteDatabase db = this.getWritableDatabase();
        int result = db.delete(TABLE_RESERVATION,
COLUMN_RESERVATION_ID + "=?", new
String[]{String.valueOf(reservationId)});
        return result > 0;
    }
```

```
    public boolean checkUsername(String username) {
        SQLiteDatabase MyDB = this.getReadableDatabase();
        Cursor cursor = MyDB.rawQuery("SELECT * FROM " + TABLE_USERS +
" WHERE " + COLUMN_EMAIL + " = ?", new String[]{username});
        boolean exists = cursor.getCount() > 0;
        cursor.close(); // Close the cursor
        MyDB.close(); // Close the database connection
        return exists;
    }

    public boolean checkUsernamePassword(String username, String
password) {
        SQLiteDatabase MyDB = this.getReadableDatabase();
        Cursor cursor = MyDB.rawQuery("SELECT * FROM " + TABLE_USERS +
" WHERE " + COLUMN_EMAIL + " = ? AND " + COLUMN_PASSWORD + " = ?", new
String[]{username, password});
        boolean exists = cursor.getCount() > 0;
        cursor.close(); // Close the cursor
        MyDB.close(); // Close the database connection
        return exists;
    }


}
```