Mobile Web Development

Model Document

Patrick Kiely – 20054995

[Patrick.Kiely@sunlife.com](mailto:Patrick.Kiely@sunlife.com)

087 2640215

Contents

[Purpose 3](#_Toc388179413)

[Description of Mobile Web Application 4](#_Toc388179414)

[1) Architectural Goals and Constraints 5](#_Toc388179415)

[Technical Platform 5](#_Toc388179416)

[Transaction 5](#_Toc388179417)

[Security 5](#_Toc388179418)

[Persistence 5](#_Toc388179419)

[Reliability/Availability (failover) 5](#_Toc388179420)

[Performance 5](#_Toc388179421)

[Internationalization (i18n) 6](#_Toc388179422)

[2) Use Cases 7](#_Toc388179423)

[a) Use Case Diagrams 8](#_Toc388179424)

[3) Wireframes 10](#_Toc388179425)

[a) Initial load 10](#_Toc388179426)

[b) Tab 1 - Home Screen 10](#_Toc388179427)

[c) Tab 2 - Adding a new course 11](#_Toc388179428)

[d) Tab 2 – Played Courses 11](#_Toc388179429)

[e) Tab 3 - Social 12](#_Toc388179430)

[f) Tab 4 – Calculate Score 12](#_Toc388179431)

[g) Tab 5 – More Info 13](#_Toc388179432)

[3) API Descriptions 14](#_Toc388179433)

[a) Add a Course Played Item 14](#_Toc388179434)

[b) View a Course Played Item 15](#_Toc388179435)

[c) Update a Course Played Item 16](#_Toc388179436)

[d) Delete a Course Played Item 17](#_Toc388179437)

[4) Logical View 18](#_Toc388179438)

[5) Deployment/Implementation View 18](#_Toc388179439)

[Overview 18](#_Toc388179440)

[Layers 18](#_Toc388179441)

[6) Data View 20](#_Toc388179442)

# Purpose

The purpose of this document is to show the design process undertaken to analyse and design a mobile web application for a Mobile Web Development project deliverable. The document contains wireframes, api model documentation, use cases and architectural diagrams.

# Description of Mobile Web Application – Golfbuddy.

I have chosen to build a mobile web app (The Golfbuddy) to show you closest golf clubs to you in your immediate vicinity, the current deployed version shows all major golf clubs in the south east as well as the official Irish top 10 as per the golfing digest website.

On initial load an element floats down to ask you to add to home screen, I have created app icons for both Android and iOS using: <http://www.gieson.com/Library/projects/utilities/icon_slayer/#.U3ioBfldWSo> this was a very useful site for resizing and applying shadow etc for professional looking icons.

You do not see the navBar to navigate to other tabs in the app unless you log in with facebook, once logged in you have access to the tabs.

As well as directing you to a golf club, on tab 2 it also allows you to record the fact you play there, the date, your handicap, your score as well as notes on the round. This information is displayed with minimal info first (golf course and date) but you can expand from the list to show all the attached information as well as edit it.

Swiping right on a golf course name brings up the delete icon, tapping this deletes the entry or swiping left cancels the delete.

On tab 3 you can log into facebook to like the app or share the face you are using the app, if others of your face book friends are using the app they will be listed here as well after you login.

On tab 4 a calculator is supplied to help you quickly tot up your scores on the round of golf if the numbers become too large..

Tab 5 is a more info tab, it contains a more info button which shows different info about the application. As well as this there is a street map button to allow you to quickly access google to see where you are.

# Architectural Goals and Constraints

This section describes the software requirements and objectives that have some significant impact on the architecture

## Technical Platform

The Golfbuddy web application will be deployed onto an Nginx web server running on an Amaxon Web Service EC2 instance incorporating node.js capabilities with a html and javascript rendered front end.

## Transaction

The Golfbuddy application is transactional, leveraging the technical platform capabilities. Transaction management model of the Mongodb platform will be reused extensively.

## Security

The app should be secured so a user can only see their own information.

The application must implement basic security behaviors:

* Authentication: Login using at least a user name and a password using Facebook

## Persistence

Data persistence will be addressed using a non-relational database, MongoDB HQ.

## Reliability/Availability (failover)

The Application is not misson critical but should targer to be available 24/7 unless undergoing a maintainence cycle (of no more than one hour a month).

## Performance

The App must start up in less than 7 seconds.

Response time on changing view must be no ore than 1 second once app is up and running.

## Internationalization (i18n)

Currently we do not cater for internationalisation but it may be included in a future release

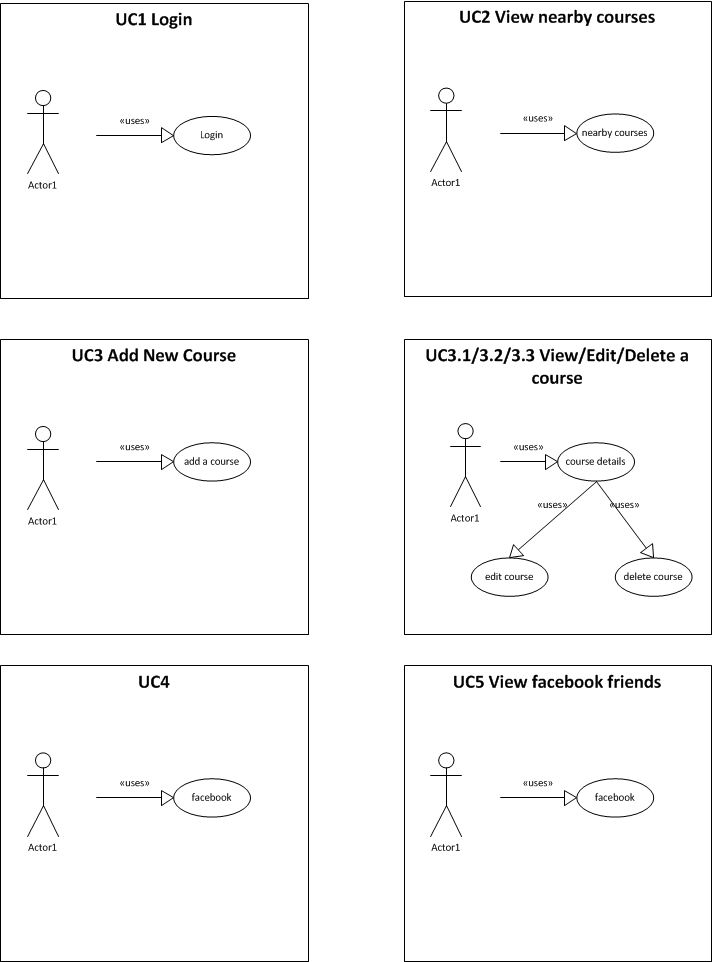
# Use Cases

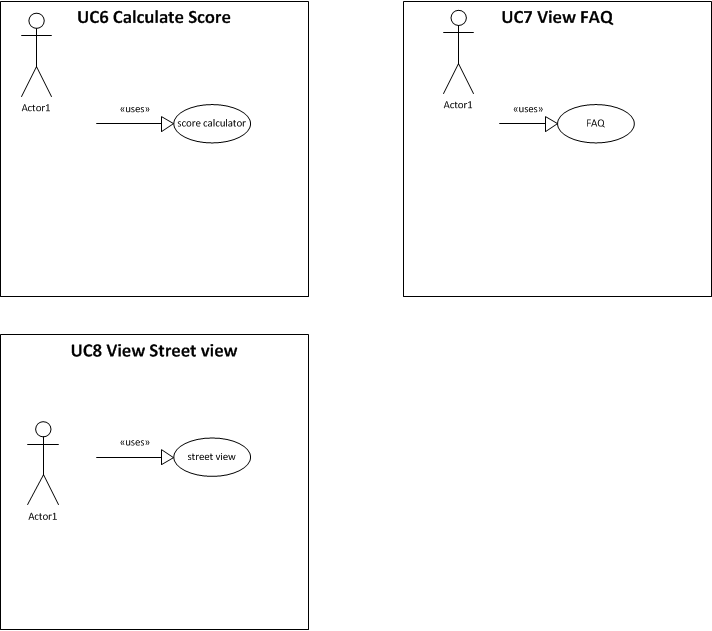
The primary actor on all use cases is the user, there are no supporting actor roles.

The stakeholder roles and Interest role is also filled by the user.

|  |  |  |
| --- | --- | --- |
| UC1 | Login | User logs into the web app |
| UC2 | View nearby courses | User views nearby courses on Google Map |
| UC3 | Add a new course | Users adds a new played course |
| UC3.1 | View a course | User views a saved course |
| UC3.2 | Edit a course | User edits a saved course |
| UC3.3 | Delete a course | User deletes a saved course |
| UC4 | Like App on Facebook | Users likes App on facebook from social screen |
| UC5 | View friends from Facebook | Users views friend on facebook who liked the app |
| UC6 | Calculate Score | User uses calculator to total score |
| UC7 | View FAQ | User views FAQ |
| UC8 | View Street View | User views street view (backbone model view) |

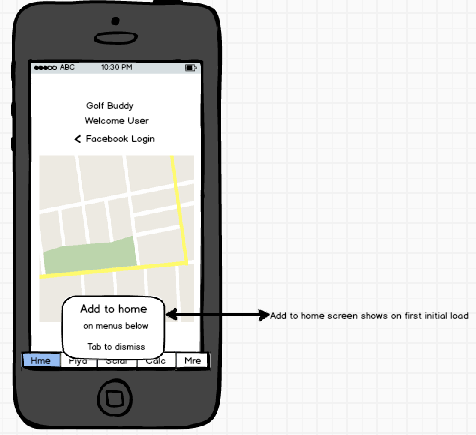
### Use Case Diagrams





# Wireframes

### Initial load



### Tab 1 - Home Screen



### Tab 2 - Adding a new course



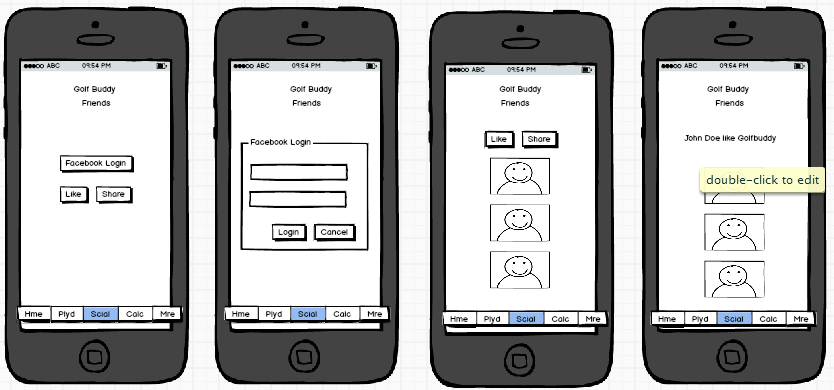
|  |  |  |  |
| --- | --- | --- | --- |
| On Initial entry there are no played coursed on Played tab | Once you tap add the add button disappears and the entry panel appears | Enter the required data and tap save. Tapping cancel will close the tab and nothing will be saved | Once the item is saved it will appear on the menu and can be selected. |

### Tab 2 – Played Courses



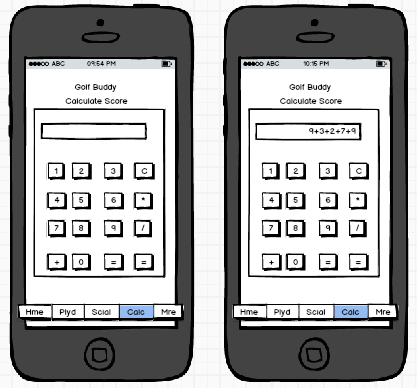
|  |  |  |  |
| --- | --- | --- | --- |
| Displaying a list of saved courses played | Clicking the info button causes a dropdown to appear with more info, this infor should be editable and clicck save once done. | Swiping right will cause the delete icon to appear, tapping that deletes the entry or swiping left cancels the action | After deletion the item on the list below moves up and that entry is permanently deleted |

### Tab 3 - Social



|  |  |  |  |
| --- | --- | --- | --- |
| Clicking op Facebook login allows you to interact with Facebook | Enter facebook login credentials | Your friends who use Golfbuddy will be listed here and you can like the app. | You name will appear as likeing the app. |

### Tab 4 – Calculate Score



|  |  |
| --- | --- |
| Tab 4 is a simple calculator app. | You can enter scores at the end of your round and total up you end score. |

### Tab 5 – More Info



|  |  |
| --- | --- |
| A simple FAQ page | A street view map generated using backbone model and view |

# API Descriptions

### Add a Course Played Item

#### Description

Adds a new course played item to the list of played courses

#### Request Parameters

**text**:input. (Course Played)

* Required : Yes

**handicap**:Numeric

* Required : No

**score**: Numeric

* Required : No

**notes**: Text,

* Required : No

**DOP**: Date (Date of Play)

* Required : No

#### Create API Implementation

create: function( req, res ) {

var input = req.body

if( !util.validate(input) ) {

return res.send$(400, 'invalid')

}

var golf = {

text: input.text,

handicap: input.handicap,

score: input.score,

notes: input.notes,

DOP: input.DOP,

}

golfcoll.insert(golf, res.err$(function( docs ){

var output = util.fixid( docs[0] )

res.sendjson$( output )

}))

},

### View a Course Played Item

#### Description

View a course played item from the list of played courses

#### Request Parameters

**Rm\_id**:text

#### Return Parameters

**text**:text. (Course Played)

* Required : Yes

**handicap**:Numeric

* Required : No

**score**: Numeric

* Required : No

**notes**: Text,

* Required : No

**DOP**: Date (Date of Play)

* Required : No

#### View API Implementation

read: function( req, res ) {

var input = req.params

console.log(req.params)

var query = util.fixid( {id:input.id} )

golfcoll.findOne( query, res.err$( function( doc ) {

if( doc ) {

var output = util.fixid( doc )

res.sendjson$( output )

}

else {

res.send$(404,'not found')

}

}))

},

list: function( req, res ) {

var input = req.query

var output = []

var query = {}

var options = {sort:[['created','desc']]}

golfcoll.find( query, options, res.err$( function( cursor ) {

cursor.toArray( res.err$( function( docs ) {

output = docs

output.forEach(function(item){

util.fixid(item)

})

res.sendjson$( output )

}))

}))

},

### Update a Course Played Item

#### Description

View a course played item from the list of played courses

#### Request Parameters

**Rm\_id**:text

* Required : Yes

**text**:text. (Course Played)

* Required : No

**handicap**:Numeric

* Required : No

**score**: Numeric

* Required : No

**notes**: Text,

* Required : No

**DOP**: Date (Date of Play)

* Required : No

#### Update API Implementation

update: function( req, res ) {

var id = req.params.id

var input = req.body

if( !util.validate(input) ) {

return res.send$(400, 'invalid')

}

console.log(input)

var query = util.fixid( {id:id} )

golfcoll.update( query, {

$set:{

text:input.text,

handicap:input.handicap,

score: input.score,

notes: input.notes,

DOP: input.DOP

}}, res.err$( function( count ) {

if( 0 < count ) {

var output = util.fixid( doc )

res.sendjson$( output )

}

else {

console.log('404')

res.send$(404,'not found')

}

}))

},

### Delete a Course Played Item

#### Description

Delete a course played item from the list of played courses

#### Request Parameters

**Rm\_id**:text

* Required : Yes

del: function( req, res ) {

var input = req.params

var query = util.fixid( {id:input.id} )

golfcoll.remove( query, res.err$( function() {

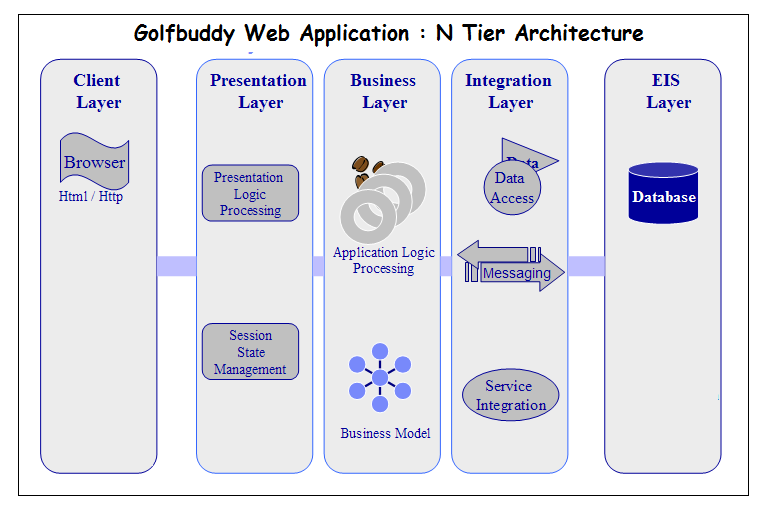
var output = {}

res.sendjson$( output )

}))

}

# Logical View



# Deployment/Implementation View

## Overview

The Implementation view depicts the physical composition of the implementation in terms of Implementation Subsystems, and Implementation Elements (directories and files, including source code, data, and executable files).

Usually, the layers of the Implementation view do fit the layering defined in the Logical view

It is unnecessary to document the Implementation view in great details in this document. For further information, refer to the Online Catering Service 1.0 workspace in Rational Software Architect.

## Layers

#### Presentation Layer

The Presentation layer contains all the components needed to allow interactions with an end-user. It encompasses the user interface

#### Control Layer

The Control layer contains all the components used to access the domain layer or directly the resource layer when this is appropriate.

#### Resource Layer

The Resource layer contains the components needed to enable communication between the business tier and the enterprise information systems (Database, external services, ERP, etc…)

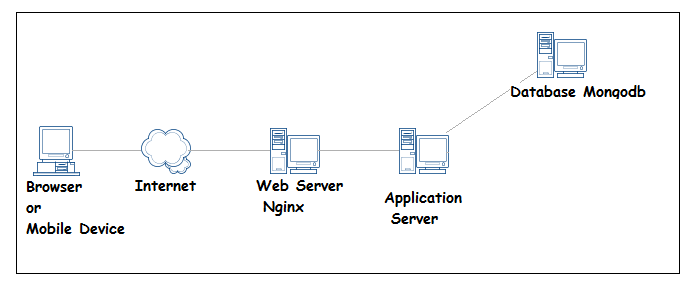
#### Domain layer

The Domain layer contains all the components related to the business logic. It gathers all the subsystems that meet the needs of a particular business domain. It also contains the business object model.

#### Common Elements Layer

The Common Element layer contains the components re-used within several layers.

#### Deployment/Implementation Diagram



# Data View

