TU REWARD DOCUMENTATION

Oscar Fabrizio de Alba Gtz.

1. Introduction

Overview

TuReward is a digital loyalty system designed to enhance customer retention for businesses such as restaurants, bars, and pharmacies. Through the use of QR codes, customers can easily collect points and redeem exclusive rewards, while businesses manage their loyalty programs efficiently.

Purpose

The primary purpose of TuReward is to simplify and streamline loyalty programs, making them more accessible and efficient for both businesses and customers.

Key Features

- Custom user management with separate types (admins and customers).
- QR code integration for point collection and reward redemption.
- Admin panel for managing rewards, adding points, and viewing transaction history.
- User panel for tracking points, viewing rewards, and linking to businesses.

---

2. Technologies Used

Backend

- Django
- Django REST Framework
- MySQL & SQLite

Frontend

- HTML
- CSS
- JavaScript

Design Tools

- Adobe Illustrator (for logos and illustrations)

---

3. Getting Started

Prerequisites

- Python 3.8+

Installation Steps

Clone the repository:

1. git clone <https://github.com/OscarFa28/TuReward>
2. cd TuReward

Create and activate a virtual environment:

3. python -m venv env
4. source env/bin/activate # For Linux/Mac
5. env\Scripts\activate # For Windows

Install dependencies:

6. pip install -r requirements.txt

Set up the database:

7. python manage.py makemigrations & python manage.py migrate

---

4. Usage Instructions

Running the Application Locally

- Start the server using the command: python manage.py runserver
- Open your browser and navigate to http://127.0.0.1:8000.

User Interface

Admin Panel

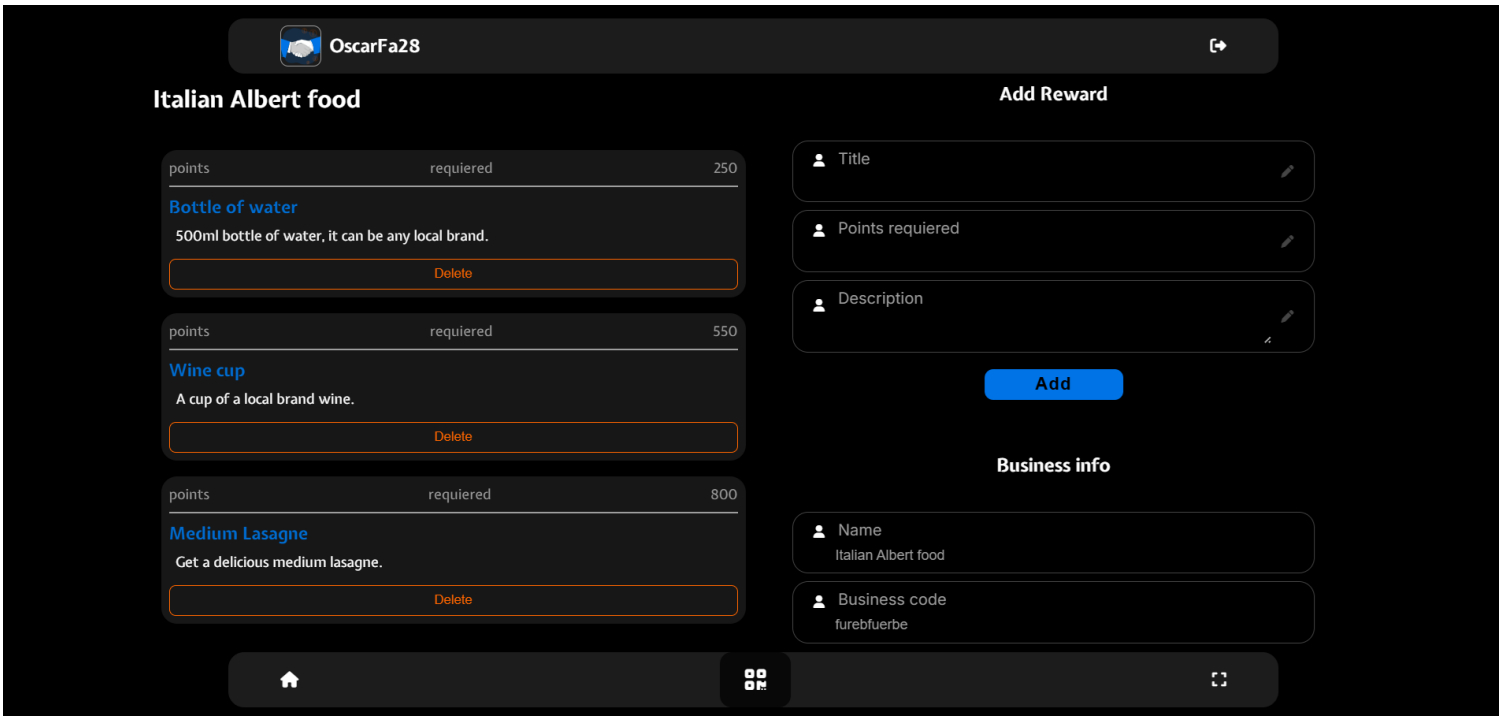
- Manage rewards, add points, and view transaction history.

User Panel

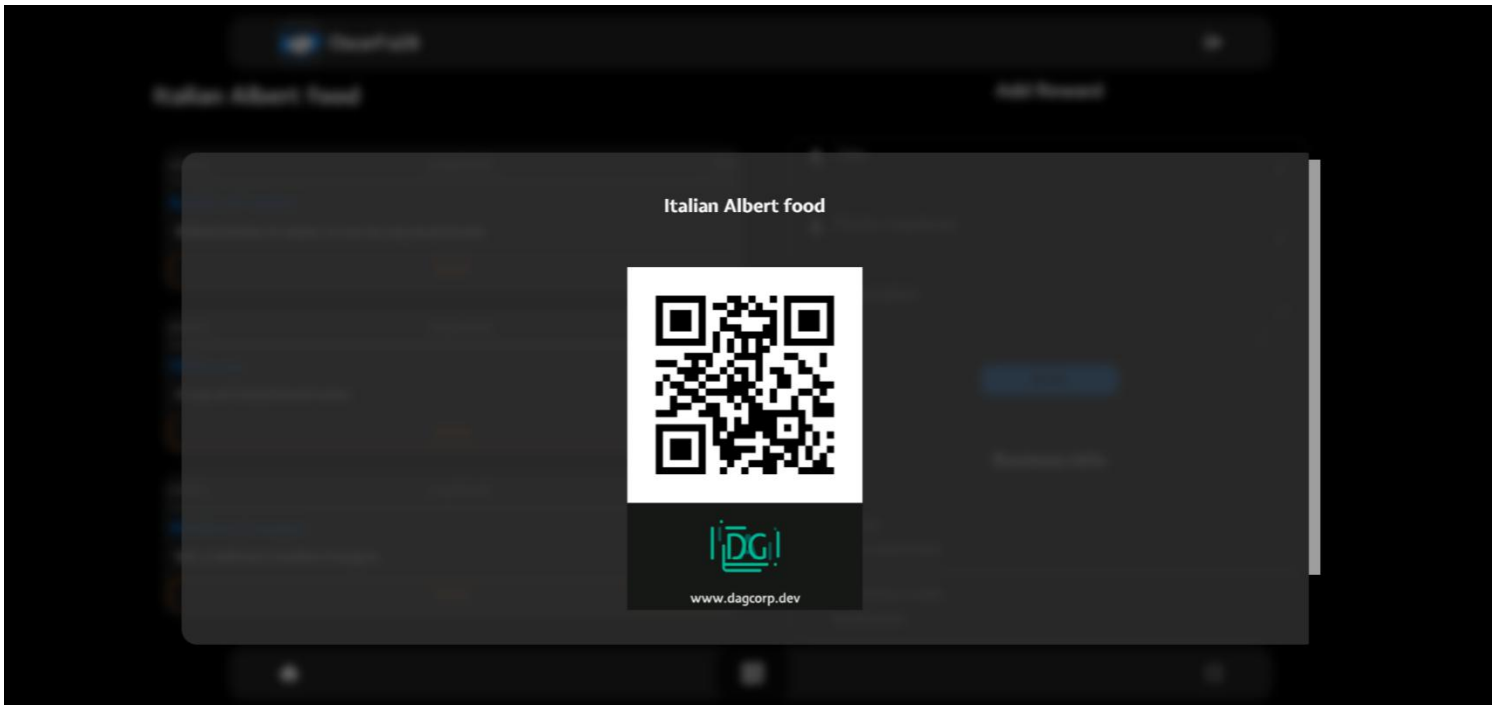
- Track points, view rewards, and link accounts to businesses using QR codes.

Screenshots

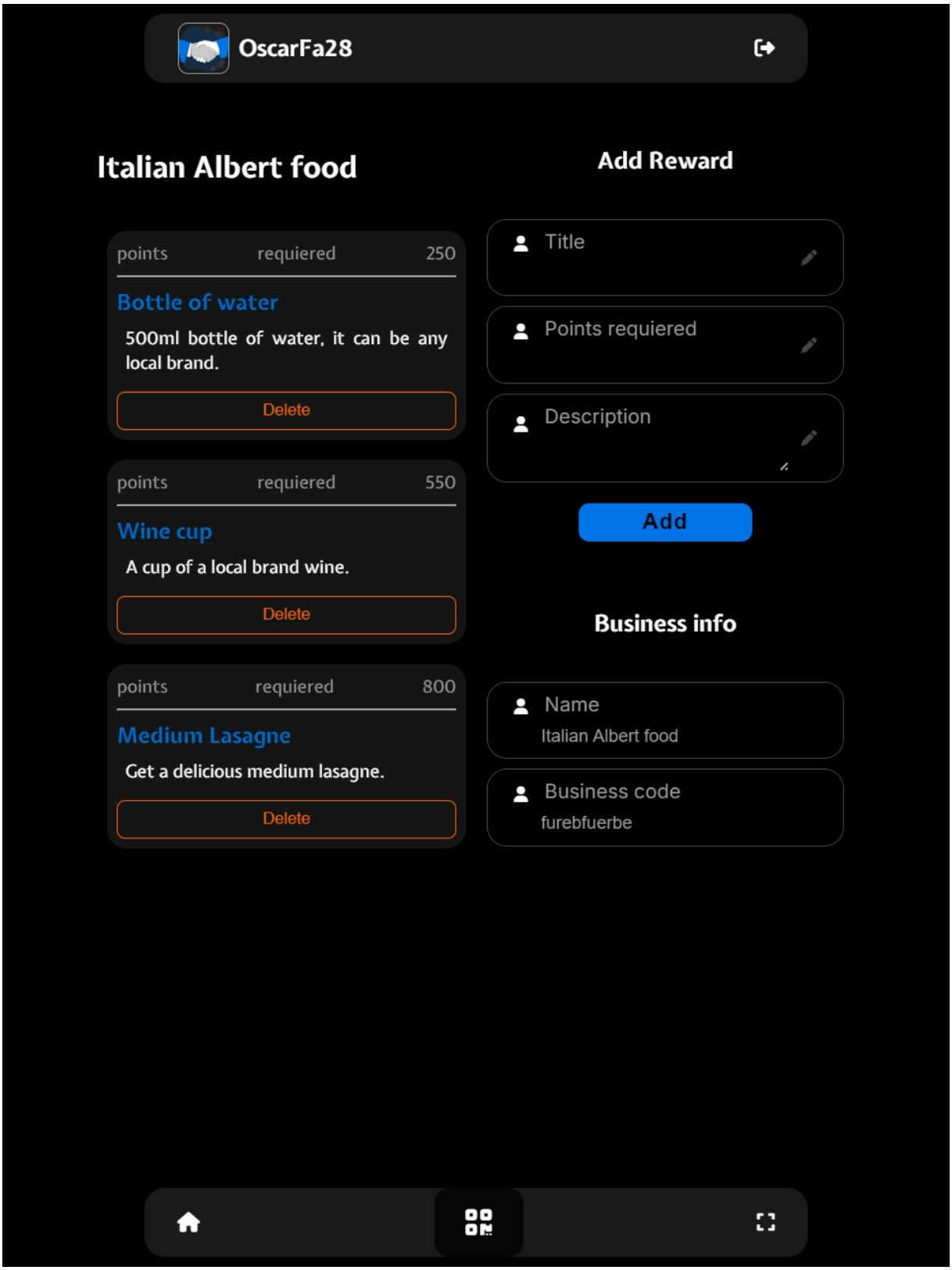
Admin panel (pc view)



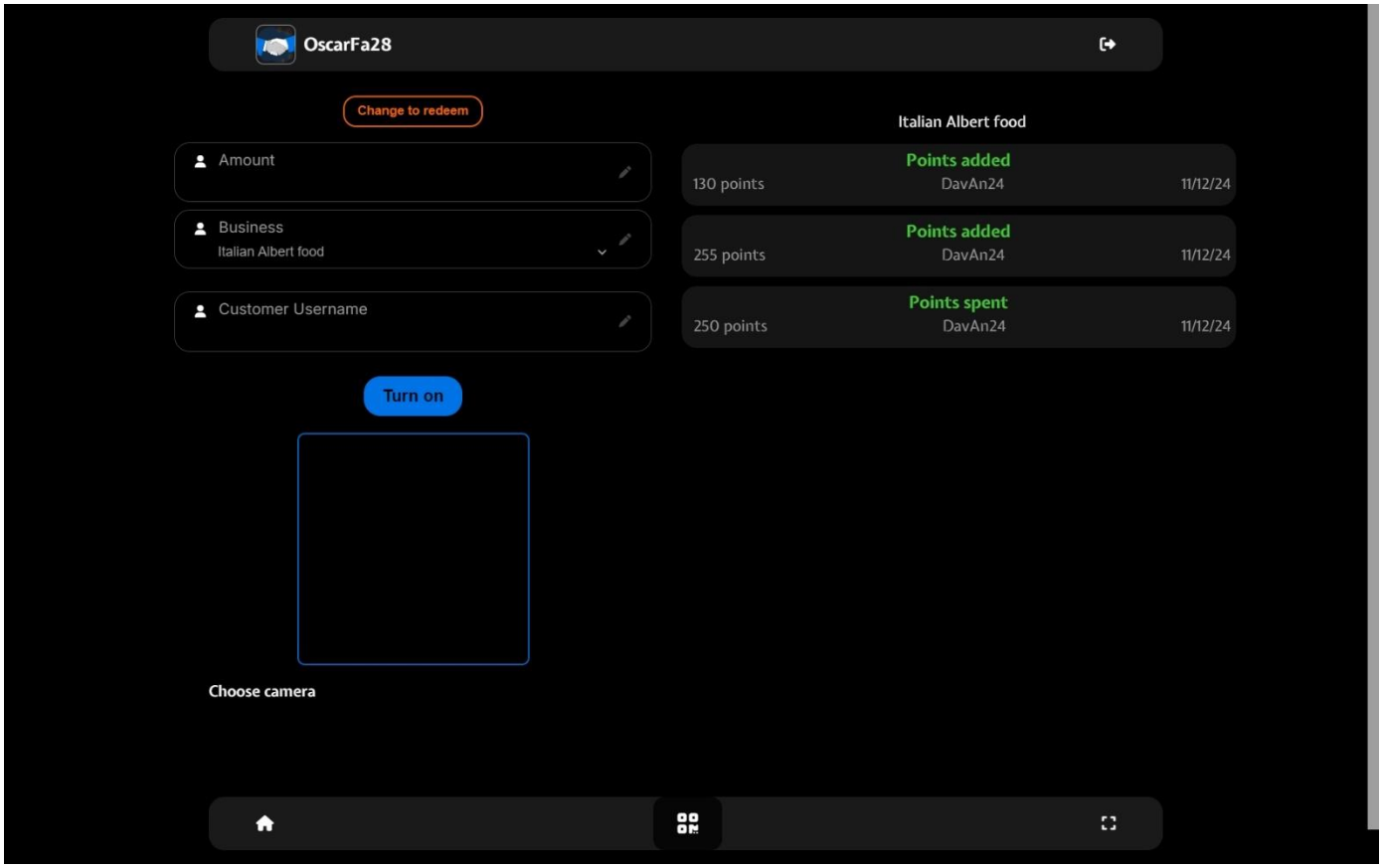
Business QR code (pc view)



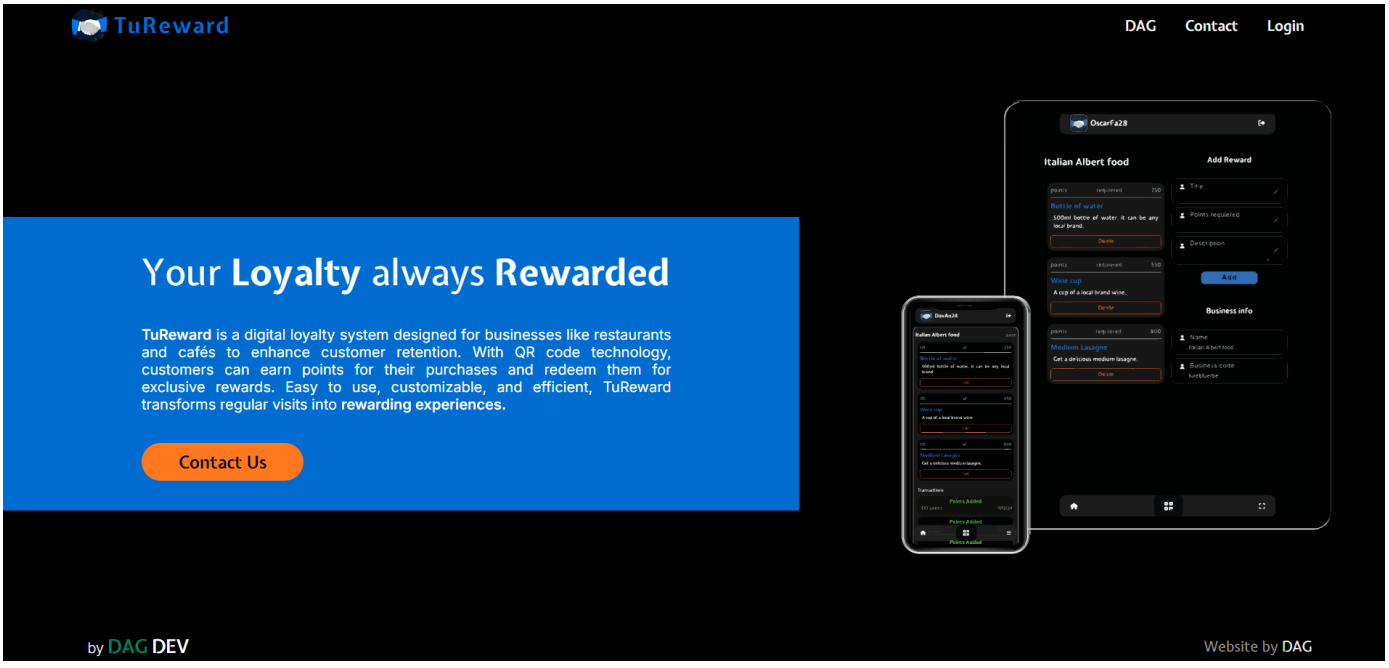
Admin panel (tablet view)



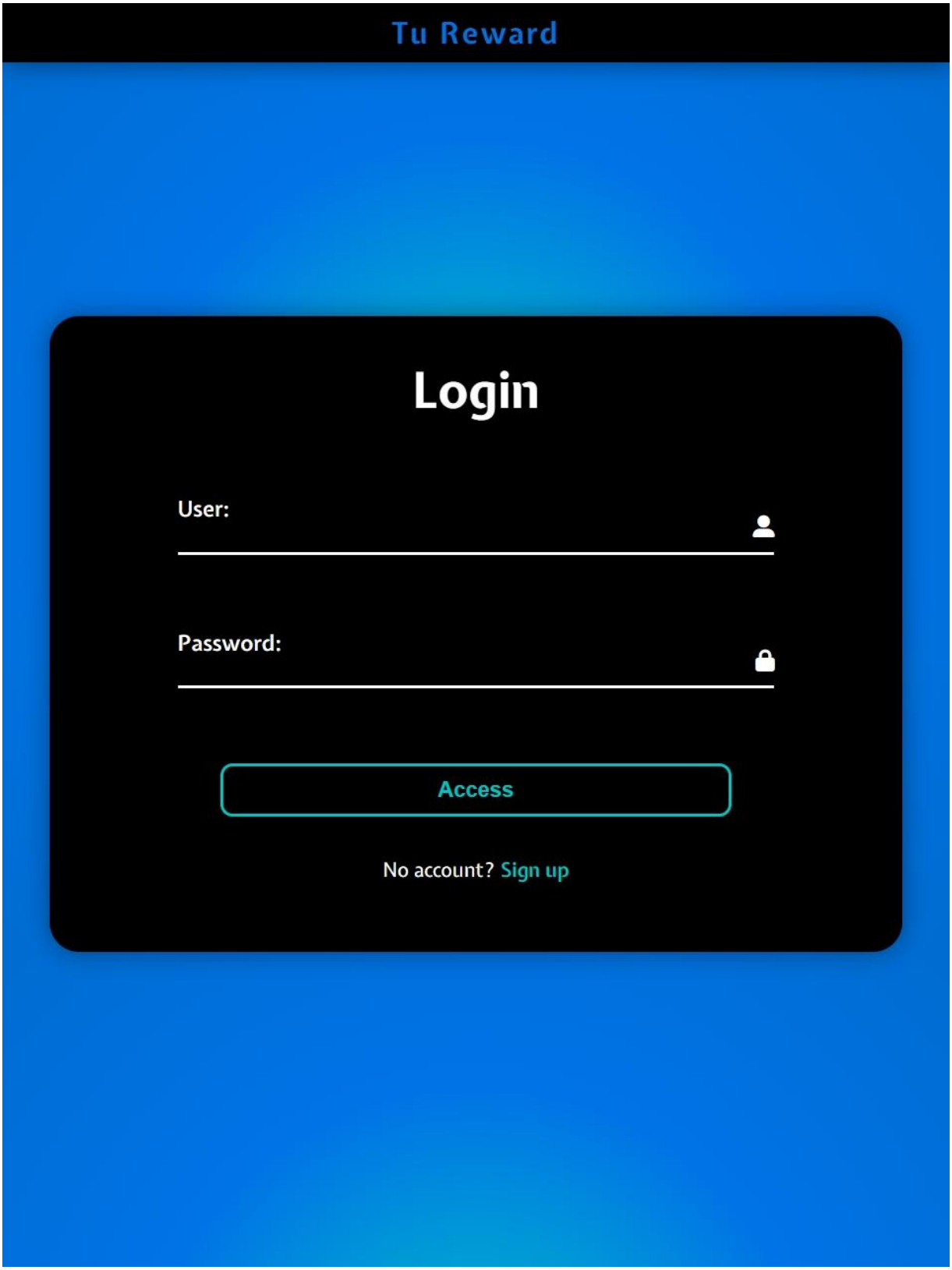
Scan Panel (pc view)



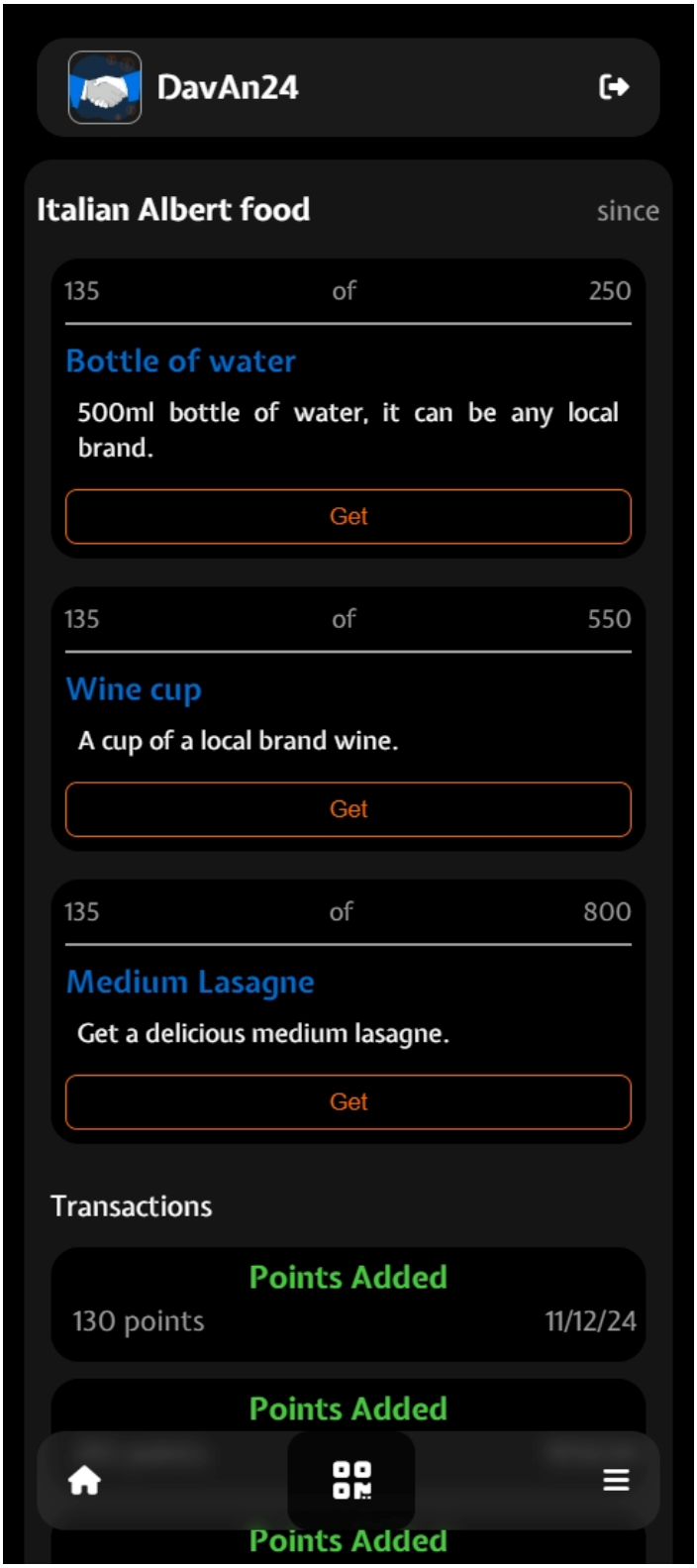
public main page (pc view)

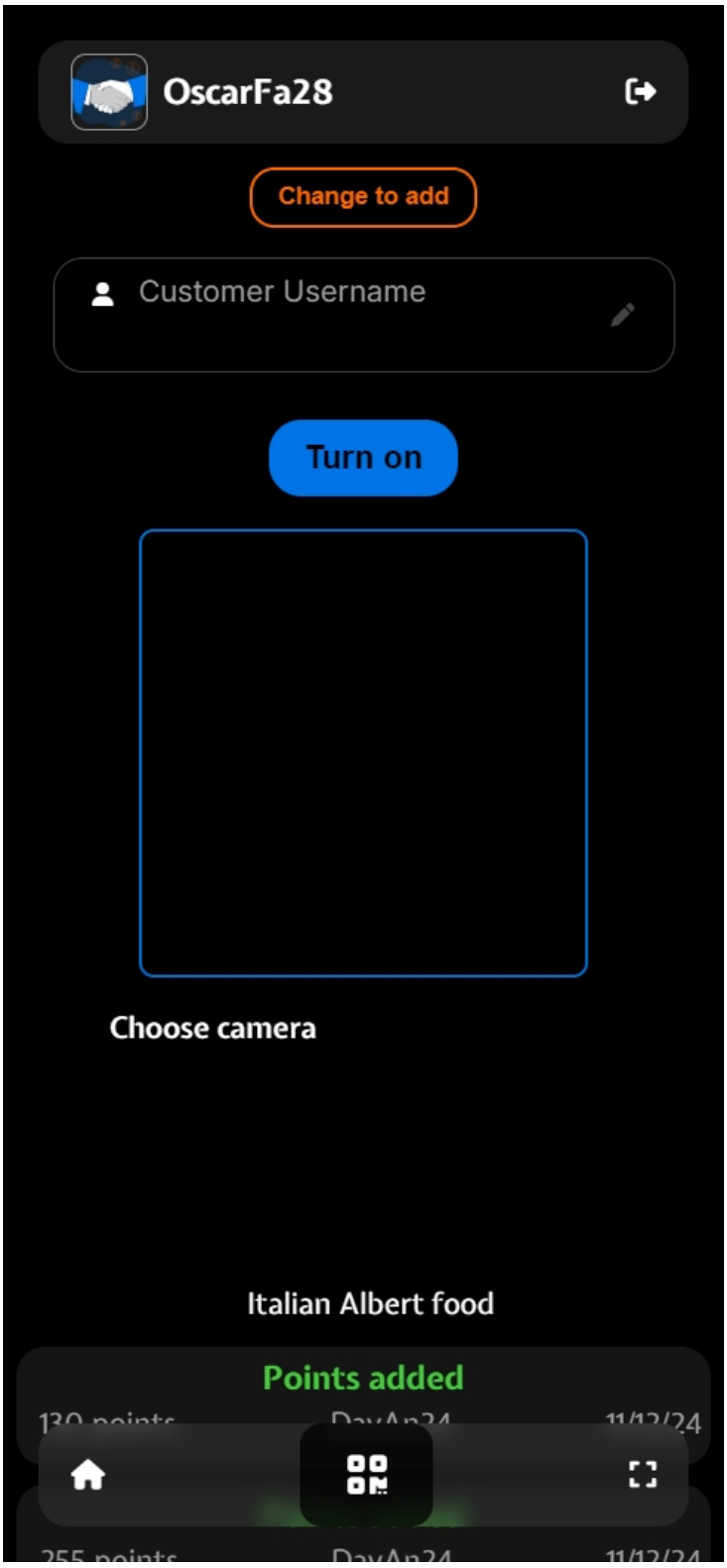


Login page (tablet view)

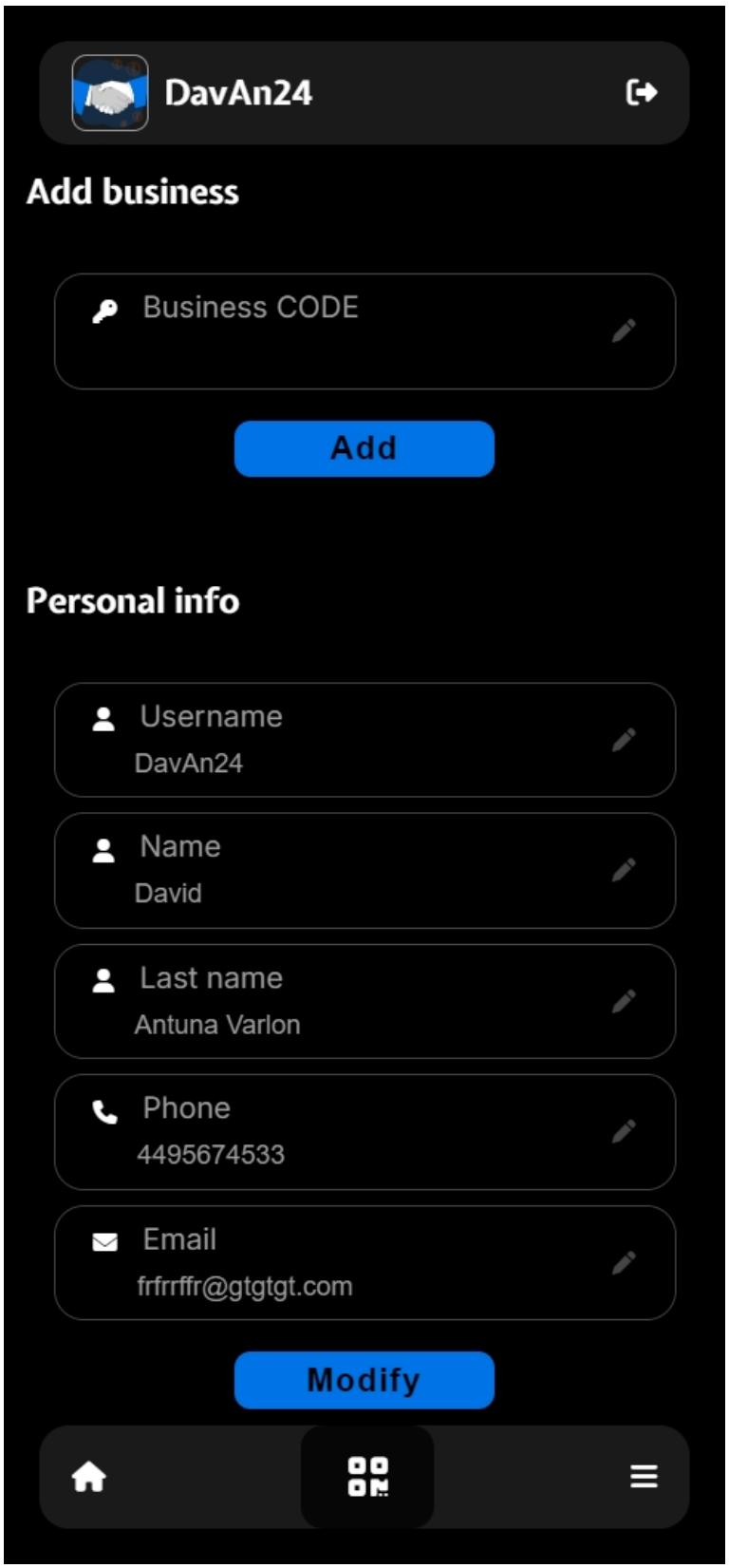

Tu Reward
Login
User:
Password:
Access
No account? Sign up

User panel and scan panel (mobile view)





Information panel and product qr code (mobile view)



5. Configuration

Default Configurations

- Default user types: normal_user and administrator.
- Predefined endpoints for managing rewards, users, and transactions.

---

6. Features

Key Functionalities

- Custom User Management: Separate user types with a custom Django user model.
- QR Code Integration: Simplified point collection and redemption.
- Admin Panel: Tools for managing rewards and transactions.
- User Panel: Interface for customers to track points and link accounts to businesses.

Unique Functionalities

- API endpoints for business-specific actions, such as linking users to businesses and validating QR codes.

7. Code Structure

Overview

The project structure follows Django's standard architecture:

- tureward/ (Main project directory)
    - settings.py: Configuration files for the project.
    - urls.py: URL routing.
- apps/
    - adminPanel/: Manages business administrators and reward operations.
    - userPanel/: Handles user functionalities such as linking businesses, viewing points, and rewards.
    - publicPanel/: Provides public pages like login, create account, and information.

Main Elements

Models

CustomUser

A custom user model extending Django's AbstractUser:

- Fields:
    - phone_number: Optional contact number.
    - code: Unique code for each user.
    - account_type: Differentiates between normal_user and administrator.
    - qrCode: Automatically generated QR code image for the user.
- Methods:
    - save: Automatically hashes the password and generates the QR code upon saving.
- Relationships:
    - Many-to-many relationships with groups and permissions.

Business

Represents a business entity:

- Fields:
    - name: Business name.
    - code: Unique identifier.
    - qrCode: QR code associated with the business.
- Methods:
    - save: Generates a QR code if not already present.

UserBusinessRelation

Tracks the relationship between users and businesses:

- Fields:
    - user: ForeignKey to CustomUser.
    - business: ForeignKey to Business.
    - points: Tracks the user's points for a specific business.
- Meta:
    - Unique constraint on user and business.
    - Indexed fields for optimized queries.

Reward

Defines rewards offered by businesses:

- Fields:
    - title: Name of the reward.
    - code: Unique identifier.
    - points_required: Points needed to redeem the reward.
    - description: Details of the reward.
    - business: ForeignKey to the associated business.

    - qrCode: QR code for the reward.

Transaction

Logs all point-related operations:

- Fields:
    - title: Description of the transaction.
    - amount: Points added or deducted.
    - date: Timestamp of the transaction.
    - user: ForeignKey to the user involved.
    - business: ForeignKey to the business involved.

Views

Admin Views

- Allow administrators to:
    - Create, update, and delete rewards.
    - Manage point redemptions by scanning user QR codes.

User Views

- Allow users to:
    - Link to multiple businesses.
    - Track points by business.
    - View available rewards and their required points.
    - Display their QR code for point redemption.

Public Views

- Provide public pages:
    - Homepage with information.
    - Login and account creation pages.

Controllers

API Endpoints

- Admin Functionality:
    - Add/delete rewards.
    - Redeem points.
- User Functionality:
    - Link to businesses.
    - Retrieve reward and point information.
- QR Code Validation:
    - Validate and process QR codes for users and rewards.

8. Deployment

Deployment Steps

1. Prepare the environment:
    - Install required software (e.g., Gunicorn, Nginx).
    - Set up a MySQL database in the production environment.
2. Configure settings:
    - Update settings.py for production (e.g., DEBUG = False, ALLOWED_HOSTS).
    - Add database credentials and other sensitive configurations to environment variables and change database setting to use mysql instead of sqlite.
3. Deploy using Gunicorn and Nginx:
4. gunicorn --bind 0.0.0.0:8000 tureward.wsgi:application
    - Configure Nginx to proxy requests to Gunicorn.
5. Apply migrations and create a superuser:
6. python manage.py migrate
7. python manage.py createsuperuser