CS3216 Group 7 - BigSpoon

Members: Shubham Goyal, Wang Gaoxiang, Yang Zhixing, Yos Riady

1. Description of the application you plan to develop.

For the CS3216 final project, we plan to build BigSpoon, a personal waiter application for diners and an order management system for restaurant staff.

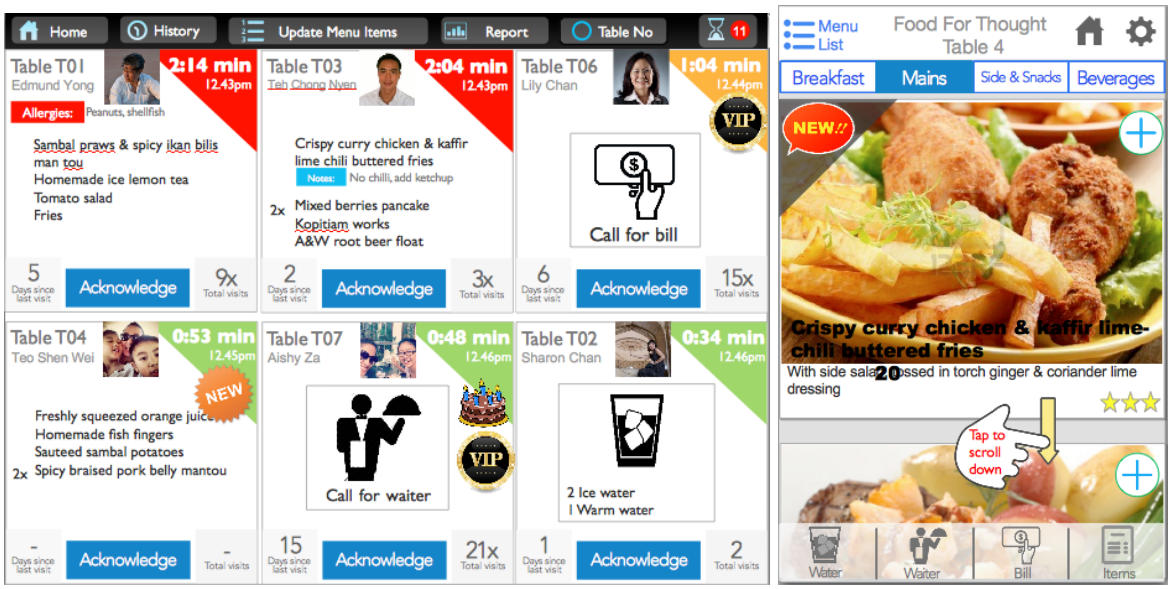

fig 1. initial UI mockups for manager and diner views

The problems we are trying to solve:

1. From the restaurant's point of view:
    a. In a restaurant, waiters often needs to attend to the many tables at a time and often has to spend considerable time waiting for diners to decide what they want to eat.
    b. The statistics generated by current Point of Sale (POS) system are usually just based on per table or per bill basis, which is not very useful for restaurant owners to learn about demographics and the diners' individual preferences.
2. For the diners:
    a. Diners often get frustrated because it is difficult to get the waiter's attention, especially during peak hours or in popular and busy restaurants.
    b. The conventional physical menu is often not enough to help the diners judge

what the menu item name actually corresponds to. Sometimes, the item names do not even mean anything in our everyday language. A digital menu can contain descriptions, images and reviews for every item which gives diners more information to work with.

c. Diners' preferences are not recorded. For example, this might be anything from allergies to a particular ingredient to a particular style in which the diner wants his food cooked. So, whenever the diner goes to a restaurant, he/she has to tell the waiter his/her preferences again.

The solution:

BigSpoon aims to make dining out a pleasant experience. It will feature interactive menus, an automatically updating shopping cart, easily accessible ratings and reviews as well as intuitive gestures to call for water or the bill. We believe that this will help diners save time and increase their overall satisfaction.

We want to give restaurant owners an easy way to track orders and allow them to see which tables are waiting for which food items and better manage their staff. And of course, better customer experience for diners translates to returning customers.

What can you do with BigSpoon?

1. Diners can use BigSpoon to view interactive menus, order dishes, leave reviews, call for waiter, ask for water, call for bill, and view their dining history.
2. Restaurant staff can create/edit/delete menu items.
3. Restaurant staff will be able to see orders in real time as they come in and then take appropriate decisions to process them.
4. Restaurant staff can view a diner's order history in that restaurant and also his/her preferences. They could also add personal notes to this diner to enable them to serve the diner better on his/her next visit.
5. BigSpoon can produce more meaningful and useful analytics which not only provide the restaurant owner a broad overview of who visits the restaurant at what time but also drill down to the level of each user.

2. Are there any existing applications out there that are similar? What makes your application special? Why did you choose your application?

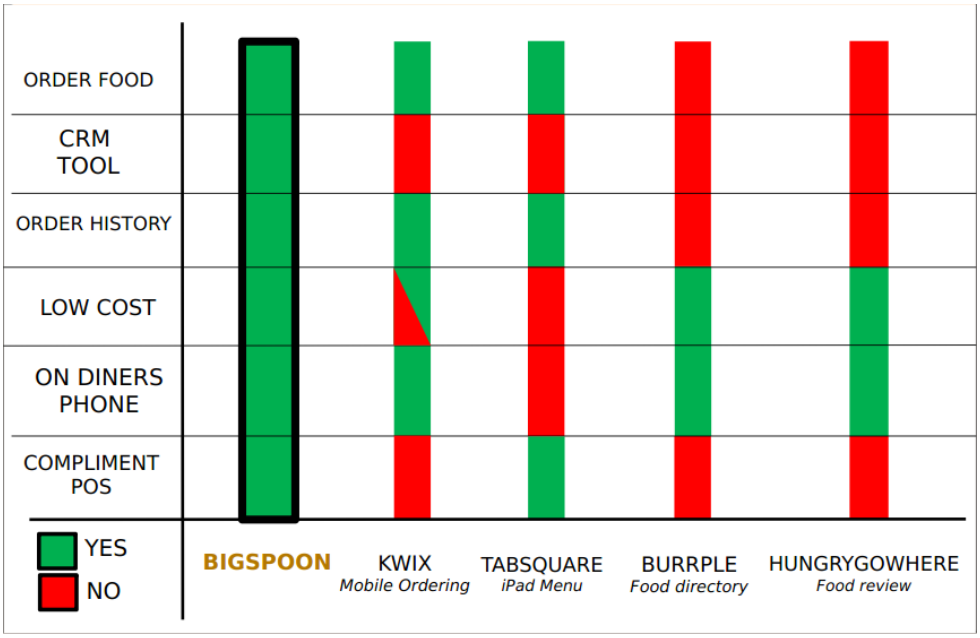

fig 2. competitors analysis

As shown in the analysis our key competitor is Kwix (kwix.com.sg).

For Kwix, it lacks of traction. Kwix started operations almost a year ago, whereas it only have 1 client restaurant listed and a few diner users. Kwix is competing in the POS market, switching costs are high. They focused on speed, not on customer satisfaction. After we tried out the diner application from kwix, we notice that the response time is slow and there are bugs in the order flow as well.

On the other hand, our app plans to enable a more long-term and mutually beneficial interaction between the diner and the restaurant. The long term interaction is enabled by the statistics that the restaurant owner can use, the reviews left by the diners, etc..

We also record the dining history and the personal preference of a diner which the restaurant manager can use to provide the diner with an even better experience. Also, the staff is able to add notes about the particular diner into the system for later reference. We believe all this adds more value for the diner and restaurant owner alike.

3. Project schedule: milestones and timeline, including an implementation and deployment plan.

We will divide our development into four iterations. The first iteration spans one week, and the following iterations spans 2 weeks each.

Date	Milestone	Description
2 October Wednesday	Proposal	- Submit this document.
6 October Sunday	Finish Iteration 1	● Diner-Frontend: Finish UI mockup in iPhone app (without the login page) ● Staff-Frontend: UI mockup ● Backend: Basic menu item CRUD admin, place order API endpoints ● Update progress document
20 October Sunday	Finish Iteration 2	● Diner-Frontend: Integrate with place order Backend API ● Staff-Frontend: Integrate with place order Backend API and socket realtime display ● Backend: Make sure place order API endpoints and socket realtime API is robust and bug free ● Schedule meeting with restaurant owners to let them try out the prototype and gather feedback, if the feedback is good, make a release to the restaurant owners ● Update progress document
22 October Tuesday	Progress Report 1	Test the prototype and bug fix
3 November Sunday	Finish Iteration 3	● Diner-Frontend: Integrate with other Backend API, UI improvement ● Staff-Frontend: Integrate with other Backend API, UI improvement, client side request queue system ● Backend: Finish other Backend API, server side request queue system ● Schedule meeting with restaurants owners again, get user survey feedback and discuss possible change in the system ● Update progress document
4 November Monday	Progress Report 2	Gather feedback from CS3216 staff group, test new MVP and bug fix

	(Oral)	
17 November Sunday	Finish Iteration 4 (final)	• Diner-Frontend: UI improvement, change according to user feedback, other features[1] • Staff-Frontend: UI improvement, change according to user feedback, other features • Backend: Test case for endpoint API and socket API, change according to user feedback, other features • Update final report document
18 November Monday	Poster Session	Gather feedback from judges and guests, make changes accordingly, update final report document
22 November Friday	Final Report Due	Final check on system and submit final report

4. Individual contribution and roles. Contributions and/or support from external partners, if any.

Wang GaoXiang:

- network infrastructure setup
- backend server (Django with socket) setup,
- server endpoint API
- merchant admin system

Yos Riady:

- backend server endpoint API
- merchant admin system
- merchant management system (html5 web application with socket)
- Staff view front-end/UI

Yang ZhiXing:

- Client version (front-end, iPhone native app)
    - Outlet list, Photo menu, List menu, Review/Place order page, Login page

Shubham Goyal:

- Client version (front-end, iPhone native app)
    - Setting page, Order history page, Profile page

[1] other features are features that are not must-do for the final project but if we maintain a good speed and manage to get some free time, we will definitely do them.

BigSpoon:

- review code and project quality
- provide project requirement and direction
- schedule meeting with merchants for user feedback

5. Long-term plan and business model (if applicable).

By November, BigSpoon is planning to roll out an alpha version of the application to 10 cafe shops and maybe 1 or 2 restaurants to test out the application.

BigSpoon has chosen to market to cafes and restaurants early on. This is because western food usually involves ordering individual dishes which fits in perfectly with our application since the use case that we envisage that many people, even though they are in a group, will order individually. But Chinese/Indian cuisine often involve common dishes and so usually one person orders for the entire group. To accommodate the latter case we would have to make significant changes to our plan for statistics generation and order handling.

Furthermore, BigSpoon is deliberately choosing cafes for alpha testing to build up a hype about itself among cafe hoppers in singapore and increase our chances of being written about in publications or blogs.

While BigSpoon has yet to finalize their business model, the co-founders have two ideas:

- BigSpoon can charge a flat subscription fee to all merchants using the application, (for example, something like $500/month). This is a recurring fee.
- The alternative is to have a commission scheme where we are paid between 0.1-0.5% of the total sales that happen through the application.

In addition, there might be an initial setup fee for merchants such as providing them access to the application, helping them add menu items initially, build their profile if they need help and so on.

6. Marketing and strategies to be employed.

The marketing strategy will be divided into two parts, one for restaurant customers (diner) and one for merchants (restaurant owners).

For diners:

We will be using Facebook as primary authentication for diners, therefore, facebook pages and other Facebook virtual marketing strategy will be used to attract friends of users to use the BigSpoon and grow our user base.

Besides that, as BigSpoon will be used in our partners' restaurant, our partners will be helping on promoting the application to the diners as well. For example, we will run promotions through our partner merchants. For example, diners who spend $15 and above using BigSpoon will receive a free dessert.

For restaurant owners:

It will be business to business (B2B) sale, this part will be handled by BigSpoon co-founders, so far they already have at least 10 restaurant owners are willing to deploy the system once it's completed.

7. High-level design

Our app can be divided to three main parts:

Diner App ● native iOS app ● consume Diner App API	Management App ● HTML5 responsive real-time web app ● consume Management App API	Backend ● Restaruant Content Management System (CMS) ● Diner App API ● Management App API

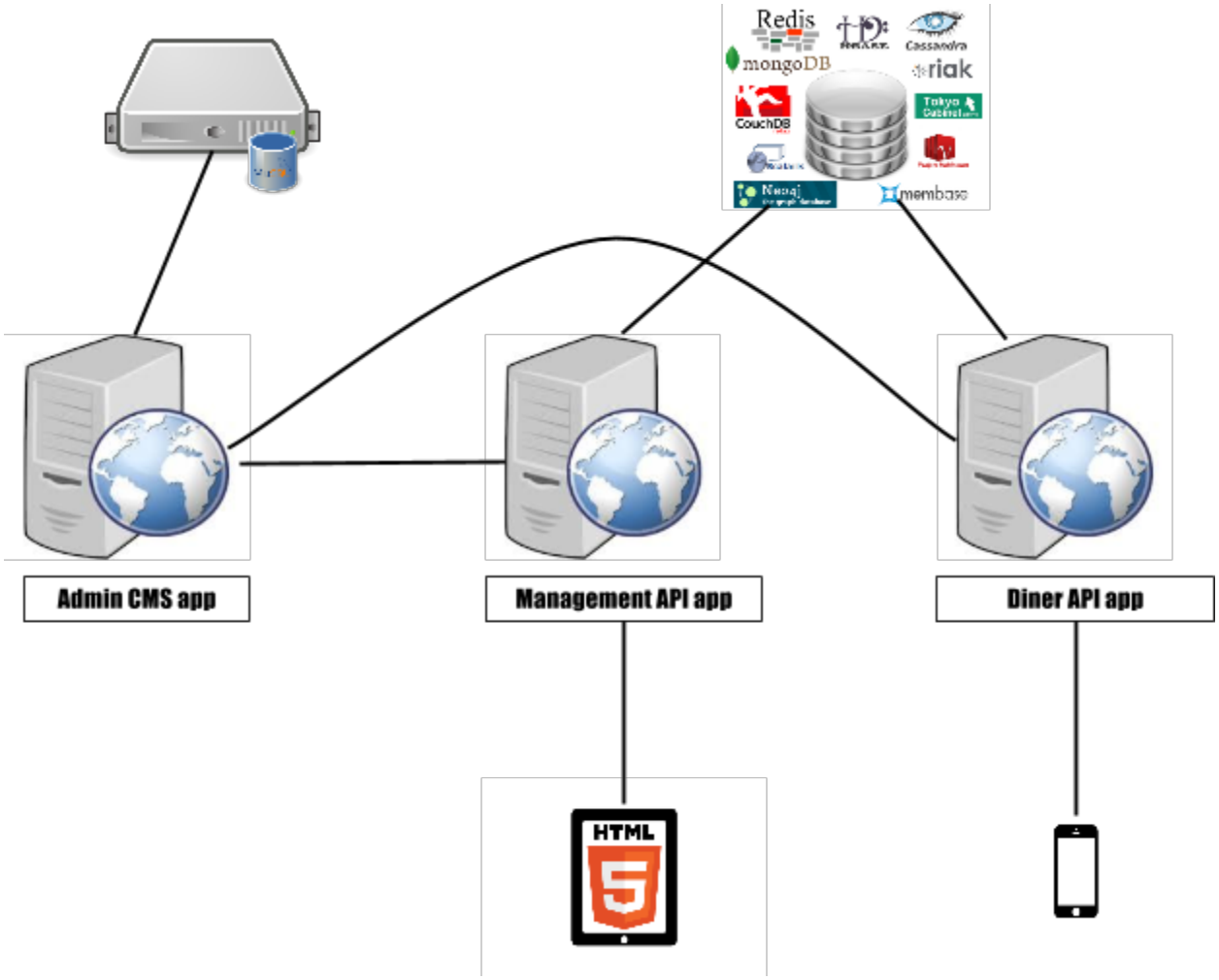

fig 3. tentative system architecture graph

BackEnd -admin app (django), management app API (to be confirmed), diner app API (to be confirmed)

Model:
Admin App - Restaurant, Outlet, Menu, Dish
Management App - Outlet, Manager, Order, Request, Table, Meal, Note
Diner App - Diner, Outlet, Review, Order, Request, Rating, Dish, Menu, Table, Meal

Relation:
Restaurant - outlets
Table - outlet, meals
Outlet - restaurant, reviews, menus, managers, tables, meals
Menu - outlet, dishes
Dish - menu, order, ratings
Diner - notes, reviews, ratings, meals
Manager - outlet, notes
Meal - outlet, diner, table, orders, requests
Order - diner, meal, dishes

Review - diner, outlet
Rating - diner, dish
Note - manager, diner
Request - diner, meal

Diner Frontend
View Controllers:
Diner, Outlet, Order, Note, Rating, Request, Dish, Menu

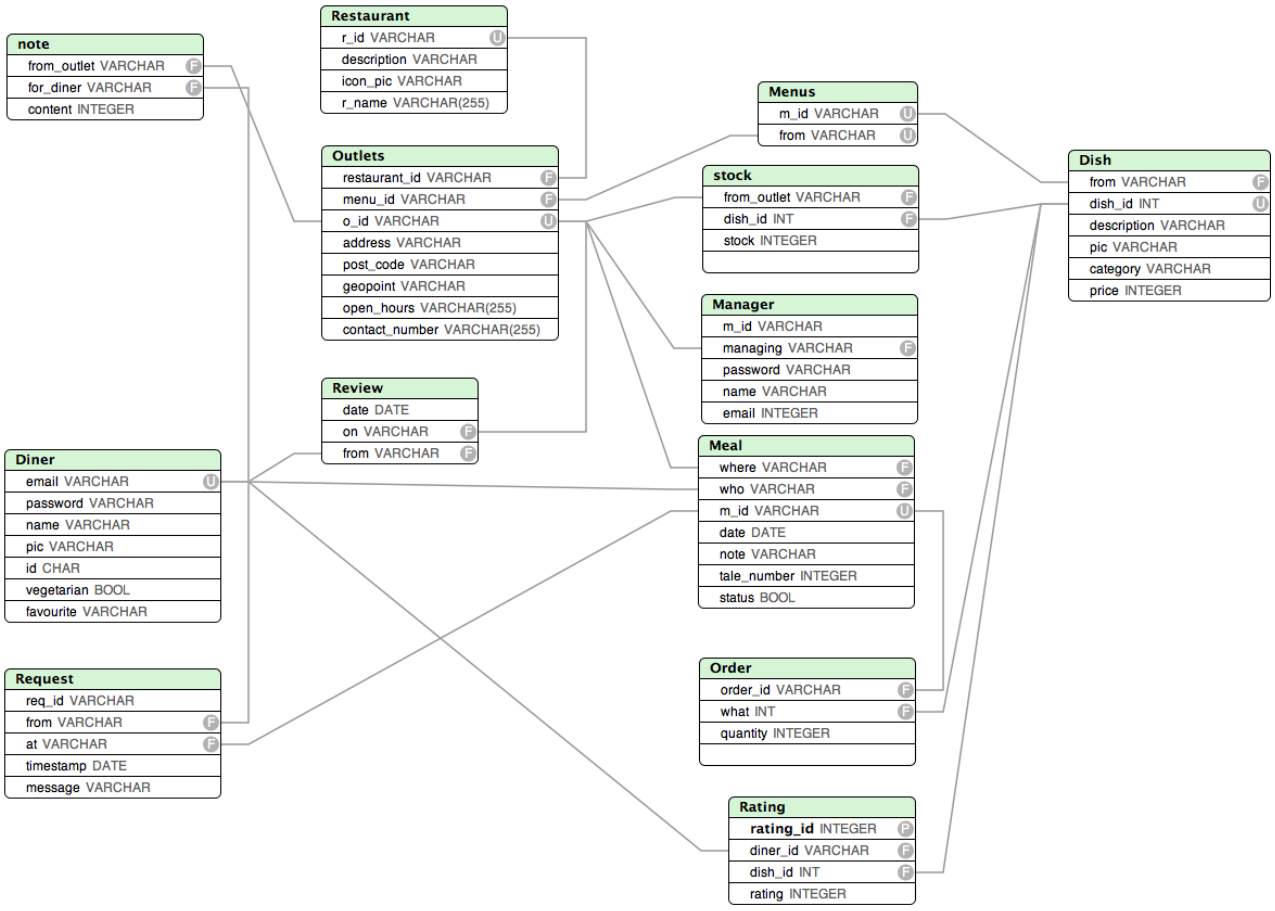

fig 4. tentative whole database schema