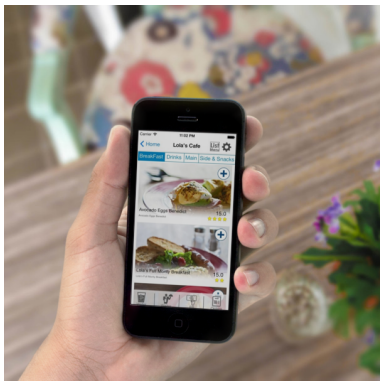
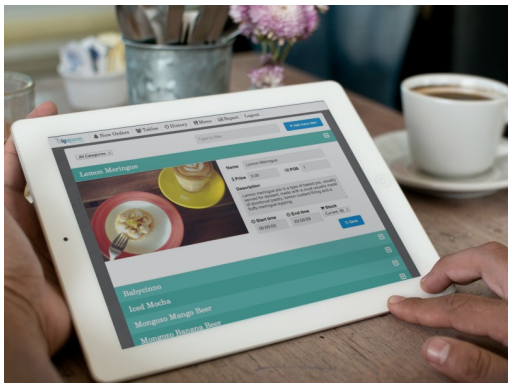




CS3216 Group 7 Final Report

Wang Gaoxiang, Yang Zhixing, Yos Riady, Shubham Goyal, Chen Liang

1. Description of the application you have developed

Our team's final project is BigSpoon, a personal waiter application for diners and an order management system for restaurant staff.

Why BigSpoon?

According to Business Times, the restaurant dining experience has been a big problem in Singapore. Due to rising labour costs, there are fewer waiters in restaurants. As a result, diners need to wait for up to 20 minutes to order their meal or get their bill. This is a problem that many of us have faced at one point or another, and it is very frustrating. Dining in a restaurant should not be a long painful waiting process. It should be an enjoyable time for friends and family.

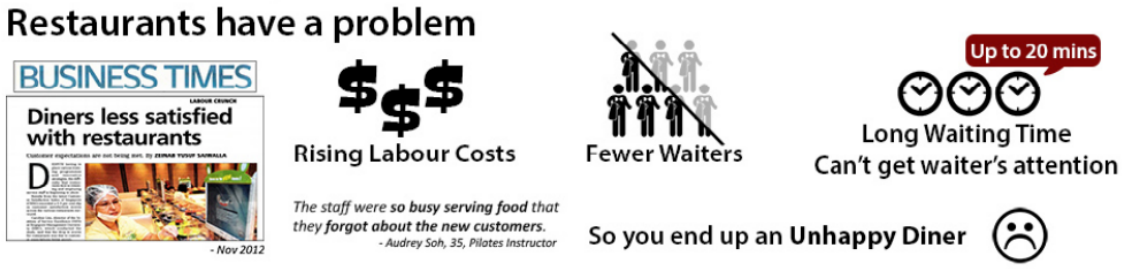


figure 1 Restaurants face manpower problems

Restaurant-Facing Problems

(a) Restaurants are facing rising labour costs and always seeking ways to reduce costs.	We notice that waiters spend a lot of time waiting for orders, because some diners often take a long time to decide on what they want to eat. This slows down service speed especially during peak hours, which makes other diners frustrated.
(b) Inefficient order taking.	Waiters need to write down diners' orders before he can key in to the Point of Sale (POS) system, it would save a lot of time if there is an easy to use system for both diners and waiters to manage orders.
(c) Slow waiter response to diners' signalling.	Very often in a restaurant, there are not enough waiters to go over to diners to take their requests/orders immediately, which delays service. results in a bad dining experience. They are mainly due to the following reasons:

	i. There are multiple tables that need the waiters at the same time. ii. Waiters need to juggle between taking orders/requests from tables and the serving of food. iii. The peak hours are often too short to be cost-effective for the restaurant owner to hire more waiters just to handle the peak traffic.
(d) Repeated obstacles that cause further delays in service after ordering.	For the same reasons in (c), there may also be a delay between between getting the order and actually passing it over to the kitchen because the waiter may need to attend to yet another table on the way back.
(e) No analytics available.	The statistics generated by current POS system are usually just based on per table or per bill basis, which does not help restaurant owners learn about demographics and the diners' individual preferences.

Diner-facing Problems

(a) Frustration in getting waiters' attention.	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) Unhelpful menu.	The conventional physical menu often has minimal pictures and the item names do not help the diners judge what the dish is like. They have to ask the waiters repeatedly. A digital menu can contain descriptions, images and reviews for every item which gives diners more information to work with.
(c) Allergies and special requests.	Diners' preferences such as allergies to a particular ingredient to specific directions are not recorded. So, whenever the diner goes to a restaurant, he/she has to tell the waiter his/her preferences again.

What is BigSpoon?

Currently, most restaurants go through 6 main steps in the entire service process.

1. Diners decide what dishes to order from the menu
2. Diner wants to make an order and waves around trying to get the attention of the waiter but the waiter is attending to someone else and cannot attend to the diner
3. Waiter takes Order(s) from Diner on paper, sometimes waiting for diner to decide

4. Enter all orders into POS System
5. POS prints orders for kitchen and Chef prepares food
6. Waiter serves food

BigSpoon is solving steps 1,2 and 3, so that the waiters can focus on just steps 4 and 6 to create better dining experiences and yet save manpower. With BigSpoon, small to medium-sized dining establishments can be empowered to hit above their weight using this full-fledged restaurant management system. Even as a small cafe owner, I can have an inventory management system, an order management system, and a CRM system all wrapped together in a neat package. In the future, we will also integrate with POS systems.

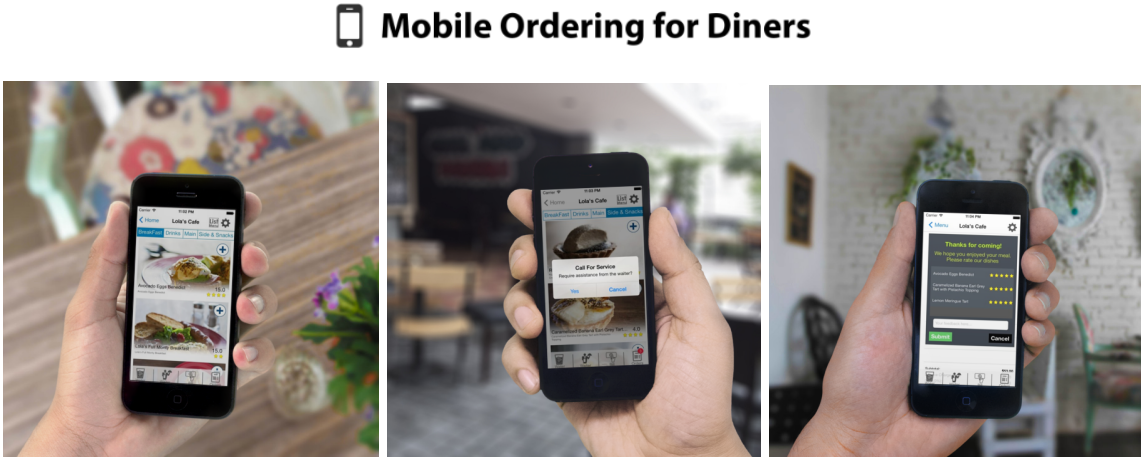
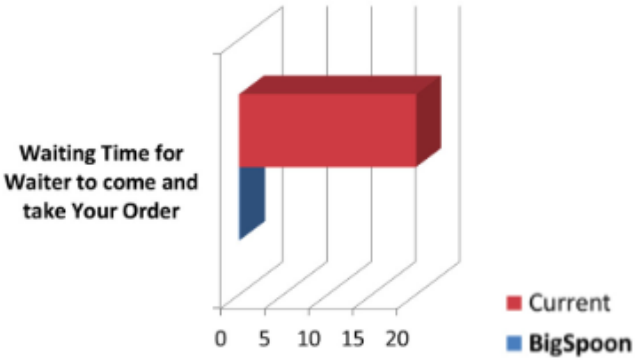


figure 2 diner application screenshot

The diner application is an mobile ordering system. It provides features such as a visual menu, instant feedback on order status, and user reviews/ratings.

Before entering the restaurant, you can choose to set up your profile (e.g. allergies, vegetarian and so on) which will be seen by restaurant waiter when you place your order. After selecting the restaurant, the visual menu from restaurant will be present to you and you can filter the dishes by categories. The menu is also smart enough to show you the availability of the dishes (e.g. out of stock, only available after 6pm).



Once you have sent out the order, restaurant staff will be notified in real time, immediately, without even a second of waiting for waiters. An improvement from the current waiting time needed before you can even make an order.

Once your order is entered into the POS system and sent to kitchen, you will be notified immediately as well. After the meal, instead of waving your hand many times to ask waiter to come, with just one click the waiter will bring your bill to you.

You can give feedback and ratings for the dishes if you have a great time at the restaurant. Your orders will be shown in your order history as well. Additionally, the diner's application also provides features such as asking water and ask waiters attention if you have some special request.

Restaurant Management Staff Portal

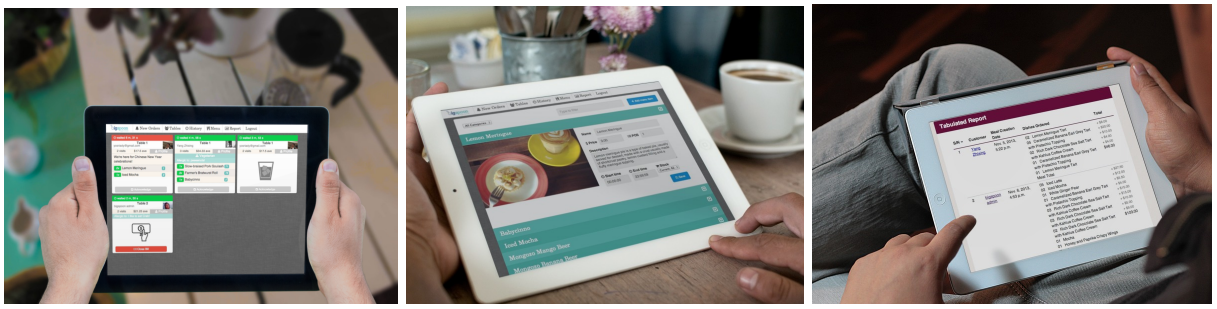
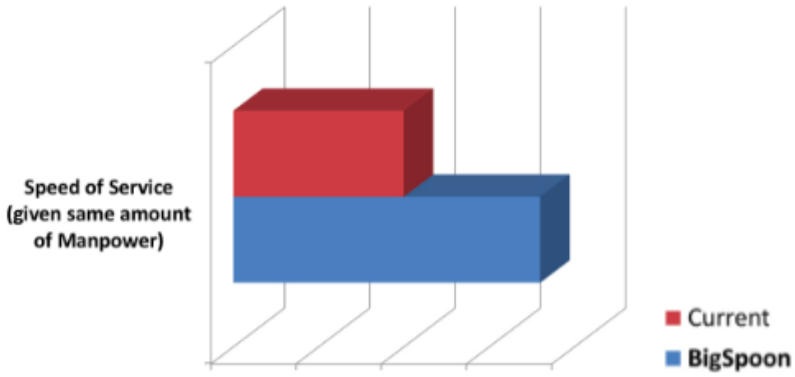


figure3 restaurant management application screenshot

The restaurant management system is an HTML5 web application which designed to support laptop, desktop as well as ipad devices. It enables restaurant staff to view orders in real time, manage and organize restaurant menu easily and view diner customized report for the restaurant.

Once diners have placed their orders, a request card followed by a sound notification pops up within the restaurant management application in real time. An important point to note is that the display is updated automatically, without needing the restaurant manager to do any refreshing, so that there is zero chance for the restaurant manager to miss anything.
When the restaurant manager wants to view other tabs in the app, there is an additional Facebook-style notification so that no orders are missed. There is also a Table view which shows a quick overview of all the restaurant's orders.

Thus, restaurant staff only need to go to diner's table when needed (e.g. bring bill, diner has special request), which will save manpower.



With the BigSpoon app, waiters can just focus on serving the dishes. In other words, the speed of service can be increased by a factor of 2 at the very least. With faster service speeds given the same amount of manpower, restaurant managers now have the flexibility to either reduce the manpower to reduce costs, or maintain the same amount of manpower to increase the quality of service.

On the Menu view, the restaurant owner can easily manage and update her restaurant's menu. Instead of printing new menu each time, she simply needs to complete a short new dish form, and it is be automatically pushed to diner's application.

Finally, restaurant staff can review useful analytics about top visitors and each diner's top favorite dishes easily. As a result, they are able to come out with a more effective strategy to attract existing diners and new diners.

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

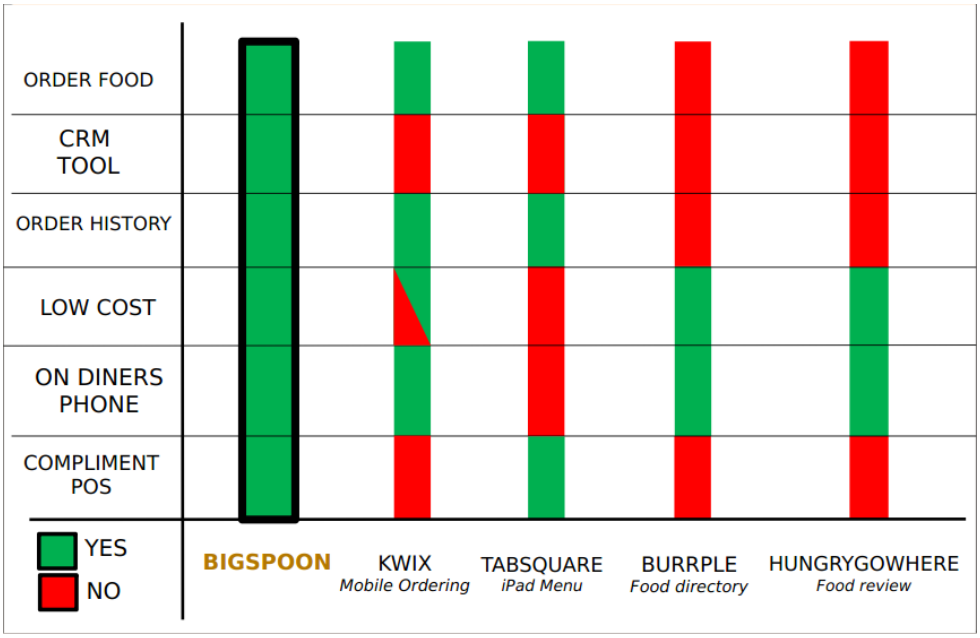


figure 4 competitor analysis

Feature	How it's special
Real time order system	Both staff and diners will receive real time feedback on the status of their orders as well.
Easy to use inventory system	Restaurant staff can update the menu effortlessly with just a few clicks.
Low cost setup	Unlike other solutions, restaurants do not need to purchase tablet devices to handle digital ordering.
Cross-platform restaurant management system	While the staff system is optmized for the iPad, it is also available for viewing on iPads, laptops, and home computers.
Advance Ordering	Because it is a mobile application, BigSpoon makes it possible to make orders before arriving.

3. Review of milestone and timeline for project

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

Date	Milestone	Description	Hits
2 Oct, Wed	Proposal	• Submit project proposal	✓
6 Oct, Sun	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 Oct, Sun	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	We missed this by a few days due to the mid term exams period.
22 Oct, Tue	Progress Report 1	Test the prototype and iterate based on user feedback	✓
3 Nov, Sun	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	During the meeting with restaurant owner, we discovered some new requirements, and it took us some time to finish.

4 Nov, Mon	Progress Report 2 (Oral)	Gather feedback from CS3216 staff group, test new MVP and iterate	✓
17 Nov, Sun	Finish Iteration 4 (final)	• Diner-Frontend: UI improvement, change according to user feedback, other features • 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 Nov, Mon	Poster Session	Gather feedback from judges and guests, iterate, and update final report document	✓
22 Nov, Fri	Final Report Due	Final check on system and submit final report	✓

4. Individual contribution and roles

Wang GaoXiang	● Network infrastructure setup (ec2) ● Backend server setup (Django/Websockets) ● Backend server endpoint API (Django-Rest-Framework) ● Restaurant order management system (Real time websocket connection) ● Endpoint API documentation
Yos Riady	● Backend server endpoint API (Django-Rest-Framework) ● Restaurant inventory system & order admin (Django) ● Restaurant management frontend view (UI/UX, HTML, CSS, JS) ● Marketing webpage design & code
Yang Zhixing	● Front-end, iPhone native app ○ Outlet list, Photo menu, List menu, Review/Place order page, Signup page, Login page, Request for water/waiter and bill function, Submit feedback and rating function ○ UI/UX Design on subtle feedback that the app provides to the user
Shubham Goyal	● Front-end, iPhone native app ○ Setting page, Order history page, Profile page
Chen Liang	● Backend merchant admin system ● Staff portal report - revenue table and revenue trend chart ● Design of the posters, and photoshop
BigSpoon (External Party)	● review code and project quality ● provide project requirement and direction ● schedule meeting with merchants for user feedback ● provide marketing strategy and support for poster session

5. Application design

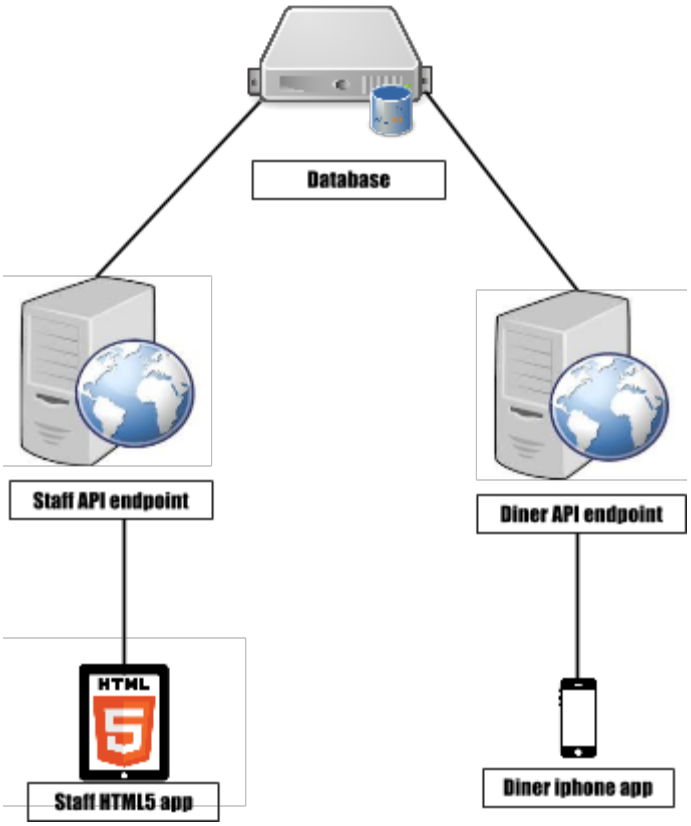


figure 5 BigSpoon's system architecture diagram

As shown in the above diagram, the BigSpoon system contains two main applications, a staff html5 web application and diner iphone application. Both are supported by a Python/Django powered REST API coupled with WebSockets for real time communication.

Restaurant-facing Web Application

The restaurant web frontend makes use of Masonry[1] to render responsive ordering card, jQuery UI[2] to create collapsible and search to search menu inventory system, howler.js[3] to create notification sound for restaurant staff. The whole application is built based on responsive design to make sure it renders as naturally as a native application on an iPad.

[1] https://github.com/desandro/masonry
[2] http://jqueryui.com/
[3] http://goldfirestudios.com/blog/104/howler.js-Modern-Web-Audio-Javascript-Library

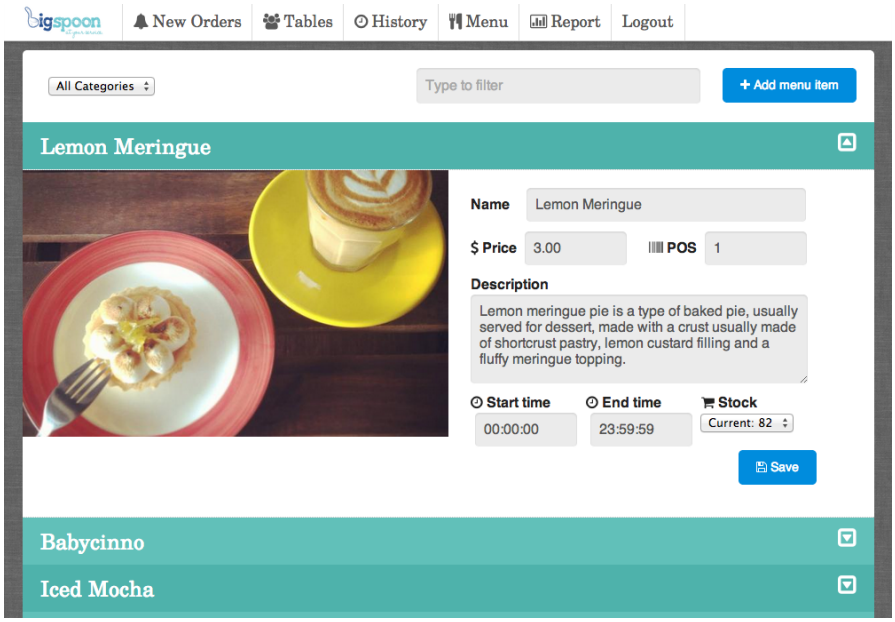


figure 6 Restaurant-facing inventory management page

We have gone through a number of iterations on the UI/UX of the restaurant web app, and we've closely listened to feedback from CS3216 consultations, the co-founders, users, as well as our those who dropped by our demo.

To illustrate, in response to concerns with navigating a long list of menu items, we have added a category filtering system (so you can browse by Breakfast, Sides, and so on) as well as a live search system (you can type the name of the dish you'd like, and the results are returned in real-time.) We have also added collapsibles and improved the performance of the page significantly through AJAX. This is just one example of how we've made efforts to iterate from user feedback.

Diner-facing iOS Application

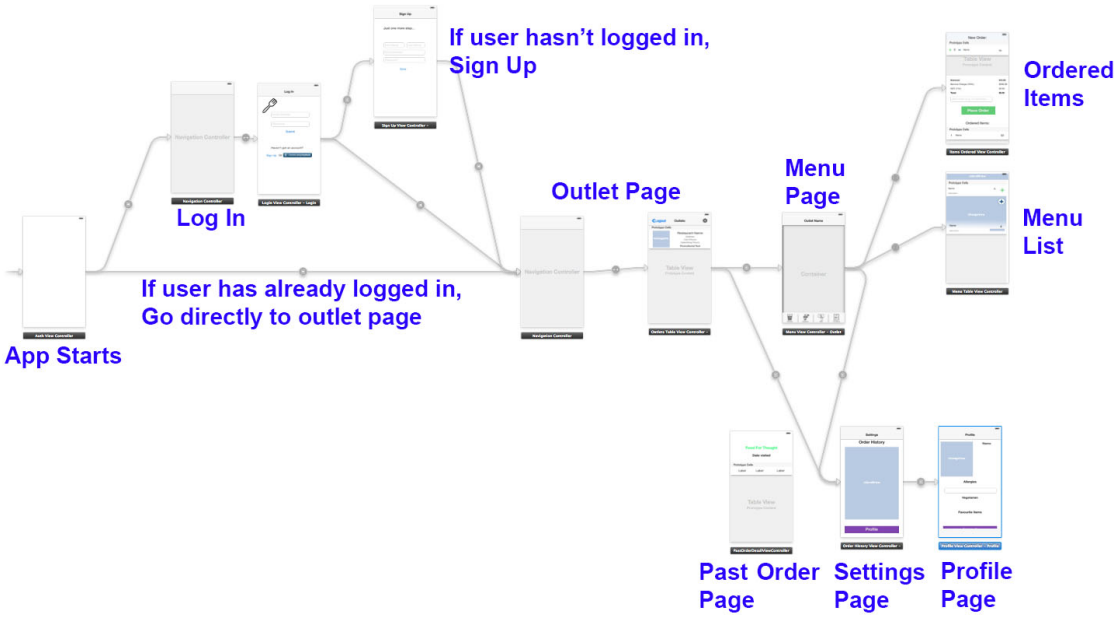


figure 5 BigSpoon iOS Application workflow diagram

The above is a screenshot of the pages in the iOS app. Each of the above page is represented by a UIViewController and the controller is representing a particular Model or a collection of Models. A few selected Models are shown in the following diagram:

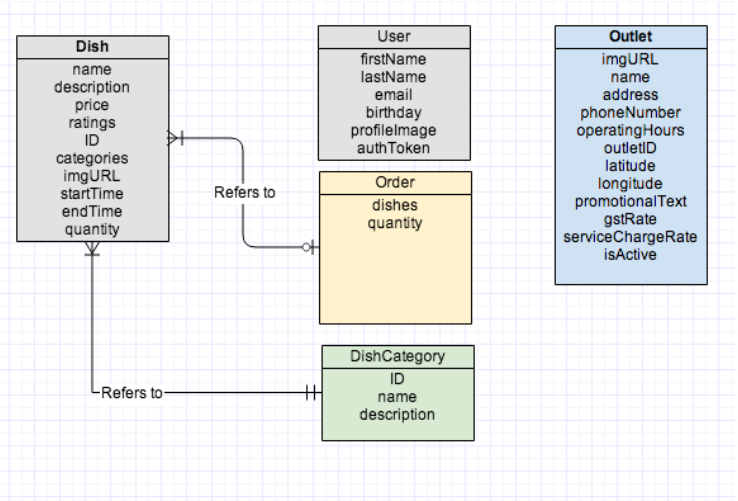


figure 6 BigSpoon iOS Model diagram

Bigspoon Backend Class Diagram

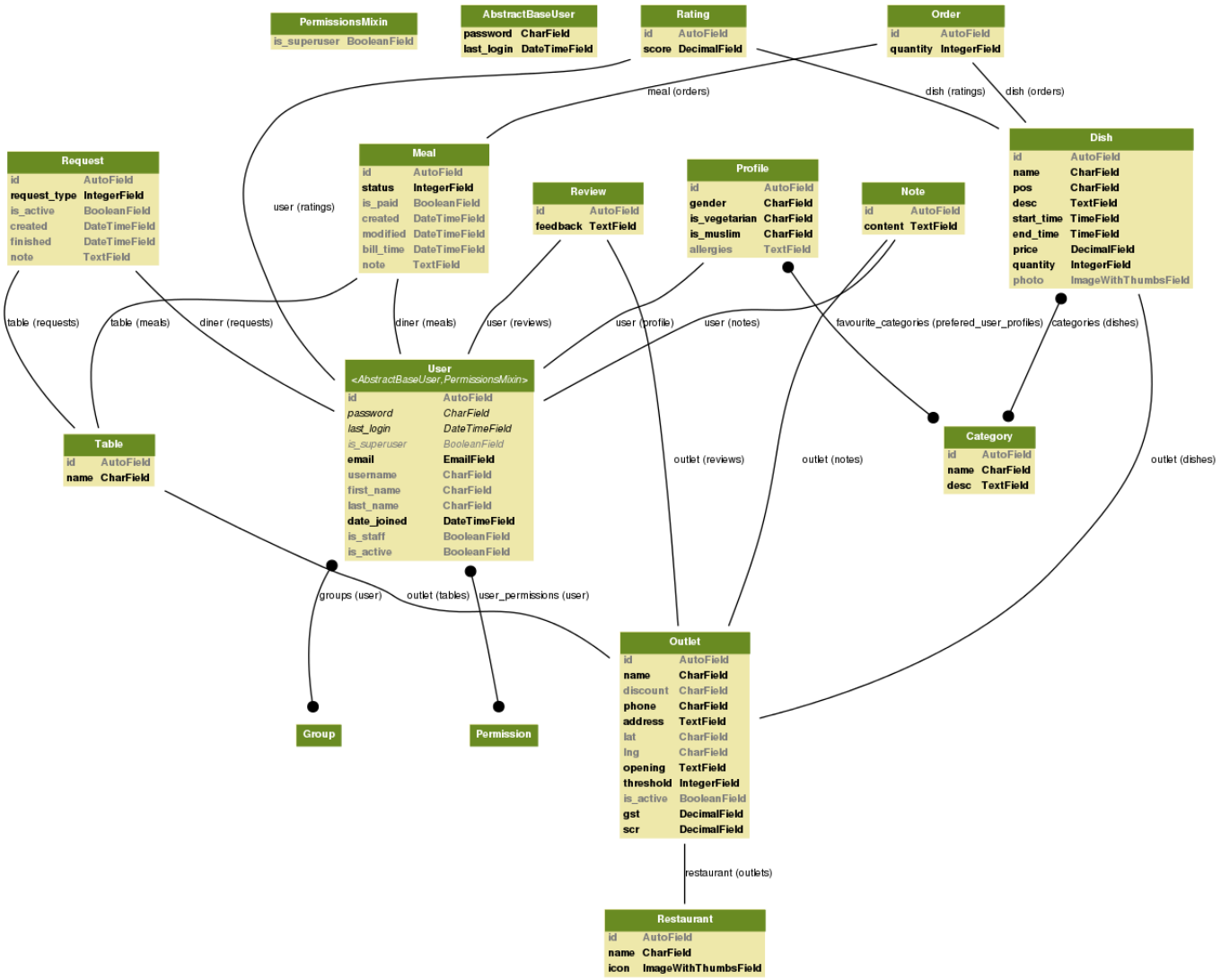


figure 7 BigSpoon backend model class diagram

The above is a graphical representation of backend model design, there are in total 13 classes, such as Restaurant, Order, User, Dish and so on. The whole design was focused on making it easy to setup the inventory system for new restaurants and to be as secure as possible. As shown in the diagram, a permission system is implemented to grant the correct permissions to only valid staff on their respect restaurant information.

6. Report on the current number of active users

Currently, Lola's Cafe is on board as a customer and Strictly Pancakes will be coming in as a customer in late December. According to BigSpoon's Jay, there are also 5 other cafes and restaurants that have expressed interest in becoming customers of the BigSpoon app.
They are: Maple Loft, Big Bites, The Book Cafe, Food for Thought, and Ice Edge.
During the poster session, we have more than 150 people who tried out the application and provided plenty of valuable feedback for us.

In general, during the project showcase and elsewhere, people have expressed interest in downloading and using the iOS application to order food and get better service.

Additionally, a few industry people have expressed interest in our project, for example, an associate of IncuVest, an NRF-Supported TIS Incubator, have expressed interest and gave us their cards. Additionally, the speaker for MavenTree who came to our demo was impressed by how mature (in terms of features, marketing, and users) the application has become in a short period of time of eight weeks.

7. Future plans and strategies

Based on the valuable feedback we've received during the project showcase, we learned the following:

- For the iOS app, we have received the following feedback:
    - Some UI elements can be placed to better positions. (Such as the submit buttons when the user requests water and bill). The style of some buttons can be polished.
    - Allow users to reorder the same set of dishes from his order history.
    - One person suggested that as a diner, she wants to know which cafes/restaurants are nearby.
    - Allow user to search restaurant by dish.
    - Allow user to edit the order before the order is acknowledged/processed by the staff.
- For the staff web app, we have considered the following improvement:
    - Import dishes from restaurant website or other source provided by the restaurant.
    - Allow staff to cancel wrong order for diners.
    - Make report more relevant for restaurant owner (e.g. segment report data by user and time, create daily & weekly report)
    - Integrate with POS system and payment system to make it more convenient for restaurant owner to manage order and inventory in one go.
    - Instead of refreshing the page via websocket, just insert or remove the relevant information to save loading time.

By the end of this month, we will launch our iOS application to the app store and make it available for download. We will also start deploying the application to a number of small cafes.

8. Insights gained from the project

What did you learn from doing the Final Project?

Name: Wang GaoXiang

- Network administration with ec2 server, using mysql database + nginx + gunicorn + django stack, using redis as data store for socket message
- Model design is very important for the project, you must think ahead of all possible scenarios or else it is very hard to change when project grows big.
- Time management in a team, based on each team members availability, it's better to arrange the development process so that at least half of the team is working on the project instead of having an 'empty' period and then 'pay back' when all teammates having time.
- Getting feedback often is really important for the project, the earlier you find out the problem the more time and more resource you have to recover from the mistake and make the project better.
- Writing documentation for API design is important, API documentation and example of usage is the best way to communicate between developers.
- Think not only from programmer's view, when present to stakeholders or product users, making the product easy to understand by them is the key, whereas technical details will not help but to make them more confused.
- UI/UX design, some basic features a product must have, a tutorial for new user, an easy to search page to find what user needs and at last a 'clean' and simple interface.
- Doing marketing is an art, most of the time what you think is not enough for actual run, so make sure run enough rehearsals to take care of all possible case could be happening.

Name: Yos Riady

- Picked up technical skills and gained more experience in the Python/Django stack, Javascript/jQuery, HTML/CSS, and a few third-party libraries.
- Software Engineering skills, such as REST API design & implementation, database model/architecture design, and systematic debugging techniques.
- UI/UX design. BigSpoon's staff frontend was one of the largest frontend projects I've worked on. Listening to feedback was a painful and enlightening experience which I believe has given me plenty to learn from in UI/UX design.
- User testing. From Su Yuen's consultation, I learned that in order to learn the most about the state of your UI/UX you should let the user try to figure it out themselves and simply observe where the user is stuck or has most difficulty. These are the parts of the app that needs to be fixed.
- The user's opinion is more valuable than any one team member's opinion. However, users don't know what they want until they see it. Thus, you need to be able to stand in their shoes in order to make something they would want to use. You can do this by talking to users and observing their workflow.
- Don't be afraid of throwing out old code and designs. If you feel too attached to any part of your work, you will never improve it. Consider all code as baggage you have

to carry; get rid of unused code the moment it becomes redundant. For any product, it's important to undergo a process of continuous iteration. Refactoring and throwing old code/design often leads to improvement. This is also true for the product requirements as a whole.

- Feedback is key. The earlier the better. As a software engineer, you are responsible to not only build things right, but ensure that you are building the right thing. It's possible that you end up with something really cool that nobody uses.
- When delegating tasks, you need to ensure that the other person is prepared and well-informed to do the task before actually passing it to them. You also need to be there to guide them whenever they are stuck.
- Constant communication is key, not just with the users but with other stakeholders. As a programmer working for other people, you are constrained.
- Sometimes an idea appears good at first, but once you've started building really thought through all the processes, you may discover that you are building a solution for a problem that doesn't need to be solved or that too few people have. At that point, you need to pivot and find the real problem.
- In retrospect, when demoing your product to users, it's best to group features with how those features benefit their daily life. Simply listing the things you can do with the application has less of an impact compared to showing how the application makes their lives easier.

Name: Yang Zhixing

- iOS development technical knowledge. Through the development of the iOS diner app, I learned much more about iOS development. I used various of frameworks and tools which I never had the chance to use before. I learned about SocketIO, AFNetworking, Customized Segue, Cache, etc. These knowlege will greatly benefit my future career.
- Meet your expectation. There was a time when I think that coding for 5 hours per day will be a great contribution, but soon I realized that the boss and the user will not care about how many long hours you have spent coding, rather, they only care about what feature is done. This is their expectation. In other words, spending 5 hours to finish a feature is the same as spending 1 hour to finish the same feature. When we code, we should have this in mind. Our ultimate goal is to finish the feature with high efficiency, but not the estimated working hour.
- Time Management. It is very important to have a clear picture of what are your tasks and how long it is estimated to finish these tasks. Some tasks are dependent on other people, and if you delay your part, you may delay other people's progress as well. When I was assigned a task, I would create a new task in my Google Calendar and keep a close on on the schedule. After this task is done, I would look back and summarize my gains and losses and thus I could have a better estimation and schedule next time.
- Project Management. Project of large scale is very hard to manage. This time we used a combination of various tools. I had difficulty selecting and understanding what each of the tools could do to benefit our project. In the end, we managed to make good use of them and we found them really very helpful:
    - **Google Docs**: Share the important documentations (such as specifications,

reports and APIs))which required working in collaboration with other teammates.
    - **Trello**: Keep track of tasks, issues and bugs.
    - **GitHub**: Needless to mention. Help us greatly in tracking versions and progress of the project.
    - **Facebook Group**: Post announcements, sharing of useful urls and articles.
    - **Facebook Group Chat**: Discuss issues lively.
- Pressure Management. I had 6 Computer Science modules this Semester and all are project-based. I had a hard time handling several projects at the same time. I had to keep a very close eye on all the deadlines and complete the missions one by one.
- Software Engineering skills. Apple's Xcode and iOS are designed in a nice Model View Controller (MVC) pattern. I had a much deeper understanding of this MVC pattern and how to design the models and view controllers.
- UI/UX Design. In the process of development the app, we often had long discussions about what's the best solution for some of the UI elements. We argued with each other and came up with solutions which will benefit the user's experience. Through these discussions and decision making, I had a deeper knowledge about the common sense and golden rules of UI/UX design.

Name: Chen Liang

- Backend design is a super important prerequisite that makes it possible for the app to be stable and polished.
- Some frameworks have nooks and crannies that are unique and not consistent - eg compression of js cannot be distributed.
- Learned Python/Django stack and gained experience with a few frameworks.
- Subtle feedback in apps is very important. Cues that indicate that something has succeeded or failed, but is in the background and does not require user response, and yet is easily noticeable and can be "understood".
- Enablers are important. Any team member who is better than another team member at something, can be an enabler by imparting the common pitfalls to his teammates, to remove the roadblocks that would occur and slow his teammates down. Otherwise, a teammate can waste many 5ours trying to solve a problem that you could have solved in 3 minutes. By being an enabler, you increase the productivity of the team as a whole, since the team can now move at the same speed.
- Need to understand the workflow of the target user.
- Need to come up with tangible measurable benefits to convince people. It is often related to the "workflow".
- It's easier to convince people about the benefit if you can help them visualize the scenario and understand it. Use an example that the person is likely to have witnessed or would be able to easily imagine in his mind - eg How long does it take for the waiter to walk from the kitchen to the dining table to serve the food.
- Is the problem caused by Internal factors or External factors? If it is external, it can be easier to use the same example for every person you pitch to. If it is internal, you need to find out if the person you're speaking to is the right target audience, and then relate to him by giving a use case he can identify with.

- Design takes a few iterations. Often have to throw out the entire old design.
- Colors matter a lot. The wrong choice of colors can hurt the eyes and give the wrong vibes even though the rest of the product is actually well-polished.
- Avoid the use of teal. It seems like a good and easy choice at first because of its blue-ish tint that looks easy on the eyes, but then it comes back to bite you. It's hard to match it with other colors and it seems to only match well with pale, dull or light greyish colors.
- Grid Card Design - The boundaries of the card matters a lot. Can only have a maximum of one eye catching boundary-denotation object on the card. Elements on the card cannot be too big or too wide (and colorful at the same time). Otherwise, the element ends up looking like it is denoting the border and makes the page look very very cluttered.
- Have a partner to bounce ideas with you. Sometimes, you don't notice bad design because you're the creator and you're used to it. In those situations, you need a change of perspective to improve on design.
- It's useful to have a 1-to-1 scale model of an A4 size paper in the PSD file when designing an A1 poster. It makes it a lot easier to imagine the sizes and make adjustments to get the right ones. An image may look small on the computer screen but is actually big enough in real life to be seen from 6-8 steps away.
- Learned Photoshop masking and manipulations.

Name: Shubham Goyal

- iOS development experience. I did the iOS part of the project along with Zhixing Yang, so I gained a lot of experience in developing iOS apps. I was reminded of how to do network communication in iOS, how to design interfaces, how to navigate, etc. etc. from CS3217 which I did 2 years ago.
- I also learnt a lot of new things in iOS. I learnt about and extensively employed auto resizing to make the interface look good on both 3.5 and 4" screen sizes. I learn about data storage on the device in the form of user preferences. I learnt about table views. I did not use the storyboard 2 years ago, but this time I forced myself to use it instead of taking the shortcut and doing everything in code. It took more time (not because the storyboard development path is slow, but rather, the learning curve is a bit steeper) but still, I am glad to say I did everything that could be done by using the storyboard on the storyboard.
- Learning from your group mates - I often learn from Zhixing Yang' code on how to do certain things. Many a time, I changed my code after looking at his or talking to him. For example, I was in the habit of using custom views inside UIScrollView. But I began to use UITableView later. There are many more examples like this.
- Time Management - I was doing my FYP. I wrote a research paper while doing part time work at SMART. I was also taking other modules. So, needless to say, I had again got myself into the situation where I had more to do than I had time for. Even though I did struggle, this experience made me stronger. Now, I never fret whatever the number of deadlines be.
- Task prioritization - Before this project, I used to attend every hackathon, big or small, in Singapore. But I have learnt to prioritize tasks now. I attended only one hackathon this semester. I also used to waste a lot of time reading questions on

Quora. I have changed that. This semester and CS3216 have forced me to use discretion while choosing a task to do at any given point of time.

- Project Management Tools - I have never seriously used any project management tool before. I am on boards in Trello but I never used to use it. Leon Qiao kind of forced/helped me to discover how much increase in productivity can be got by using these tools.
- I also learnt how to implement a design on PPT into code. We actually had a stub version of our app in the form of a ppt slideshow. This will be helpful in future because now I know how to make an app just given its design and interface :)