

Networked Dreams

Technical Report

Table of Contents

1.Summary.....	5
2.Introduction.....	5
2.1.Background.....	5
2.2.Research.....	7
2.3.Aims.....	8
2.3.1.Categorising the Land of Nod.....	10
2.3.2.Ambitions Vs What Was Realised.....	10
2.4.Technologies.....	11
2.4.1.Libraries.....	12
2.4.2.Development Environments.....	12
2.5.Structure.....	13
3.System.....	13
3.1.Requirements.....	13
3.1.1.Evolution of Requirements from Initial Brief.....	13
3.1.2.Summary of Requirements Modifications.....	15
3.1.3.Functional Requirements.....	16
3.1.4. Data Requirements.....	24
3.1.5.Performance / Response time requirement.....	25
3.1.6.User Requirements.....	25
3.1.7. Environmental Requirements.....	25
3.1.8.Usability Requirements.....	25
3.2.Design and Architecture.....	25
3.2.1.Entity Relationship Diagrams & Data Driven Development.....	25
3.2.2.User Journey Design.....	29
3.3.Implementation.....	31
3.3.1.DreamDB class.....	32
3.3.2.User Journey.....	34
3.3.3.Sessions.....	34
3.3.4.Notes on Control Flow.....	38
3.3.5.Data Analytics & Visualisation.....	39
3.4.Testing.....	42
3.5.Graphical User Interface GUI Layout.....	50
3.6.Evaluation.....	57
4.Conclusions.....	57
4.1.Web Implementation vs App Realisation.....	57

4.2.Implementation Reflections.....	58
5.Further Development.....	58
5.1.Pragmatic Next Steps.....	58
5.1.1.Back Ups.....	58
5.1.2.Robust Security.....	58
5.1.3.User Journey.....	59
5.1.4.Expanded Membership Profile Data.....	59
5.2.Iterative Stages of Improvement.....	59
5.2.1.Phase 1 – Providing Network Diagram Visualisation.....	59
5.2.2.Phase 2 – Improve Logging Experience and Incentive.....	60
5.2.3.Phase 3 – Trial Dream Communities.....	60
5.2.4.Phase 4 – Integrated Alarm Clock.....	60
5.3.Plug-In & App extensibility of User Contributed Data.....	61
5.3.1.Facebook and other social media API integration.....	61
5.3.2.Enhanced what were you doing before sleeping functionality.....	62
5.4.Other Technologies.....	62
5.4.1.Databases.....	62
5.5.Further Research.....	62
5.5.1.Analytics, Demographics and Taxonomising the Human Psyche.....	62
6.References.....	64
7.Appendix.....	66
7.1.User Manual & Technical Manual.....	66
7.1.1.User Manual.....	66
7.2.Technical Manual.....	68
7.3.Other Materials Used.....	71
7.4.Original Proposal.....	75
7.5.Original Proposal Wireframes.....	80
7.6.Deprecated Requirement Specification.....	87
7.7.Project Management Plan.....	100

List Of Figures

Fig 1 - Quantified Self Examples	6
Fig 2 - Network Diagram Visualisation - Example	9
Fig 3 - Initial Use Case Diagram	14
Fig 4 - Revised Use Case Diagram	17
Fig 5 - part 1 of ER diagram	26
Fig 6 - part 2 of ER diagram	27
Fig 7 - Part 3 of ER diagram	28
Fig 8 - 'User Journey' Sequence Diagram	29
Fig 9 'User Journey' Flow Diagram	30
Fig 10 - Deployment Diagram	30
Fig 11 - Singleton Design Pattern	31
Fig 12 - snippet formatting function, source Includes/db.php	32
Fig 13 - snippet, comma separated values to associative array, source logdream.php	33
Fig 14 - snippet DreamDB function to handle csv associative array, source db.php	33
Fig 15 snippet, preserving user input on form refresh, source join.php	34
Fig 16 - 'User Journey' PHP Sequence Diagram	35
Fig 17 - Collaboration Diagram	36
Fig 18 - \$_SESSION flow diagram.	37
Fig 19 snippet - ensure one dream per 24 hours Source: logdream.php	38
Fig 20 snippet - dynamically populate forms using jQuery & PHP source - complete.php	39
Fig 21 - SQL, create view demographics	40
Fig 22 - snippet -dynamically visualise data part 1 Source dataviz.php	41
Fig 23 - snippet -dynamically visualise data part 1 Source dataviz.php	41
Fig 24 - snippet -dynamically visualise data part 1 Source dataviz.php	42
Test Records	44- 49
GUI Illustrations	51-56
Fig 25 - Social Media Demographics	72
Fig 26 - Android App wireframes part 1	80
Fig 27 - Android App wireframes part 1	81
Fig 28 - Platform Agnostic Wireframes: sign up process	82
Fig 29 - Platform Agnostic Wireframes: sign up process complete	83
Fig 30 - Platform Agnostic Wireframes: data entry	84
Fig 31 - Platform Agnostic Wireframes: data visualisation	85
Fig 32 - Platform Agnostic Wireframes - meta categorise user tags	86

1. Summary

Networked Dreams is a prototype venture into my efforts to create a dream logging app. The app aspires to maximise the potential of information visualisation to both compel users to keep using the service and to offer users unique insight into that great unknown realm - the subconscious. This report describes the implementation of a Create-Read-Update-Delete application in the vein of the above concept and appraises the prototypes suitability to the apps ambitions.

2. Introduction

2.1. Background

I wished to develop a dream logging app, for users to easily log their dreams, and to provide data visualisation as a means for users to understand the connections existing within and between their dreams.

I wished to use those visualisations and personal logging as a means into creating dream communities linked by trends discerned from aggregate data. I am targeting self-trackers (or Quantified Self) and lucid dreamers in particular and a causal audience curious about their dreams.

Dreams are an aspect of peoples lives that people are always curious to learn more about. Dream logging is an activity long undertaken by humans. For instance, there exists a 1955 microfiche digital archive of dreams: it holds the collected records of people's inner lives: their dreams, their life histories [[Lemov, R. The Database of Dreams](#)]. Dream Logging has several implementations in both web and mobile app form. As such I devoted a considerable amount of my time to researching existing applications in the field to best determine how to distinguish my offering.

One of the more interesting developments in this area has been the overlap between Quantified Self Communities and Lucid Dreaming. Quantified Self designates people

across a wide range of demographics who use devices, spreadsheets or even humble pen n' paper journals to note data from their life, and analyse the trends that emerge from their tracking.

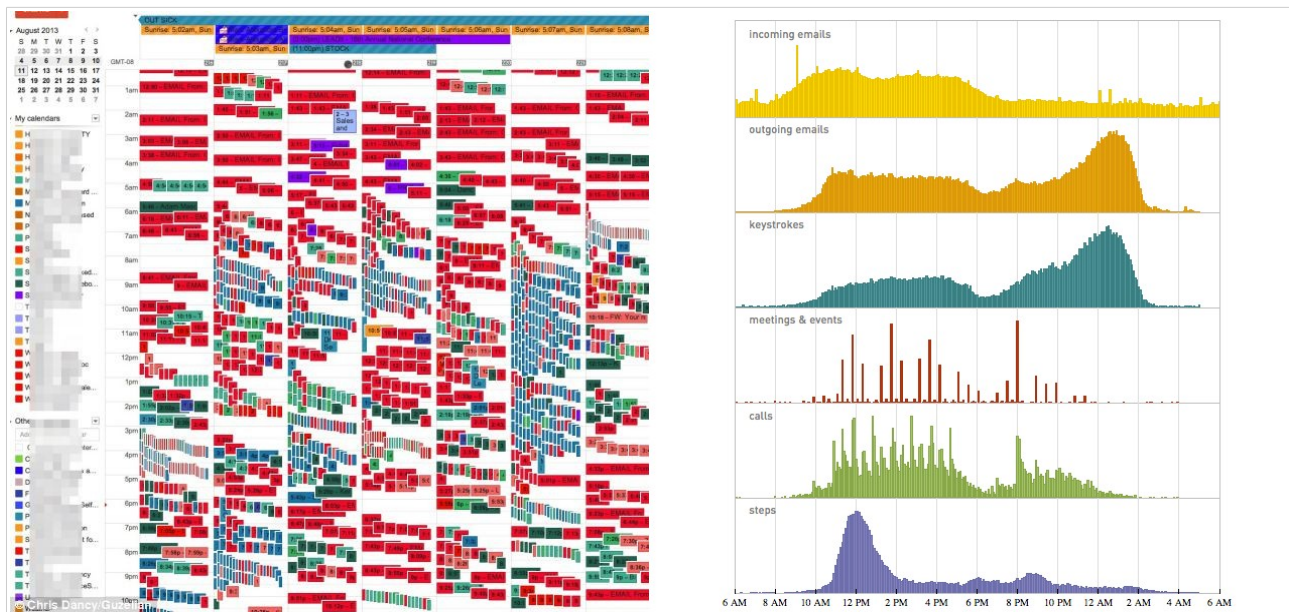


Fig 1. Left - Chris Dancy's quantifies self journaling. Right Stephen Wolfram's Decade of Personal Analytics

Selection of Quantified Self Communities

<http://your.flowingdata.com/>

<http://daytum.com/>

<https://www.chartmyself.com/>

<https://fluxstream.org/welcome>

A sample of Quantified Self Communities and Services

Lucid dreaming is a form of dreaming where the dreamer is aware that they are dreaming, and can shape their dreams at will. It is a skill that can be cultivated (Wiseman, R. Night School 2014), and one way to do so is to log your dreams. The more you log your dreams, the more you remember the content of your dreams and the better able you are to lucid dream.

The intersection of these two trends was the spur for my project. I have a long standing

interest in quantified self, and the pedagogical power of infographics. Quantified Self communities believe in self knowledge through numbers. Lucid dreaming is likewise associated with unlocking inner potential and improving self-awareness (Barrett, A Study of Dream Incubation 1993). I wished to develop an easy to use (in terms of data entry) application which would render the users dreams into some form of quantitative analysis. I likewise wanted to open dreams up to data visualisation, an area which is widely regarded as having the power to illuminate complexity in a concise immediate manner. It is my opinion that responsible use of infographics can be a powerful aid to self-awareness.

I have covered digital technology trends for the past three years in my previous employment, and engaged with what data & data visualisation mean to people through my digital art practice. Creating an app that married the subjective and poetic realm of dreaming with an ascendant means of understanding everything you do via databases and data analytics struck me as an engaging topic.

2.2. Research

To focus the breadth of data possibly contained within any given users dream a categorisation feature was added, which is a stipulation of any dream logged. These 'meta categories' provide an anchor for data analytics. To establish what types of categories users wish to label their dreams as, I researched as many of the existing web offerings for dream journaling as possible (see section [2.3.1 Categorizing the Land of Nod](#)).

A crucial part of making this app relevant to data analytics, as existent within the domain of personal analytics, is providing non-dreaming data (i.e. 'waking data') to correlate with the contents of ones dreams. The app takes account of what a given user was doing the night before they had a dream. This is currently done as a subsequent step of any dream being logged.

Inspiring the 'waking data' gathering was a focus on 'Time Use Research'. Time Use Research originated in contemporary workforce management, its a means for a worker to diagnose their work life balance. This field of research understands time as falling into the following categories

1. Contracted time
2. Committed time
3. Necessary time
4. Free time

[Dagfinn Aas *Time Use Studies: Dimensions and Applications*, 1986]

I utilised these categories as meta-ordering principles for the options available to a user to designate what they were doing prior to sleeping and dreaming. This is a feature which can be elaborated upon as the project develops. It also aligns the app with the prerogatives of a significant section of quantified self users (concerned with optimising productivity) and also with the aspirations of lucid dreaming (to unlock ones hidden potential).

'Time Use Research' enables my project to channel the research hypothesis 'threat rehearsal dream hypothesis'.

The ambition is to create this project as an app - but in its current incarnation it has been developed principally as a website - that can be accessed by any browser, mobile or desktop. The aim is to continue building toward the mobile experience. This app needs to compete with other dream logging apps - which all have the advantage of immediate access because they can interacted with on a mobile device.

2.3. Aims

This app allows dream logging in a fashion that permits later elaboration by data visualisation.

To distinguish my application and service from existing websites and apps I focused on data visualisation and capturing data from the user related to what they did prior to sleeping and dreaming a given dream. The latter category was supported by research into the taxonomies of Time Use Research. Capturing what the user did prior to dreaming was intended to aid the users self discovery and also add a powerful additional category to the

data analytics side of the project

Utilising visualisation sets my project aside from browser based efforts like

www.dreamsccloud.com/ and www.dreamdoze.com/.. In both of these website services

'expert' interpretation is offered as a service (both www.dreamsccloud.com/ and

<http://www.dreammoods.com> follow this model), or a 'wisdom of the crowd' approach to

interpretation is offered (<http://dreamdoze.com/>). My project would instead offer the power

of infographics and the ability to form communities with people who dream about similar

things as the means of interpretation. In allowing users to create micro communities of

shared dream subjects I am aiming to channel some of the 'splintered social media' that

has been evident in the preference for closed media, like Whatsapp, versus broadcast

social media - such as Facebook & Twitter.

My desire to realise this project was driven by a curiosity to see whether the sociogram

visualisation (predominant in network culture, and social media especially, which makes up

most of our digital lives) would provide an engaging window into our sleeping hours.

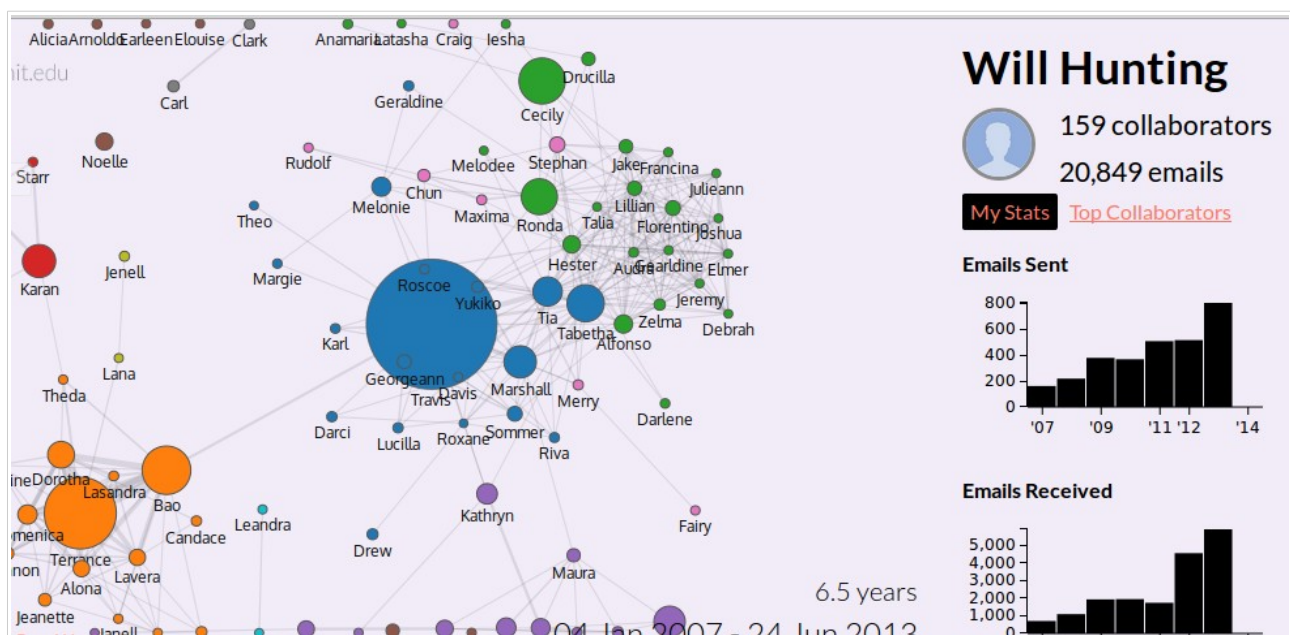


Fig 2 Immersion - a network diagram interface to email

I wished to explore the affordances of Network Visualisation infographics. This is because Network Diagrams are very apt to our current state of digital technology interaction - where

vast amounts can be inferred about who you are based on the connections between the metadata you produce in interaction with digital systems. It struck me that foregrounding the data visualisation as an interface to comprehension, and using networked diagrams as the principal visualisation metaphor was a means to distinguish my product from extant Dream Logging offerings.

In the middle of my development it came to my attention that comparable research had been recently completed by DARPA and Indiana University's Center for Complex Networks and Systems Research [Varol, O Menczer, F. *Connecting Dream Networks Across Cultures* 2014]. To me this is indicative that I have hit upon an area of attention to a great many actors - academic, societal and personal.

2.3.1. Categorising the Land of Nod

Dreams are easily the most theorised area of human existence, and as such I kept a relatively arms length approach to the detail of some theories. Instead, when designing 'keyword categories', I considered the taxonomies that already existed at a range of websites offering similar dream logging services.

Examples of Sites Researched
http://www.dreammoods.com/dreambank/
http://dreamdoze.com/
http://www.sleeps.com/dictionary/dictionary.html
http://www.dream-analysis.com/interpretations.htm

Through a process of data trawling (please see the section on data scraping in [Appendix Other Materials Used](#) for more) I determined 25 categories for use. I am less concerned with these categories faithfulness to theories of dreaming (which are many and diverse, and present in many disciplines). What is more interesting for me is providing some means of equivalence between the activities of the dreamer before they went to sleep, and what their dream was about. This establishes a homology between time-use research' and 'threat rehearsal dream hypothesis'.

2.3.2. Ambitions Vs What Was Realized

With that noted, it was not possible to implement the visualisation as desired for this iteration. However, as noted subsequently in [Section 2.1 Data Analytics & Visualisation](#) endeavours have been made in terms of data representation that these visualisations could be readily implemented.

The initial aim (as illustrated in [Appendix: Original Proposal](#), and [Appendix: Original Wireframes](#)) was to provide this service as a mobile phone application integrated with an alarm clock. The reason for this is that it is well researched that the moments immediately after awakening are the most important for logging your dream. Rather than rely on long form text the application was to expedite the entry process by means of four fields (emotions, locations, people and objects) which could be entered visually. However as development advanced it was clear that an android application would need to wait until a further iteration.

The revised aim consisted of a website, accessible as mobile application. While the alarm clock is absent, this service should still be accessible on mobile devices, thus expediting the data entry process upon awakening. The distinguishing features of this app would be data visualisation, waking data capture, and microcommunities made possible by aggregate analytics. All three features would be extended to any future mobile application, with all the affordances for augmentation said platform would afford (see [Section 5 - Further Developments](#) for more on this).

2.4. Technologies

This project was realised utilising web technologies. The nature of web technology development entailed a variety of technologies specifically PHP, HTML and HTML5, CSS, JQuery, and Javascript. A MySQL database was utilised for the projects data requirements.

A web technologies project can run on any device equipped with internet access and a browser.

MySQL was familiar to the author. Together with Apache and PHP it represents an open source solution to hosting a web service which is incredibly ubiquitous among web hosting providers. This ubiquity was a positive attraction to using these technologies.

HTML, HTML5 and CSS were used for presentation purposes, to style the visual elements of the project. Ensuring as much cross device compatibility is possible through responsive design.

PHP ran server side, performing data handling duties. All connections to the MySQL database were made through PHP and MySQL. Any data retrieved from the database was accessed and encoded for presentation by PHP. Use of PHP was a decision reached in consultation with my lecturer. It also makes sense for planned (but presently postponed) subsequent integration with Facebook API, which has excellent support for PHP given that it originated in that language.

Combinations of client side jQuery and Javascript with server side PHP handled the data validation portions of the website. JQuery also enables dynamic interaction with webpages - transforming the static webpage paradigm of PHP into a dynamic one. PHP executes server side where jQuery and javascript execute client side.

2.4.1.Libraries

Several Javascript Libraries were used for the data visualisation portion of the project. Sigma was experimented with, but I decided to postpone representing the MySQL database with graph abstract data types until a subsequent version. I instead utilised the D3 javascript library. D3 stands for 'data driven documents' - it binds arbitrary (i.e., developer specified) data to the Document Object Model. Once bound data-driven transformations can be executed on said data. It outputs visualisations in SVG format, by performing HTML and CSS transformations through javascript scripting.

2.4.2.Development Environments.

The PHP and Javascript portions of the the project were developed within the Netbeans IDE, which has good support for PHP, and HTML5 technologies. It's support for javascript is less useful. With hindsight a from first principles framework focused approach integrating these technologies would be better.

The WebFlow visual editor was used to construct the HTML and CSS for this project. It is a browser based WYSIWYG editor, akin to Dreamweaver or Mozilla Komposer.

2.5. Structure

All imagery contained in this report likewise accompanies the submission, on the media containing the code.

Please note that occasionally this document transforms from a default portrait orientation to landscape, for the sake of clarity of the information presented

3. System

3.1. Requirements

3.1.1.Evolution of Requirements from Initial Brief

The realised project evolved from the initial brief according to two particular domains.

There were:

- The initial plan for a phone app implementation,
- The primacy of data visualisation as the main customer take-away

My initial requirements envisioned provisioning this service in the form of an android app (the entirety of this proposal and supporting artifacts and rationale can be viewed in [Appendix: Section 7.4](#), and [Appendix Section 7.5](#). The reasoning was that dream logging is best conducted immediately upon waking, and therefore an application that

incorporated/integrated alarm clock functionality was an excellent way to meet this criteria. To further consolidate the ease of use graphical data entry was envisioned where users could quickly 'tag' people, places and object from precalibrated thumbnail databases. Below was the original use case diagram (compare with Fig 4 - Revised Use Case Diagram, p17).

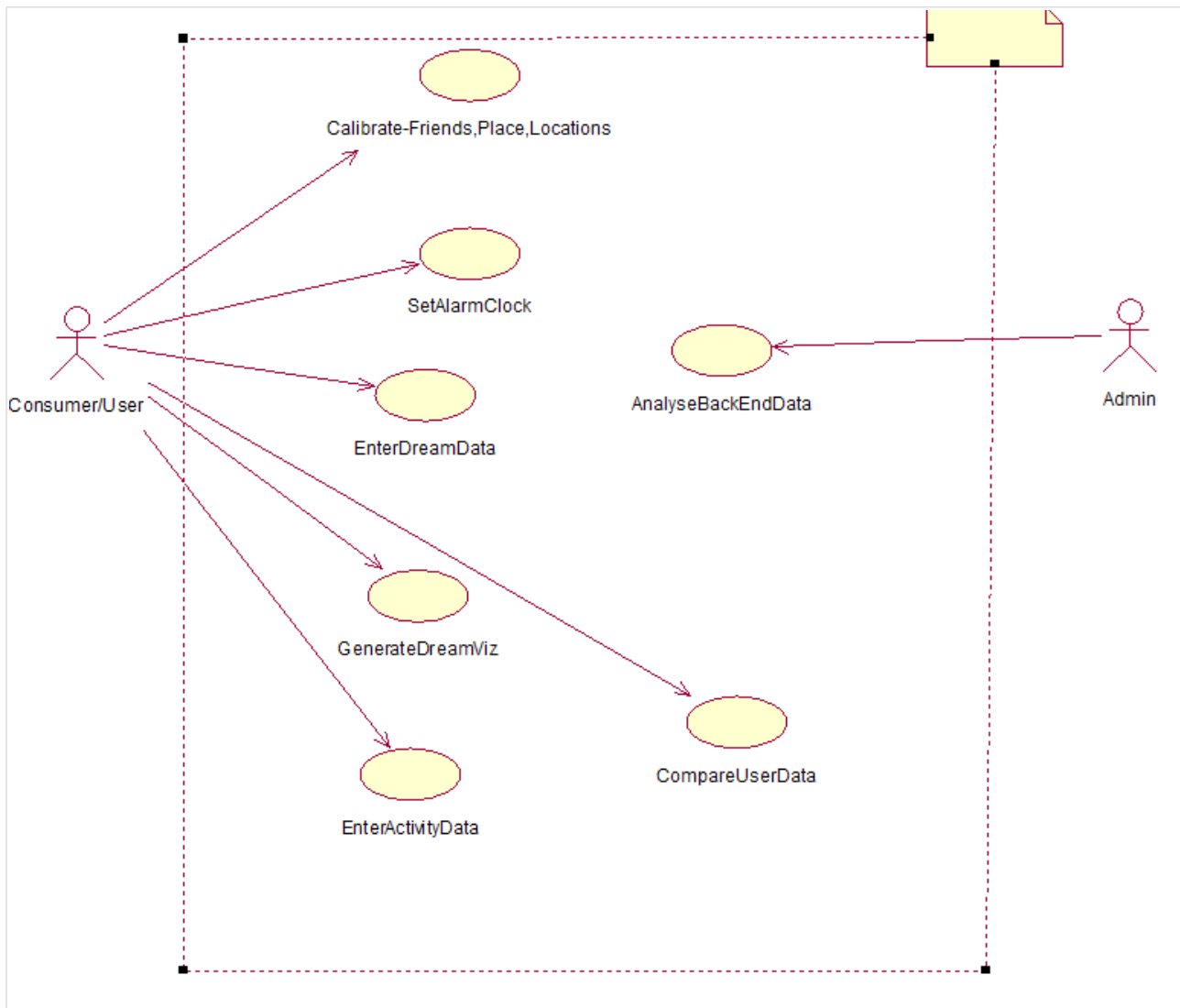


Fig 3: initial use case diagram use case diagram. Additionally, wireframes of this initial mobile app project can be viewed in [Appendix: Section 7.5 Wireframes](#)

Providing the user with network diagram visualisations of the people, places, objects and emotional content present in their dreams was also a strong guiding imperative of the project. This was due to the authors desire to create a platform that permitted the understanding of dreams through the power of dynamic infographics.

Both these requirements are not part of the delivered prototype. The app instantiation of the project idea was removed circa week 4. The personalised visualisation during week 12. (see [Project Management Gantt Chart: Appendix 7.7](#))

The rationale behind dispensing with a mobile app delivery was that I concluded that the data analytics the most interesting and crucial component of this project, and that effort was best expended realising this as properly as possible. Moreover a website could be ported to a phone app with less effort than an android app could be ported to a website.

An agreement was reached in consultation with my supervisor where the app/service would be provisioned via a website. An option to package the application using PhoneGap was entertained as a potential means of delivering the service as an app. However with phonegap the ability to develop an integrated alarm clock was negated.

The visualisations removal was borne of a realisation that network visualisations were perhaps too complex a requirement to undertaking within the context of project delivery timelines. Visualisations still play a part in the project, just not on the personal level I had originally intended them to be. Requirements 4 & 6, from the original requirement spec ([Appendix Section 7.6](#)), were intended to be consolidated into a single requirement, with three distinct use-cases / entry points.

However in the end only 1 entry point was realised, as it became apparent that an overhaul of how data was accessed from the database and presented in JSON was requisite to a seamless functioning of the data visualisation.

3.1.2. Summary of Requirements Modifications

Requirement	Functionality	Reason for Revision
8	Android app	App no longer provisioned as android mobile app - alarm clock functionality not applicable to website
9	Android app	App no longer provisioned as android mobile app - graphical tagging not a priority is user has keyboard available to them. Moreover API integration was deprioritised as a requirement for the project

5	Database design	Permitting the user with free form dream categorisation would result in problematic database design. It also risked diluting the quality of analysis that could be derived from the data. This requirement was revised that all dreams are mandatory categorised upon data entry, but that users can provide their own suggestion if none of the provided categorisation fit their needs
4 & 6	User Experiences	Complexity appraisal - deemed beyond scope of this iteration of project

While it has not been possible to implement the visualisation portion of this project, all necessary work has been done to ensure that data visualisations can be implemented easily - i.e. providing a means for converting the underlying data into JSON for use with a plug in visualisation library.

3.1.3.Functional Requirements

Updated Use Case Diagram

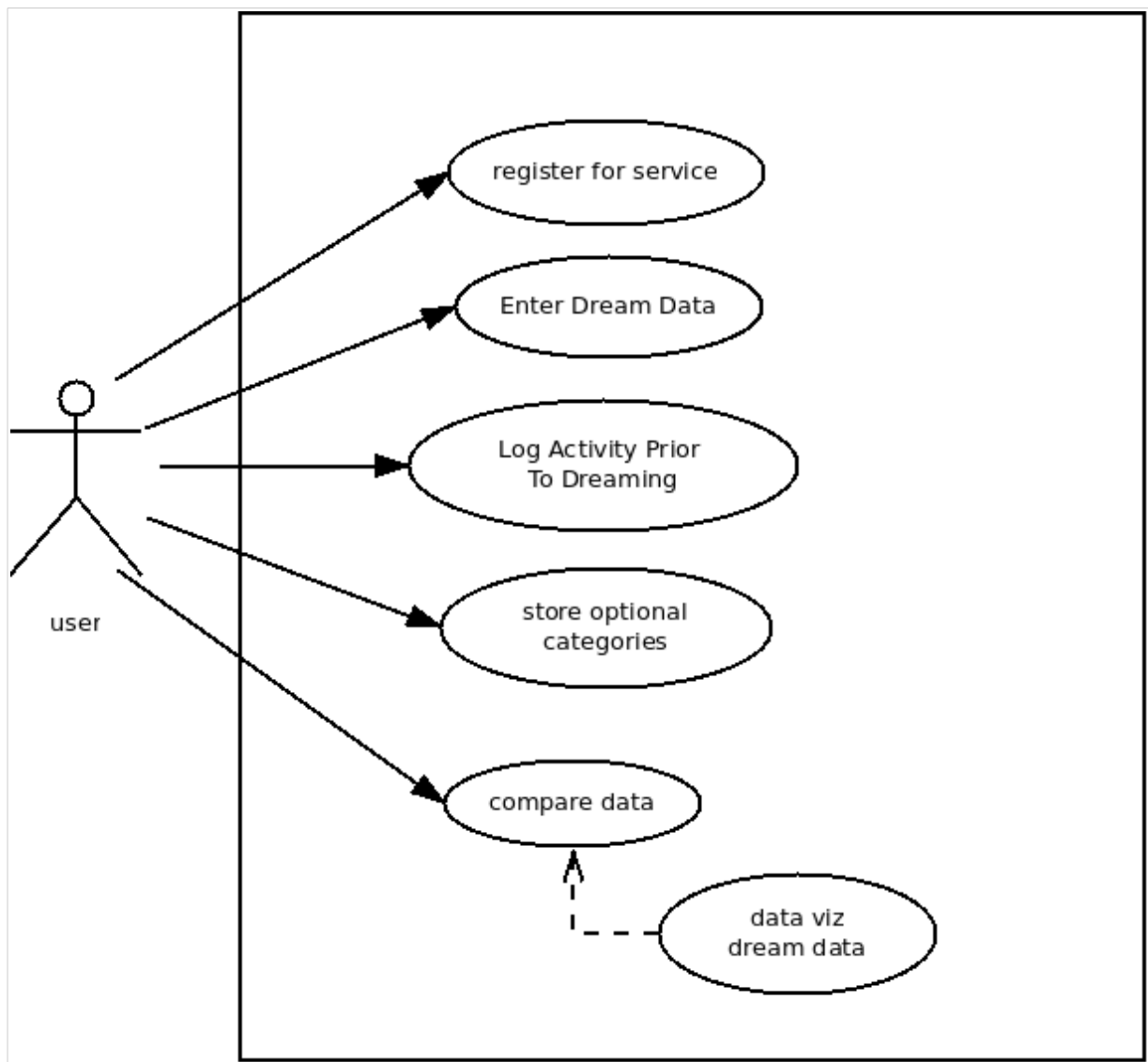


Fig 4: Revised Use Case Diagram

3.1.0 Requirement 1 <User Registration>

3.1.3.1.1.Description

This use case shall register a user for the service and application. It is integral to providing the app with data to populate subsequent use cases.

3.1.3.1.2.Use Case

Scope

The scope of this use case is to register a new user

Description

This use case describes the process for setting up user profile data for the first time

Flow Description**Precondition**

If the website is being accessed for the first time

Activation

When the user clicks the 'register' or 'sign up' button

This use case can also be accessed via log in (see alternate flow)

Main flow

1. The user is presented with a slide-by-slide set up wizard
2. Name, age, gender, and profession are entered, next button is pressed to advance to next set up section

Sub Flows**Alternate flow**

The alternate flow is a user logging in. It follows the same access flow, minus the set up wizard. No changes are made to user database during “log in” alternate flow

Termination**Post condition**

Database containing the users data is readied

3.1.1 Requirement 2 <Enter Dream Data>**3.1.3.1.3.Description**

This use case shall enable a user to log their dream data. This use case is integral to gathering the data which the user will be encouraged to visualise and analyse.

3.1.3.1.4.Use Case**Scope**

The scope of this use case is to gather dream contents data from a user of the service

Description

This use case describes how a user enters dream data via text entry in accordance with the four parameters of dream data.

Flow Description

Precondition

A registered user must be logged in.

Activation

When the user accesses the website / When the user logs in to the website

Main flow

1. User enters text into "Friends | Object | Location | Action" text fields
2. Any text separated by a comma (or similar delimiter) is stored as an item for the field it is associated with
3. User clicks complete
4. Data is logged to a database

Sub Flows

2.1 If the user leaves any field empty a prompt window will seek assurance from user that they deliberately intended to leave field blank.

Termination

The flow terminates once all data entered

Post condition

User's database is updated with latest data

3.1.2 Requirement 3 <"Waking Data" Survey>

3.1.3.1.5.Description

This use case shall harvest information relating to the users activities before going to sleep. This is done through a categorisation format, that unfolds from an initial yes | no question. The yes | no question determines if the user had free/leisure time

before going to sleep. This enables a more cohesive drop down menu to be presented to the user.

3.1.3.1.6. Use Case

Scope

The scope of this use case is to gather supplementary data related to a users waking activity. The user annotates their waking data activities through a bespoke data entry interface designed for quick enjoyable navigation.

Description

This use case is important to including robust data gathering into the system from the outset. In order that the app provides useful information against which users dream data can be compared against this use case seeks to determine what the users waking activity consisted of before they went to sleep and dreamed

Flow Description

Precondition

A registered user must be logged in. User must have logged a dream.

Activation

User clicks log my dream data. The waking data form is presented to the user

Main flow

1. User clicks link and/or logs in.
2. User completes the question and selects from subsequent drop down menu
3. Upon completion the data is saved to their profile

Sub Flows

2.1: On each tab with questions, user may annotate the data with tags ("Friends | Object | Location | Action" format) or a diary entry

2.2: this data is added to sum data to be saved to user profile

Alternate flow

Exceptional flow

E1.1: The user attempts to access the screen past the allotted time.

Termination

When the user clicks save

When 12-24 hours have elapsed since the notification was sent

Post condition

Waking Activity table updated

3.1.3 Requirement 4 <Optional Category Storage>

3.1.3.1.7.Description

3.1.3.3 All dreams must be classified with a meta-category. All waking data entries must have a cause and a time-use category. For occasions when the categories do not conform to what the user wishes to log there must be a way of facilitating their desire while capturing all relevant metadata.

3.1.3.1.8.Use Case

Scope

This use case must capture relevant categorical data when the databases existing meta-categories are insufficient to a users needs. Users data is categorised, and a table populated with data that will later inform additions/revisions to the databases metacategories

Description

On occasion when dream 'meta-category' or activity prior-to-sleeping 'meta-categories' are ill fitting to the user, the user may enter a freehand category to classify the dream.

Flow Condition

Precondition

A registered user must be logging either a dream, or logging waking data (i.e. on the respective pages for both the activities detailed in Requirement 2 & 3).

Activation

The user must select 'give me another option' from the meta-category drop down menu.

Main Flow

1. user selects 'give me another option'
2. The meta category drop-down menu is replaced with a free-text form field
3. User submits data, along with all other forms requisite (as detailed in Requirement 2 & 3)

Exceptional Flow

The user enters no suggestion. This results in the dream being lost to metacategory data analysis

Termination

When the user clicks save.

3.1.4 Requirement 5 <Compare Dream Data>

3.1.3.1.9.Description

Sum total of users dream data can be compared for analysing trends. This is an admin specific function, which can be reimplemented and extended to users.

3.1.3.1.10. Use Case

Scope

The scope of this use case is to provide a means of analysing the aggregate data available on the platform with reference to a given users data.

Description

Logging into an additional section of the website enables the user to compare their dream content against demographic parameters. Users can compare their own dream trends to the dream trends of the population of platform users

Flow Description

Precondition

Each user of the platform must have completed their member registration (demographic information) part of their profile.

Activation

This feature can always be accessed by the admin users of the system
Regular users of the system have this feature unlocked after certain criteria are met (N amount of data inputted, completion of metacategories, completed end user license agreement)

Main flow

1. The user selects a meta category of dreams from a menu
2. The app generates a visualisation illustrating what a subset of users do before they dream of certain topics
3. This visualisation remains displayed until the user makes another selection from the first drop down menu

Sub Flows

1.1 User adds additional parameters (demographics) via a GUI, the query sent to the database is amended accordingly

Exceptional flow

User has no data, but accesses this screen anyway.

Termination

When the user returns to homepage.

Post condition

3.1.4. Data Requirements

The system will require back up mechanisms to preserve the data logged.
This will include regular back ups of the main MySQL Database.

Security

The database is protected from malignant attacks by using `mysqli::real_escape_string()` and `strip_tags()`.
Subsequent builds will implement proper password hashing.

3.1.5.Performance / Response time requirement

For aggregate data comparisons, speed of recall will be important

Access transparency will be secured via use of database permission tables

3.1.6.User Requirements

Future iterations user experience is 100% contingent on an intuitive portable experience.

3.1.7. Environmental Requirements

Internet access is a prerequisite to sync dream data to server and to access networked visualisations.

For future phone iterations certain server resources will need to be made available to local app instantiations

Fault tolerance is not envisioned as being essential to this build of the app.

The front end for the system must function in any internet browser

Maintainability: subsequent builds maintainability will be contingent on server size and load: for initial build this is not a necessity.

3.1.8.Usability Requirements

Reliability: the service must always be accessible whenever a user wishes to add data to their dream log

3.2. Design and Architecture

3.2.1.Entity Relationship Diagrams & Data Driven Development

The projects design and architecture followed a data driven development approach. Per the necessity of users logging data standard CRUD functionality was implemented as web service.

Data Driven Development necessitates a well modeled entity relationship diagram. The entity relationship diagram determined the database that was core to this project.

I have split the diagram in two (you can view it full scale in the supplementary files accompanying this submission) to illustrate the reference tables and their relationship to the core tables.

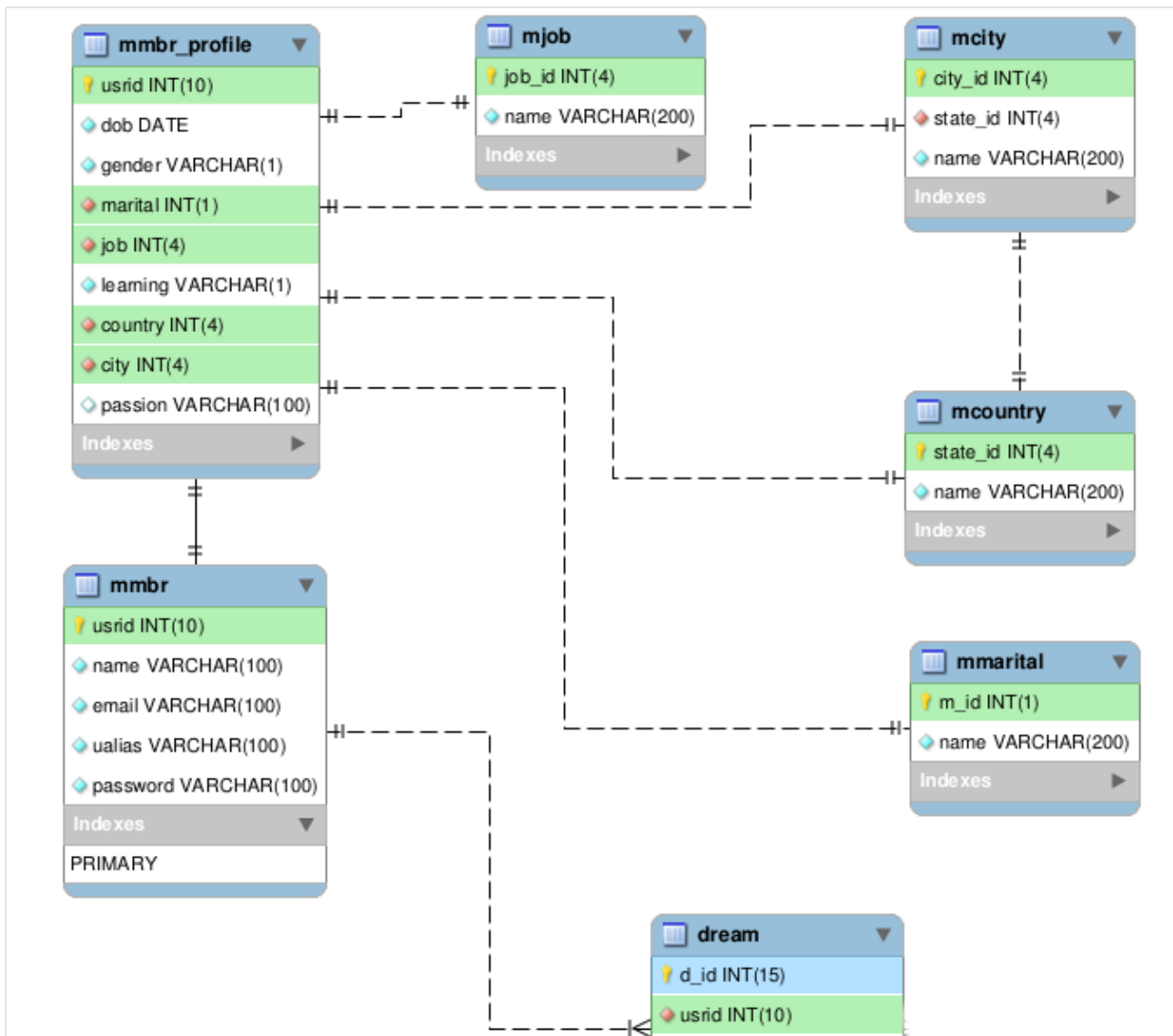


Fig 5 above is part 1 of ER diagram

User details essential to authentication are stored in the **mmbr** table. An additional mmbr_profile table holds the users demographic data, age, gender, location, marital status and employment status. Mjob, mmarital, mcountry and mcity are all reference tables. This allows, for instance in the case of jobs and country&city, for subsequent additions (of rows) to be made to these tables, extending the scope of demographic information attainable.

EG:

Currently registration caters for users who wish to join from Ireland with a county by county

granularity. The rest of Europe is catered for in terms of capital cities.

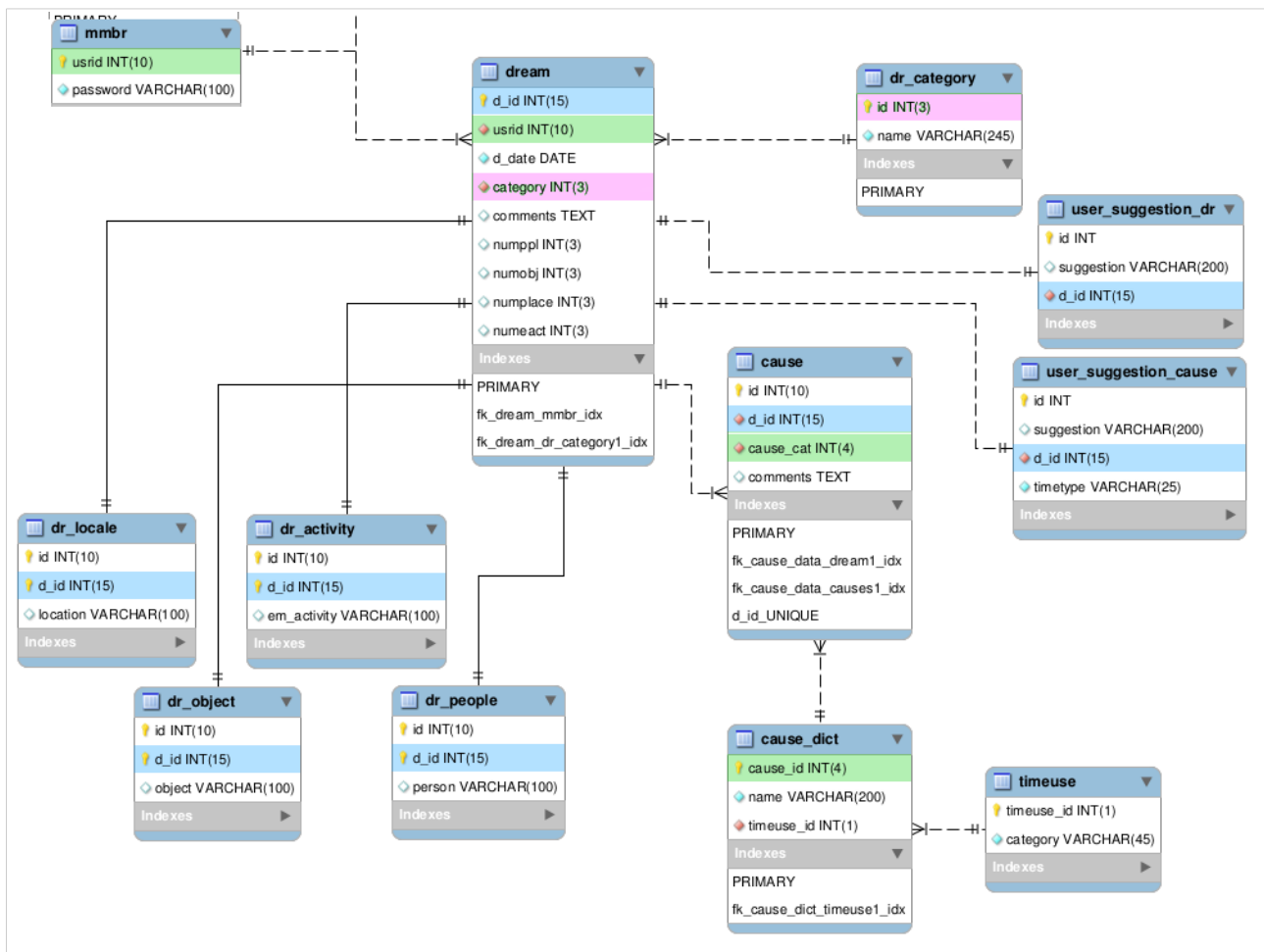


Fig 6 above is part 2 of ER diagram

Dream is the principal table, which stores the date, category and user id of dreams logged on the service. Dreams are taxonomised via 25 categories stored in the reference table dr_category. A similar organisation of categories operates in cause, which logs the users activity prior to dreaming.

The cause table has an additional reference table, timeuse. This reflect the incorporation of timeuse researcher into this project.

The four tables dr_locale, dr_activity, dr_object, and dr_people are not utilised for the current core functionality of the project. However they will be extended in functionality (See [Section 5.2.1](#) for more details).

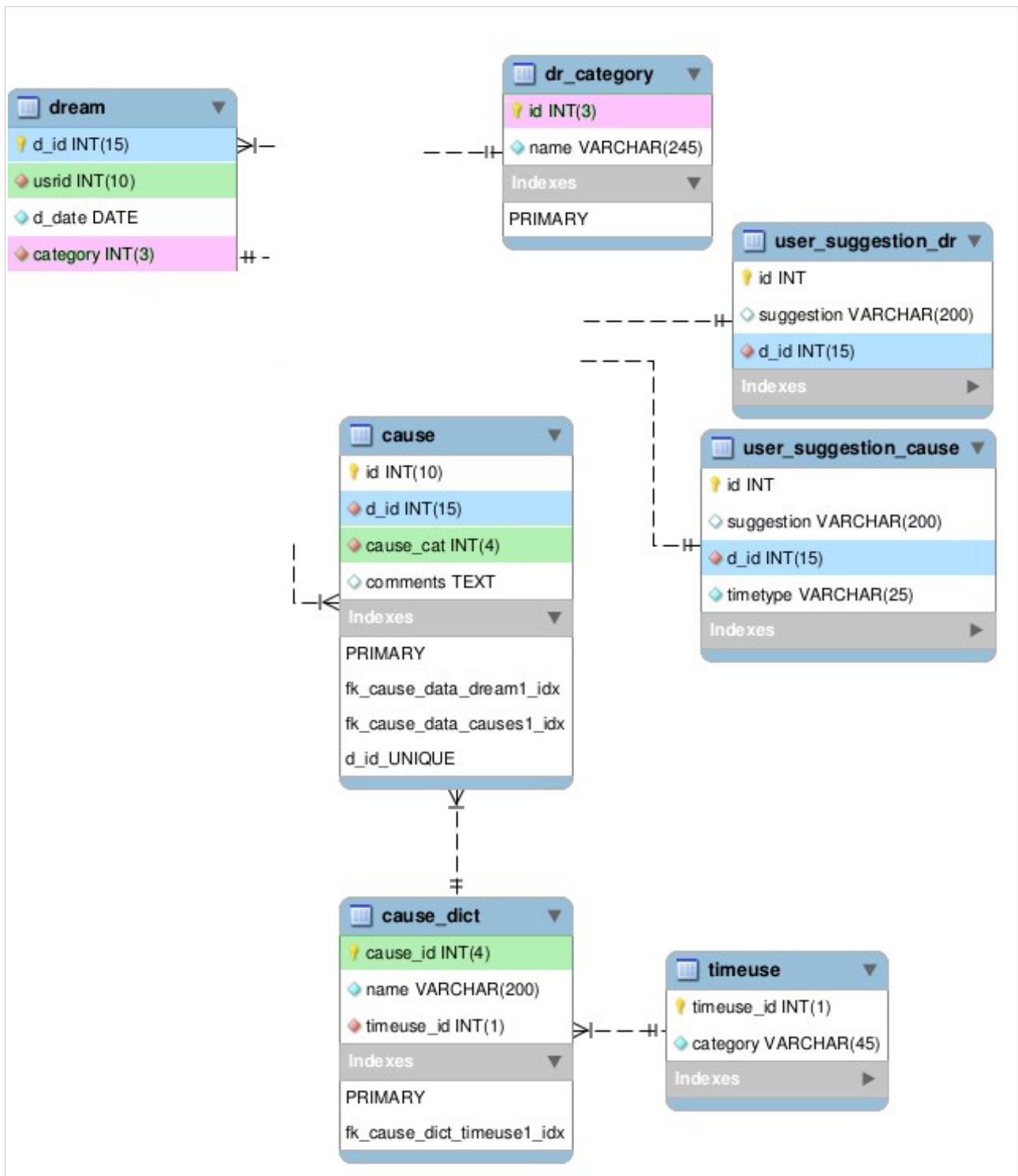


Fig 7 Part 3 of ER diagram (please note that it has not been possible to display the connection lines in the above crows foot diagram, they are the same as illustrated in Par 2)

Both cause and category have the option for the user to offer their own category for the dream, if there are no categories that fit their desires. This is reflected in the tables user_suggestion_cause, and user_suggestion_dr. An earlier implementation that used one

table for all suggestions was revised once time-use-research was incorporated into the project.

3.2.2. User Journey Design

In order to guarantee a user journey that maximises the amount of data gathered it must not be possible for the user to navigate, using the browser, past a screen that captures data.

This was designed in the following manner:

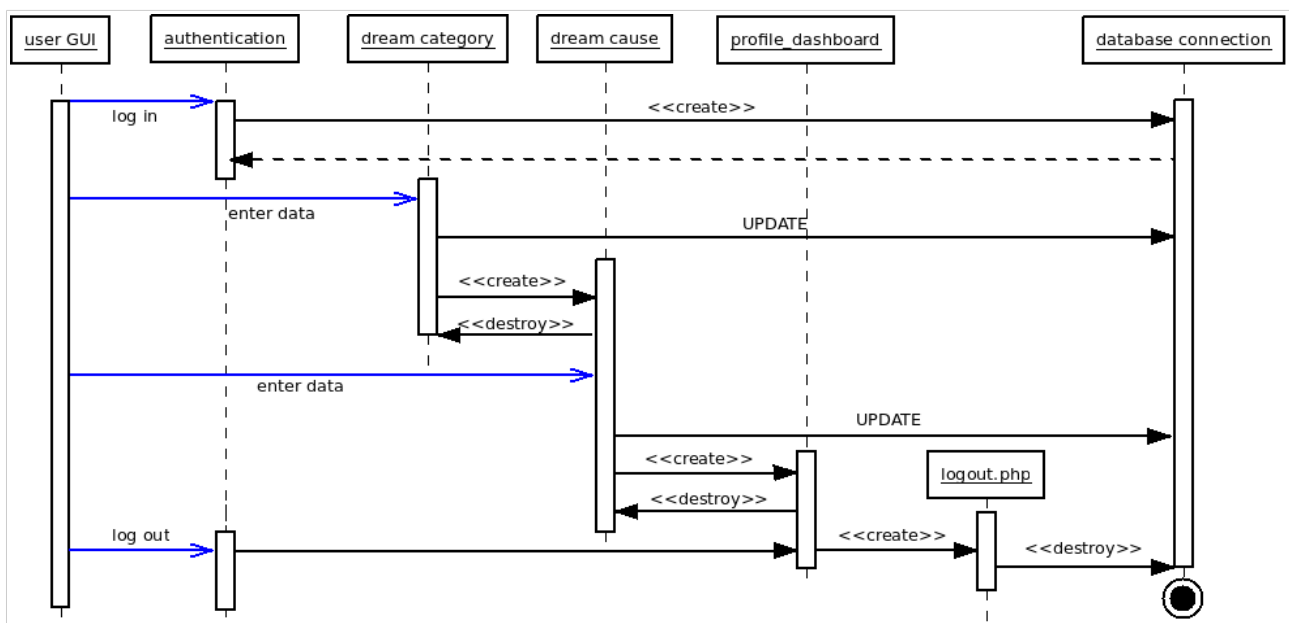


Fig 8 - User Journey Sequence Diagram

This was further designed with additional control flow steps

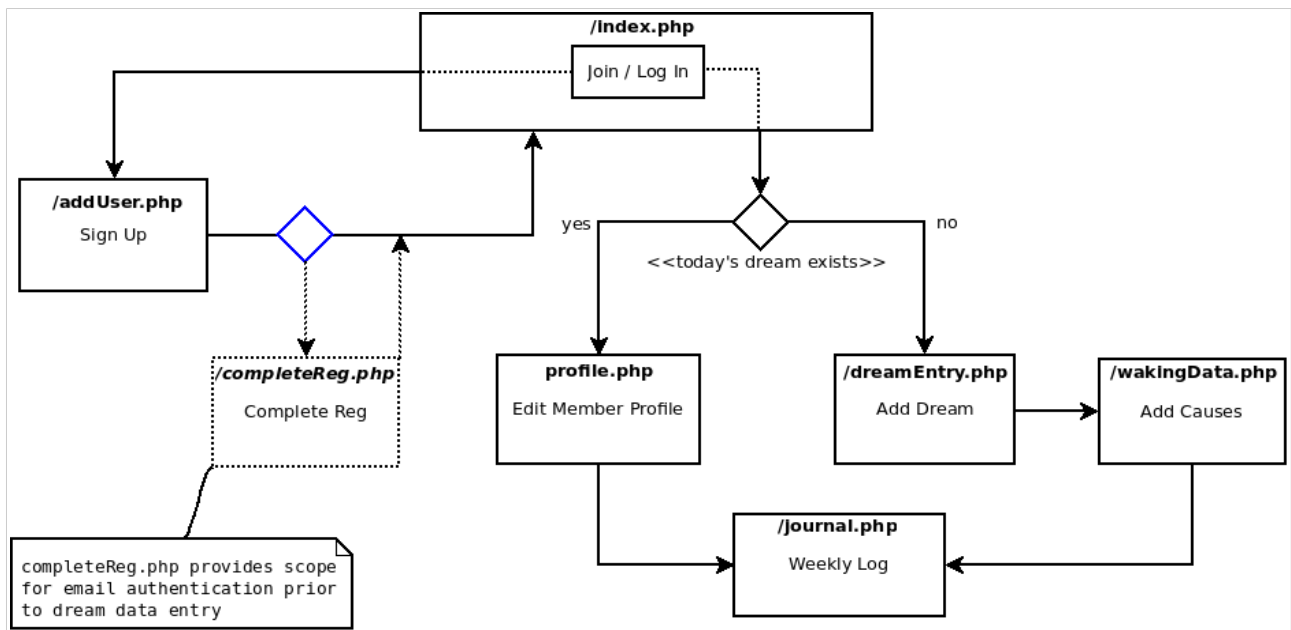


Fig 9 'User Journey' Flow Diagram

The registration process is broken into two steps, indicated above as addUser.php and completeReg.php. This is to permit an immediate sign up and to separate the additional data from the bare minimum data required to begin.

Partitioning the sign up process in this manner permits an entry point for a subsequent email authentication step.

Architecture

The components are architecture of the project conform to this diagram.

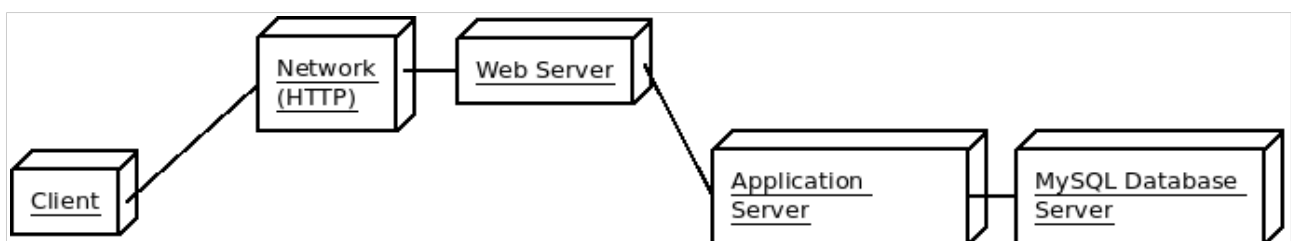


Fig 10 - Deployment Diagram

3.3. Implementation

Per data driven development a well modeled entity relationship diagram was key. Using MySQL workbench I visually (re)designed the database allowed constraints to be added. This both guided the correct procedure for updating tables and their associated tables, and thus reduced my reliance upon PHP error catching correctly updating the appropriate foreign key fields. This ensures a database not likely to be corrected, and ensures users data will be captured.

The project is comprised of PHP files which utilize HTML & CSS for display and javascript for dynamic interactions.

db.php is responsible for generating the dreamDB class.

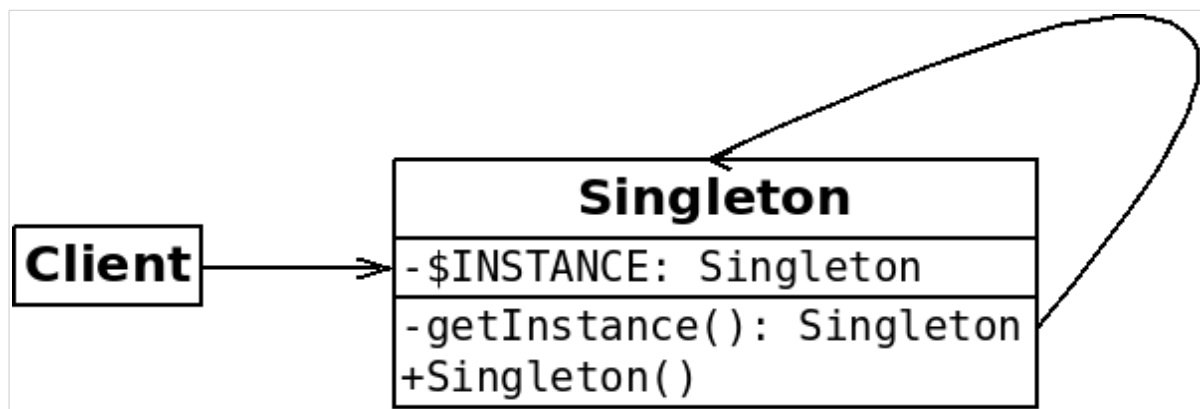


Fig 11 - Singleton Design Pattern

The dreamDB class employs the singleton design pattern, and keeps the connection variables private to itself. The Singleton design pattern can only have one instance and provides a global point of access to itself. It handles the connections to the MySQL and the DML and DDL queries executed in response to the users input. Requests are issued and executed to the MySQL database using the object oriented mode of mysqli.

The user interacts via HTML in a browser, with PHP used to process the users input. All PHP files require an instance of the dreamDB class.

I have learned that much of the time I spent on hand rolling CRUD functionality could be saved through the utilisation of frameworks, such as Smarty, Cake or Yii (given my language choice of PHP). This is a lesson learned for subsequent development. (please see section 3 for further elaboration upon my appraisal of how well suited this development model was)

3.3.1.DreamDB class

Most of the functions in this class pertain to connection, maintaining Singleton Class encapsulation, and 'CREATE', 'READ' and 'UPDATE' functions.

A private function handles the formatting of text values entered to the database. It removes whitespace and other alphabetical-character-separating characters and replaces them with underscores. It additionally converts all text to lower case. This ensures consistency of comparison queries executed on the database.

```
177 private function formatting($param){  
178     $param = trim($param);  
179     $param = preg_replace('/-/','_', $param);  
180     $param = preg_replace('/ /','_', $param);  
181     $param = preg_replace('/^[^\\w\\']+|\\'(?![\\w])|(?![\\w])\\'/', '', $param);  
182     $param = strtolower($param);  
183     return $param;  
184 }
```

Fig 12 - snippet formatting function, source Includes/db.php

There is one instance where the DreamDB class encapsulation may be broken, in terms of one of the the PHP pages also knowing the names of the tables being updated.

This is located within logdream.php, where the comma separated values entered in the form fields 'Embodied Activity', 'People', 'Objects' and 'Places' are allocated into an associative array, which uses the MySQL table names as KEY indexes.


```

$id = $_SESSION['uid'];
echo "trying to add data " . $_POST['category'] . " " . $_POST['comment'] . " for user " . $id . "<br>";
$form = array(
    'ea' => array(
        'TABLE' => "activity",
        'VALS' => array()
    ),
    'ppl' => array(
        'TABLE' => "people",
        'VALS' => array()
    ),
    'obj' => array(
        'TABLE' => "object",
        'VALS' => array()
    ),
    'plc' => array(
        'TABLE' => "locale",
        'VALS' => array()
    )
);
foreach($_POST as $key=>$value){
if($value){ // provided the form has text, grab the data
$form[$key]['VALS'] = explode(',',$value);
}
}
$d_id = dreamDB::getInstance()->get_dreamid_by_uid($_SESSION['uid']);

foreach ($form as $i) {
    if(sizeof($i["VALS"])>0){
        //dreamDB::getInstance()->tables($d_id, $i["VALS"], $i["Q"]);
        $x = dreamDB::getInstance()->many($d_id,$i["VALS"], $i["TABLE"]);
    }
}
echo "end of many function<br>";

```

Fig 13 - Snippet, comma separated values to associative array, source logdream.php

Using an associative array in this manner is favoured for the sake of DRY (don't repeat yourself) within the DreamDb class. The **many** function can update several different MySQL tables (which share the same column names and structure) through the one function.

```

158 public function many($d_id, $array, $table){
159     $string = "";
160     $qID = $this->query("SELECT MAX( id ) +1 FROM dr_.$table"); // get query object
161     $row = $qID->fetch_row(); // create an array listing of above object
162     $id = $row[0]; // access item indexed at 0 for value desired
163     foreach($array as $e){
164         $entry = $this->real_escape_string($e); // user input
165         $entry = $this->formatting($entry);
166         $this->query("INSERT INTO dr_.$table." VALUES('".$id."','".$d_id."','".$entry."')");
167         $id++;
168     }
169     //return $string; - if debug
170 }

```

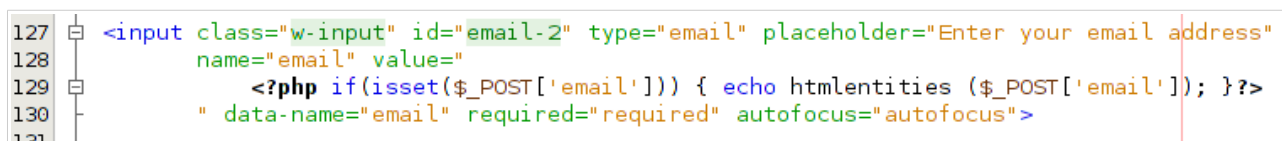
Fig 14 - Snippet DreamDB function to handle csv associative array, source Includes/db.php

3.3.2. User Journey

It was earlier noted in [Section 3.2.2 User Journey](#) that the partitioning of sign up allows an entry point for subsequent email authentication.

It is also a matter of expediency. The partitioning ensures that there is no entry field within complete.php that would cause a page refresh that prompts the user to reenter details (as is the case with password, email and username on join.php).

For instance, within join.php (the file that handles the users signing up) several fields must be subject to server side validation checks, necessitating the reload of the page. To prevent a poor user experience, only those fields which fail the validation are re-presented as empty. The fields that pass are preserved, through the following code:



```
127 <input class="w-input" id="email-2" type="email" placeholder="Enter your email address"
128       name="email" value="
129       <?php if(isset($_POST['email'])) { echo htmlentities ($_POST['email']); }?>
130       " data-name="email" required="required" autofocus="autofocus">
```

Fig 15 snippet, preserving user input on form refresh, source join.php

By contrast complete.php consists nearly entire of dropdown selection menus. Preserving user selection here is not possible, as many of these forms are lengthy (Country, City, Job) and to save on both repetition of code and improve code legibility these fields are autopopulated by php files (contained in Includes/functions.php). Preserving the users selection upon a form refresh (as is necessary with the server side validation performed in join.php) is difficult to implement.

3.3.3. Sessions

In [Section 3. Design and Architecture](#) and [Section 3.2.2 User Journey](#) the user journey was shown in abstract form

Here is a sequence diagram illustrating the PHP implementation of that same logic

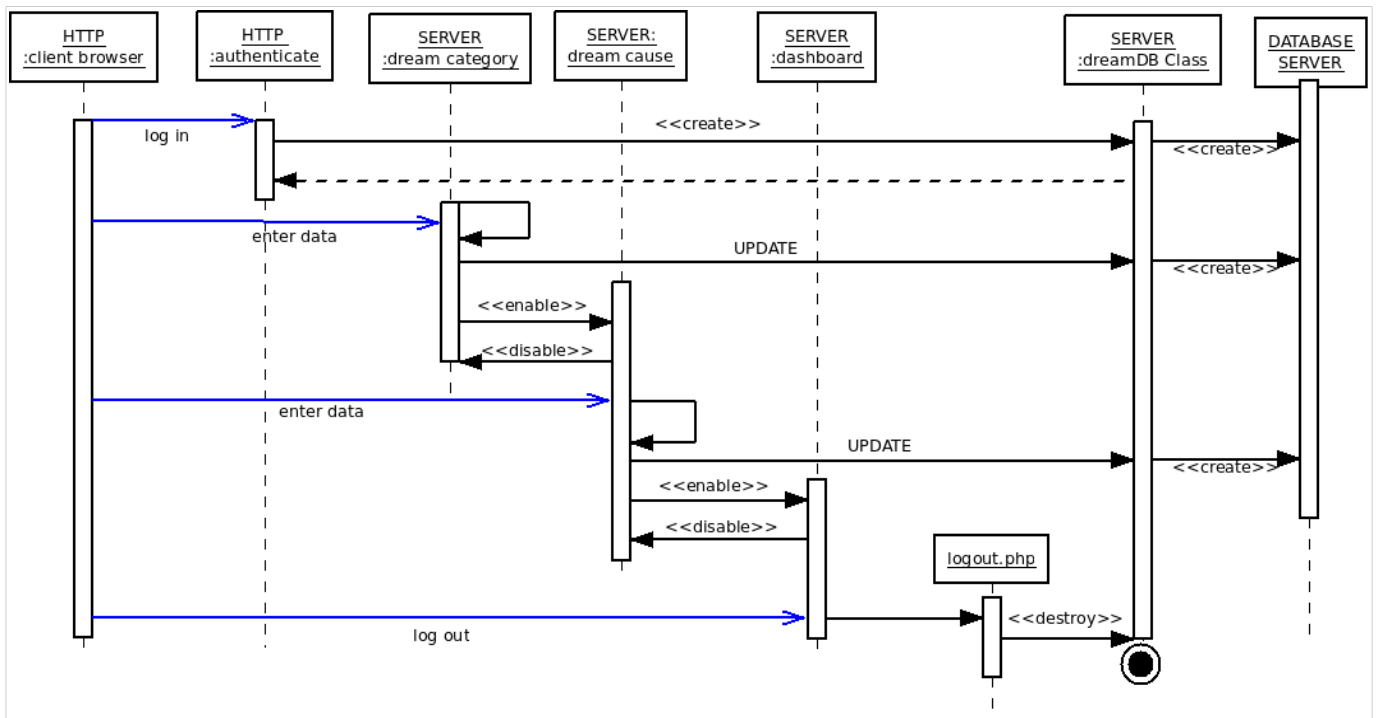


Fig 16 - 'User Journey' PHP Sequence Diagram

Where in Fig 8 p29 the user journey was envisioned as being controlled by destruction of data entry objects it is here accomplished by rendering pages inaccessible through session variables. Sessions, a feature of PHP, are used as part of the sites control flow which guarantee a certain user flow, as indicated by this sequence diagram.

Several code sections have been written to issue header redirects dependent on whether certain variables have been saved & set

The collaboration would operate as follows

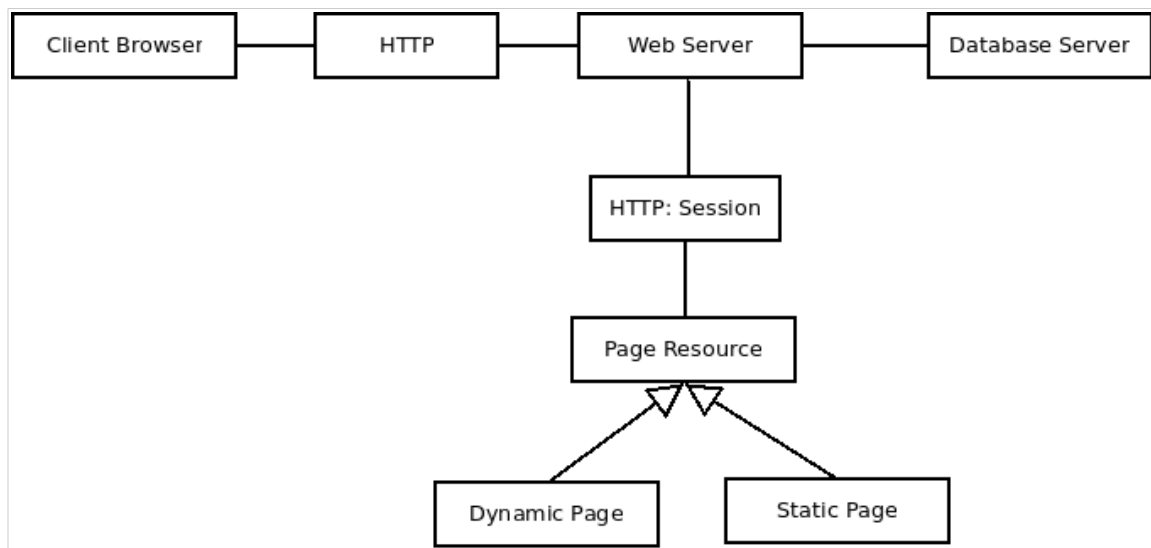


Fig 17 - Collaboration Diagram

Sessions generate a unique id (UID) for each visitor. It permits storage of bespoke variables, through an associative array called `$_SESSION`. A detailed activity diagram of how the projects PHP files methods and variables orchestrate the 'User Journey' control flow follows.

In the subsequent diagram the pages methods can all modify the `$_SESSION`. You can conceptualize it in object oriented terms, even though that is not strictly what is happening. I did so, to work with UML and also as an aid to ordering the logic of my pages.

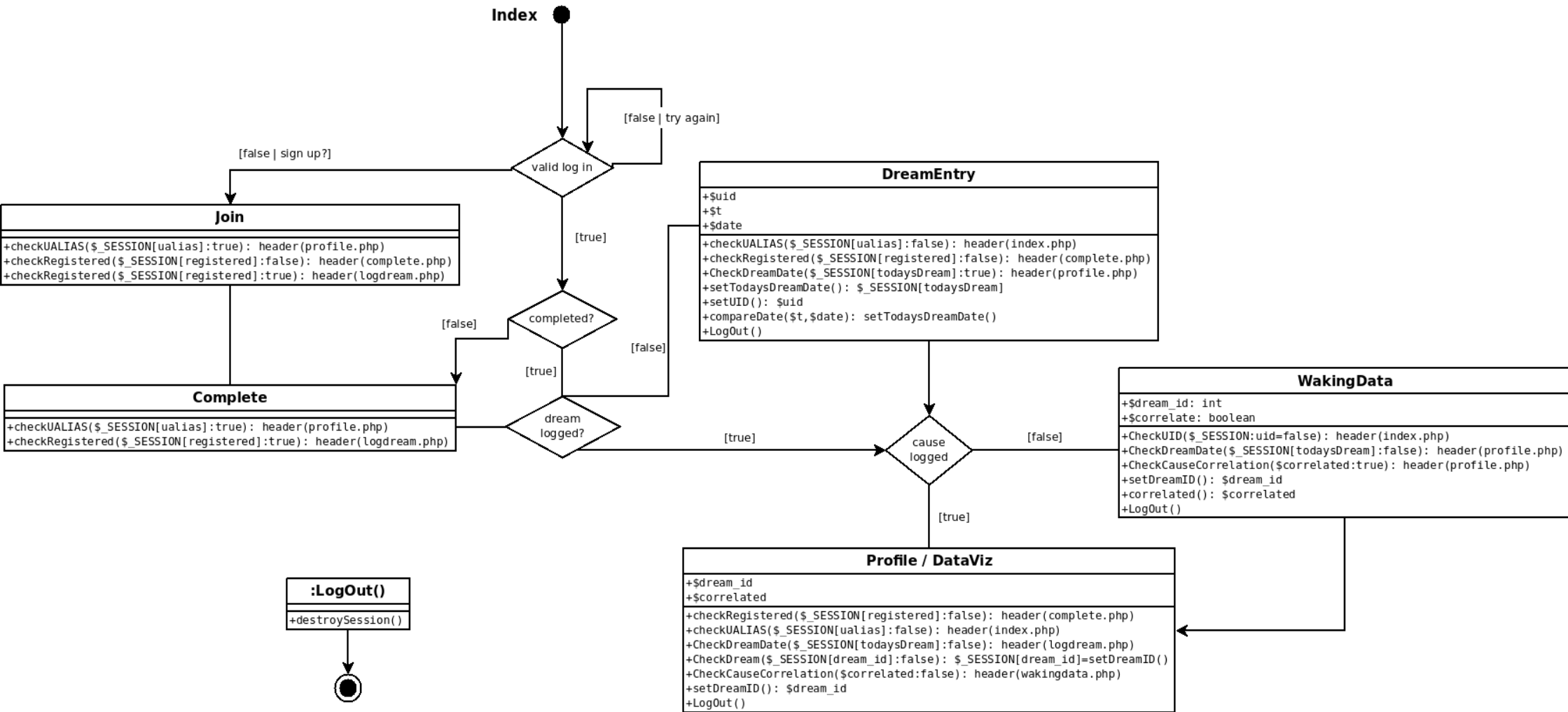


Fig 18 \$_SESSION flow diagram. Diagram Note - Profile and DataViz share the exact same control flow methods, and for the sake of space they have been represented as the one object above. The author acknowledges that this would not be proper practice in actual software engineering practice

3.3.4. Notes on Control Flow

Most of the session assignment consists of checking whether there is correspondence, in terms of primary and foreign keys or of other unique indexed columns, of a given table to determine whether a new update operation is permitted.

The one area which requires different logic is the detection of whether a users dream date is current or not. MySQL records Date in a manner distinct to how date can be recalled via PHP. A conversion of these two would never equate. Therefore a conversion to epoch time, limited to the YYYY-MM-DD is necessary in order to determine if a dream has already been logged for the given date (at time of login).

```
83      // second session check - has a dream been logged for today
84
85      $id = dreamDB::getInstance()->get_uid_by_alias($_SESSION['ualias']);
86      // retrieve user id
87      $_SESSION["uid"] = $id;
88      $date = dreamDB::getInstance()->most_recent_dream($id);
89      // retrieve the date of the most recently entered dream for assigned user id
90      $t = date("Ymd", time());
91      if (strtotime($t) == strtotime($date)) {
92          // if a dream has already been entered
93          $_SESSION["todaysDream"] = true;
94          header('Location: profile.php'); // redirect to member profile page
95      } else {
96          $_SESSION["todaysDream"] = false;
97      }
```

Fig 19 snippet - ensure one dream per 24 hours Source: logdream.php

Javascript

The javascript used client side is rudimentary/perfunctory. It is ad-hoc and designed to be as functional as possible. It lends dynamism where needed. For instance it enables the population of 'cities' based on the country selected on the same screen

```

63 <script src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>
64
65 <script>
66     $(function() {
67         // passes parameter via GET call to method in functions.php
68         $("#country").change(function() {
69             // when change detected in CSS selector ID: country
70             $("#city").load(
71                 // populate CSS selector ID: country with data returned via
72                 // GET request made to PHP file specified
73                 "Includes/functions.php?choice=" + $("#country").val()
74             );
75         });
76     });
77 </script>

```

Fig 20 snippet - dynamically populate forms using jQuery & PHP source - complete.php

I am convinced that beginning with jquery earlier in a projects development would lend itself to much cleaner and better encapsulated code.

HTML & CSS Design and Residual from WebFlow

The WebFlow visual editor was used to construct the HTML and CSS for this project. Webflow provides comprehensive javascript client side validation and server side handling - however as I had coded my own backend in PHP none of this functionality was included.

WebFlow provides a very neat solution to responsive design which I have availed of.

The sophisticated and visually appealing animations and transitions possible via Webflow have not been used in the GUI. I would need to ensure for graceful degradation across a range of browsers and a plethora of devices - this represented an increase in the amount of time for GUI implementation disproportionate to the time I had available to deliver the project.

3.3.5.Data Analytics & Visualisation

In order to implement the functionality of data analytics requirements it was necessary to populate the database with pseudo entries.

Data Scraping

Code snippets and the steps implemented to populate the database are detailed in [Appendix Section 7.3](#). Several command line executable scripts were necessary. These scripts were written in PHP and utilise CURL to fetch information from the internet.

Database

When querying the database during data analytics a view is created, to make demographics queries easier.

Members date of birth is stored as YYYY-DD-MM. It is cumbersome to query that format. Therefore the created view calculates the age based on the entered data of birth.

```
1 CREATE VIEW demographic AS
2 SELECT m.usrid, m.gender, m.marital, m.job
3 , DATE_FORMAT(FROM_DAYS(TO_DAYS(NOW())-TO_DAYS(m.dob)), '%Y')+0
4 AS age
5 FROM mbr_profile m
6
7 -- querying the view
8
9 SELECT COUNT(d.d_id) as occurrences, m.gender, m.marital, m.job
10 ,m.age
11 FROM dream d
12 INNER JOIN demographic m ON d.usrid = m.usrid
13 WHERE d.category = $PARAMETER
14 -- AND m.gender = 'M' || m.gender = 'F'
15 -- AND m.marital = $INDEX
16 -- AND m.job = $INDEX
17 -- AND m.age BETWEEN 18 AND 24
18 GROUP BY d.usrid
```

Fig 21 - SQL, create view demographics

Data Viz

The project deploys the D3 javascript visualisation library. (see Section 1.3 Libraries for further details). This allows the dynamic generation of visualisations based on data from the database.

As is convention for the project DreamDB handles the retrieval of data from the database. In accordance with the server side / client side cooperation of PHP and Javascript (respectively) an intermediary PHP file has been used in order to dynamically access data based on the selections made by the user.

In a manner similar to the jQuery snippet illustrated above in [Section 3.3.4 Notes on Control Flow](#) changes in a dropdown menu are detected when the submit button is pressed.

```
42 <form class="selector_box" action="#" onsubmit="return formUpdate(this)">
43   <label class="dream_selector_label" for="dreams">Show me who dreams about:</label>
44   <select class="w-select dream_selector" id="dreams" name="category" data-name="Dreams">
```

Fig 22 snippet -dynamically visualise data part 1 Source dataviz.php

This invokes the formUpdate functions

```
176 function formUpdate(form){
177
178   var url = parseInt(form.category.value);
179   d3.json("Includes/jsonFetch.php?val="+url, function(json) {graph(json)});
180   console.log("within the db.json section");
181
182   function graph (jsondata) {
183     var data = jsondata;
184
```

Fig 23 snippet -dynamically visualise data part 2 Source dataviz.php

formUpdate is a function that contains calls to two subsequent functions

d3.json(); // part of d3 library (see below for function definition)

graph(jsondata); // user defined

// function definition

```
d3.json = function(url, callback) {
  d3.text(url, "application/json", function(text) {
    callback(text ? JSON.parse(text) : null);
  });
};
/*
callback = function(json) {graph(json)}; in above diagram
*/
/
```

Complicating matters a little is the use of a Javascript anonymous function call, illustrated in the above diagram with the curly braces highlighted yellow `{ }`. The function definition is included for reference.

The x Axis of the visualisation is defined as a linear scale, with a domain between 0 and 100. This allows the data returned to be visualised as percentages.

EG when you query what people did before dreaming of 'abandonment or betrayal', the bars of the bar chart are scaled according to a percentage of the total amount of those who dreamed of 'abandonment or betrayal'.



Fig 24 snippet -dynamically visualise data part 1 Source *dataviz.php*

The function 'returns false' in order to prevent a screen reload, which would prevent the graph from being displayed.

3.4. Testing

My test plans followed an ad-hoc testing process [note examples on a spreadsheet approach]. The possibilities for scripting tests was diminished due to the need to check PHP, javascript and other interactions as they unfolded in the browser.

Before detailing the process it is worth noting the two notable edge cases identified

through the process which lack a resolution at this point (see section 'Further development or research' for how these are planned to be remedied).

The edge cases relate to the User Journey which, as explained in [Section 3. Design and Architecture](#) and [Section 3.2.2 User Journey](#) has been instantiated via the `$_SESSION` array. One notable edge case surfaced.

Edge Case: One Dream a Day.

In order to ensure that users do not enter multiple dreams on a given day several control flow structures are in place. However an edge case occurs between 00:00 and 01:00 GMT where the comparison function (detailed below) cannot reconcile a day having passed.

To test the CSS of the website, I went to <http://jigsaw.w3.org/css-validator/> and used the tool there to check the CSS.

To Test the HTML I accessed <http://validator.w3.org/> and used the tool there to test the HTML. For HTML purposes the PHP contents of the file were commented out?

My project doesn't implement a great deal of classes. In future revisions, PHPUnit would present a useful way of testing this project.

index	TECHNOLOGY	FUNCTIONALITY	ACTION	PARAMS	EXPECTED	RESULT
1	PHP: \$_SESSION	Ensure that @URLS(join.php, complete.php) can only be accessed when a registered user is not logged in	visit each page when logged in			
1.1	\$_SESSION	[[same as above]]	index.php	HTTP interaction	redirect:profile.php	PASS
1.2	\$_SESSION	[[same as above]]	join.php	HTTP interaction	redirect: profile.php	PASS
1.3	\$_SESSION	[[same as above]]	complete.php	HTTP interaction	redirect:profile.php	PASS
1.4	\$_SESSION	[[same as above]]	login.php	HTTP interaction	redirect:profile.php	FAIL
1.2.1	\$_SESSION	EDGE CASE [[for above scenario]]	visit join.php when redirected to complete.php	HTTP interaction	redirect:profile.php	FAIL
1.2.2	\$_SESSION	EDGE CASE: Ensure that @URLS(join.php, complete.php) can only be accessed when a registered user is not logged in, and that join.php is inaccessible once the db update it triggers has occurred	invoke function DreamDB->create_user_debug(). Log in. Visit join.php	HTTP Interaction – FORM param {username: test, pwd: test}	redirect: complete.php	PASS
2	\$_SESSION	Ensure that @URL(logdream.php, wakingdata.php, profile.php, dataviz.php) can only be accessed when a registered user logged in	visit each page when not logged in			
2.1	\$_SESSION	[[same as above]]	logdream.php	HTTP interaction	redirect: index.php	PASS
2.2	\$_SESSION	[[same as above]]	wakingdata.php	HTTP interaction	redirect: index.php	PASS
2.3	\$_SESSION	[[same as above]]	profile.php	HTTP interaction	redirect: index.php	PASS
2.4	\$_SESSION	[[same as above]]	dataviz.php	HTTP interaction	redirect: index.php	PASS
2.5	\$_SESSION	[[same as above]]	logout.php	HTTP interaction	redirect: index.php	PASS

index	TECHNOLOGY	FUNCTIONALITY	ACTION	PARAMS	EXPECTED	RESULT
3	:\$_SESSION	Ensuring that User Journey Control sequence is preserved on event of logout prior to User Journeys Completion	logging out at various pages of the User Journey sequence (without completing them), and logging back in to ensure that the flow is preserved	HTTP interaction		
3.1	:\$_SESSION	[[same as above]]	logdream.php	HTTP interaction	Auto redirected to logdream.php	PASS
3.2	:\$_SESSION	[[same as above]]	wakingdata.php	HTTP interaction	Auto redirected to wakingdata.php	PASS
3.3	:\$_SESSION	[[same as above]]	profile.php	HTTP interaction	Auto redirected to profile.php	PASS
3.4	:\$_SESSION	[[same as above]]	logdream.php	HTTP interaction	Auto redirected to wakingdata.php	PASS
				DETAILS	see edge case 3.5 below	
3.5	:\$_SESSION	EDGE CASE – Todays Dream Check: Midnight.	Ensure that two dreams cannot be logged within 24 hours. Testing epoch time conversions to ensure that session flow operates as planned	HTTP interaction	that epoch time function will recognise when the DD portion of a YYYY-MM-DD is 24 hours different to a record in the database	
3.5.1	:\$_SESSION	EDGE CASE	testing dream logging (logdream.php & wakingdata.php)user journey around 23:00 to 01:00	HTTP interaction	PASS	FAIL
				DETAILS	Database logs dates based on GMT, which has an hours difference in discerned epoch time.	

index	TECHNOLOGY	FUNCTIONALITY	ACTION	PARAMS	EXPECTED	RESULT
3.5.1	\$_SESSION	EDGE CASE If user has successfully registered without inputting username and password	add two dreams for the same date to database, visit logdream.php		PASS	PASS
				DETAILS	user not permitted to enter dream, but the header redirects breakdown	
4	\$_SESSION	EDGE CASE	see PHP test 8.1			
				DETAILS	redirects go haywire, but no page will load. See Test 6	
5	\$_SESSION	Ensure session time out acts as programmed	set time out to 30 seconds in epoch, attempt to access pages	HTTP interaction		
6	\$_SESSION	[[same as above]]	when redirect logic malfunctions	HTTP & PHP interaction	redirect to a 404 or other error page	FAIL
				DETAILS	the header logic which is rewritten for each file may be a problem to solving this	
7	PHP	Check client side validation of join.php @FIELDS(username, password, password_confirmation)				
7	PHP	[[same as above]]	disable javascript, leave username vacant	HTTP interaction	page reload, prompt username required	PASS
7.1	PHP	[[same as above]]	disable javascript, leave password vacant	HTTP interaction	page reload, prompt password required	PASS
7.2	PHP	[[same as above]]	disable javascript, leave password2 vacant	HTTP interaction	page reload, prompt 'please confirm password'	PASS
7.3	PHP	[[same as above]]	disable javascript misspell one of the two password fields	HTTP interaction	page reload, prompt password mismatch	PASS

index	TECHNOLOGY	FUNCTIONALITY	ACTION	PARAMS	EXPECTED	RESULT
7.4.1	PHP	EDGE CASE	add a user name with quote marks, which would disrupt the query wrapping and format conducted by the DreamDB class	USERNAME: username'test	error due to quote escapes	FAIL
				DETAILS	good news for robust regex	
7.5	PHP	does form preserve data if the wrong fields are entered incorrectly	disable javascript, enter username that is already taken	HTTP interaction	Email & name to retain values	PASS
7.6	PHP	[[same as above]]	disable javascript misspell one of the two password fields	HTTP interaction	Username email & name to retain values	PASS
8	PHP HTML					
8	PHP HTML	ensuring that if no value entered for a form field, nothing is added to DB	disable javascript, join.php	HTTP interaction	that DB is not updated	FAIL
				DETAILS	HTML5 recognises that the email field is blank, but whitespace will suffice on the other entry points	
8.1	PHP HTML	ensuring that if no value entered for a form field, nothing is added to DB	disable javascript, complete.php	HTTP interaction	that DB is not updated	PASS
				DETAILS	however, the page can progress and alters the session such that sequence advances to logdream, where multiple dreams can be entered	
8.2	PHP HTML	EDGE CASE: user has advanced to logdream.php without fulfilling mabr_profile table	disable javascript logdream.php	HTTP interaction	that DB is not updated	FAIL
				DETAILS	\$_SESSION[todaysDream] remains unset, user cannot progress to wakingdata.php or profile.php	
8.3	PHP HTML	ensuring that if no value entered for a form field, nothing is added to DB	disable javascript logdream.php	HTTP interaction	that DB is not updated	PASS

index	TECHNOLOGY	FUNCTIONALITY	ACTION	PARAMS	EXPECTED	RESULT
8.4	PHP HTML	ensuring that if no value entered for a form field, nothing is added to DB	disable javascript wakingdata.php	HTTP interaction	that DB is not updated	FAIL
				DETAILS	optimal behaviour – if no entry it reloads but doesn't commit to DB. This can be circumvented by clicking submit however	
9	javascript	disable JS test pages for expected deprecated functionality and logic				
9.1	javascript	[[same as above]]	index.php: log in	HTTP interaction	PASS	FAIL
				DETAILS	nav menu doesn't degrade	
9.2	javascript	[[same as above]]	index.php: join	HTTP interaction	FAIL	FAIL
9.3	javascript	[[same as above]]	login.php	user who has not logged dream	redirect to logdream.php	PASS
9.4	javascript	[[same as above]]	login.php	user who has logged dream	redirect to profile.php	PASS
		[[same as above]]	SOLUTION: problem with logic of login.php, fixed – see user journey below			
9.5	javascript	[[same as above]]	join.php: submit		PASS	FAIL
				DETAILS	header redirect to complete.php fails	
9.6	javascript	[[same as above]]	join.php: password prompt		FAIL	FAIL
9.7	javascript	[[same as above]]	join.php: email detection		FAIL	PASS
9.8	javascript	[[same as above]]	complete.php: cities		FAIL	FAIL
9.9	javascript	[[same as above]]	complete.php: radio button		FAIL	PASS
10	javascript	[[same as above]]	logdream.php			
10.1	javascript	[[same as above]]	logdream.php: submit	Val = 3	PASS	FAIL
10.2	javascript	[[same as above]]	logdream.php: Submit & comments field as sanity check	FORM Val = 1 l: Comments = sanity	PASS	PASS

index	TECHNOLOGY	FUNCTIONALITY	ACTION	PARAMS	EXPECTED	RESULT
10.3	javascript	[[same as above]]	logdream.php: submit suggestion & comments field as sanity check	FORM VAL = 100 : comments = sanity	PASS	FAIL
				DETAILS	updates dream table fine, also updates dream suggestion with blank suggestion. Acceptable	
10.4	javascript	[[same as above]]	wakingdata.php: radio button		FAIL	FAIL
10.5	javascript	[[same as above]]	wakingdata.php: select boxes		FAIL	FAIL
10.6	javascript	[[same as above]]	wakingdata.php: submit	radio button = free time Comments = sanity	FAIL	PASS
				DETAILS	logs as 0	
10.7	javascript	[[same as above]]	wakingdata.php: submit & comments field as sanity check	radio button = no free time comments = no free time	PASS	PASS
				DETAILS	logs cause cat as 0	
10.8	javascript	[[same as above]]	dataviz.php: select data for comparison		FAIL	FAIL
10.9	javascript	[[same as above]]	logout.php	destroy session,	PASS	PASS

3.5. Graphical User Interface GUI Layout

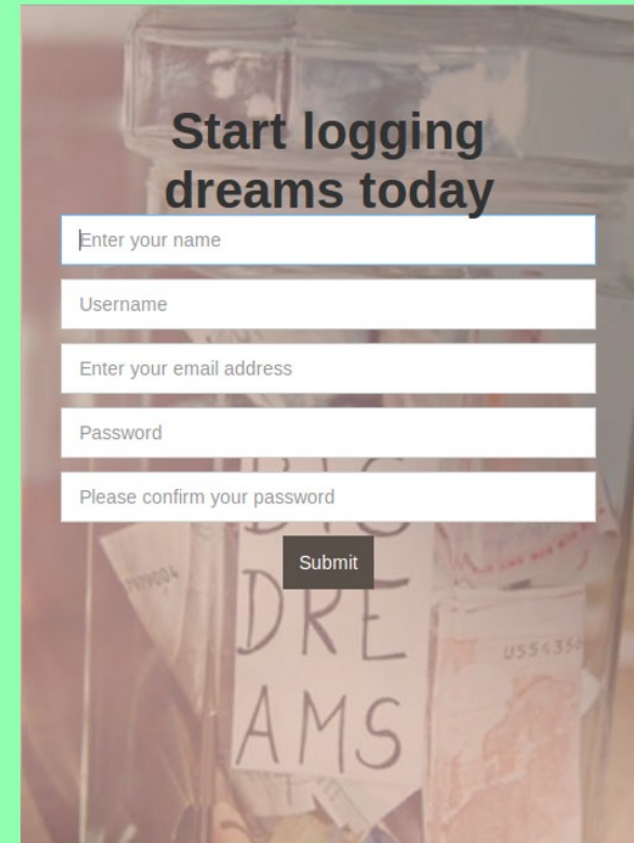
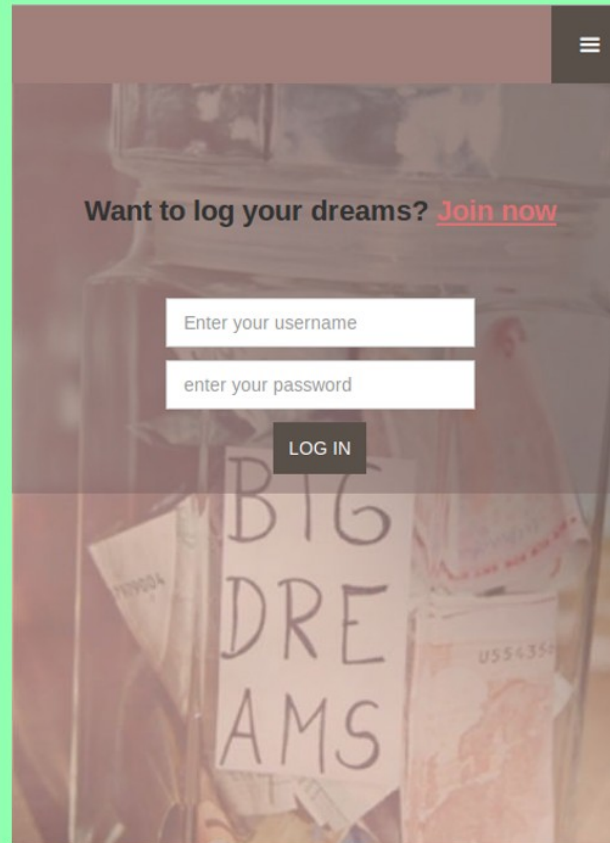
The GUI for this application runs in a web browser. The application utilises responsive design to be functional on both desktop browsers, tablet browsers and mobile browsers. Both desktop and mobile screens are included in side by side comparisons. The key flows are demonstrated here.

Sign Up



Phone - Portrait Affects 479px and below

Phone - Portrait



Sign Up cont...

Phone - Portrait
Affects 479px and below

Start logging dreams today

Enter your name

Username

Enter your email address

Password

Please confirm your password

Submit

Start logging dreams today

test

test

test@te

.....

.....

Submit

Usernames must be at least 5 characters.

OK

Nearly there!

Complete to start logging

Age:

January

1

1914

Gender

male

Relationship

Single

Your Profession

Art / music / writing

Currently learning?

yes

no

Country

Ireland

City

Kildare

Your Passion!

Your Passion

Complete

52

Dream Entry

The image displays three mobile app screens for a 'Dream Entry' application. A vertical bar on the left indicates the 'Phone - Portrait' view, affecting widths of 479px and below. Each screen has a top navigation bar with 'Home' and 'Log Out' links.

Screen 1: enter last nights dream

- Category:** A dropdown menu with the placeholder text 'Select dream category...'.
- Embodied Activity:** A text input field with the placeholder text 'enter the emotions and actions from your d'.
- People:** A text input field with the placeholder text 'enter people from your dreams'.
- Objects:** A text input field with the placeholder text 'enter objects you remember from your drea'.
- Places:** A text input field with the placeholder text 'enter the locations in your dream'.
- Comments:** A text input field with the placeholder text 'any additional comments'.
- A 'Submit' button is located at the bottom right.

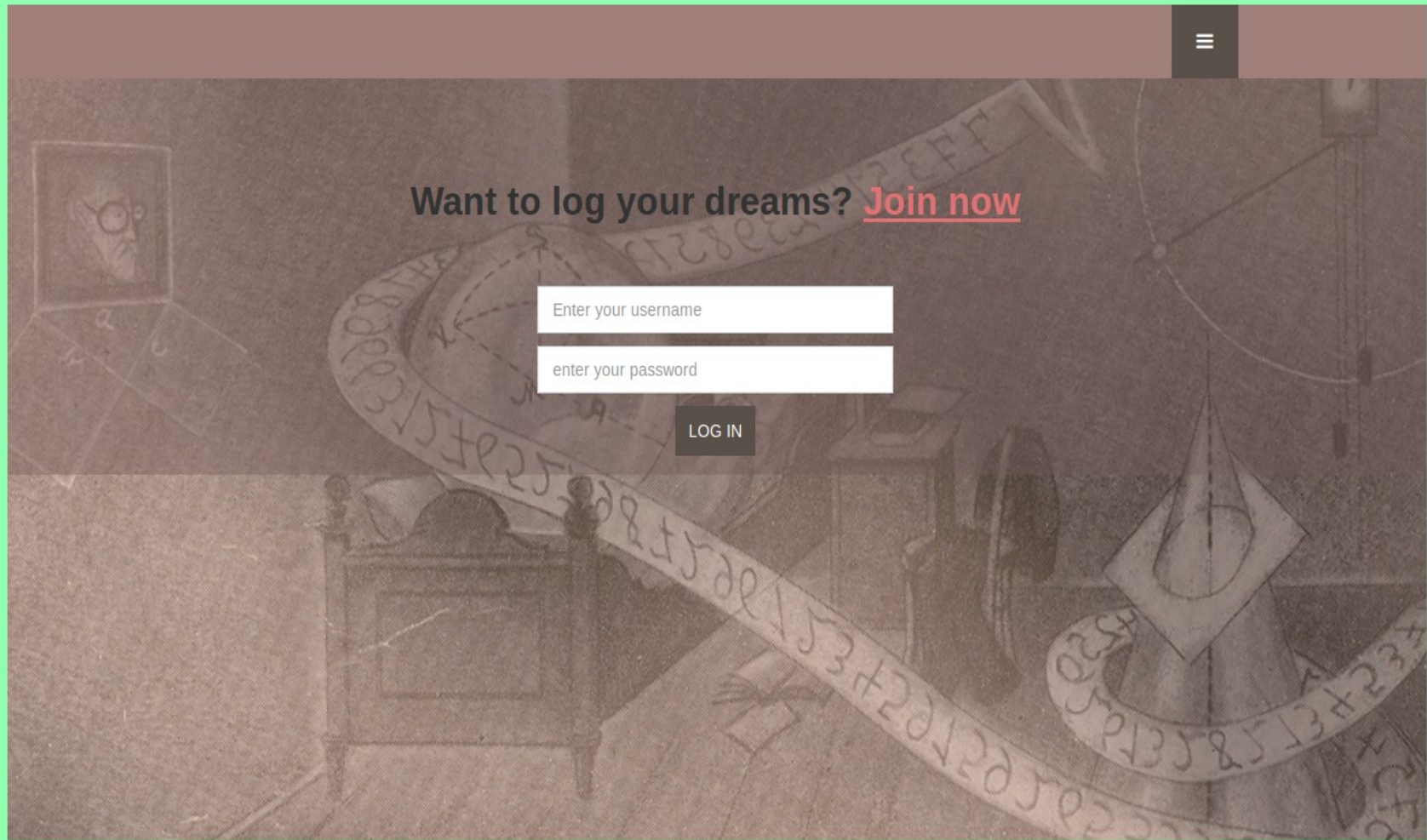
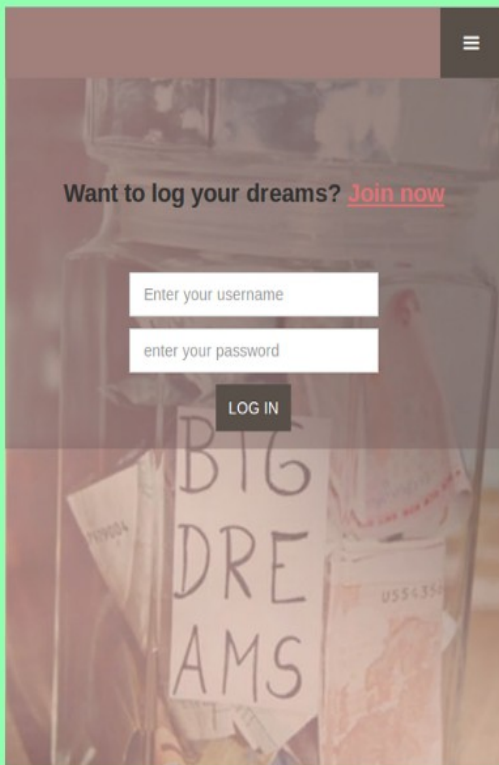
Screen 2: what did you do yesterday, before sleeping?

- Did you have free time before bedtime?** A radio button group with options 'Free Time?' and 'No Free Time?'. The 'No Free Time?' option is selected.
- A text input field with the placeholder text 'No free time huh?'.
- Comments:** A text input field with the placeholder text 'Example Text'.
- A 'Submit' button is located at the bottom right.

Screen 3: User Dashboard

- User name:** Jay. A link for 'Account Settings' is provided.
- Dream Data:**
 - Top Dream Topic: **losing control, forgetting, or disintegration**
 - Top Dream Person: Benjamin
- Last Fortnights Dream Entries:** A bar chart showing the frequency of dream entries over the last fortnight.
- Three links are provided: 'Amend Todays Dream', 'View Dream Graph Visualisation', and 'Compare Data To Community'.

Desktop (several of the pages are comparable to the Mobile version, so only those that differ are displayed). A responsive counterpart for the data visualisation (dataviz.php) is not provided because responsive design with the D3 library.



Profile Page

[Home](#) [Log Out](#)

User name: Jay
[Account Settings](#)

Dream Data

Top Dream Topic: **losing control, forgetting, or disintegration**

Top Dream Person: Benjamin

Last Fortnights Dream Entries 

[Amend Todays Dream](#)

[View Dream Graph Visualisation](#)

[Compare Data To Community](#)

[Home](#) [Log Out](#)

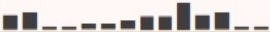
User name: Jay
[Account Settings](#)

Dream Data

Top Dream Topic: **losing control, forgetting, or disintegration**

Top Dream Person: Benjamin

[View Dream Graph Visualisation](#)

Last Fortnights Dream Entries 

[Compare Data To Community](#)

[Amend Todays Dream](#)

Show me who dreams about:

problem solving or accomplishm ▾

Submit

household (14%)**internet (7%)****necessity (7%)****sideproject (50%)****solo_leisure (21%)**

Show me

dates:

start date ▾

end date ▾

Submit

Show Filters

☐ Female ☐ Male

Age:

Select one... ▾

Relationship

Select one... ▾

Job

Art / music / writing ▾

Submit

3.6. Evaluation

Customer Testing was not undertaken in this phase of the product.

The user experience was tested with and without javascript, which illustrated that javascript is essential to its current functioning - something that could lead to interoperability consequences. See [Section 3.4 Testing \(above\)](#).

I modified my MySQL population scripts (using the autogeneration scripts see [Appendix Section 7.3](#)), filling the database with 10000 members and a corresponding ratio of dreams as was contained in the original script (29 per member)

4. Conclusions

4.1. Web Implementation vs App Realisation

For this iteration the project has been implemented using web technologies. This has benefits in that it exists as a an accessible web based app. The customer can use the website in much the same way as they would a mobile phone app.

However I believe that integrating this functionality with an alarm clock app, as has been the norm within several other apps (see [Section 2.2 Background & Research](#) earlier) would enhance the apps purpose. It is therefore a limit of this app that it doesn't integrate with a users alarm clock service - precious human recollection time is lost in this regard.

That noted I believe it was good to appreciate the complexity required in realising something in languages and technologies (i.e. web technologies) with which I am acquainted. The added complexity of navigating Android app development (a new technology/development environment in terms of the skills I possess) would have hobbled the realisation of this project.

4.2. Implementation Reflections

I have learned that much of the time I spent on hand rolling CRUD functionality could be saved through the utilisation of frameworks, such as Smarty, Cake or Yii (given my language choice of PHP). This is a lesson learned for subsequent development.

The project is a long distance away from being what I hoped for. Creating an engaging data analytics dashboard is clearly a huge undertaking in and of itself. It is very apparent that to make such a system work the project would need to be designed with a framework like jQuery (for the GUI elements of the website) from the very beginning.

That noted, there were some advantages of hand-rolling the CRUD functionality. My database makes use of many foreign key relations and reference tables. To properly implement this in a framework would merit a commitment to learning the best practices of a framework. This is a learning objective I have set myself, but it was a time commitment which was evaluated and considered incommensurate to timely delivery of the project. I do not doubt that learning a framework will be easier having hand rolled CRUD system.

5. Further Development

Pragmatically several areas require implementation prior to making this project live for consumer use.

5.1. Pragmatic Next Steps

5.1.1. Back Ups

back up mechanisms to preserve the data logged.

This will include regular back ups of the main MySQL Database, wherever it is hosted

5.1.2. Robust Security

Security is a requirement - subsequent builds will implement proper password hashing.

5.1.3. User Journey

The tests illustrated that there are several edge cases of the user journey which need to be caught and remedied (see [Section 3.4 testing](#))

5.1.4. Expanded Membership Profile Data

Currently registration caters for users who wish to join from Ireland with a county by county granularity. The rest of Europe is catered for in terms of capital cities. Subsequent iterations will take in USA, EMEA and Rest of World.

Pragmatics aside I have prioritised the subsequent stages of development requisite to implementing the project as originally proposed (see [Appendix Section 7.4](#))

5.2. Iterative Stages of Improvement

5.2.1. Phase 1 – Providing Network Diagram Visualisation

The most pressing next iteration of the project is the inclusion of dynamic network diagram visualisations. I call these indi-visuals.

I think that providing the users with an engaging network diagram visualisation provides the following benefits.

- Enforces an artificial period of data accumulation during which user would build up something sufficient to compare to the aggregate
- Supports harvesting of unique keywords that can be later taxonomised (for instance, by a more sophisticated association determination algorithm) into aforementioned metacategories - or indeed into different metacategorisations
- Permits the network visualisation in a more rewarding fashion. I believe the network visualisations are important because
- Keep the user busy accumulating their data

Constructing the dataset with network visualisations at the fore enables dovetailing with several exciting uses of this abstract data type, as exemplified by [Nodus Labs](#).

Their [Graph Database Structure Specification](#) [Nodus Labs, 2014] uses a totally different method of data storage, neo4J - implemented in javascript. As the main form of visualisation I want to provide to users is a Graph Abstract Data Type driven visualisation, it may prove prudent to consider the viability of this alternative method of data storage and retrieval.

5.2.2.Phase 2 – Improve Logging Experience and Incentive

With the visualisation & analytics section consolidated, the next step would be improving the user experience on the existing product (logging dreams via a browser). [see: [Plug-In & App extensibility of User Contributed Data](#) below]

5.2.3.Phase 3 – Trial Dream Communities

Allow members to form communities based on dream commonalities. Do this in a consent focused manner - such that people within the community could form pseudo-anonymous communities where they can share their dream data and content. Dream communities can form from commonalities in dreams, and allow users access to a similar form of aggregate data visualisation as it presently available to admin access in Requirement 4.

5.2.4.Phase 4 – Integrated Alarm Clock

A final step would be an integrated app, with alarm clock functionality. With the above components this app would have many distinguishing features from existing dream logging offerings.

5.3. Plug-In & App extensibility of User Contributed Data

The first three words you
see, will be yours in 2014



Imagine generating a similar graphic from the names, place, people and objects the user has entered, with a similar horoscope like appeal to the irrational in human nature. Trinkets and ephemera like this would increase the share-ability of content produced within the Networked Dream Platform

5.3.1. Facebook and other social media API integration

Provides indexing across multiple datasets

Automation of the users activity the day previously

Extends the analytics capability of the product

<https://developers.facebook.com/blog/post/2013/05/08/platform-updates--operation-developer-love>

5.3.2. Enhanced what were you doing before sleeping functionality

One of the great appeals of many of the current crops of life-logging apps is the extent to which they can automate the story of what you did on a given day through data analysing social media.

Some exemplary instances of this automated lifelogging include [Saga](#) and [Kennedy](#)

Failing such automation emails (such as 'this is my jam') or bookmarklets which expedite the passive logging of waking-day relevant information.

5.4. Other Technologies

5.4.1. Databases

The network diagram database as a model to pursue in the future

e.g. <http://www.neo4j.org/> - Graph Database

5.5. Further Research

5.5.1. Analytics, Demographics and Taxonomising the Human Psyche

One which interesting design consideration that arose during creating the project is how does one concisely categorise i) what dreams are about, ii) tally those themes with elements of the human psyche. This is an area for further fruitful research – e.g.

the Hall/Van de Castle System of Quantitative Dream Content Analysis

<http://www2.ucsc.edu/dreams/Coding/>

6. References

BOOKS

Barrett, D. The “Committee of Sleep”: A Study of Dream Incubation for Problem Solving. *Dreaming*, Vol. 3, No. 2, 1993

Dagfinn Aas (ed) *Time Use Studies: Dimensions and Applications*, Central Statistical Office of Finland 1986

Varol, O Menczer, F. : Connecting Dream Networks Across Cultures, *arXiv:1402.2297 [cs.SI]* <http://arxiv.org/abs/1402.2297> accessed 17/05/2014

Lemov, R. *The Database of Dreams: Social Science’s Forgotten Archive of How to Be Human (Ongoing Research)* Max Planck Institute for the History of Science
http://www.mpiwg-berlin.mpg.de/en/research/projects/DeptII_Lemov_Dreams

Nodus Labs - Cognitive Graph Database Structure Draft Proposal, 2014,
<http://noduslabs.com/cases/graph-database-structure-specification/> accessed 17/05/2014

Wiseman, R. *Night School: Wake Up to the Power of Sleep*

Wolfram S, The Personal Analytics of my Life
<http://blog.stephenwolfram.com/2012/03/the-personal-analytics-of-my-life/> accessed 17/05/2014

CODE

Creating a Database Driven Application With NetBeans,
<https://netbeans.org/kb/docs/php/wish-list-tutorial-main-page.html> accessed 17/05/2014

* <http://underscorejs.org> accessed 17/05/2014

* (c) 2009-2013 Jeremy Ashkenas, DocumentCloud and Investigative Reporters & Editors

* Underscore may be freely distributed under the MIT license.

WEBSITES

Apps

Kennedy <http://brendandawes.com/projects/kennedy> accessed 17/05/2014

Twitch <http://twitch.stanford.edu/> accessed 17/05/2014

DreamOn: <http://www.dreamonapp.com/> accessed 17/05/2014

Saga: <http://www.getsaga.com/> accessed 17/05/2014

Shadow: <http://www.discovershadow.com/> accessed 17/05/2014

Dreams

<http://www.dreammoods.com/dreambank/> accessed 17/05/2014

<http://dreamdoze.com/> accessed 17/05/2014

<http://www.sleeps.com/dictionary/dictionary.html> accessed 17/05/2014

<http://www.dream-analysis.com/interpretations.htm> accessed 17/05/2014

<http://dreamstudies.org/category/dreaming-technology> accessed 17/05/2014

Graph Abstract Data Type & Network Diagram Visualisations

Immersion: <https://immersion.media.mit.edu/demo> accessed 17/05/2014

neo4j Graph Database <http://www.neo4j.org/> accessed 17/05/2014

Nodus Labs <http://noduslabs.com> accessed 17/05/2014

Quantified Self & Personal Data

<http://your.flowingdata.com/> accessed 17/05/2014

<http://daytum.com/> accessed 17/05/2014

<https://www.chartmyself.com/> accessed 17/05/2014

<https://fluxtream.org/welcome> accessed 17/05/2014

Chris Dancy: <http://www.chrisdancy.com/> accessed 17/05/2014

<https://webflow.com/>

7. Appendix

Tables of Contents

User Manual & Technical Manual – make into header

Other Materials Used

Original Proposal

Wireframes

Deprecated Requirements

Project Management Plan

7.1. User Manual & Technical Manual

7.1.1. User Manual

Members can register via [join.php](#)

User names must be at least 5 chars long, and passwords must contain capitals and numbers

You must complete the second screen of registration before you can log your dream details.

How to Log Your Dreams...

Each member is only permitted to log one dream and its associated causes for a given day

Step One: LogDream.php

On the dreamlogging screen you have a number of entry form fields. The first is the most important, you log your dream via a metacategory. This helps produce the data visualisations, and also enables you to form private communities with like minded dreamers.

The four subsequent forms, 'Embodied Activity', 'People', 'Objects' and 'Places' are fields for you to tag the things that appear in your dream.

The last field, comments, is where you can extensively document your dream if you so wish. It can be hard to recall all of your dream initially, so try to tag the above fields as memory prompts!

Step Two WakingData.php

Have you ever wondered if there might be some connection between what you do before bedtime and what's running through your mind during dream time? You know, beyond the cheese nightmares?

Here's where you can make a note of what you were up to the night before. We've broken things down into two categories: free time and no free time. More often than not the latter is unavoidable, but when you do have time to unwind before bed it can make a difference to the quality of your sleep.

Don't see a category that fits?

Not to worry. We've taken a lot of time to find out what the most common broad categories of dreams and before-bedtime pursuits are. But we don't doubt we've missed some.

Just select 'something else' and fill out the suggestion box. Your dream will be tagged with that category so keep it succinct!

Visualisations.

Once you've logged a fortnight worth of dreams you'll be able to access the networked dreams visualisations.

For now you can see what the existing data says regarding what people do before they go to bed.

Those advanced options are off limits, until we roll out the communities functionality.

Also coming soon are the networked dream visualisations, where you can see the common occurrences of people, places and objects with the emotions you experience

while in the land of nod.

7.2. Technical Manual

Installation

A MySQL database is required to run this project

Step 1: Database Installation

Open MySQL > enter the following commands.

```
create database dreamdemo;  
use dreamdemo;  
source dreamDB.sql;  
populate script
```

OR

```
source installation.sql
```

To run this project a WAMP LAMP or MAMP local virtual server must be installed. Depending on your operating system the directory where you need to place this project source files will differ

Windows: c:\wamp\www

Linux: /var/www

Mac: /Applications/MAMP

unzip **project.zip** into the indicated folders above

You should see the following

CSS

js

images

Includes

nbproject

complete.php

dataviz.php

index.html

index.php

join.php

logdream.php

login.php

logout.php

profile.php

wakingdata.php

Step 2: Configuration

Includes/db.php contains settings that will need to be changed according to your local MySQL user configurations

```
private $user = "SQL_USER";  
private $pass = "SQL_PASSWORD";  
private $dbName = "dreamdemo"; // or beta prototype  
private $dbHost = "localhost";
```

AMEND:

SQL_USER

SQL_PASSWORD

For these parameters it is recommended that you add the username and password of a user profile for your PHP installation that has root access rights

Step 3: Launch

navigate to localhost/project to proceed through the project.

You can log in with username test, and password test.

Code Errata

Notes for Code

addUser.php – refactored to join.php

completeReg.php – refactored to complete.php

dreamEntry.php = refactored to logdream.php

userdataViz – dataviz.php

Notes on Databases

The provided dataset was used by stephen fortune to log his own dreams, as extension of his analog dream diary.

Stephen's ID was 2 – all this data has been removed from the database.

Accordingly the mmbrr table has no entry for 2, but continues from 0 – 1, and then 3 – 52

7.3. Other Materials Used

In order to test my database queries it was necessary to populate the database with pseudo entries.

I wished to have a depth of data to draw upon, so rather than populate completely randomly I scraped data from existing websites. That process is now detailed.

I utilised scripts URLHarvester.php, URLregex.php and dreamHarvester.php to parse the user logged dreams and categories of dreamdoze. This returned the HTML as text, which I ran through some regular expression (regex) text parsing scripts.

All of these scripts were ran as batch files from the command line, in an ad-hoc manner. See Scripts Read Me below.

The end result of mass scraping was 1470 URLs, however not all were comment regex compliant equates to 29 dreams for 50 people.

I hand classified the 1470 URLs, allowing this process to inform the 'meta-categories' I had tentatively devised thus far. I now possessed a CSV file of URLs and their 'categories' per my projects ontology.

The database was constructed in such a way that I kept a record of each dream URL which had been assigned to a given member. This enables me to, at a later point, do more in-depth and accomplished text mining of the URLs, filling out the tables dr_activity, dr_category, dr_locale and dr_people with data that might at least have some originary relation to the taxonomy I imposed on the raw data.

With an indexed CSV the next stage was to assign these dream_IDs to pseudo-users of the website. I again erred away from total randomness, instead using the names of people who had already appeared in my dreams. This list was complemented through performing a sociogram analysis of my friends on facebook. 50 members were desired, in order to allocate them approximately a month of the dreams contained in my dream_ID csv.

The member table auto-population tables saw two phases, which corresponded to my decisions to partition the registration process into two steps (explained elsewhere)

dream_21042014.sql – contains the 52 members, indicative of the combined work of
mmbrSQL_generate.php
dreamSQL_generate.php scripts

The names were from my social network. The ages I instead randomly assigned according to demographic breakdowns of age groups. I took Facebook as my inspiration.

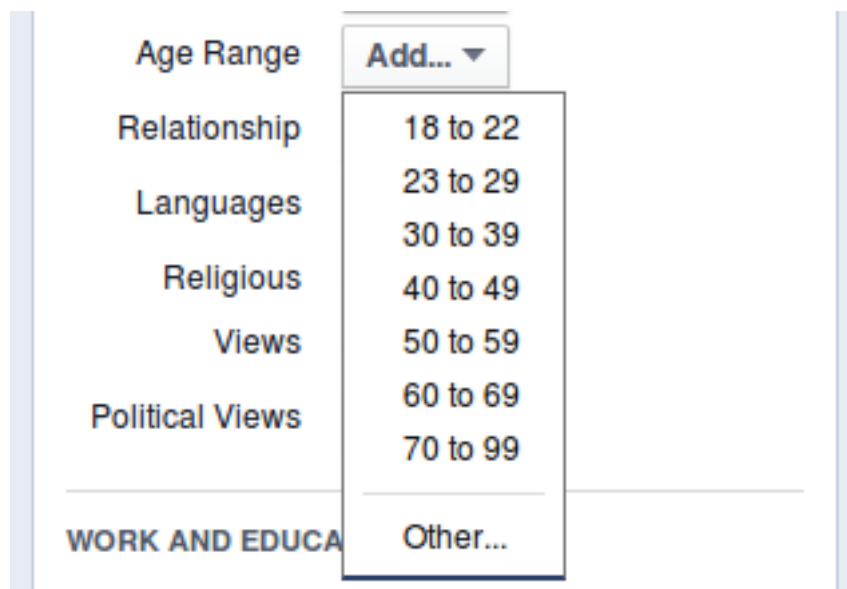


Fig 25 – Social Media Demographics, e.g. Facebook

Each user was randomly assigned a profession from the list of ().

Gender was split 50/50

Marriage was randomly assigned from the list of ()

Currently learning was randomly assigned.

When the mmbr_profile table was amended (reflecting a reorganisation of the Entity Relationship Diagram driving the project) a second stage of auto generation of entries was necessitated.

The original ages were used, because there was sufficient work in modifying the ages

into a YYYY-MM-DD format

Once member profile was generated, the final autogeneration was in the wakingdata table. This was a simple random assignment exercise.

Scripts Read Me

taggrab.class.php – script sourced from this website, enables use of CURL to scrape HTTP data

-- URLHarvester.php

Requires an array or a file (to be read as an array) of URLs

It will parse all URLs, and return a text file that contains every link that is hyperlinked to from a given URL in the above array.

-- URLregex.php.

Requires URLHarvester.php to have been executed

Reads URLs from a carriage return separated .txt file

Writes cleaned text to a new text file

-- dreamHarvester.php

Requires an array or a file (to be read as an array) of URLs that contain data formatted as dreams per the dreamdoze.com format.

Writes title of dream to a file

-- dreampara.php

Unutilised, but can be used to strip all descriptive words used in relation to a dream.

-- LogicSearch.php

Extension of above unutilised parser – designed to locate words ending with “-ing” and “-ed” as a means of quickly locating keywords relating to activity (and this ea_activity database table)

Compress.php

Utilised to strip recurring URLs from a text file

Important because I utilised a CRON job implementation of the data scraping scripts, designed to randomly choose URLs from an array, and to append rather than overwrite to the scraper file (over time this helped account for timeout requests on the part of CURL)

7.4. Original Proposal

Project Proposal

Networked Dreams

Stephen Fortune, 13119508, stephen.fortune@student.ncirl.ie

Degree Programme: Name HDSSD

Date 17/02/2014

1 Objectives

Develop a dream tracking app for mobile Android, combining an alarm clock and ability to tag feelings, friends, objects and places occurring in the users dream. The alarm clock awakes users and prompts them to tag their 'dream meta-data' along the quadrants of emotions, people present, objects and places/locales (see diagrams)

The users data permits the generation of a network diagram data visualisation that illustrates connections between emotions and places, people, and objects. An interactive section of the app lets user select an emotion tag, and then visualise the connections (in people, places and objects) accrued to that emotion over time spent in the land of nod.

The ambition would be to build this singular experience (intended to be compelling enough to prompt repeat visits and use) it into a self-tracking community. Users could build private communities with one another and compare if they share recurring themes in their dreams. Connections would be visualised using 'network diagram' visualisations. This is a deliberate echo of sociometric diagrams commonly deployed to visualise social network connections. The inherent appeal is that users can compare their dreams against other people and see if there is any commonality. As proof of concept a personal, singular experience will be designed. I will harvest the data from users of the app to a central server located database.

The alarm clock feature is intended to meet the immediacy of logging and data entry. This is a crucial design factor – dreams are most easily recalled on the moment of awakening. Dream diary apps suffer from the fact that a touchscreen keyboard is not as immediate as pen and paper. Tagging elements of your dream in a visually compelling manner along with auto suggested tags is this apps answer to that problem. A question asking unlock screen, (as inspired by Stanford's [Twitch data crowdsourcing](#) Android launcher), is a UX inspiration. Any dream journaling will be the last data entry point in this app, and is largely incidental to the apps core functions. However it is anticipated that the lack of such of function would deter users.

In order to glean meaningful trends from the user data it is desirable to have some form of 'waking' data to compare to the content of dreams. Ideally this process would be automated, following the lead of apps like [Kennedy](#). However such automation may be too complex for the purposes of this project.

2 Background

Sleep tracking and dream journaling apps are a niche section of the app market. A recent app [Shadow](#) generated a lot of interest on kickstarter for its mooted approach to dream journaling.

Dreams are an aspect of peoples lives that people are always curious to learn more about – this app will attempt to sate and stir that curiosity through the medium of data visualisation. Specifically it will surface relationships over time (and the strength of those relationships), information that network diagrams usefully explicate. potential for comparison to aggregate analytics that data analytics of the bulk user data affords

The aspiration is that the app will provide memory recall tools. It is a noted phenomena that logging your dreams aids you with subsequent recall of dreams. The data analytics will be a feasibility study of the scale required to create micro networks for users of the app

I have conducted research on sleep tracking apps that exist in the market. I have concluded that only one iOS offering [[DreamOn](#)] currently incorporates a friend tagging mechanism.

3 Technical Approach

Research,

Product research of existing sleep logging apps

All of the sleep tracking apps include a custom alarm clock feature, so I am following a predominant design paradigm in the market.

Research leads me to believe that design aesthetics would be a crucial factor of any such app. Ultimately the app will strive for a fluid user experience and an aesthetic reskin would be saved for future versions

I have not attempted data visualisation before – to familiarise myself I will get acquainted through GEPHI. I do not believe this can be implemented on android ergo the use of GEPHI will be purely a learning exercise for the author.

Requirements Capture

Requirements will be determined via Object Oriented design process.

literature review,

My literature review is concerned entirely with network visualisations. The literature on what dreams 'mean' is vast and ultimately inconclusive. The point of reference I am drawing upon is the 'threat rehearsal' thesis of why dreams happen. Threat Rehearsal Thesis posits that the dreaming brain is acting out 'fight or flight' scenarios, in order to practice for what to do if such a scenario happens during waking hours. The thesis makes the interesting observation that there are scenarios of survival common to all humans (due to our shared evolutionary history), but that these scenarios are cross pollinated with the scenarios which constitute contemporary day to day life. Hence the bizarre chase dreams transpiring against the backdrop of a call centre.

The aim of the app is to not laden the app with interpretative bias of what dreams mean, and instead provide a tool that captures data and presents the data back to them in as clean a data visualisation paradigm as possible.

4 Special resources required

Feasibility appraisal from experienced developers

Server for syncing user data to SQL database

Application Logic

Android SDK

Eclipse IDE

Data Visualisation software

API access

Frameworks

Design Patterns.

5 Project Plan

Gantt chart using GanttProject (cross platform open source Project Management Tool) with details on implementation steps and timelines

6 Technical Details

I intend to implement the App in Java as much as possible

7 Evaluation

I will capture an offline dream diary per the parameters of the app, to see how much my posited quadrant is a useful filter for dreams to be captured through.

The aim will be to have a working app by the beta data and then begin gathering data by my interaction with it. The aspiration here it to get it to a potentially usable state by a select group of friends. I will need to enlist good friends as a favour. This phase will be conducted in order that the data analytics facet of the project can be tested.

Stephen Fortune __19/02/2014

7.5. Original Proposal Wireframes

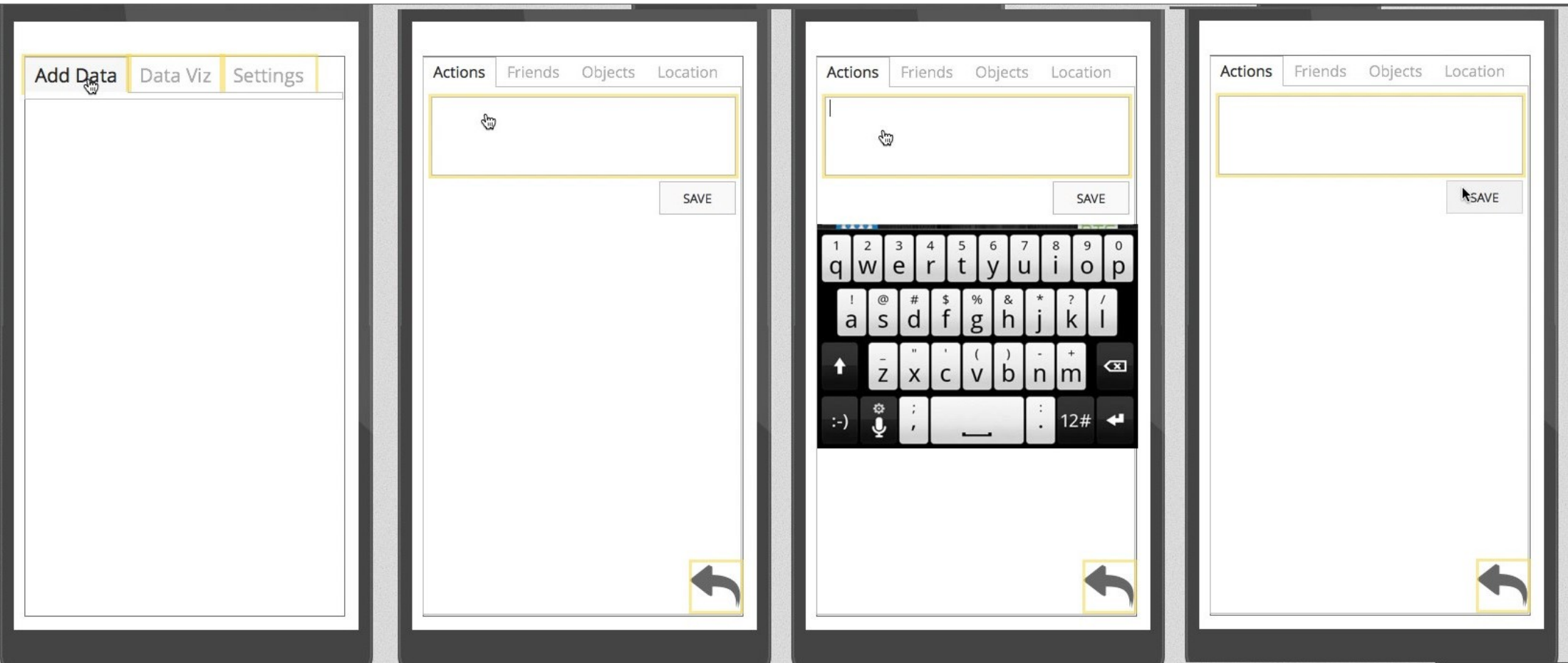


Fig 26 – Android App wireframes part 1

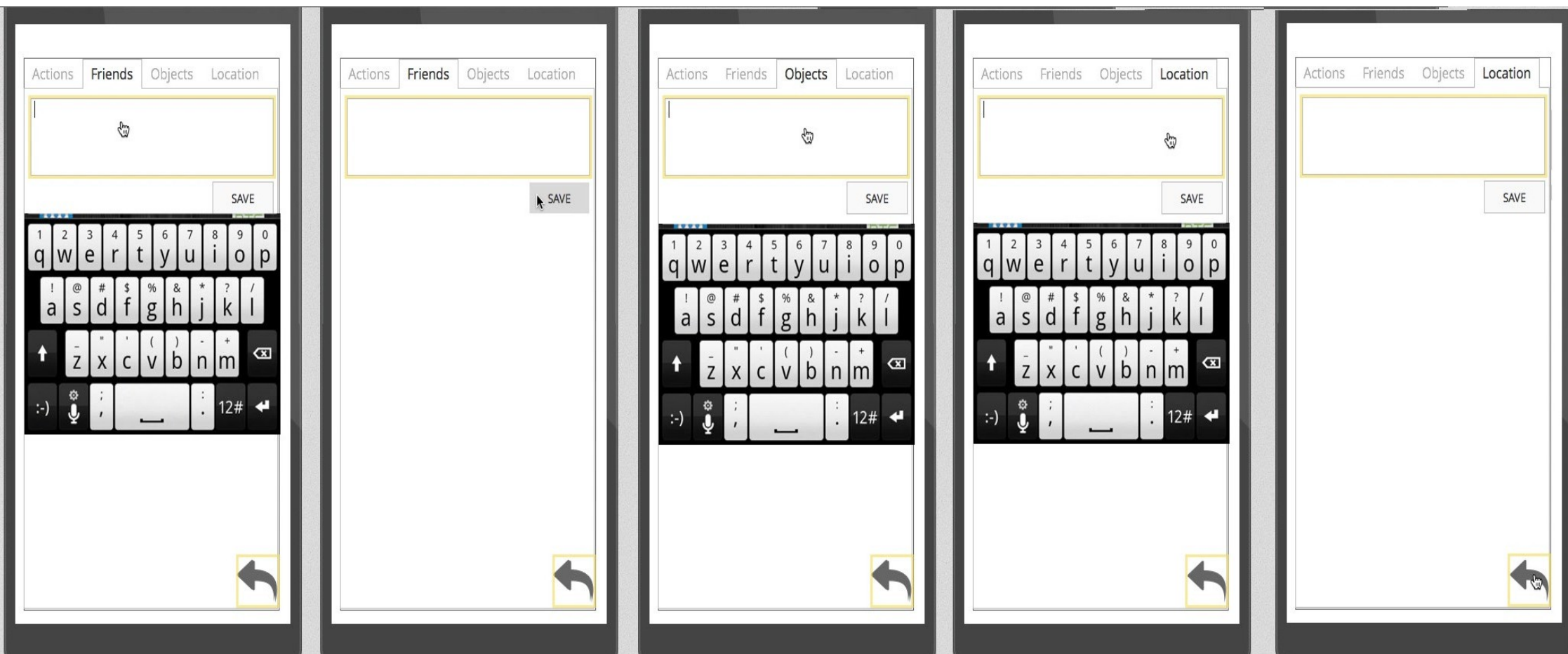


Fig 27 – Android App wireframes part 2

Fig 28 – Platform Agnostic Wireframes – sign up process

Networked Dreams

http://

One Two Three

Gender

Profession

Hobbies

Home Town

Next

This wireframe represents the first step of a three-step sign-up process. It features a browser window titled 'Networked Dreams' with a search bar containing 'http://'. A progress indicator at the top shows three tabs: 'One' (selected), 'Two', and 'Three'. The main content area contains four labeled text input fields: 'Gender', 'Profession', 'Hobbies', and 'Home Town'. A 'Next' button is positioned at the bottom center of the form.

Networked Dreams

http://

One Two Three

Name

email

confirm email

DOB

Next

This wireframe represents the second step of the sign-up process. It features a browser window titled 'Networked Dreams' with a search bar containing 'http://'. A progress indicator at the top shows three tabs: 'One', 'Two' (selected), and 'Three'. The main content area contains four labeled text input fields: 'Name', 'email', 'confirm email', and 'DOB'. The 'DOB' field is pre-filled with '/' and has a calendar icon button to its right. A 'Next' button is positioned at the bottom center of the form.

Fig 29 - Platform Agnostic Wireframes – completed



Fig 30 – Platform Agnostic Wireframes: data entry

The image displays two side-by-side wireframe screenshots of a web application titled "Log Dream". Both screenshots feature a browser-like header with navigation icons (back, forward, close, home) and a search bar containing "http://". In the top right corner of each page, there are links for "home" and "settings".

The left screenshot shows a data entry form with the following elements:

- A section labeled "Actions" with a text input field.
- A section labeled "Places" with a text input field.
- A section labeled "Objects" with a text input field.
- A section labeled "People" with a text input field.
- A "Log" button at the bottom center.

The right screenshot shows a data entry form with the following elements:

- A section labeled "Waking Data" containing a list of items with checkboxes:
 - ☒ Work
 - ☐ Work Out
 - ☒ Dinner
 - ☐ Drinks
 - ☐ <3 Life
 - ☒ Gadgets
 - ☐ Downtime
- "Specify?" and "That'll Do" buttons at the bottom center.

Fig 31 – Platform Agnostic Wireframes: data visualisation

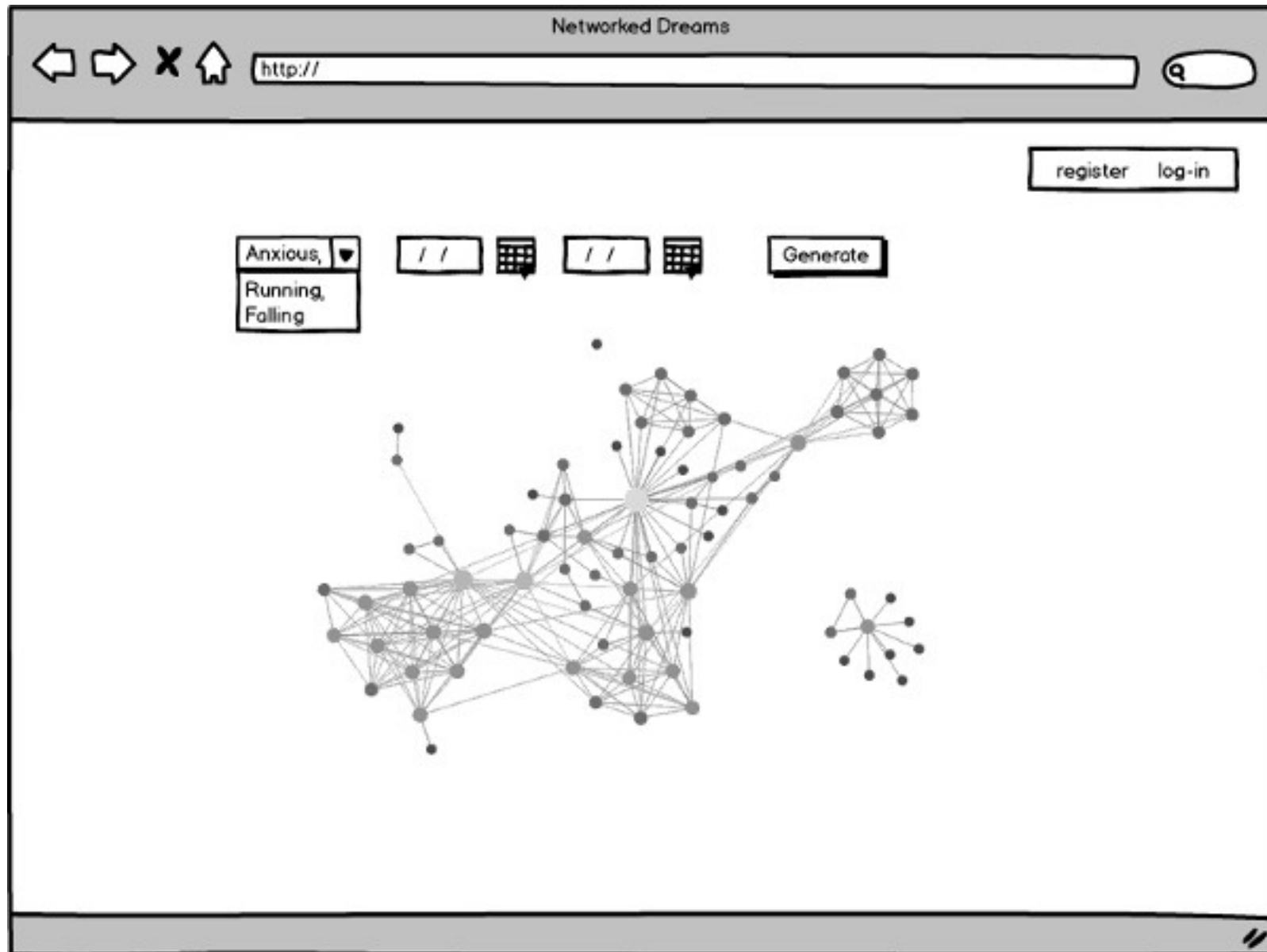
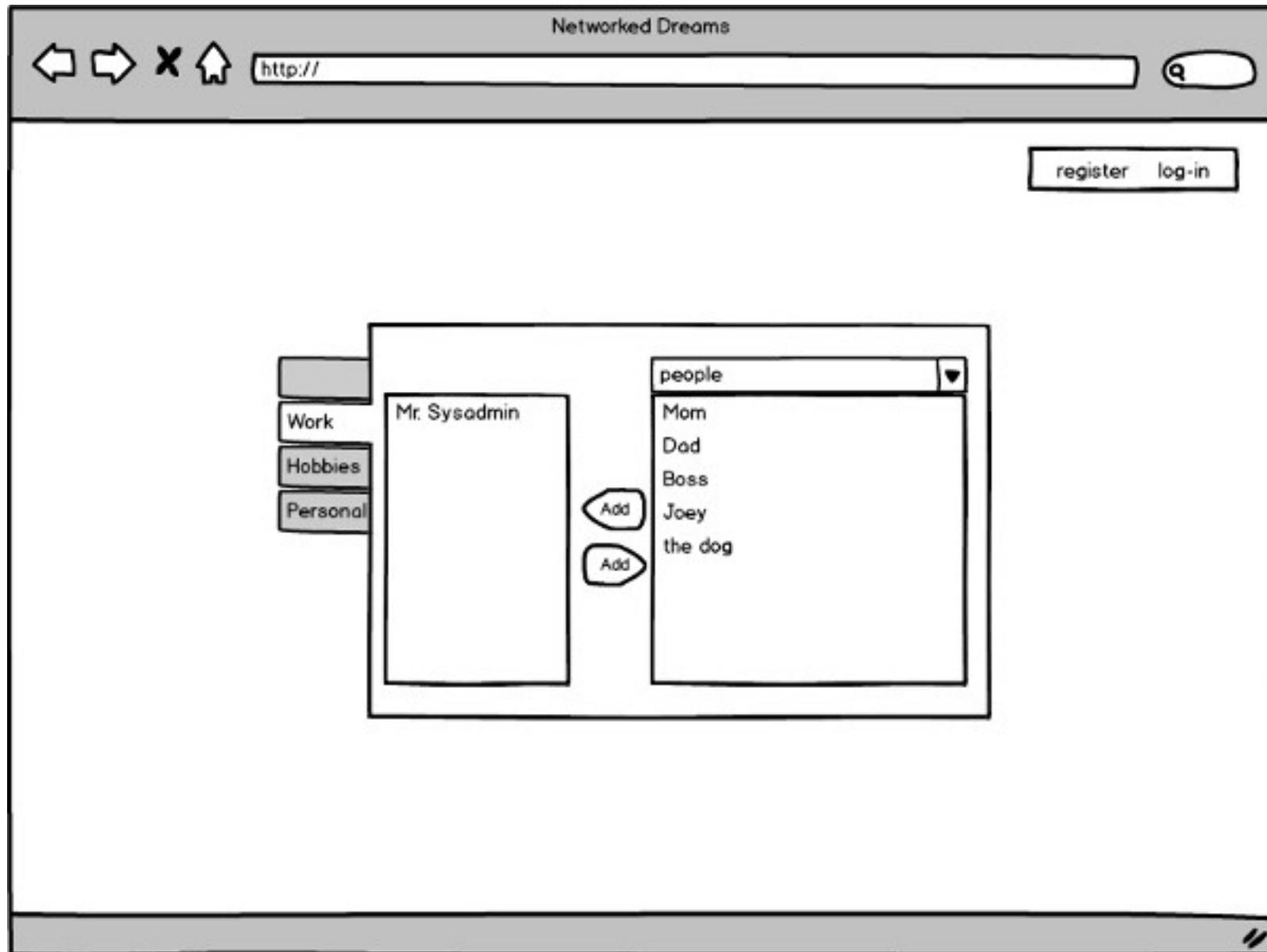


Fig 32 – Platform Agnostic Wireframes - meta categorise user tags



7.6. *Deprecated Requirement Specification*

Requirements Specification (RS)

Document Control

Revision History

Date	Version	Scope of Activity	Prepared	Reviewed	Approved
19/2/2014	1	Create	AB	X	X
10/3/2014	2	Update	CD		

Distribution List

Name	Title	Version
Stephen Fortune		

Related Documents

Title	Comments
Title of Use Case Model	
Title of Use Case Description	

Stephen Fortune 13119508
[10/03/2014]

Table of Contents

Requirements Specification (RS)	
Document Control	
Revision History	
Distribution List	
Related Documents	
1 Introduction	
1 Purpose	
2 Project Scope	
3 Definitions, Acronyms, and Abbreviations	
2 User Requirements Definition	
3 Requirements Specification	
4 Functional requirements	
2.1.1 Use Case Diagram	
2.1.2 Requirement 1 <User Registration: Calibrate friends, location, objects>	
2.1.2.1 Description & Priority	
2.1.2.2 Use Case	
2.1.3 Requirement 2 <Enter Dream Data>	
2.1.3.1 Description & Priority	
2.1.3.2 Use Case	
2.1.4 Requirement 3 <Waking Data Survey>	
2.1.4.1 Description & Priority	
2.1.4.2 Use Case	
2.1.5 Requirement 4 <Generate Dream Data Viz>	
2.1.5.1 Description & Priority	
2.1.5.2 Use Case	
2.1.6 Requirement 5 <Configure Meta Categories>	
2.1.6.1 Description & Priority	

2.1.6.2 Use Case.....	
2.1.6 Requirement 6 <Compare Dream Data>.....	
2.1.6.1 Description & Priority.....	
2.1.6.2 Use Case.....	
2.1.6 Requirement 7 <Calibrate Friends, Objects, Locations databases>.....	
2.1.6.1 Description & Priority.....	
2.1.6.2 Use Case.....	
2.1.7 Requirement 8 <Alarm Clock>.....	
2.1.7.1 Description & Priority.....	
2.1.7.2 Use Case.....	
2.1.6 Requirement 9 <Graphical Dream Data Entry>.....	
2.1.6.1 Description & Priority.....	
2.1.6.2 Use Case.....	
5 Non-Functional Requirements.....	
2.1.8 Performance/Response time requirement.....	
2.1.9 Availability requirement.....	
2.1.10 Recover requirement.....	
2.1.11 Robustness requirement.....	
2.1.12 Security requirement.....	
2.1.13 Reliability requirement.....	
2.1.14 Maintainability requirement.....	
2.1.15 Portability requirement.....	
2.1.16 Extendibility requirement.....	
2.1.17 Reusability requirement.....	
2.1.18 Resource utilization requirement.....	
6 Interface requirements.....	
6.1 GUI.....	
6.2 Application Programming Interfaces (API).....	
7 System Architecture.....	
8 System Evolution.....	

1 Introduction

1 Purpose

This document sets out the requirements for the development of the 'Networked Dreams' application and service

The intended customers are anyone who is curious to gain insight into their dreams

2 Project Scope

The scope of the project is to develop a web based application where users can store their data, and utilise the service/application/platform to gain insight into their dreams via data analytics and visualisation.

The system shall have means for data entry, facilities for data visualisation, and user management services

The ranking of the requirements has been determined by a shift from an initial 'mobile application first' development imperative, to creating a web native application which could be scaled to mobile through future iterations/developments

To establish requirements Stephen Fortune logged his own dreams for the period of 4-6 weeks.

3

3 Definitions Acronyms & Abbreviations

Meta Categories

Dream Data parameters

4 User Requirements Definition: Functional Requirements

1 Functional requirements

1.1.1 Requirement 5 <Configure Meta Categories>

1.1.2 Description & Priority

This shall permit the user to taxonomise their data. This increases the utility of visualisations. Users may associate people, places and object with the parameters of 'work' 'leisure' and 'exertion'. These categories are meta categories, which let the user generalise the occurrence of specific actions emotions, people and places into

categories which can (a) inform the construction of time series or other temporal visualisation (b) be compared back to an aggregate dataset of the service.

1.1.3 Use Case

Scope

The scope of this use case is to categorise a users dream content into metacategories used in data analytics of entire user dataset.

Description

This use case provides the parameters through which a user can engage with the aggregate data analytics function of the service.

Flow Description

Precondition

At least N weeks of user data must be completed

Activation

This use case is accessed via the settings screen.

If user has not activated this section then they will be prompted to do so before accessing aggregate comparison

Main flow

1. User selects settings,
2. A side bar menu of metacategories exists alongside a tab interface that includes every word that has been logged beneath "Friends | Object | Location | Action" fields
3. Selecting a word occurrence associates that word occurrence with a metacategory
4. user completes the above process iteratively until satisfied
5. User saves
6. system updates metacategories database

Sub Flows

Alternate flow

Exceptional flow

Termination

The user exits this section of the app

Post condition

Database containing the users data is updated

1.1.4 Requirement 7 <Calibrate friends, location, objects databases>

1.1.4.1 Description & Priority

The window for recalling the most about what one has dreamt about is quite a narrow time period. The time lost in logging in, and manually entering data points by typing amounts to two additional screens impeding immediate data entry. A graphical entry system would improve user experience. Calibration of said entry system on a user by user basis is necessary. This use case describes this requisite calibration.

1.1.4.2 Use Case

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

Scope

The scope of this use case is to itemise data in the users life to streamline data entry

Description

This use case describes the process for setting up user data into a menu format amenable to a graphical representation

Flow Description

Precondition

User must be registered on service

User must have been using the service for a certain period of time

Activation

When the user logs in

When the user accesses the bookmarklet

Main flow

1. The user selects a tab or link to friends, objects, locations, actions
2. The user completes each of the above
3. The user saves their entry
4. Entry is committed to system with appropriate time stamp

Sub Flows

1.1. Friends SubFlow

1.1.1 User has option to get friends from facebook (or from google contacts).

1.1.2 Request is sent from app to facebook

1.1.3 User is prompted to authenticate the app by facebook

1.1.4 Upon authentication the app gets [Name, Profile Thumbnail] from facebook

1.1.5 It populates a menu (3 thumbnails wide) with all the friends, ordered alphabetically.

1.1.6 All thumbnails selected by default. User is prompted to deselect those they consider unsuitable to their persons list.

1.1.7 User is then prompted to add people who may not be on facebook

GOTO 1.4 Manual Thumbnail Creation Subflow

1.2 Objects SubFlow

1.2.1 Objects subflow is the only one that can pregenerate and come bundled with the app

1.2.2 User is presented with a grid presentation of objects

1.2.3 User presses to highlight each object they have in their life

1.3 Locations SubFlow

1.3.1 Locations subflow will come prebundled with generic icons to represent places (like those on Google Maps?)

1.4 Manual Thumbnail Creation Subflow

1.4.1 They are directed to a screen where they enter a name for “Friends | Object | Location “

1.4.2 User selects a photo from their gallery to append as a thumbnail (autonavigate to gallery part of phone – from there should be able to access Instagram and other folder for images and such!)

1.4.3 Thumbnail and name created as button, added to the “Friends | Object | Location “ database and menu Class

Alternate flow

A3.2 Access calibration via settings screen

A3.2.1 User may select from a menu containing Friends, Objects and Locations.

A3.2.2 Dependant on above selection sub flows 3.1.1, 3.1.2, or 3.1.3 may be triggered

Termination

Post condition

Database containing the users data is readied

1.1.5 Requirement 8 <Alarm Clock>

1.1.5.1 Description & Priority

The user sets an alarm clock of their choosing for the following morning.

1.1.5.2 Use Case

Scope

The scope of this use case is to set an alarm, which facilitates data entry upon awakening

Description

This use case describes the process of setting an alarm, the use case that triggers the data entry..

Flow Description

Precondition

User has opened the app.

Activation

Selection of an alarm clock time, or create new alarm option.

Main flow

User selects preexisting alarm

1. User creates new alarm
2. User modifies alarm time parameters (HH:MM, repeating)
3. User modifies snooze and wake up enhancers
4. User selects save or end to indicate that they wish this alarm to be set for X hours subsequent.

Alternate flow

A1. User creates new alarm: User stipulates hour, minutes

A1.1 User selects preexisting alarm

A2. User deletes alarm or otherwise modifies alarms not directly related to ones due to trigger in next 24 hours.

A3.

A3.1 User accesses settings to change the sound of their alarm clock, volume and other parameters.

A3.2 User can modify the volume of their alarm

A3.3 User can select sounds from the library of sounds available to phone

Exceptional flow

E1. An exceptional flow is that there is no alarm set for the next day. The system needs to be able to recognise this and feed it back to the user.

E2. An exceptional flow is that more than one alarm is set to trigger within 24 hours. This could create junk entries in the user database. A solution may be that once a dream log entry is acquired, then no subsequent screen lock overrides by app can occur.

E3 A user may decide to change the sounds for the alarm

Termination

Post condition

Another alarm clock is saved to the catalogue (sic) of alarm clock times available to the user.

The system is readied to expect another instance of Enter Dream Data use case.

The system is in a wait state, counting down until stipulated alarm clock time at which point another use case can be triggered.

1.1.6 Requirement 9 <Graphical Dream Data Entry>

1.1.6.1 Description & Priority

The data added in use case 7 provides a dataset to graphically represent the users friends, objects, locations and common emotion-actions. This allows the user to graphically select what occurred in their dream and can serve as a memory aid to things which may have occurred in their dream

1.1.6.2 Use Case

Scope

The scope of this use case is for the app to gather data from the user about their

dream via linear progression through a series of interfaces

Description

Combined with the alarm functionality this use provides the user with a graphical interface upon awakening. This case enables the user to quickly enter dream data

Use Case Diagram

Flow Description

The use case described is a loop or iterative (i.e. each screen concludes, returns to home, asks for more data, until finished)

Precondition

Calibrate friends, location, objects databases use case must be completed

Activation

When the user accesses the website / When the user logs in to the website

Use case will continue to iterate until the user has checked DONE on each screen of the quarter divided circle.

Main flow

1. User selects a quadrant, proceeds to quadrant specific data entry screen (see sub flows below)
2. User selects/ticks done,
3. User returns to main screen
- 4.Repeat

Sub Flows

1.2 Friends SubFlow

1.2.1 People in dream subflow

A-Z list only – later an order by facebook graph option.

1.2.2 App presents an A-Z ordered list of people (represented by thumbnails and name)

1.2.3 User scrolls through list and taps the image of each person who was in their dream

1.2.4 App greys outs / displays a tick over each user when selected

1.2.5 User selects done button to complete tag logging

1.3. Objects in dream subflow

1.3.1 App presents an input bar at top of screen. User may enter items by keyboard, with commas separating items

1.3.2 User may select objects from an A-Z thumbnail list

1.3.3 App presents an A-Z ordered list of objects (represented by thumbnails and name)

1.3.4 User scrolls through list and taps the image of each object/item who was in their dream

1.3.5 App greys outs / displays a tick over each user when selected

1.3.6 User selects done button to complete tag logging

1.4. Places in dream subflow

1.4.1 App presents an A-Z ordered list of people (represented by thumbnails and name)

1.4.2 User scrolls through list and taps the image of each person who was in their dream

1.4.3 App greys outs / displays a tick over each user when selected

1.4.4 User selects done button to complete tag logging

1.5 Emotions/actions in dream subflow

1.5.1 App presents an input bar at top of screen

1.5.2 User enters emotions and actions from dream and uses commas to separate their entries

1.5.3 User selects done button to complete tag logging

Alternate flow

A1 At any given quadrant user could select done without selecting anything (in order to quickly get a longer snooze). This behaviour will be accommodated by asking the user “are you sure you don't remember anything”?

A1.1 If yes then snooze reverts to regular interval

A1.2 If user still says yes an alternate case will involve the screen being presented on the eventual disablement of the alarm in question

Exceptional flow

Termination

The flow terminates once all data entered

Post condition

Today's alarm disabled for 2 minutes

Today's alarm disabled for remainder of day (when all data gathered)

Previous night's dream data added to database

2 Non-Functional Requirements

1.1.7 Performance/Response time requirement

For aggregate data comparisons, speed of recall will be important

Access transparency will be secured via use of database permission tables

1.1.8 Availability requirement

Internet access is a prerequisite to sync dream data to server and to access networked visualisations.

For future phone iterations certain server resources will need to be made available to local app instantiations

1.1.9 Recover requirement

As a learning goal it would be good to learn how to have best practices in ensuring

database recovery.

1.1.10 Robustness requirement

Fault tolerance is not envisioned as being essential to this build of the app.

The front end for the system must function in any internet browser

1.1.11 Security requirement

In either case of users entering personal data or users opting out of statistical monitoring (aggregate comparison analytics) there has to be very good security

1.1.12 Reliability requirement

The service must always be accessible whenever a user wishes to add data to their dream log

1.1.13 Maintainability requirement

Subsequent builds maintainability will be contingent on server size and load: for initial build none.

1.1.14 Portability requirement

Future iterations user experience is 100% contingent on an intuitive portable experience

1.1.15 Extendibility requirement

To meet the requirement for a phone app being a subsequent iteration of the system I will need to require comprehensive documentation of how a users account is accessed.

APIs (to social services) may become an important part of the platform, ergo documented app API access will be important.

Neither of the above features are intended for this build, but they will be borne in mind when constructing the build

1.1.16 Reusability requirement

The app will be built to work web native, but with a consideration of how code produced during this iteration could be reused for future platform variations of the idea.

The server side database is envisioned as a component which can be reused for future iterations

1.1.17 Resource utilization requirement

System shall be built to prototype on localhost, with an online server earmarked as

a potential future resource.

6. Interface requirements

8. System Evolution

A bookmarklet web extension that allows people to log their dreams with one click from their browser.

Allow members to form communities based on dream commonalities. Do this in a consent focused manner – such that people within the community could form pseudo anonymous communities where they can share their dream data and content. This pseudoanonymity has informed the API thrust when pulling data from Facebook and Google.

Incorporating an alarm clock based mobile app to make the experience better suited to capturing dream data

A further scope for the app to evolve would be through the integration of more APIs into the system. This API integration would aid the automation of aspects of the user experience which may impede the crucial period during which dream recall is most potent

Use case 7: Calibrate friends, location, objects databases

At present the user must input data in two areas where inferred feedback (such as is possible from aggregating data from APIs) would be beneficial.

The Objects data list. A scripted means of scraping data from the users Amazon wishlists, Pinterest account, or indeed a smart selection of what brands the user has engaged with on facebook. This scraped data could be autopopulate the Objects List with suggestions relevant to that user.

A further fun addition could be made for the burgeoning internet of things – linking users to the internet activity of items which appear in their dreams.

The second area is the “Waking Data” Logging portion of the experience.

There exist many clever (iOS) based apps which analyse a users daily activity

conducted online (their Google calendar, check ins, tweets and pictures posted on Instagram) to auto generate a journal entry for their days activity. If the same algorithmic automation could be plugged into this app the users could have an automatically generated journal to compare their dream data to - “I guess it makes sense I dreamt of pirates given what I watched on Netflix last night...”

Another area that would benefit from automation is the metacategories assignment section. This would be possible once a certain threshold of active users on the network have participated in that feature. This would both offer a feature to the end user, and also improve the data analytics back end by providing a heuristic taxonomy from which analysis can be derived

7.7. Project Management Plan

(see below for graphics)

