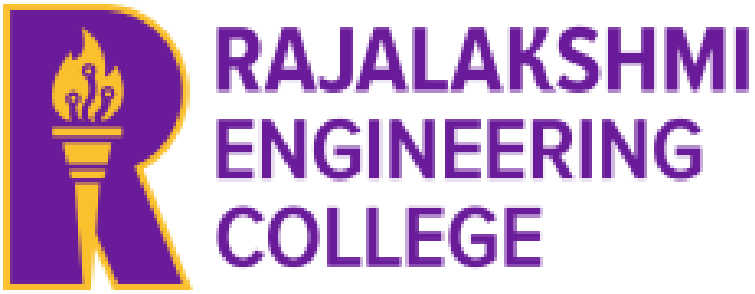


LOCALINK - LOCAL SERVICE MARKETPLACE

A MINI-PROJECT REPORT

Submitted by

SUBASH R 240701538
SIVAGURU D 240701517

in partial fulfillment of the award of the degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI
An Autonomous Institute

NOVEMBER 2025

BONAFIDE CERTIFICATE

Certified that this project **"LOCALINK - Local Service Marketplace"** is the bonafide work of **"SUBASH R, SIVAGURU D"** who carried out the project work under my supervision.

SIGNATURE
Mrs. M. ANITHA
ASSISTANT PROFESSOR SG
Department of Computer Science and Engineering,
Rajalakshmi Engineering College,
Chennai

This mini project report is submitted for the viva voce examination to be held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

Local services play a major role in urban and suburban life. The local market for services like plumbers, electricians, and tutors remains fragmented and often relies on outdated word-of-mouth methods. To address this gap, our team has developed a database-driven desktop application, "Localink." This system is designed to help a local community organize and access skilled professionals efficiently.

The main objective of this project is to create a two-sided marketplace that connects customers needing a service with verified local providers.

This system helps customers find, book, and review professionals. For providers, it offers a platform to create a profile, list their services, and manage incoming job requests. This allows skilled local individuals to beat the competition from larger, impersonal agencies by providing a trusted and efficient service.

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honorable chairman, **MR. S. MEGANATHAN**, and the chairperson, **DR. M.THANGAM MEGANATHAN** for their timely support and encouragement.

We are greatly indebted to our respected and honorable principal, **Dr. S.N. MURUGESAN,** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by our Head Of The Department, **Dr. E.M. MALATHY,** and our Deputy Head Of The Department, **Dr. J. MANORANJINI,** for being ever ever-supporting force during our project work

We also extend our sincere and hearty thanks to our internal guide, **Mrs. M. ANITHA**, for her valuable guidance and motivation during the completion of this project.

Our sincere thanks to our family members, friends, and other staff members of the computer science engineering department.

1. SUBASH R
2. SIVAGURU D

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

The project helps users find and book verified local service professionals. The platform categorizes providers by skill (e.g., plumbing, electrical) and allows customers to assess them based on profiles and community reviews. The necessary information about a provider's services, rates, and availability will be mentioned for the user's convenience.

1.2 SCOPE OF THE WORK

The "Localink" marketplace will help people access local professionals amidst the heavy demand for reliable services. It helps customers make informed decisions quickly and provides an easy-to-use platform for a wide range of people.

1.3 PROBLEM STATEMENT

The need for this project is that finding trustworthy local service providers is often difficult and inefficient. Customers must rely on word-of-mouth or unverified online listings. Simultaneously, independent, skilled professionals have limited marketing budgets and struggle to reach a local audience, facing competition from large, expensive corporations.

1.4 AIM AND OBJECTIVES OF THE PROJECT

The main objective of this project is to build a dual-role desktop application to connect customers with local service providers.

This system helps maintain a database of providers and their service offerings.

- To develop a secure login system with two distinct roles: Customer and Provider.
- To allow customers to search for providers, view profiles, and book services.
- To allow providers to create a public profile, list their services, and manage job requests.
- To implement a review and rating system to build community trust.
- To design and maintain a PostgreSQL database to store all user, booking, and review data.

This will allow local providers to compete effectively by showcasing their skills and reliability directly to the community.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

- **Processor**: Intel i3 or Equivalent
- **Memory Size**: 4GB (Minimum)
- **HDD**: 4 GB Remaining Disc Space (Minimum)

2.2 SOFTWARE SPECIFICATIONS

- **Operating System**: WINDOWS 10
- **Front - End**: Java (JavaFX)
- **Back - End**: PostgreSQL
- **Language**: Java, SQL
- **IDE**: IntelliJ IDEA

CHAPTER 3

MODULE DESCRIPTION

This application consists of two primary modules. When the program runs, it will present a login window. The person who interacts can log in as a Customer or as a Provider, each with a unique interface and capabilities.

The description of the modules are as follows:

1. Customer Module When a user logs in as a Customer, they are given access to the service-finding features.

- They can search for providers based on service type (e.g., "Plumber").
- They can browse a list of available providers and view their detailed profiles, including past reviews and hourly rates.
- They can initiate a booking request for a specific job.
- After a job is marked as complete, they can leave a rating and review for the provider.

2. Provider Module When a user logs in as a Provider, they access the service-management dashboard.

- They can create and edit their public-facing profile, including their bio, services offered, and rates.
- They receive notifications for new job requests from customers.
- They can view, accept, or decline pending job requests.
- Their public profile is automatically updated with new ratings and reviews from customers.

SAMPLE CODING

JavaFX Login Screen UI

The JavaFX code for creating the main login screen. It uses labels, text fields, and buttons for a clean user interface.

```
package com.localink.controller;

import com.localink.util.ViewNavigator;

import com.localink.service.AuthService;

import com.localink.model.User;

import com.localink.util.Session;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.Hyperlink;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextField;

import javafx.scene.control.Label;


public class LoginController {

   @FXML private TextField emailField;

   @FXML private PasswordField passwordField;

   @FXML private Button loginButton;

   @FXML private Hyperlink registerLink;

   @FXML private Label errorLabel;

   private final AuthService authService = new AuthService();

   @FXML
```

```
    private void onLogin(ActionEvent e) {
        String email = emailField.getText();
        String password = passwordField.getText();
        try {
            User user = authService.login(email, password);
            if (user == null) {
                errorLabel.setText("Invalid email or password");
                return;
            }
            Session.setCurrentUserId(user.getId());
            // Navigate by role if needed; for now, go to Categories flow
            ViewNavigator.navigate("/fxml/categories.fxml", "Localink - Browse
Categories");
        } catch (IllegalArgumentException ex) {
            errorLabel.setText(ex.getMessage());
        } catch (Exception ex) {
            errorLabel.setText("Login failed");
        }
    }

    @FXML
    private void onGoToRegister(ActionEvent e) {
        ViewNavigator.navigate("/fxml/register.fxml", "Localink - Register");
    }
}
```

Java JDBC Database Connection (DAO)

The database access code for validating a user's login. This `UserDAO` (Data Access Object) class uses JDBC to connect to the PostgreSQL database, execute a prepared SQL query, and return a `User` object if the login is successful**.**

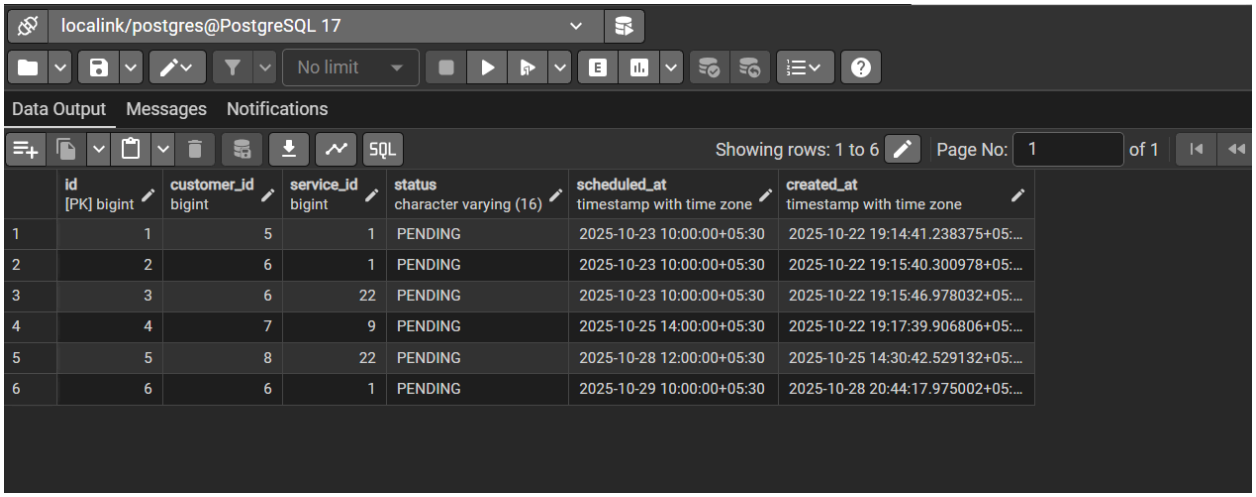
```
package com.localink.dao;

import com.localink.config.Database;
import com.localink.model.User;

import java.sql.*;

public class UserDao {
   public User findByEmail(String email) throws SQLException
{
       String sql = "SELECT id, name, email, password_hash,
role, created_at FROM users WHERE email = ?";
       try (Connection con = Database.getConnection();
            PreparedStatement ps =
con.prepareStatement(sql)) {
           ps.setString(1, email);
           try (ResultSet rs = ps.executeQuery()) {
               if (rs.next()) {
                   User u = new User();
                   u.setId(rs.getLong("id"));
                   u.setName(rs.getString("name"));
                   u.setEmail(rs.getString("email"));

u.setPasswordHash(rs.getString("password_hash"));
                   u.setRole(rs.getString("role"));
                   Timestamp ts =
rs.getTimestamp("created_at");
```

```
                    if (ts != null)
u.setCreatedAt(ts.toInstant().atOffset(java.time.ZoneOffset.
UTC));
                    return u;
                }
                return null;
            }
        }
    }

    public long insert(String name, String email, String
passwordHash, String role) throws SQLException {
        String sql = "INSERT INTO users (name, email,
password_hash, role) VALUES (?,?,?,?) RETURNING id";
        try (Connection con = Database.getConnection();
             PreparedStatement ps =
con.prepareStatement(sql)) {
            ps.setString(1, name);
            ps.setString(2, email);
            ps.setString(3, passwordHash);
            ps.setString(4, role);
            try (ResultSet rs = ps.executeQuery()) {
                if (rs.next()) return rs.getLong(1);
            }
        }
        throw new SQLException("Failed to insert user");
    }
}
```

Database Schema

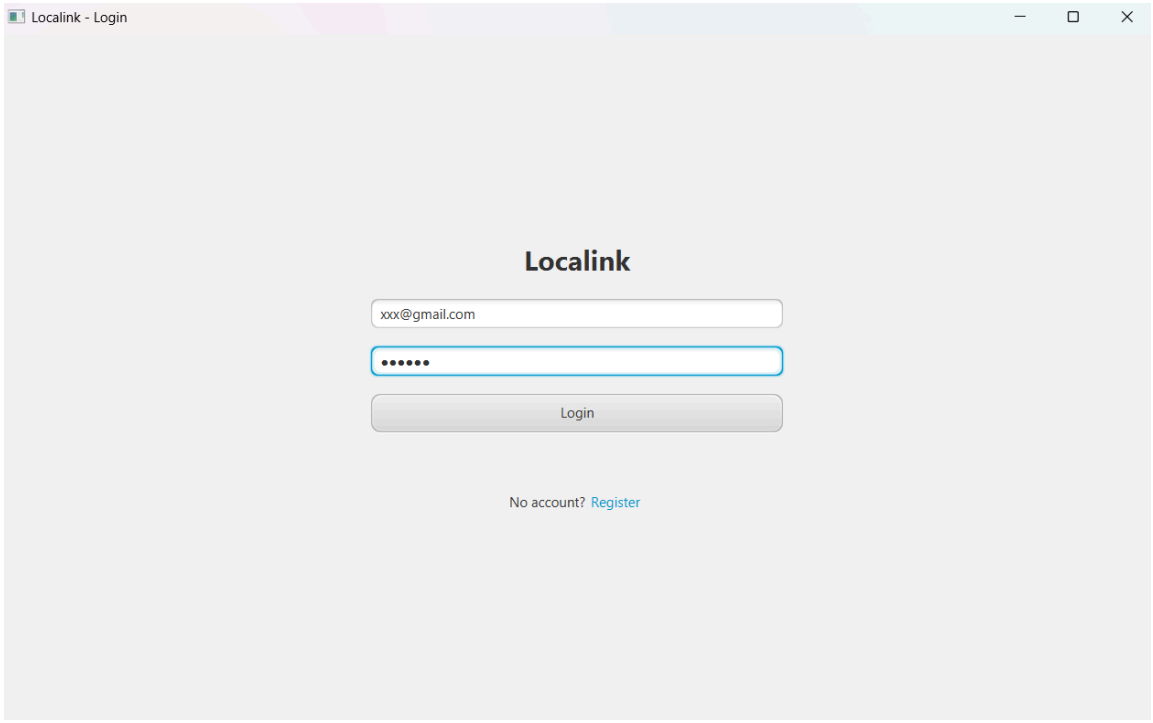
```
CREATE TABLE IF NOT EXISTS users (
   id BIGSERIAL PRIMARY KEY,
   name VARCHAR(120) NOT NULL,
   email VARCHAR(160) NOT NULL UNIQUE,
   password_hash VARCHAR(100) NOT NULL,
   role VARCHAR(16) NOT NULL DEFAULT 'CUSTOMER',
   created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
   CONSTRAINT chk_user_role CHECK (role IN
('CUSTOMER','PROVIDER','ADMIN'))
);

CREATE TABLE IF NOT EXISTS services (
   id BIGSERIAL PRIMARY KEY,
   provider_id BIGINT NOT NULL,
   title VARCHAR(160) NOT NULL,
   category VARCHAR(80) NOT NULL,
   description TEXT,
   hourly_rate NUMERIC(10,2) NOT NULL,
   is_active BOOLEAN NOT NULL DEFAULT TRUE,
   FOREIGN KEY (provider_id) REFERENCES users(id)
);


DO $$ BEGIN
   IF NOT EXISTS (
       SELECT 1 FROM pg_indexes WHERE schemaname = 'public'
AND indexname = 'ux_service_provider_title'
   ) THEN
       CREATE UNIQUE INDEX ux_service_provider_title ON
services(provider_id, title);
   END IF;
END $$;

CREATE TABLE IF NOT EXISTS bookings (
   id BIGSERIAL PRIMARY KEY,
   customer_id BIGINT NOT NULL,
```

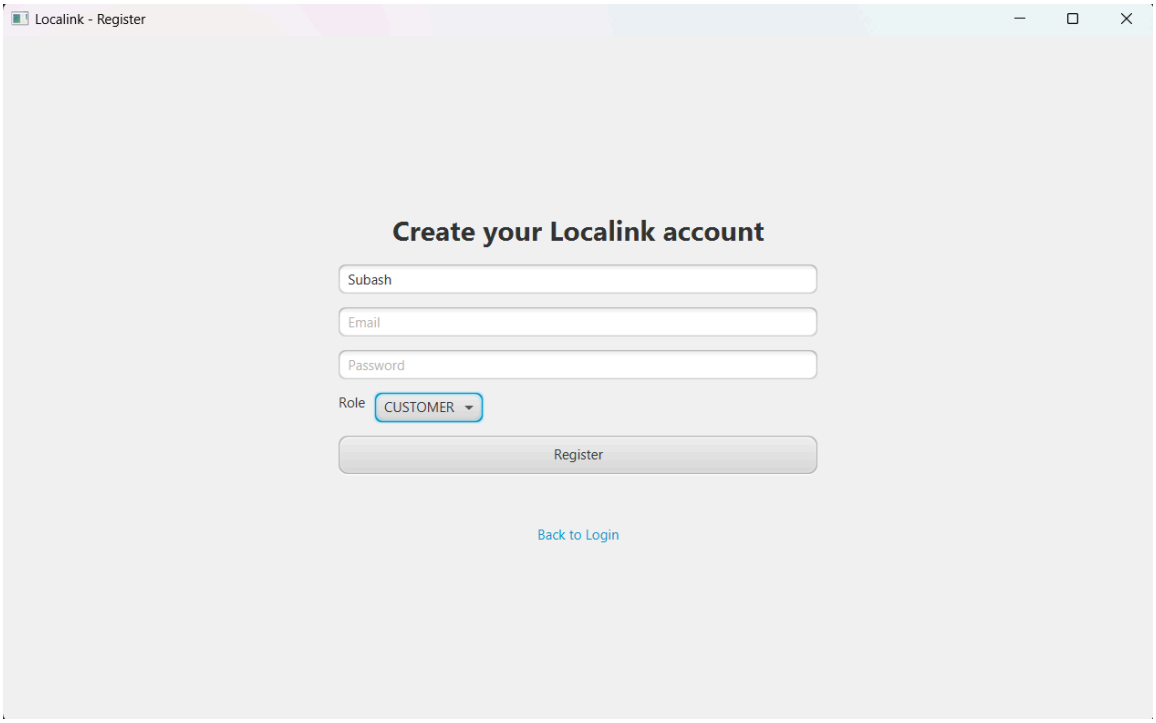
```
    service_id BIGINT NOT NULL,
    status VARCHAR(16) NOT NULL DEFAULT 'PENDING',
    scheduled_at TIMESTAMPTZ,
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (customer_id) REFERENCES users(id),
    FOREIGN KEY (service_id) REFERENCES services(id),
    CONSTRAINT chk_booking_status CHECK (status IN
('PENDING','ACCEPTED','REJECTED','IN_PROGRESS','COMPLETED','
CANCELLED'))
);

CREATE TABLE IF NOT EXISTS reviews (
    id BIGSERIAL PRIMARY KEY,
    booking_id BIGINT NOT NULL,
    rating SMALLINT NOT NULL CHECK (rating BETWEEN 1 AND 5),
    comment VARCHAR(1000),
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
    UNIQUE (booking_id),
    FOREIGN KEY (booking_id) REFERENCES bookings(id)
);
```



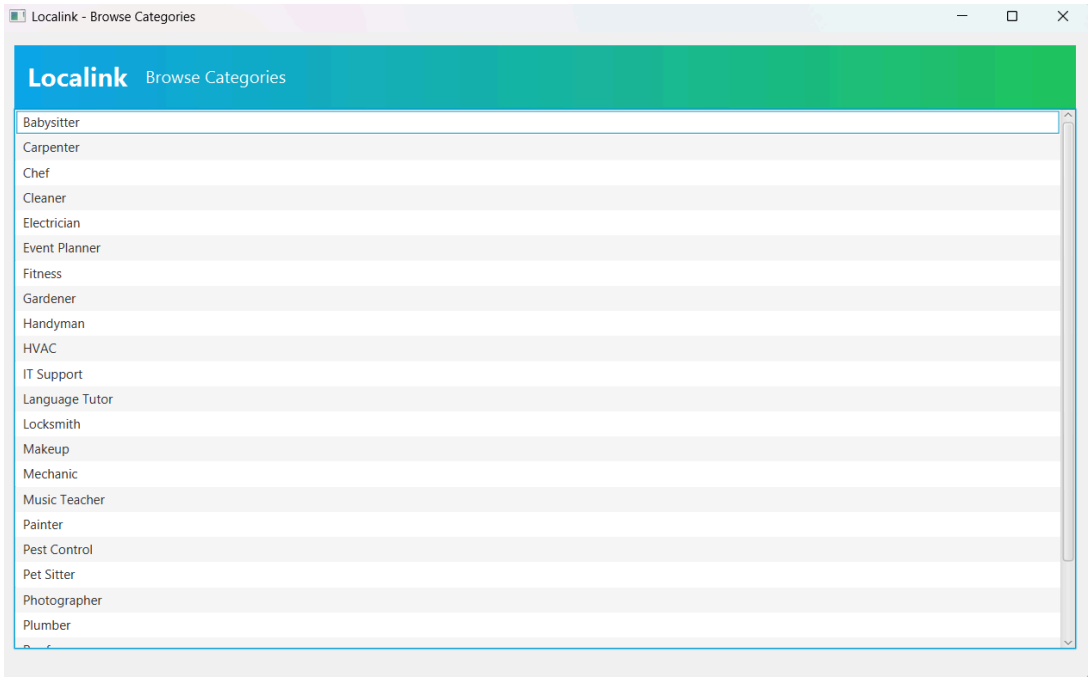
	id [PK] bigint	customer_id bigint	service_id bigint	status character varying (16)	scheduled_at timestamp with time zone	created_at timestamp with time zone
1	1	5	1	PENDING	2025-10-23 10:00:00+05:30	2025-10-22 19:14:41.238375+05:...
2	2	6	1	PENDING	2025-10-23 10:00:00+05:30	2025-10-22 19:15:40.300978+05:...
3	3	6	22	PENDING	2025-10-23 10:00:00+05:30	2025-10-22 19:15:46.978032+05:...
4	4	7	9	PENDING	2025-10-25 14:00:00+05:30	2025-10-22 19:17:39.906806+05:...
5	5	8	22	PENDING	2025-10-28 12:00:00+05:30	2025-10-25 14:30:42.529132+05:...
6	6	6	1	PENDING	2025-10-29 10:00:00+05:30	2025-10-28 20:44:17.975002+05:...

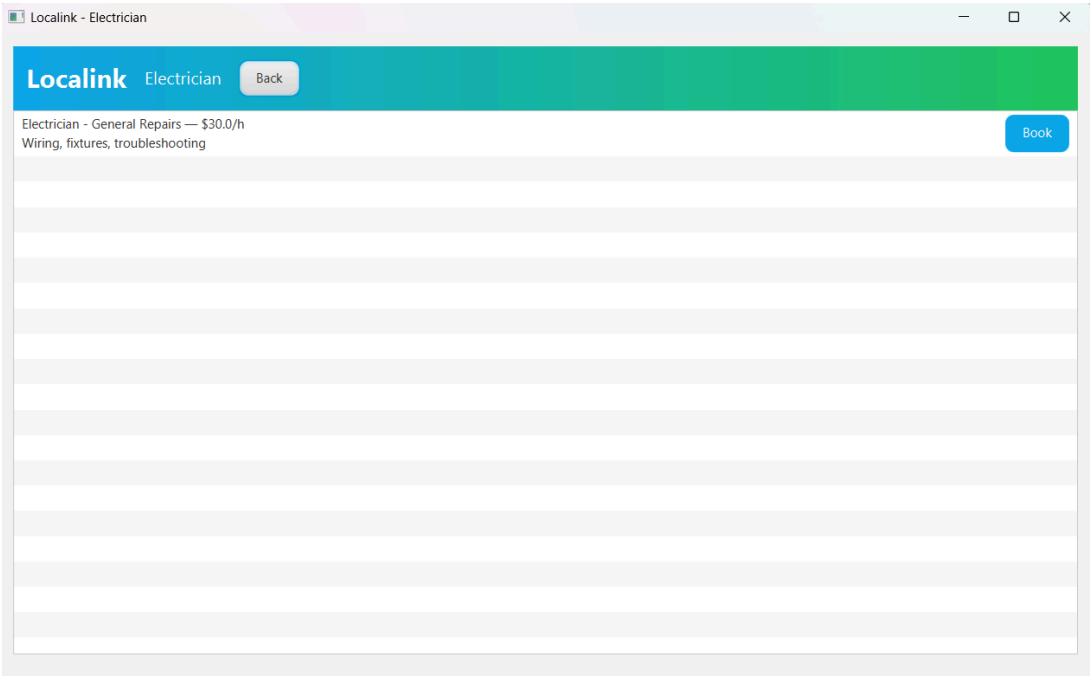
CHAPTER 5
SCREENSHOTS



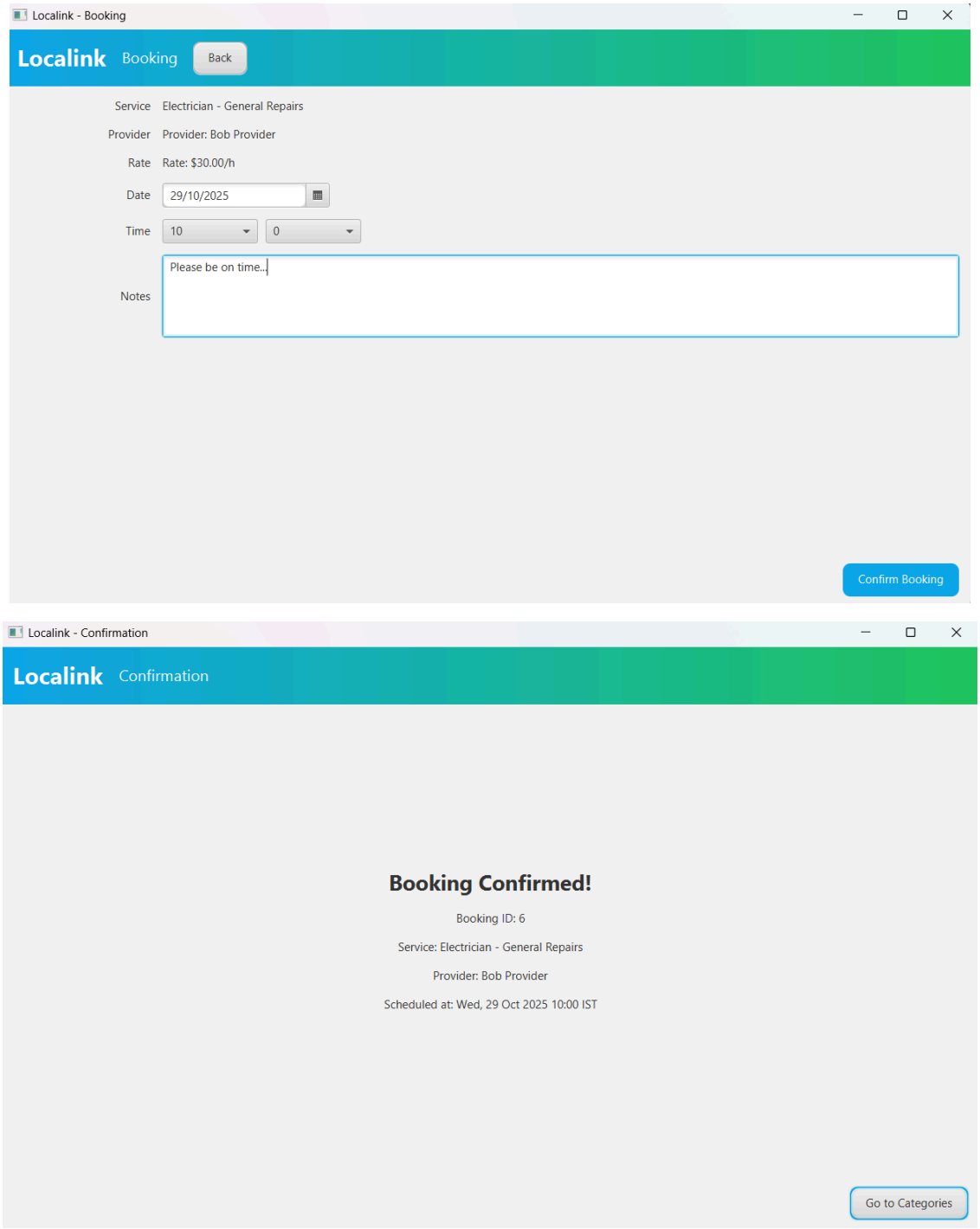


LOGIN AND REGISTER USER

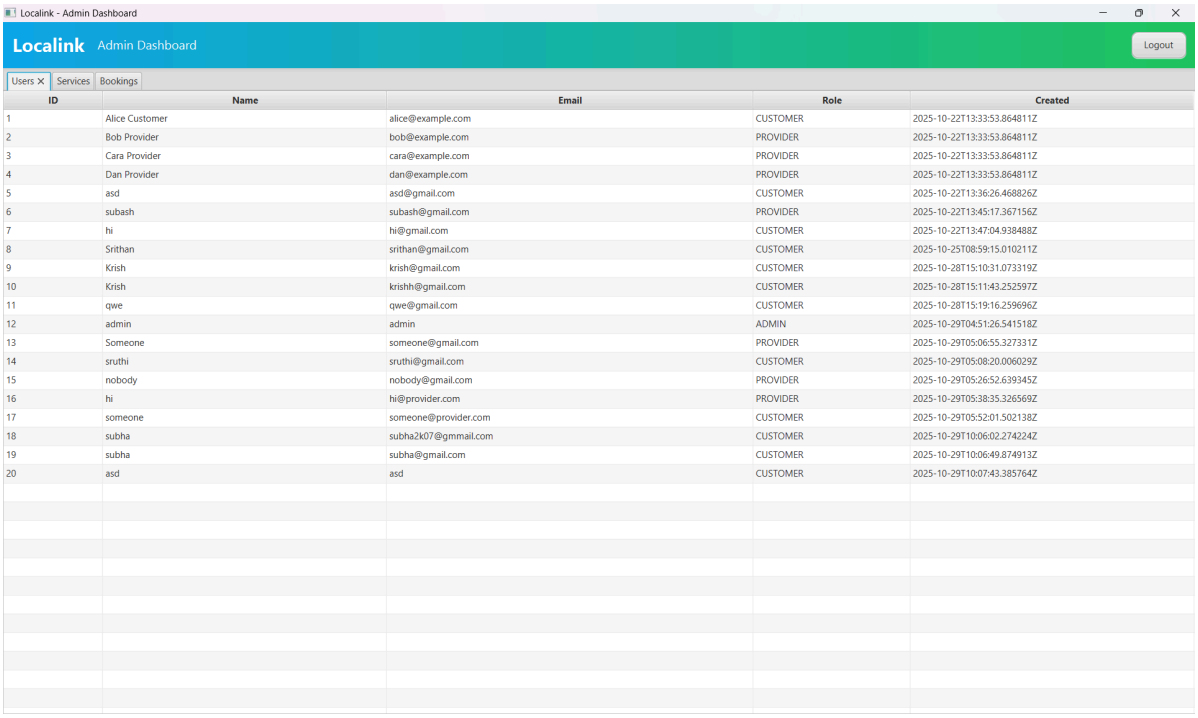




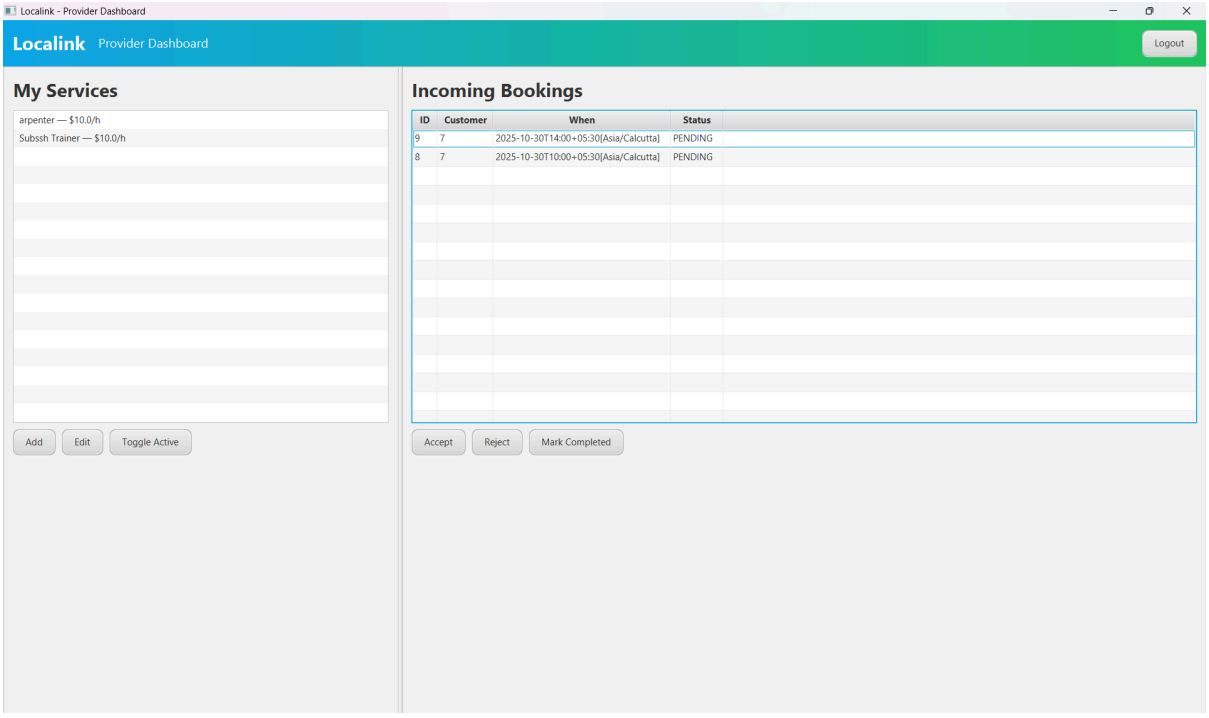
BROWSE SERVICES



BOOKING CONFIRMATION



ADMIN DASHBOARD



PROVIDER DASHBOARD

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

With the help of our project, customers can easily find, book, and review local service professionals, while providers are empowered with a tool to market their skills to a wider audience. The system successfully implements a dual-role platform using a JavaFX frontend and a robust PostgreSQL backend, demonstrating a clear separation of concerns and effective database management.

In the future, the project can be enhanced with several key features. A real-time chat module would allow customers and providers to communicate directly. Integrating a secure payment gateway would streamline the transaction process. Finally, developing a companion web or mobile application would make the platform accessible to an even larger audience and provide on-the-go convenience.

Hence, this project makes the local service marketplace more efficient and trustworthy for all users.

REFERENCES

1. **https://www.w3schools.com/postgresql /**(PostgreSQL Tutorial)
2. **https://openjfx.io/openjfx-docs/** (Official JavaFX Documentation)
3. **https://www.postgresql.org/docs/** (Official PostgreSQL Documentation)
4. **https://jdbc.postgresql.org/documentation/** (PostgreSQL JDBC Driver Docs)
5. **https://www.geeksforgeeks.org/javafx/**