Milestone 0: Describe the problem that your application solves.

Problem:
At the end of semester, I have quite a lot of stuffs that I no longer use. However, I just do not want to throw it away, as it still works well. And the stuffs include my lecture notes which I spent lots of efforts on with lots of notes. So I feel it will be a waste if I throw it away. But I also do not want to keep it as it messes my room. What about posting it to Carousell for money? Forget, it is not very valuable. So post to Carousell for free? Hmm too complicated, just a small thing and no one want to travel a lot to collect it. What should I do?

Solution:
It will be the best if I can give the stuffs away, so that they can be useful for someone. So with this idea, we planned to create an app for giving away stuffs. User just needs to post the stuffs on the app and that's all. However, we do not want that someone just get all the stuffs, even the things that he does not need. So with the idea of giving more getting more, we created a bidding system. So that users have to bid for products. Who has the highest bid will get the product and these points are given to who posted the product. Then with these credits, users can use that to get more stuff that they need more.

Moreover, at the beginning, each user will be given 20 credits and the credits are not real money. So it is still free anyway.

Milestone 1: Describe your application and explain how you intend to exploit the characteristics of mobile cloud computing to achieve your application's objectives, i.e. why does it make most sense to implement your application as a mobile cloud application?

Until recent, there is no longer a big gap in performance and features between native mobile app and mobile web app. It is much easier and faster to develop a mobile web app.

The app is implemented as a mobile cloud application as we want to exploit the characteristics of a mobile web app and the processing power on cloud servers. Using any devices, user can login by their username and password to start using the app.
The app will run across platforms. Users can use any mobile devices that has a phone camera to take pictures and post a listing on our app, either Android or iOS. They can actually use their desktop web browser if they want to.
Processing will be handled by the cloud server. That includes image processing, validation of any requests, calculating and changing of bid points for users. As a result, the app will run smoothly on mobile devices.
We also want user to see the products when they are not online. Of course they will only see the old products that they've seen before. So a mobile cloud application seems to be the best choice.

Milestone 2: Describe your target users. Explain how you plan to promote your application to attract your target users.

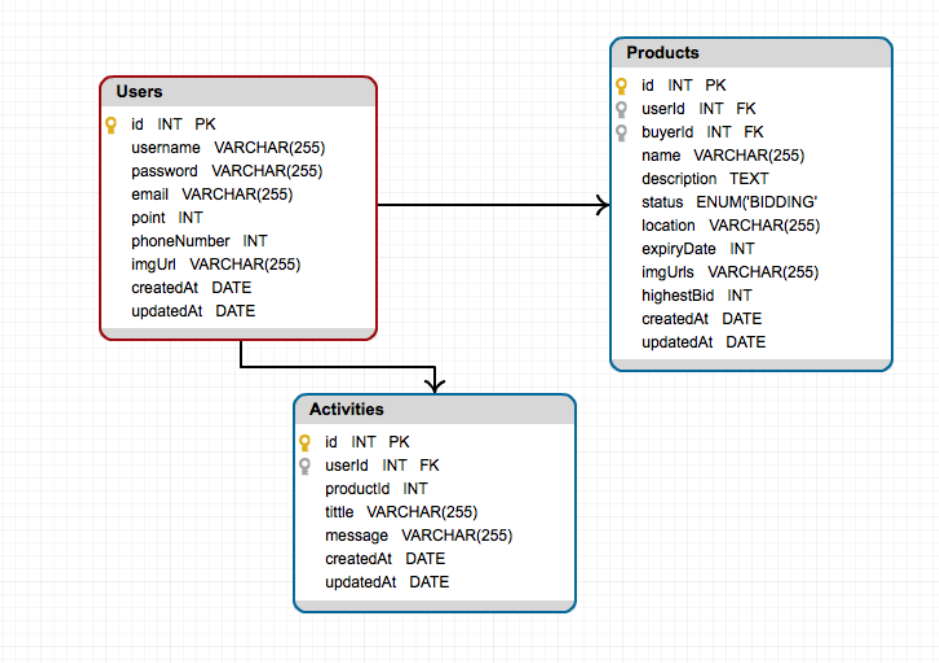
As they are free stuffs, users do not want to spend lots of efforts for giving it away (i.e collecting time, long distance, etc). Because of that, we target on NUS students so that it makes easier for contacting and collecting the stuffs. More than that, NUS students (like us) meet this problem usually, at the end of the semester. As most of us live in hostel, we usually need to change room / halls at the end of the semester and usually have lots of unused stuffs.

Currently, our target users is just NUS students. This is a small and close-knitted community where we can market our app using many ways. One of the most significant method is through social media and word of mouth. In the future, for our plan to promote the application, when a user signed up with a referrer, both of them will be awarded credits, which can be spent in the app to get items. This way, existing users will refer to more users and therefore our user base will grow. Also, we might want to use airbnb's spamming technique to Carousell, the same as what they did to craigslist to get more page views and sign ups and our target listings will be cheap items that doesn't worth selling, such as lecture notes, invaluable books, or unused pots and pans...

Milestone 3: Pick a name for your mobile cloud application. *(Not graded).*

FreeLah - as everything is free. Giving more getting more.

Milestone 4: Draw the database schema of your application.



Milestone 5: Design and document (at least 3) requests of your REST API. The documentation should describe the requests in terms of the triplet mentioned above. Do provide us with a brief explanation on the purpose of each request for reference. Also, explain how your API conforms to the REST principles and why you have chosen to ignore certain practices (if any.)

Note: All request and response data for the below API will be in JSON format. So to make it easier, only attributes in JSON format will be mentioned.

- POST /api/users
  - Description: This is to create a new user.
  - Parameter (data)
    - {'username', 'password', 'email', 'phoneNumber'}
  - Return value(s)
    - Response code: 200 (OK)
      {'id', 'username', 'email', 'phoneNumber', 'createdAt', 'updatedAt', 'point'}
    - Response code: 400 (Bad request)
      {'errors':[{'path', 'message'}], 'message'}
- GET /api/users/:id
  - Description: This is to get user information with id=$:id.
  - Header
    - Authorization: Basic $basic-auth-code
  - No parameter
  - Return value(s)
    - Response code: 200 (OK)
      {'id', 'username', 'email', 'phoneNumber', 'createdAt', 'updatedAt', 'point'}
    - Response code: 400 (Bad request)
      {'errors': 'user does not exist'}
    - Response code: 401 (Unauthorized)
- POST /api/users/:id/products
  - Description: This is to create a new product that belongs to user with id=$:id.
  - Header
    - Authorization: Basic $basic-auth-code
  - Parameter (data)
    - {'name', 'description', 'location', 'expiryDate', 'image'}
  - Return values(s)
    - Response code: 200 (OK)
      {'id', 'name', 'description', 'location', 'expiryDate', 'imgUrls', 'status', 'highestBid', 'buyerId'}
    - Response code: 400 (Bad request)
      {'errors':[{'path', 'message'}], 'message'}
    - Response code: 401 (Unauthorized)

- GET /api/products/:id
  - Description: This is to get product information with id=$:id.
  - No parameter
  - Response
    - Response code: 200 (OK)
      {'id', 'name', 'description', 'status', 'expiryDate', 'highestPoint', 'createdAt', 'updatedAt'}
    - Response code: 400 (Bad request)
      {'errors': 'user does not exist'}
- GET /api/users/:id/activities
  - Description: This is to get all activities belong to user with id=$:id
  - Header
    - Authorization: Basic $basic-auth-code
  - No parameters
  - Return values(s)
    - Response code: 200 (OK)
      {[{'id', 'title', 'message', 'productId', 'createdAt', 'updatedAt'}]}
    - Response code: 400 (Bad request)
      {'errors'}
    - Response code: 401 (Unauthorized)

The above API conforms to the REST principles and follows REST API standard. Specifically, it uses nouns for all actions and adds 'id' for getting/accessing specific object. By the way, it also shows the relationship between objects for some APIs. For example /users/:id/activities refer to all activities of a user (follow REST standard).

What is not included in REST API is the heading /api. This is just a way to differentiate the api routes and the normal web routes. Like for Facebook Graph API, they uses /v2.3 or /v2.4 to specify API routes and versions of the api as well.

Milestone 6: Tell us some of the more interesting queries (at least 3) in your application that requires database access. Provide the actual SQL queries you used.

Query 1:

```
INSERT INTO `Products`
(`id`,`name`,`imgUrls`,`description`,`status`,`location`,`expiryDate`,`userId`,`buyerId`,`highestBid`,`updatedAt`,`createdAt`) VALUES (DEFAULT,'watch',' images/product-7-550111.png','my old watch. No longer used so I want to give it away','bidding','pgpnus','1','7',NULL,0,'2015-09-25 04:52:54','2015-09-25 04:52:54');
```

This query is executed when a product is created. It simply insert a new row into the table Products.

Query 2:

```
INSERT INTO `Activities`
(`id`,`message`,`title`,`productId`,`userId`,`updatedAt`,`createdAt`) VALUES
(DEFAULT,'You have just created a new product.','New product: [watch]',84,7,'2015-09-25
04:52:54','2015-09-25 04:52:54');
```

This query is executed every time user execute an action. It will then insert a new row into the table Activities.

Query 3:

```
Executing (default): SELECT `id`, `name`, `imgUrls`, `description`, `status`, `location`,
`expiryDate`, `userId`, `buyerId`, `highestBid`, `createdAt`, `updatedAt` FROM `Products`
AS `Product`;
```

This query is executed when user wants to see all products. It will fetch all products from database and give an array of products.

Query 4:

```
SELECT `id`, `name`, `imgUrls`, `description`, `status`, `location`, `expiryDate`, `userId`,
`buyerId`, `highestBid`, `createdAt`, `updatedAt` FROM `Products` AS `Product` WHERE
`Product`.`id` = '84' LIMIT 1;
```

This query is executed when user wants to see a single product. It will find the product with it's id inside the database (with key WHERE you can see above, the LIMIT is also set to 1).

Milestone 7: Find out and explain what [QSA,L] means. Tell us about your most interesting rewrite rule.

QSA means append the query string to the end of the rewritten URL.
For example,
http://example.com/product/1/?view=short
would be rewritten into http://example.com/product.php?product_id=1&view=short
instead of just http://example.com/product.php?product_id=1
L is a flag that is used to stop the processing if this rule passed, all the rules below this line will not be checked

For our app, to make the URL more user-friendly, we used a routing library to route the URL (react-router) instead of URL rewriting. Basically, they serve the same purpose and have similar functionalities: allowing user-friendly URLs, but on different methods.

URL rewriting analyse the request URL and change it to a different URL on the server-side, it has no knowledge of what is the corresponding handler for the URL.
Routing is used to dispatch a handler upon calling a URL by mapping the URL to the routes that we specified.

Our app consists of some basic routes:

- `/`
- `about`
- `login`
- `logout`
- `signup`
- `products/new`
- `products/:id`
- `activity`

Milestone 8: Create an attractive icon and splash screen for your application. If you did not implement a splash screen, justify your decision with a short paragraph.

Icon and splash screen are implemented already. For the splash screen, Androi does not support that so we cant do anything.

However, we got some weird things that we could not figure it out in iOS. As we tested, the splash screen worked well on iPhone 5 with iOS 8, but not iPhone 5 with iOS 9. For iOS 9, it always shows a white screen. We tried lots of ways but not successful. So we believe this may be a bug on iOS9 as it is too new anyway).

Milestone 9: Try adding your application to the home screen to make sure that they are working properly. Make sure at least Safari on iOS and Chrome on Android are supported.

The app works perfectly on Androi.

However, for iOS again, the button to show the left-navigator does not work well for iPhone 5. As sometimes when we click on that button, it automatically moves to the home page / redirect to Safari. We tested and that button works well for Safari on iPhone. But when we make it a standalone application, that thing happens. We also tested on some other iOS devices (iPad air iOS 9) but we never meet that bugs.

After lots of attempts but not successful, we believe that this is an iOS bug again. However, we still leave the app as a standalone app, so that it can meet the requirements. But if you ever meet this bug, we just want you to believe that comes from iOS and test on Safari app instead.

Milestone 10: Style different UI components within the application using CSS in a structured way (i.e. marks will be deducted if you submit messy code). Explain why your UI design is the best possible UI for your application.

We designed our UI following Material Design, the design rules by Google. We used material-ui library to implement this. We also use Bootstrap Grid system for some of our components for the responsiveness. As a result our app is fully responsive and suitable for use in all mobile devices.

Material-UI, the UI library that we are using, used inline style for styling UI components. Using this UI library means our code will have styles inside the rendered HTML. In fact, there has been a lot of debate over using CSS versus using inline style. For reference, this is an article by vjeux, a Facebook Software Engineer working in the React team.

Put asides the UI library, we implemented our styling using a separate CSS file and when needed, using a modular way to style our components in ReactJS using JS, since it's difficult and yet to become hard to manage if we tried to use override the library's inline styling using CSS with `!important`. That is to put the styling in objects at the beginning of the JS block inside the rendering function of the component.

Milestone 11: Implement and briefly describe the offline functionality of your application. Explain why the offline functionality of your application fits users' expectations. State if you have used Web Storage, or any other technology. Explain your choice. Make sure that you are able to run and use the application from the home screen without any Internet connection.

For the offline mode, user can visit the app just the same as the online mode. Only except some online interactive actions, i.e. make a bid, login, signup and check newest activities. These actions need the interaction with server, so they are impossible in offline mode. However, in offline mode, user can still see the landing page, check all products and activities as updated the last time they went online. Moreover, the countdown for each product is still running so user can still check if that product is expired or not. In other words, user can use the full version of the app, except some interactive actions. And this should fit users' expectations.

We use Web Local Storage for storing the state of the app. As it is easy to use, fast writing and reading. The app also does not need to store too many things, so the Local Storage seems to be a good choice.

Milestone 12: Explain how you will keep your client synchronized with the server. Elaborate on the cases you have taken into consideration and how they will be handled.

In online mode, it will be the most updated data for every user's requests. So client will be always synchronized with the server. And for offline mode, because there is no interaction with server (so no access and change the database), we do not have to consider this case too much.

However, there are still some cases that user does not reload the app for a long time and the data may change during this time. There are some solutions for this, one maybe just to reload page/check data every 10/15 seconds. However, for this app, that reloading is not very necessary as the only actions that may be affected by this change of data is "making a bid". And for this action, every logical thing will be done in server. So server will return an error if the action is not possible in the new data. Then client gets the error, knows the data is outdated and update it.

Milestone 13: Compare the advantages and disadvantages of basic access authentication against other schemes such as digest access authentication, cookies, or OAuth. Justify why your choice of authentication scheme is the best for your application.

Basic access authentication has some advantages and disadvantages against other schemes such as digest authentication, cookies or OAuth.

First, for the advantages, basic access authentication is simple and easy to use. It just needs to convert user's credential to base64-encoded string and inject it into request's header. This is the easiest way to identify user among all considering authentication.

However, as basic authentication is simple, it has lots of disadvantages:

- As base64-encoded can be converted to original string, network hackers can recover the original password from this encoded string. The digest authentication does the same as basic authentication, except that it hashes the password using MD5 (or relevant) before sending it to the network. By this way, hackers cannot recover the original password, even when they get the hashed password.
- For basic authentication, the credentials of users are left wide open and are used illicitly; we will have no idea when and where these credentials are compromised. Moreover, there is no token management capability inherent in Basic authentication. This deficiency can make it nearly impossible to limit access to secured resources using Basic auth, without potentially having to disable the user's credentials completely. Compared to OAuth, it provides a token management that can store every information about where and when the requests are sent, it also gives the ability to limit user's access by storing data for each token. Refresh token and expiry time also make it safer.

For cookies, it is not very useful as it mostly identifies devices instead of users. So it will be hard for users when they want to use the app in another device.

Overall, for FreeLah, security is not an important problem as there is nothing related to sensitive information or money, etc. As the app's purpose is sharing and giving away. We only need a way to identify the users. For this purpose, basic authentication seems to be the best candidate.

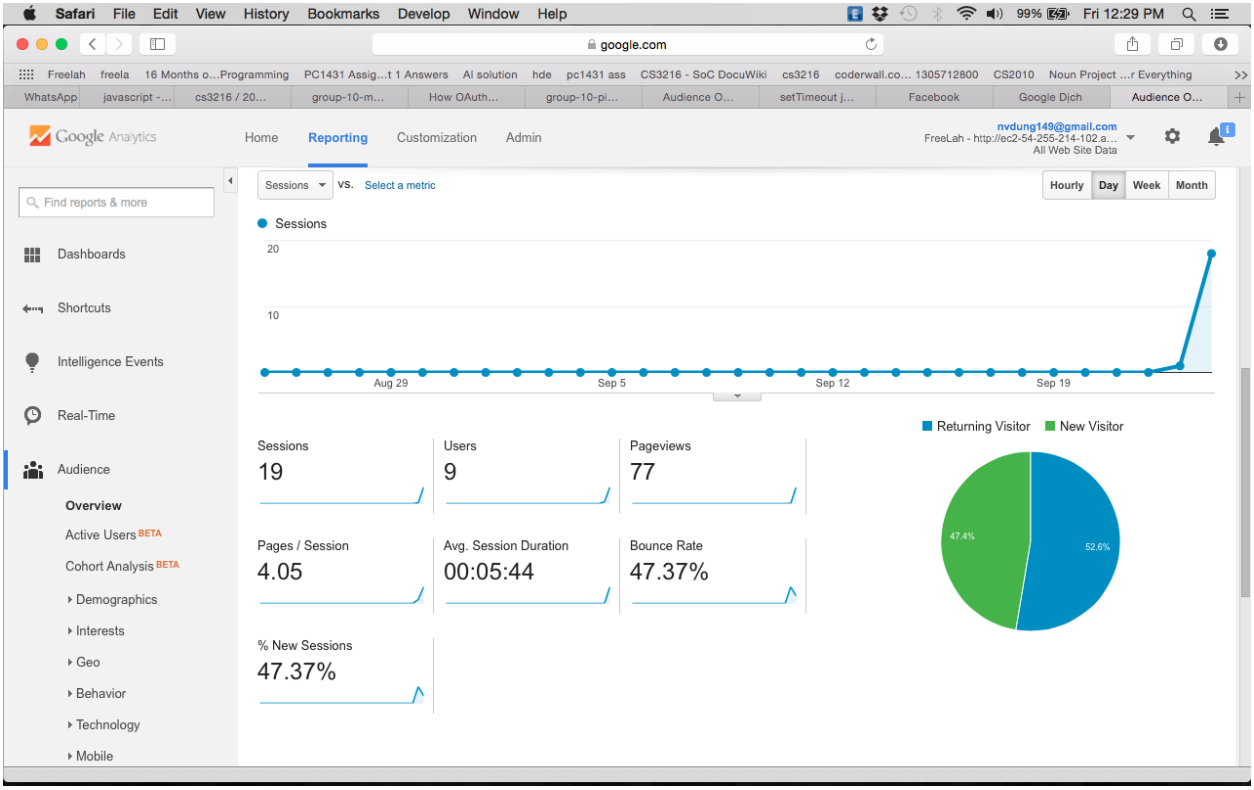
Milestone 14: Describe 3 workflows within your application. Explain why those workflows were chosen over alternatives with regard to improving the user's overall experience with your application.

1. When user gets in the app, he will stay at the landing page if he has not logged in or be redirected to product-page that shows all products if he has logged in already. So that for non-user people, they can take a look at the overview of the app, get its concept and know how to use the app. Then they can click 'Get started' to view the products as well as login, signup, etc. For already-user people, they of course quite understand the app and know how to use. So it is unnecessary to show to landing page. However, they can still see this page by clicking "About" in the left navigator bar.

2. After creating a product, users will get the inline errors if there are some troubles with the product's information. If not, the product is created and user will be redirected to this product's page. First for the inline error handler, it is such a necessary thing. So that users can know if they give the wrong information; and even better when users type something wrongly, they do not have to retype everything but just fix the wrong field and submit again. When the data is correct and users submit successfully, there are some actions we can do next. One is showing a dialog saying that "Product is created successfully ..." and then attaching a button linking to that product's page. However, we feel that dialog is not very necessary; and we want to simplify the app by reducing the number of actions users have to make. So we just ignore the dialog and move to the product's page directly.

3. In the products page, just some latest products will be showed. User have to scroll down to load more products. This is to fasten the loading speed. Moreover, we do not remove or hide expired or given products. So to make it easier for user, we move these products to the end of the product list. Because normally, user cares more about bidding products.

Milestone 15: Embed Google Analytics in your application and give us a screenshot of the report. Make sure you embed the tracker at least 48 hours before submission deadline as updates are reported once per day.



Milestone 16: Identify and integrate with social network(s) containing users in your target audience. State the social plugins you have used. Explain your choice of social network(s) and plugins. (Optional)

It is good to integrate this app with Facebook. So that it makes easier for user to use, share items with friends, invite friends, etc. However, time is soft and we may implement these features in the future.

Milestone 17: Make use of the Geolocation API in your application. Plot it with Bing or Google Maps or even draw out possible routes for the convenience of your user. (Optional)

We may want to implement this to show the location of the items. However, for the same reason, time is soft and we may implement these features in the future.

Milestone 18: Justify your choice of framework/library by comparing it against others. Explain why the one you have chosen best fulfils your needs. (Optional)

Front-end:
ReactJS has less steeper learning curve comparing to AngularJS. Having a team of lack manpower, ReactJS is a good choice for front-end development since our developers can learn it to a high standard in much shorter period of time.
Material-UI library by callemall (www.github.com/callemall/material-ui) is a well-maintained and well-documented ReactJS UI kit that can be used to develop UI for mobile web application. It provides great mobile usability and closest-to-native look and feel comparing to other ReactJS UI kits.

Back-end:
NodeJS with ExpressJS is a lightweight server-side framework, suitable for building a fast and responsive cloud server.

Build and deploy process:
We set up a Push-to-Deploy git server to quickly deploy our app. We also used webpack and gulp as module bundler and build automation system for easiest and hassle-free build process, which helped our development a lot.