eCoffee: Remote Ordering Development

STUDENTS: Qihuang Chen, Yilin Wang, Colleen Park, Celeste Xulei Cheng

FACULTY ADVISORS: Prof. Ken Eguro, Daniel King

INDUSTRY ADVISORS: Mr. Michael Nassirian, Mr. Marko Munda

SPONSOR COMPANY: ARVR Academy

Motivation/Requirements

- In today's market, retailers lack customer exposure and have difficulties getting rid of inventory, and customers lack access to sale items.

- Our mission is to design and implement a web interface for retailers to manually send notifications to customers, and an App for customers to make orders.

- Requirements: The main work is divided into three parts: a web interface for retailer, an Android App for customer, and back-end services.

Back-end Architecture

- Build REST APIs with Spring framework and 3rd party toolkits, which help manage communication between frontend and backend.

- Use MongoDB Atlas as a cloud database service for data persistence.

- Utilize Square API for payment and order management.

- Deploy the backend services to an AWS EC2 host.

Front-end

- The figure on the right is the UI design of the app. It shows basic user flow like how a typical user would use the app to place orders.

- The figure on the top left is the current homepage for the eCoffee app running on an Android emulator. It is the front-end for all customers who download our app to use. The process of demo would be better showed by video.

- The figure on the top right is the retailer control panel written in React. This web-page is designed for retailer to add sales items, send notifications, check ongoing order status and browse order history.

Deliverables

- A functional remote order application with frontend and backend components.

- Online Ordering: features including 1) choosing retailers by geo-coordinates, 2) displaying menu items, 3) making orders, and 4) paying for orders with credit cards. In addition, customers’ purchase histories and habits of coffee are recorded in the database.

- Daily Recommendation: customers will receive daily, tailored notifications about featured items that they may be interested in, based on their previous order histories.

- Retailer Recommendations: Retailers are able to create featured items for promotion. Customers who subscribed to the store can receive notifications about them.

Services Implementation

- Account Service: Firebase Authentication provides backend services, SDKs and UI libraries to identify a user in the app.

- Square provides APIs to create and track orders, manage a product catalog, accept payments.

- Order and Payment Services: Users could complete orders showing in retailer’s Square port.

- Store service: Users could scan menu of any store in our app.

- Notification service: Firebase Cloud Messaging (FCM) allows retailers to send notifications to users.

- Discovery service: Find nearby stores realized by Android LocationManager and MongoDB Geospatial queries.

- Onboarding Service: OAuth to realize retailer authentication to access their store information stored in their Square account.

Conclusion, Future Work, and References

- Most functions are available. Another function of Notifications is designed for the first-used customers when people pass by a store, which is registered in the eCoffee application.

- Further improvements of frontend design and implementation.

- Consider more edge tests in software development process and Quality Assurance test preventing defects in production. Realize network security for data communications.

- Optimize location accuracy, signal intensity, battery efficiency and update speed for discovery service using tools like BlueTooth 5.0 Long Range functionality.

Reference: Spring Rest API Tutorial database MongoDB Atlas