

159.333 Individual Project

REPORT

CHI FUNG STANLEY YEUNG (15316357)

MASSEY UNIVERSITY

Supervisor: Dr Andrew Gilman

2016 Semester 2

Acknowledgement

Firstly, I would like to thank Dr Andrew Gilman for supervising me and providing me with this project idea. He also gave me support and ideas that the project is based upon. I would also like to thank him for the patience he provided as I always forget to update him on the progress of the project.

I would like to thank Dr Martin Johnson for providing me with the opportunity in participating in this paper.

Finally, I would like to thank Professor Chris Scogings, Dr Andre Barczak and Dr Peter Kay for the support before the project started.

Abstract

This is a final year project for a Computer Science Degree.

With each passing year, more powerful and efficient mobile devices are released. People are relying more and more on their smartphones to solve their everyday problems. Some include using maps to locate their destinations, receiving up to date news on social media, even reduce the consumption of paper by taking notes with a phone. Phones now are much more than just a phone.

The purpose of the project is to allow better efficiency in monitoring individual health and fitness by the use of mobile devices. This can be achieved by allowing users to record the amount of exercise and food consumption through a device that is simple to use.

Contents

Acknowledgement	1
Abstract	2
1 Project Information	5
1.1 Project Title	5
1.2 Project Justification	5
1.3 Project Aim & Objective	5
1.4 Project Scope	5
2 Introduction	6
2.1 About the project	6
2.2 Working Environment	6
2.2.1 Hardware	6
2.2.2 Software	6
2.2.3 Logo	6
3 Backend	7
3.1 Serverless Architecture	7
3.1.1 Traditional Client-Server Architecture	7
3.2 Amazon Web Services (AWS)	8
3.3 DynamoDB	9
3.4 Simple Storage Solution (S3) Bucket	9
3.5 Application Programming Interface (API) Gateway	9
3.5.1 Representational State Transfer (RESTful) API	9
3.5.2 Structure	10
3.5.3 Implement methods	10
3.6 Amazon Lambda	12
3.7 Identity and Access Management (IAM)	12
4 Mobile Application	13
4.1 Ionic 2	13
4.2 UI Design	13
4.3 Specifications	14
4.4 Design Choices	14
4.4.1 Page	14
4.4.2 Providers	15
4.4.3 Persistence	16

4.4.4 Camera and Gallery	16
4.5 Deployment	17
4.5.3 Icon and Splash Screen.....	17
4.5.2 Testing	17
4.5.3 Build	19
5 Web Application	20
5.1 Hypertext Preprocessor (PHP)	20
5.2 UI Design.....	20
5.3 Specifications.....	21
5.4 Design Choices.....	21
5.4.1 Model View Controller (MVC)	21
5.4.2 Persistence	22
5.4.3 Object Orientated Programming (OOP).....	22
5.4.4 cURL Function.....	23
5.4.5 Sorting Algorithm	24
5.5 Deploying.....	24
5.5.1 Debugging	24
5.5.2 Testing	24
5.5.3 Hosting	25
6 Improvements	26
6.1 Backend	26
6.1.1 Responses.....	26
6.1.2 Authentication Security.....	26
6.1.3 Security against attacks.....	26
6.2 Mobile Application	26
6.2.1 Refresh	26
6.2.2 Additional pages.....	27
6.2.3 Preference settings	27
6.3 Web Application	27
6.3.1 More functionality	27
6.4 Additional Features	27
6.4.1 Standardised styling	27
6.4.2 Calorie and BMI calculation	27
7 Conclusion	28

1 Project Information

1.1 Project Title

Implement cross platform Mobile and Website application to utilise a cloud based backend storage.

1.2 Project Justification

The project is based upon an idea from a Food Science PHD student. The requirement is to produce an application in a mobile environment to record logs, photos and dates of diet and exercise routines of patients.

1.3 Project Aim & Objective

Construct a hybrid mobile application for logging descriptions and pictorial files to a cloud storage and monitor with a website application.

1.4 Project Scope

This project includes a client mobile application for Android and iOS based on Ionic 2, an administrator web application based on PHP, and a “serverless” RESTful API approach using AWS Amazon Services.

2 Introduction

2.1 About the project

The project can be divided into 3 main components: the backend, the mobile application and the website application. The 3 components

2.2 Working Environment

2.2.1 Hardware

The project will be produced using personal devices. These include:

- Macbook Pro mid 2010
- Samsung Galaxy J5 - Android version 5.1.1
- OMEN HP Laptop – Windows 10 Home, Intel i7 CPU, 16GB RAM

2.2.2 Software

Software used as testing environments include:

- PHP - Vagrant
- Android – NOX, Bluestacks
- iOS – Xcode

Sublime text 3 is used to produce most of the code

Most diagrams will be produced by DIA (<http://dia-installer.de/>)

Adobe Photoshop is used to produce icon, splash screen, logo, fav icon

2.2.3 Logo

The logo is designed to be clean and simple. healthy colours to match the purpose of the project. Font used is Rockwell Extra Bold, colour used is #84f365 - Light Green and #ffffff - White



3 Backend

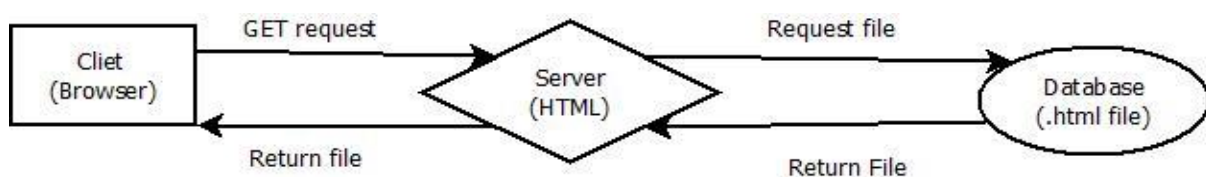
The backend of the project uses a serverless approach. Most functionalities are included in the backend such as check user authentication, retrieve and upload logs, storage. With the help of RESTful APIs, the backend can be easily implemented to multiple platforms that has internet access.

3.1 Serverless Architecture

Serverless is a term to describe an application that runs its back end functionalities through a device which is fully maintained and managed by a third party. This device is stateless event-triggered. Stateless means that each request sent to the device is treated individually. Event-triggered means that the device only performs an action if an event occurs, events can include requests, file modification, database delete record, etc. One way to think of this is that a device that provides a Backend function as a service.

3.1.1 Traditional Client-Server Architecture

The traditional client-server architecture is a server device or program which provides a specific set of functionalities that a client device or program uses. An example of this would be a HTML server.

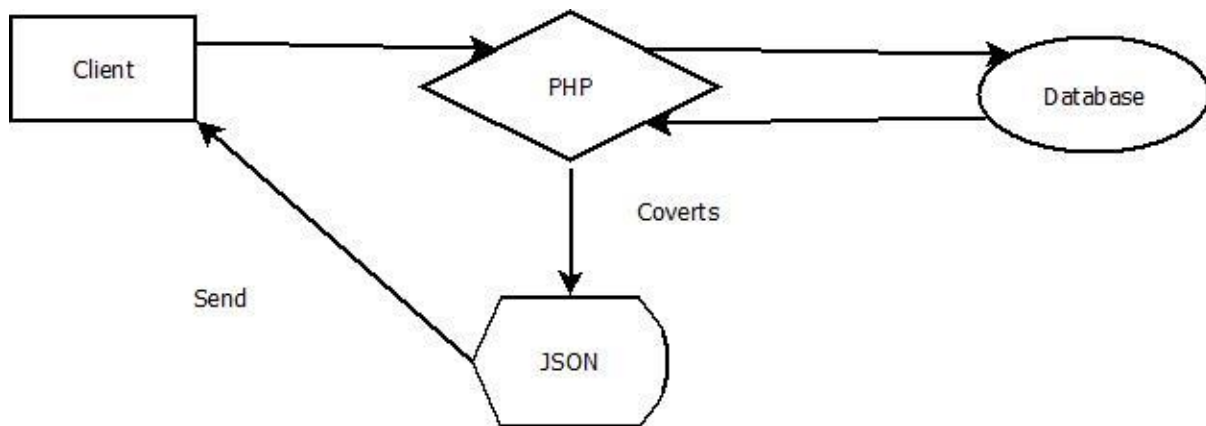


The HTML server acts as a request receiver which the client can make requests to obtain or modify information on the server database.

An advantage of this implementation is that the programmer has full control of the server. This ensures that the server can obtain all the functionalities that the client requires, while also avoid excessive functionalities so that it contains only functionalities that the client may use. This also allows the programmer to implement a function in a way that is more efficiently for that specific task.

Although the freedom is attractive, there are a few drawbacks. First, there is reusability. Server programs with similar functionalities are often duplicated. Taking database data conversion as an example, most database transfer a set of data

through data-interchange languages such as XML or JSON. The program that converts the database information to metadata can be written by the programmer. This results in multiple versions of the same code, which is not efficient in resources such as time and storage.

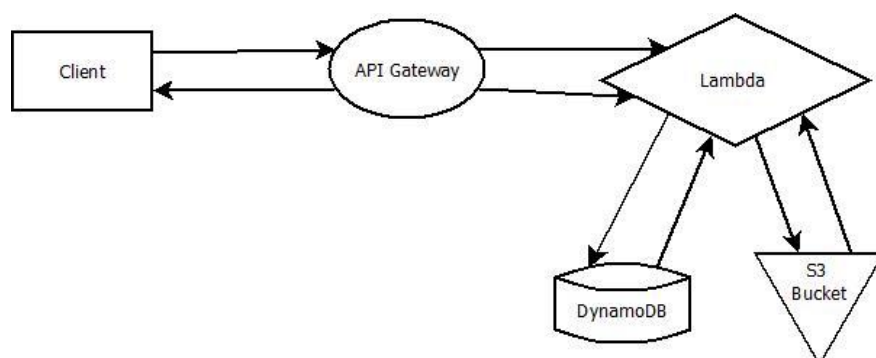


Another problem that may arise is hard ware cost. Concurrent servers require high costs to setup and maintain. This may not be achievable for an average developers, complicated implementations may be needed which increases the complexity of the server, making it harder to manage.

3.2 Amazon Web Services (AWS)

The AWS is a serverless approach. It is a cloud computing platform that allows a stateless, event-triggered environment to process requests as a service. The main reason to use this approach is due to free setup and low maintenance cost. The charges are calculated based on the amount of on-demand throughput towards the server, which means that the cost accumulate only when it is being used. It is also easier to implement as it has most of the default services ready to use.

AWS consists of many services. In this project, only some of the services are used to fulfil the application's backend functionalities.



3.3 DynamoDB

DynamoDB is a database service in AWS. It acts as a database server similar to a Structure Query Language (SQL) server. DynamoDB is a key-value store NoSQL database, meaning that it is not modelled in means of relations between tables. It uses a dictionary to store the keys of each entry which allows uniqueness. The tables do not have default fields initialized. They are added upon the first record insert.

Two tables were used for the project:

User	Log
* <u>user_name</u> String	* <u>log_id</u> Number
*user_pass String	°user_name String
*user String (JSON)	°log String (JSON)

The primary key field can take 3 types: String, Number and Binary, secondary fields has the option to take more types. Something to note is that I used JSON strings to store “user” and “log”. This is done to allow easier retraction of data. The field “user” contains “email” and “name”. The field “log” contains “date”, “description” and “image”. The date is stored as a String, this is because the supported date type changes the time to UTC. The “image” field stores Amazon Resource Name (ARN) which is a URL that points to the image which is stored in the S3 bucket.

3.4 Simple Storage Solution (S3) Bucket

S3 Bucket the Amazon Service that is used as a storage for objects. A bucket is a logical unit that stores a list of data and metadata for the objects. For this project, a bucket is created to store the images for logs. The bucket URL is unique, meaning that the name of the bucket is unique. The bucket name for this project is called: dev333-diet-image.

3.5 Application Programming Interface (API) Gateway

An API is a set of protocols or methods to allow easier program implementation when building an application. The API Gateway in Amazon catches requests and responds with result of the appropriate functions or resources that was requested.

3.5.1 Representational State Transfer (RESTful) API

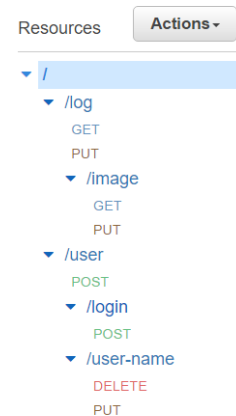
RESTful API is used to describe APIs that handle HTTP requests. These requests may include GET, PUT, POST, DELETE. RESTful architecture aims to efficiently and reliably

process requests through the World Wide Web(WWW). Also, development of applications based on RESTful APIs can be easily expanded due to the ease of reuse code.

AWS API Gateway is built based on RESTful API.

3.5.2 Structure

The main resource structures for the API is “log” and “user”. There is no logout function since applications has their own persistence functions, which will be discussed later. The “image” resource is there since Lambda functions are not made to perform large file transfers, therefore, S3 bucket uploads are made directly through from the API Gateway to S3 bucket.



3.5.3 Implement methods

There are multiple ways to modify the AWS API Gateway. The 2 ways that were used to develop this project was:

- SwaggerHub (<https://swaggerhub.com/>)
- AWS Console Web Application

SwaggerHub is a website application that is used for building APIs. SwaggerHub works closely with AWS and can therefore upload APIs directly to the AWS API Gateway upon providing authorisations to the application. SwaggerHub supports JSON/YAML languages which are widely used for metadata. The API that was built in this project was implemented using YAML. AWS also has the option to export existing APIs to Swagger.

```

/users/user_name:
  put:
    tags:
      - user
    summary: Updated user
    description: ""
    operationId: updateUser
    produces:
      - application/json
    parameters:
      - in: path
        name: username
        description: name that need to be deleted
        required: true
        type: string
      - in: body
        name: body
        description: Updated user object
        required: false
        schema:
          $ref: "#/definitions/User"
    responses:
      "404":
        description: User not found
      "400":
        description: Invalid user supplied

```

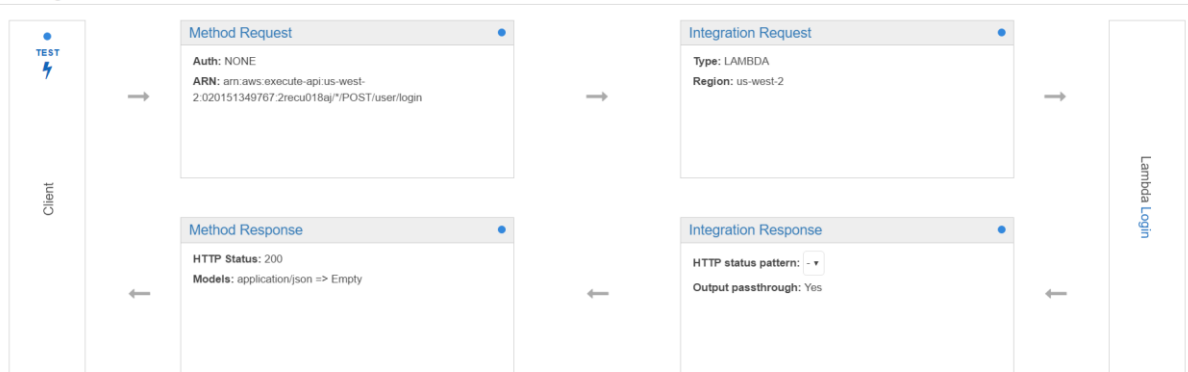
```

definitions:
  User:
    type: object
    properties:
      user_name:
        type: string
      user:
        type: string
      user_pass:
        type: string

```

AWS console has a Graphical User Interface(GUI) representation of the process of upon receiving the request. The console provides options to link requests to integration points for AWS services. It also allows modification to responses such as HTTP status code response and response mapping.

/user/login - POST - Method Execution



Another very useful function that the GUI provides is test. One can modify test contents to test the API, integration points, responses or other functionality within the console.

← Method Execution /user-name - GET - Method Test

Make a test call to your method with the provided input

Path

No path parameters exist for this resource. You can define path parameters by using the syntax **{myPathParam}** in a resource path.

Query Strings

user_name

Value

Headers

No header parameters exist for this method. You can add them via Method Request.

Stage Variables

No [stage variables](#) exist for this method.

Request Body

Request Body is not supported for GET methods.

⚡ Test

After deployment, the AWS Console provides a URL address that applications to use to access the API method.

Diet - POST - /user

Invoke URL: <https://2recu018aj.execute-api.us-west-2.amazonaws.com/Diet/user>

3.6 Amazon Lambda

Lambda is used as a model to handle user requests through and from the API Gateway. The API Gateway receives a request, and will call the Lambda function that meets the request. This request query and body is also mapped to the Lambda function. An example is the “login” request. The parameters in the body are “user_name” and “user_pass”, which is then integrated into the Lambda function. The Lambda Function will then respond the data to the API Gateway, which is responded to the client.

Lambda supports Python or Node.js. For this project, all functions are implemented with Node.js 4.3.

```

1 var AWS = require('aws-sdk');
2 var docClient = new AWS.DynamoDB.DocumentClient();
3
4 exports.handler = function(event, context) {
5
6     var user_name = event.user_name;
7     var user_pass = event.user_pass;
8
9     var params = {
10         TableName: "User",
11         Key: {
12             "user_name": user_name
13         }
14     };
15
16     var cb = function(err, data) {
17         if(err) {
18             console.log('error on Login: ',err);
19             context.done('Server error', null);
20         } else {
21             if(data.Item && data.Item.user) {
22                 if(user_pass == data.Item.user_pass){
23                     context.done(null, data.Item.user);
24                 } else {
25                     context.done('Username and password does not match', {});
26                 }
27             } else {
28                 context.done('Username ' + user_name + ' not found', {});
29             }
30         }
31     };
32 }

```

3.7 Identity and Access Management (IAM)

Amazon allows multiple users for a single account. For the scope of this project, one administrator user was created which has 2 roles:

- APIGatewayLambdaExecRole
- APIGatewayS3Role

The Lambda Execution role has been setup to have full access permissions to DynamoDB and Lambda. The S3 Role has full access permissions to all S3 bucket files. These permissions extend to the buckets, database and functions that are authorized to this AWS account.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-west-2:020151349767:function:python*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeStream",
        "dynamodb:GetRecords",
        "dynamodb:GetShardIterator",
        "dynamodb:ListStreams"
      ],
      "Resource": "arn:aws:dynamodb:us-west-2:020151349767:table/User/stream/*"
    }
  ]
}

```

4 Mobile Application

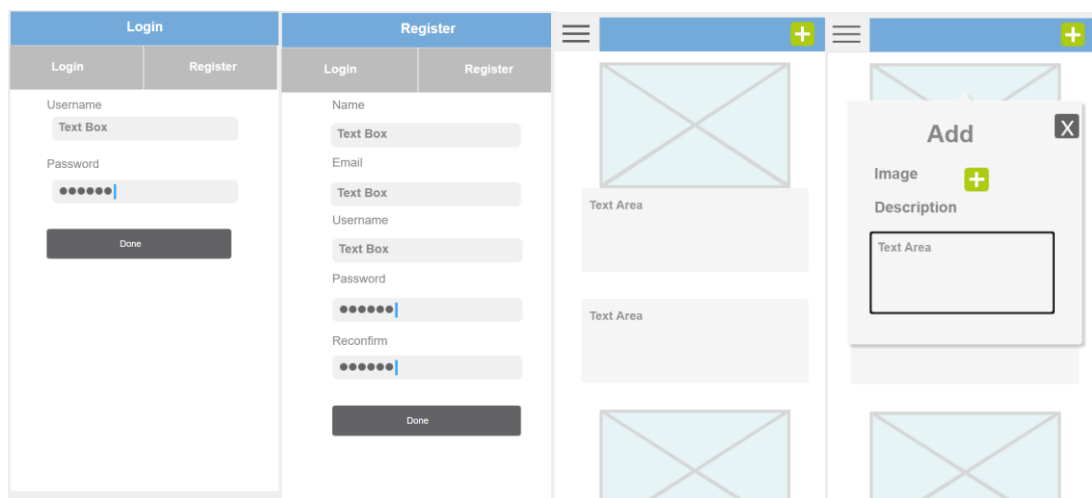
The mobile application is developed in a hybrid environment. This differs to the tradition native approach where one function will need to be implemented multiple times to allow cross platform support.

4.1 Ionic 2

Ionic 2 is a hybrid development environment for mobile devices. It can be deployed to all major mobile platforms (Android, IOS, Windows Phone). It is based on HTML, CSS and JavaScript (Angular JS). Development is simple due to cross platform support using simple development environment.

4.2 UI Design

The main design of the application consists of 4 pages. Mock-up produced by FluidUI (<https://www.fluidui.com/>)



The login and register page will be displayed if the user is not logged in. After login, the home page consists of only 2 interactive buttons, add and logout. Add page is a popup page. There is one more page called “tabs” that is called to display login and register page.

4.3 Specifications

A simple mobile application that provides the following functionalities:

- 1) Log-in / Sign-up / Log-out
- 2) Persistent login
- 3) View list of logs on home screen
- 4) Add logs
- 5) Choose image from camera or gallery

4.4 Design Choices

4.4.1 Page

A page consists of 3 files: .html, .sass, .ts. The html is the view of the page, what contents the page shows. Sass is the style of the page, how the contents are displayed. The ts file is very similar to a JS file, it acts as a controller using @Component to define parameters the page displays. It also contains functions t

Pages were added using the built-in generate function.

```
ionic\Diet>ionic generate page login
```

Various default components were used while creating the layout of the application(<https://ionicframework.com/docs/v2/components/>).

```
<ion-list>
  <ion-item>
    <button ion-button icon-only (click)="takePicture()"><ion-icon name="camera"></ion-icon></button>
    <button ion-button icon-only (click)="openGallery()"><ion-icon name="photos"></ion-icon></button>
    <img [src]="image" *ngIf="image" />
  </ion-item>

  <ion-item>
    <ion-label floating>Description</ion-label>
    <ion-textarea rows="4" [(ngModel)]="description"></ion-textarea>
  </ion-item>
</ion-list>

<button full ion-button (click)="saveItem()">Save</button>
```

4.4.2 Providers

Providers are services or injections to implement methods or functions. They are similar to a class or interface in Java. There are 2 providers which were implemented into the project: “User” and “ProviderService”. The “User” provider is used to store the metadata of the current logged in user. It has functions to login and logout of the current user.

```
login(user_name, user_pass){
  let temp = JSON.stringify($.extend({}, user_name, user_pass));
  this.providerservice.postHTTP("login", temp).then((new_user) => {
    this.storage.set('name', new_user.name);
  }, (err) => {
    console.log(err);
  });
}

logout(){
  this.storage.clear();
}
```

The “ProviderService” provider has the functions to communicate with the backend RESTful API by using the GET, POST and PUT requests.

```
getHTTP(resource) {
  return this.http.get(baseUrl + resource)
    .map(res => res.json())
    .catch(this.handleError);
}

postHTTP(resource, params) {
  let body = JSON.stringify(params);
  let headers = new Headers({ 'Content-Type': 'application/json' });
  let options = new RequestOptions({ headers: headers });
  return this.http.post(baseUrl + resource, body, options)
    .map(res => res.json())
    .catch(this.handleError);
}

putHTTP(resource, params) {
  let body = JSON.stringify(params);
  let headers = new Headers({ 'Content-Type': 'application/json' });
  let options = new RequestOptions({ headers: headers });
  this.http.put(baseUrl + resource, body, options)
    .catch(this.handleError);
}
```

The providers work closely with JSON due to the fact that the API Gateway request JSON format input and responses with JSON format body. Converting between JSON and JavaScript variables is important for a successful interaction between the application and the API.

4.4.3 Persistence

Persistent login is achieved using local storage to store the user's information. This is done by using a SQLite Cordova plugin.

```
>cordova plugin add cordova-sqlite-storage --save
```

The provider "User" is used to store this information, the "User" provider uses the SQLite "Storage" plugin.

```
getName() {  
    return this.storage.get('name');  
}  
  
length() {  
    return this.storage.length();  
}
```

The app.component.ts file is the controls the application when it starts. The "rootPage" variable decides which page is displayed on application start-up.

4.4.4 Camera and Gallery

The add page allows the user to add a photo to the log. This is optional as the log can still be saved with only the description. The user has 2 options to add a photo: camera, gallery. Both options uses the cordova camera plugin.

```
>ionic plugin add cordova-plugin-camera
```

The images are converted into base64 strings for easier handling and upload. The images are resized to a set dimension for easier display option.

```
takePicture(){  
    Camera.getPicture({  
        destinationType: Camera.DestinationType.DATA_URL,  
        quality: 100,  
        targetWidth: 1000,  
        targetHeight: 1000,  
        encodingType: Camera.EncodingType.JPEG,  
        correctOrientation: true  
    }).then((imageData) => {  
        // imageData is a base64 encoded string  
        this.image = "data:image/jpeg;base64," + imageData;  
    }, (err) => {  
        console.log(err);  
    });  
}  
  
openGallery(){  
    Camera.getPicture({  
        sourceType: Camera.PictureSourceType.PHOTOLIBRARY,  
        destinationType: Camera.DestinationType.DATA_URL,  
        quality: 100,  
        targetWidth: 1000,  
        targetHeight: 1000,  
        encodingType: Camera.EncodingType.JPEG,  
        correctOrientation: true  
    }).then((imageData) => {  
        // imageData is a base64 encoded string  
        this.image = "data:image/jpeg;base64," + imageData;  
    }, (err) => {  
        console.log(err);  
    });  
}
```

4.5 Deployment

4.5.3 Icon and Splash Screen

The icon (192x192 px) and Splash Screen (2208x2208 px) can be generated by ionic built in commands given that the images are in correct sizes:

```
>ionic resources
```

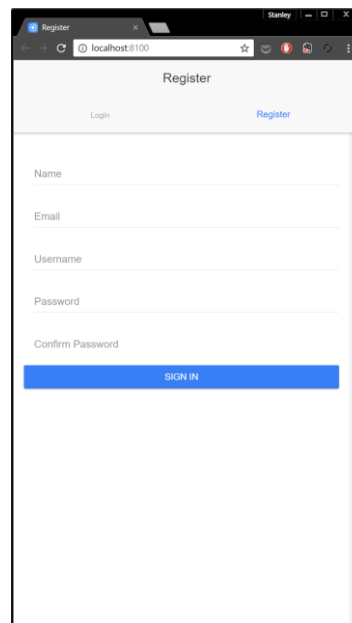
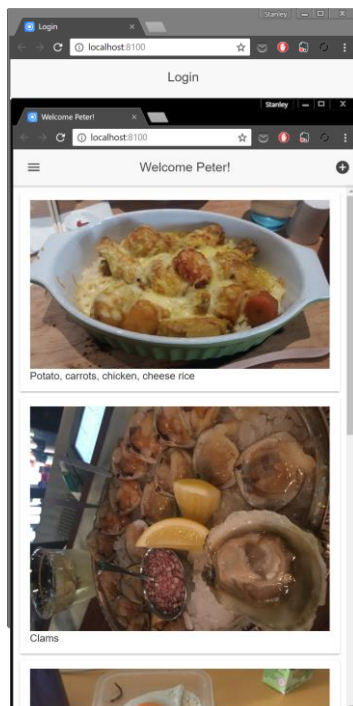
4.5.2 Testing



Testing was initially conducted with the built in “serve” option.

```
>ionic serve
```

This uses a standard browser as a tool for deploying the application as a GUI. Most of the functions can be tested through this approach.

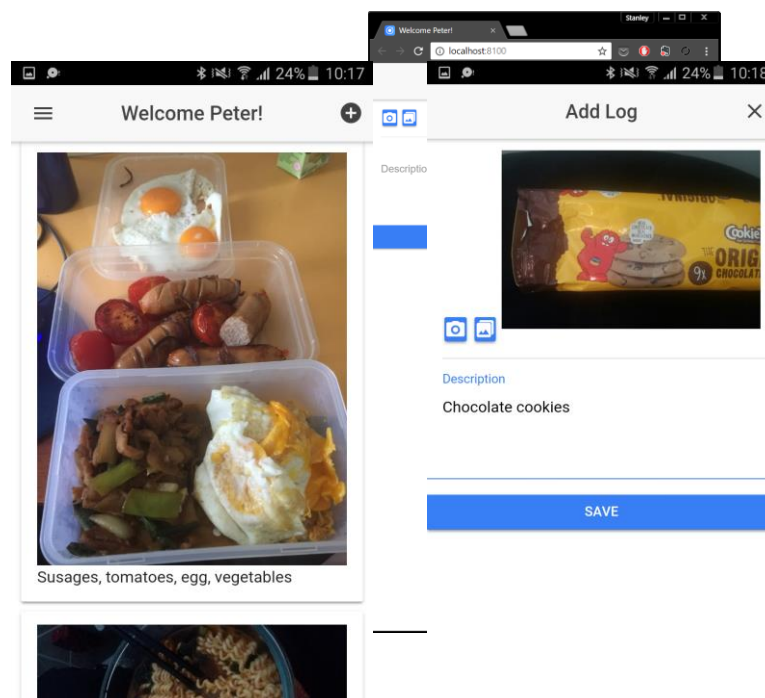


The browser used was Chrome. This approach does not support @ionic/native injections since they are device and platform specific functions. In this project, the function that requires the injection is the “Camera”. Another approach that was used was the built in “upload” option.

```
>ionic upload
```

This approach requires an ionic account. The application is uploaded to the ionic server which is then viewable using browsers or the ionic view mobile application:

- Android Google Play (<https://play.google.com/store/apps/details?id=com.ionic.viewapp&hl=en>)
- iOS App Store (<https://itunes.apple.com/nz/app/ionic-view/id849930087?mt=8>)



The device used for this approach was a Samsung Galaxy J5. Although it is available in both platforms, unfortunately I do not acquire a iOS device. Therefore, I used the simulator testing method which ran successfully. The method consists of using a computer running the OSX operating system with Xcode, ionic and cordova installed. Then using the following ionic built in command:

```
$ ionic build ios
```

```
$ ionic emulate ios
```

4.5.3 Build

After testing has been conducted, the application can be prepared to for deployment build. Different platforms need to be dealt with individually.

For android, the apk needs to be assigned with a credential key. This key can be generated with the following command:

```
>keytool -genkey -v -keystore Diet.keystore -alias Diet -keyalg RSA -keysize 2048 -validity 10000
```

The keytool program will ask for credentials to be entered:

```
Enter keystore password:
Keystore password is too short - must be at least 6 characters
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]: Stanley
What is the name of your organizational unit?
[Unknown]: Massey University
What is the name of your organization?
[Unknown]: Massey University
What is the name of your City or Locality?
[Unknown]: Auckland
What is the name of your State or Province?
[Unknown]:
What is the two-letter country code for this unit?
[Unknown]: NZ
Is CN=Stanley, OU=Massey University, O=Massey University, L=Auckland, ST=Unknown, C=NZ correct?
[no]: y
Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 10,000 days
for: CN=Stanley, OU=Massey University, O=Massey University, L=Auckland, ST=Unknown, C=NZ
Enter key password for <Diet>
(RETURN if same as keystore password):
Re-enter new password:
[Storing Diet.keystore]
```

A Diet.keystore file will be produced, which is used to authenticate the application after it has been deployed onto Google play store. Then generate the .apk file using the built with ionic built-in commands:

```
>ionic build android
```

The .apk file can be installed onto different devices. Using NOX and Bluestack emulators, the apk file worked with various different screen sizes.

The iOS build requires the Apple Developer ID to be present before building, therefore, the Apple app still lacks abit of testing.

Both Android and iOS requires a paid developer account for the apps to be published.

5 Web Application

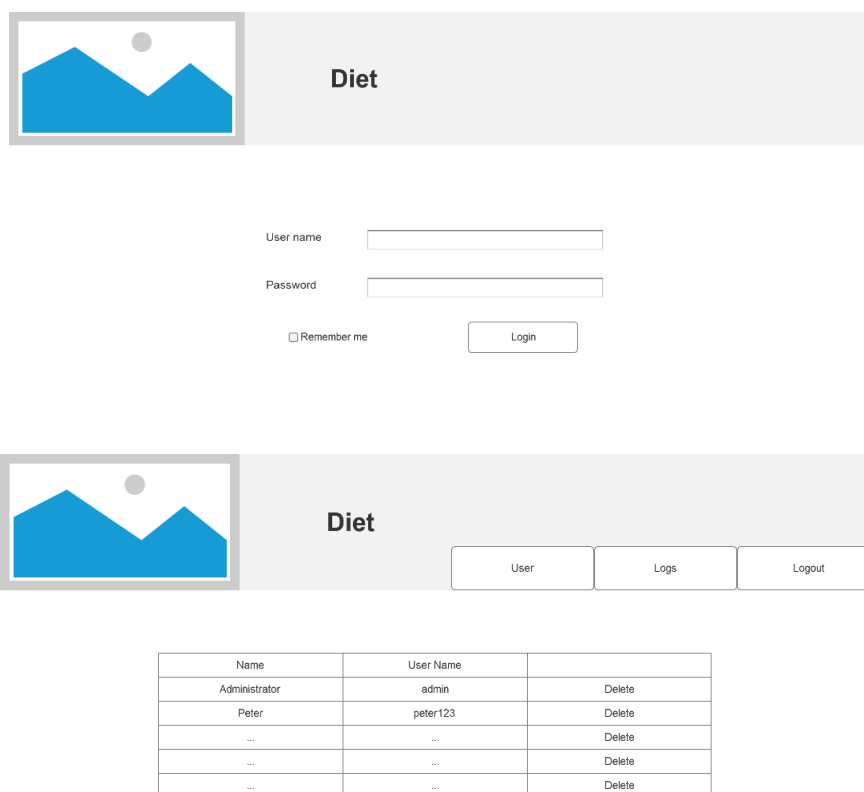
The purpose of the web application is for the administrator to monitor users on their logs. The administrator would also be able to make modifications such as edit and delete.

5.1 Hypertext Preprocessor (PHP)

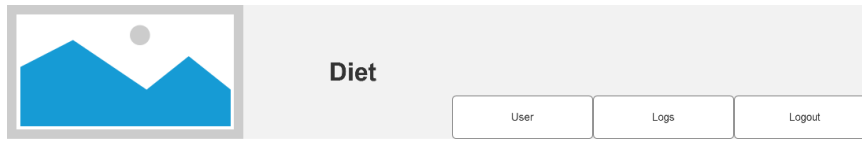
PHP is a server-sided scripting language. It is mainly used for processing large amounts of information located from the server. Although this web application will not be hosted or located within the server (serverless approach), PHP can make implementing table functions easier, such as sorting the table.

5.2 UI Design

The web application is simple. It has 3 web pages. The mock up was produced using Axure RP 8.



Name	User Name	
Administrator	admin	Delete
Peter	peter123	Delete
...	...	Delete
...	...	Delete
...	...	Delete



Date	Name	Description	
4 Nov 2016	Peter	Cookies	Delete
2 Nov 2016	Peter	Potato, tomato cheese	Delete
...	...		Delete
...	...		Delete
...	...		Delete

5.3 Specifications

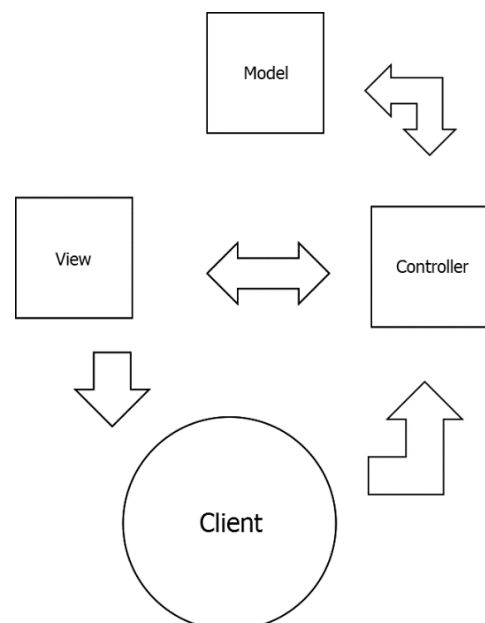
A simple mobile application that provides the following functionalities:

- 1) Log-in / Sign-up / Log-out
- 2) Persistent login
- 3) View list of users
- 4) View list of logs
- 5) Sort table
- 6) Delete entries

5.4 Design Choices

5.4.1 Model View Controller (MVC)

The bases of the application follow the MVC architecture. The Client first interacts with the controller, which will decide on what information from the model it needs and what view would the client interact with. The client sees the page from view and interacts with the controller. The application is built around this architecture.



5.4.2 Persistence

The application is made persistent using both Sessions and Cookies variables. I chose to use Cookies to store the logged in user to allow timing logins. I understand that this is dangerous as the cookie is stored on the client and may cause security issues. Due to this being an assignment, this is ignored, but if further development is done, this will be the first priority that needs to be attended.

There are also Session variables used. The base controller class is stored as a session variable. This is done so that when the client closes the browser and decides to prompt for the server again, all variables are reset and the client starts the session new. (With login if the cookie time limit is not exceeded.)

```
private function logInAction()
{
    if(isset($_POST['user_name'], $_POST['user_pass'])){
        $name = $this->user_model->login($_POST['user_name'], $_POST['user_pass']);

        if (isset($_POST['remb'])) {
            $time = time() + (60 * 60 * 24 * 30); // 30 days
            setcookie('remb', 'y', $time, "/");
        } else {
            $time = time() + (60 * 15); // 15 mins
        }

        setcookie('name', $name, $time, "/");

        header("Location: index.php");
    }
    include ("views/loginview.php");
}

private function logOutAction()
{
    $this->user_model->logout();

    setcookie('user_name', "", time() - 3600, "/");

    // remove all session variables
    session_unset();

    // destroy the session
    session_destroy();

    header("Location: index.php");
}
```

5.4.3 Object Orientated Programming (OOP)

The application follows OOP (object orientated programming). The controllers and models are contained as classed objects. There is also a user and log class used to store these objects, functions in these classes also allow modifying the information stored and or provide linkage to other functions or parts of this application.

The code snippet on the right shows the User class. It is a class for 1 User variable, which contains user name, name and email variables. It also contains functions to retrieve these variables and modify them.

```
class User
{
    private $user_name;
    private $name;
    private $email;

    public function __construct()
    {
        $this->$user_name = NULL;
        $this->name = NULL;
        $this->email = NULL;
    }

    public function getUser_name()
    {
        return $this->$user_name;
    }
    public function getName()
    {
        return $this->name;
    }
    public function getEmail()
    {
        return $this->email;
    }

    public function setUser_name($user_name)
    {
        $this->$user_name = $user_name;
    }
    public function setName($name)
    {
        $this->name = $name;
    }

    public function setEmail($email)
    {
        $this->email = $email;
    }
}
```

5.4.4 cURL Function

cURL functions in PHP allow modification to requests to compensate for different HTTP requests. Due to the fact that this project uses RESTful API as backend, cURL function calls are used to make the requests through PHP.

```
<?php
class RestAPI {
    private $base_url;

    public function __construct()
    {
        $base_url = "https://2recu018aj.execute-api.us-west-2.amazonaws.com/Diet";
    }

    public function handle_request($method, $resource, $request_parm)
    {
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);

        $request_url = $base_url . $resource;

        if ($method == 'GET') {
            $request_url .= '?' . http_build_query($request_parm);
        } else {
            if ($method == 'POST') {
                curl_setopt($ch, CURLOPT_POST, TRUE);
            } else {
                curl_setopt($ch, CURLOPT_CUSTOMREQUEST, $method);
            }
            curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($request_parm));
        }
        curl_setopt($ch, CURLOPT_HTTPHEADER, array('Accept: application/json'));
        curl_setopt($ch, CURLOPT_URL, $request_url);
        curl_setopt($ch, CURLOPT_HEADER, TRUE);
        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);

        $response = curl_exec($ch);
        $response_info = curl_getinfo($ch);

        curl_close($ch);

        $response_header = trim(substr($response, 0, $response_info['header_size']));
        return substr($response, $response_info['header_size']);
    }
}
```


5.4.5 Sorting Algorithm

PHP provides built in sorting algorithms. In this project, `usort()` was used since the arrays are built on objects. `Usort` allows user defined functions to be used to sort an array. Since the arrays are object arrays, user defined functions allow specific values to be sorted. This allows the table viewed to have the option to choose sorting fields and order.

```
public function sortBy($name, $field)
{
    $order = "DESC";

    if ($this->last_sort == $field && $this->last_order == $order)
        $order = "ASC";

    $res = $this->getTable($name);
    if ($field == "date") {
        if ($order == "DESC") {
            usort($res, function ($a, $b) {
                if (strtotime($a->getDate()) == strtotime($b->getDate())) return 0;
                return strtotime($a->getDate()) < strtotime($b->getDate()) ? 1 : -1;
            });
        } else {
            usort($res, function ($a, $b) {
                if (strtotime($a->getDate()) == strtotime($b->getDate())) return 0;
                return strtotime($a->getDate()) > strtotime($b->getDate()) ? 1 : -1;
            });
        }
    } else if ($field == "description") {
        if ($order == "DESC") {
            usort($res, function ($a, $b) {
                return strcmp($a->getDes(), $b->getDes());
            });
        } else {
            usort($res, function ($a, $b) {
                return strcmp($b->getDes(), $a->getDes());
            });
        }
    } else if ($field == "name") {
        if ($order == "DESC") {
            usort($res, function ($a, $b) {
                return strcmp($a->getName(), $b->getName());
            });
        } else {
            usort($res, function ($a, $b) {
                return strcmp($b->getName(), $a->getName());
            });
        }
    }

    $this->last_sort = $field;
    $this->last_order = $order;

    return $res;
}
```

5.5 Deploying

5.5.1 Debugging

The PHP server that I was testing on did not support cURL functions. A extension was installed:

```
$ sudo apt-get install curl php-curl
```

5.5.2 Testing

The web application was tested using the Vagrant box `laravel/homestead` as a virtual hosting server and then using a browser to make requests to this server. `Laravel homestead` was chosen because it is easy to install and host. Modifications to the virtual environment was also not hard due to it being popular and a lot of support was given in the forums.

The application is tested with Chrome, Internet Explorer, Safari, Opera and Firefox. These browsers were chosen since they are the more popular browsers.



Diet


Username*



Password*

☐ Remember Me (30 Days)



Name	User Name	
Administrator	admin	Delete
Peter	pete123	Delete
Simon	simon123	Delete



Date	Name	Description	Image	
4 Nov 2018	Peter	Potatoes, carrots, chicken, cheese rice		Delete
6 Nov 2018	Peter	Chicken		Delete

5.5.3 Hosting

To host this website live, a domain name is required. This can be purchased from Domain Name System (DNS) servers. The pricing for the names vary by the simplicity of the domain name and the extension.

Website hosting servers are also needed and can be hired from 3rd party hosting companies. Some companies provide this service free of charge with the downside of having advertisements pop up while browsing.

The application is ready to be launched if a DNS name and hosting account is provided.

6 Improvements

Although the project meets the basic requirements of the project scope, there are still many areas in which further implementation can be made.

6.1 Backend

6.1.1 Responses

The API only contains methods and requests that may be of use. Some also lack failure responses. Most of the methods only contain 404 Not found responses. Further implementation on it will allow better compensating power to unexpected events.

6.1.2 Authentication Security

The API currently has very low security. Any individual with the correct user name can obtain a JSON of the logs if a GET request is called correctly. A better solution to this would be to use API Keys and tokens. Each individual user has their own unique token for each login. Which is used with the API key to obtain secured information. This ensures that authentication is made before information can be retrieved.

6.1.3 Security against attacks

Another feature that would be useful for security is limiting failed logins for a given time period. An example would be, a user fails to enter correct authenticate information for 3 times within 5 minutes, the IP address is recorded in a database and will be removed after 15 minutes. Users in these IP addresses are unable to attempt login until the 15 minutes has passed. This protects the server from brute force attacks, which may also increase the server costs.

6.2 Mobile Application

6.2.1 Refresh

The refresh update function is not implemented in the most user-friendly way. Currently, the only way to update the home page is to add a new log or reopen the application. A scroll gesture used for refresh can be helpful.

On the add page, pressing the back button currently closes the application. This can be fixed with additional gesture functions.

6.2.2 Additional pages

The number of pages is implemented at a minimum to allow the basic functionalities to be performed. Adding pages such as about us, news or even health statistics page will allow more information to be presented to the user.

6.2.3 Preference settings

There are currently no personalisation settings. These settings can include background image, fonts, font size, additional themes. Adding personal preferences to the application allows users to be more attracted to the application and be more comfortable while operating within it.

6.3 Web Application

6.3.1 More functionality

The web application only has the necessary functionalities. It lacks the power to modify logs or users (it can only perform DELETE requests). The administrator can only perform limited activities using the website application. The application is capable of more functionalities such as resetting user passwords, push notifications to users to remind them of long periods of inactivity, filtering or searching for specific entries.

6.4 Additional Features

6.4.1 Standardised styling

This project focuses on the functionality of the applications. The styling of the applications are implemented at a minimum. Although the applications are functional throughout various platforms, it is not standardised, meaning that the applications may have differing layouts across different platforms. Having standardised layouts allow users to be more comfortable with the applications.

6.4.2 Calorie and BMI calculation

With the help of a large database which includes the type of food and exercise routines, and allocating an estimating calorie calculation, can allow better and more efficient monitoring.

7 Conclusion

This project demonstrates the necessary features that the scope requires. The project aim is to build an environment to observe and monitor patient health by allowing patients to upload logs into a cloud environment. Through this project, the clients can record their own logs through a mobile application that runs in Android or iOS. The recorded logs are uploaded to a cloud server. Administrators are able to log in to a website application to obtain the details of the logs. The mobile application is ready to be deployed to Google play store and Apple app store with a developer account. The website application is ready to be hosted with a web server account and a DNS address.

Although the project meets its minimum requirement, further development can allow additional features for better efficiency, layout, user friendliness and functionalities.

I learned a lot of new technologies and techniques in this project. Serverless and Hybrid development were the 2 most important aspects that was gained throughout this project.

Again, I would like to thank Dr Gilman for providing me with this opportunity.