Mobile Cloud Assignment

Milestone 0: Describe the problem that your application solves. (Not graded)

0.0 - Positioning Statement

For travellers and people who want to buy items from overseas,
Bacon Traveller is a online shopping platform
that leverages on the expanding travel network
to get items from all around the world for shoppers at home.
Unlike competitors in the ecommerce market, **our application** has **lower costs** as there is no shipping cost, allowing us to underprice the competition. Our application also **adds convenience**, cutting out the processing time that couriers take days to do.

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?

1.0 - Description

Many products are available globally, but some products may only be found in a certain country. Though shoppers may desire to have such products, they may not be willing to travel to that country just to buy the product, or they may not have the money to do so. Bacon Traveller is a platform that pairs up shoppers who want products from overseas with travellers who have extra luggage space to spare. Through Bacon Traveller, shoppers can get their hands on exclusive products that are only found overseas, while travellers get to make extra cash by helping others fulfil their wish list.

1.1 - Application Objectives

1. Allow Travellers to accept requests from Buyers
2. Allow Buyers to post items they want and get help from the Travellers
3. Browsing of items and get inspired what to buy
4. Simple user flows for both Travellers and Buyers

1.2 - Characteristics of Mobile Cloud Computing

1. Processing services in the cloud obviates the need to have a powerful mobile device
    a. Cloud computing enables all the hard work to be done on the cloud, eliminating the need for powerful hardware. It allows mobile access to information via smartphones. Considering over 2.6 billion smartphones are being used globally today, it would be smart for businesses to share their data through phones too. With cloud computing, an internet connection is all people need to use the application.

2. On-demand storage, easily scalable
    a. For businesses with fluctuating bandwidth demands, cloud based computing allows them to scale easily. Businesses can increase their cloud capacity when the need arises due to more user demand, drawing on the service's remote servers. Conversely, if there is a need to scale down, the service is flexible. Such agility can give businesses using cloud computing an edge over competitors, since most cloud computing services are pay-as-you-go, helping businesses save cost and scale easily.
3. Software availability and security
    a. Cloud computing provides the software for businesses. Cloud computing suppliers update their software, including security aspects, regularly so businesses do not need to waste time maintaining the system by themselves. Instead, they can focus on more important things, such as growing their business.
4. Security
    a. The loss of hardware is a harrowing matter due to the loss of sensitive data inside it. Cloud computing provides greater security when this happens. As data is stored in the cloud, it can be accessed no matter what happens to the machine.
5. Disaster recovery
    a. Disasters ranging from power outage to natural disasters can seriously affect a business. Downtime in your services leads to lost productivity, revenue, and brand reputation. Cloud-based services provide quick data recovery for all kinds of emergencies. With disaster recovery solutions such as cloud backup, businesses can recover from disasters gracefully and with little cost.

1.3 - Why We Use a Mobile Cloud Application

1. Processing services in the cloud obviates the need to have a powerful mobile device
2. On-demand storage, easily scalable
3. Software availability
4. Security
5. Disaster recovery

Characteristics of mobile cloud computing	**Why we use a mobile cloud application**
Processing services in the cloud obviates the need to have a powerful mobile device	We want our users to access the application with any kind of mobile device, especially travellers who may not bring their laptops with them when they travel. This would attract more users, since mobile devices are widely popular and much more convenient than computers.

	Moreover, cloud computing service gives us convenience since there is no need to install (from an app store), increasing our attractiveness to users.
On-demand storage, easily scalable	It is easy to scale according to our application needs. If more people use our application, the demand for cloud capacity will increase, and we can scale up easily. Such easy scalability also helps us save cost as we only pay for the features and capacity that we need.
Software availability	Cloud computing services provide us with all the features that we need, without us needing to maintain the system by ourselves. Hence, we can focus more on marketing plans to grow our user base, and also improve our app to increase our appeal to users.
Security	With our app data stored on the cloud, we would not need to worry about the loss of sensitive data through the theft of hardware. Cloud computing would allow us to retain the information in the hardware and at the same time wipe the data from the lost hardware remotely.
Disaster recovery	In the event of disasters, such as a power outage, we want an effective fast recovery time to restore data and functioning infrastructure, so that our users can get back to our app. Hence, we require backups among other disaster recovery solutions to help us not lose data and recover gracefully from disasters. We also want to save cost by not investing in third party expertise to help us in disaster recovery.

1.4 - Main Features of our Application

<u>Basic Features</u>

1. Post requests for products (by Buyer)
    a. Buyer can post requests on the products that they want to get from overseas
2. Accept requests to buy products (by Traveller)
    a. Traveller can accept Buyer's request and help them buy product from overseas
3. Browse requests with category filters

    a. Users can browse through requests to see which products are popular and which Travellers are reliable
4. Edit profile page
    a. Users can edit their own profile page to update their information and customise their profile pic
5. Search for users and products
    a. Users can search for other users or requests of products by using the search tool
6. Follow other users
    a. Users can follow other users they like

Nice To Have Features (to be added in the future)

1. Geolocation
    a. Fill in exact Buyer's location when creating request
    b. Notifications to Buyer about Traveller's location
2. Better tracking system
    a. Travellers to update location and status (e.g. already bought item, delayed flight)
3. Micro blog
    a. Travellers can post about products they are willing to buy and attract buyers
4. Super/verified Traveller
    a. Sourced for super (long term/ frequent) travellers, trusted by the platform
    b. Verify Travellers with good record, e.g. more than 20 deals closed with no complaints lodged by Buyers
5. Travellers can post deals and trips
    a. Travellers can post products that they are willing to buy
    b. Travellers can post about the trips and buyers can ask them to buy the products that are available at that location
6. Traveller need list of buyers
    a. Instead of each post having one Buyer, there can be many Buyers to one post by Traveller, since there may be more than one person wanting the product
    b. Traveller can limit the number of Buyers

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

2.0 - User Persona Cards

Jane

Age: Around 20-30

Occupation: University Student

Behaviours:

- Likes to shop
- Does not have much free time
- Does not have much money to travel

Needs and goals:

- Wants to buy products only available overseas
- Wants to save on shipping costs

Alex

Age: Around 30-50

Occupation: Working

Behaviours:

- Travels often for business and pleasure
- Has extra luggage space
- Krisflyer member, loves to rack up miles

Needs and goals:

- Wants to earn extra cash
- Wants to save money on his travels if possible

Catherine

Age: Around 30

Occupation: Working

Behaviours:
- Median income
- Buys luxury items
- A economical shopper

Needs and goals:
- Wants to buy those expensive branded items at lowest price possible

2.1 - Analysis of User Persona Cards

Through our user persona cards, we inferred that our application has two main benefits for shoppers.

Firstly, our application provides exclusive products at lower costs for our users. By getting travellers to help others buy items on their travels, we do away with expensive shipping costs that shoppers would face on usual shopping platforms such as Taobao, lowering the total cost of the product that shoppers would have to pay.

The application could also save the buyers a large sum of money when buying items from their country of origin. For example, buying Louis Vuitton from France, Michael Kors from US and Champion from Japan could be exceptionally cheaper than those in other countries. Especially for expensive items, the price different could usually be huge. This could be attractive for persona type 3 (Catherine) users.

Moreover, our application can bring convenience, by crowdsourcing to meet the demands of shoppers that are not willing or able to travel to buy the products they want. Shoppers can potentially get their items earlier than the usual overseas shipping duration which can take up to months, depending on the traveller they choose to help them. This could be attractive to people who need the items urgently, as they can attempt to find a willing traveller who is travelling back soon to help them buy the products.

For travellers, there are money making opportunities, which can help them pay for part of their travelling cost. In addition, they may also want to buy the same product, hence helping others to buy would be a convenient way to make some extra cash.

To deliver on our promise for convenience, our application should have a fast and intuitive UI, which makes it easy for users to make requests (for shoppers) and accept requests (for travellers) among other functions. There should also be little hassle after a deal is made, and no unnecessary procedures that could deter users from using our application.

Another point to note would be security. Since our application is based on trust between travellers and shoppers, we need to provide a secure way of making deals. For example, a feedback system is useful for reviewing travellers and shoppers, and deterring bad behaviour such as backing out of deals.

2.2 - Marketing Strategy

Target Audience:

- University students
- Working adults
- Frequent travellers, especially those with extra luggage space
- Online shoppers, especially those who want to buy products from overseas

Marketing Strategies

1) Social media (Facebook, Instagram, Youtube), Google

- Advertisements:
    - Advertise our app on Facebook Ads, targeted at our target audience as mentioned above
    - Advertise our app on Google Ads, targeted at our target audience as mentioned above
    - Advertise our app on travel/ air ticket booking sites to increase our traveller user base, e.g. Tripadvisor, Expedia, Skyscanner
- Teasers (before product launch):
    - Identify people that our target segments follow. Invite them, e.g. ecommerce sellers, traveller bloggers, to write about their thoughts about our app
    - Post bite-size information about our app at regular intervals e.g. twice a week
- Giveaway:
    - Publicise about the mystery giveaway item through social media
    - Participants are required to download and register on Bacon Traveller
    - Participants who use Bacon Traveller (as a Traveller or Buyer) will have a higher chance of winning
    - Winner gets the mystery giveaway item

2) Partnership with Ecommerce Platforms

- Collaborate with popular ecommerce platforms
- Ask ecommerce platforms to feature Bacon Traveller in their apps or websites
- In exchange, feature the shopping platform in Bacon Traveller's marketing efforts

3) Physical Media

- Put up posters around various places (with permission), e.g. NUS, to promote our app to our target audience

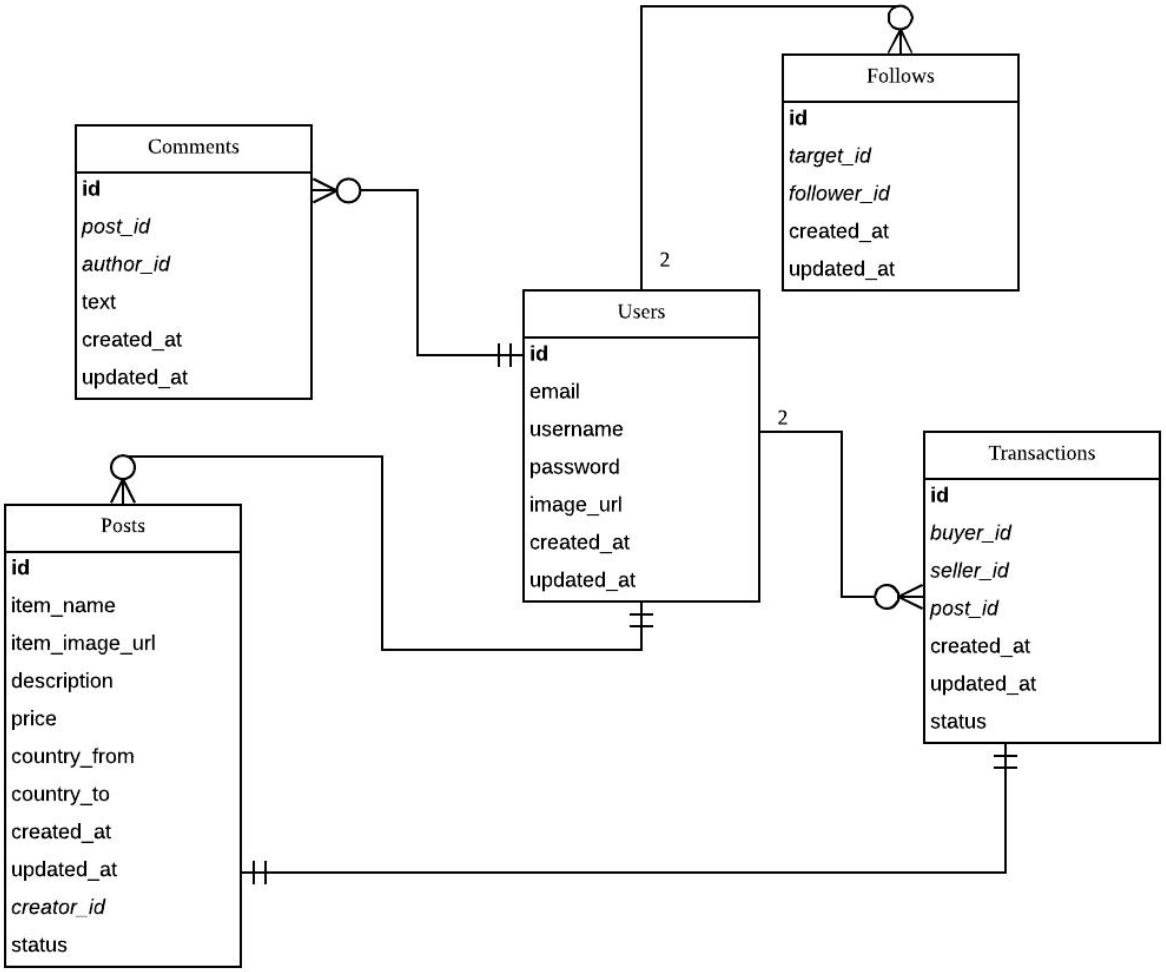
4) App Benefits

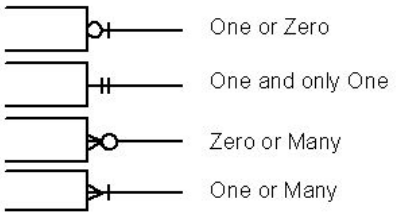
- Benefits for travellers
    - Travellers who close a deal successfully can get credits, which can be used to purchase products
    - In order to seed the app with enough traveller to handle requests, we have to contact enough frequent travellers that are willing to take up our offer to participate in app activities. In turn, we can publicize their travel stories, or help them get followers on Instagram
- Referral program
    - Users can get credits for each person who successfully registers on our app. These credits can be used to purchase products.

5) Landing Page

- Purpose of landing page:
    - To introduce our app to people and let them know how our app can benefit them
    - To get people to download our app after product launch (call to action button)
    - To update people about our app status e.g. launch day, new features
- Put up landing page link on social media and in advertisements

Milestone 3: Draw an Entity-Relationship diagram for your database schema.



Legend	Crow's Foot Notation
Bold: primary key *Italics:* foreign key	**Summary of Crow's Foot Notation** 

Milestone 4: Design and document all your REST API. If you already use Apiary to collaborate within your team, you can simply submit an Apiary link. The documentation should describe the requests in terms of the triplet mentioned above. Do provide us with an 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). You will be penalized if your design violates principles with no good reasons.

4.0 - Apiary Link

Apiary Link: https://bacontraveller.docs.apiary.io

4.1 - How our API Conforms to REST Principles

REST Principles	Our API
1. Uniform Interface Individual resources are identified in requests using URIs as resource identifiers. Each message sent between the client and the server is self-descriptive and includes enough information to describe how it is to be processed.	✓
2. Stateless Interactions All information necessary to service the request is contained in the URL, query parameters, body or headers. The server will not store data about the client's HTTP requests.	✓
3. Cacheable Clients can cache the responses, which improves performance for client side while increasing scalability and decreasing the load for server side.	✓
4. Client-Server The client and server are able to evolve separately without any dependency on each other. The client only knows the resource URIs and is not concerned with the data storage, improving the portability of the client code. Also, the server is not concerned with the client interference thus the server is simpler and easy to scale.	✓
5. Layered System The client cannot tell if it is connected to the end server or to an intermediary along the way. The intermediate layer helps to enforce the security policies and improve the system scalability by enabling load-balancing	✓

Milestone 5: Share with us some queries (at least 3) in your application that require database access. Provide the actual SQL queries you use (if you are using an ORM, find out the underlying query and provide both the ORM query and the underlying SQL query). Explain what the query is supposed to be doing.

5.0 - SQL Queries

1. Retrieve the list of posts

This query retrieves all posts, and joins them with the post's creator information, and transactions and comments made based on the posts, if any.

ORM:

```
Post.findAll(self.fullAttributes())


fullAttributes: function() {
  return {
    include: [{
      model: Transaction,
      as: 'transactions',
    },{
      model: Comment,
      as: 'comments',
    },{
      model: User,
      attributes: ["id", "username", "image_url"]
    }],
  }
}
```

SQL:

```
SELECT p.*, u.*, t.*, c.* FROM "Posts" p
INNER JOIN (select id as user_id, username, image_url from "Users") u
on p.creator_id=u.user_id
LEFT OUTER JOIN "Transactions" t on t.post_id=p.id
LEFT OUTER JOIN "Comments" c on c.post_id=p.id;
```

2. Create a new post

This query inserts a new post when a buyer makes a new offering request. THE SQL query given here is simplified as sequelize automatically appends datetime creation and update columns.

ORM:

```
Post.create({
   creator_id: req.user.id,
   item_name: req.body.item_name,
   item_image_url: req.body.item_image_url,
   description: req.body.description,
   price: req.body.price,
   country_from: req.body.country_from,
   country_to: req.body.country_to,
   status: 'PENDING'
})
```

SQL:

```
INSERT INTO "Posts" (item_name, item_image_url, description, price,
country_from, country_to, creator_id, status)
VALUES ('apple', 'https://apple.com/apple-img.png', 'red thing',
'$1000', 'sg', 'my', 2, 'PENDING');
```

3. Update a post

This query allows the creator of the post to update it.

ORM:

```
Post.findById(req.params.postId)
   .then(post => {
    // not shown: some endpoint level validation here to ensure
post exists and creator is requestor
     return post
       .update({
         item_name: req.body.item_name || post.item_name,
```

```
          item_image_url: req.body.item_image_url ||
post.item_image_url,
          description: req.body.description || post.description,
          price: req.body.price || post.price,
          country_from: req.body.country_from ||
post.country_from,
          country_to: req.body.country_to || post.country_to,
          status: req.body.status || post.status
        })
```

SQL:

```
UPDATE “Posts”
SET item_name = ‘new name’, item_image_url = ‘https://new-img-url’
WHERE id = 1
```

Milestone 6: Create an attractive icon and splash screen for your application. Try adding your application to the home screen to make sure that they are working properly. Include an image of the icon and a screenshot of the splash screen in your writeup. If you did not implement a splash screen, justify your decision with a short paragraph. Add 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.

6.0 - Icon

Our icon is a simple ‘B’ on a teal background, as we feel that simplicity is key in branding. ‘B’ stands for Bacon Traveller, the name of our app. It is clear, bold, and users will be able to remember it easily.

Icon:

6.1 - Justification for Splash Screen

Splash screens are used by some large applications to notify the user that the program is loading. People argue that splash screens enhance the look and feel of an application or web site, and provide a smooth loading experience by creating a positive distraction for users while assets are being loaded.

However, since splash screens often increase the wait for the desired content and may take a long time to load, users experience may be adversely impacted. The recommended practice is to allow user to interact with the default home page immediately, which is why Whatsapp and Telegram do not have splash screen when user launches them.

Newfangled also written* how the splash screen has caused at least 25% of their user to bounce upon reaching splash screen. This shows how the waiting time of the splash screen can deter people from using the application.

In our PWA context, all required assets are already downloaded when they choose to install the web app onto their mobile device; there’s no additional load time at all. This makes the splash screen unnecessary. Any additional assets are downloaded in the background as they continue to browse the application.

However, Chromium does not offer the option to turn off splash screen as of now so we are forced to include the splash screen as part of our PWA manifest. The splash screen in our app is rather short lived. We include the screenshot for reference.

*https://www.newfangled.com/agency-website-gaffe-2-the-splash-page/

Splash Screens:

We also included iphone devices splash but only those with specific dimension will see the changes.

Milestone 7: 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. Choose one of the CSS methodologies (or others if you know of them) and implement it in your application. Justify your choice of methodology.

7.0 - CSS Methodology: CSS Modules

Since we use ReactJS as frontend, we choose to organize our files as CSS Modules* (require react-script 2.0.0-next+). A CSS Module is a CSS file in which all class names and animation names are scoped locally by default. This feature is available as babel plugin (previously exists as separate package, now deprecated).

Firstly, we chose this CSS methodology as we wanted to create a one-to-one correspondence between our React Component and CSS, which makes maintaining styles much easier. In order to edit the styles of one component, we just need to refer to corresponding component in the css folder. There is no need to search through the whole styles file to find the styles that apply to the component we want to edit, increasing the efficiency of making style changes.

Furthermore, these CSS modules are scoped within their own component and thus do not take effect globally compared to traditional css unless explicitly marked as global (See Interoperable CSS^). Besides having an easier time naming classes, it also eliminates the need for BEM naming convention, which can get very long at times. We can reuse a class name in other components without worrying about whether other components will be affected since the styles are scoped locally. In other words, CSS Modules do not have specificity issues.

Lastly, CSS modules can work with both .css or .scss which ReactJS also supports.

*https://github.com/gajus/babel-plugin-react-css-modules
^https://github.com/css-modules/icss

Milestone 8: Set up HTTPS for your application, and also redirect users to the https://version if the user tries to access your site via http://. HTTPS doesn't automatically make your end-to-end communication secure. List 3 best practices for adopting HTTPS for your application. Explain the term "certificate pinning" and discuss the pros and cons of adopting it, as well as justify your choice whether or not to use it in your app.

Our application makes use of Google App Engine to deploy our API server. Our frontend is hosted on Firebase. Both are configured to automatically redirect to HTTP to HTTPS by default. Assuming it is an Apache server, a URL rewrite url will make this happen.

These two platforms are also signed using Let's Encrypt. However, as mentioned it is not secured by default. The typical approach to break the protocol is to perform a HTTPS downgrade attack, man in the middle, or even bypass with cookie hijack.

8.0 - 3 Best Practices For Adopting HTTPS

One practice is to include the HSTS protocol to tell browser strictly use HTTPS by specifying Strict-Transport-Security in the HTTP header. Subsequently, the browser will block any attempt by attempt to access the web app using HTTP entirely.

According to RFC7525 Section 4.1*, it is recommended that the SSL/TLS should never negotiate using RC4 cryptography suite as cipher due to their vulnerabilities. It is also recommended that using 256-bit ciphers will remain strong until the next fundamental technology breakthrough. Additionally, 128-bit ciphers will remain strong for next several years. This is the default length used in our Let's Encrypt certificate.

Last but not least, we should choose a reputable and reliable CA to sign our certificate because it has huge impact to the security if the CA leaked them accidentally. One such example is CA DigiNotar^ leaked more than 500 certificates and that incident causes major browser blacklist the CA and subsequently declared bankrupt in 2011.

*https://tools.ietf.org/html/rfc7525#section-4.1
^https://www.wired.com/2011/09/diginotar-bankruptcy/

8.1 - Certificate Pinning

Background:
Security is of paramount importance when communicating over a public network. The information sent and received should be secure and hackers should not be able to gain access to them. SSL/TLS protocol is greatly used to secure communications. It uses digital certificates to authenticate and encrypt data. These certificates must be digitally signed by a root certificate belonging to a trusted certificate authority (CA). Many operating systems as well as browsers

have a list of trusted CA root certificates in order to verify these certificates that the CAs have issued and signed.

One big problem is that protocols that rely on certificate chain verification are susceptible to dangerous attacks like man-in-the-middle attacks. These attacks occur when an unauthorized party can view and change the traffic passing between the mobile device and the backend systems.

What is Certificate Pinning:
Certificate pinning has emerged to counter these man-in-the-middle attacks. It is the process of associating a host with its expected public key. By owning both the server-side code and the client-side code, the client code can be configured to accept only a specific certificate for the domain name of the application rather than any certificate corresponding to a trusted CA root certificate recognised by the operating system or browser.

The application holds a copy of the certificate. When the SSL handshake is performed, IBM MobileFirst Platform Foundation for iOS client SDK verifies that the public key of the server certificate matches the public key of the certificate that is stored in the application.
If the pinning is successful, the public key inside the provided certificate is used to verify the integrity of the MobileFirst Server Certificate during the secured request SSL/TLS handshake. If the pinning fails, all SSL/TLS requests to the server are rejected by the client application.

Pros and Cons of Certificate Pinning:

Pros	**Cons**
1. Prevent man-in-the-middle attacks Authenticity is ensured as we do not need to rely on the CA to verify the website operator's public key. Hence this protects against man-in-the-middle attacks on the network which would compromise confidentiality.	**1. Lost/ mismatch of identities** Each certificate pins is associated with a unique cryptographic identity, which is public key or certificate, that the application must have to be verified. If these identities are lost or they are past the validity period without the pins being renewed, then no clients will be able to verify it and the application will not be functional. Hence to overcome this, every valid HPKP configuration has to include at least one backup key. Also, it can be troublesome when revoking or renewing the certificate. If the certificate is revoked and a new one is gotten for the domain, the key will not match what web browsers already have pinned.
2. Hard to bypass	**2. DoS attacks**

Bypassing certificate pinning is much more complex than man-in-the-middle attacks that can be prevented by certificate pinning. In order to bypass certificate pinning, the attacker would need physical access to the targeted device. Then, the attacker would need to root or jailbreak the device and modify the functions performing certificate pinning during runtime.	One of the risks of HPKP is that an attacker to purposefully deploy HPKP as a way to DoS your website.

Why we do not use Certificate Pinning:

Even though Certificate Pinning makes the application very secure and guards against man-in-the-middle attacks, using it is not easy. We have to consider many questions such as how many backup keys to pin, which part of the certificate chain to pin and others. One wrong decision can compromise on security. For example, we can pin to our own leaf keys and eliminate the risk of mis-issuance. However we will be risking the availability of your website against a few public keys that we need to protect and make sure they are updated all the time. Smashing Magazine was inaccessible for four days* when they renewed their certificate using a new key which had not been pinned.

Moreover, we feel that Certificate Pinning is not necessary in our case. CA compromise and unauthorised issuance of certificates are not very applicable to us. A compromised CA can be detected quickly. It is likely that only very popular sites would find Certificate Pinning to be useful, in order to guard against targeted attack. For us, the risks and work required to do Certificate Pinning would outweigh the security benefits that Certificate Pinning can bring for our application.

Hence, we decided not to use Certificate Pinning.

*https://www.smashingmagazine.com/be-afraid-of-public-key-pinning/

Milestone 9: Implement and briefly describe the offline functionality of your application. Explain why the offline functionality of your application fits users' expectations. Implement and explain how you will keep your client synchronised with the server if your application is being used offline. Elaborate on the cases you have taken into consideration and how they will be handled.

9.0 - Offline Functionality

The offline functionality allows our users to access our application anytime and anywhere even without internet connection. That could be often the scenario when travellers are in a foreign country, where internet connection is not as conveniently available as that is their home country. Additionally, the offline functions helps to serve the application to the users at a much faster speed, presenting persisted information stored while we fetch for new data from the server. Therefore, the app provides a faster and better experience to the users with the offline functionalities.

For our React application, we use Redux to store and manage the application states. When the user first launched the application and navigates around the application with internet connection, the data fetched from the server and the application states are stored in a single source store in Redux. We then persist the redux store to disk in order to be able to render meaningful conteen even when the user opens the application offline.

On the other hand, the Progressive Web Application service worker is responsible of caching relevant assets needed for the application to work offline.

As of the current stage of development, users can browse all the application content that has been fetched through network after they lose internet connection. That is, Read-resilience. However, we have yet to implement offline write functions to make the app write-resilient. The idea is to store all network-bound actions in a queue to be performed later. At the same time, we can do optimistic and pessimistic application state update to the Redux store.

Optimistic updates are to be implemented on simple actions such as:

1. Follow/Unfollow a user
    a. Simply update the followers count optimistically
2. Comment on the post or edit/delete comment
    a. Optimistically update the comment under the post

Offline application states are changed immediately with these action calls, as they will mostly likely be successful. The server will be updated once the internet connection is back.

Pessimistic updates are to be implements to actions such as:

1. Edit a post
    a. We cannot just optimistically update the state at the user side as it may give a false impression that the item information has been changed

2. Edit user profile
    a. As it may affect new sign-in, the user has to know whether the update request has been successful
3. Offer to help
    a. As there is constraint on the number of people who can offer to help, it is not guaranteed whether the user's offer is successfully accepted

These functions require a pessimistic update of the application state as there is a high chance that the request does not go through. Therefore, we plan to show "submitting" status for these functions and the user would know the actions are yet to be completed.

For both kinds of updates, we need a rollback function of every specific actions so that it handles the case when the request permanently fails even with a network connection. In such cases, the application state should roll back to conform that of the data from the server.

Milestone 10: Compare the advantages and disadvantages of token-based authentication against session-based authentication. Justify why your choice of authentication scheme is the best for your application.

Token-based authentication

- Server creates token with a secret, sends token to client
- Client stores token and passes it along in subsequent requests
- User's state is stored in token on client side
- More secure: signed with secret, can blacklist/revoke token if tampering detected
- Vulnerable to XSS, but easier to deal with XSS than CSRF

Session-based authentication

- Server creates session for client
- Session id is stored as cookie on client (browser), sent along in subsequent requests
- Server uses session id to compare against session information stored in memory
- User's state is stored in memory on server side
- Vulnerable to CSRF

We chose token-based authentication because:

- Token is stored on client side, instead of server's memory. This increases scalability
- Makes server stateless, adhering to RESTful api principles
- Should we choose to implement a native mobile platform in future, we will not have access to browser for cookies
- Better security (Easier to deal with XSS than CSRF)
- JWTs are generally accepted to be the more modern approach as compared to traditional session-based authentication, adheres to modern best practises

Milestone 11: Justify your choice of framework/library by comparing it against others. Explain why the one you have chosen best fulfils your needs. Lastly, list down some (at least 5) of the mobile site design principles and which pages/screens demonstrate them.

11.0 - Choice of Framework

ReactJS:

- Can be extended to ReactNative easily in future if we want to support Native IOS or Android
- Less steep learning curve since both server and client implement in javascript
    - VueJS is easy to learn as well, but since our team is more familiar with ReactJS we decided to use the latter instead.
- Traditional DOM vs Virtual DOM rendering performance
    - Traditional DOM requires re-rendering of the entire page if one of the node changes, resulting in a larger overhead for Jquery manipulating the DOM.
    - ReactJS has virtual DOM which handles changes with different algorithm so that it only re-renders the changed node.
- It is PWA ready with create-react-app CLI
    - registerServiceWorker.js is provided with the create-react-app cli.

Semantic UI

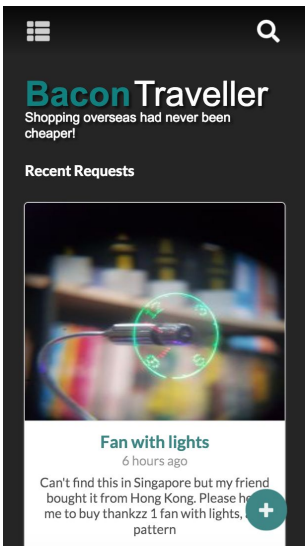
We also make use of Semantic UI to draft our app skeleton and then apply our own color and branding on top of it. We refer to Google's Material Design Documentation for some elements such as Floating Action Button (FAB) which displays different options based on the content being viewed at the moment. This offers more space for the main content compared to a static navigation bar. The FAB is also fixed to the position. It allows for better UX since user do not have to scroll all the way to the top to take action.

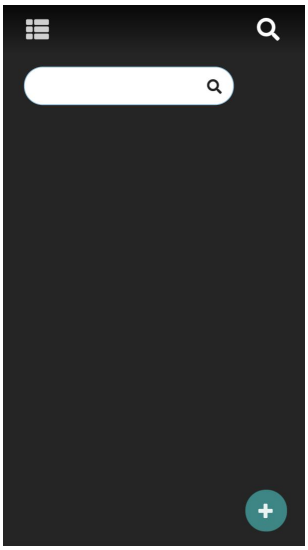
Mobile Site Design Principles:

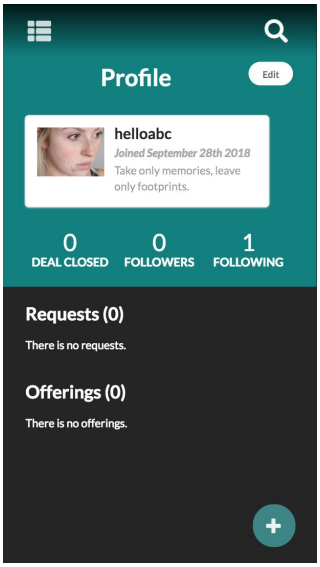
Mobile Site Design Principles	**Screens/ Pages**

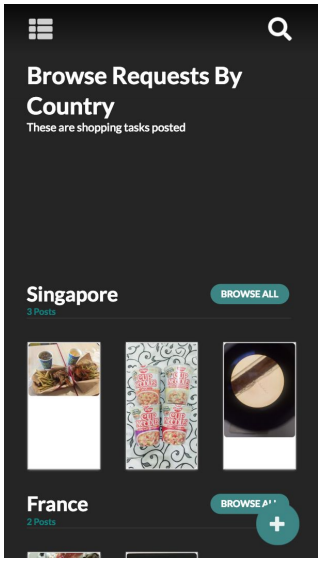
1. Make Site Search Visible
The search button is visible and obvious. Users can search the posts and followers using the search button that is present in the header of most pages. This makes finding specific information more efficient and provides a better user experience.

Pages: home, search, profile, country-listing



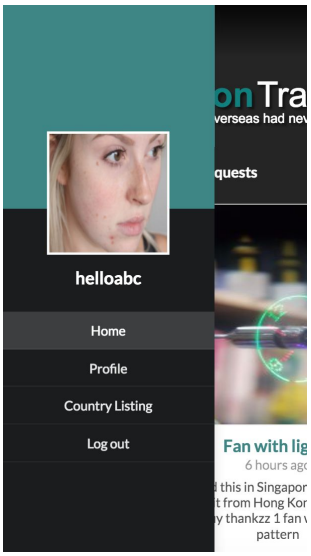






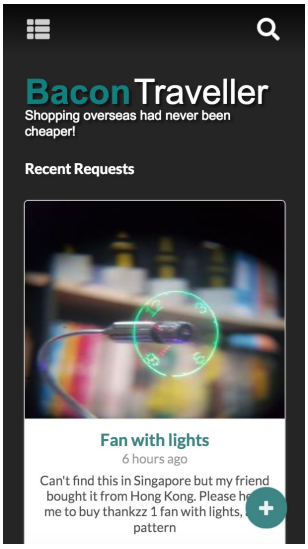
2. Short and Useful Menus
Our menu only has the most important links such as profile page and country listings. This way, users don't have to scroll through many options. The menu links are ordered based on value to users, with home being the most important since users can see all the requests posted.

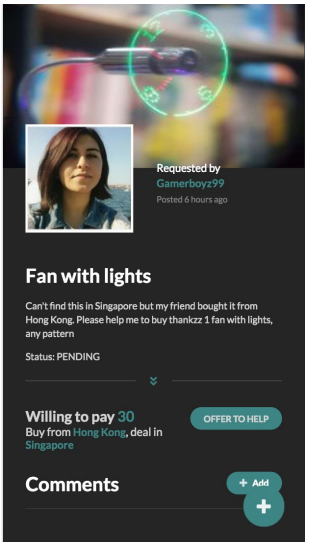
Pages: side-bar

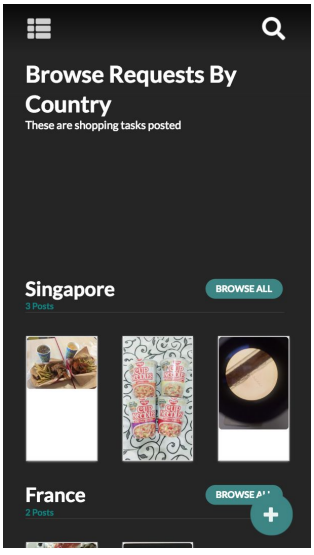


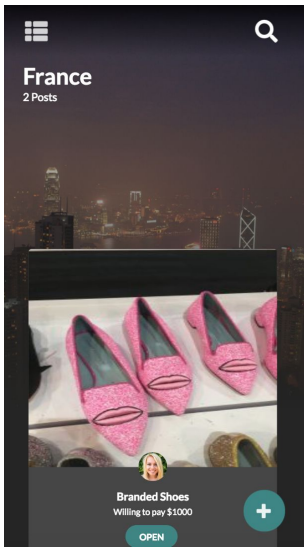
3. Makes Call-to-action button user friendly
Our Call to action buttons are big and attention catching. For example, our floating action button is fixed at the bottom right page where users can easily see. It is in a teal colour that contrasts well with the background. This catches the user attention, encouraging them to post requests.

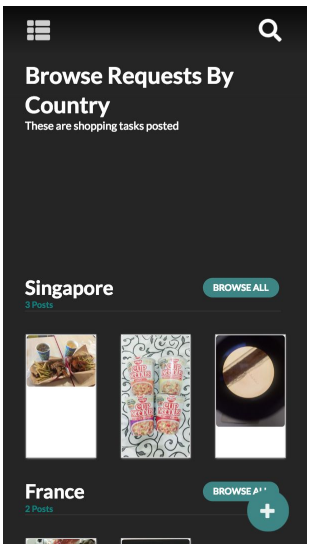
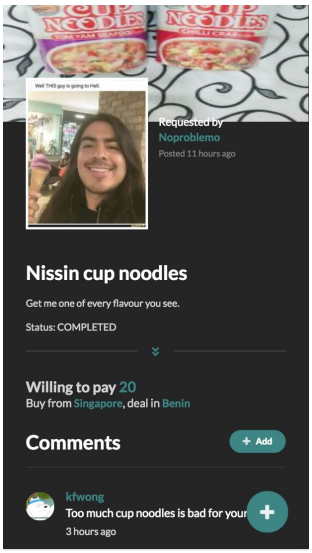
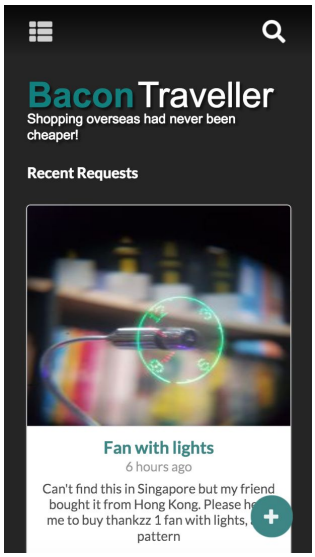
Pages: home, country-listing, item-detail, item-listing-in-country

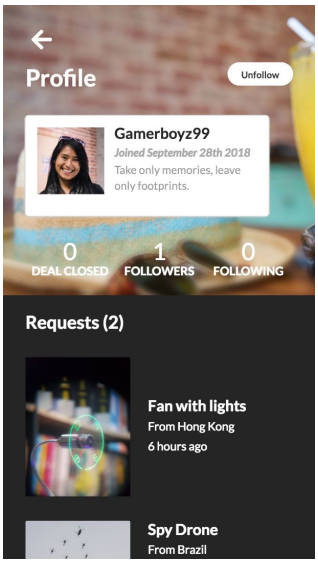


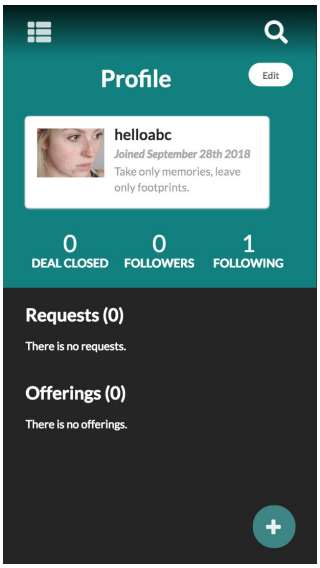


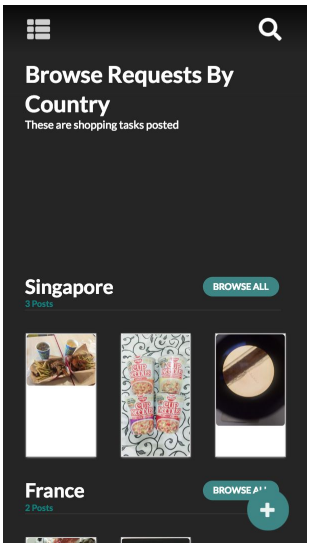




4. Let user explore before commit When the user first enters the site or does not have an account with us, they can still browse through the country listings. We do not want to force them to register too early as it can cause them to abandon the site. Instead, we let them browse through the content and get a sense of what we can offer them. Pages: country-listing	
1. Responsive Optimization Our site is optimised for mobile. On a mobile device, our application is not cramped; there is not too much content on the pages. We employ single column site layout for clarity and simplicity. The words on our application is optimised for mobile such that they increase in size so that the mobile user can read them easily. This prevents users from getting frustrated when they need to zoom in to read the text. Neither is our application difficult to navigate on small screens. Our buttons are big and easy to click, and the links are obvious. Pages: all pages, (shown pages are home, profile, country-listing, detail)	 


Profile
Unfollow
Gamerboyz99
Joined September 28th 2018
Take only memories, leave only footprints.
0
DEAL CLOSED
1
FOLLOWERS
0
FOLLOWING
Requests (2)
Fan with lights
From Hong Kong
6 hours ago
Spy Drone
From Brazil


Profile
Edit
helloabc
Joined September 28th 2018
Take only memories, leave only footprints.
0
DEAL CLOSED
0
FOLLOWERS
1
FOLLOWING
Requests (0)
There is no requests.
Offerings (0)
There is no offerings.


Browse Requests By Country
These are shopping tasks posted
Singapore
3 Posts
BROWSE ALL
France
2 Posts

Milestone 12: Describe 3 common workflows within your application. Explain why those workflows were chosen over alternatives with regards to improving the user's overall experience with your application.

12.0 - 3 Most Common User Flows

These three workflows are chosen as they go in line with our application objectives. Firstly, buyers can post items that they want to buy from overseas, in hopes that a willing traveller will accept them. Secondly, travellers can accept requests by buyers if they are willing to help them and earn extra cash. Lastly, users can browse through the app content, especially the requests, to get a feel of which products are more popular and which travellers are experienced. This helps them with their decision making process, should they want to buy a product or help others fulfill their requests.

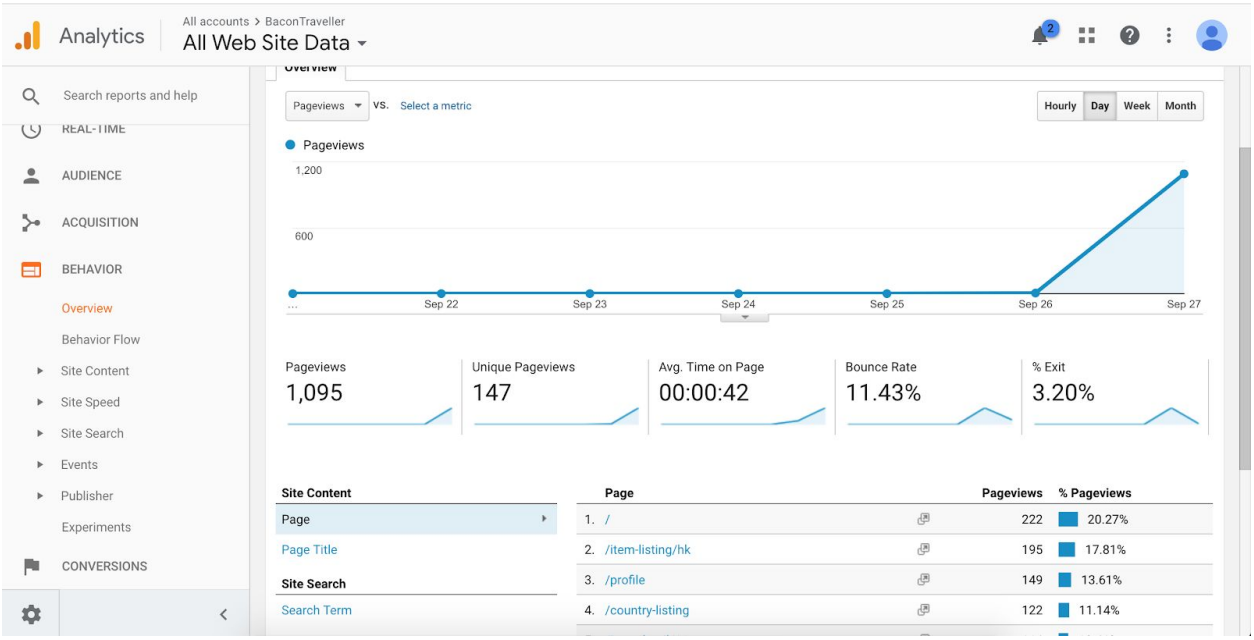
Use Case UC1: Buy Product	
Identifier	UC1
Description	Buyer wants to buy something from overseas
Actors	Initiated by Buyer Communicates with Traveller
Preconditions	The Buyer and Traveller are logged in.
Flow of events	1. Buyer can browse other postings from other buyers/sellers to see what the usual offer price is. 2. Buyer creates a posting of an item he/she wants to buy by clicking on the floating '+' button. 3. Buyer is brought to a form. Minimum fields required: Item name, description, country to buy from, country to deal in, offer price, image of item. 4. After making an offer, Buyer waits until he/she gets notified that a Traveller accepted his/her request. 5. Traveller can communicate with Buyer outside the app if the posting is unclear. 6. Payment is settled by the Buyer and Traveller, outside of the application. 7. Delivery is settled by the Buyer and Traveller, outside of the application. Buyer receives his/her items. Transaction is closed. 8. Buyer and Traveller rate each other after transaction ends.
Post conditions	Transaction is closed, and status of post is 'completed'.

Use Case UC2: Accept Request to Buy Product	
Identifier	UC2
Description	Traveller wants to help Buyer buy a product
Actors	Initiated by Traveller Communicates with Buyer
Preconditions	The Buyer and Traveller are logged in.
Flow of events	1. Traveller browses through all the posts and finds one that he/she likes. 2. Traveller clicks on offer to help button on the post. 3. Traveller accepts the request. 4. Traveller can communicate with Buyer outside the app if the posting is unclear. 5. Payment is settled by the Buyer and Traveller, outside of the application. 6. After payment is confirmed, Traveller can buy the item. Delivery is settled by the Buyer and Traveller, outside of the application. Buyer receives his/her items. Transaction is closed. 7. Buyer and Traveller rate each other after transaction ends.
Post conditions	Transaction is closed, and status of post is ‘completed’.

Use Case UC3: Browse Requests	
Identifier	UC3
Description	User (Traveller/ Buyer) looks through requests to see what products are being bought/sold on the platform, and who is selling them
Actors	User
Preconditions	User is logged in.
Flow of events	1. User browses through requests on the posts page. 2. User can filter the requests by country, dates or by using the search tool (keywords). 3. User clicks on the request to view details about each individual request. 4. User can click on the post’s author link.

	5. User sees the deals closed, number of followers, number of followers user is following of post's author
Post conditions	-

Milestone 13: 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 14: Achieve a score of at least 90 for the Progressive Web App category and include the Lighthouse html report in your repository.

Milestone 15: 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)

15.0 - Social Network: Facebook

For our prototype, we used Facebook login to allow users to bypass the signup process. In extension, we can implement Facebook Social Plugins like the ‘Save’ or ‘Like” plugins. This will help us gain the traction of users and utilize the network effect to grow our application.

In particular, the ‘Save’ plugin will allow users to save items that they are interested in buying, to a list. The ‘Like’ plugin allows users to share what they bought or sold on the platform with their friends, who might be interested in the same items.