Major Project Report

Course: Interactive Multimedia Design

Year: 4th Year

Name: Edward Price

Student Number: B00565433

Mentor: Leo Galway

Project Name: Spotnote

Acknowledgements.

I would like to firstly thank Leo Galway for his tremendous help throughout the whole final year, including report advice, coding problems and IOS development licenses and profiles.

George Moore, Liam Burns and Raymond Bond for their guidance throughout the project and problem solving input.

Peter Nichol for his willingness to help on any matter and stand up for all student problems.

Paul McCormack and Tim Potter for their fantastic design inspiration and advice.

All the PhD students who were able to answer questions regarding certain architectures.

I would also like to thank stack overflow and all the great people on it for their incredible help to any problem I ran into.

Sorry to anyone who has been left out but you help hasn’t gone unnoticed and everyone’s input has been a major help throughout the past year.

Table of Contents

1. Introduction

This report will cover the overall process of design and implementation of the IPhone application Spotnote. It is a native app that is used for reminder-based locations.

Spotnote is the name of the iPhone app that is used to set up reminders that can be triggered by arriving at a location. The reminders can also be set for a specific time, or even a combination of both. So for example if you arrive at Jordanstown University in the morning, you can have a reminder set up to tell you that a fry in the students union is only £2.99. It's not a revolutionary idea that is going to change how the world works, but it will be a slightly more advanced way to remind yourself about things you would sometimes forget.

The idea came about because of how busy and hectic life can be, so it's not always easy to remember the right things at the right time. The native reminders app that comes with IOS does have the ability to remind at location but it is very confusing to use and isn't very well designed. The user interaction and workflow of the app is so confusing it just leaves you frustrated.

1.1 The Project Challenge

The main challenge of the project is to show the ability to create a native and fairly complicated app that uses a range of native features of the IPhone without having to actually code natively using objective-c. This is to show that someone who doesn't have the correct skills to code natively can still create a seamless looking app. Learning new coding languages can be hard and time consuming and it can be frustrating if you already have skills in JavaScript, HTML and CSS. It is better to be a master of one language than a jack-of-all-trades. It is a way of showing that a web designer with knowledge of web standard coding languages is still able to create an app that was previously thought to be out of reach for someone with only web design skills.

1.2 Overall aims and objectives

The aim is to create a native app that will have the following high-level functionality:

- Create a reminder based on a location

- Track the users location
- Check the users location against the stored location reminders
- Deliver a reminder to the user when if they are within a certain radius of a location

The following technologies have been identified to be used instead of native IOS programming:

- HTML
- CSS
- JAVASCRIPT
- JQUERY

1.3 Overview of work undertaken

The project will consist of the ability to create a reminder with the property of a location assigned to it. This reminder will have the ability to be edited and will be saved into a database for secure permanent storage. The reminder item will automatically notify the user of their position at the appropriate time. These ability's will all come from a native IOS application but all of the coding will be through standard web technologies such as HTML, CSS and JavaScript.

2. Background and Research

Before undertaking any sort of design coding or implementation at this stage it is important to take time to research similar products, available technologies and suitable APIs that could be used. Gathering appropriate information at this stage helps the project to run smoothly and address any potential roadblocks that could arise. Finding out potential problems at an early stage increases the ability to overcome them

2.1 Current Available Products

The app store is the place to start researching similar products and view other companies approach to building an app that has the same sort of functionality as this project. It will give a good gauge of what features are important and what user interaction workflow is the best method. Obviously the simpler the better.

2.1.1 Native IOS Reminders

This is Apples solution to the location-based reminders. It is one of the main reasons for this project because of how badly it is designed and functions. Firstly if you have multiple email address assigned on your phone then it will automatically create a new list for each of these that cant be deleted. It also creates another separate list for your calendar. Not giving the user the ability to properly manage this list is frustrating. Creating a new list this is synced with your email address means you cant apply a location to it. Locations can only be added to lists that are synced with the phones ICloud account. All of the UI is confusing and is hard to navigate through and simply frustrating and annoying to use. There isn’t even a map to select locations on. Locations have to be searched for and selected from a list of potential options that aren’t always accurate. Put simply this is one of the worst apps that apple has designed and it is widely documented that its poor functionality leads to a very low percentage of usage.

2.1.2 LocationMinder - Location Based Reminders

This is an app that is available on the IOS app store (LocationMinder 2014). It is one of the most simplistic and easy to use reminder apps available. The user interface is simple to understand as there is either a map with all of the current locations plotted on it or else a list view of them. This map view is a very nice feature and is something that will be implemented in the project. Apart from these two screens and a form to create a reminder that is all it really consists of. This is very basic but it is a good approach to take so that a user will quickly understand the app. The radius around a point can be changed which is another nice feature to give a user control over, and again it should be considered for this project. The only downfall of the app is that its £0.69. Although this doesn't seem like much, it is really quite a lot for such a basic app that doesn't have many additional features

2.1.3 Checkmark 2

Checkmark 2 is the second app that has been released by Snowman (Checkmark 2 2014). It is one of the most advanced location based reminder applications with a lot of advanced features including lists, groups for locations, custom notification radius, iCloud backup and a really nice IOS7 optimised user interface. The only issue is that to get all of these features you have to pay a pricy £2.99. This is just the cost for a well designed and richly functional app. Spotnote will hopefully be the bridge between cost and functionality. If released on the app store it will be a free app and will hopefully have many of the intricate features of Checkmark 2

2.2 Research into Technology.

Next stage of research is to look into the technologies that can be used to create a location based reminder app. This will include tracking location through GPS, GPRS and 3G, mapping location, alerts and notifications, storing data securely on and IOS mobile device. Technology research will additionally take into account the use of hardware and software that is needed to create an IOS app and any software's that can improve workflow or reduce time.

2.2.1 Editors

One of the most crucial parts that can sometimes be overlooked in a project like this is the use of editors. Choosing the right editor with all of the needed functionality can save time in the long run. For instance choosing an editor with integrated FTP capabilities will reduce the time in having to use two separate applications.

Sublime Text 2

This is a very simple editor with a neat zoomed out view of code for quick navigation. It's lightweight and fast but does lack a lot of features that would be good to have. There are a lot of add-ons created by lots of people but managing all of that type of thing can get annoying an confusing. Version 3 will soon be released so maybe this next version will have all of the required features to make it a powerful enough editor to do all the needed commands for a large complex project such as this

Notepad ++

This is a very popular editor and has a lot of great features and might have been the selected editor but unfortunately it isn't available for the Macintosh. The Mac will be used for development because of the need for Xcode, which is required for native IOS development. This will be discussed in more detail later on.

Coda 2

This is the application that will be used to do most of the coding and development for this project. It is very rich in features including code snippets, inbuilt web preview, project management and ftp capabilities to name a few. It is very versatile and quick and can be used for all aspects of the assignment.

2.2.2 PhoneGap

PhoneGap is a free open source framework that allows the use of web technologies and web API's to build native phone apps (PhoneGap 2014). Not just for IOS but also for Android and Windows Phones. There are other such frameworks like this such as Appcelerator and NimbleKit but none stay as close to web development as PhoneGap does, and some of them require either a one off fee or else a monthly subscription which isn't realistic on a project like this with no real budget. The main thing that PhoneGap will do is bundle up the web app that you create into a web view within a native app. From this web view native calls can be made to device specific technologies such as geolocation and notifications through simple calls.

2.2.3 Database Options

At this early stage the decision was already made to use store the data locally on the device rather than in a remote database because this will reduce the amount of Ajax calls which might cause performance issue on a weak mobile signal. PhoneGap supports all the following researched web storage options.

Local Storage

Also known as web storage, simple storage, this API provides synchronous key/value pair storage, and is available in underlying Web View implementations.

Indexed DB

Indexed DB is a web browser standard interface for a local database of records holding simple values and hierarchical objects. It is now the supported web standard for local storage but unfortunately Safari in IOS doesn't yet support this method so even. This is rather disappointing but wont hold anything back

WebSQL

Web SQL Databases is a spec that brings SQL to the client side. It isn't part of the HTML5 specification any more as it is solely based on SQLite but for it to be web standard there needs to be multiple independent implementations. Internet Explorer and Firefox don't support it but this isn't a problem as these browsers wont be used. Its more advanced than the key value pair of Local Storage and is still supported by the target browser so it is the chosen method. In the future if support is dropped for it then the database can be easily ported to Indexed DB.

2.2.4 JQuery and JavaScript

JavaScript and JQuery are the main languages that will be used for manipulation of dom elements, tree traversal, data collection and manipulation and form validation (JQuery 2014). Pretty much all functionality will have to be done through JavaScript. When it comes to things like animations JQuery will not be used, this is because of the limitations on a mobile device. JQuery animations can be slow, jumpy and unresponsive and would make the app feel like it wasn't native which is one of the main aims. To combat this any animations will be done via CSS transforms and translates because these are hardware accelerated on the IPhone.

When building for the web it is recommended to link to a remote version of JQuery that is usually hosted on Google's servers and if for any reason it cannot be retrieved then fallback to a local version. The problem with using this method is that it could take a long time to retrieve the library on a weak phone signal, which would decrease the usability. Speed is a crucial thing when it comes to mobile apps as users expect everything to happen instantly so there is no point in forcing them to wait for a remote version when a local version will do exactly same thing without potential delays.

2.2.5 Framework Templates

When it comes to designing the visuals of the app and the user interface then using a base template to start with is always a great help. It never dictates the design or influences how it looks, it only increases the speed at which a design can be effectively implemented. For that reason multiple different started frameworks for mobile development where researched to get the best useful functionality as well as minimal file size. Keeping file size down is a very crucial factor that needs to be constantly thought about throughout the project. Using a large library just because it helps with one tiny feature is the wrong approach as it probably wont help the user experience that much and will only hinder the response time when using the app.

Framework7

Framework7 - is a free and open source HTML mobile framework to develop hybrid mobile apps or web apps with IOS 7 native look and feel (Framework7 2014). It is only for IOS 7 so therefore doesn’t have any extra features for other mobile platforms that wont go unused and take up file space. It tries to cover a lot of native functionality that is seen on the iPhone and uses CSS transitional properties for animation, which improves performance. When testing out the framework it was found to have a few bugs because it had only recently reach version 1.0. it would have been chosen for the project if there wasn’t these bugs in it.

Sencha Touch

Sencha Touch, a high-performance HTML5 mobile application framework (Sencha 2014). It also enables developers to build apps that work on IOS, Android, BlackBerry and Windows Phone. This might be very useful on another project that required cross device development but in this assignment only IOS is needed so the code for the other platforms will go to waste and simply take up file space. Sencha is one of the most feature rich frameworks but that isn’t necessarily a good thing. Not all of the features are needed so again it will just take up file space. It is one of the biggest libraries available, which in this case isn’t a good thing. Its bloated size lets it down as the chosen framework is only needed as a starting point.

ChocolateChip-UI

ChocolateChip-UI is a framework for making mobile Web apps. It uses the three main components of web design; semantic HTML5 markup, CSS and JavaScript (ChocolateChip-UI 2014).

It has very easily customized CSS themes. With ChocolateChip-UI you only write one app with its layouts and interactions, and then you provide the correct CSS theme for the target operating system. This cuts down on unused code for different operating systems. File size is very small for this framework and it doesn't have lots of unnecessary extra features that wont be used. All aspects of it are very easily customizable and for this reason it is the chosen framework to go forward with.

2.2.6 IOS Specific Development

Developing for the IOS platform it isn't just a case of creating an app and then putting it on an iPhone. There are a few steps that you have to go through to before this can be done

Developer License

To be able to develop for IOS, build on your personal device and then put the app on the app store you have to acquire a developer license, which costs $99. Because this project has no budget the university was able to provide access to their development program for free. This meant access to all required documentation, features and forums.

Provisioning Profile

This is a file that allows applications in development to be installed on an IOS-based device. To test properly this is a crucial part, which is accessed through the developer license.

Macintosh and Xcode

All development will have to be done on a Mac because of the need for Xcode (Xcode 2014). This is the program that will be used to build onto an iPhone using the developer profile. The whole process for creation an IOS app is quite complicated but this is just because Apple is very protective over their operating system and don't want people to be able to easily create and deploy apps without their permission.

3. Concept Definition

The idea came about because of how easy it is to forget important things and to help to stay organized. The native reminders app that comes with iOS does have the ability to remind at location but it is very confusing to use and isn’t very well designed, and any good apps on the app store are all paid.

Spotnote will be a simple easy to navigate app that uses location to trigger specific reminders as well as the traditional time based reminders.

3.1 Requirements Survey.

The purpose of a survey is to get the opinion of others rather than just the opinion of an individual. This will prevent a biased opinion when defining features that will be included in the final product. The users opinion may different completely from your own but if enough people express a certain opinion then it should be recognized

A survey was devised to gauge the importance of proposed features within the app, and participants were asked to rate the importance of each statement from one to five. The results of this survey can be seen in figure 3.1

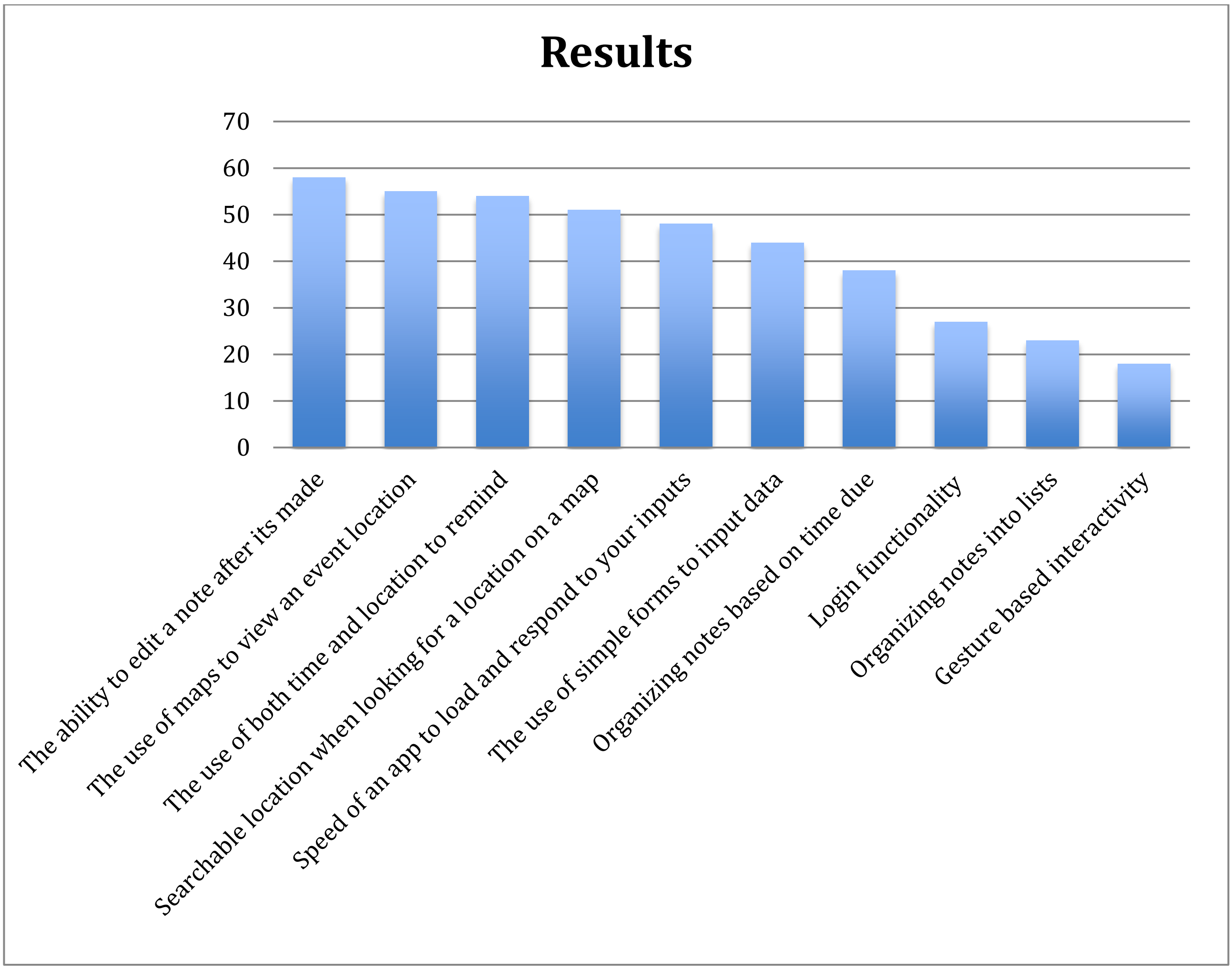


Fig. 3.1 Survey Results

Twelve people were asked to take part and rate the statements importance. Once completed the results were totaled and graphed which can be seen in figure 1. The graph gives a good indication of the least and most important features. These results will be used to help define the requirements and the MoSCoW rating system of features. So instead of trying to do everything, features will be added in order of importance.

From the results it can be concluded that the current proposed features in order of importance are:

1. The ability to edit a note after its made
2. The use of maps to view an event location
3. The use of both time and location to remind
4. Searchable location when looking for a location on a map
5. Speed of an app to load and respond to your inputs
6. The use of simple forms to input data

7. Organizing notes based on time due
8. Login functionality
9. Organizing notes into lists
10. Gesture based interactivity

The survey that was given to participants can be seen in appendix A.

3.2 Requirements specification

The purpose of this requirements documentation is to cover all of my systems features and functions. It will give me a good oversight into what exactly needs to be built designed and coded. For my location based reminder app I need to figure out all the aspects that I need to include into my design and build process so this document will help me get a good visualization of it and prevent me overlooking crucial parts of the project.

3.2.1 Volere Snow Cards

To start of with a high level approach was taken when defining the basic functionality of the app. To do this Volere Snow cards were used (Volere 2014). This method makes it easy to reference a feature, give it a purpose as well as map out what features depend on each other. It is good to detail only the high level features first so that everything is covered and note lost in defining the detail. Requirements can be fleshed out once the basics have been laid out.

Below is example of the Volere cards data for the Map View feature:

Requirement Number: 4
Title: Map view of reminders
Description: A map that has all of your reminders on it as pin points as well as your current location. This means you can see where the closest reminder is and go and do it. Each pinpoint on the map can be pressed to show all the details of the reminder.
Rationale: Because the main focus of my app is to have reminders at specific locations, I think it would be good to have a map that shows you these locations so you can easily visualize what you have to do and where.
Fit Criteria: A map with points on it at each reminder location
Dependences: #2 (Create/edit reminders)
Requirement Type: Functional

There were six main high level functions defined at the Volere stage:

- Login
- Create/edit reminders
- Manage reminders (Check off or delete)
- Map view of reminders
- Timeline view of reminders
- Website

A full list of the Volere snow cards can be seen in section B of the appendix.

These features are in no specific order and their importance will be defined after all details have been though of. Even before evaluating each point further it is clear that some features wont have a very high importance.

3.2.2 Detailed Requirements

Once the high level requirements had been mapped out using Volere then it was time to outline more specific detail of the functionality which can then be used to guide the development stages of the project. Going into as much detail as possible at this stage will reduce any oversights. Table 3.1 is an example of one of the detailed requirements sections.

Table 3.1

2. Create/ Edit Reminders		
Reference Number	Title	Description
2.1	Create	To create a reminder there will be a simple button on the landing screen that will take you to a new form. This button will probably be in a consistent position on all screens in the app. The form that it brings you to will have three different fields to fill in. 1. Firstly there will be the title of the reminder, which will appear in home screen along with all the other reminders. 2. Then there will be a time and date input. This will use the generic

		data and time input scroller that is used in IOS 3. Lastly there will be a location input. Once you click on this input I would like it to reveal a map. You can type a location in the field and search for it, which will show the location on the map. You can then move the marker If you want to another point or else search again You must fill in the title and either the location or data fields and then save the reminder. Once it is saved you are brought back to the main screen
2.1.1	Title input	An input field that takes the title value for a reminder. It is required.
2.1.2	Time input	This will only take a certain format of time and date, which is dependent on the iOS date/time picker. It is only required if there is no location entered
2.1.3	Location input	This will not be manually input by the user to avoid mistakes. It will be generated by a selection on a map. It is only required if there is no time inputted
2.2	Edit	User will click on the reminder they want to edit and will be brought to the same screen as the create reminder. But the input fields will be populated with the information of the item you have clicked. You can then edit any other the information and resave it as the same item so it doesn't create a new reminder.

To flesh out the requirements a table structure was used to lie out all the required data appropriately. The reference number will come in very useful near the end of the project as it can be used to test against each case to see if the functionality is present within the app. the description is the most important part for the upcoming build stage. It outlines exactly what and how to implement the feature. Some requirements for design have also been drawn up, as it is important to keep design in mind even at an early stage.

A full list of all the requirements can bee seen in section C of the appendix

3.2.3 MoSCoW Method of Requirements

To prevent from just blindly trying to accomplish all of the requirements the MoSCoW method is used to rank the importance of each feature. Knowing what features can potentially be dropped for technical reason or time constraints helps the development stage run smoother. If for any reason a roadblock occurs and time is lost then this MoSCoW chat can be referred to. Table 3.2 shows the full MoSCoW list of requirements.

Table 3.2

Reference	Title	Functional/Non Functional	Importance
1.1	Login	Functional	Could
1.2	Register	Functional	Could
1.3	Forgot Password	Functional	Could
1.4	Login with Facebook	Functional	Could
2.1	Create	Functional	Must
2.2	Edit	Functional	Must
3.1	Complete	Functional	Must
3.2	Delete	Functional	Must
3.3	Alert	Functional	Must
4.1	Map	Functional	Must
4.2	Pointer Options	Functional	Should
4.3	Edit	Functional	Must
5.1	TimeLine of reminders	Functional	Must
5.2	Manage Reminders	Functional	Must
6.1	Website	Functional	Could
7.1	Transparency	Non-functional	Should
7.2	Simplistic Icons	Non-functional	Should
7.3	Geometric shapes	Non-functional	Should
7.4	Sans-serif	Non-functional	Should
7.5	Flat design	Non-functional	Should
7.6	Negative space	Non-functional	Should
7.7	Borderless buttons	Non-functional	Should
7.8	Color palette	Non-functional	Should

3.3 Paper prototyping

Paper prototyping is not only for the visuals of the app but it is also to lay down the workflow and navigation. It must fully take into account all of the requirements and accurately portray them in a visual way as to give an indication of their uses.

Before starting to wireframe anything research was done into apples IOS7 UI design Basics (IOS7 Design Resources 2014). It is a great resource for learning how apple design all of their apps and how they lay everything out and emphasize importance onto certain parts of an app. After spending time reading about reading these guidelines paper prototyping was started. All full size images of paper prototyping can be found in appendix D.

So to start of with the flow of my app was mapped out. This includes the first page you would land on and what interactions you could do on each page. Below is a summary graphic of what was decided on. The list date and map view are just different ways of viewing your list of reminders and each will be explained in more detail. Figure 3.2 shows the site map for the app.

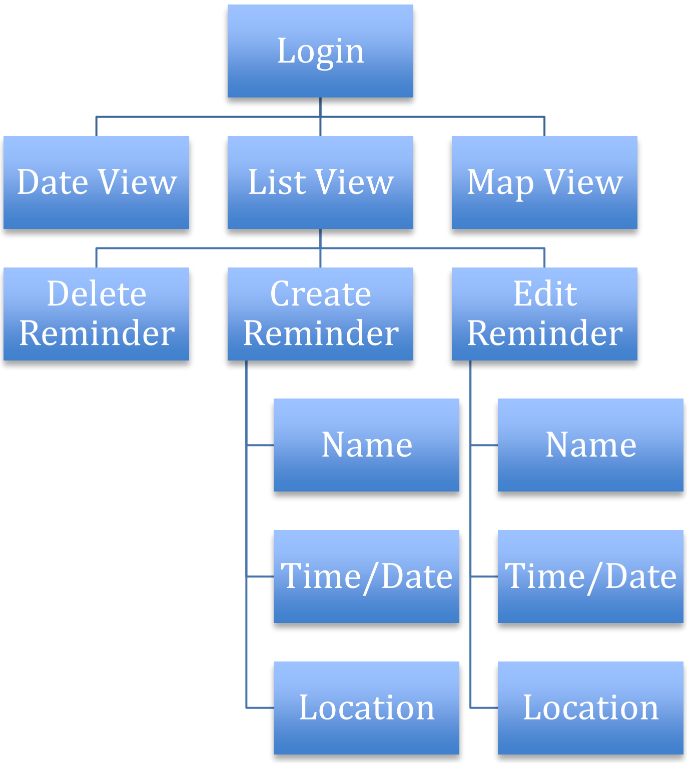
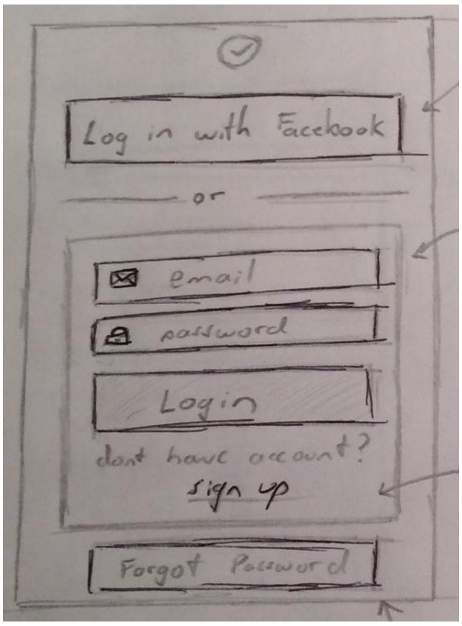


Fig. 3.2 Site Map

It's not really clear in the graphic but you will be able to create, edit and delete reminders from any of the three view screens.

3.3.1 Login Screen

So the first screen that was wire framed was obviously the login/sign up screen, which can be seen in figure 3 and 4. It is still unsure if this will actually make it into the app as it is only a 'should' but it is best to wireframe it anyway instead of leaving it out and having to go back to the wire framing stage. The only reason to have a login is so that the users data can be stored in a database and all their reminders will be linked to their login, but because it has already been decided that the data will be stored on the users device then it kind of makes a login redundant. It would only come in necessary if in future the reminder data was stored on an external database so that it could be accessed from multiple devices, but this is more of a future feature. At the current stage the login functionality is being kept in until it is clear that it is definitely not needed. Once a functional prototype is made then it will be clearer if it is needed.



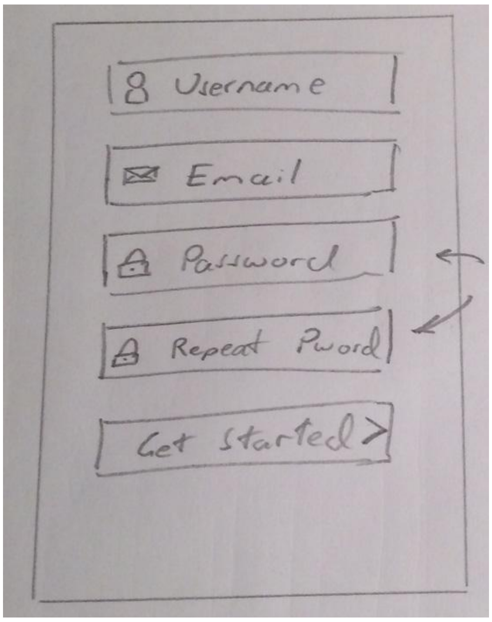


Fig. 3.3 Initial Login Screen, Fig. 3.4 Login Form

So the first thing that you see on the login screen is a button to log in or sign up with Facebook. This makes the whole login and signup process a lot easier for the user as they don't have to fill in a form. All they have to do is allow the app access to their profile information and it creates the account for them. There is then a form for people who already have an account to login. It just asks for email address and password.

If you click the sign up button you will be taken to the form below. It asks for username email address and password and once all these required fields are filled in you are taken into the app.

The forgot password button will ask for the users email address and then it will send a new password to that address if it is in the database. If not you will get an error message.

3.3.2 6up's (List View Screen)

Figure 5 is a picture of 6 wireframes that were made for one of the most important screens, the list view screen. To see the full size picture of this and all other pictures go to the appendix. Multiple different layouts were tried for navigation items as well as the reminders themselves. The edit functionality was also looked into, as it would be a good user experience to be able to do everything on the same page.

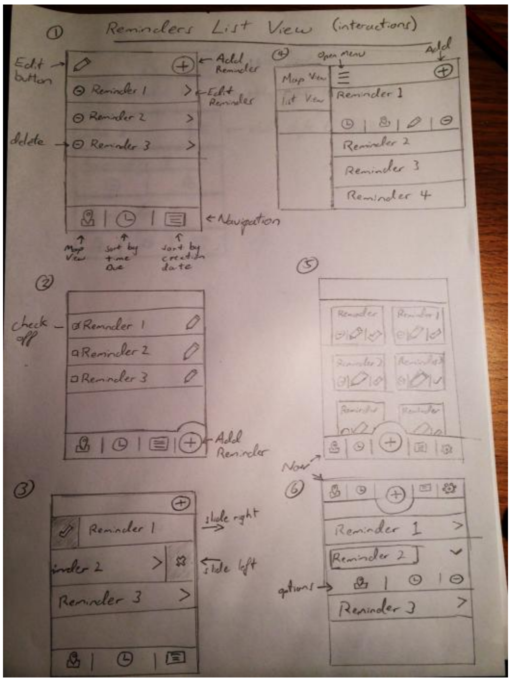
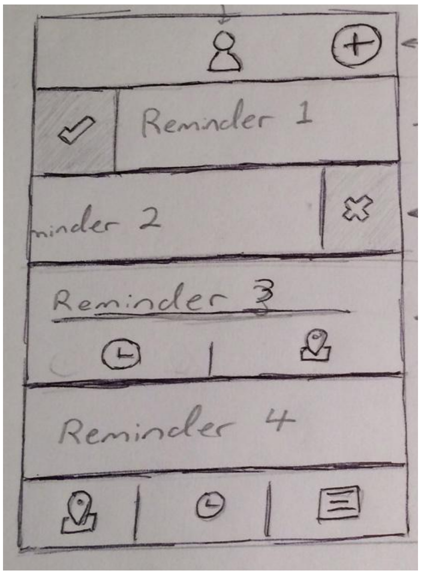


Fig. 3.5 List view ideas

The idea of weather or not to use gestures to delete or check of a reminder was explored because this would remove the need for two extra buttons. Figure 3.6 is the final List view mock up. It has actually taking ideas from all 6 initial ideas.



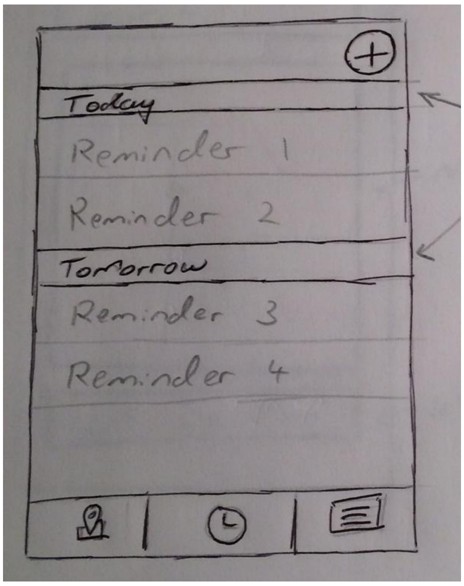


Fig. 3.6 List view with gestures, Fig 3.7 Chronological View

This explanation of what is going on in figure 3.6:

- At the top there is a plus button. This will take you to a new screen where you can add a to do. Beside it is an icon of a person but this is going to be replaced which my app icon
- Reminder 1 (with the tick beside it) shows a reminder being slid to the right to reveal a tick. This is how a user ticks of a reminder as done
- Reminder 2 (with an X beside it) shows a reminder being slid to the left. This is how you will delete a remind that you no longer want
- Reminder 3 has been clicked on and this reveals two buttons and makes the text editable. This is the edit state where you can change the text or else press one of the two icons to change the location or time and date. Both these buttons will take you to a new screen
- Reminder 4 is just the default state
- Along the bottom of the screen are three icons, which are used to switch between the list view, map view and date/time view.

3.3.3 Chronological View

Figure 3.7 is the chronological view of reminders. It is pretty much the same as the list view and it will have all the same functionality but it separates reminders into different days so you can see what is the most important.

3.3.4 Map View

Figure 3.8 represents the map view of current created reminders. This view replaces the list of reminders with a map. On the map will be pointers, each of which represent a reminder. When you click a pointer it will bring up its name. The map will also show your current position, which I have tried to show with a dark circle within a circle. This lets the user know is there is anything close buy to them that they could do. I don’t want to allow the edit function on this screen.

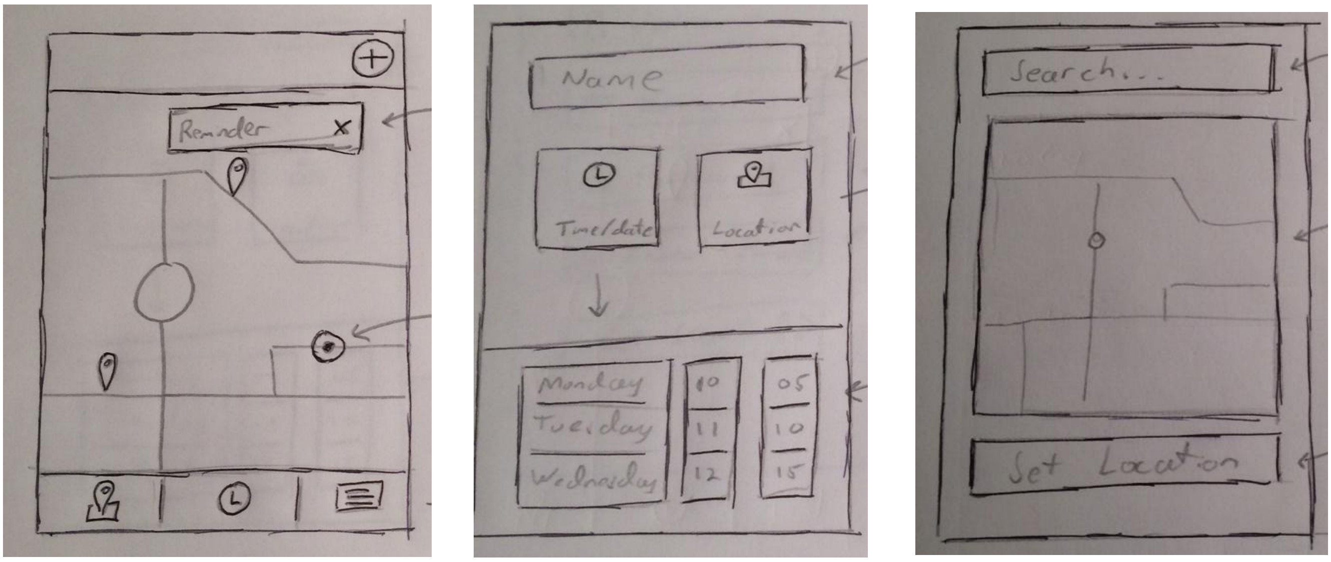


Fig. 3.8 Map View, 3.9 Create Reminder Form, 3.10 Select Location Map

3.3.5 Create / Edit Screen

Figure 3.9 shows one screen that will be used for the functionality of both creating a reminder and editing one. Again this screen has been kept very simple. It has a name input field and two buttons, time and location. If you press the time button the IOS date and time keyboard will appear, letting you input the time you want. If you press the location button it will open a popup of the next map screen. Once you are done filling in the information there will be a button to save the reminder.

3.3.6 Choose Location Screen

The final wireframe in figure 3.10 shows how a user will set the location of a reminder or edit the location. It allows you to search for a location in the input box or else you can just use the map by moving it around and tapping it to drop a marker on it. Once you have a marker at the right location you just press the set location button and it saves the location for that reminder and takes you back to the previous screen.

This report will focus on what was originally planned for the project and what has changed from this first concept. Some things will have obviously changed because of the complexity or else because the idea has changed and went in a different direction. It will also look at any potential roadblocks, which might arise, and an alternative approach will have to be though of in order to stay on track.

3.4 Feasibility

Up to this point the main focus has been on the creation of a native IOS app which is still the overall aim, but there was also the idea to develop a website that is linked to the app so that a user can create reminders online and receive them on their phone. It doesn't quite look feasible, as there isn't a long enough time frame to create a site and app. If the project were to go down this route it would need a central database to store all the reminder and user data so that both could access it, but a simpler option for this is to just store the data on the mobile device using local storage. This completely removes the website and gives more time for proper development of the app. Using local storage will also remove the need for a user login which would make the process of using the app a lot quicker and smoother. If it turns out that there is extra time to create a functional linked website to the app then it wouldn't be to hard to implement the database functionality as it would only be small differences in code to change from local storage to a database, but for the minute it seems to be out of scope so it is being removed as an aim.

Even though there wont be a functional website there should still be some sort of showcase website. This could be relatively quick to design and build and would greatly improve the power and brand of the app.

Another feature that was proposed in the concept report was "The ability to have multiple different lists of reminders". It was thought that this isn't really a necessary feature. If the user can view the reminders on a map as well as a list sorted by date and time then giving them the ability to create their own separate lists isn't needed. It is definitely a feature to keep in mind if there is any extra time at the end of the development, but for the first iteration of the app it doesn't need to be included.

Setting time and location as a way of triggering a reminder is still the main focus. Once a reminder is triggered by either of these then a notification is sent to the user. The time aspect should be relatively easy to create, the hard part is the location. The best solution that has been found at the moment is the use of geofencing of a position. This is where an imaginary fence is created around a point and once the users device enters the fenced area it triggers the reminder. This will require the app to constantly track the users location in the background. This functionality is available in phonegap so it should be possible to do it although very hard. The notification functionality is also available through phonegap but again might be hard to implement, but no one ever said it was going to be easy. Being able to do challenging things like this will be a steep learning curve but nothing is impossible. It’s a big risk taking on such a hard task but in the end the benefits should show. The best course of action would be to get a functional prototype of the geofencing and notifications working. It won’t need to have any design or user interface just the functionality. This is what the first functional prototype was.

Gesture based interactions is a crucial part of the project that will make it stand out and look and feel simple to use. But touch, swipe and slide interactions are very hard to do properly and there is no guarantee that phonegap will allow the integration of it. There are a few library’s that have been written specifically for these type of interactions like hammer.js but again it is dependent on how well it can be integrated into phonegap and how smoothly it will work. If it turns out that this cant be done then the fallback will just have to be buttons instead of gestures. It doesn’t create a huge problem but it would involve some redesign so the sooner it is found out if gestures work, the better.

3.5 Risks

The main risk in the app is the geolocation and push notification aspect of it but because this is the main function and selling point of the app there isn’t really any way of getting around it. In the case of these features not being achievable then the fall back will have to be creating a web app instead of a native app. As for the notifications, there are lots of services that allow you to push notifications to an IOS device so this shouldn’t be a problem either.

3.6 Methodology Selection

There are a lot of different development approaches which all have their advantages and disadvantages below are just a few as well as the chosen methodology that will be used through the development phase of the project. Some approaches are more suited for a team and others are better for individuals.

3.6.1 Waterfall Model

This method uses a very rigid approach with 6 pre-defined stages.

1. Requirements specification
2. Software design
3. Implementation and integration
4. Testing
5. Deployment
6. Maintenance

The development process is very easy to understand and easy to manage because of its rigid structure. It would work well n a small project where an initial product could be produced fast. The disadvantages to it is that once you are at a later stage it is difficult to go back and change something previous as it can have a detrimental impact to everything that has been done. There won't be any working software till the very end of the project so there cant be any real testing throughout.
Although it does have its advantages it wont be used within this assignment.

3.6.2 Spiral Model

The spiral model involves risk management at regular stages within the development cycle. It utilizes aspect of both the waterfall model and rapid prototyping. To put it simply, it is small iterations of the waterfall method until there is a fully finalized product. Usually there wouldn't be many more than 4 or 5 cycles or else the timeframe would be too long. Each development cycle will identify the biggest risk factors and try to accomplish them. Because it is a mixture of a couple of methods it does inherit their pitfalls.

3.6.3 Agile Development

Using the agile development cycle, software is develop in incremental and rapid cycles. It results in small incremental releases, each of which builds upon the previous. There is a lot of test involved with agile as it happens at the end of each cycle so it is good at preventing faults. It is a very flexible development cycle as it constantly allows changes throughout the project and it develops a usable piece of software very early on. This sounds very useful for the Spotnote project as an early prototype can be made using this method. Agile Development will be the process that is used and this leaves the ability to make changes throughout without any major impact.

3.7 Timeline

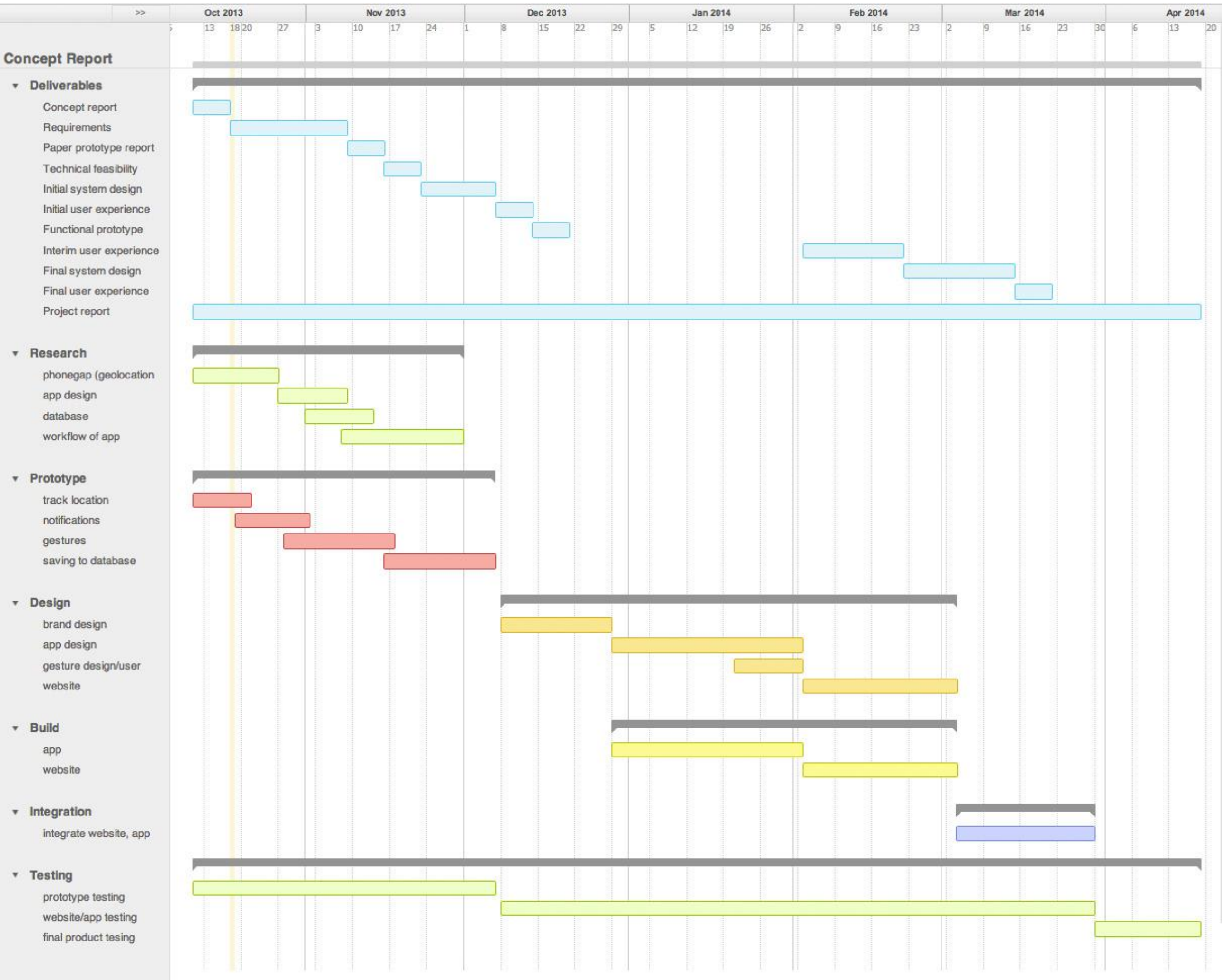
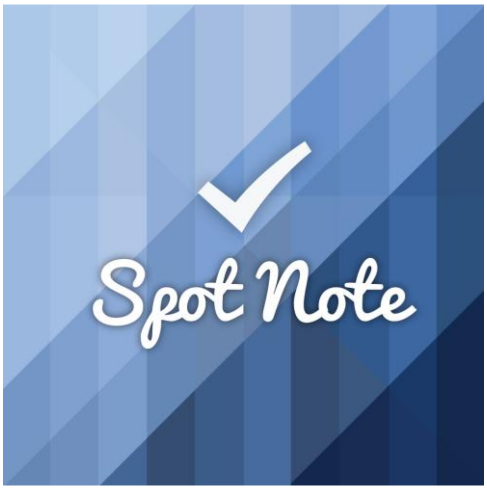


Fig. 3.7.1 Gantt Chart Timeline

Now that it is known what work is going to be undertaken a timeline has been drawn up (figure 3.7.1), so that time is used wisely and on the most important parts. Time management is a crucial part of development as you could easily get tunnel vision and spend too much time on a relatively unimportant aspect, which in turn doesn’t leave enough time for other more important parts.

4. Design



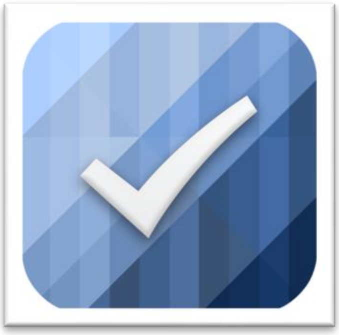
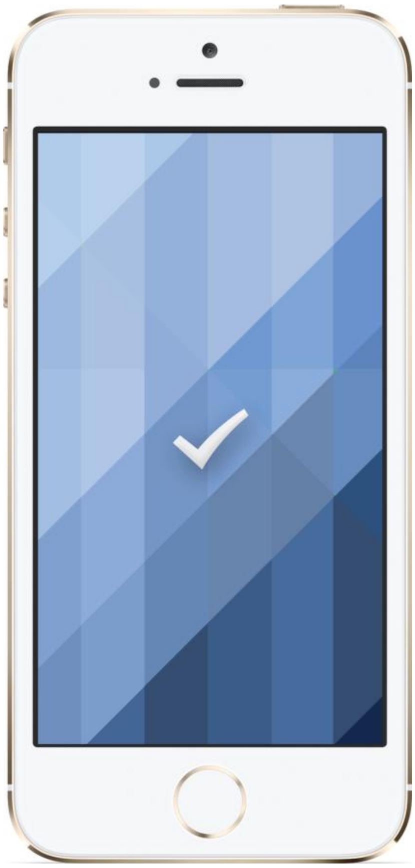
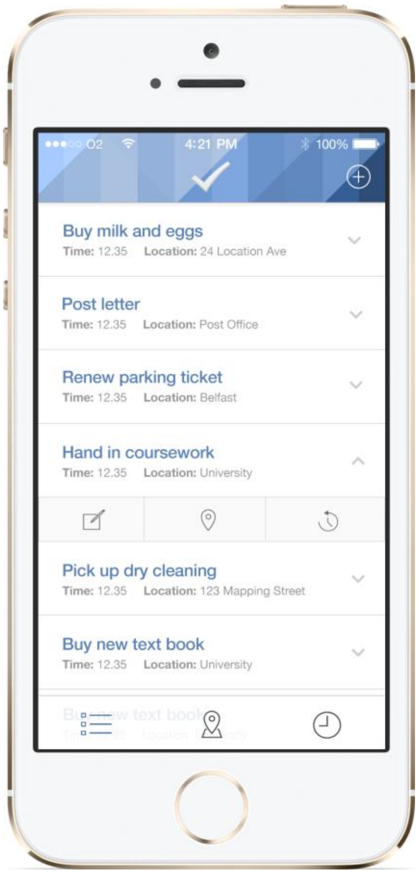


Fig. 4.1 Branding, Fig. 4.2 App Icon

All variations of design screens can be seen in appendix D. The initial design off the application started off with the branding shown in figures 4.1 and 4.2. It's a good place to start as it gives you an idea of how the whole project will look before starting into the actual design. It was decided that the project style should move away from the flat one color approach that a lot of apps are using which is reflected in IOS7. Instead it has been focused more on an irregular, triangular, geometric style pattern as the theme for the app. The pattern that has been created uses different shades of blue make it look as if there is a diagonal gradient. The font used is called Pacifico and was picked because it wasn't too corporate or professional looking. It's more friendly and fun and gives the app a more likable persona. The app icon uses the same background as the branding to tie it in. The actual tick icon on the app icon has added detail such as a subtle bevel, gradient and more prominent shadow. This is to make it stand out more from the background, otherwise it looks to flat and gets lost in the complicated background.

4.1 UX design and evolution





Buy milk and eggs
Time: 12.35 Location: 24 Location Ave

Post letter
Time: 12.35 Location: Post Office

Renew parking ticket
Time: 12.35 Location: Belfast

Hand in coursework
Time: 12.35 Location: University

Pick up dry cleaning
Time: 12.35 Location: 123 Mapping Street

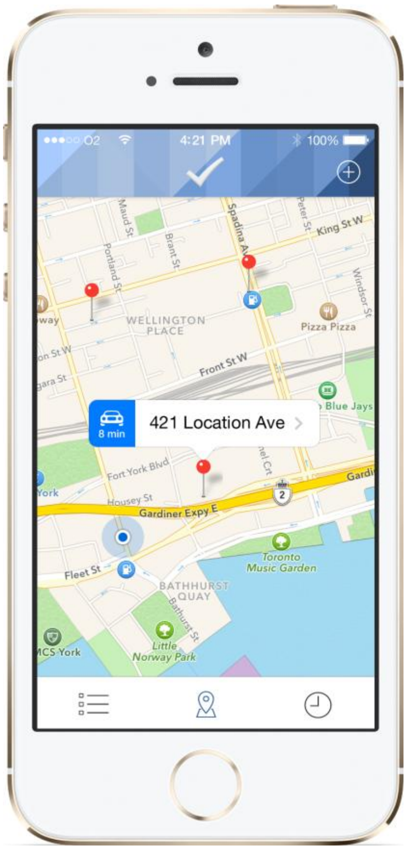
Buy new text book
Time: 12.35 Location: University

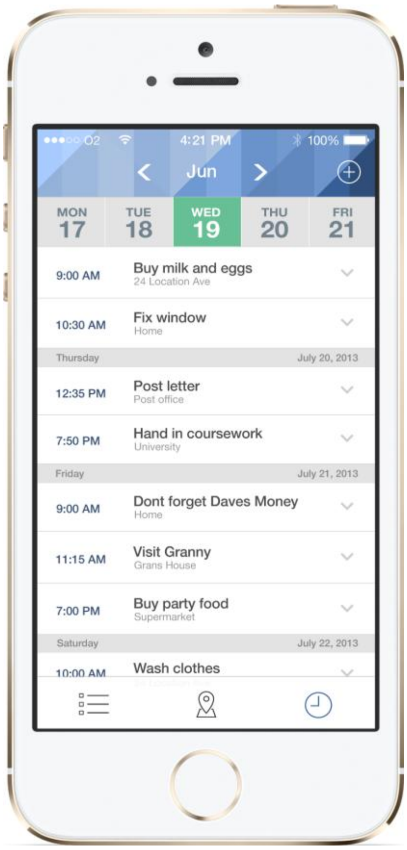
Fig. 4.1.1 Splash Screen, Fig. 4.1.2 Main List View/ Landing Screen

4.1.1 Initial landing Screen

Figure 4.1.1 just shows the loading/splash screen that will appear momentarily when the app opens. Beside it in figure 4.1.2 will be the first screen you see when opening the app. It's the main view of all reminders that have been created. If there is no reminders currently created then the user gets a prompt to add a reminder using the button in the top right. The positioning of this button follows convention of where a creation button should be. There is three different ways you can view reminders: List view, map view and chronologically. On all of these screens the create button is in the same place for consistency and ease of use.

Although the branding of the app has stayed away from the flat design of other apps, the layout style and icon design has been tied in with IOS7. This is to give the user a familiar feel when they first use the app. It means they will intuitively know where to go to do certain actions. The line style icons also follow the IOS7 design style and keep everything looking simplistic and light. The use of font size, weight and color has been carefully thought out as to make the user see the most important information first.





MON	TUE	WED	THU	FRI
17	18	19	20	21

9:00 AM Buy milk and eggs
24 Location Ave

10:30 AM Fix window
Home

Thursday July 20, 2013

12:35 PM Post letter
Post office

7:50 PM Hand in coursework
University

Friday July 21, 2013

9:00 AM Dont forget Daves Money
Home

11:15 AM Visit Granny
Grans House

7:00 PM Buy party food
Supermarket

Saturday July 22, 2013

10:00 AM Wash clothes

Fig. 4.1.3 Map View, Fig. 4.1.4 Chronological View

4.1.2 Map and Chronological View

The two screens of figure 4.1.3 and 4.1.4 are the map view of reminders and chronological view. Map view is basically all of the reminders that a user has created and that have a location are plotted on a map. This is so that you can see if you are close to something that you need to do. The markers on the map can all be pressed to reveal the reminder name, if the reminder name is pressed this will take you to the create screen where you can edit the details of it such as change the location name or time.

The second screen is the chronological view and is probably one of the most complicated screens. At the very top in the header bar there is the option to change the month. This is to let the user skip ahead and see reminders they have set far in the future. The next thing is a slider to pick the date. When the slider is changed it will automatically change the list below and will jump to that date. This method was decided upon instead of a search box, as it is quicker and nicer to use instead of just inputting a name, which you might forget. The selected date is always in the middle and will be highlighted in a different color. Small grey separators indicating the dates separate the actual list of reminders. If the list of reminders is scrolled through it will update both the date slider and the month selector when needed. Again a great amount have time has been spent making sure that font weight and size help to direct the users attention to the most important information on the screen. This screen has a lot of impressive functionality but because it is slightly more complicated than the initial reminder list view it was decided that this should not be the first screen that you land on when opening the app.

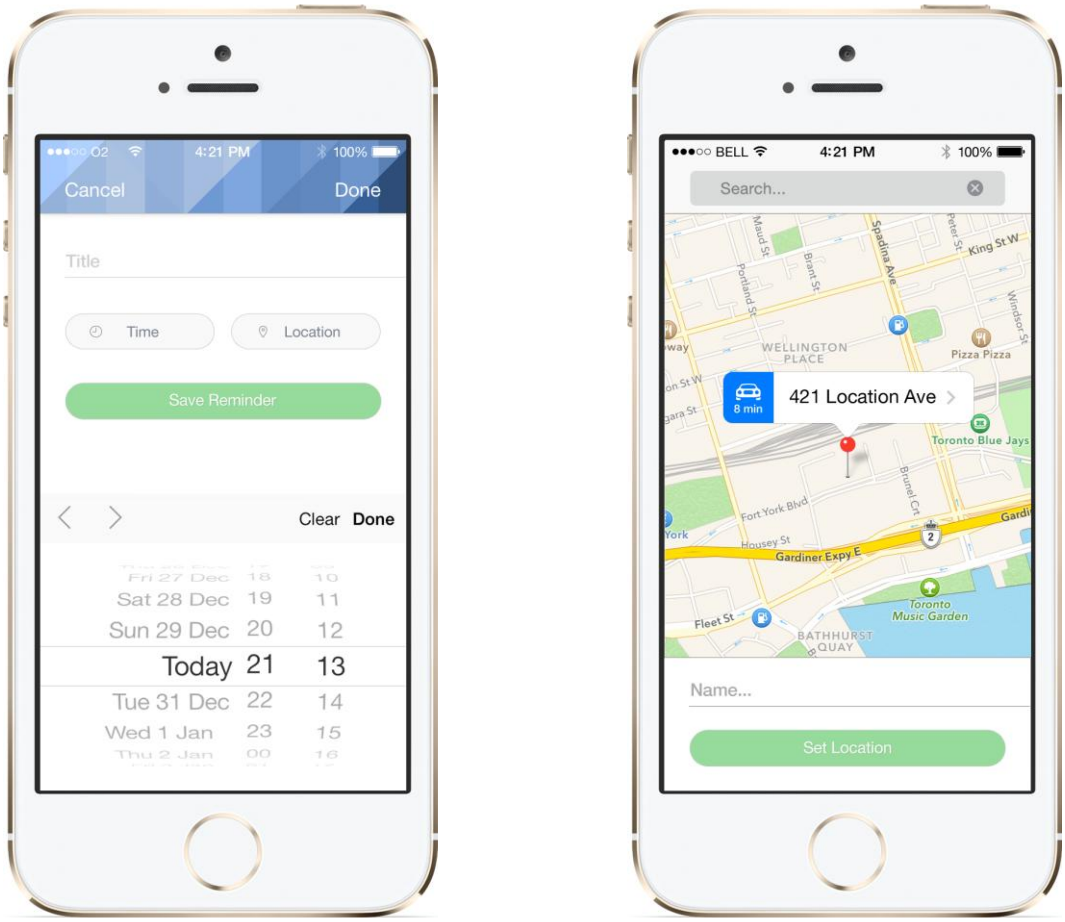


Fig. 4.1.5 Create Reminder Form, Fig. 4.1.6 Select Location Map

4.1.3 Create and Edit Reminder Screen

The user can access the create reminder screen from any of the three main views. On this screen there is only a minimum of two required fields that need to be filled in before the reminder can be saved. Title is required and then either time, location or both. Title is a simple text input where the user can enter whatever they want. The time button will open the native IOS date/time picker so it is easy to quickly input a time.

For the location input, you are taken to a new screen, which has a map and a search box. The search box is to help find a specific location. One the location is found just tap that point on the map and a pointer will appear. You will notice on the map screen that the header is no longer blue. This is because this screen will be a pop up and will appear over the top of the main create/edit screen.

Both these screens have been kept as simple and streamlined as possible to help ease of use. When saving, if there is an error then a message will appear beside the input that has caused the error prompting the user to fix the problem. IOS7 guidelines have been followed in regards to the placement and wording of buttons such as “Cancel” and “Done”.

4.2 Development

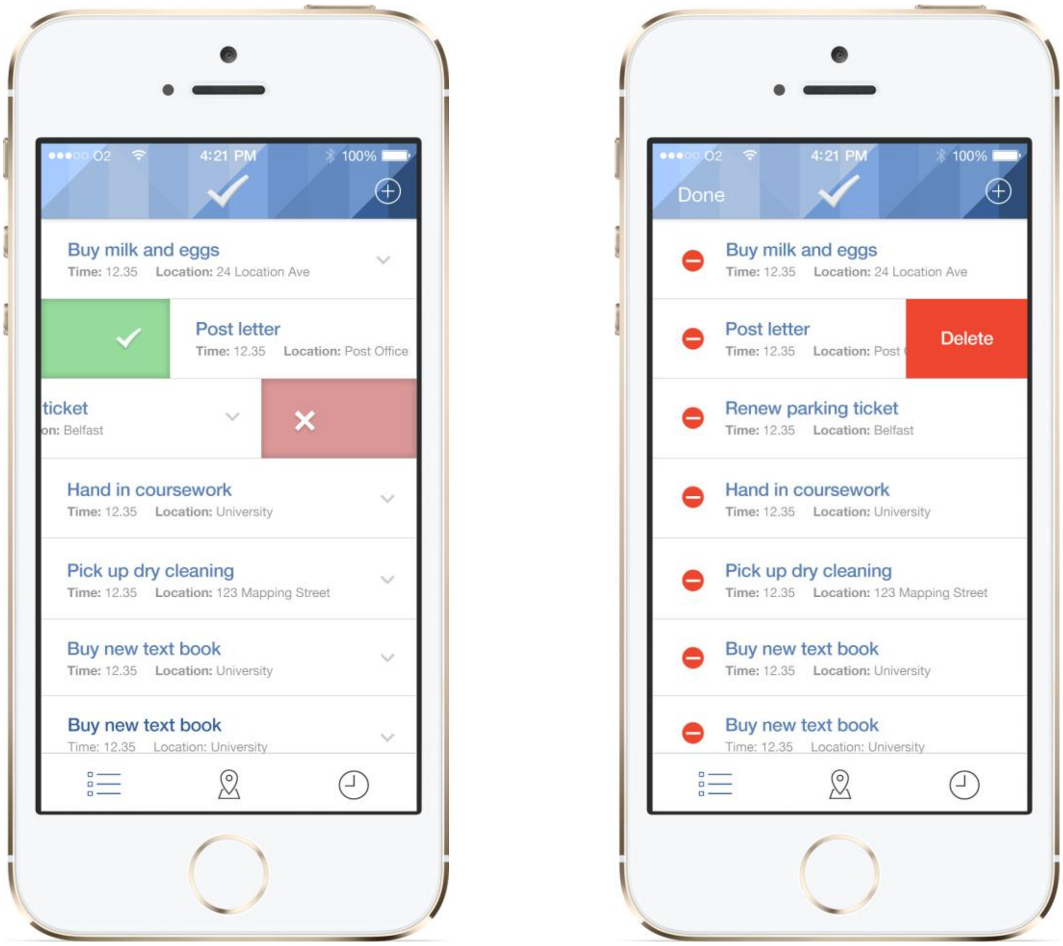

●●●○○ O2
4:21 PM
100%
Buy milk and eggs
Time: 12.35 Location: 24 Location Ave
Post letter
Time: 12.35 Location: Post Office
ticket
on: Belfast
Hand in coursework
Time: 12.35 Location: University
Pick up dry cleaning
Time: 12.35 Location: 123 Mapping Street
Buy new text book
Time: 12.35 Location: University
Buy new text book
Time: 12.35 Location: University
●●●○○ O2
4:21 PM
100%
Done
Buy milk and eggs
Time: 12.35 Location: 24 Location Ave
Post letter
Time: 12.35 Location: Post
Delete
Renew parking ticket
Time: 12.35 Location: Belfast
Hand in coursework
Time: 12.35 Location: University
Pick up dry cleaning
Time: 12.35 Location: 123 Mapping Street
Buy new text book
Time: 12.35 Location: University
Buy new text book
Time: 12.35 Location: University

Fig. 4.2.1 List View Interactions V1, Fig. 4.2.2 List View Interactions V2

4.2.1 Interactions

Initially the user interaction of checking off or removing a reminder was going to be through sliding the desired row to one side or the other. This way it is a lot quicker to and easier to use a slide gesture instead of a button press. But when it came to development it was found out that this method could be too complicated to implement so if there is no way to have it properly incorporated into the app then the fallback is to have and edit button situation in the top left of the screen in the header bar. When it’s pressed then all reminders go into an edit mode where they can then be individually selected and deleted. This way also provides and extra confirmation step which stop the deletion of a reminder by accident. Visually the initial design of using gestures to remove reminders is the better option but depending on development time it might not be feasible. Another reason that gestures for deletion might not be able to be implemented is because of hardware restrictions. The app is being created in HTML, CSS and JavaScript and bundled up using PhoneGap so that it can be deployed for IOS. Having multiple sliding elements using JavaScript could heavily impact the performance and speed of the app. Speed is massively important to a user because if they have a slow and unresponsive app ten they will simply not use it and uninstall it.

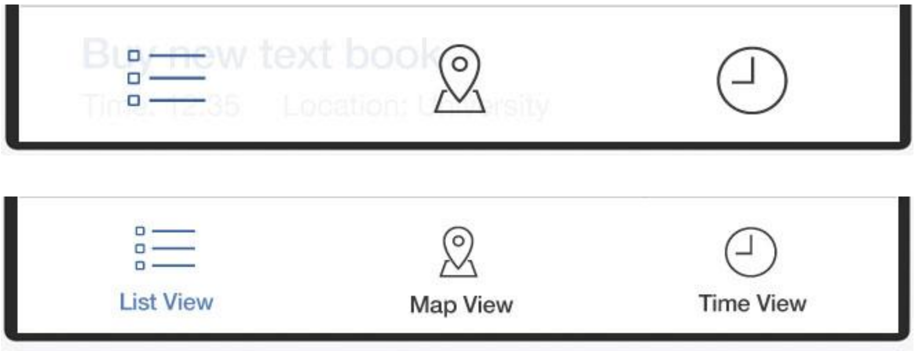


Fig. 4.2.3 Navigation Bar versions 1 and 2

4.2.2 Navigation

After a bit of research, it was found that navigation items in IOS usually have both an icon and text. This is quite an important oversight in the original design as it is important for the user to know what they are pressing. It’s a very small addition but it will greatly improve the ease of use and improve navigation speed. It makes the icons a lot more obvious as to what they are. The background of the navigation bar is semi transparent so that the content behind it will be visible. This is just to add some extra depth to the design

Final Design

The final design didn’t change a great deal from the original design apart from a few user interactions and navigation improvements. This is because a lot of research was done at the start to ensure the initial design was as close to finished as possible. Doing all of the design at the start of the project has changed the development cycle from an iterative process to a waterfall approach. This isn’t what I wanted, as waterfall is a lot stricter and doesn’t give the opportunity to go back and change design easily when there is a functionality change.

The final design is still subject to change while the app is being developed. If a problem arises or it is found out that something isn't feasible then it will be changed to accommodate the functionality. But enough research and testing has been done to make sure everything should be possible.

4.3 System design

4.3.1 System diagram

The project will be using SQLite for the database, which will be housed on the users mobile device instead of a remote server. This means it wont use the typical client server model where the database is housed on a separate server and the client side talks through the Internet to the server to get the data. Instead everything will be done client side. So the markup and front end scripting will talk directly to the storage scripting on device. One of the benefits of this is that it doesn't require an Internet connection, so you could have no reception on your mobile device but still access all of your data instantly.

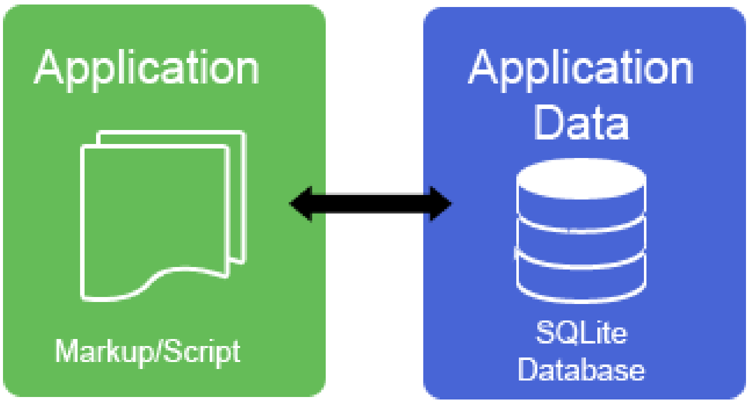


Fig. 4.3.1 Application data

Figure 4.3.1 is a very high level of how the application and data will work in the system. The next diagram shows how the PhoneGap mobile development framework works.

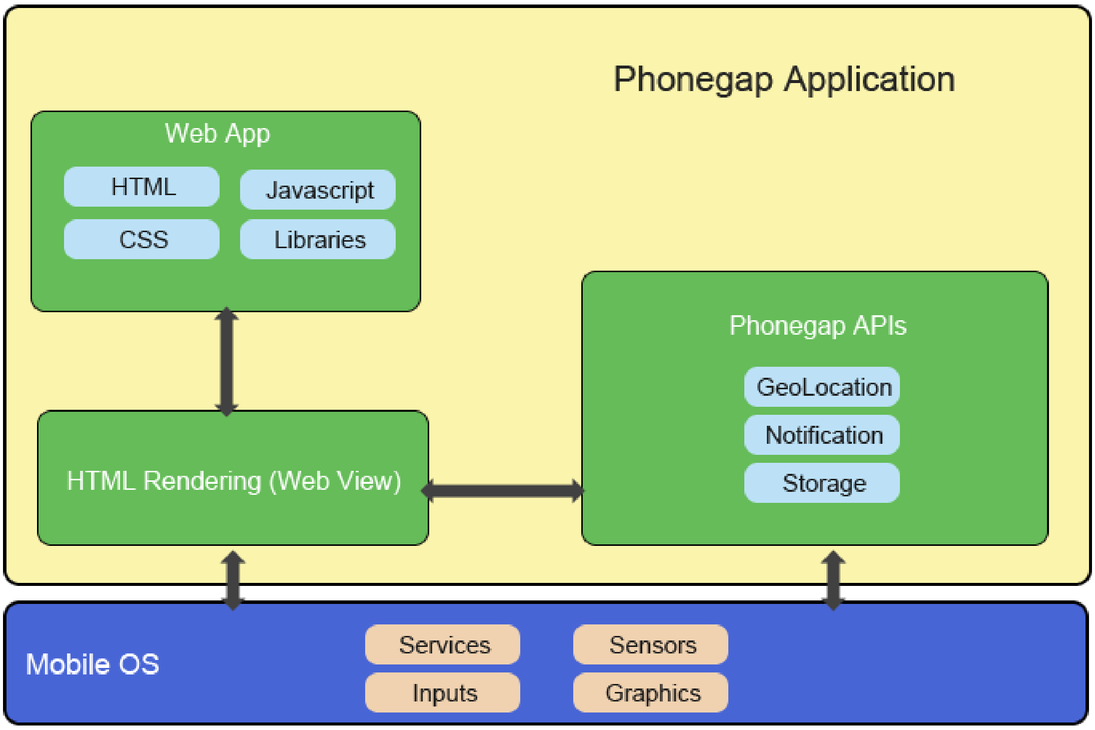


Fig. 4.3.2 PhoneGap Application Diagram

In essence, when you are creating a mobile application using PhoneGap, you are just making a web app and PhoneGap wraps it up so that it can be deployed to mobile devices. It also gives you the ability to access native functions of the mobile device through different APIs. Figure 4.3.2 displays how the native HTML CSS and JavaScript is bundled into a PhoneGap application which gives the ability to talk to the native functionality For example, the app will use geolocation to get the users position and check it against positions that are stored in the SQLite database. Below are the two functions that will be used to get positional values

```
Navigator.geolocaiton.getCurrentPosition()
```

```
Navigator.geolocaiton.watchPosition()
```

Both functions have success, error and options that can be passed into them. PhoneGap offers this type of access to all sorts of native mobile functionality that will be made use of in this project.

4.3.2 Model View Controller

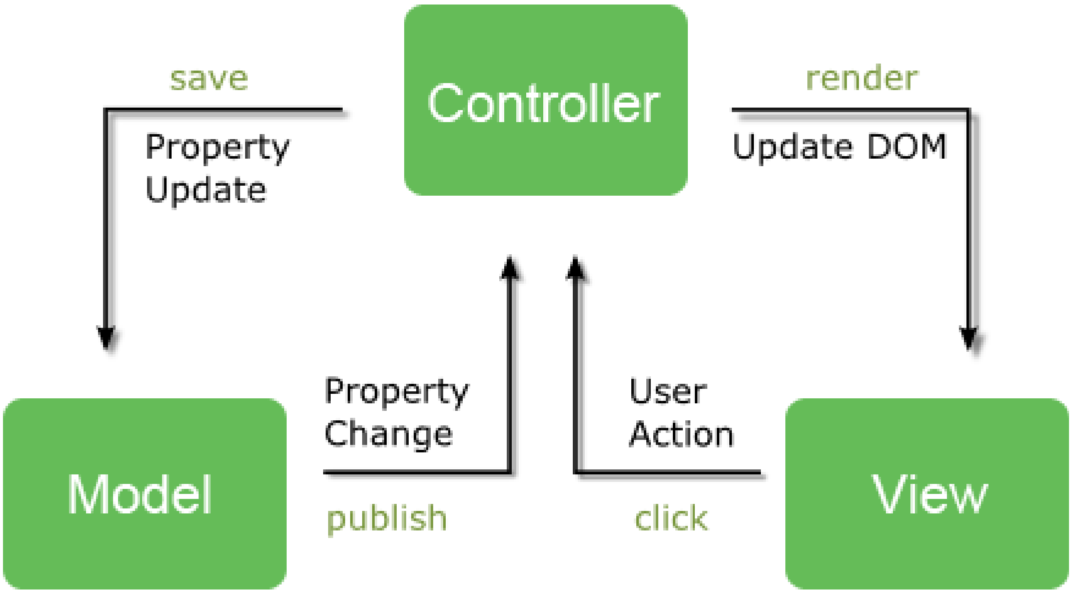


Fig. 4.3.3 MVC

There are many different JavaScript MVC libraries' including Ember.js, Angular.js backbone.js to help build apps using the model view controller pattern. Some of these can be very useful to quickly start a project and have a ready built file structure. This project won't use any of these as it will have a simple enough MVC pattern and wouldn't really benefit from an additional library. Figure 4.3.3 shows how MVC works with an example of the interaction/process that is happening

4.3.3 Data Model

Figure 4.3.4 is a diagram of the model view controller software architecture pattern that is used within the app to show how each bit of technology is used.

The view will consist of the HTML CSS and JavaScript as well as the custom CSS and JavaScript that is used buy the mobile framework Chocolate Chip UI. The project uses Chocolate Chip UI as a base style template and it has a lot of functions to help make a one page, multi view app. It is a good simple starter template that is very easily customized so the style can be completely changed to look unique.

The controller is made up of JavaScript and phonegap API that sends and receives data to the database as well as dynamically changing what is displayed in the view depending on what the user inputs or what is in the database.

The model is the data part of the system and in this case is a WebSQL database that stores all of the reminder data.

All parts of the MVC architecture pattern are contained within the app so there is no need for any queries leading to an external server.

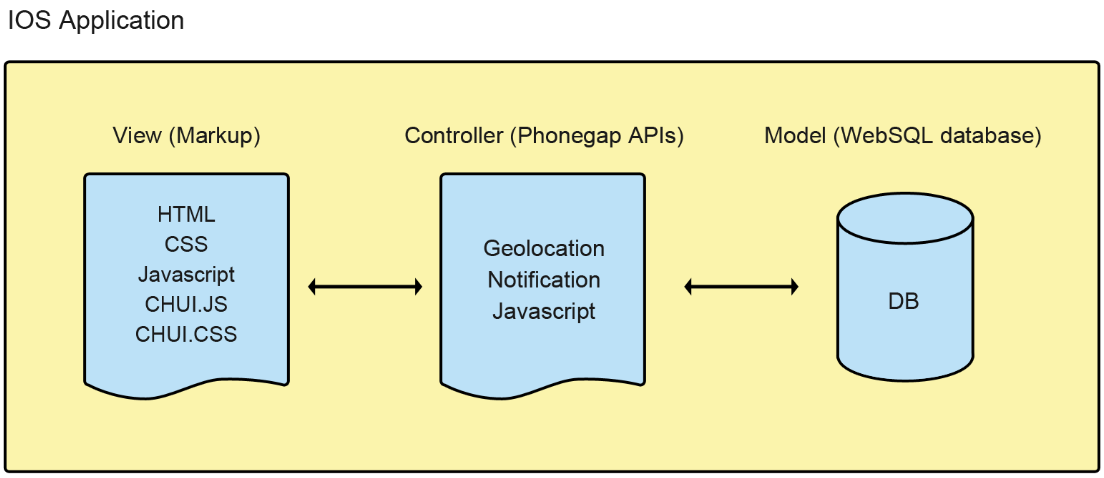


Fig. 4.3.4 PhoneGap MVC

4.4 Logic design

The next thing to work out is the actual logic of how the app works and the data flow to and from the database. The data flow should be pretty simple as there is only create and edit capabilities of the one group of data. It is simplified even more by removing the login functionality.

4.4.1 User Interaction Workflow

The main user interaction within the app is to create a reminder. To make sure the whole process is done correctly and no steps are overlooked a workflow diagram has been created. This diagram can be used as a reference when coding and shows things like calls to and from the database. Figure 4.4.1 is a visual representation of the user interaction for creating and editing a reminder.

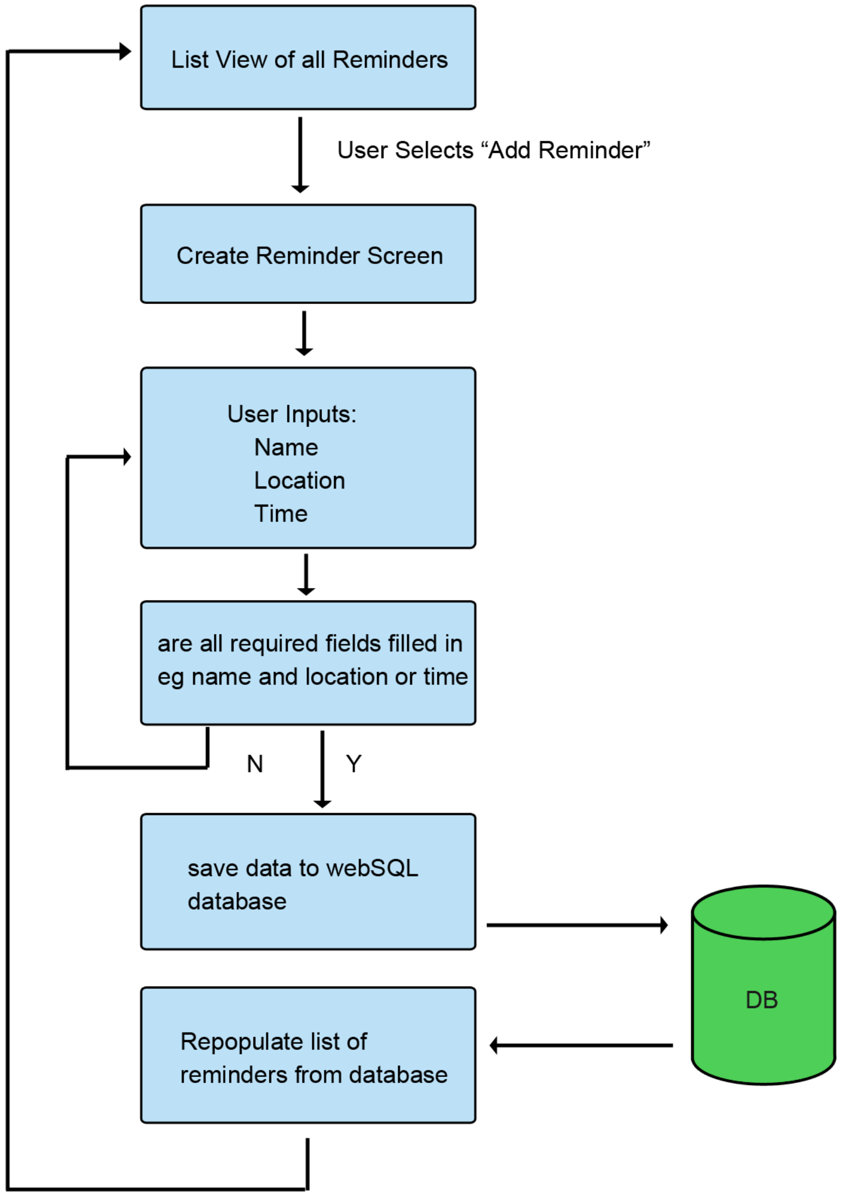

List View of all Reminders
User Selects “Add Reminder”
Create Reminder Screen
User Inputs:
Name
Location
Time
are all required fields filled in eg name and location or time
N
Y
save data to webSQL database
DB
Repopulate list of reminders from database

Fig. 4.4.1 User Interaction Workflow

The process starts of when the user opens the app and they land on the list view of reminders. The only places they can navigate to are different layouts of reminders or else opt to create a reminder. Once they do this they are taken to the create reminder screen, which is a form with three different inputs: Name, Location and Time. Name input is a required field and either Location or time must also be entered. Once the required amount of information has been entered then the data is saved in a new row in the database. After the data has been save the user is directed back to the original screen they were on which is a list of reminders and the new one they entered will appear in the list.

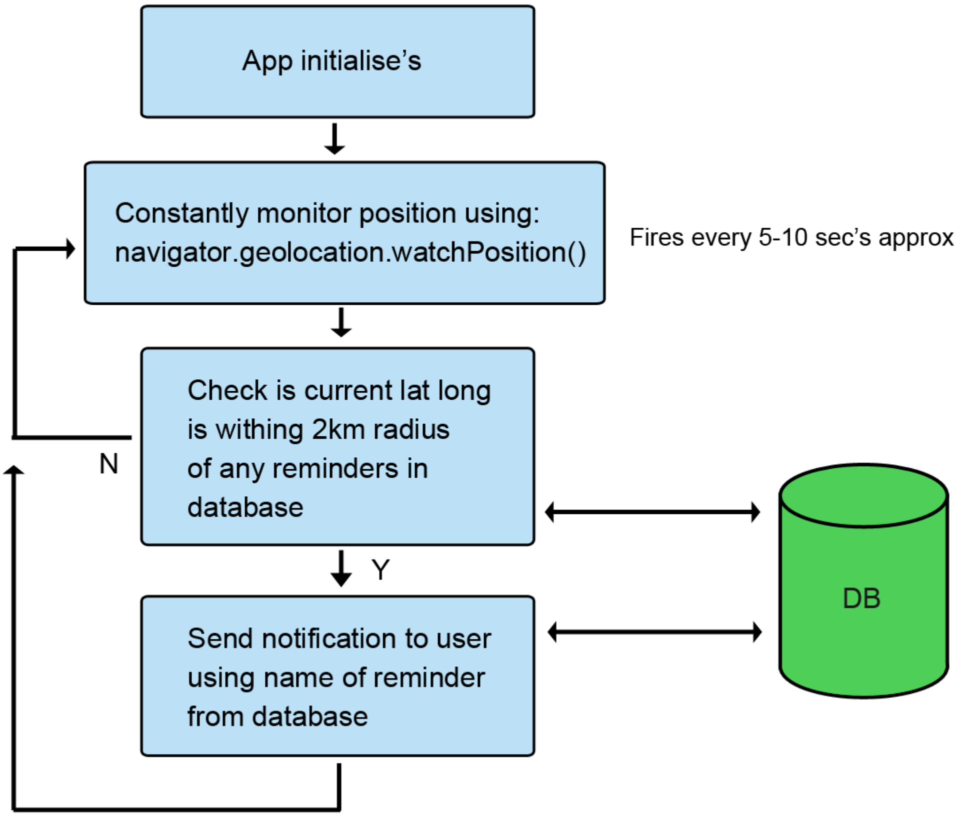

App initialise's
Constantly monitor position using: navigator.geolocation.watchPosition()
Fires every 5-10 sec's approx
Check is current lat long is withing 2km radius of any reminders in database
N
Y
Send notification to user using name of reminder from database
DB

Fig. 4.4.2 Geolocation Workflow

4.4.2 User Geolocation Tracking Workflow

This next workflow diagram in figure 4.4.2 lays out the steps needed to be taken to track the users location and check it against locations already stored in the database. If it finds a location in the database that is within a certain radius then it will send a notification to the user alerting them of it. All of this can happen in the background when the app isn't open.

At the minute if a user decides to completely close the app this will force it to stop tracking their location and therefore wont be able to send a location based reminder. This is a major pitfall of the app but it's because of the limitations of PhoneGap. It doesn't have a way of continually tracking as all code stops exciting once the app is forced to quit. A way around this might be to write a native plugin using objective-C that ties in with the PhoneGap features. But unfortunately at the moment this is a step too far to start developing in these languages. If there is time to learn a little objective-c it could be an option to create a plugin to get around the problem but until then it is just going to be a known bug within the app.

4.5 Data design

4.5.1 Data Structure

Within the database there will only be one table that is used to store all of the reminder data. The first column named id is just used to create a unique identifier for each row. The name, lat, long, date and time columns are all populated by the data that the user inputs. Either lat and long, or date and time can be left out but not both. The final column that is called alerted is used for the sending of notifications. When a user is within the certain radius of a point then a notification is sent. Once it is sent then this alerted value is changed to true so that they don’t continually get alerted about the same thing. Once they leave the radius of a point then it is set back to false. It is just a way of preventing multiple notifications when you are within a geofence.

4.5.2 Database Design

Table 4.1

ID	Name	Lat	Long	Date	Alerted
1	Get Milk	54.439609898030355	-5.68793296814	12/02/14	false
2	Post Letter	54.422254257445374	-5.69278240203	13/03/14	false

In the reminder app the only data that will be stored is the reminders themselves. There is no need to store the users data or ask for a username or password. So all the data can therefore be stored on the one table, which might look like the table below

The **‘reminder_id’** is just an id to keep each reminder unique, so it will be a number that will auto increment. The type will be INT because there is no telling how many reminders could be entered in the long run so this should allow for a big enough number
‘Name’ is just the name of the reminder that the user has inputted. So that the user can input anything here the field will be VARCHAR.

Lat and long is the co-ordinates of where the user wants to get reminded about. This is the value that will be monitored by the geofencing plugin and when the user is within a certain distance of the point then a push notification will be sent to inform them. It can be updated when the user edits reminders information. The field will be DECIMAL so that it will accept a large decimal number which is the format both latitude and longitude will be in.

‘Date’ is going to be used to set up the notification but it won’t be constantly checked against like lat and long. Once it sets up a notification it isn’t needed again unless the user decides to edit it. This field will be DATETIME.

‘Alerted’ will only ever bee true or false so the field will be TINYTEXT

5. Implementation

5.1 Technology/tool use

There are a few prerequisite that need to happen before development of a PhoneGap application can start. PhoneGap isn't just a library you download and link in the head of the file.

First of all node and npm had to be downloaded from their website and installed. Once they are installed the next few commands have to be run in the terminal to install PhoneGap and its command line interface, and then set up a project with all the required files.

```
sudo npm install -g phonegap
//this installs phonegap

phonegap create Spotnote
//this sets up all of the files and folders for a project called
Spotnote

cd Spotnote
//this navigates to the Spotnote folder

phonegap run ios
//builds the application into an xcode project and runs it on an
emulator
```

Or just to build it and not run it

```
phonegap build ios
```

At this stage there is a few more technologies that have been identified that need to be used in the project including the originally stated ones

5.1.1 PhoneGap

For the geolocation, the basis of watching the devices location is called by the following:

```
var watchId = navigator.geolocation.watchPosition(
          geolocationSuccess,
          geolocationError,
          geolocationOptions);
```

This returns the device's current position when a change in position is detected. When the device retrieves a new location, the geolocationSuccess callback executes with a **position** object as the parameter. If there is an error, the geolocationError callback executes with a PositionError object as the parameter. When it successfully returns a position then this position can be check against any reminder positions that are stored in a database.

Unfortunately PhoneGap doesn’t have the out of the box ability to create local notifications that this project will need. But there is a lot of plugins that developers have created to try and bridge gaps like this. Sebastián Katzer has developed a very prominent and widely used notification plugin. Its easy to install has a lot of features and great documentation. The basic code for a local notification is:

```
window.plugin.notification.local.add({
     id:             String,   // A unique id of the notification
     date:             Date,       // This expects a date object
     message:        String,    // The message that is displayed
     title:      String,  // The title of the message
     repeat:         String,    // Has the options of 'secondly',
     'minutely', 'hourly', 'daily', 'weekly', 'monthly', 'yearly'
     sound:      String,  // A sound to be played });
```

What it does is create a unique notification with multiple different options that can be passed into it that will come in very useful.

Apart from these two mentioned methods within PhoneGap, the rest of the coding should be web based.

5.1.2 PhoneGap CLI

This is the command line interface for phonegap and is installed at the same time as PhoneGap. It is used to do things like check plugins or build the app and run it on a simulator a few of my most run commands are:

```
phonegap plugin ls
//checks what plugins are installed
```

5.1.3 Grunt

Grunt is a JavaScript task runner, which is used for such things as compiling less files for Chocolate Chip UI

Grunt is installed through npm by executing the lines

```
npm install -g grunt
npm install -g grunt-cli
```

Once installed the command "`grunt less`" will compile all of the less files into one CSS file

5.1.4 Bit Bucket

Bit Bucket is used instead of Github for backup and versioning because it allows unlimited private repositories. If it was a public project then Github would be used but because it is a university assignment then Bit Bucket is used.

A simple example of a git command for Bit Bucket is:

```
git push -u origin –all
```

This will push all committed changes that have been made to the code to the original branch of code on bit bucket

5.1.5 Google maps

PhoneGap doesn't give access to the native maps that are on IOS so Google maps are used to visualize all of the reminders on the map as well as help the user create new reminders. When the user wants to add a location to a new reminder or change an existing reminder, they are taken to a Google map where they can search for a specific point or else move around the map and place their own point.

5.1.6 Ripple Emulator

Ripple is an in browser emulator that emulates the PhoneGap environment therefore eliminating some of the need to constantly build to a device. It means development can move a lot faster.

The ripple emulator that can be installed as a chrome extension doesn't actually support the latest version of PhoneGap 3.0, which was found out the hard way. To get it to emulate for the right version it had to be installed locally and run on localhost through the node environment. This seems like a very complicated process but the following two lines will install ripple and start the emulator:

```
npm install -g ripple-emulator
ripple emulate
```

5.2 Notable challenges

5.2.1 PhoneGap, Xcode and objective-c

Initially when PhoneGap was researched it seemed to be a very simple platform that could be used to port web technologies to native IOS but as the development started it turned out to be more complicated than originally thought. These complexities came from the command line interface through terminal. There are a lot of commands to learn and dependences to install as well as emulators before even the test app can be run. Although it wasn't a major hurdle it was still a challenge to get to grips with.

To install some of the necessary plugins for this project it required a very small bit of objective-c coding. It was only small things like adding a pre defined line or uncommenting other parts, but to someone who has never used Xcode before or seen the file structure of an application it was hard to follow and quite daunting. Once the basic file structure is understood of the project then it gets a lot easier but to actually build the whole thing in objective-c, from a web designer's point of view, would be very difficult.

5.2.2 WebSQL Database

Another notable challenge was the actual creation and implementation of a WebSQL database. This is a technology that has never been covered in university so it is a very new challenge. After following a few tutorials and experimenting with it a database scheme was easily created. The simplicity of the table structure made this process a lot easier. Database calls are basically the same as MySQL, which makes the learning curve a lot easier. Below is the function to select a reminder from the defined database that has the id of 1.

```
var db = spotnotedb.webdb.db;
//define the database
db.transaction(function(tx){
//set up the transaction function
     tx.executeSql("SELECT * FROM reminder WHERE ID = 1",
     //execute sql query
     success,
     error);
     //error and success callbacks
});
```

WebSQL has nearly all of the same functionality as MySQL but obviously it wont have the same capacity.

5.2.3 Workflow for testing

To properly test the app it needs to be built on a device but this is rather impractical to do as it takes quite a long time to compile all of the code into an Xcode project and then build it onto a device and test the feature. To get around this long workflow the ripple emulator is used for most of the development and design. After a notable section or feature is complete then it is built to a device. This is a much more productive way to develop and works a lot quicker. If there are problems on the device it can be inspected through the Safari web inspector. This is possible because the app is actually a web view. This makes debugging a lot easier and solutions to problems can be built quicker

5.2.4 Geolocation

Geolocation through phonegap works perfectly when the app is in the foreground but once it is minimized into the background the tracking halts. After research it was found this was because of a setting within a property list file. This file defines the properties and capabilities of the app and for geolocation to work it needed the blow line added to it:

```
Required background modes: app registers for location updates
```

This allows the app to receive location updates when in the background. Unfortunately what it doesn’t allow is location updates when the app is fully closed by a user. This is simply because of the limitations within the design of phonegap. One the app is closed it closes the web view that all of the JavaScript actually runs in. Because of this there is no way to process any of the location data that is received. To do this a native objective-c plugin would need to be written. This plugin would take over if the app gets closed and it would receive and process the location information. This sort of defeats the initial purpose of the project, which was to develop the app using only web technologies. So at the minute it is just a known bug that can be fixed in the future

5.3 Notable achievements

5.3.1 Calculating the distance between two points

To find the distance between two lat and long points over the surface of the earth a formula called the ‘haversine’ formula is used. This is the most accurate formula to find the distance as the crow flies. The online web development community is very active and has translated this formula into a JavaScript function, which can bee seen in the source code. It uses advanced circle trigonometry to find the difference between the two provided latitude and longitude positions and then calculate the curved distance using the radius of the earth. The below code shows the formulae:

```
function getDistanceFromLatLonInKm (lat1,lon1,lat2,lon2) {
  var R = 6371; // Radius of the earth in km
  var dLat = deg2rad(lat2-lat1);  // deg2rad function below
  var dLon = deg2rad(lon2-lon1);
  var a =
    Math.sin(dLat/2) * Math.sin(dLat/2) +
    Math.cos(deg2rad(lat1)) * Math.cos(deg2rad(lat2)) *
    Math.sin(dLon/2) * Math.sin(dLon/2)
    ;
  var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
  var d = R * c; // Distance in km

  return d;

}

function deg2rad(deg) {
  return deg * (Math.PI/180)
};
```

5.3.2 Map view screen

This was quite a nice accomplishment as it had a good visual output. It involved using the geolocation.watchposition method to update the position of a marker on a Google map but the position is only updated on the map if it has been loaded. It only gets loaded whenever the user navigates to the view it is in. That is a lot of intricate integrated functions using multiple checks to see if the map tiles are loaded and then checking if the marker exists. The reminder markers are also placed on the map by iterating through all of the items in the database and using the lat and long attributes. If these attributes are missing then it is skipped. There is a good amount of functionality behind the map view screen just to show a map and a few pointers, which looks very simple. That is probably the key to a successful app, do a lot of complex programming but still make it looks simple and visual to the user.

5.3.3 Delete functionality

Originally in the artwork stage it was envisioned that the delete functionality would be achieved through using a gesture of sliding the reminder list item to one side. When further research was done it was found out that this was not the best way to do it according to apples guidelines (iOS Human Interface Guidelines 2014). They recommend a two-stage process, which prevents accidental deletion. Also to implement the gestures would require quite JavaScript heavy coding, which mobile browsers around good at dealing with. It would most likely slow down the whole app and compromise usability, which is one of the top priorities.

5.3.4 Form functionality

The functionality of one small form ended up being very complex and in the end was a very good accomplishment. It required validation checks, automatic population for editing capabilities, dom manipulation, map population, database queries and a few other functions. All this for one form with only three inputs, and only one input the user can actually type into. This was one of the big surprises of the project because of how much time it took up to get it working seamlessly but it was also one of the most satisfying parts when completed.

6. Testing

Software testing is and investigation to provide detailed information about how well the product functions based on original requirements. The best testing comes from an independent source that is unbiased towards the product.

Testing takes in a wide variety of aspects of the project. It is not just if a feature works, but does it work in the desired way, is it properly validated, are the images and code optimized and many other factors.

6.1 Testing approach selection

6.1.1 White box testing

This is testing carried out by an individual or set of individual that has prior knowledge of the internal workings of the product. It is to test internal workings including things like database queries and expected results from user inputs. This kind of testing was continually done throughout at every different stage of development to ensure all aspects worked together properly. Most of the testing was carried out on the ripple emulator and then a larger test would be carried out on a device. All database queries are first tested in a browser via the developer console, then implemented into code and test a second time to insure they still have the current output. Rigorous testing like this was the normal.

Form testing and validation was done from the point of view of someone who has never seen the app before. This means any and all types of data that can be entered, was entered. It firstly made sure that the validation stood up to all types of data and secondly made sure the database could handle any data that was sent to it. Luckily enough there is only one input that the use can input into directly. The other inputs will be automatically filled depending on what the user selects. This was actually a conscious effort to reduce complexity and the amount of testing needed.

6.1.2 Black box testing

This type of testing is done by someone who has no previous knowledge of the system or how any of it works, they will only be aware of what it is meant to do. Twelve volunteers within the class carried out this testing. The following instructions/ test cases were given to the users and they were asked to carry them out.

1. Navigate to the map view screen
2. Navigate to the time view screen
3. Create a reminder with just a time attribute
4. Create a reminder with just a location attribute
5. Create any type of reminder
6. Edit a reminder from the main list view
7. Edit a reminder from the time view screen

These test cases should cover all functionality of the app. After carrying out these tests the users were asked to carry out a survey to find out how they felt about using the app. The survey can be seen in appendix F. The results of the survey can be seen in section 7.1.

6.2 Testing process

Testing on the app continually happened through the development stage. It started of by testing in the Ripple Emulator in the Chrome browser. Once it was found to functionally work it was then tested on a device. This two stage testing was very effective at eliminating bugs.

Optimization

The first thing that can be optimized is the use of images. A good practice is to put all icons into one sprite, so instead of loading lots of small images it just loads one big one. It might not make a huge different or even a noticeable difference but it is good practice as it can make a difference on larger projects. The next thing to do is minify all files. There are a lot of handy tools online and downloadable that can be used to do this. Luckily enough the editor that is being used, coda 2, has an inbuilt minify function. This cuts down on file size that in turn cuts down on loading time.

Spelling, Grammar and Punctuation

Sometimes this can be overlooked or for forgotten about but it is important for the user experience. All spelling was checked and double-checked.

6.3 Test results

Table 6.1 is an example of the requirements being used to test against. Not everything gets a pass due to either the function being dropped or changed.

Table 6.1

4. Map view of reminders TEST		
Reference Number	Description	Pass
4.1	A map view of all the reminders a user has created as well as their current position. To change from the list view to a map view, there would be a navigation bar at the bottom of the app, which is always there so you can quickly change how you view your reminders.	Yes
4.2	Click on a pointer to reveal information about it and options. It will give you the title as well as the option to move the	No

	pointer to another location by dragging it across the map.	
4.3	On the map view when you click a pointer you will be given the option to edit the reminder, which will take you to the edit reminder page.	No

Test cases were drawn up from the original requirements. Each individual requirement was tested against and then given a pass or a fail. As can be seen in table 6.1 some aspects did fail. In the example the failures were from parts of the system that were removed. Testing the app in relation to the original requirements is a great way to methodically test everything out. All test cases can be seen in appendix E.

7. Evaluation

This part of the project focuses on how smoothly the project ran using the chosen methodology, evaluates the outcome and produces feedback from users experience of the app. in general the project ran very smoothly because of the strict timetable that was drawn up. It simply ensured that each feature that was to be included was done at the right time and didn't impact other parts.

7.1 Evaluate test/survey results

The following table is the results from the user survey that 12 people took part in.

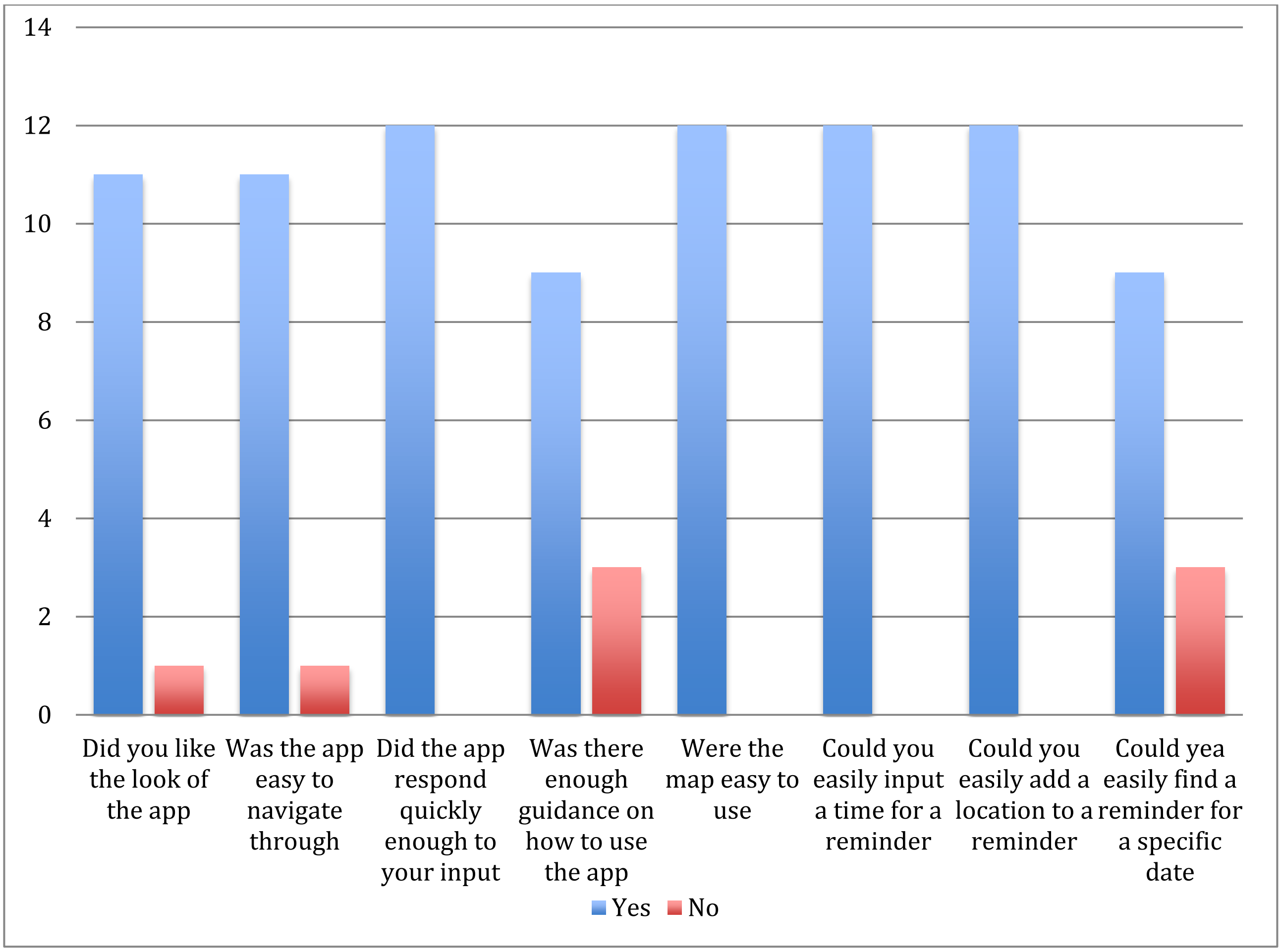


Figure 7.1 User server results

7.2 Evaluate project outcomes

Some of the initial requirements in the project were dropped as they were either no longer needed or else no longer a viable option. One such feature was a login screen. It was no longer needed because all of the data was being stored on the users device. Another feature that was dropped was the ability to organize reminders into editable lists. This requirement was dropped because of the results of the survey. The level of importance people give it didn’t match up to the amount of coding effort that would have been required for such a complex component. Both of these features could still be added in the future in the next version of the app.

The overall functionality of the app works really well and the location tracking is very accurate. The only disappointing part is PhoneGap's lack of functionality to let this location tracking happen even when the app is closed. This is the one part that is stopping the app.

Unfortunately due to a lack of time some of the functionality on the chronological list view screen had to be dropped. This included the slider to pick a month an date. It doesn't have any real impact to the functionality but it would have been a good feature to have. It is definitely one of the first features that would be added to the next version of the app

7.3 Evaluate the methodology

The chosen methodology was agile, but due to the schedule of the project it was forced to be a lot more like a waterfall approach. Instead of incrementally building sections of the app and testing it the process was forced to follow the waterfall method. This didn't hinder the overall development and what was achieved but it did disrupt the schedule by forcing things to happen in a different order. Luckily using the waterfall method didn't have any adverse effects on the project.

7.4 Evaluate the plan

The plan and timeline for work that had been drawn up worked fairly well. There was a few features that were dropped that had originally bee factored into the timeline so this freed up more time for other parts. This was a good decision to drop some parts like the login screen and the slider within the time view screen as it meant that some parts that ended up being more complicated, like the geolocation could be worked on for longer. Due to the nature of how the project is assessed through multiple reports, these weren't properly factored into the timeframe and so were slightly underestimated in terms of how long they would take to complete. It meant loosing a bit of development time but because it was properly managed it didn't have a huge impact.

8. Conclusion

This conclusion will look at summarizing the report as well as reflecting on all of the work that was done. Finally it will look at what future work and improvements could be made. It is necessary to take a step back and reflect on all work that has been done to properly appreciate it and critique it.

8.1 Summary

This report has look at the overall development of the native IOS app Spotnote. It has went through the initial research of technologies and frameworks to the design and implementation. Finally came the testing and evaluation of how the project and implementation went.

8.2 Reflection

The overall process of research, setting up requirements and following a specific methodology through development was a very good process to follow and ensured the whole project ran very smoothly. Research enabled the

The aim of showing that an IMD student with only web technology knowledge can create a native app was pretty much fulfilled. The only part that hinders the proper use of the app is that it no longer tacks location once the user closes it. This could easily be remedied by showing a message alerting a user to this fact, but for desired functionality is that tracking wouldn't stop. I believe that any sort of native app could be built using web technologies and PhoneGap, which is a great prospect for all IMD students. PhoneGap is definitely a technology that I would like to experiment with and use in the future.

I have always wanted to create a native IOS app as it is a massive market but the thought of learning objective-c was very daunting so the overall project has been a great personal success and I will continue to develop the application and hope to release it in the app store in the future.

8.3 Suggest future work

The first feature that I would like to include in the future would be the slider in the time view page so that specific dates can be easily navigated to. The feature had to be dropped late in development due to time constraints so it would be nice to add it back in.

The next feature to add back in would be the ability to continue tracking location even when the app is closed. Although it would require writing an objective-c plugin it would still be worth it.

A very quick addition to the app would be the ability to tap on a map pointer on the map view screen and see the name of the reminder. Unfortunately again this was left out due to time constraints and other more important features taking longer than expected.

The final feature that would be nice t have is a website that syncs with the app. so all reminders can be managed from the website and will automatically appear on the app. it is a very big feature and would require quite a lot of development but it would definitely be worth it.

9. References

LocationMinder - Location Based Reminders on the App Store on iTunes. 2014. LocationMinder - Location Based Reminders on the App Store on iTunes. [ONLINE] Available at: https://itunes.apple.com/gb/app/locationminder-location-based/id383031428?mt=8. [Accessed 11 April 2014].

Snowman – Checkmark 2 – To-do App for iPhone. 2014. Snowman – Checkmark 2 – To-do App for iPhone. [ONLINE] Available at: http://builtbysnowman.com/checkmark/. [Accessed 11 April 2014].

Sublime Text . 2014. Sublime Text . [ONLINE] Available at: http://www.sublimetext.com/2. [Accessed 11 April 2014].

Notepad++ Home. 2014. Notepad++ Home. [ONLINE] Available at:http://notepad-plus-plus.org/. [Accessed 11 April 2014].

Coda 2. 2014. Coda 2. [ONLINE] Available at: https://panic.com/coda/. [Accessed 11 April 2014].

PhoneGap | Home. 2014. PhoneGap | Home. [ONLINE] Available at:http://phonegap.com/. [Accessed 11 April 2014].

jQuery. 2014. jQuery. [ONLINE] Available at: http://jquery.com/. [Accessed 11 April 2014].

Framework7 - Full Featured HTML Framework For Building iOS7 Apps.
2014. Framework7 - Full Featured HTML Framework For Building iOS7 Apps. [ONLINE] Available at: http://www.idangero.us/framework7/. [Accessed 11 April 2014].

Mobile App Development Framework. JavaScript and HTML5. Download Sencha Touch Free. | Sencha Touch | Products | Sencha. 2014. Mobile App Development Framework. JavaScript and HTML5. Download Sencha Touch Free. | Sencha Touch | Products | Sencha. [ONLINE] Available at:http://www.sencha.com/products/touch/. [Accessed 11 April 2014].

ChocolateChip-UI - A Mobile Web Framework in HTML5, CSS & Javascript.
2014. ChocolateChip-UI - A Mobile Web Framework in HTML5, CSS & Javascript. [ONLINE] Available at: http://chocolatechip-ui.com/. [Accessed 11 April 2014].

iOS 7 Design Resources. 2014. iOS 7 Design Resources. [ONLINE] Available at: https://developer.apple.com/library/ios/design/. [Accessed 11 April 2014].

Summary of Don Norman's Design Principles. 2014. Summary of Don Norman's Design Principles. [ONLINE] Available at:http://www.csun.edu/science/courses/671/bibliography/preece.html. [Accessed 17 April 2014].

iOS Human Interface Guidelines: Bars. 2014. iOS Human Interface Guidelines: Bars. [ONLINE] Available at:https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/Bars.html#//apple_ref/doc/uid/TP40006556-CH12-SW1. [Accessed 17 April 2014].

Free source code hosting for Git and Mercurial by Bitbucket. 2014. Free source code hosting for Git and Mercurial by Bitbucket. [ONLINE] Available at: https://bitbucket.org/. [Accessed 17 April 2014].

Grunt: The JavaScript Task Runner. 2014. Grunt: The JavaScript Task Runner. [ONLINE] Available at: http://gruntjs.com/. [Accessed 17 April 2014].

Xcode - What's New - Apple Developer. 2014. Xcode - What's New - Apple Developer. [ONLINE] Available at: https://developer.apple.com/xcode/. [Accessed 17 April 2014].

10. Appendix

Section A – IOS Reminders App Survey

IOS Reminders app survey

Using a scale of 0= not important at all to 5 = very important please rate the following statements in regards to notes and reminders

	0	1	2	3	4	5
The use of maps to view an event location						
Speed of an app to load and respond to your inputs						
Login functionality						
Organizing notes into lists						
Organizing notes based on time due						
Searchable location when looking for a location on a map						
The use of both time and location to remind						
The use of simple forms to input data						
Gesture based interactivity						
The ability to edit a note after its made						

Section B – Volere Snow Cards

Requirement Number: 1

Title: Login

Description: Login functionality so the user has to enter their username and password before they can view their reminders (phone will keep you logged in)

Rationale: Every user has their data (reminders) stored in a database so they are able to login from any device. This will also keep all of their data secure

Fit Criteria: Login screen with a username and password field. It will also have a register option and a forgot password option

Dependences: A database will need to be designed and built which has the functionality to store the users login details and their reminders.

Requirement Type: Functional

Requirement Number: 2

Title: Create/edit reminders

Description: The ability for the user to add a reminder or else edit a reminder that has already been created.

Rationale: The main functionality of the app is to create reminders. You should also be able to edit them incase something needs to be changed

Fit Criteria: A smooth and easy way to add and edit reminders

Dependences: #1 (Log in capability)

Requirement Type: Functional

Requirement Number: 3

Title: Manage reminders (Check off or delete)

Description: View all of you reminders at the same time in a list. You can then individually check off a reminder as done or else delete a reminder if you no longer want to have it

Rationale: You need to be able to check it off because there is no point in still having it if you have already done the task. This is the same for the delete. If a reminder is no longer relevant then you will obviously want to delete it

Fit Criteria: A screen that has a list of reminders that have been created by the user. Any of these reminders can individually be removed/deleted or checked off as done.

Dependences: #2 (Create reminder capability)

Requirement Type: Functional

Requirement Number: 4

Title: Map view of reminders

Description: A map that has all of your reminders on it as pin points as well as your current location. This means you can see where the closest reminder is and go and do it. Each pinpoint on the map can be pressed to show all the details of the reminder.

Rationale: Because the main focus of my app is to have reminders at specific locations, I think it would be good to have a map that shows you these locations so you can easily visualize what you have to do and where.
Fit Criteria: A map with points on it at each reminder location
Dependences: #2 (Create/edit reminders)
Requirement Type: Functional

Requirement Number: 5
Title: Timeline view of reminders
Description: A list view of all of a users reminders ordered into the date and time they are due so that the user is able to see what they need to do next
Rationale: It makes sense for the user of the app to see their reminders in this order so that they can do the most urgent one first
Fit Criteria: A list view of all reminders sorted by date and time
Dependences: #2 (Create reminder capability)
Requirement Type: Functional
Supporting Materials:

Requirement Number: 6
Title: Website
Description: A website version of the app that will have all of the same functionality, but it means you can view and manage your reminders on a desktop computer or laptop.
Rationale: It gives the user and extra way to set up and manage reminders. A lot of people would like to schedule their day from a computer
Fit Criteria: A fully functional website that has all of the same functionality as the app but is optimized for the desktop experience.
Dependences: have a complete app as well as a database
Requirement Type: Functional

Section C – Detailed Requirements

1. Login		
Reference Number	Title	Description
1.1	Login	Enter your username and password into two separate fields and then press

		a login button. If the login is successful you are taken to the landing page of the app. The app will then remember you as a user and keep you logged in at all times on your phone.
1.2	Register	A register button that will take you to a form where you enter a username email address and a password. Once you have entered all of the information you will be redirected to the landing page.
1.3	Forgot Password	If you forget your password you press a button that will take you to a new screen where you input your email address and a new password will be sent to that account for you to login
1.4	Login with Facebook	Press this button to allow the app access to the users Facebook account and it will use the information from Facebook to create a user profile and log them in

2. Create/ Edit Reminders		
Reference Number	Title	Description
2.1	Create	To create a reminder there will be a simple button on the landing screen that will take you to a new form. This button will probably be in a consistent position on all screens in the app. The form that it brings you to will have three different fields to fill in. 1. Firstly there will be the title of the reminder, which will appear in home screen along with all the other reminders. 2. Then there will be a time and date input. This will use the generic data and time input scroller that is used in IOS 3. Lastly there will be a location input. Once you click on this input I would like it to reveal a map. You can type a location in the field and search for it, which will show the location on the map. You can then move the marker If you want to another point or else search again

		You must fill in the title and either the location or data fields and then save the reminder. Once it is saved you are brought back to the main screen
2.1.1	Title input	An input field that takes the title value for a reminder. It is required.
2.1.2	Time input	This will only take a certain format of time and date, which is dependent on the iOS date/time picker. It is only required if there is no location entered
2.1.3	Location input	This will not be manually input by the user to avoid mistakes. It will be generated by a selection on a map. It is only required if there is no time inputted
2.2	Edit	User will click on the reminder they want to edit and will be brought to the same screen as the create reminder. But the input fields will be populated with the information of the item you have clicked. You can then edit any other the information and resave it as the same item so it doesn't create a new reminder.

3. Manage Reminders		
Reference Number	Title	Description
3.1	Complete	Once a user gets alerted about the reminder and they have done whatever task it is, they will want to check it off. It will have a checkbox or button option, which when clicked checks it of and removes it from the overall list. I would like to have instead of just a checkbox or button some sort of gesture (sliding your finger across the screen) but I don't know how feasible that is at the moment, as I need to do more research into it.
3.2	Delete	If you want to delete an item there will be a button to give you this option. But again I would like to incorporate a gesture to delete instead of a button. It would give a really nice user experience to have this but it isn't necessary for the app to work
3.3	Alert	Once a reminder is triggered by either time or location the user is alerted by a vibration sound and text saying what it

		is. They will then have the option to go into the app to view the reminder, so they can check it of, view it in more detail or even edit it so that it will happen at a later time or location.

4. Map view of reminders

Reference Number	Title	Description
4.1	Map	A map view of all the reminders a user has created as well as their current position. To change from the list view to a map view, there would be a navigation bar at the bottom of the app, which is always there so you can quickly change how you view your reminders.
4.2	Pointer Options	Click on a pointer to reveal information about it and options. It will give you the title as well as the option to move the pointer to another location by dragging it across the map.
4.3	Edit	On the map view when you click a pointer you will be given the option to edit the reminder, which will take you to the edit reminder page.

5. Timeline view of reminders

Reference Number	Title	Description
5.1	TimeLine of reminders	Reminders will be listed in order of time due. So the next reminder to send the user an alert will be listed first. The reminders will be separated by dates. This is to clearly so that you can visually group them and makes a long list easier to understand.
5.2	Manage Reminders	This is just the same ability to check of edit and delete a task on this screen

6. Website

Reference Number	Title	Description
6.1	Website	Depending on time constraints, I would like to make a website that used your same login information. It would have all

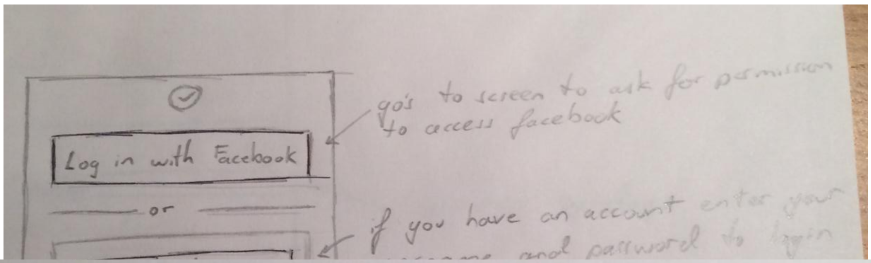
		of the same functionality as the app such as create, edit and map view. It would just pull all of the users information from the database where it is stored. The app and the website would automatically sync. So if you delete a reminder on one it gets deleted from the other.

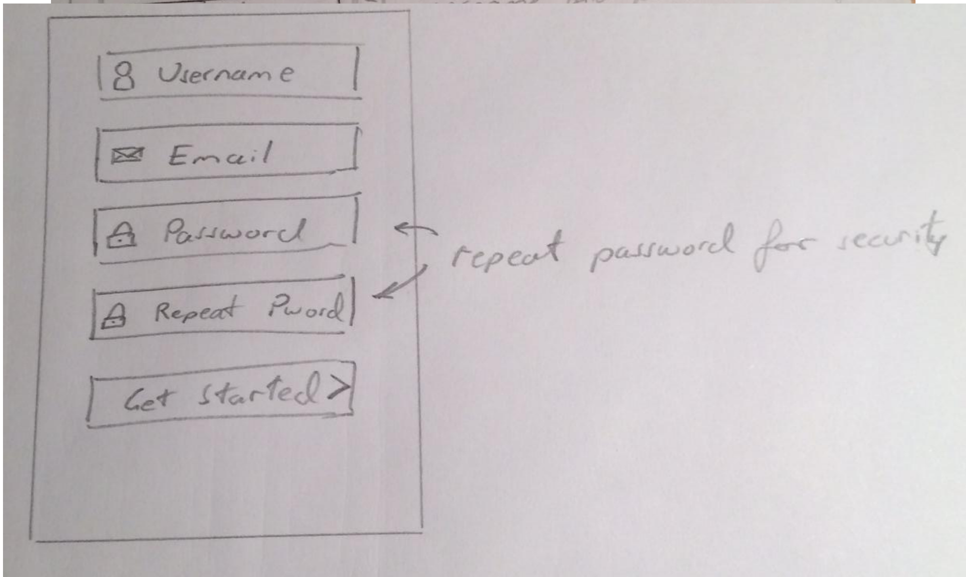
7. Design

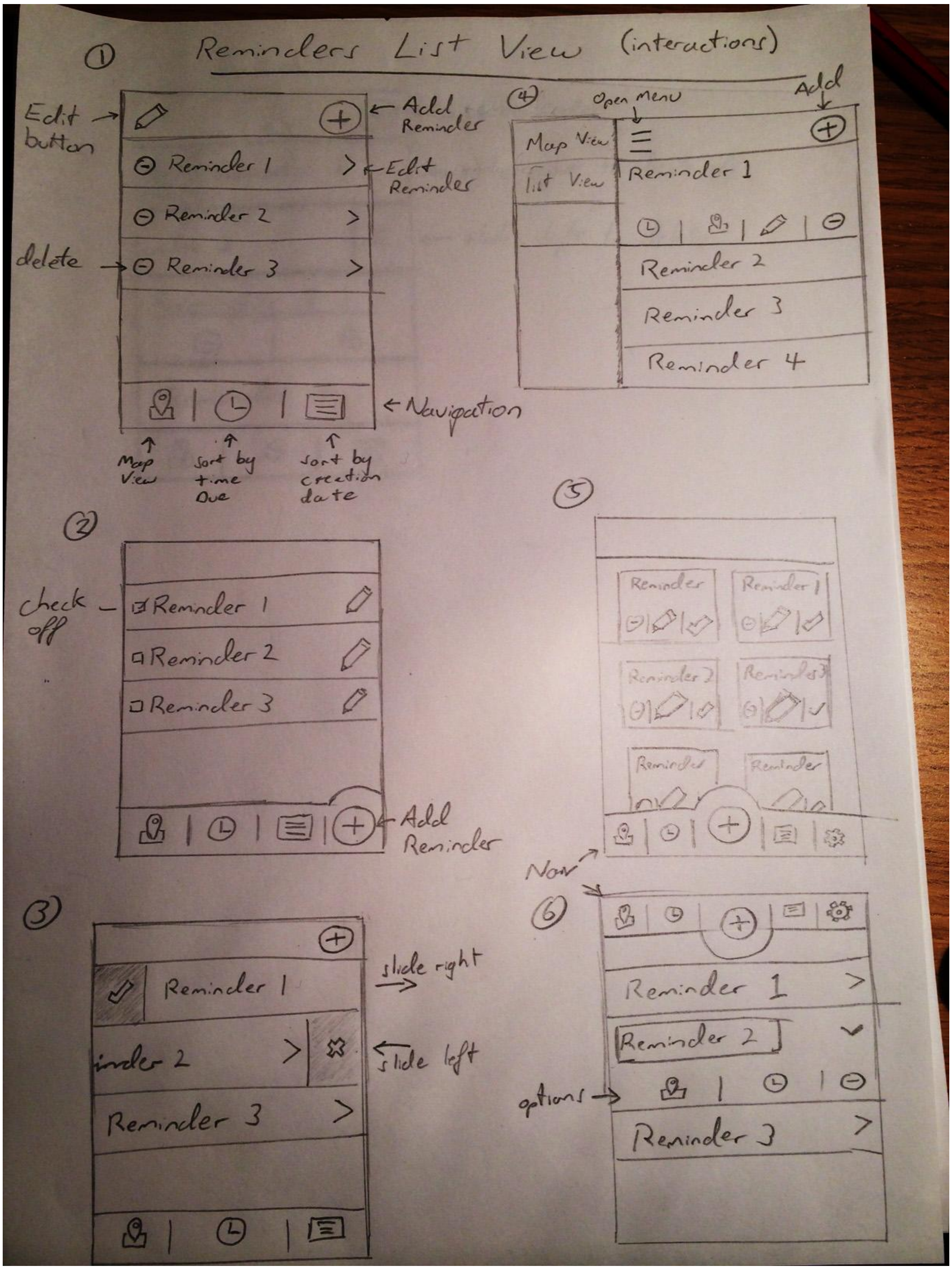
Reference Number	Title	Description
7.1	Transparency	There is a new design trend creeping in to IOS7, which is translucency or semi transparency. It has translucent header bars, translucent overlays and translucent backgrounds. Its nice to let translucency hint at the content that is behind it
7.2	Simplistic Icons	Icons in IOS7 have changed to be even more simplistic as they are now just outlines without fill. It gives a really nice clean design.
7.3	Geometric shapes	I really like the use of geometric shapes and geometric patterns. Not for main visual elements or layouts but for more subtle things like backgrounds of a button. It adds a new dimension to design.
7.4	Sans-serif	Thin sans-serif fonts are another new design craze that is again very simplistic and clean looking.
7.5	Flat design	What I mean by flat design is no shadows or gradients. That is very web 2.0 and I don’t really want that. It has to be in keeping with IOS7. Even so, there is still a feeling of depth within IOS7! It’s more of a layered realistic depth. This is sort of explained in the IOS weather app where its background changes depending on the weather and looks very realistic. Its not shadows to create depth it’s a more realistic feeling of depth
7.6	Negative space	Use of negative space is crucial to make the app look as easy and simple to use as possible. But I don’t want to overdo this where the app ends up looking bare.
7.7	Borderless buttons	Web 2.0 really pushed the idea of big in your face buttons with bright colors,

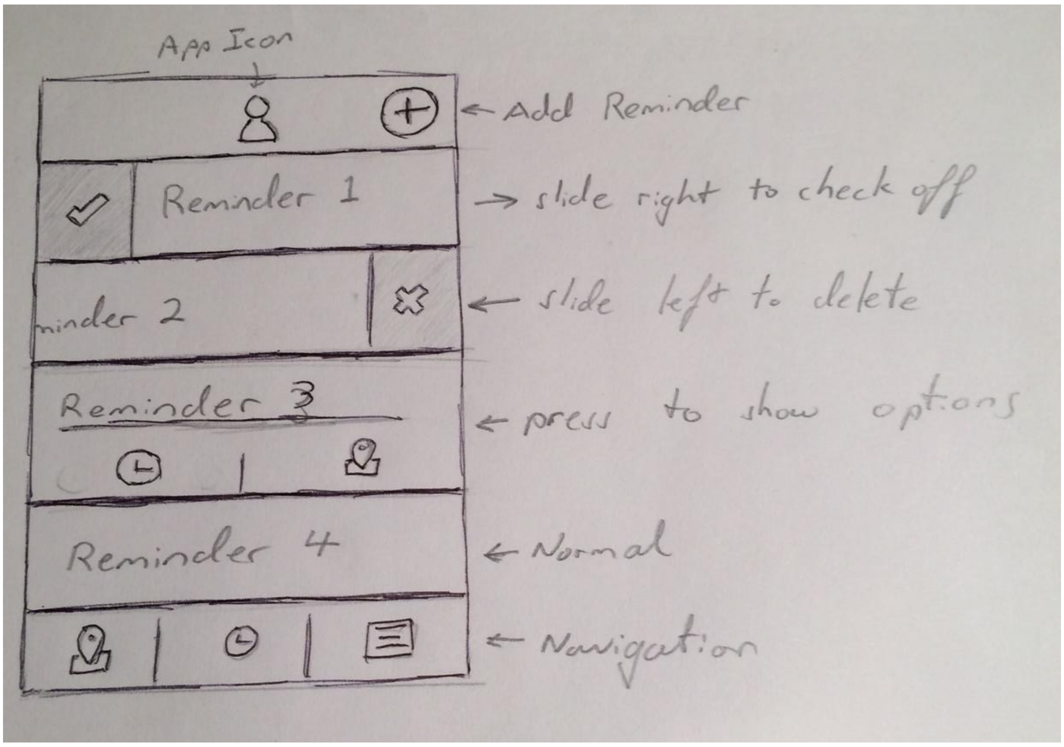
		gradients borders and shadows. It now feels like everything has just been thrown at buttons. I think simplicity is key here. Who is to say you cant just represent a button cleverly with only text?
7.8	Color palette	I don't want a too varied a color palette. I would like to stick with just one main color and use it as this helps to build a good brand identity.

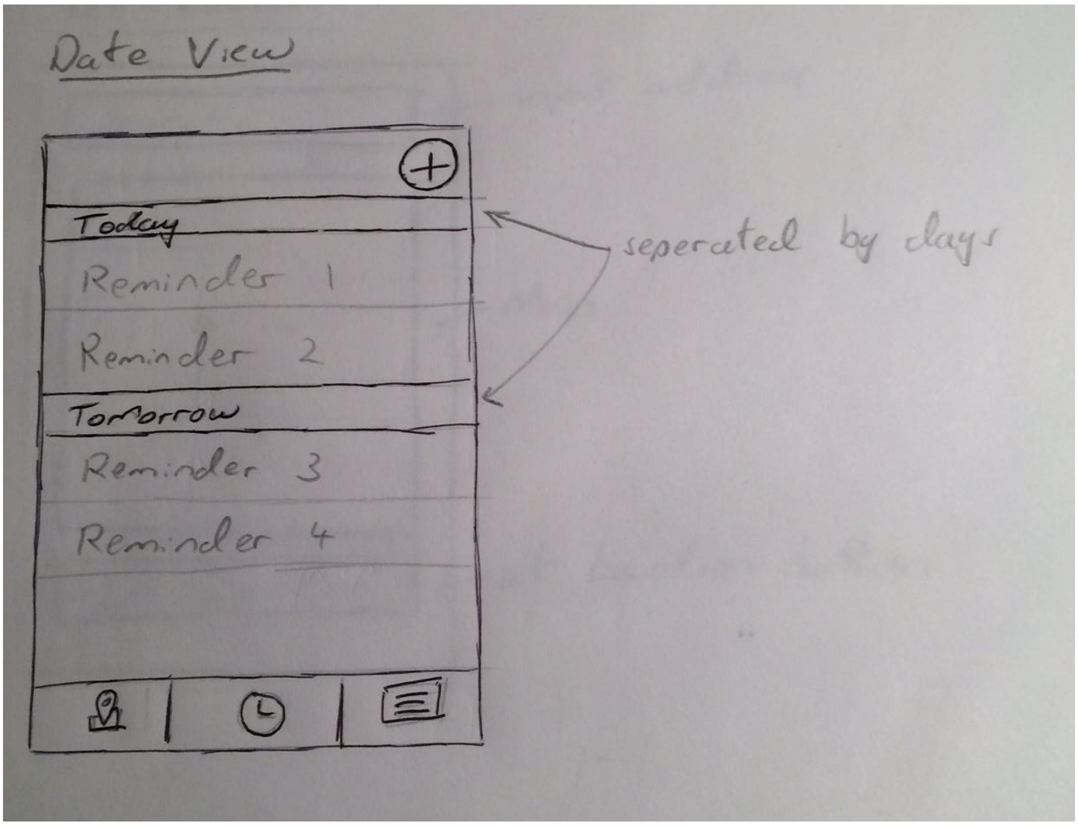
Section D

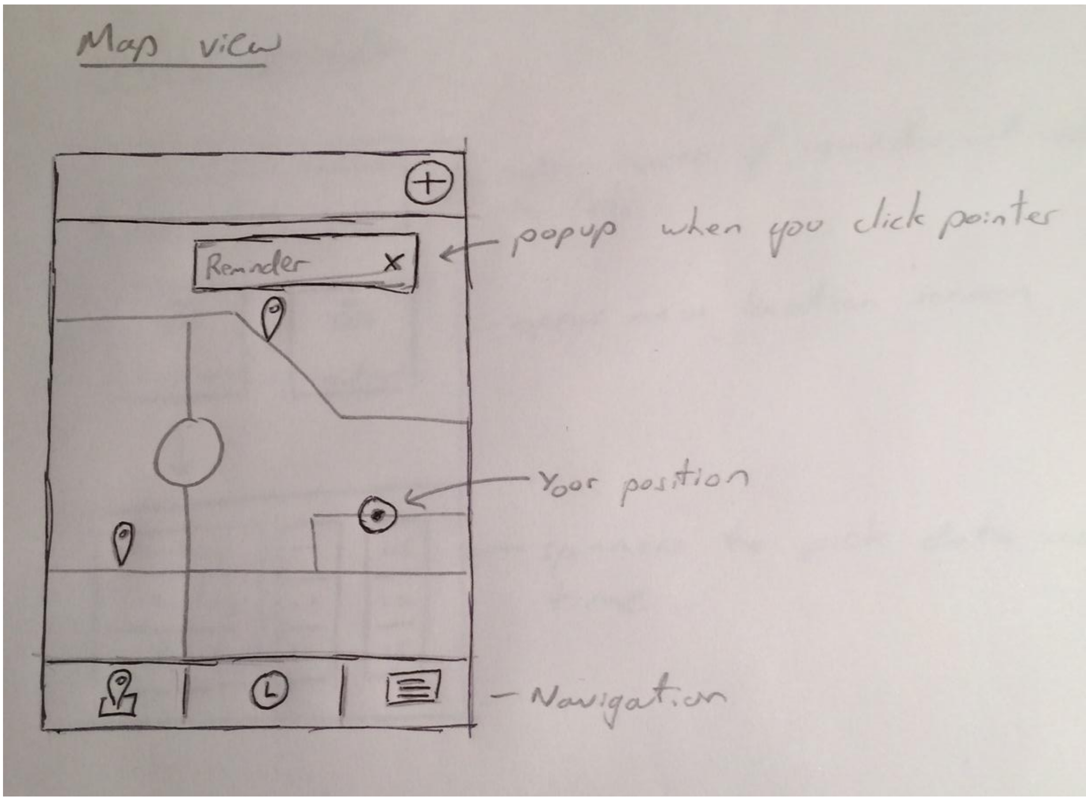

Log in with Facebook
or
go's to screen to ask for permission to access facebook
if you have an account enter your
username and password to login

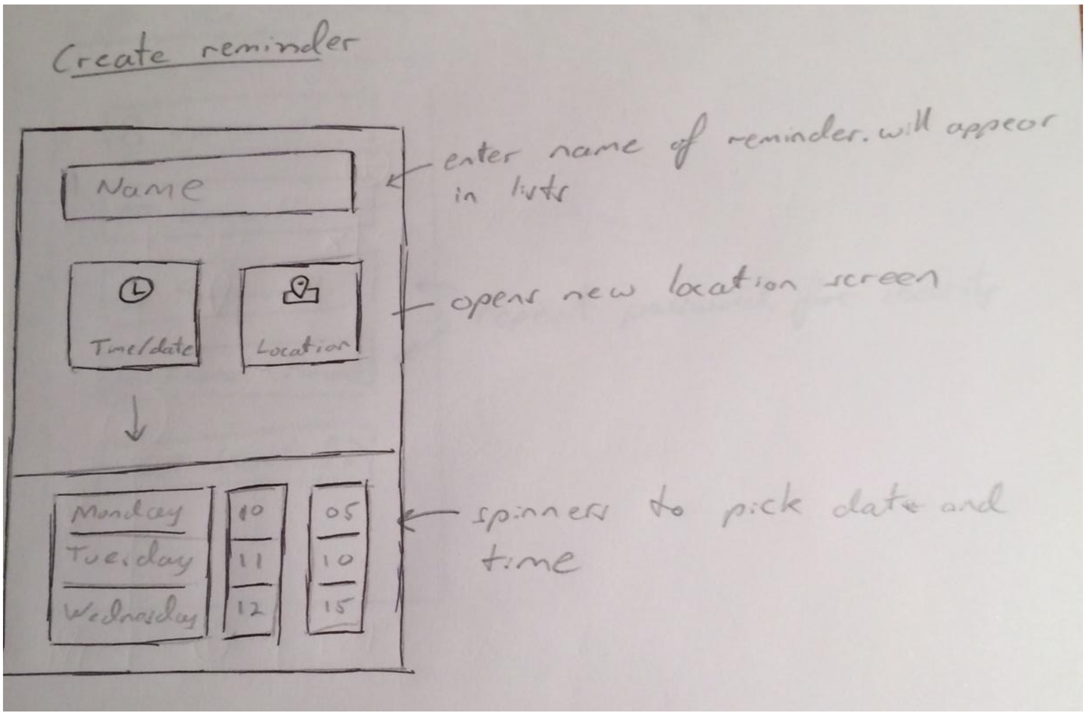


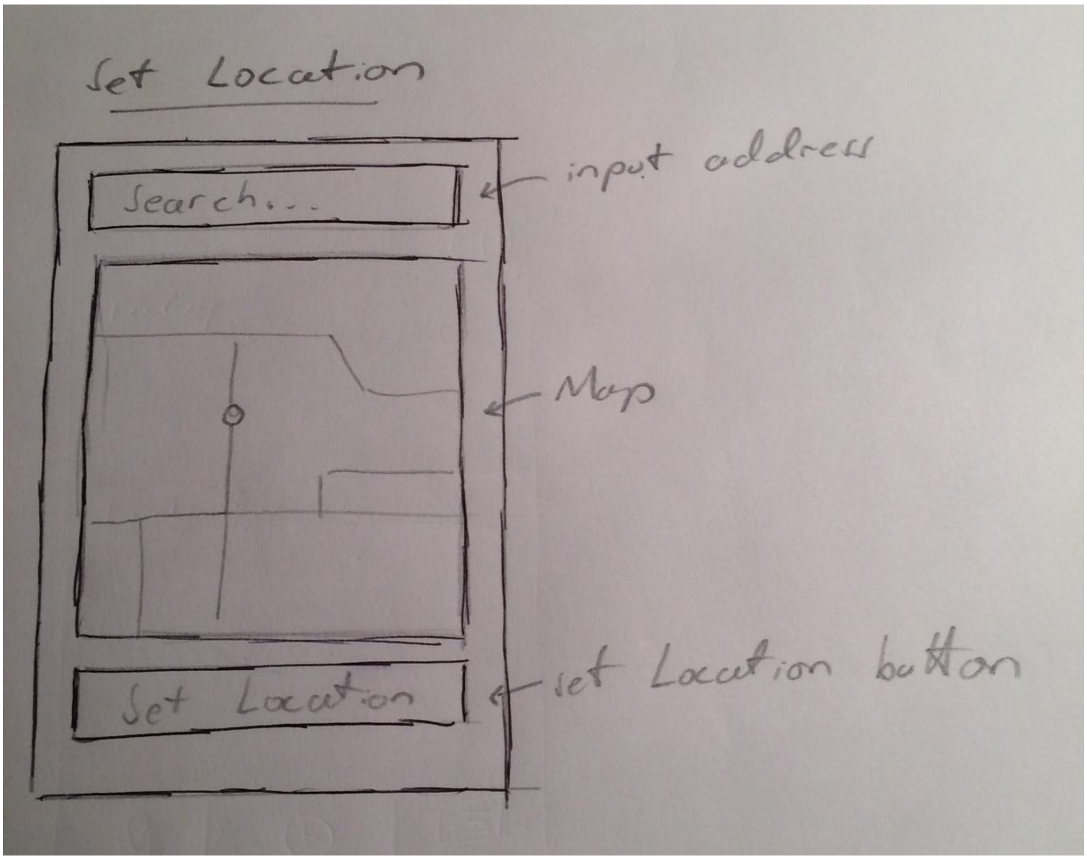

① Reminders List View (interactions)
Edit button
Add Reminder
Reminder 1
Reminder 2
Reminder 3
Edit Reminder
delete
Navigation
Map View
sort by time Due
sort by creation date
④
Open Menu
Add
Map View
List View
Reminder 1
Reminder 2
Reminder 3
Reminder 4
②
check off
Reminder 1
Reminder 2
Reminder 3
Add Reminder
⑤
Reminder
Reminder 1
Reminder 2
Reminder 3
Reminder
Reminder
Nav
③
Reminder 1
slide right
inder 2
slide left
Reminder 3
⑥
Reminder 1
Reminder 2
options
Reminder 3


App Icon
Add Reminder
Reminder 1
slide right to check off
minder 2
slide left to delete
Reminder 3
press to show options
Reminder 4
Normal
Navigation


Date View
Today
Reminder 1
Reminder 2
Tomorrow
Reminder 3
Reminder 4
seperated by days


Map view
Reminder
X
popup when you click pointer
Your position
Navigation


Create reminder
Name
enter name of reminder. will appear in lists
Time/date
Location
opens new location screen
Monday
Tuesday
Wednesday
10
11
12
05
10
15
spinners to pick date and time


Set Location
Search...
input address
Map
Set Location
set Location button

Section E – Test results

1. Login TEST

Reference Number	Description	Pass
1.1	Enter your username and password into two separate fields and then press a login button. If the login is successful you are taken to the landing page of the app. The app will then remember you as a user and keep you logged in at all times on your phone.	No
1.2	A register button that will take you to a form where you enter a username email address and a password. Once you have entered all of the information you will be redirected to the landing page.	No
1.3	If you forget your password you press a button that will take you to a new screen where you input your email address and a new password will be sent to that account for you to login	No
1.4	Press this button to allow the app access to the users Facebook account and it will use the information from Facebook to create a user profile and log them in	No

2. Create/ Edit Reminders TEST

Reference Number	Description	Pass
2.1	To create a reminder there will be a simple button on the landing screen that will take you to a new form. This button will probably be in a consistent position on all screens in the app. The form that it brings you to will have three different fields to fill in. 1. Firstly there will be the title of the reminder, which will appear in home screen along with all the other reminders. 2. Then there will be a time and date input. This will use the generic data and time input scroller that is used in IOS 3. Lastly there will be a location input. Once you click on this input I would like it to reveal a map.	Yes

	You can type a location in the field and search for it, which will show the location on the map. You can then move the marker If you want to another point or else search again You must fill in the title and either the location or data fields and then save the reminder. Once it is saved you are brought back to the main screen	
2.1.1	An input field that takes the title value for a reminder. It is required.	Yes
2.1.2	This will only take a certain format of time and date, which is dependent on the iOS date/time picker. It is only required if there is no location entered	Yes
2.1.3	This will not be manually input by the user to avoid mistakes. It will be generated by a selection on a map. It is only required if there is no time inputted	Yes
2.2	User will click on the reminder they want to edit and will be brought to the same screen as the create reminder. But the input fields will be populated with the information of the item you have clicked. You can then edit any other the information and resave it as the same item so it doesn't create a new reminder.	Yes

3. Manage Reminders TEST		
Reference Number	Description	Pass
3.1	Once a user gets alerted about the reminder and they have done whatever task it is, they will want to check it off. It will have a checkbox or button option, which when clicked checks it of and removes it from the overall list. I would like to have instead of just a checkbox or button some sort of gesture (sliding your finger across the screen) but I don't know how feasible that is at the moment, as I need to do more research into it.	Yes
3.2	If you want to delete an item there will be a button to give you this option. But again it would like to like to incorporate	Yes

	a gesture to delete instead of a button. It would give a really nice user experience to have this but it isn't necessary for the app to work	
3.3	Once a reminder is triggered by either time or location the user is alerted by a vibration sound and text saying what it is. They will then have the option to go into the app to view the reminder, so they can check it of, view it in more detail or even edit it so that it will happen at a later time or location.	Yes

4. Map view of reminders TEST

Reference Number	Description	Pass
4.1	A map view of all the reminders a user has created as well as their current position. To change from the list view to a map view, there would be a navigation bar at the bottom of the app, which is always there so you can quickly change how you view your reminders.	Yes
4.2	Click on a pointer to reveal information about it and options. It will give you the title as well as the option to move the pointer to another location by dragging it across the map.	No
4.3	On the map view when you click a pointer you will be given the option to edit the reminder, which will take you to the edit reminder page.	No

5. Timeline view of reminders TEST

Reference Number	Description	Pass
5.1	Reminders will be listed in order of time due. So the next reminder to send the user an alert will be listed first. The reminders will be separated by dates. This is to clearly so that you can visually group them and makes a long list easier to understand.	Yes
5.2	This is just the same ability to check of edit and delete a task on this screen	Yes

6. Website TEST

Reference Number	Description	Pass
6.1	Depending on time constraints, a website that used your same login information. It would have all of the same functionality as the app such as create, edit and map view. It would just pull all of the users information from the database where it is stored. The app and the website would automatically sync. So if you delete a reminder on one it gets deleted from the other.	No

7. Design TEST

Reference Number	Description	Pass
7.1	There is a new design trend creeping in to IOS7, which is translucency or semi transparency. It has translucent header bars, translucent overlays and translucent backgrounds. Its nice to let translucency hint at the content that is behind it	Yes
7.2	Icons in IOS7 have changed to be even more simplistic as they are now just outlines without fill. It gives a really nice clean design.	Yes
7.3	Geometric shapes and geometric patterns. Not for main visual elements or layouts but for more subtle things like backgrounds of a button. It adds a new dimension to design.	Yes
7.4	Thin sans-serif fonts are another new design craze that is again very simplistic and clean looking.	Yes
7.5	Flat Design yet still keeping depth	Yes
7.6	Use of negative space is crucial to make the app look as easy and simple to use as possible. But I don’t want to overdo this where the app ends up looking bare.	Yes
7.7	Web 2.0 really pushed the idea of big in your face buttons with bright colors, gradients borders and shadows. It now feels like everything has just been thrown at buttons. Simplicity is key.	Yes

7.8	Simple color palette	Yes

Section F – finished product usability survey

IOS Reminders app survey

Please answer the following questions after tying out the reminders app Spotnote

	Yes	No
Did you like the look of the app		
Was the app easy to navigate through		
Did the app respond quickly enough to your input		
Was there enough guidance on how to use the app		
Were the map easy to use		
Could you easily input a time for a reminder		
Could you easily add a location to a reminder		
Could yea easily find a reminder for a specific date		