Final Project Report:

APP Design and Development for Hibachi -- A Connected Self-heating Lunchbox

Yumeng HOU
Supervised by Prof. Denis Gillet

Optional Project in Computer Science (CS-596) - 2016 Spring

Section: Computer Science (IN)
School of Computer And Communication Sciences (IC)

June 2016

Contents

1. Introduction

Eating at lunch can be difficult sometimes for people who pursue a healthy lifestyle. Bringing one's own homemade food and warming it up can be frustrating and a waste of time in many cases. Due to these constraints, most people have to resort to less satisfactory alternatives, either to bring a cold dish which is less healthy and tasty, or to unnecessarily invest more time and money in seek for an acceptable meal.

Hibachi (as shown in Figure 1) is designed to be a smart solution, as an elegant, connected and wireless self-heating lunchbox allows you to plan your lunch in advance via an APP (named "*hibachi*") remotely. Hibachi could warm your homemade food get it ready for your preset lunchtime without being plugged to a socket. It provides intelligent customization of the cooking process according to the contents of your meal. It means that the design goal of Hibachi is in one hand a **portable autonomous self-heating lunchbox** which is well-designed both visually and industrially, in the other hand it is a **connected device** to a smart terminal such as mobile phone so that information is communicated between the two devices and users should be able to have an **integrated, intelligent and friendly experience** via a software to access, monitor and control the lunchbox wirelessly and easily.

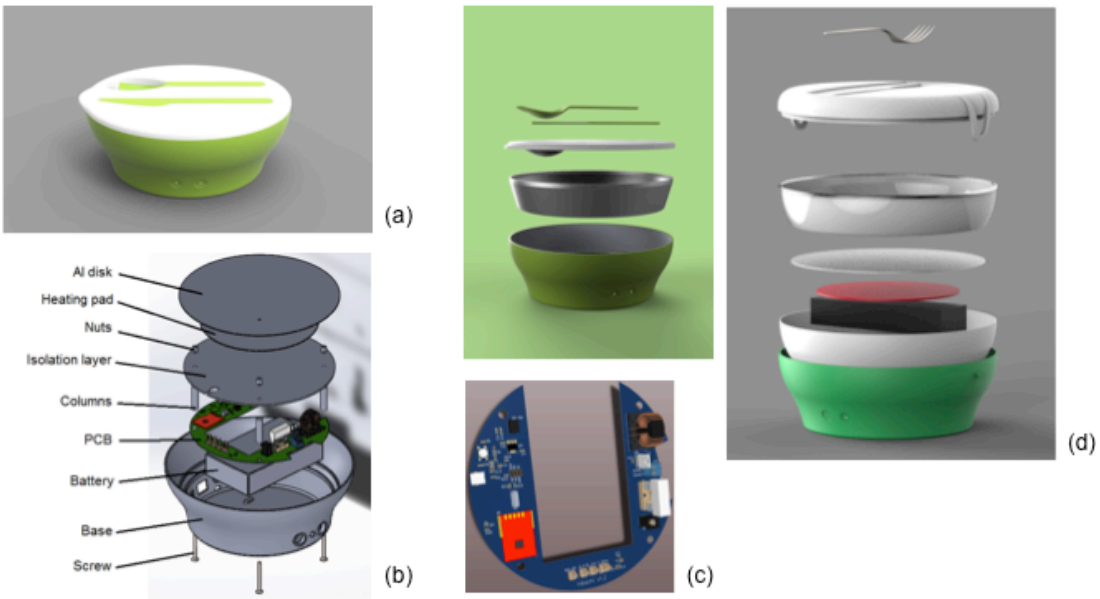

Figure 1: An overview of the design idea of Hibachi. (a) The overall appearance and industrial design. (b) The mechanical design solution. (c) The PCB design and the result of SolidWorks. (d) The industrial design and rendering of the components and the structure.

Hibachi is designed and developed by a group in CHIC (China Hardware Innovation Camp) with a sufficient diversity consisting of a businessman, two designers and four engineers. Due to my educational background of computer science, I majorly take the responsibility as the **software engineer** for this project, taking care of the **APP design and development**. In the meantime, I also collaborate frequently with the businessman and UX designer on the APP's **UX solution**, and also with other engineers to **review the technical solutions** in different aspects.

In this report, I firstly talk about the background and initial intentions of this project together with MVP specifications in Section 2, followed by a description of the methodologies in

Section 3. Section 4 is devoted to a clarification about the designing phases and how they were conducted specifically. In Section 5, I discuss the relevant technical solutions based on a review on all the existing ones, and how they are implemented. Section 6 covers the contents about the user evaluations and how they drive the iterations. Afterwards, in Section 7 I present an outlook on future work in relevant disciplines to myself. Last but not least, I give a brief reflection in Section 8 on how I feel and what I've learnt in the process of this project.

2. Background

2.1. China Hardware Innovation Camp (CHIC)

China Hardware Innovation Camp (CHIC) brings together engineering students from EPFL, designer students écal and business students from UNIL to form a team that is versatile enough to work on designing and developing a connected device. Each team members needs to take responsibility for one or two roles (e.g. Mechanical Engineer, Electronic Engineer, Firmware Engineer, Software Engineer, Businessman, Interaction Designer, Industrial Designer, etc.) and collaborate with each other from ideation to manufacture in small batches, and finally to a market-ready product.

CHIC involves design thinking, teamwork, project management and communicating with students from different disciplines, offering students a quasi-integral view of what it takes a product from idea to market. After the half-year prototyping and development of the device in Lausanne, teams fly out to China to finalize the devices at a local factory. In parallel, students do some visits and pitch the products in front of incubators, accelerators and Chinese makers.

Hibachi is one the four groups in CHIC 2016. There are seven members in the group, taking care of the following responsibilities either individually or collaboratively of business, industrial design, UX design, mechanical engineering, material engineering, electronic engineering, firmware engineering and software engineering.

2.2. Persona

A user persona is a representation of the goals and behavior of a hypothesized group of users. It is useful in considering the goals, desires, and limitations of groups of potential users in order to help to guide a user-centered design process. [1] Based on the data collected through our pre-design interviews and market research, we generate the personas among which the persona of Jessica serves as our primary focus for the design of Hibachi.

As shown Figure 2, Jessica takes care of her nutrition and wants to achieve a better lifestyle by eating healthy homemade food. But since she is quite busy and moves a lot in her daily working life, she suffers from the frustrations from not being able to enjoy the lunchtime with a warm and tasty meal in reasonable timeframe. She is eager for a smart solution to have a satisfactory lunch everyday ideally in an easy, quick and warm way.

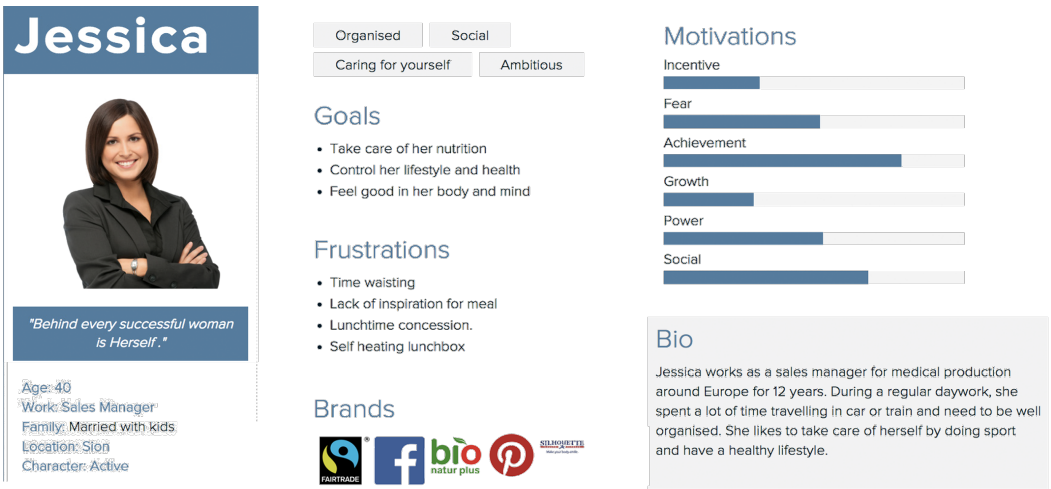

Figure 2: Primary user persona of Hibachi.

2.3. Our Value Propositions

People like Jessica need to go though several steps from planning, shopping, cooking, carrying, eating and post-lunch efforts to reach the goal of enjoying a satisfactory homemade warm lunch. We illustrate the whole process with a user journey map by Figure 3. It tells the story of our potential customer's experience and helps us to identify the gaps and points that are disjointed or painful [2], which serves a basic to work out our value propositions.

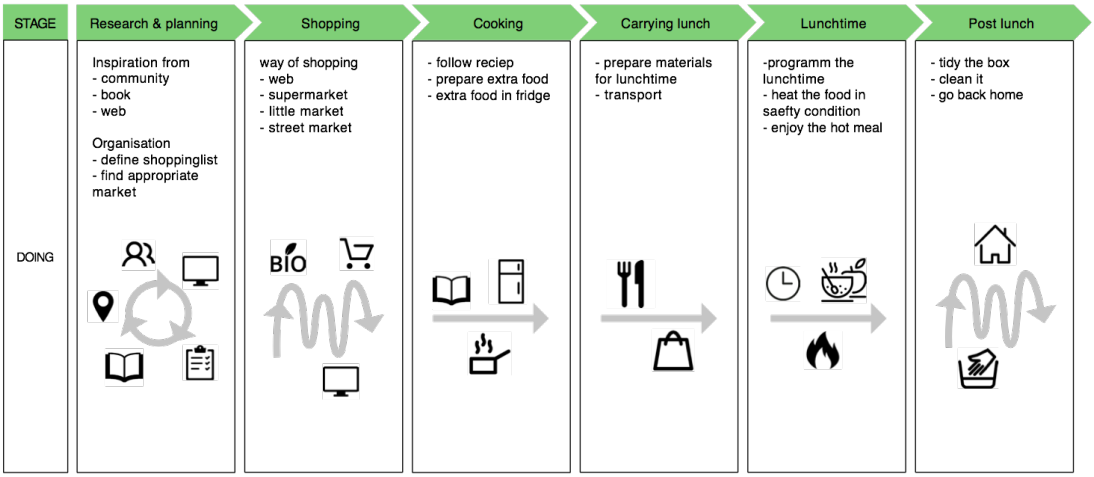

Figure 3: Typical journey map of our potential user to have a satisfactory lunch.

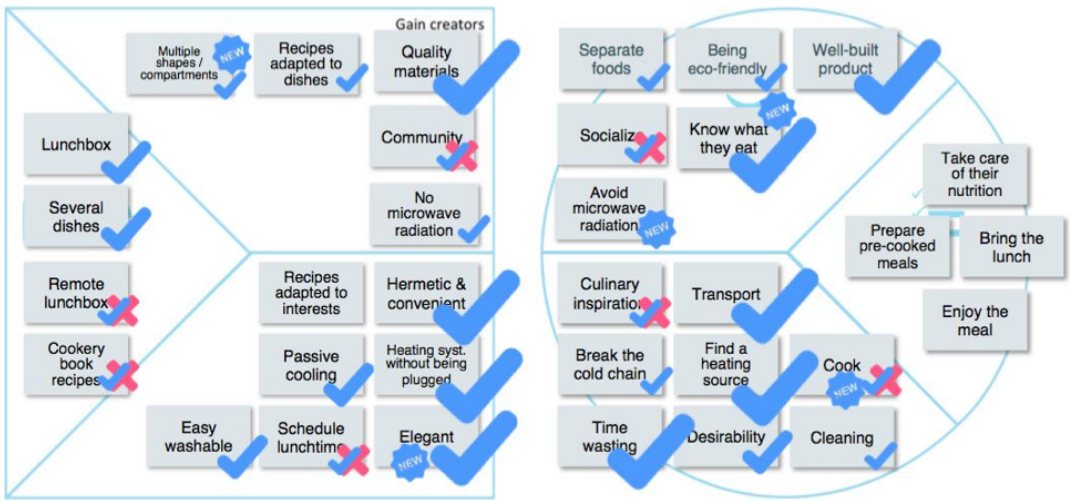

Figure 4: The latest value proposition canvas of Hibachi.

Figure 4 demonstrates our value propositions. We validate through the iterations that in doing the jobs to fulfill the ultimate goal of enjoyable lunchtime, our potential customers suffer the pains such as time wasted, bulky transportation, cooking and desirability. While they enjoy the gains from existing process for example if they are possible to know what they eat and if they could have a well-built product. Therefore, we defined our solution as an elegant, connected and wireless lunchbox which achieves the factors to promote the gains and also relive the pains. The product is positioned at people who take care of their nutrition, but have unsuitable conditions for lunchtime.

3. Methodology

3.1. User-centered Design

User-centered design is a framework of processes in which the needs, wants, and limitations of end users of a product, service or process are given extensive attention at each stage of the design process. User-centered design tries to design around how to enhance the experience on easily reaching the needs and goals for users rather than focusing on accommodating specific tasks and behaviors. It not only requires designers to analyze and foresee how users are likely to use a product, but also to test the validity of their assumptions in real world tests with actual users at each stage of the process. Such testing is necessary as it is often very difficult for the designers of a product to understand intuitively what a first-time user of their design experiences, and what each user's learning curve may look like. [3]

We go through our design process with a user-centered mindset in seek for validation from requirements, concepts, pre-production models, mid production and post production, finally to a circle of proof back to the original requirements to see if it's confirmed or to be modified.

3.2. MVP (Minimum Viable Product)

"The minimum viable product (MVP) is that version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort." In product development, the MVP is a product that has just enough features to gather validated learning about the product and its continued development. [4] A MVP has just those core features that allow the product to be deployed, and no more. The product is typically deployed to a subset of possible customers, such as early adopters that are thought to be more forgiving, more likely to give feedback, and able to grasp a product vision from an early prototype or marketing information. Gathering insights from an MVP is often less expensive than using a product with more features which increase costs and risk in the case where the product fails, for example due to incorrect assumptions. It is a strategy targeted at avoiding building products that customers do not want, which seeks to maximize the information learned about the customer per dollar spent. [5]

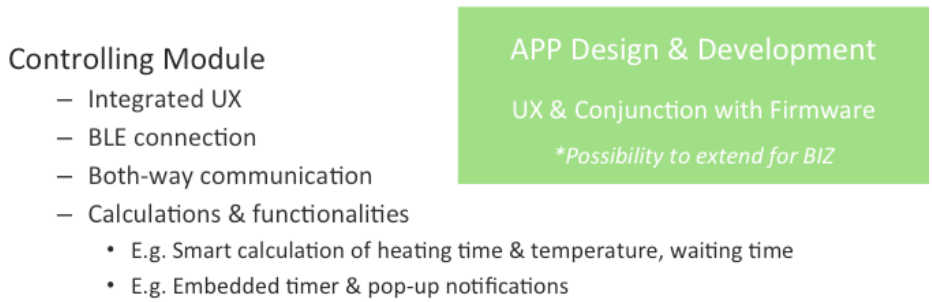

Figure 5: MVP requirements regarding software engineering.

We use the MVP as our guiding tool in product development. MCP requirements regarding to Software Engineering (SE) are demonstrated in Figure 5 with a list of core features. The SE requirements are majorly about an APP to control the lunchbox, which achieves an integrated user experience, works smoothly in conjunction with the firmware in the lunchbox, and saves the possibility for business extension. Technical features in specific, it should basically

achieve the BLE connection and both-way communications. In addition, the APP should achieve relevant calculations and functionalities internally, e.g. generate heating time, heating temperature and waiting time according to user's selections. It should also be able to count down time and push pop-up notifications in situations e.g. when food is ready.

3.3. Sprint planning

A sprint is a get-together of people involved in a project to further a focused development of the project. Usually during the sprint planning meeting, the product owner first describes the highest priority features to the team. The team then asks enough questions that they can turn a high-level user story of the product backlog into the more detailed tasks of the sprint backlog. Thus two defined artifacts are generally the results from a sprint planning meeting A sprint goal which is the description of what the team plans to achieve during the sprint, and a sprint backlog which refers to the other output(s) of sprint planning. [6]

We adopt the approach of sprint planning to plan and review software requirements. The sprint backlog in our case is a list of backlog items plus necessary blocks that we commit to delivering for the MVP. The blocks are organized along with CHIC schedule in consistence with the deliverables for each milestone, as described in Figure 6.

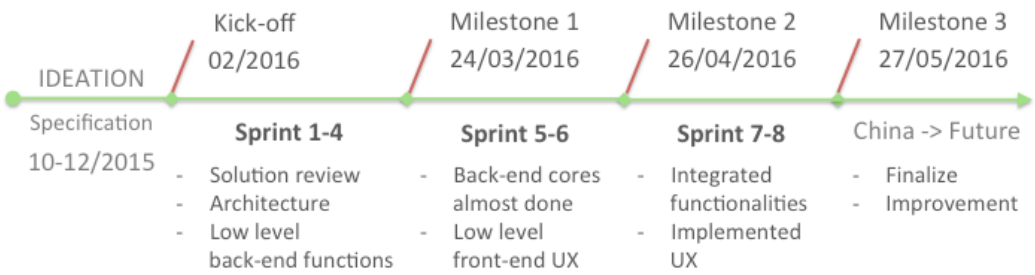

Figure 6: The sprint planning for my part of work.

4. Designing Phases

4.1. Information Architecture

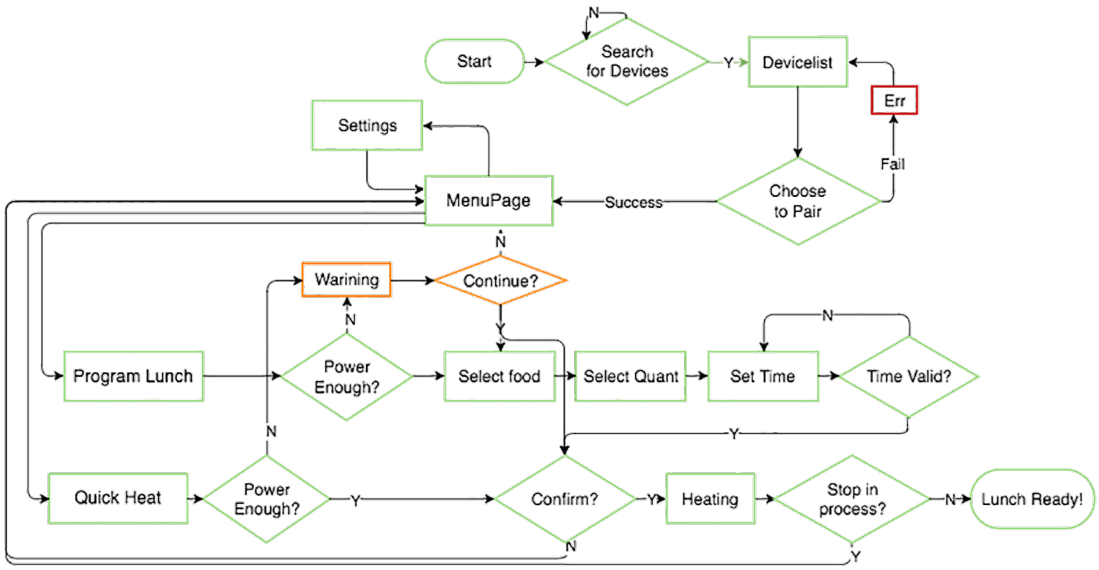

Figure 7: Information architecture graph of the APP *hibachi* [1]

Information architecture (IA), a concept borrowed from website design, is the structural design of shared information environments [7] and the way content is organized and categorized. It is the backbone of navigation, taxonomy (labeling in the case of website) and user experience. In our context of structuring the APP *hibachi*, as demonstrated by Figure 7, the user mental flow is also indicated by the IA graph.

4.2. Wireframe

A "wire frame" is a non-graphical visual guide that represents the skeletal framework of sites and software mostly, created for the purpose of arranging elements to best accomplish a particular purpose. [8] Wireframe is a tool broadly used for the prototyping of websites, mobile applications, computer software, or other screen-based products that involve human-computer interaction. [9] It depicts the page layout or arrangement of the content, including interface elements and navigational systems, and how they work together. [10]

Doing the wireframe is the first stage involving visually interactive designs following the information architecture, during which we majorly focus on: the range of functions available, the relative priorities of the information and functions, the rules for displaying certain kinds of information and the effect of different scenarios on the display.

Figure 8 illustrates our initial wireframes of *hibachi* in the first iterations. It aims to present the basic UX flow and serves as the initial guide for the initial visual prototyping and helps with software structuring in the early stages.

[1] The architecture stays to be updated according to the feedbacks from my project presentation.

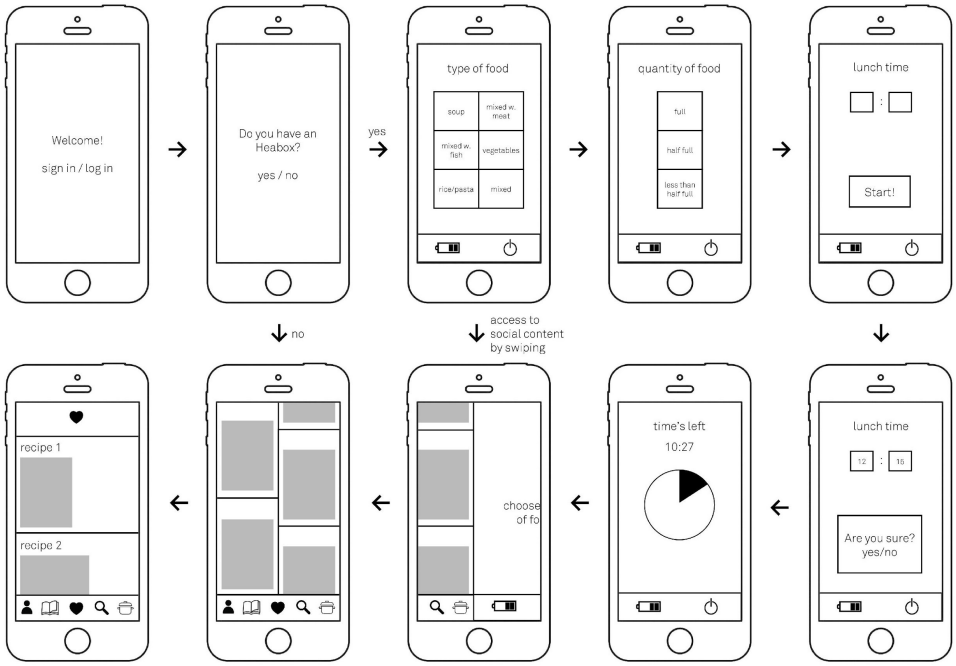

Figure 8: Initial wireframes of the APP *hibachi* in the early iterations.

4.3. Software Prototyping

Software prototyping is the activity of creating prototypes of software applications, color schemes and detailed interaction elements are involved during this stage. The outcome stays a simulation to but not necessarily the same with the final product. It serves for the purpose of allowing users to evaluate developers' proposals for the design. Prototyping can also be used by end users to describe and prove requirements that have not been considered, which is crucial in the commercial relationship between developers and their clients. In general, the prototyping process involves four steps: identifying basic requirements, developing initial prototype, reviewing, and revising. [11]

(a) (b)

Figure 9: Prototyping the APP *hibachi*. (a) Color schemes of 1st iteration. (b) Final color schemes.

Figure 9 shows two of the iterations that we have gone through for prototyping, indicating the process that we totally changed the color schemes. It is due to the reasons that when testing with and interviewing the users with our first version, we got the feedback they felt the red

color exerted too aggressive mental influence and thus distracted the ease with healthy food. Thus the whole group got together for a design sprint review directed by UX designer, discussing the color solutions and revising the mental flow to be more smoothly based on initial user testing during the interviews. We finally decide on using green theme for the purpose of conveying healthy lifestyle via Hibachi, which proved to be satisfying based on the results from the later interviews.

5. Technical Solutions

5.1. Pre-development Technical Reviews

5.1.1. A Brief Review on Wireless Communication Techniques

Wireless communication is essential for Hibachi to allow remote operations. There are many different techniques that can be utilized for wireless communication, such as infrared wireless communication, broadcast radio, Wi-Fi, mobile communication systems, Bluetooth, Bluetooth Low Energy (BLE), etc.

A. Infrared Wireless Communication

Infrared wireless communication communicates information in a device or systems through IR radiation. It is used for security control, TV remote control and short-range communications. A photo LED transmitter and a photo diode receptor are required for a successful infrared communication. The LED transmitter transmits the IR signal in the form of non-visible light that is captured and saved by the photoreceptor. So the information between the source and the target is transferred in this way. [12]

B. Broadcast Radio

The open radio communication is the first wireless communication technology to seek out widespread use, and it still serves a purpose nowadays. Handy multichannel radios permit a user to speak over short distances, whereas citizen's band. Mostly an audio broadcasting service, radio broadcasts sound through the air as radio waves. Radio uses a transmitter that transmits the data in the form of radio waves to a receiving antenna. To broadcast common programming, stations are associated with the radio N/W's. The broadcast happens either in simulcast or syndication or both. Radio broadcasting may be done via cable FM, the net and satellites. A broadcast sends information over long distances at up to two megabits/Sec (AM/FM Radio). [12]

C. Wi-Fi

Wi-Fi is a low power wireless communication technique that is used by various electronic devices like smart phones, laptops, etc. In this setup, a router works as a communication hub wirelessly. These networks allow users to connect only within close proximity to a router. Wi-Fi is very common in networking applications which afford portability wirelessly. These networks need to be protected with passwords for the purpose of security, otherwise it will access by others. [12]

D. Mobile Communication Systems

The advancement of mobile networks is enumerated by generations. Cellular and cordless phones are two examples of devices that make use of wireless signals. Typically, cell phones have a larger range of networks to provide the coverage. But cordless phones have a limited range. Similar to GPS devices, some phones also use signals from satellites to communicate. [12]

E. Bluetooth (Classic)

The main function of the Bluetooth is that permits you to connect a various electronic devices wirelessly to a system for the transferring of data. Cellphones are connected to hands free earphones, mouse, wireless keyboard, etc. By using a Bluetooth device, the information transfers easily from one device to another. It is applicable in various functions and it is used commonly in the mainstream wireless communication market. [12]

F. Bluetooth Low Energy (BLE)

Bluetooth low energy is a wireless personal area network technology designed and marketed by the Bluetooth Special Interest Group aimed at novel applications in the healthcare, fitness, beacons, security, and home entertainment industries. Compared to Classic Bluetooth, BLE is intended to provide considerably reduced power consumption and cost while maintaining a similar communication range. [13]

5.1.2. A Review on BLE Development

A. Native Development

Native apps are considered to be the highest quality apps in terms of performance and appearance. Native apps are developed for a specific platform (Objective-C or Swift for iOS and Java for Android), which some see as a drawback only because the iOS technology cannot be translated for the Android and vice versa, it has to be re-written. This means that typically, native apps cost more to produce because the code needs to be written twice in different programming languages.

Despite the drawback of cost, there are several pros that outweigh the cost including: increased app performance, getting promoted through the app store leading to more robust monetization and marketing in terms of rankings on APP stores. [14]

B. Web-app Development

Web apps are developed in HTML5, JavaScript, or CSS. Their primary advantages lie in affordability and flexibility as mobile web (often referred to as HTML5) apps are able to run on multiple platforms. Apps built this way can be accessed from iPhone and Android devices, as well as Windows and BlackBerry products. Convenience, expedience, and cost aside, there are some drawbacks to this type of app design tool. Mobile web apps are lacking in the following areas: offline storage security, device access (e.g. Bluetooth/BLE and camera) and management features. Additionally, unlike with other app development options, mobile web apps are not promoted via the app store. [14]

C. Hybrid Development

Hybrid apps ideally would be the best of both worlds as hybrids combine the best-of mobile web apps as well as the best-of native apps. They are developed using HTML5, CCS, or JavaScript before being put in a native application, which enables users to work in any

framework. Hybrid apps boast various merits such as diversity, cost-effectiveness (compared to both) and easy maintenance, but there are still some limitations. For example, hybrid apps have lower level device access than native and thus plug-ins, if available, are required to access the hardware. Like mobile web apps, hybrid apps are dependent on the native browser, which means that they lack the speed of their native counterparts. [14]

5.2. Adopted Solution: BLE + Hybrid Development

5.2.1. Why BLE?

As introduced in 5.1.1.(F), among the existing communication methods, Bluetooth is the most popular one used in smartphones which can already work individually (unlike Wi-fi). Bluetooth is a wireless protocol that allows nearby devices to communicate over radio waves. The problem with classic Bluetooth is its power-hungry nature and users need to recharge the devices regularly. Thus it is getting unfeasible for practical popularization of relative electronics.

Bluetooth Low Energy (BLE) is one of the standards offered by Bluetooth 4.0, which is much more energy-effective for many devices. It is especially suitable for the devices that don't need to send data constantly or only need to send tiny bits of data to achieve lower energy consumption, which is exactly the case for Hibachi.

Due to the reason that battery is already a big challenge for such a portable energy-based heating device and we want to save the energy consumption for both the lunchbox and the controlling terminal, we finally decided on using BLE as our solution.

5.2.2. Why Hybrid?

I go for hybrid APP as my development solution due to the following concerns.

Firstly, according to CHIC schedule, fast and cost-effective iterations are needed so that hybrid app proves to be a better solution than native app. Hybrid solutions boast an extreme advantage of a reduced cost of development across multiple platforms and the same HTML components can be used for different mobile OS. In the meanwhile, it makes maintenance easier than native apps in the meantime without asking for the efforts to upload whenever there is an update, which makes great convenience for frequent iterations.

Secondly, unlike a pure web app, hybrid solutions allow the full use of necessary device features (e.g. Bluetooth and BLE) via plug-ins, and allow functional access both online and offline, which largely eliminates the constraints of web development.

Therefore, hybrid app turns out to be a much better option over both native app and web app and combines the best of both in our context to reach the MVP requirements efficiently.

5.3. Development Environment and Platforms

5.3.1. RedBear Blend Micro

We use Blend Micro Arduino board as our firmware development platform (device).

Blend Micro is the first integrated development board from RedBear, which has the blended Arduino with Bluetooth 4.0 Low Energy into a single board. It runs as BLE peripheral role

only, it allows BLE central role devices to establish connection with. Blend Micro utilizes Nordic Bluetooth Smart SDK for Arduino, supports over-the-Air download of sketch and works with both Android and iOS. In addition, RedBear provides sufficient support in terms of plug-ins in its online developers' community. These features fit our technical requirements. [15]

5.3.2. iPhone and iOS

We use iOS as our software development platform and iPhone 5s/6 as the deployment device for the MVP due to the fact iOS has better native supports for BLE communication and also concerning the resources in hand.

5.4. A Hybrid Solution: Cordova/Phonegap Framework

Cordova (formerly PhoneGap), or Apache Cordova is a mobile application development framework that enables software developers to build mobile apps using CSS3, HTML5, and JavaScript instead of relying on platform-specific APIs like those in Android, iOS, or Windows Phone. [16] It enables wrapping up of CSS, HTML, and JavaScript code depending upon the platform of the device. It extends the features of HTML and JavaScript to work with the device. The resulting applications are hybrid, which means that they are neither truly native applications nor purely web-based. Because all layout rendering is done via web views instead of the platform's native UI framework and in the meantime they are packaged as apps for distribution and have access to native device APIs. [17]

Cordova/Phonegap is considered to be a perfect framework for hybrid solution in my case of development because it is one of the most stable mainstream frameworks and supports a wide range of operating systems and firmware devices including RedBear Blend Micro. It's easily extended with a sufficient supply of native plug-ins from various open-source communities. The plug-ins allow developers to freely add functionalities that can be called from JavaScript, allow access to the device's Bluetooth/BLE, camera, file system, microphone etc., and make it communicate directly between the native layer and the HTML5 page.

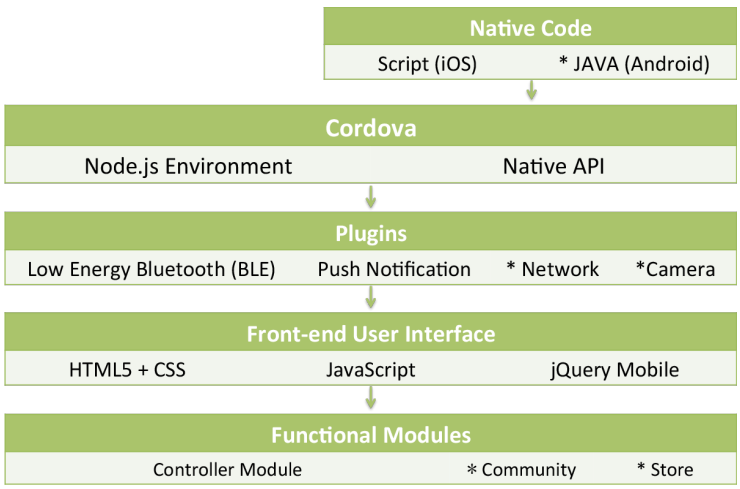

Figure 10: The mechanism of development with Cordova/Phonegap framework.

As demonstrated in Figure 10, the Cordova/Phonegapp framework simply works as a bridge between native development and web-based front-end development. It is easy to implement

and asks for no more efforts to learn a new programming language or a sophisticated tool.

5.5. Implementation

5.5.1. Specifying Communication Mechanism

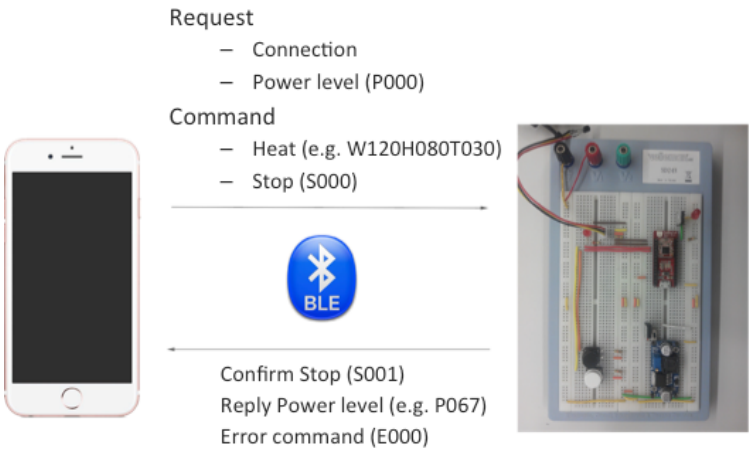

Figure 11: Message specifications between the software and firmware.

Figure 11 clarifies the message specifications that we used as the communicating mechanism between the software mobile app and the firmware board. To be specific:

(1) It's only possibly to launch the Bluetooth connection by the app.

(2) The app is going to ask for power status by sending a message "P000" and the board will reply in the form of "P0*xx*" where *xx* refers to the percentage of battery level.

(3) When user chooses to start a heating program, either quick heat or program the lunch, the app is going to send the heating parameters to the board with a message in the form of "W*xxx*H0*yy*T0*zz*", where *xxx* refers to the minutes for which Hibachi needs to wait before starting, *yy* refers to the expected temperature to which Hibachi should reach during heating and *zz* refers to the time length for the whole heating process.

(4) When user decides to pause the heating program in the process, the app will send "S000" and wait for maximum 5 seconds for a reply from the board with a message of "S001" to confirm the success of stop commands.

(5) Whenever the app is sending invalid message to the board, it would receive a message "E000" claiming that there is something wrong with the software.

On the software side, the function to send message is *app.sendData()*. And the key API to use is *bluetoothSerial.write(data, success, failure)*, provided by the plug-in that will be introduced in 5.5.2.1. *data* is the variable for the value of message to send, while *success* and *failure* are the variables of functions to call when the sending process is successfully conducted or not. The function to listen to the coming messages from the firmware is *app.onData()*, which is continuously called after a successful connection is established by *app.connect* with the API *bluetoothSerial.subscribe('#', app.onData, app.onError)*, where "#" is the token used to recognize the valid incoming messages from the firmware of Hibachi. As indicated by the API "*subscribe*", once a Bluetooth connection is established, the APP will start a stably continuous subscription relation with the connected device, here firmware of Hibachi.

5.5.2. Back-end development

5.5.2.1. BLE Plug-in for Cordova/Phonegap

After doing a series of research, I found there are a lot of BLE plug-ins supporting Cordova/Phonegap framework but only part of them are compatible with RedBear Blend Micro. Among the existing plug-ins, Don Coleman's open-source *Bluetooth Serial Plugin for PhoneGap* [18] to program Bluetooth communication is supposed to be most clearly documented one so I went for it after trying several ill-documented open-source plug-ins. This plug-in enables effective serial communication over Bluetooth with an Arduino. It supports various platforms such as Android, Windows Phone 8 and iOS with RedBearLab BLE hardware, Adafruit Bluefruit LE, Laird BL600, or BlueGiga. Android and Windows Phone use Classic Bluetooth, while iOS uses Bluetooth Low Energy (BLE).

Don's version also proves to be very efficient and easy to apply with a sufficient set of APIs compared to others. The most relevant methods in my codes are listed as follows:

- *bluetoothSerial.connect* – to connect with a specified deviceId and call *subscribe* to it
- *bluetoothSerial.disconnect* – to disconnect with a specified deviceId and call *unsubscribe*
- *bluetoothSerial.write* – to send (write) the message to the Arduino of connected device
- *bluetoothSerial.subscribe* – to continuously listen to the message from connected device
- *bluetoothSerial.unsubscribe* – to stop the *subscribe* status

5.5.2.2. Other Plug-ins from Various Providers

There is still a diversity of plug-in supplies for Cordova/Phonegap framework accessible online working in conjunction with RedBear Arduino, e.g. BedBear and Cordova/Phonegap related developer forums. The plug-ins altogether help me to easily achieve other different native access to functionalities besides Bluetooth, e.g. pushing notifications, battery access, camera usage, location access etc. All of them are easy to apply by simply using and modifying the APIs in JavaScript.

5.5.2.3. Back-end Programming Languages

A. JavaScript and jQuery for Major Functions

Most of the back-end programming, both structural and functional, is achieved by JavaScript in the same way of web development. I also use jQuery, a fast, small, and feature-rich JavaScript library to achieve HTML document traversal and manipulation, event handling, animation, and Ajax because it has much simpler and easy-to-use APIs that work across a multitude of browsers. [19]

The functions such as smoothing the page directions and redirections, the whole communication process together with looking for available devices and starting a stable connection are accomplished with JavaScript and sometimes the jQuery library. Other high-level functions leading the core workflow from choosing the mode from "quick heat" to "program your lunch", to set the food category, to set the food quantity, to set the time for lunch, to stop or to push notification and redirect when heating is finished, and the calculations and message generations related to each process are also achieved in JavaScript

programming. For example, the following functions are used during the setting flow and consequent calculations:

- *getFoodName(cate), getFoodQuant(quant), getWaitingTime()*
  - called in the .html file directly when user interact on selection/setting pages to get the contents of user's interaction
- *setFoodName(), setFoodQuant(), calwaitingTime(hourN, mimuteN, hours, minutes)*
  - called in the .js file whenever user confirms the selection/setting and they sometimes access the terminal (e.g. smartphone) information for relevant calculation
- *calHeatingPro(), calMsg2send()*
  - called for calculations when the relevant information are collected
- *quickHeat()*
  - called when user chooses to quick heat instead of programming the lunch
- *askPower(), confirmPro(), inprogress(), lunchReady(totalTime), stoppro(), successStop(), pushnotification()*
  - called for crucial behaviors inner- and intra- device(s) to lead the workflow
- *backmenuPage(), backFoodPage(), backQuantPage() ,back2Time()*
  - called for high-level UI flow related functions

B. Swift for iOS Native Programming

Under rare cases, native programming may also be needed. I abandoned the traditional native iOS language, Objective C, but used Swift instead due to its novel advantages. Swift is a new programming language for iOS, OS X, watchOS, and tvOS apps that builds on the best of C and Objective-C, without the constraints of C compatibility. It adopts safe programming patterns and adds modern features to make programming easier (to read and to write with less code), safer, more flexible, more compatible, more interactive and more fun. [20] Yet I didn't need to do a lot significant Swift programming in the whole development process because it is already mostly accomplished by the Cordova/Phonegap framework.

5.5.3. Front-end development

The UI of the app *hibachi* is achieved by web-based development using front-end programming languages such as HTML, CSS, JavaScript and XML-based languages. jQuery UI [21] is also used to program rich interactions, effects, widgets, and themes built on top of the Library. Responsive HTML5 is adopted for the future possibility of extension to the app developed on various terminals. The whole development process of front-end user interface is largely similar to the process of pure web programming.

5.5.4. Deployment - iOS Development Provisioning

Provisioning is the first process of preparing and configuring an app to launch on devices and to use app services and it is significant in my deploying. Development provisioning is done

by indirectly configuration through choosing options in Xcode. [22] During the provisioning, I operated in Xcode 7 to choose which device(s) can run the app and which app services can be accessed. I pack my codes into an APP named “chat2” and test it with my iPhone 6 with iOS 9.3. After building my provisioning profile for “chat2”, I download it from my iOS developer account to the deployment target device (my iPhone). The profile is as a result embedded in the app bundle, and the entire bundle is code-signed. This step is significant because the embedded provisioning profile should be installed on the device before the app is launched. If the information in the provisioning profile doesn’t match certain criteria or if the profile has not been validated by the target device to deploy, the app won’t launch. After finishing the provisioning process, I can launch deployment directly on Xcode 7.

In the deployment process of this project, I also benefit from the new mechanism of *free iOS development provisioning profiles* provided by Apple, which allows free deployment for individual developer to test a non-commercial APP on personal device, but only restricted to deployment on a private device belonging to the same developer’s account. It is a new feature only available after Xcode. Prior to Xcode 7, to test one’s iOS application on any physical device needs a payment to Apple for the privilege. But afterwards, it is allowed for to create free iOS development provisioning profiles and then use these to create standalone applications, which run on your own iOS devices. This is how I deploy for *hibachi* MVP in the budget-restricted development process. As for how to do a free iOS development provisioning profiles, detailed steps are referable online [23].

6. Evaluations and User-driven Iterations

We did rapid iterations in the design and development driven by user testing and evaluation, in the form of quantitative surveys and qualitative interviews. Four surveys and three rounds of interviews have been conducted to validate or modify our design in different levels and various aspects.

6.1. Pre-design Evaluations and Observations

The first round of evaluation was conducted quickly in October 2015 in order to observe the market and our potential users. Quantitative surveys and qualitative interviews were done for this purpose. Four qualitative interviews were conducted in total during this stage. Each interview lasts around 30 minutes starting with a 20-minute business description followed by a 10-minutes question session. There were two main objectives to achieve via the interviews and observations. Firstly, we want to identify the real gains and the real problems about the way they are achieving their goals (or not). Secondly, we want to produce evidence that our solution has a real need, and people have the willingness to pay for this solution.

As for the survey, questionnaires were spread online on Facebook in Veggie Group and Regime Group. We got 102 responses in total, among which 90% were female and it indicates the potential percentage of our future customers. Some key findings from the survey results are described in Figure 12. 75% of the participants bring food for lunch and 30% of them have trouble heating the food to satisfaction. An amazingly good result was that 50% of the participants showed interest in the initial proposal of Hibachi and 30% of them even left their emails showing the willingness to be further updated as well as interviewed.

Figure 12: Key findings in the first pre-design survey.

According to the results from pre-design evaluations and observations, we conclude that basically there is a potential market among group of people with special diet habits and we decided to focus on them to do interviews and more advanced surveys in the next stage. Some initial assumptions in our value propositions were proved for example the pain of bringing food and heating food true exists among the participants pain, while they show an interest to having a portable self-cooking lunchbox as a solution to address this pain.

6.2. Early-stage Observation and Validation

A. Qualitative Interviews

We conducted 8 interviews in early stage of the product design to further validate (or not) our assumptions in detail and to iterate the value propositions and design solutions accordingly. Two observation sessions were also conducted with two of the participants to see their cooking and eating environment.

Based on the interview results, we got valuable feedbacks to enrich and modify our value

propositions as well as inspiring new design ideas. We validated that the lack of efficient heating approach is a real pain of users and they want an alternative to microware for this purpose. What's important is that apart from the heating capability, users needs cooling as well since the broken cold chain is also a general pain among them. Overall, they enjoy having a smart lunchbox and plan for their lunch in advance via an APP believe it to be a pain reliever. In terms of the goals to achieve by the APP, they would like to have sufficient but easy control over their lunch and get notified with what important thing is going on, which are considered to be significant pain relievers. In addition, they look forward to achieve some social interactions such as culinary sharing and suggestion feeding via a community on the phone. And they expect the APP to have an appealing visual design to enhance their sense of wellbeing. These features play an important role as key gain improver and inspire a lot in our design plans, which were initially less valued by us.

B. Quantitative Surveys

The second round of survey was to explore the users' preferences of the product in detail and to get more specific inspirations in the design of our product. Questionnaires were spread on Facebook within the same groups as previously, 72 people participated and provided valuable responses. From the results, we conclude the key features to include in different disciplines. To be specific, in terms of industrial design, features such as multiple compartments (most useful to separate sauce, liquids and solid foods), good-quality and eco-friendly materials, non-bulky appealing design and washable dishes (by washing machine) are of highly importance among the participants. Regarding to technical requirements, they value the capability of both heating and cooling, and care about the frequency to charge the battery. While in the meantime, they don't care much about being totally potable as long as it can be changed with USB. In respect to the application design, participants have a strong expectation of awareness. They prefer the app to be easy and simple, but in the meantime they highlight their concern about interaction by accident, which is later taken as our crucial standards in interaction design and evaluations.

6.3. In-process Validation and Modification

A. Qualitative Interviews

Along the process of development, we made a branding decision of our product as Hibachi, which is originated from Japanese meaning an ancient on-fire grilling tool and conveys a better sense of good diets due to culinary cultural relevance. (The previous name was not a serious branding but only a temporary *nickname* of our product.) The initial iteration of brand (logo) design, UI design, UX flow and industrial design were generated in accordance with the new name. For example we chose the color schemes with a majority of deep red in the beginning, indicating the on-fire grilling which is relevant to the meaning of the branding. Afterwards, we did another round of interview for validating feedback (or invalidation). Four interviews were conducted. Each interview starts with presenting the intermediate results of APP UI, APP UX and industrial models, followed by some questions about users' impression and also in seek for critics. Some questions regarding the possibility in extension of our business model were asked at the end.

Overall, we got quite positive feedbacks regarding the designs in this round of interview,

especially in terms of the simplicity of designs. We received two critics on the colors claiming that the red color (as shown previously in Figure 9(a)) was too aggressive to convey a pleasant feeling of healthy diets consisting with current trends. To handle this problem, the whole team did a design review and discussed with related professionals. As a result, we change to a new set of color schemes with a center of green (as shown previously in Figure 9(b)), which is more satisfactory among the users.

B. Quantitative Surveys

In parallel to the in-process interview, we did another round of survey containing mostly the same questions (modified to be more targeted) as the previous two surveys plus a question about payment allowance but with different population, students from UNIL and EPFL. 81 students in total participated. This round of survey aims to re-validate the design decisions as well as to lead necessary modifications to refine the details in value propositions and design solutions. No major changes were derived from the results at this stage. But it led to several more in-deep team review meetings on the design and development solutions that helped fix some details in industrial and mechanical designs.

6.4. Post Design Evaluation

The last evaluation was majorly aimed for evaluating the latest design decisions and also the industrial model rendering results with users' feedback before the third milestone. We did a round of quantitative survey where 14 participants were involved. The questionnaire majorly contains scaling questions on different design decisions (from the general product concept to details such as the shape of the heating plate) and presentations of app UI together with model renderings in seek for validation (or not). For each of these questions, we asked for users feedback on agreement for each decision with a level scale 1-5 (from strongly disagreement to strongly agreement). Partial results are displayed in Figure 13.

Some questions about customers' pricing, marketing and functional expectations were asked in the end.

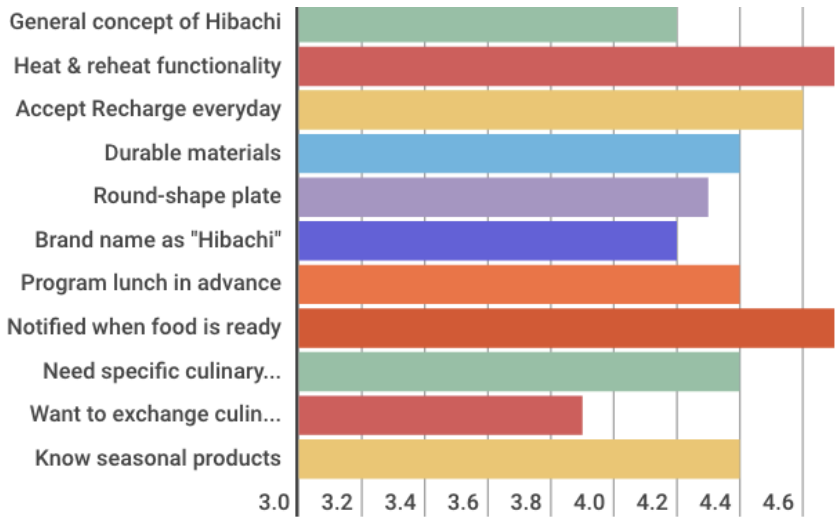

Figure 13: Partial results to agreement level of design decisions in the post design evaluation.

6.5. Supplementary: A List of Decisions and Modifications to Users' Feedbacks

Based on the collected user feedback plus market research results, we have updated our value propositions by preserving the validated elements, removing or modifying the invalidated ones and adding the new ones from users' responses (as shown previously in Figure 4) and our business model (Figure 13). Software requirements and UX designs have also been continuously modified, and sometimes inspired according to users' comments and expectations as well. As an example in specific, Table 1 gives a list debriefing some of the decisions and modifications made according to the users' comments and feedback.

Comments & Feedback	Decision / Modification
Users Like:	
- Manage remotely the time to eat	- Focus on controlling module for MVP
- Be notify when ready	- Add notification to MVP requirements
- Back into simplicity	- Strike for easy interaction flow
	- Strike for simple UI design (squares)
- Be aware of time left	- In balance with battery consumption, show time to eat instead of continually counting down time left, and plus notification
- Be aware of battery before start	- Add the battery level icon to the first interface and change the information flow accordingly
- Receive suggestions for meal	- Add community to future outlook
- Be aware of seasonal products	- Give the priority to group interaction and food suggestions in designing the community
- Have a community	
- Exchange with similar people	
- Send & receive suggestions	
- Send culinary pictures	
- Have advices in nutrition	
Users sometimes need:	
- Larger texts	- Add possibility to larger font size
- Stop during the heating process	- Add an emergency stop button on both app and box
Users dislike:	
- Start by accident	- Pop up confirmation widgets for crucial commands
- Stop by accident	- Change the emergency stop button to a slider, which is not easily accessed by accident

Table 1: A list of partial decisions and changes made according to user feedback.

7. Outlook and Future Works

7.1. Community and e-Shop

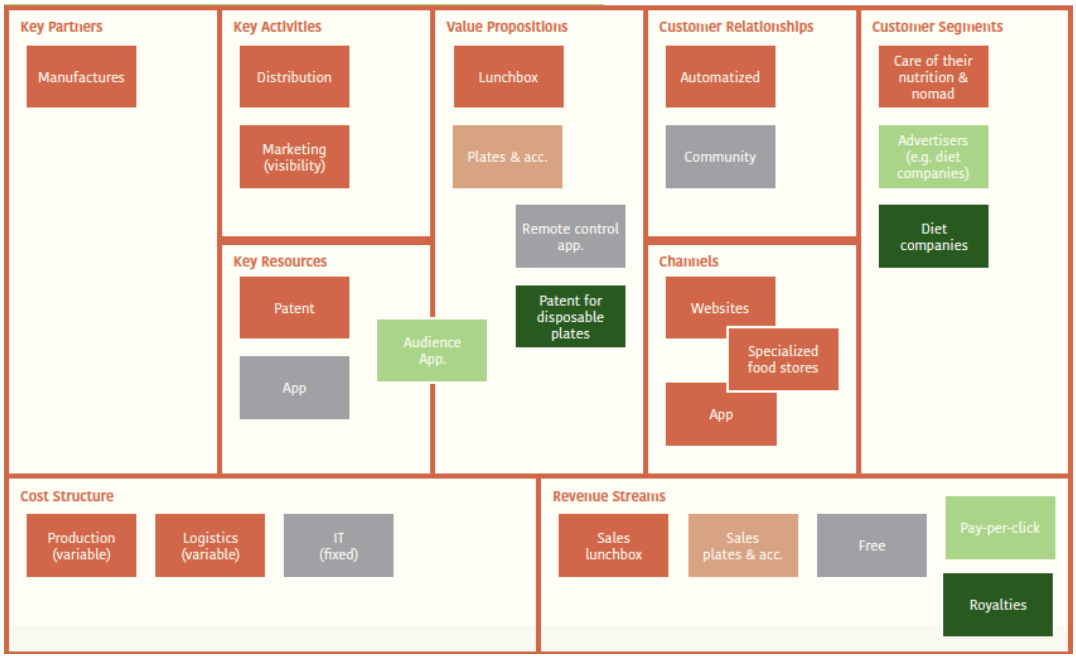

Figure 14: Latest Business Model of Hibachi.

As indicated by the business model of Hibachi as Figure 14, though in the first stage, our revenue stream is based on sales, but in the long-term, an extension with cooperative chains is achievable. Therefore, a community is sufficient important to gather the current users together with relevant social interactions for potential new stream of revenue based on those loyalties. In addition, a platform as official website and also community can also attract the advertisers to get more recurrent revenue with pay-per-click, and the collaborators for example diet companies to build disposable plates fitting to Hibachi. To better serve for the latter case, an e-shop is also within our consideration to be developed in the future.

7.2. Intelligent Food Recognition

Image process has been diversely utilized in daily life, for example food recognition. Kawano Y and Yanai K [24] proposed a real-time mobile food recognition system with the purposes for users to estimate calorie and nutritious and record the eating habits. We are interested in this application area and include it into the future possibility of our app, which achieves a better sense for our users of intelligence and autonomy.

7.3. Easy Pairing

When thinking of the future possibility of *hibachi* prevailing in the physical market, we figure out there will be an issue device pairing since it should be both convenient and well secured personally. Unique ID and QR code (inside the box) pairing are two approaches easily applicable. Other possibilities reserves the space for further researches.

7.4. Responsive UX on Various Devices

Wide cross-platform application is relevant in the future market. Therefore, besides the back-end compatibility, responsive UX is significant to achieve this goal. HTML5 has made the space for responsive development but set of codes stays to be developed.

7.5. Post-development Project Evaluation among Different Populations

Though we've done several iterations with separate evaluations in different branches, an overall post-development project evaluation aimed to test the overall UX plus functionalities and usability is desperately in need. Tests with actual users should be conducted among populations with different demographic backgrounds. And as a result, different UX designs and branches of modifications (for different versions) are possibly generated to better cater to users from different markets.

8. Reflection

In such a project, I encountered challenges from different aspects including pressures from practical development and fast iterations, real-world constraints and diverse team alignment. Overall, I've learnt a lot in this process either from my discipline or others.

8.1. Self-enhancement in My Discipline

With no previous experience in app development, I took up the responsibility of software engineering for *hibachi* by myself, which means that I've totally changed myself by learning and trying a lot of times from a pure novice to a real developer. I learnt about how to review and analyze the existing technical solutions and choose the most efficient one suitable for the requirements. I've greatly enhanced my skills in software engineering especially in real-world software project development. I also learnt a lot on user-centered design and how to align the project with users' expectation, which pushes me close to a UX designer.

8.2. Learning from Others' Disciplines

I also learn a lot from my teammates about others' disciplines. For example, I recognize that engineer's mindset is always about being functional, designers are always thinking about experience, and business people will definitely focus on sales and marketing. I got to know how designers design for experience, how businessmen iterate on VPs and business models and how engineers from other domains deal with real-world constraints.

8.3. Team Alignment and Project Management

I got tons of experience in how to align the teamwork with members from super diverse backgrounds and how to manage such a collaborative project accordingly. I learnt some techniques such as Sprint review/planning, which is quite an efficient approach to align the team tones. I got the knowledge that communications, documentations and specifications are significant in parallel development in team project, which should be the most valuable experience that will help in my future life.

Reference

[1] Persona (user experience). (2016, April 14). In *Wikipedia*. Retrieved June 10, 2016 from https://en.wikipedia.org/wiki/Persona_(user_experience)

[2] Boag, P. (2016, January 8). *ALL YOU NEED TO KNOW ABOUT CUSTOMER JOURNEY MAPPING*. Retrieved from http://www.cmodigitalforum.com/2016/01/08/need-know-customer-journey-mapping/

[3] User-centered design. (2016, April 30). In *Wikipedia*. Retrieved June 10, 2016 from https://en.wikipedia.org/wiki/User-centered_design

[4] Eric, R. (2009, August 3). *Minimum Viable Product: a guide*. Retrieved from http://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html

[5] Minimum viable product. (2016, May 17). In *Wikipedia*. Retrieved June 10, 2016 from https://en.wikipedia.org/wiki/Minimum_viable_product

[6] Mountain Goat Software. (2016). *Sprint Planning Meeting*. Retrieved from https://www.mountaingoatsoftware.com/agile/scrum/sprint-planning-meeting

[7] Rosenfeld, L., & Morville, P. (1998). *Information architecture for the world wide web*. Sebastopol, CA: O'Reilly Media, Inc.

[8] Brown, D. M. (2011). *Communicating design: developing web site documentation for design and planning (2nd ed.)* . San Francisco , CA: New Riders.

[9] Michael, A. (2009, October 22). *Wireframes*. Retrieved from http://konigi.com/wiki/wireframes/

[10] Garrett, J. J. (2010). *Elements of user experience, the: user-centered design for the web and beyond (2nd ed.)*. New York, NY: Pearson Education.

[11] Blackwell, A. H. (2015). Prototype. In *UXL Encyclopedia of Science (3rd ed.)*. Farmington Hills, MI : U·X·L.

[12] Agarwal, T. (2016). *Different Types of Wireless Communication with Applications*. Retrieved from https://www.elprocus.com/types-of-wireless-communication-applications/

[13] Bluetooth low energy. (2016, June 5). In *Wikipedia*. Retrieved June 10, 2016 from https://en.wikipedia.org/wiki/Bluetooth_low_energy

[14] Blue Whale Apps, Inc. (2016). *Native App vs. Mobile Web App: Know the Differenc*e . Retrieved from http://www.bluewhaleapps.com/native-apps-vs-mobile-web-app-know-the-difference.php

[15] Red Bear Company Limited. (2016). *Blend Micro*. Retrieved from http://redbearlab.com/blendmicro/

[16] Fermoso, J. (2009, April 5). *PhoneGap Seeks to Bridge the Gap Between Mobile App Platforms*. Retrieved from https://gigaom.com/2009/04/05/phonegap-seeks-to-bridge-the-gap-between-mobile-app-platforms/

[17] Apache Cordova. (2016, June 8). In *Wikipedia*. Retrieved June 10, 2016 from https://en.wikipedia.org/wiki/Apache_Cordova

[18] GitHub, Inc. (2016). *Bluetooth Serial Plugin for PhoneGap*. Retrieved from https://github.com/don/BluetoothSerial

[19] The jQuery Foundation. (2016). *What is jQuery?*. Retrieved from https://jquery.com/

[20] IOS Developer Library. (2016, March 21). *About Swift*. Retrieved from https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/

[21] The jQuery Foundation. (2016). *What's New in jQuery UI 1.11?*. Retrieved from https://jqueryui.com/

[22] IOS Developer Library. (2016, April 29). *Creating Your Team Provisioning Profile*. Retrieved from https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppStoreDistributionTutorial/CreatingYourTeamProvisioningProfile/CreatingYourTeamProvisioningProfile.html

[23] Roger, N. (2015, August 13). *How to Create a Free iOS Development Provisioning Profile*. Retrieved from https://livecode.com/how-to-create-a-free-ios-development-provisioning-profile/

[24] Kawano, Y., & Yanai, K. (2013). *Real-time mobile food recognition system*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 1-7).

Appendix

1. Official webpage of the product Hibachi: www.hibachi-lunchbox.com

2. Blog of the CHIC team Hibachi: http://www.chi.camp/projects/hibachi/

3. Link to online resources of the 3 Milestones (Demo presentation included): https://drive.google.com/drive/u/0/folders/0B_AItkZxxOtYb3RKX1hlYlREdmc

4. GitHub link for the APP *hibachi*: https://github.com/rainie26/chic-app