**Architecture**

**Main Architectural Requirements**

* **Our fundamental requirement was for our project to be available as a mobile application for both iOS and Android:** We chose to build the front end of this application with React Native due to it providing native cross-platform development for both systems, as well as supported libraries for addressing the following requirements
* **The application must contain the ability to take photos that the user can then view via a “Diary” page:** Upon consultation with our client, **these photos are to be stored locally on the users’ phone**. The diary page will therefore read from local storage and display all photos taken in chronological order using the timestamps stored with pictures metadata.
* **The application must contain an email page where the user can select a photo that is then loaded into there email application of choice to be sent to there GP:** Again, this requires reading photos from local storage. There is no sending of emails within the application, the selected photo is simply loaded as an attachment to a new email on the users preferred email service e.g. Gmail
* **The application must contain a patient information page that can be updated dynamically with other resources:** The resources in question will be text, links to webpages etc stored within a database on a server provided by our client. Our application will use HTTP GET requests to request and receive any updates from the server and apply any changes to the page. We have decided to use NodeJS to provide the server-side logic as we require many I/O operations without the need of heavy server-side computing. The resources themselves will be stored on a NoSQL database as we do not require multiple related tables, simply one table storing fields containing text, time stamp of last edit (so we can distribute new changes), and possibly links, images etc.

**Overall structure**

1. **Front end:** Contains the following: A home page, a patient information page displaying information that is dynamically updated from a server, a camera page that can writes images to local storage, a diary page and email page, both of which will read images from local storage.
2. **Local storage:** Stores all photos taken within the application
3. **Server:** Stores database containing all the resources to be displayed on the patient information page. Sends any resources added to the database after the timestamp of the clients most recent update to said page.

**Home Page**

**Diary and Email Pages**

**Patient Resource Page**

**Camera Page**

Send photo(s)

Write photo

GET response with any new resources

GET request for new resources

Request photo(s)

**Server**

**Local Storage**

Stores photos

Stores NoSQL database of resources for patient information page