GitHub Username: Michael Huang

TinyPOS

Description

TinyPOS is a point of sale client for restaurants. Its goal is to provide a simple and reliable solution to the restaurant users. With this app, you can easily pull out online orders. It eliminates all traditional complexities, providing you the easy interface and the simple hardware installation. It's a flexible solution for saving your money and time.

Intended User

This app is designed for Restaurant, Deli Store, Food Court, etc.

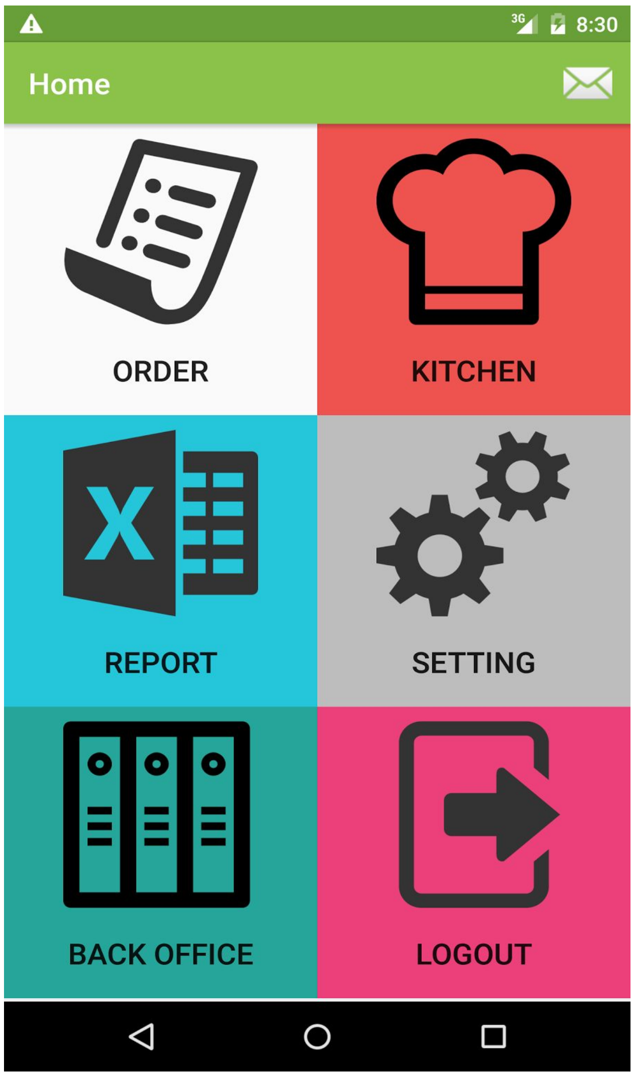
Features

- Online order handling.
- Fast address lookup.
- Delivery time estimation.
- Simplified user interface.

* Inherit basic functions of POS: taking and monitoring order, receiving payment, managing tables, managing customers, report, etc.

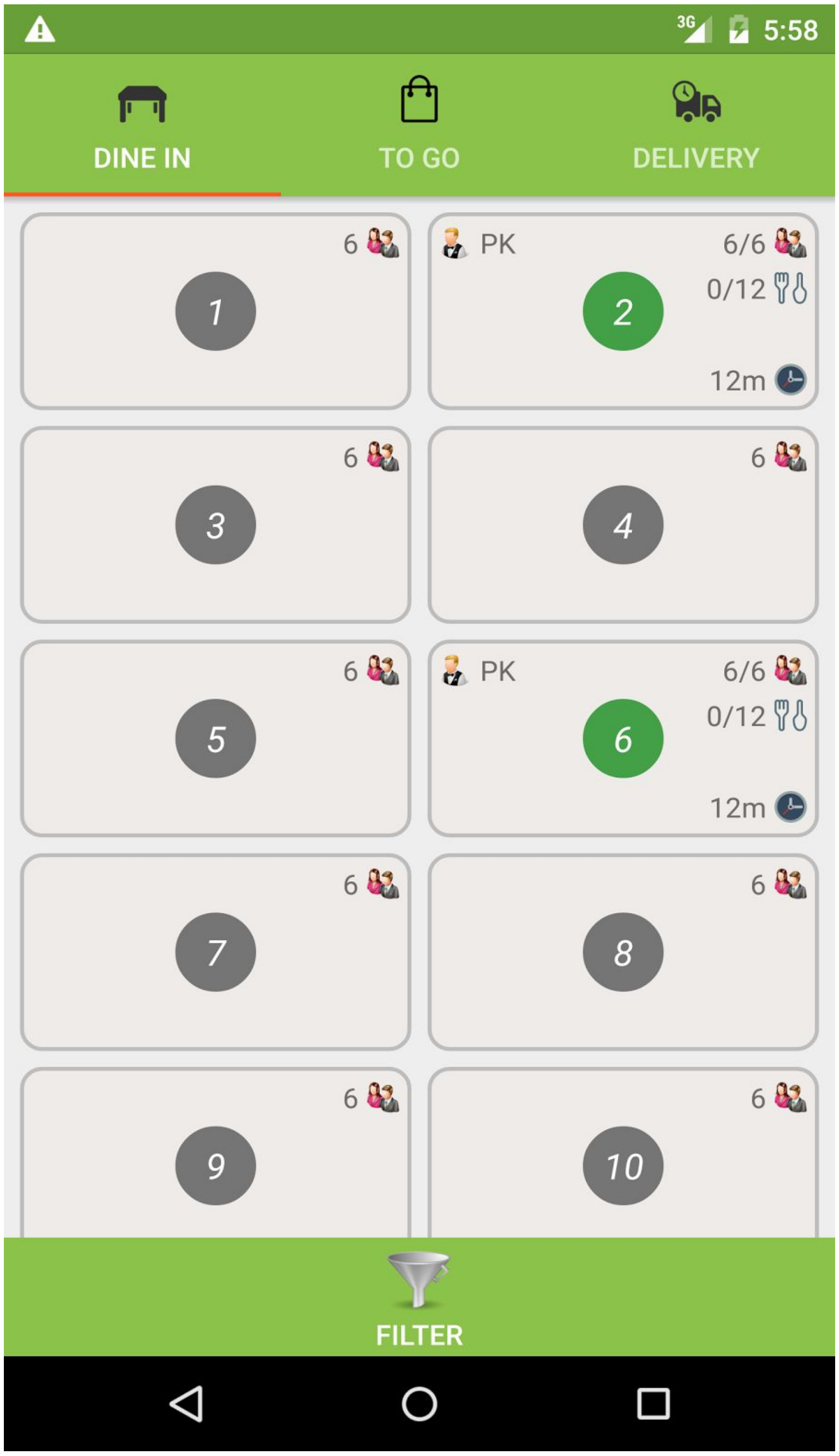
User Interface Mocks

Screen 1



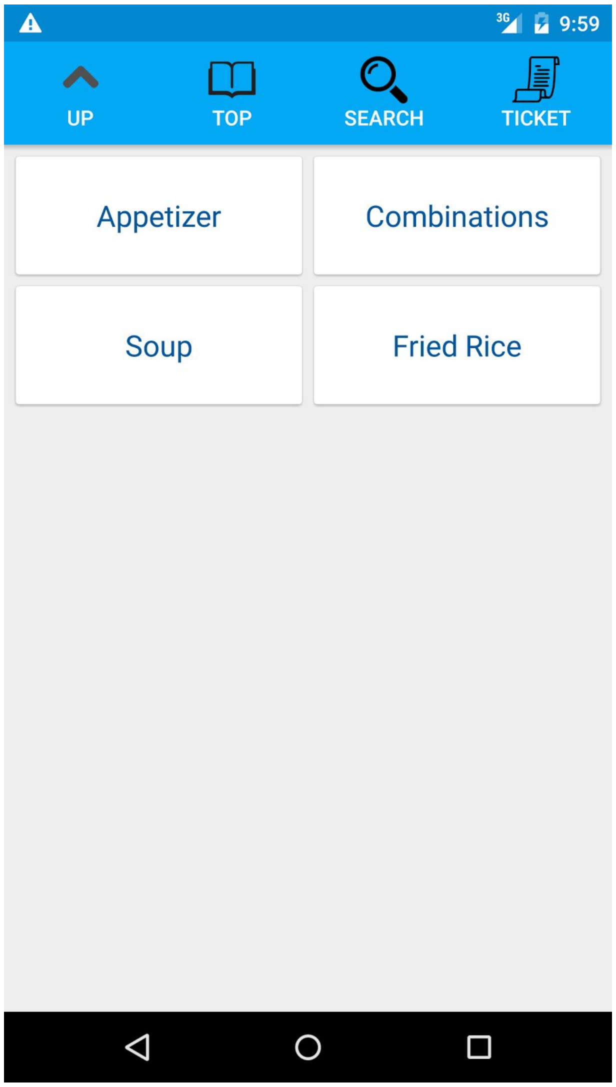
Home Menu

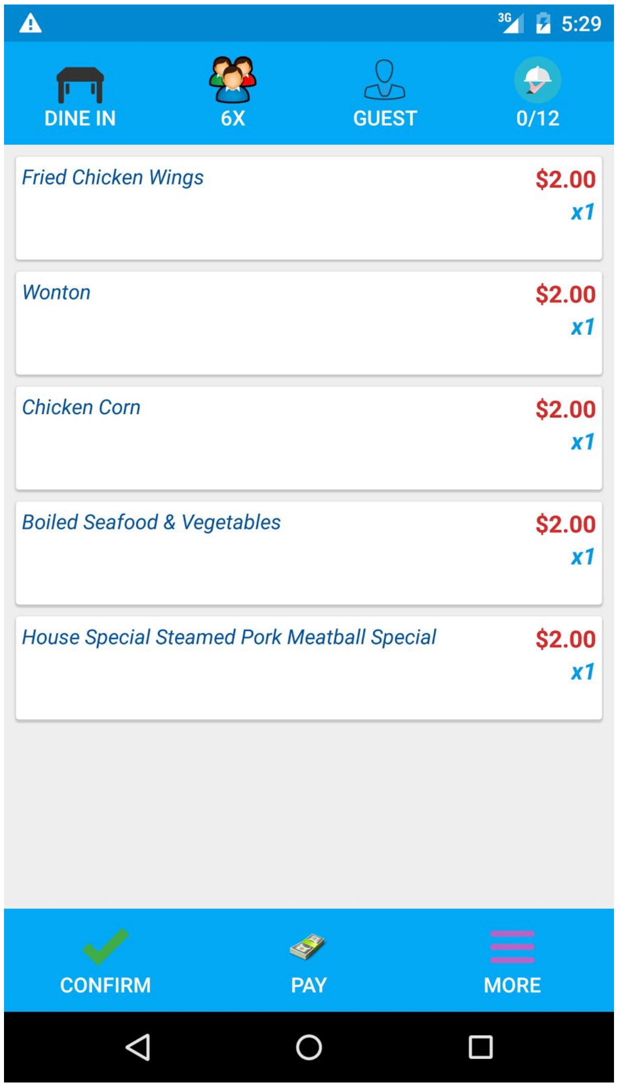
Screen 2



Order Home - order monitor, including Dine In, To Go, and Delivery

Screen 3

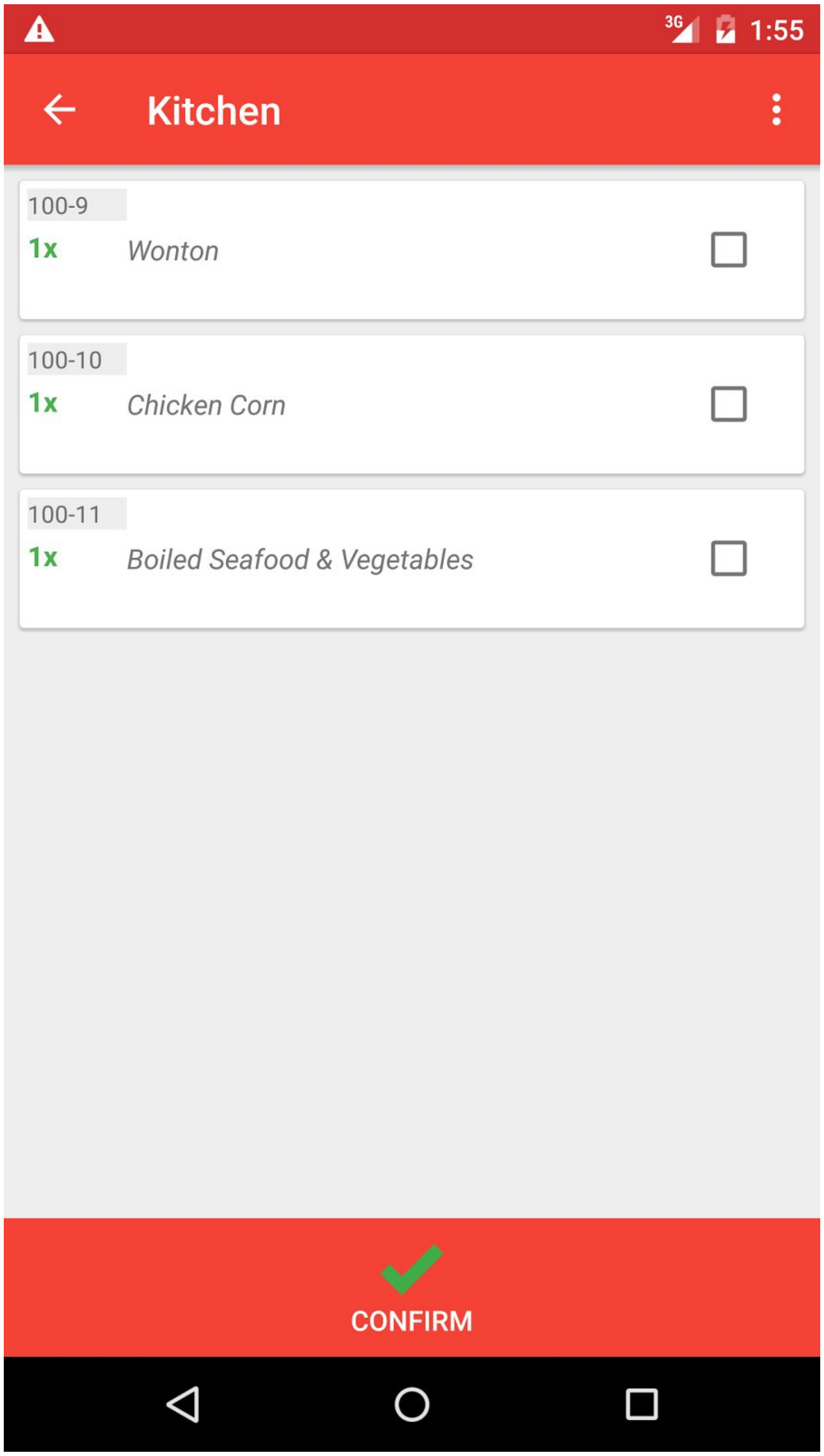




Fried Chicken Wings	$2.00	*x1*
Wonton	$2.00	*x1*
Chicken Corn	$2.00	*x1*
Boiled Seafood & Vegetables	$2.00	*x1*
House Special Steamed Pork Meatball Special	$2.00	*x1*

Order Ticket - a single activity, which is swipeable between food menu and ticket detail, is designed to take orders with fewer steps.

Screen 4



100-9
1x Wonton

100-10
1x Chicken Corn

100-11
1x Boiled Seafood & Vegetables

Kitchen - displays a list of food items which are in progress.

Key Considerations

How will your app handle data persistence?

The data is stored on the server, but the app also has its own copy of food menu and progressing orders. The data saved on the local database can be accessed through Content Provider.

Describe any corner cases in the UX.

- If the device disconnects from the server, it's still able to take orders. Once the device reconnects to the server, the built-in service will synchronize any changes.
- if an order is currently being modified by multiple devices, only one device can make the change to that order successfully.

Describe any libraries you'll be using and share your reasoning for including them.

GSON - Extract data from Server.
Volley - Communicate with the server or any HTTP access.
Butter Knife - Bind views and reduce code.
Picasso - Load and Cache Images.

Describe how you will implement Google Play Services.

Maps, Places - Provide hints when creating a new address and get routes for delivery orders.
Analytics- Collect user data for improving user experience in the future.

Next Steps: Required Tasks

* Since this project should focus on Android app, I am not going to include the structure of server-side implementation here, but I will attach the source.

Task 1: Project Setup

- Update SDK.
- Configure libraries.
- Import icon package.
- Create signing task.

Task 2: Implement UI for Each Activity and Fragment

- Build UI for LoginActivity
- Build UI for HomeActivity
- Build UI for OrderMainActivity
    - DineInFragment
    - ToGoFragment
    - DeliveryFragment
- Build UI for OrderActivity
    - OrderMenuFragment
    - OrderTicketFragment
- Build UI for KitchenActivity
- Build UI for ReportActivity
- Build UI for SettingActivity
- Build UI for CustomerActivity

Task 3: Create Local Data Store

- Design table layout.
- Create models.
- Implement Content Provider.
- Create a service to synchronize data.

Task 4: Implement Business Logic

- Create controllers to handle order, payment, report, customer, etc.
- Implement helper libraries.
- Implement server API call.
- Implement Address-Auto-Correction and Delivery-Routing with Google Map.
- Implement User-Tracker with Google Analytics.

Task 5: Bring up UI

- Implement event listeners.
- Configure activity stack.
- Connect list views or grid views to local data store.

Task 6: Handle Errors

- Add exception handlers as needed.
- Add unit tests.
- Apply stress testing.