Durham University
School of Engineering and Computing Sciences

Design Document

2011-12

Project Name: The Durham Cultural Hub

Team Number 5

Team Members

Antanas Brazenas

Lydia Felton

Jonathan Gaskell

James Kitson

Ahmed Saghir

Document Information

Project Name:	The Durham Cultural Hub		
Prepared By:	Antanas Brazenas, Lydia Felton, Jonathan Gaskell, James Kitson, Ahmed Saghir	**Preparation Date:**	**29/02/2012**
Email / Phone:	l.j.felton@dur.ac.uk		
Document Version No:		**Document Version Date:**	**29/02/2012**

Version History

Ver. No.	Ver. Date	Revised By	Description
0.1	20/02/2012	Jonathan Gaskell	Introduction completed
0.2	22/02/2012	Lydia Felton	User interfaces completed
0.3	25/02/2012	Ahmed Saghir	Process descriptions completed
0.4	26/02/2012	Jonathan Gaskell	Element descriptions completed
0.5	26/02/2012	Lydia Felton	Changes to requirements completed
1.0	26/02/2012	Lydia Felton	Design document created
1.1	26/02/2012	Lydia Felton	Definition of terms added to
1.2	27/02/2012	Lydia Felton	User interfaces edited
1.3	27/02/2012	Jonathan Gaskell	Element descriptions edited
1.4	28/02/2012	Jonathan Gaskell	Element descriptions edited
1.5	29/02/2012	Jamie Kitson	Review of all sections, website and web service sections edited
1.6	29/02/2012	Jonathan Gaskell	Edited element descriptions, references and definitions
1.7	29/02/2012	Lydia Felton	Edited Introduction
1.8	29/02/2012	Jonathan Gaskell	Reviewed and edited process descriptions, edited introduction

Changes to Requirements

Requirement Number	Change	Reason	Consequence
FR001 – Add Categories	Rewording: The administrator is called a category manager rather than an administrator.	Our client doesn't want to have a central administrator due to funding reasons, and the wording was unclear.	Functionality is unaltered, just that the wording is clearer for the programming team to be able to implement the requirement.
FR006 & FR022 – Change account settings	Rewording of the remarks row: The top level categories the user can choose from are: Sport, Arts, College, Lectures, Cultural Events The user can set a preference for accessibility information.	To ensure the programmer knows what categories should be implemented. Our client asked for the ability to search for events based on accessibility.	Explicitly listing the categories will make it clearer for the programming team to implement the requirement. The user will be able to search events based on accessibility .
FR015 – Change skin depending on category	Remove requirement.	We thought changing the skin would reduce the website consistency. Also, if the category manager created a new category, they would have to create a new skin, which would be quite time consuming.	Functionality is unaltered, the skin just no-longer changes.
NFR005 & NFR010 – Easy Navigation	Priority change from low to high.	Our stakeholders cover a wide range of abilities, therefore the app and web app must be easy to navigate.	Improved functionality.
FR012 and FR029 – Notify me about this event	Remove requirement	To client didn't ask for this and the user is able to export events to their calendars (FR014 & FR031) which can notify them if them wish.	User will no longer be notified about events by the Web app or BlackBerry app. FR026 & FR009 – 'a link to get notified' will no longer be displayed
FR032 - Register to attend an event	The BlackBerry app will no longer check for time conflicts and limits on the number of attendees before accepting a user's request to	Not everybody who clicks 'attending' will be able to actually attend the event. Therefore, if there is a limit, others who are interested in the event won't be able to get notifications about the event or be able to add the event to their calendar.	The number of people 'attending' the event may exceed the events capacity, but this shouldn't be an issue as clicking 'attending' on the app doesn't mean the user is definitely attending the event.
FR003 – Delete an event	An event manager can also delete an event.	To allow the event manager to have more control over their event.	Functionality is unaltered for users, but it will allow the event manager to delete events.
FR016 & FR035– Display history of an event	Renamed requirement to 'Past Events'. A past events button will be included on both the Web App and BlackBerry App. When the user clicks the button, they can view the same information the 'Display History of an event' link could.	Instead of having an event history on an event, we thought it would be more logical to have a separate page to past events	FR010 and FR026 – event history will no longer be displayed.

Table of Contents

Contents

1. Introduction

1.1 Purpose

The purpose of this design document is to demonstrate the functionality of our system and detail how the system will fulfill the requirements that have been set. This document will provide a high-level view of the system as well as the lower-level technical details of how the system will work. It will provide all the information that is required for the implementation to be carried out, taking into account implementation issues and avoiding any ambiguity.
The BlackBerry app will be programmed using an object-oriented approach because Java is the development language for the BlackBerry operating system. Therefore, this document will use object-oriented design to propose a collection of objects that can interact with each other; together these objects will form the complete software system.

System Benefits

The concept is to improve communication of events throughout the University population and the general public. The system will collate information about the various events that take place in Durham and present information in an easily accessible manner. The BlackBerry app will enable users to browse and register for events on the go and provide directions to the event. It is hoped that the system will improve not only awareness but attendance and allow constructive development of further events through feedback.

Objectives

The objective of the system is to create a system that can replace the 'What's On' webpage on The University of Durham's website. The system should be: easier to use, more accessible and more popular with its stakeholders. The system will comprise of both a web app and BlackBerry app, and improve on the functionality of the existing system.

Goals

The system will be more accessible by creating the BlackBerry app meaning that the system can be accessed on the move. The system will add events to a user's calendar and provide an email notification service. The system needs to have adequate scalability in order to cope with the expected increase in popularity. To interact with software, an intuitive GUI must be developed in order to reduce the need for training. To communicate events effectively to users, the system should display a personalised list of events to each user showing the events that will interest them most based on their account preferences. The system should also allow the submission and display of user feedback for events so that users can have a better understanding of an event, and the event organisers can improve their event based on feedback.

Design Strategy

During the design process we will follow the waterfall model and also incorporate agile methods. Due to the nature of the SEG project, the implementation and design have to occur simultaneously, hence the decision to use agile methods. We will use ideas from the V model, reviewing sections after the formation of other sections; for example, the implementation process may invoke changes to requirements, and subsequently design.

1.2 Scope

The scope of the system is based around the requirements that were set in the requirements document. This includes any changes to the requirements that arise as a result of further analysis, feedback from the client and the design of the system. The completed system should have all of the functionality that is listed below, however the functionalities have been given different priorities if, for some unexpected circumstance, the team become unable to deliver all of the functionalities.

Essential

The system will:

- Enable users to create an account
- Enable users to log-in
- Allow users to change their account settings (category preferences and password)
- Enable events and event categories to be added
- Allow the removal of an event
- Display a list of selectable events according to a user's preferences
- Display events on a calendar
- Display individual events in detail
- Enable users to find the event on a map (get directions on BlackBerry app)
- Enable users to register to attend an event
- Enable users to cancel their registration to an event

Desirable

The system will:

- Enable users to delete their account
- Display past versions of an event
- Enable users to provide feedback for an event and display it on the event page
- Allow the moderation of feedback for an event

1.3 Definitions, Acronyms and Abbreviations

CSS: Cascading Style Sheets - a language used for maintaining consistent styles across a website Facebook: social networking service and website

GUI: Graphical User Interface, this is a graphical representation of what state the system is in and what actions the user can carry out to interact with this state.

HTML: HyperText Markup Language - a markup language for display rich content on the web.

Lumzy: A mockup and prototype creation tool for websites and applications.

MySQL: A relational database management system that allows SQL queries to be performed on the data contained within.

Paint.NET: free graphics editor program

PHP: HyperText Preporcessot - a server side scripting language.

REST based interface: Representational State Transfer - an online interface for communicating with the database via http requests

V-Model: a software development process which demonstrates each phase of the development life cycle and its associated phase of testing

Waterfall Model: a sequential software development process, in which each phase is completed before moving onto the next

XML: eXtensible Markup Language is a file format which is useful for storing structured information in a simple to read form.

.

1.4 References

Pressman, R.,2005. *Software Engineering: A Practitioner's Approach*. 6th Ed. New York: McGraw-Hill

Nielsen, J., 2000, *Designing Web Usability*. 1st Ed.Indianapolis: New Riders Publishing.

Budgen, D., 2003, *Software Design*, 2nd Ed. Wokingham: Addison Wesley

Goma, H., 2011, *Software Modelling and Design*, 1st Ed. Cambridge: Cambridge University Press

1.5 Overview

This overview provides an outline of the major sections of the document and what is contained within each of the sections

Section 2.1 - contains a description of what our system does through the graphical user interface and the development of the GUI design.

Section 2.2 - contains simple scenarios for certain tasks that our system will perform when complete. It covers the processes that take place in the system and describes them through use case diagrams.

Section 3.1 – looks at the architectural patterns and styles used when developing the system and the major design decisions taken. There is also a high level overview of the system, highlighting the responsibilities of each component.

Section 3.2 - contains class and architectural design diagrams to show the intermodule dependencies and the static structure of the overall system.

Section 3.3 - shows the interprocess dependencies of the system and its dynamic behaviour through the use of sequence diagrams. The sequence diagrams highlight the communication between the different objects in the system. Activity diagrams have also been used to show the behaviour of major elements and the responsibilities of each component.

2. Descriptions

2.1 User Interfaces

The following section details how the Graphical User Interface (GUI) will be produced for both the Web App and the BlackBerry App and provides an overview of its components and features.

When designing the GUI our goal is to make both the Web App and BlackBerry App easy to use (NFR005, NFR010) and also to make them aesthetically pleasing in order to appeal to the stakeholders. Every user interface, according to Pressman (2005) should be:

1. Easy to use
2. Easy to learn
3. Easy to navigate
4. Intuitive
5. Consistent
6. Efficient
7. Error-free
8. Functional

Therefore, whilst designing our Web App and BlackBerry App, we will keep these factors at the forefront of every decision we make in order to "provide the end-user with a satisfying and rewarding experience" (Pressman, 2005).

A survey will also be conducted on 100 potential stakeholders (25 students, 25 members of staff, 25 members of the general public and 25 potential tourists) to find out what the stakeholders would like from the User Interface. It is important that we keep the stakeholders involved in the design process as they will eventually be the end-users of the Apps and therefore we need to satisfy their wants and needs.

The following subsection details the initial user interface design phase and how the designs develop based on functional requirements, user feedback and group opinions.

BlackBerry GUI Design Phase

Initially, GUI designs were produced on Lumzy for the BlackBerry App to plan the design for each screen. This enabled us to decide how the app would function and how the user navigates the app. We kept in mind that the app should be easy to navigate (NFR010) and therefore decided that having a toolbar down the left-hand-side of the app with all of the options would be the easiest way for the user to navigate the app. This is because other apps we've looked at on the market,, have a toolbar on the left-hand-side, so in using this layout as well, the app should be 'easy to learn' and 'easy to navigate' as it will be an intuitive layout for BlackBerry users. Figure 1 shows the original general layout GUI designs for each page of the BlackBerry App. After discussing the designs, we came to the conclusion that the 'Home' button should be placed on the top of the screen to the right

of 'Logout' as it seemed a more logical place for the button to be and therefore making it easier for the user to navigate. Also, to improve usability, we decided to include dropdown menus (see [1] on figure 1) to reduce the amount of typing the user has to do, to make the app more efficient, and also to reduce the number of human errors that can occur with typing.

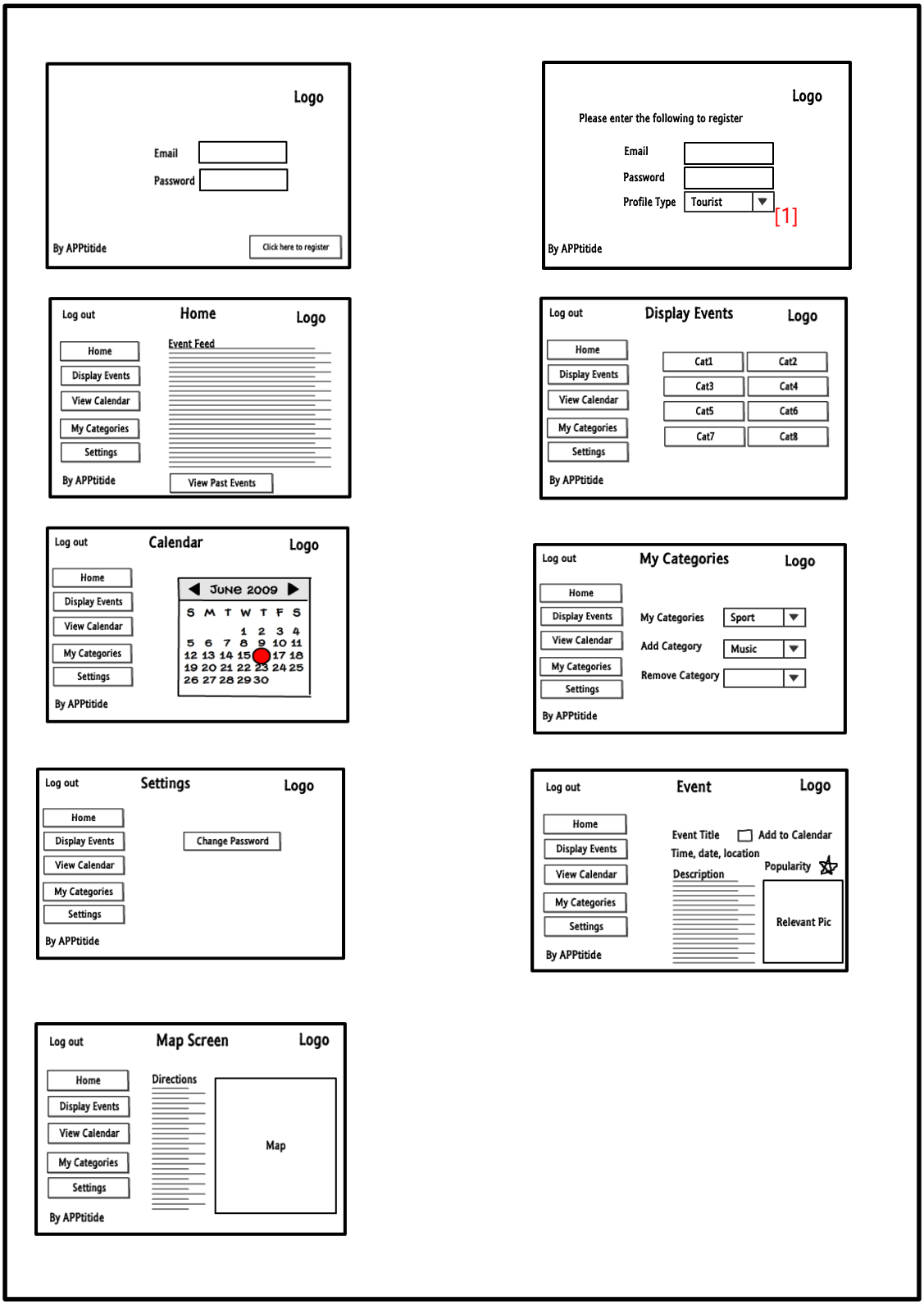


Figure 2.1.1: Initial GUI design

The next step of the GUI design process involved aesthetic design. Aesthetic design involves making decisions on the: “colour schemes, geometric layout, text size, font and placement” (Pressman, 2005). The app’s stakeholders span a large number of people, with different tastes, therefore, the app must not have an extreme design, but should try to appeal to all stakeholders as much as it can. The survey results indicated that the majority of stakeholders would like a palatinate purple as the dominating colour of the App, so we decided to use this as the predominant colour of the GUI. The decision to use a dark colour for the background, led us to come to the conclusion that the text should be a contrasting colour, such as white, to ensure the users can see the text.

Figure 2.1.2: Initial Log In Screen

Figure 2.1.3: My Categories Screen

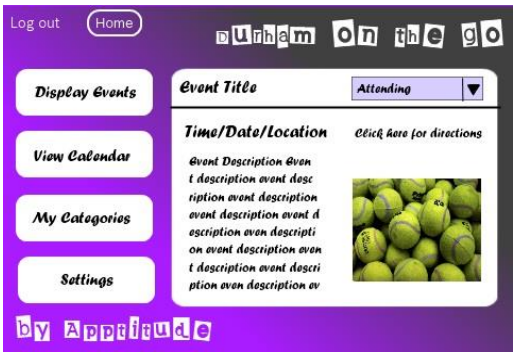


Figure 2.1.4: Specific Event Screen

Figures 2.1.2-2.1.4 have been designed to be aesthetically pleasing for the stakeholders and aimed to fulfil the user-interface factors identified by Pressman.

Figure 2.1.2 shows the log in screen design. Here the user enters his/her email address and password and can login to their ‘Durham on the go’ account. We’ve ensured that the textboxes on this screen line up to improve aesthetics, and having a good visual structure should make using the app a more satisfying experience.

Figure 2.1.3 shows the ‘My Categories’ screen design. Here the user can view his/her chosen categories, add categories and remove any categories they wish. We decided that a drop down menu would be the best way to display the categories (for reasons explained on p9). Also, we chose a dropdown menu over radio buttons or tick boxes would take up too much space on the BlackBerry screen, which already has limited space.

Figure 2.1.4 shows the specific events screen design. When a user clicks on an event that is displayed on the events feed, they will be directed to this page. A photo relevant to the event will appear on this page to improve aesthetics.

All of the screens have the same background, fonts, logo’s and layout (except the log in screen). This has been done to make the pages consistent and also to make the app easy to learn and easy to use.

Design Changes

Once we began implementing the BlackBerry App, changes had to be made to the design. Due to the nature of the BlackBerry platform, we decided that some aspects of the design would be more aesthetically pleasing if they were displayed in a slightly different way. Also, user feedback from a prototype demonstration resulted in some changes being made.

[1]

Figure 2.1.5: Log In Screen

Figure 2.1.6: Events Feed

Figures 2.1.5-2.1.7 show the final GUI designs for the BlackBerry app.

Figure 2.1.5 is the GUI design for the final log in screen. The only alteration that was made was that a submit button was included. We decided that it would be easier for the user to have a button they can press to log in, rather than just having to press enter on the BlackBerry [1], which the previous design assumed, as this could be quite ambiguous, and ultimately, we want to make the APP as user-friendly as possible.

Figure 2.1.6 is the final GUI design for the events feed screen. We decided to include arrows on the buttons to make them stand out more, again to improve the usability. We also made the buttons look 3D as we thought this made the design more aesthetically pleasing and the buttons change colour when pressed so that user knows that they have pressed the button and that the app is working.

Figure 2.1.7 The biggest change we made was to the specific event screen. Due to user-feedback, we decided to remove the left-hand-side menu buttons so that more information about the events can fit on the BlackBerry's limiting screen size. The user now either has to press the 'Home' button, or the return button [2] to get back to the main menu page with all of the navigational buttons. Neilsen (2000) said that "a typical page should be 80% content with the remaining real estate dedicated to navigation and other features". We have used this knowledge and transferred the percentage content suggestion to aid our app design. Removing the left-hand-side menu emphasises the content, which ultimately the user is using the app to see, it also gets the percentage of content on the screen closer to what Neilsen advised was optimal.

The final GUI designs for the BlackBerry APP were fairly similar to the original designs that were based on group ideas as well as survey feedback. We did make several changes, as a result of user feedback, and group decisions based on how the GUI looked once we started to implement the APP. We think the app conforms to the 8 factors Pressman (2005) identified a user-interface should be and should therefore provide the end-user with a satisfying experience.

Web App GUI Design Phase

Initially, the Web APP was designed on Lumzy. We then discussed the Lumzy design as a group and decided whether the layout was suitable to fulfil the requirement that the APP has a consistent design (NFR001), consistent layout (NFR002) and is easy to navigate (NFR005).

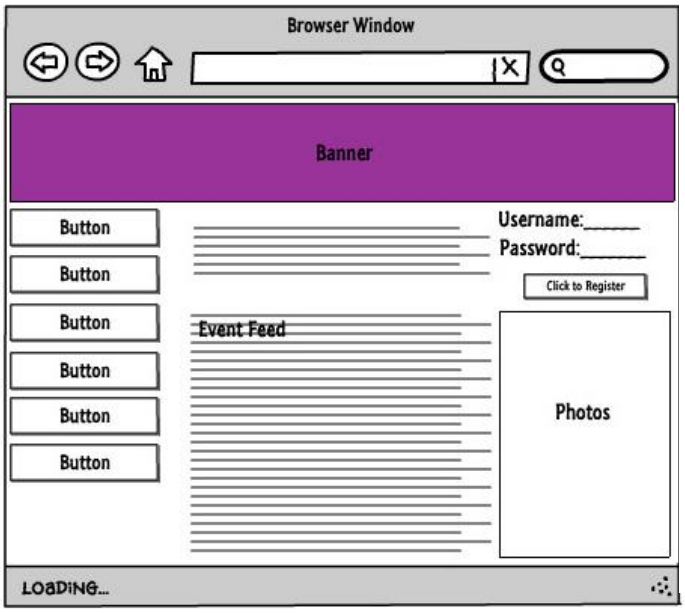


Figure 2.1.8: Initial Lumzy Design

This was our initial design. We decided that the buttons should be coloured to reflect the different categories so that the user can identify the category quickly, to aid efficiency of the system. We agreed that the general layout was suitable to fulfil the consistent layout (NFR002) and easy to navigate (NFR005) requirements. This is because the layout will be appropriate for all of the pages, so it will be consistent, and the buttons, feed and login areas are in the locations people expect to find these

The next stage involved designing the banner for the Web APP. Several designs were created on paint.net before we agreed on final banner. We based our decision on user-feedback, as well as what we thought looked the best and would appeal the most to the stakeholders.

Figure 2.1.9: First Banner Design

This was the initial banner design. The text that is used in the APP was chosen to feature in the Web APP banner for consistent branding. However, we decided that we wanted a cleaner font for the Web APP as a computer screen is a lot bigger than a BlackBerry screen and we think a different font would be more aesthetically pleasing. Also, we think that the age range who will use the app would be younger than the web app users, and as a result we decided to use a font in the app that we thought would appeal to a younger age range. For the web app we decided we should use a font that would appeal to a wider age range. Also, we thought that the cathedral and bridge images were a good idea as they highlight some aspects of Durham, but the way they are displayed

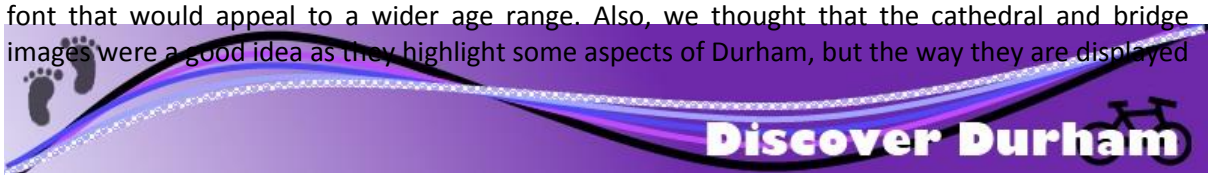


Figure 2.1.10: Final Banner Design

As a group we decided that this banner design was the most suitable for our website and would appeal the most to the stakeholders. Footprints and a bike were included in the design to incorporate the idea that Discover Durham can be accessed on the go. Also, palatinate purple has been used again, to relate to Durham, and to keep the colour consistent with the BlackBerry APP.

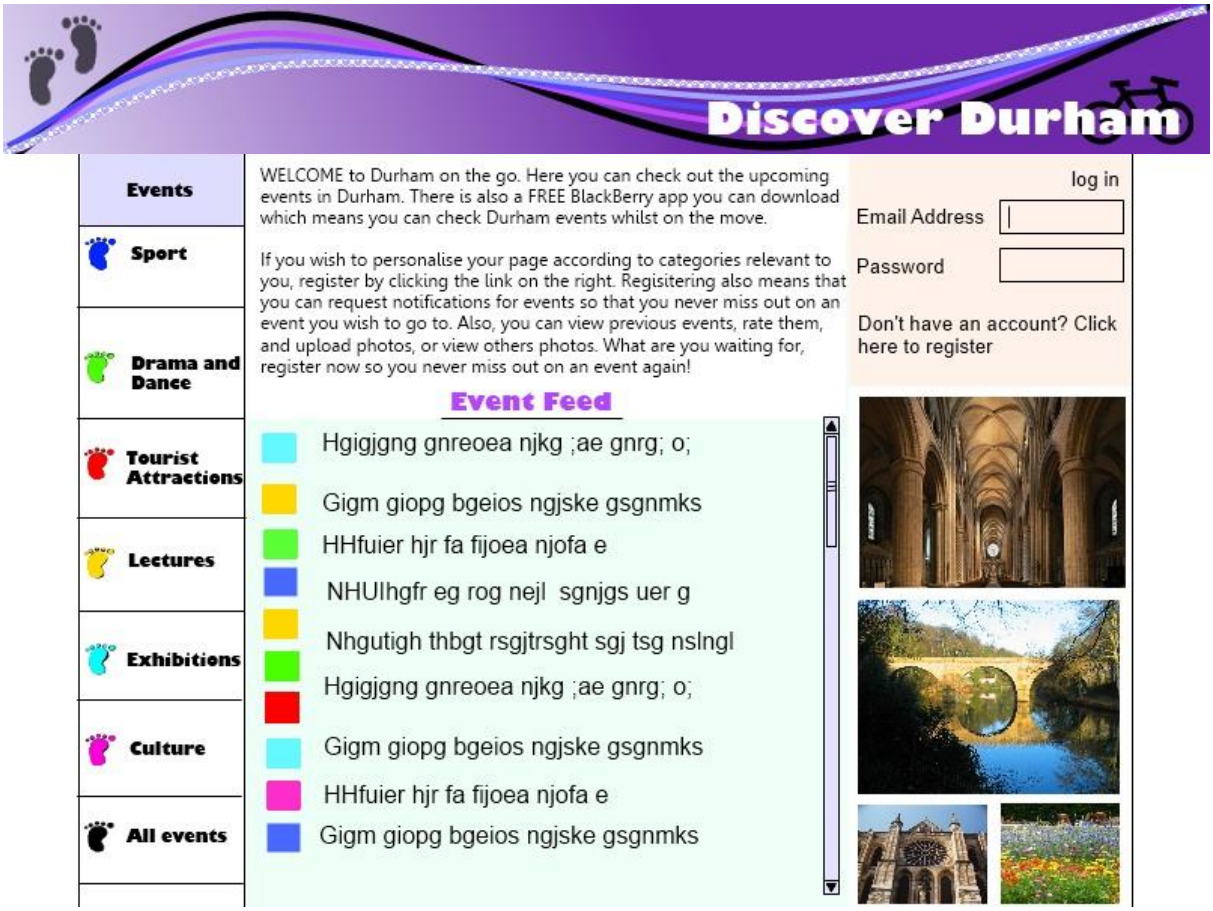


Figure 2.1.11: Web App GUI design

This is the initial web app layout design. The user's personalised categories are in the left-hand-side menu, and if the user clicks on them, they will be directed to a similar screen but the events feed will only have events associated with the category selected. We've chosen to have the categories on the left-hand-side as a lot of websites have a menu there, which should make the layout intuitive for the user. We've placed the log in box in the top right corner of the page as this is where most websites, such as Facebook, place their log in details entry box, which should make the website easier to use. Finally, we've ensured that the user doesn't have to scroll down the screen to see all of the possible buttons on the menu, as this could cause the user to miss a category, thus making the website difficult to navigate.

Design Changes

As a group, we decided that the coloured footprints were a bit immature for our target market so we changed them to just text buttons. Also, the implementation stage led us to change the left hand side menu from displaying the user's categories to items such as 'My Settings' and 'My Events', as we thought this would make the navigation of the webapp easier. The user will still be able to view their categories by clicking on 'My Settings'. Here they can also edit their categories.

Another change we made was to include our logo on the bottom of each page. We felt that it gave a more professional look and also it ensures that we fulfil the requirement that the web app will display the system logo (NFR003).

User-feedback resulted in some changes being made too. For example, when showing the web app to potential customers, they suggested that we may want to embed a google maps image rather than giving the user a link, as this aids navigation and also it will reduce the number of clicks required for the user to find out where the event is (NFR005).

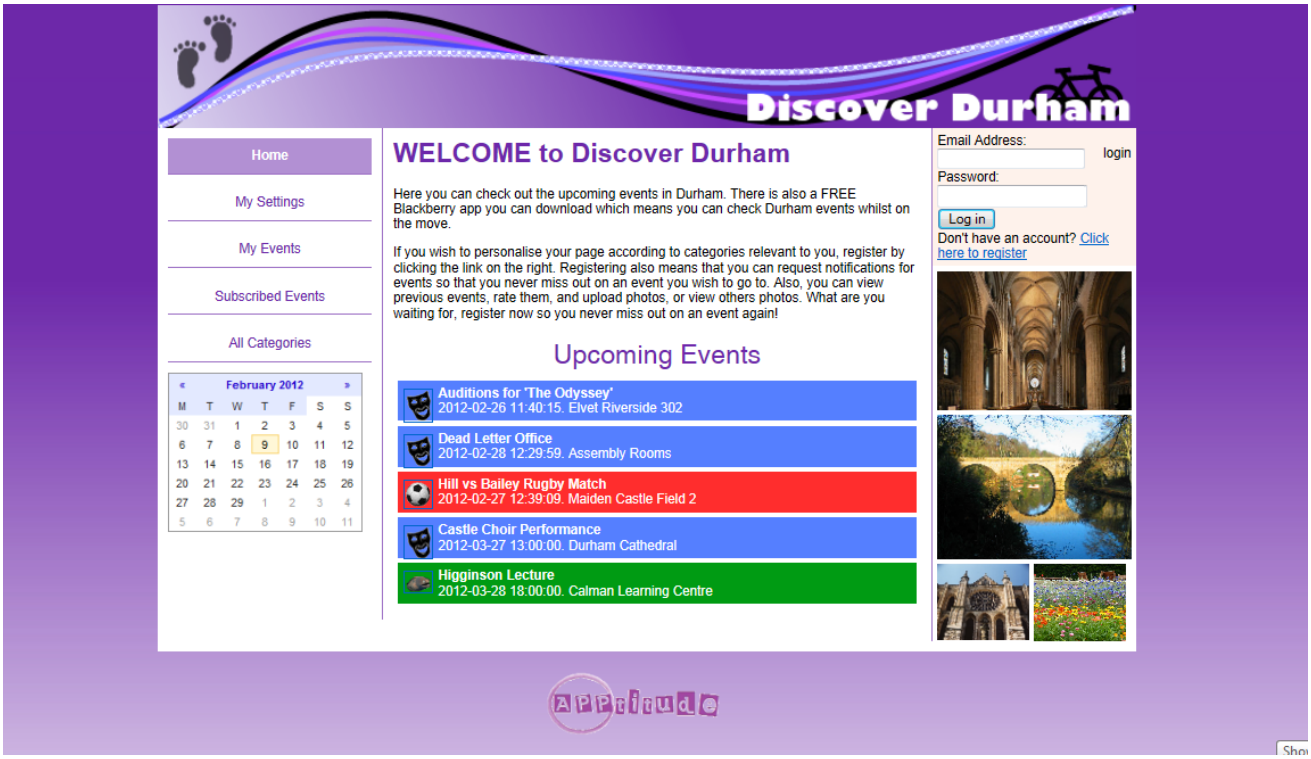


Figure 2.1.12: Final Web App home page

2.2 Process Descriptions

This section of the document describes the main operation of the websites, using "use case diagrams" and "sequence diagrams". The use case diagrams show how the main website and the BlackBerry App will be used by registered and non-registered users, as well as specialized users such as "category managers". The sequence diagrams show in more detail how the main processes are sequenced, and how messages are passed between the key objects in the system.

Figure 2.2.1 below illustrates how the main Discover Durham Website would be used by registered and non-registered users.

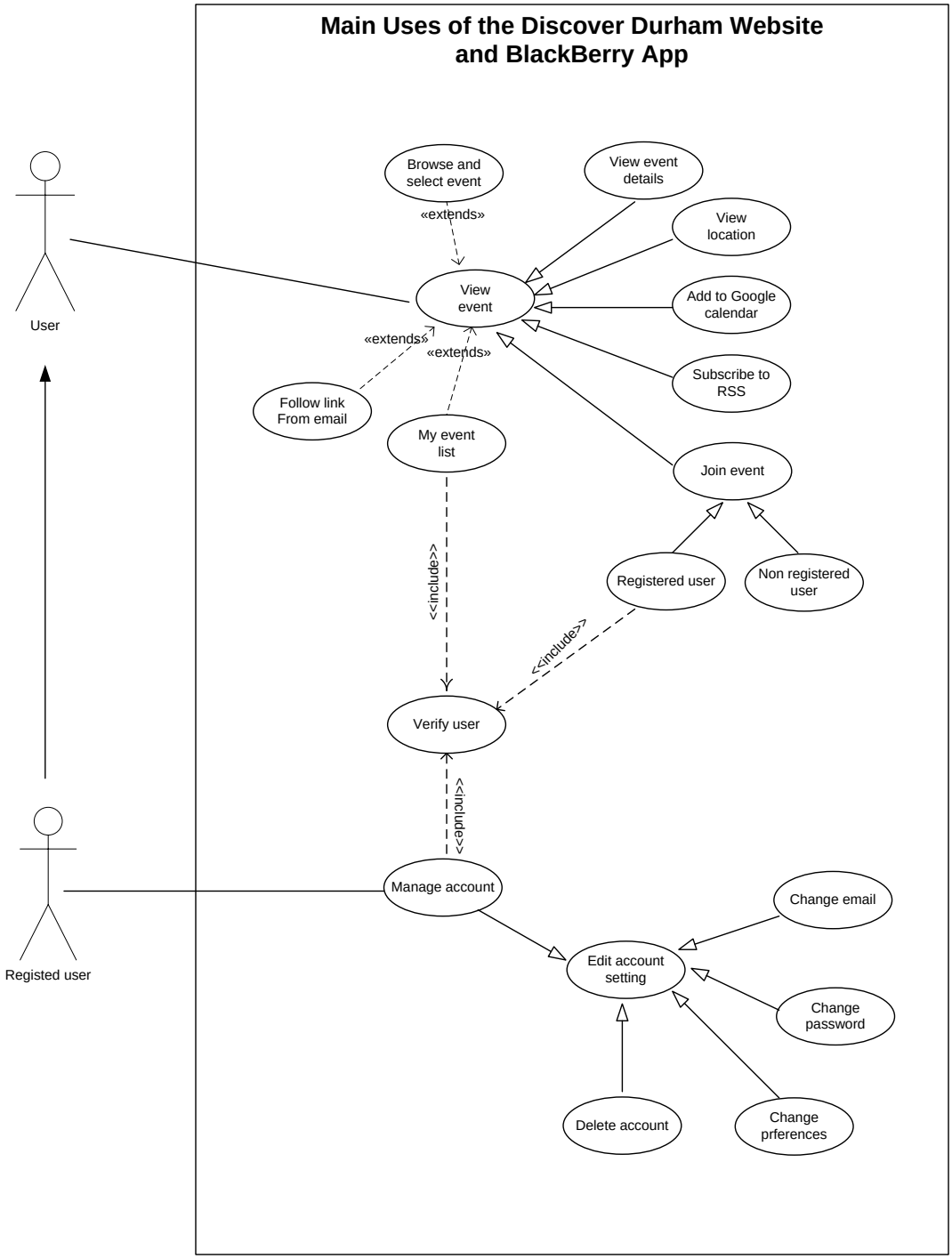


Figure 2.2.1: Use case diagram – Main uses of the web app and BlackBerry App

This diagram shows that the main use of the website by both registered and non-registered users is to be able to view events, select events of interest, view more details about an event and then join an event. Additionally, users would be able to add events to their personal Google calendar, as well as subscribe to RSS feeds relating to the event.

The diagram also illustrates that there are three routes to viewing an event: browsing by category, following links from automatically generated emails, and viewing a list of preferred events. Any user should be able to simply browse events by category on the main website. Additionally, registered users may opt to receive an email notification for events of interest, and simply arrive at an event details page by following a link in the email. Furthermore, registered users are able to browse a list of preferred events, generated automatically based on their stored preferences.

This diagram also illustrates a second main use for registered users: managing their accounts. Registered users should have the ability to edit their account details, as well as delete their account.

Use case: category management

Figure 2.2.2 below illustrates a second main use of the website: category management.

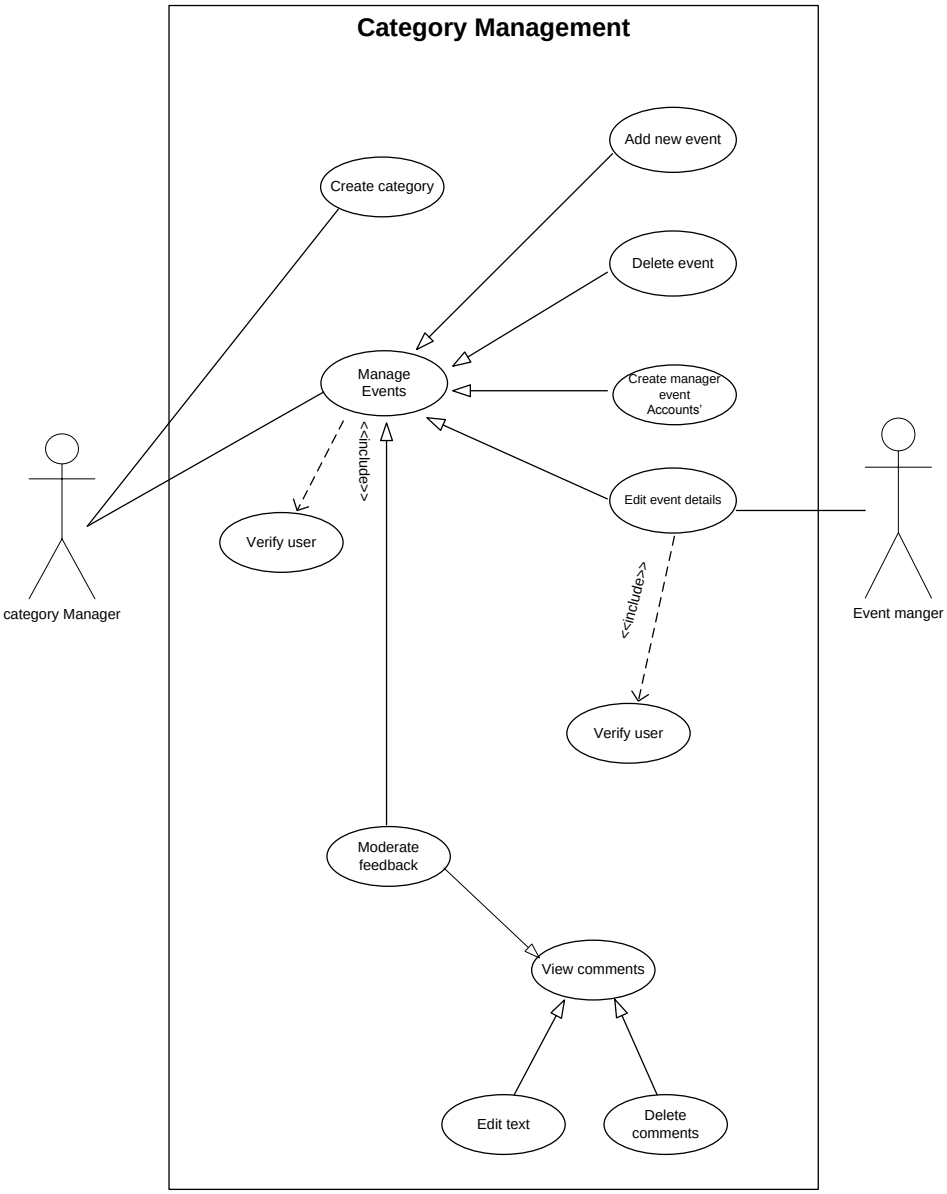


Figure 2.2.2: Use case diagram – Category management

In this use case (figure 2.2.2), a specific type of user called a 'category manager' is responsible for the management of categories and events. As such, category managers have the ability to create categories, manage events and moderate feedback.

The category manager is able to create categories, which can be named after activities or venues, such as 'sports' or 'cathedral'. Category managers then have the ability to create and manage events within any category they have created. Additionally, they should be able to create a specific type of user account called an 'events manager account'; this type of account is limited to only being able to edit event details. Furthermore, category managers have the ability to moderate feedback, which includes being able to edit feedback or remove it completely.

Sequence diagrams

The sequence diagrams below show the sequence of events which are required by the following key processes of the system:

- Login
- Browse by categories
- Browse "My Events"
- View event details
- Join event
- Category management
- Manage user account

Using the website as an event manager includes two main functions:

- Manage user accounts
- Category management

For each diagram, the main objects are as follows:

- **User:** this refers to the current user who is using the website.
- **Website:** is the main interface between the users and the system. In the case of the BlackBerry App, this will be the main GUI.
- **Web service:** is gives the extra database functionality to the website.
- **Database:** is used to save users, categories and events details.

Login

Figure 2.3 bellow shows the sequences involved the login process. This is a general login, which is referred to in the other sequence diagrams. It follows a general login procedure of verifying a username and password entered by the user against information held in the database.

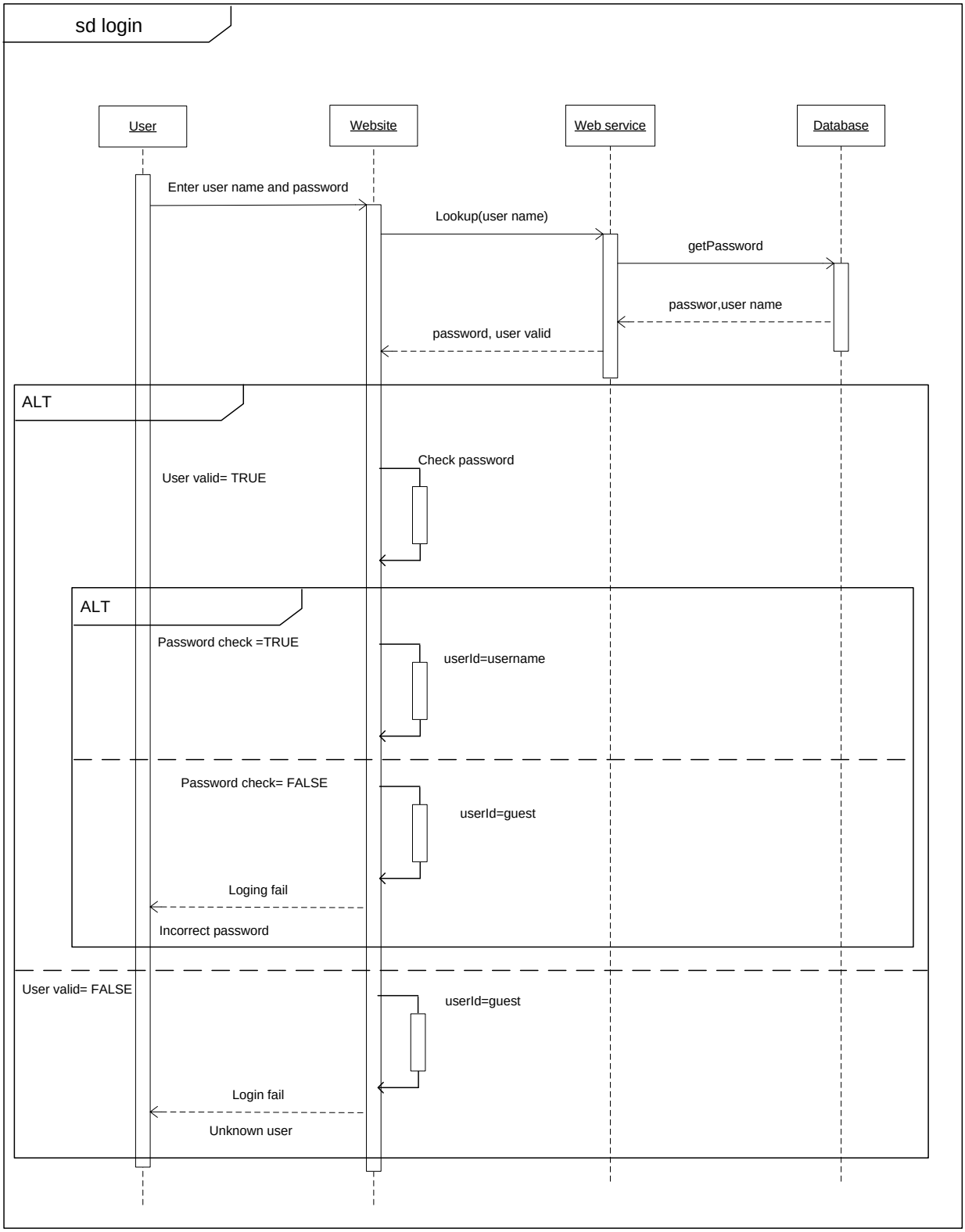


Figure 2.2.3: Sequence diagram – log in

Browsing by category

Figure 2.2.4 below shows the main sequence of events involved when a user browses events by category. This will be the default method of browsing events.

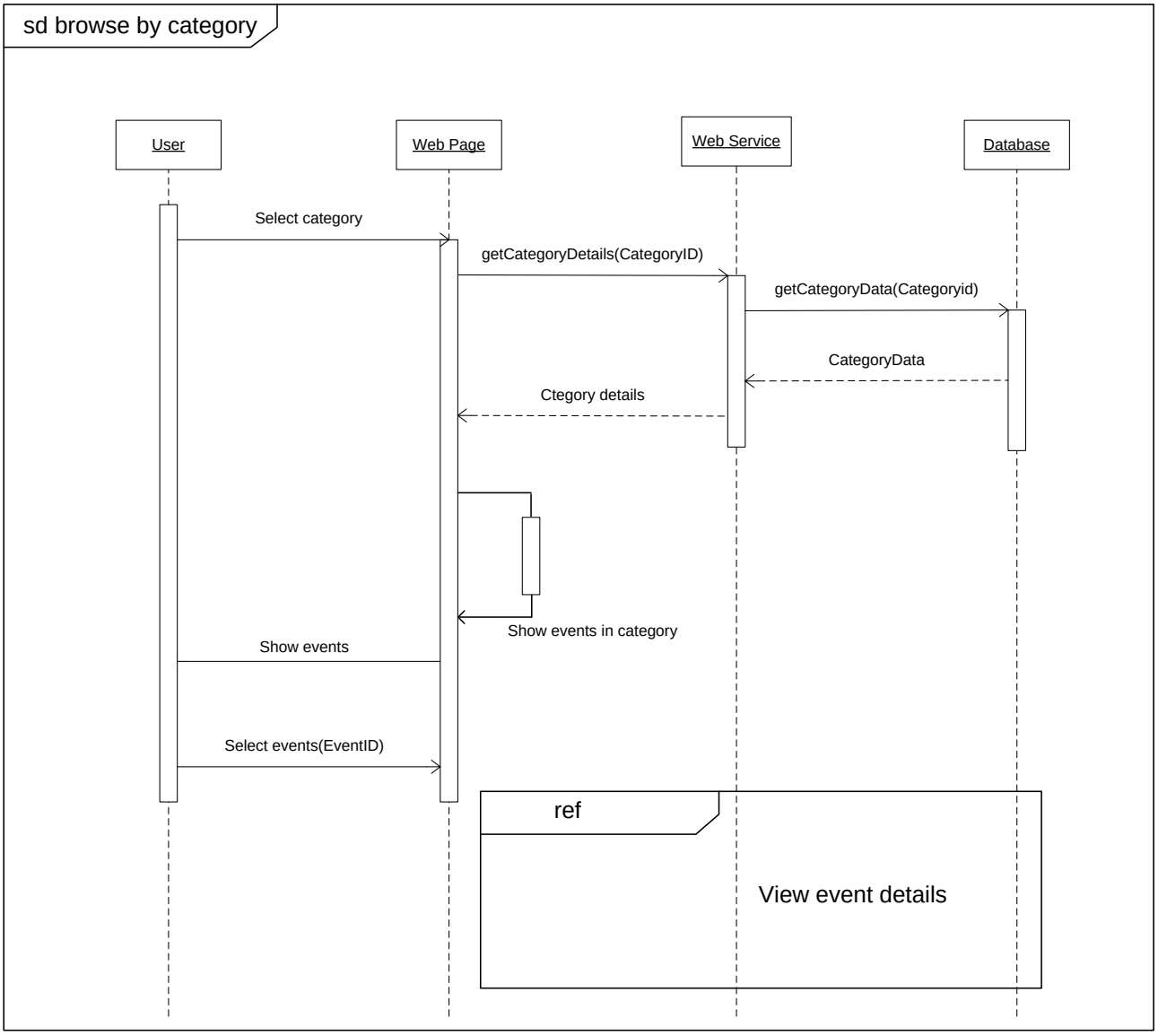


Figure 2.2.4: Sequence diagram – browse by category

Browsing "My Events"

An alternative way to browse events would be for a registered user to browse a list of categories based on their stored preferences (see Figure 2.2.5 below). After checking/asking the user to login, the user's preferences would be retrieved from the database and used to show a specific list of events.

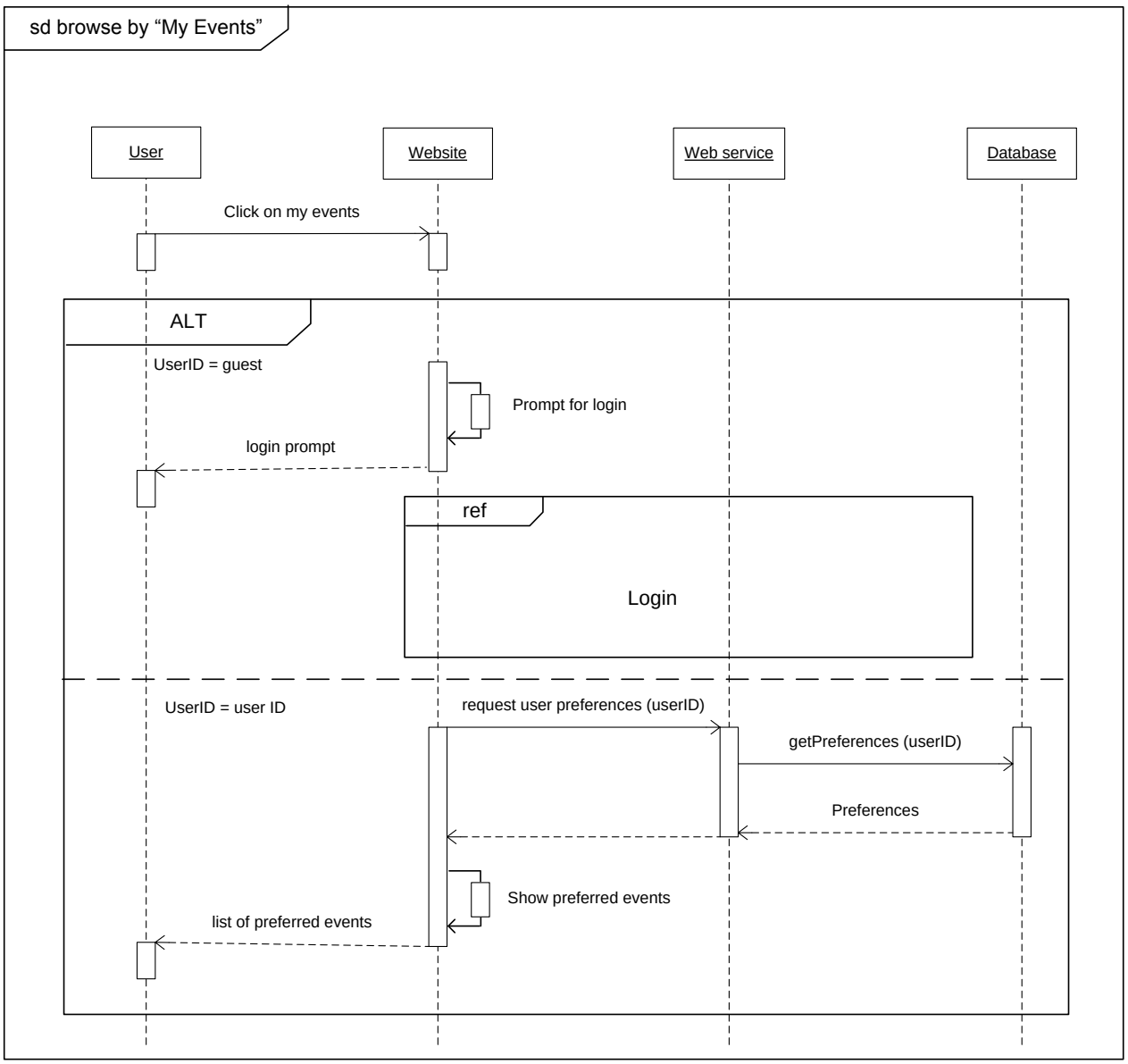


Figure 2.2.5: Sequence diagram – browse by ‘My Events’

Viewing event details

Once a user has selected an event, they would then be able to view the details of that event and would be presented with various options: view map; add to calendar; add RSS feed; join event. Figure 2.2.6 below shows the main sequence of operations involved in viewing event details and selecting options.

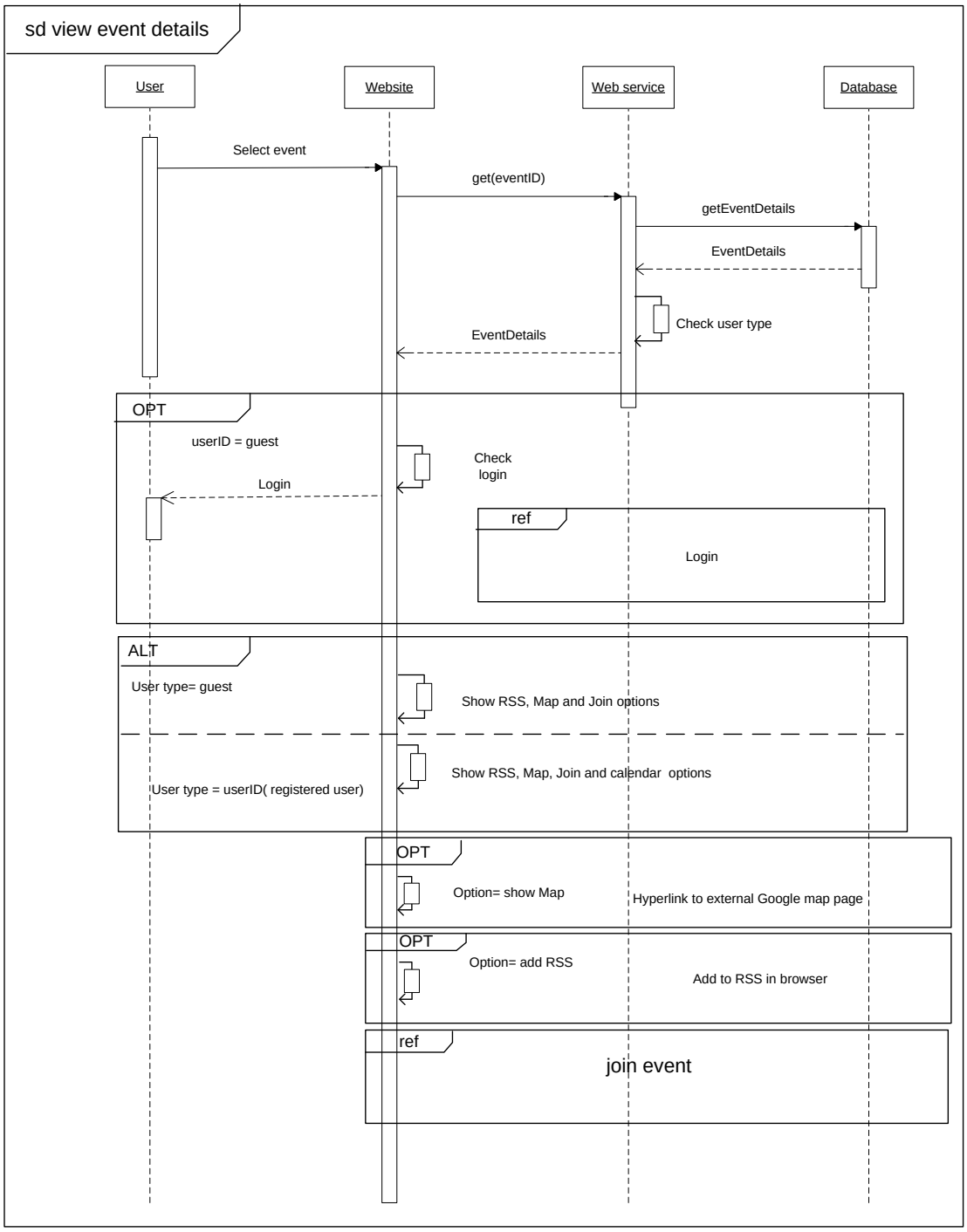


Figure 2.2.6: Sequence diagram – View event details

As can be seen in Figure 2.2.6 above, the parameter 'user type' is used to keep track of whether a user is logged in as a registered user, or is using the website as a guest (non-registered) user. The 'add to calendar' option would only be available to registered users who have logged in. Users will be able to log in at any time, and after doing so, 'user type' would be updated.

Joining an event

Figure 2.2.7 below shows the main sequence of events when a user chooses to join an event. After first checking the availability of the event, the userID and eventID are stored in the database. A 'confirmation' or 'unavailable' message is returned to the user as required.

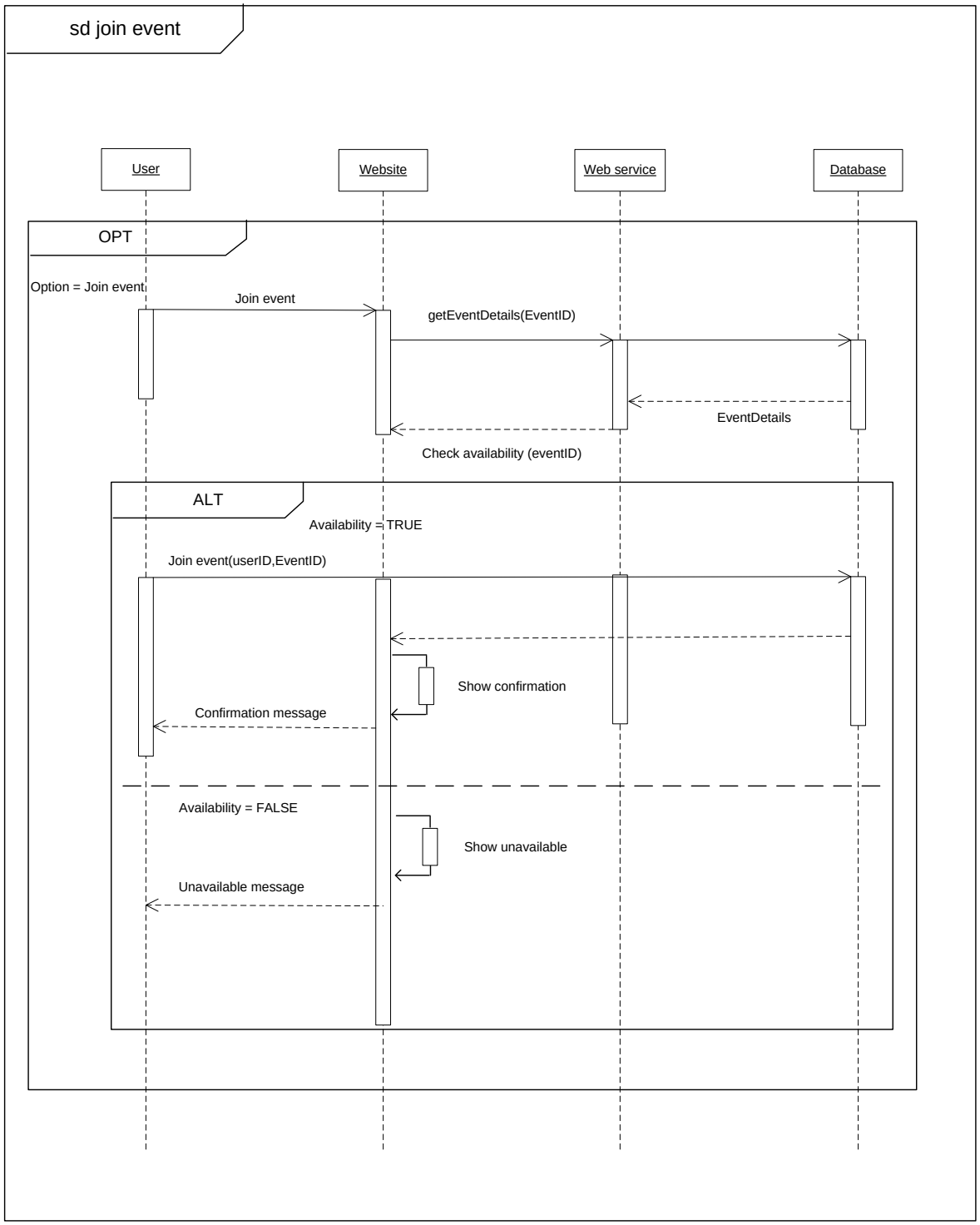


Figure 2.2.7: Sequence diagram – join event

Management processes- Category management

The category management process, shown in Figure 2.2.8 below, will be available to users identified as being category managers (see Figure 2.2.2 - use case diagram for category managers). When a category manager creates a new category, the database is first checked to see whether or not that category already exists; if it does not, the new category is then stored in the database.

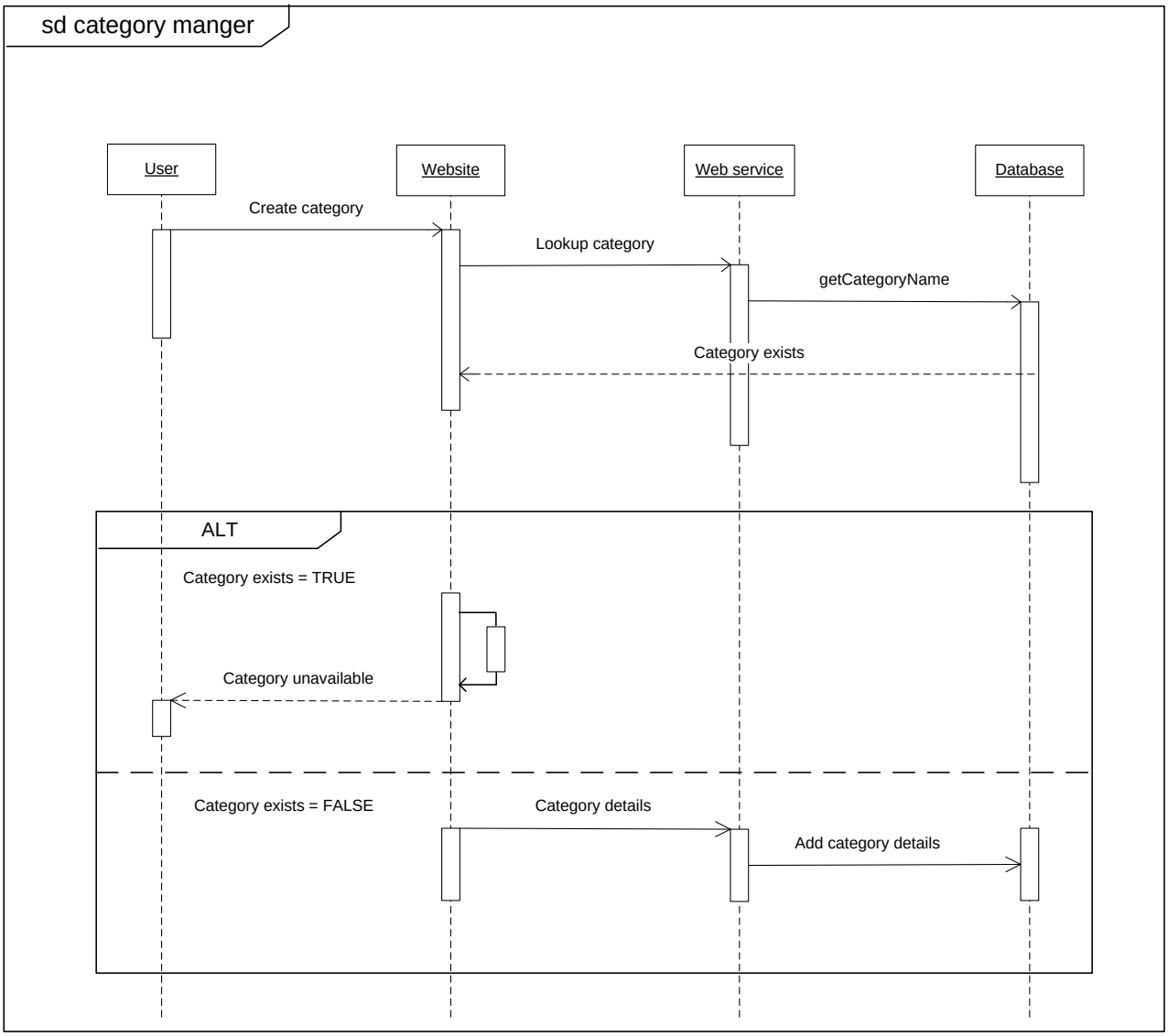


Figure 2.2.8: Sequence diagram – category manager

Managing user account

Figure 2.2.9 below shows the main sequence of events involved when a registered user wishes to manage their account. The 'user type' flag is checked, and if the current user is logged in as a 'guest' they are prompted to log in as a registered user. After verification, they will then have the option to edit or delete their account. And changes would be updated to the database.

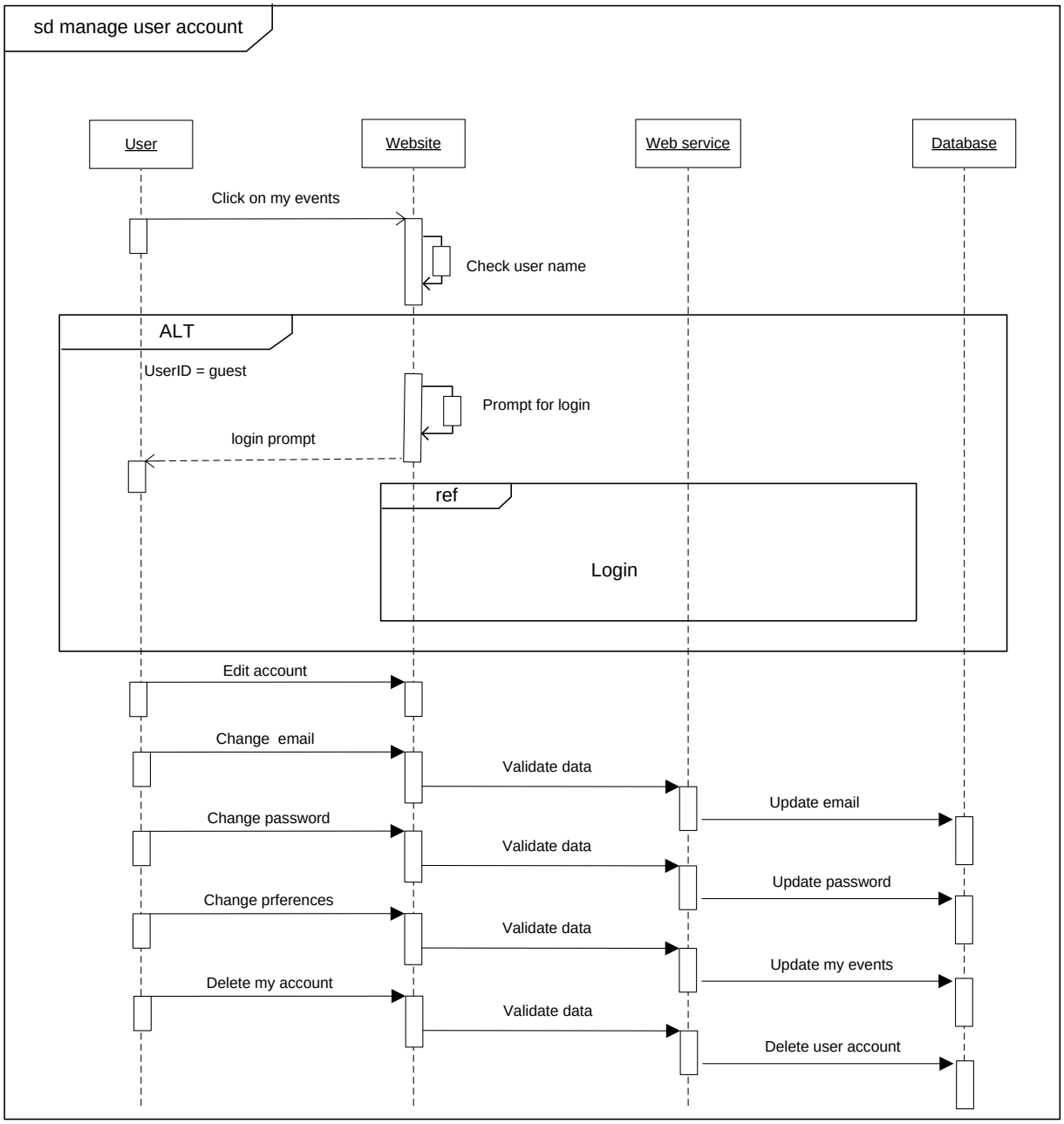


Figure 2.2.9: Sequence diagram – managing user account

3. Element Descriptions

Users of the system will be able to use two pieces of software (the BlackBerry app and the web app) to interact with the system and this software will therefore need to share the same data. These two applications will have access to a central MySQL database where the data required for the system's functions will be stored. It is on this principal that decisions for the architectural and design level abstractions have been taken.

System Architecture

The system architecture is drawn up to provide a framework on which design decisions can be made. It provides an abstract view of our solution and sets out the characteristics that the system will take on at design level.

The system is being designed using a data-centered repository architectural style; there is a central, permanent storage location for all the data – this is kept completely separate from some arbitrary number of applications which can access and process the data. More specifically the system will take the form of a client-server model; the web app and BlackBerry app will be the clients and the MySQL database will be the server for these clients.

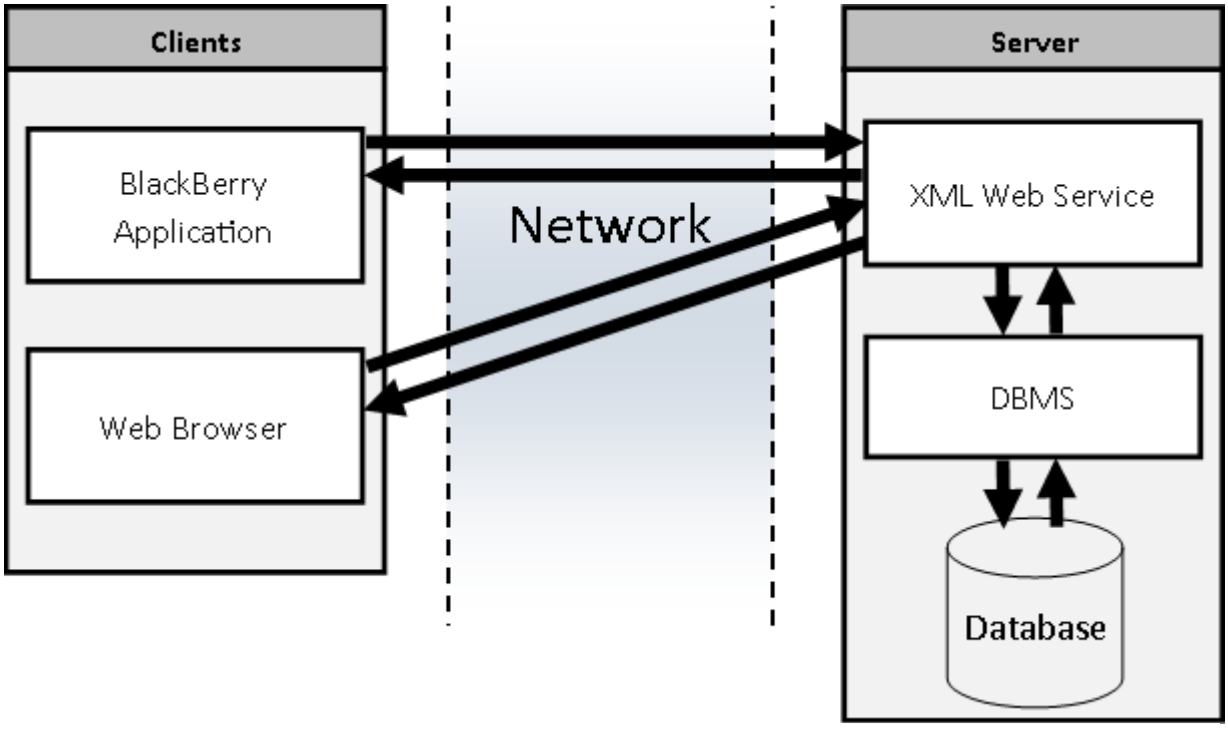


Figure 3.0.1: System representation of a client-server model

Figure 3.0.1 shows how the BlackBerry app and the web app will connect to the XML web service in order to access the MySQL database. The clients send parameters to the web service which then retrieves the relevant data from the database through the database management software (DBMS). The web service then sends the requested data back to the client in the format of an XML document. The client can then interpret this XML document to present information to the user through a graphical user interface. Below is a more detailed explanation of the system architecture:

Database

This is the raw data store which is accessed by the DBMS. It stores all the data about events and users that is accessed by the clients.

Database Management Software (DBMS)

The DBMS we are using is MySQL; it enables the web service to access the data stored in the database using SQL (Structured Query Language) commands. By using a DMBS it means that the XML web service does not need to consider details concerning the physical storage of the database, such as indexing and hashing. It also improves the scalability of the system, for example, if the physical storage needs to be transferred to a larger/different storage location, this will have little effect on the web service.

XML Web Service

This REST based web service takes parameters from the clients as input, creating MySQL queries based on this input to access the database. The service then formats the results of these queries into an XML document which provides the data that the client originally requested.

The web service has been created to make the system more flexible in terms of maintenance and scalability. If the clients connected directly to the DBMS using queries, if any changes needed to be made to the database it would be necessary to re-write all the queries in all of the clients. This would make maintenance very difficult, especially if the system is to expand onto more platforms in the future. By having the web service, we would only have to change the queries in the web service to reflect any changes made in the database. Additionally, by having the web service separate from the clients, testability is improved because it will be easier to isolate any problems.

XML is being used as the output of the web service because it is extensible, meaning that tags can defined to meet our requirements. It can also be easily integrated into our two clients as it is application independent, and its similarity to HTML makes it particularly suitable for the web app. XML uses plain text format and separates the information from its presentation, facilitating a customized view of the data which is needed both for providing customised event information for each user, but also to enable the data to be used differently by the different clients. This also makes the system scalable in the future; making it easier to add new clients – for example, apps on different operating systems

Web Browser and BlackBerry Application

The web browser app and BlackBerry app provide a graphical user interface (GUI) for users to interact with the system. Based on user inputs, the apps send requests to the web service; these requests can be to add, modify, retrieve or delete data from the central data repository. The requests are processed by the web service and the relevant data is returned to the app as an XML document. The apps use this XML document to display information to the user through a GUI.

The web app will be written as a HTML page using PHP and will display the information from the XML web service in a user-friendly way using a graphical user interface (GUI). The interface is designed to be suitable for use within a web browser on a personal computer.

The BlackBerry app will be written in Java and make use of the BlackBerry API to present information from the XML web service in a graphical user interface (GUI). The GUI will be designed to present this information in a format that is suitable for a use on a BlackBerry.

Advantages and Disadvantages

Using the client-server model for our system had several advantages and disadvantages:

Advantages

- The system is easily scalable because the number of clients the system can have is effectively limitless. In addition, all the resources are on the server-side so to increase storage for example, would only require changes to the server.
- System changes and maintenance can be done easily because only the server-side components would need to be changed

Disadvantages

- Too many requests from clients at the same time can cause congestion and provide a slower service to clients

3.1 Intermodule Dependencies

The intermodule dependencies show the static behaviour of the system

XML Web Service

The XML web service interacts with other modules to provide scalability to the overall system. The XML web service acts between the clients and the DBMS to allow the communication of data. For the client to request data from the DBMS, it sends a URL request to the web service, which returns an XML document containing the data requested. The web service generates this document by creating an SQL query which is passed to the DBMS, and then it processes the results into an XML document. The requests that the web service should accept and provide and provide an XML document for can be seen below:

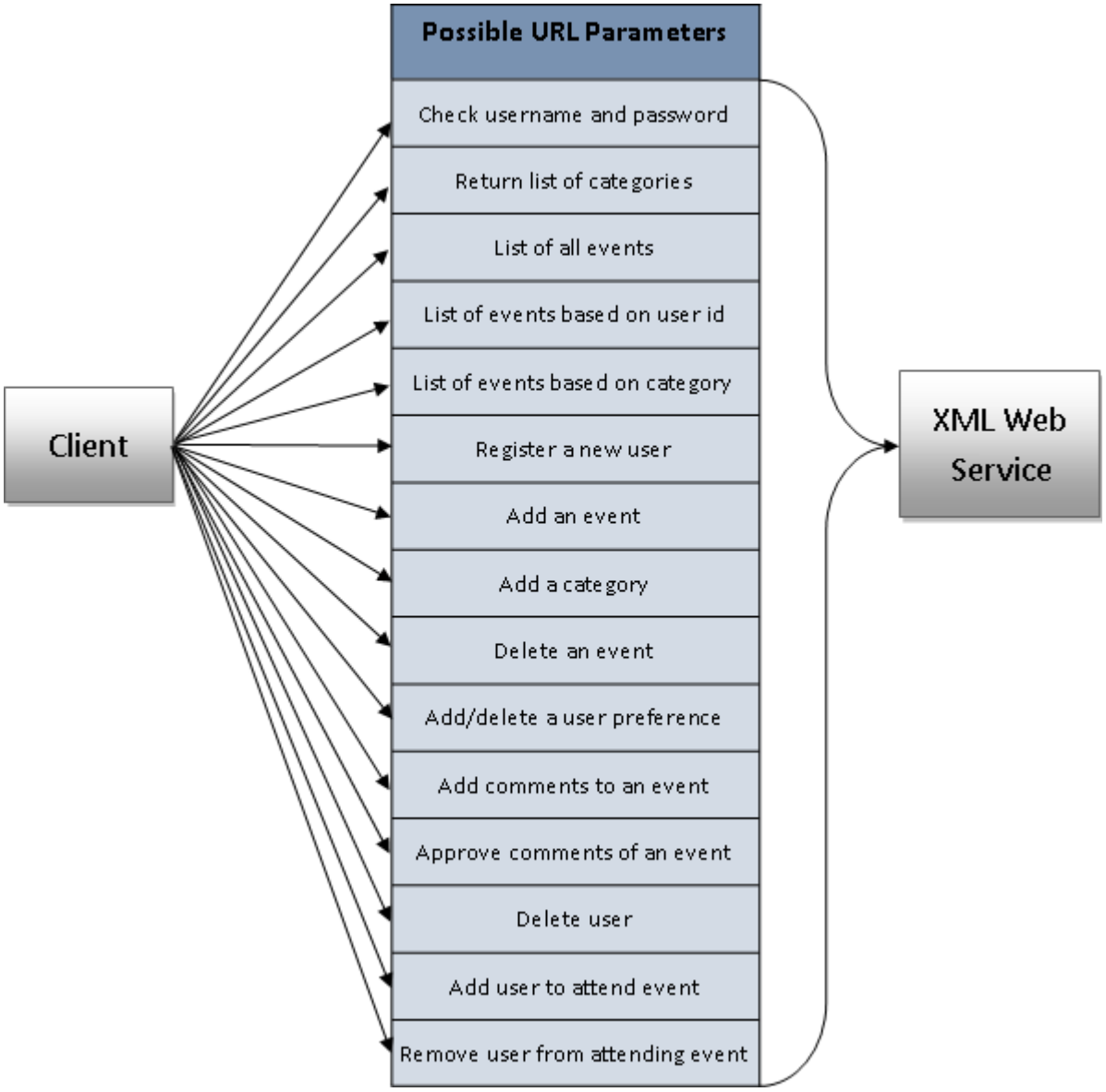
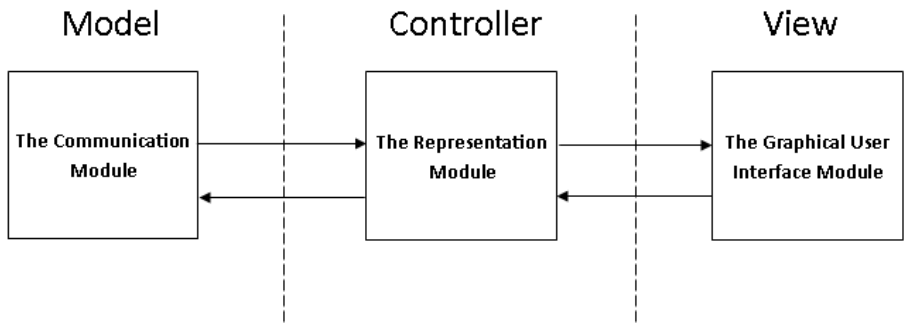


Figure 3.1.1: Possible parameters for the XML Web Service

BlackBerry App

The BlackBerry app has been broken up into modules, each of which performs its own task within the overall app. The idea is to reduce the complexity of the program so it is easier to write, maintain and test. To help achieve this reduction in complexity the modules have been designed in a way to try and ensure that interdependence is minimized. The app has been spilt up into modules using the MVC architecture as outlined below:



THE COMMUNICATION MODULE

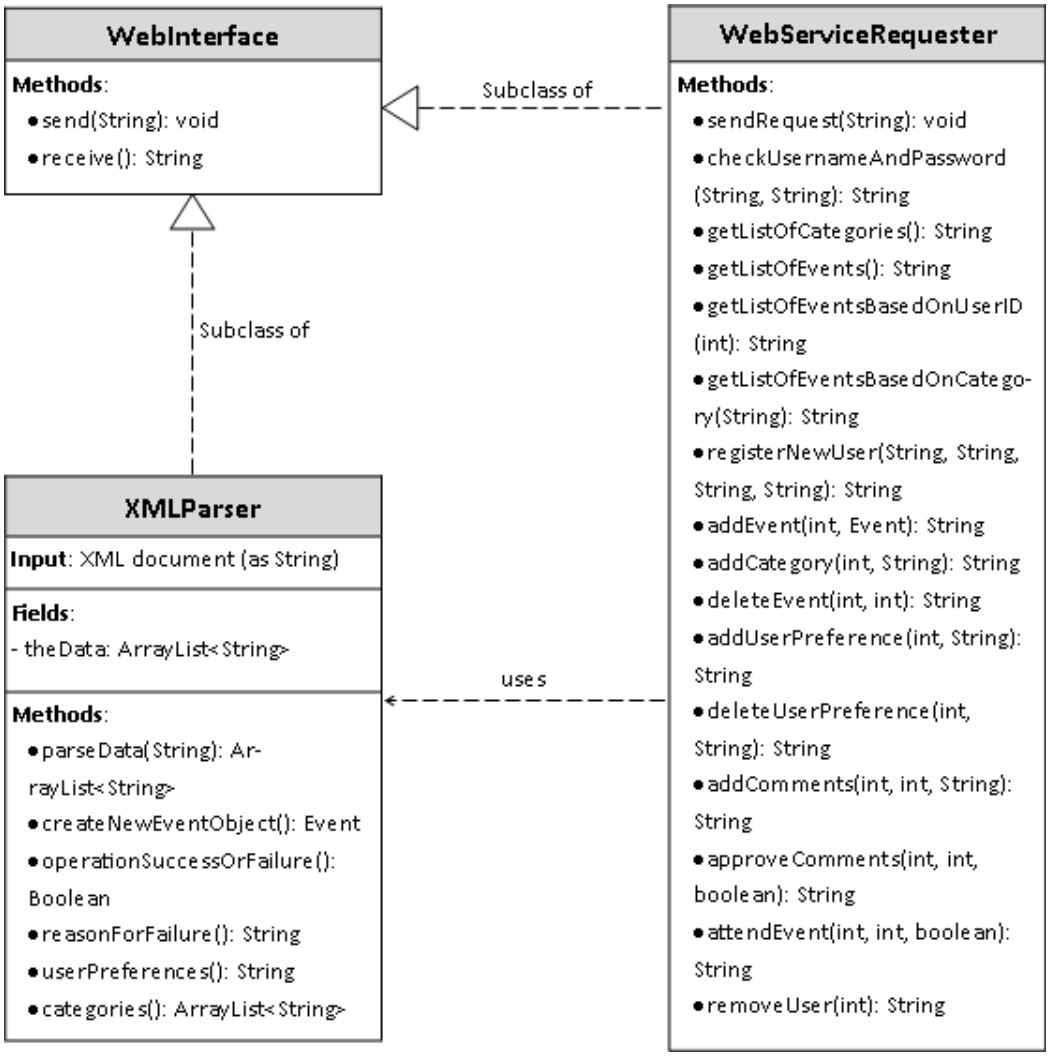


Figure 3.1.2: Class Diagram of the Communication Module

The communication module deals with the sending and receiving of data to and from the XML web service. The 'WebInterface' class deals with the networking and physical data aspect of the module. It is used to send a URL request and it listens for any data that is sent back by the web service. The 'WebServiceRequester' class has many methods which can be called to generate a URL that is passed to the 'WebInterface' class for sending. This URL acts as a parameter for the web service. The 'XMLParser' class receives the XML document from the web service and stores it in a field. Depending on the data received, the class will use the data in as parameters for methods to carry out tasks within the app.

THE REPRESENTATION MODULE

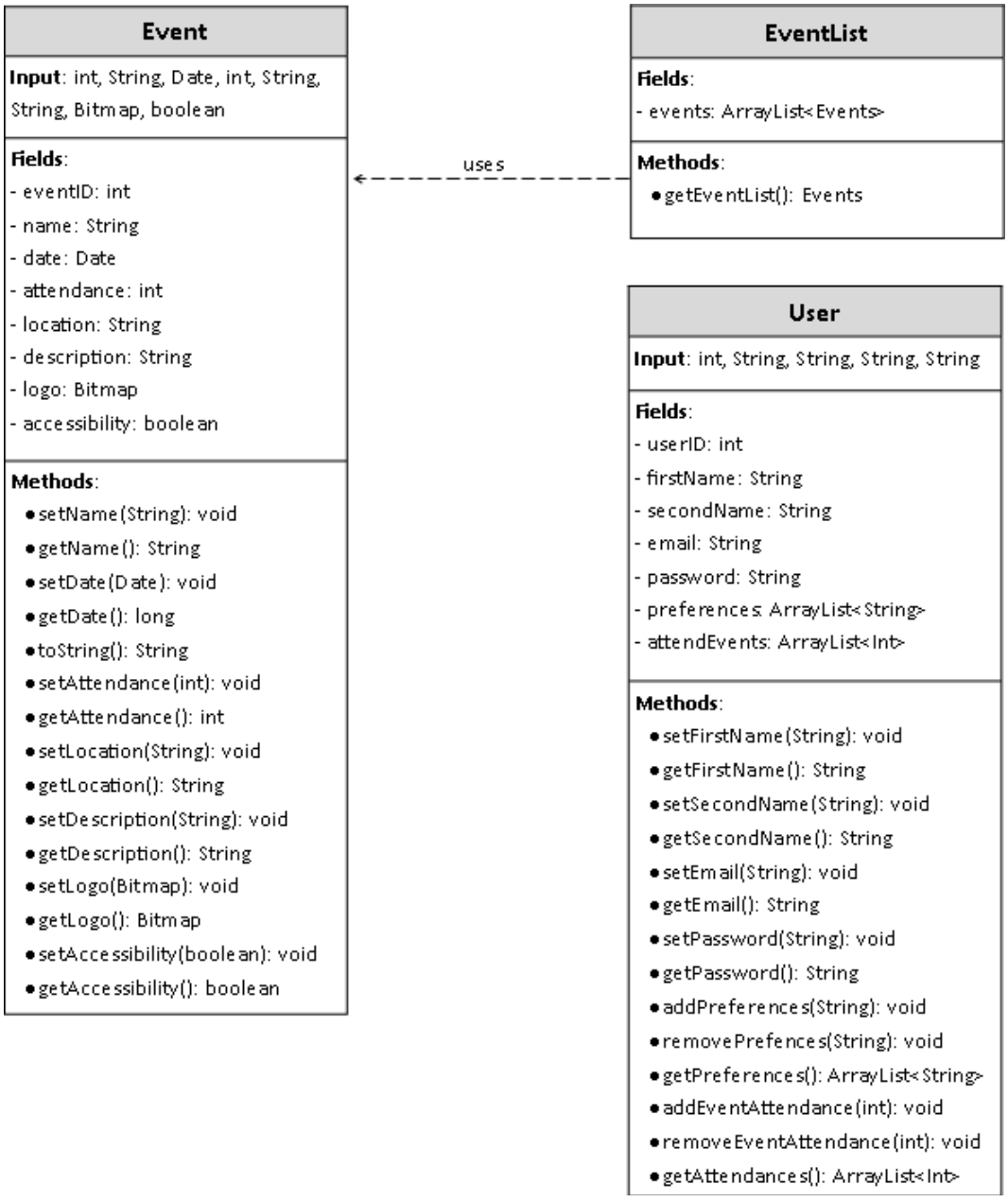


Figure 3.1.3: Class Diagram of the Representation Module

The representation module deals with temporarily storing data locally within the app so that it can be accessed by the GUI module. It stores information about the user in the 'User' class and information about an event in the 'Event' class. The 'EventList' class stores details about many events by storing instances (objects) of the 'Event' class. All of this storage is temporary in the sense that when the app is closed, information will not be stored permanently in the app; all information is stored in the central repository on the server-side.

THE GUI MODULE

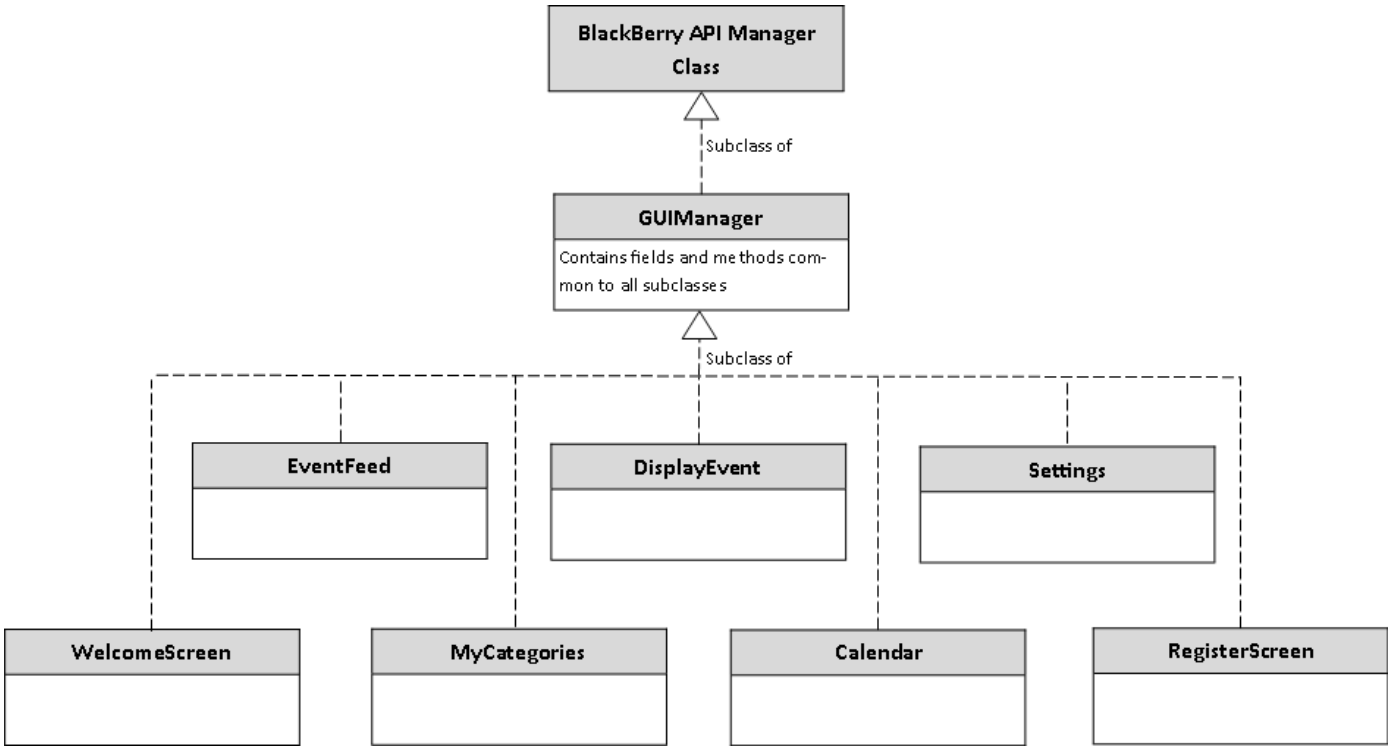


Figure 3.1.4: Class Diagram of the GUI Module

The GUI module will provide a robust and interactive user interface using the BlackBerry API. The 'GUIManager' class contains all the common properties that can be found across the different screens within the app. There are separate classes for each screen within the app, which will contain the appropriate fields relating to the design of that screen. The GUI module displays data from the representation module, and passes any user interactions onto that module.

Web Browser App

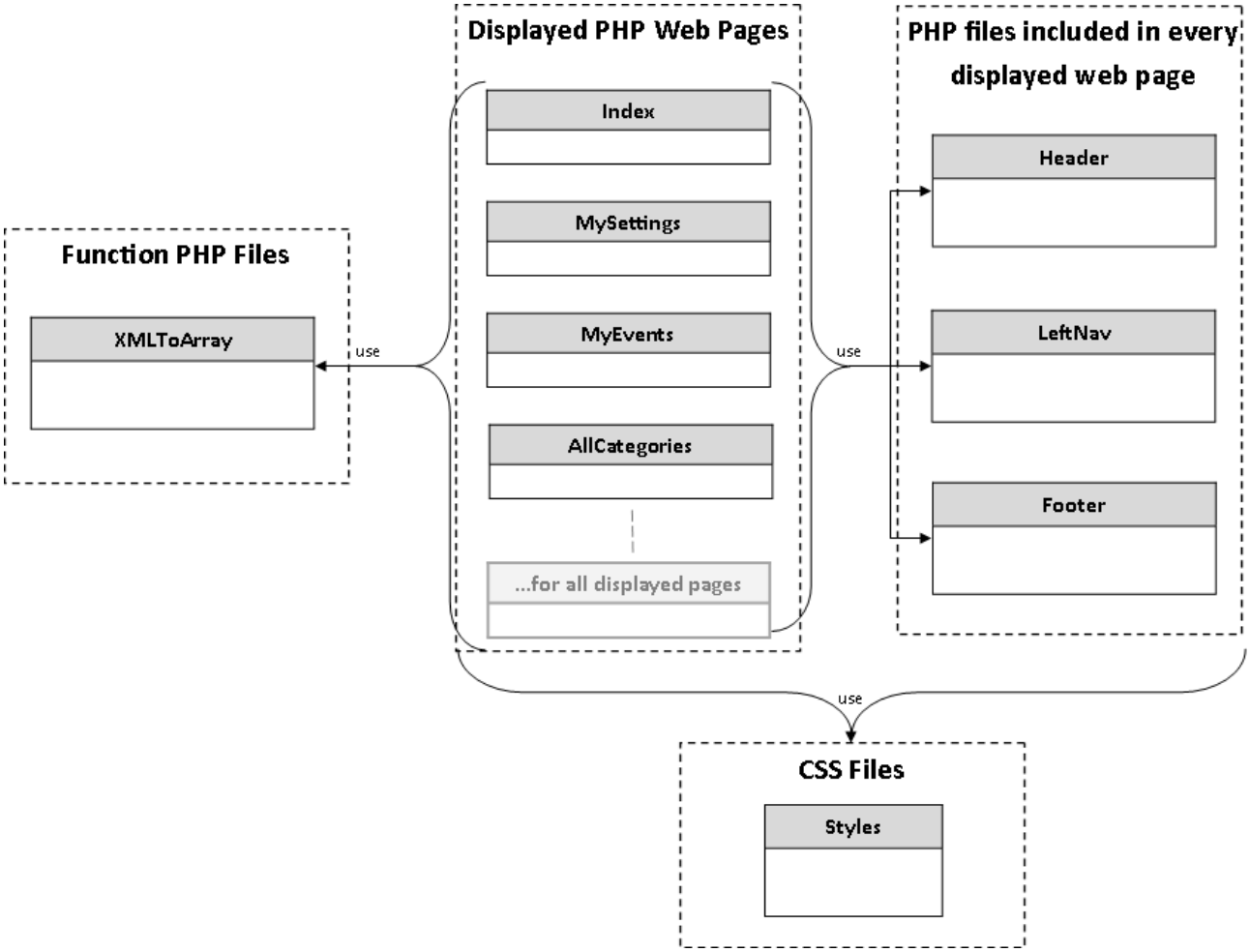


Figure 3.1.5: Web App Architecture Design

The 'XMLToArray' file converts an XML document, accessed using the native php method 'file_get_contents(String xml_location)', into an array of details that can be used by the website. The array returned is then used by each page to display the results of the web service call. The web service also handles updates and insertions to the database so it is also used by the MySettings and the Category pages to facilitate updating user preferences and settings.

All of the displayed web pages access the 'Header', 'LeftNav' and 'Footer' files to provide a graphical layout. Doing this instead of coding the graphical layout on every single displayed web page makes the website more maintainable; it will be easier to add new pages and change the graphical layout of the pages. In this case, only the 'Header', 'LeftNav' and 'Footer' files would need to be changed. Additionally the styles for the web pages (excluding the 'XMLToArray' file which will have no styling) are all contained in one CSS file to provide users with a consistent look across all pages.

Some of the web pages will only be accessible to users that are logged in as event managers. These web pages will provide features that should only be accessible to event managers, such as adding and changing events.

Database

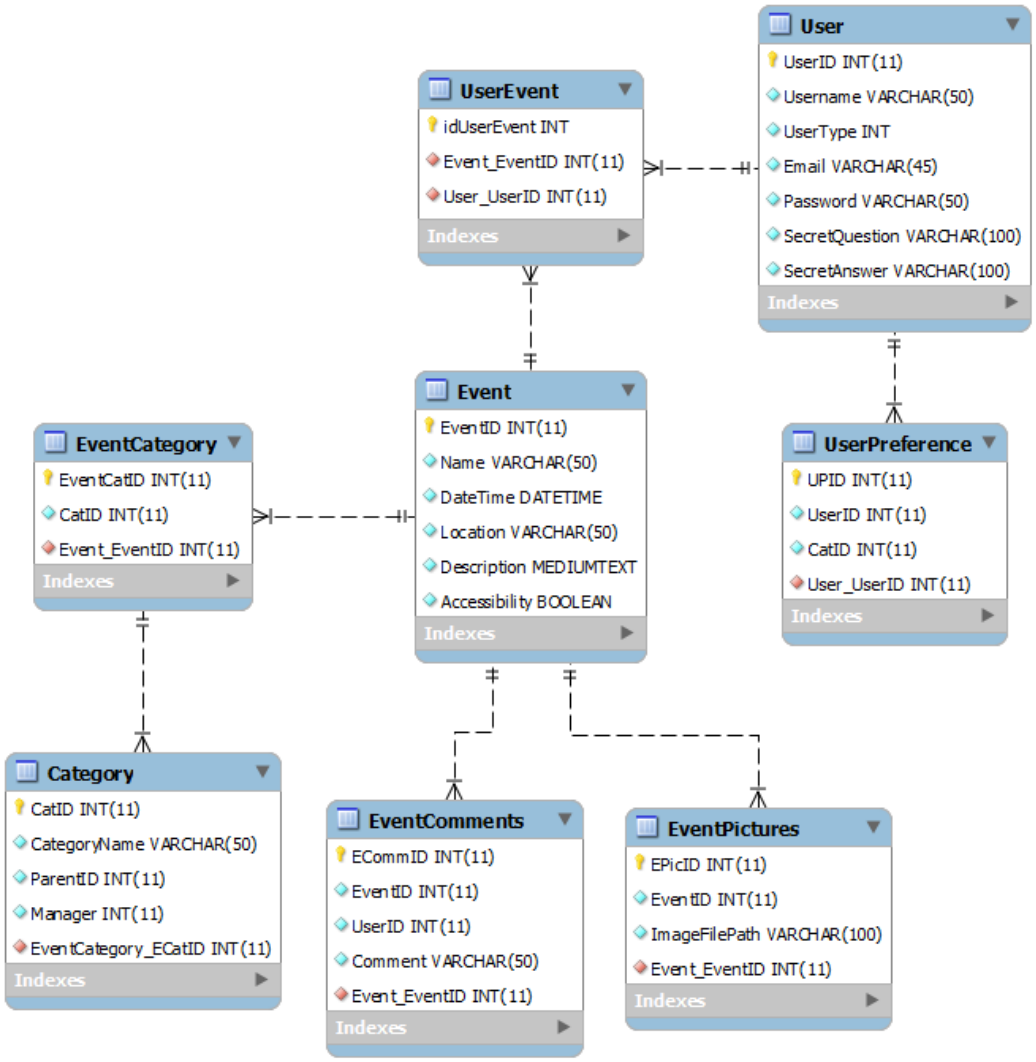


Figure 3.1.6: Entity Relationship Diagram for the Database

The database is the central store for the data that is used across the two clients. The diagram was created by drawing up entities based on representations of real world things from our requirements, such as events and users. The attributes required to meet the requirements of our clients were then added. The diagram was then modified to conform to remove any many-to-many relationships and to ensure the database conforms to 3rd normal form.

In the cases of the relationships between 'Events' and 'Category', and also between 'Event' and 'User', there were many-to-many relationships present. Relational databases cannot support this type of relationship, so link entities have been created between the entities in the form of 'EventCategory' and 'UserEvent'.

The database should conform to 3rd normal form; that it conforms to these rules from 1st, 2nd and 3rd normal form:

- 1NF: The intersection of each row and column in a relation (table) contains only one value
- 2NF: Every non-primary key attribute is functionally dependent on the primary key
- 3NF: Every non-primary key attribute is not transitively dependent on the primary key

3.2 Interprocess Dependencies

This section focusses on the dynamic behaviour of the system; showing what will take place inside the different parts of the system and how they will interact with each other. Below are a set of sequence diagrams to demonstrate processes inside and between different parts of the system when using the two clients. There are then some activity diagrams to show more detail about the processes that take place.

The Web App

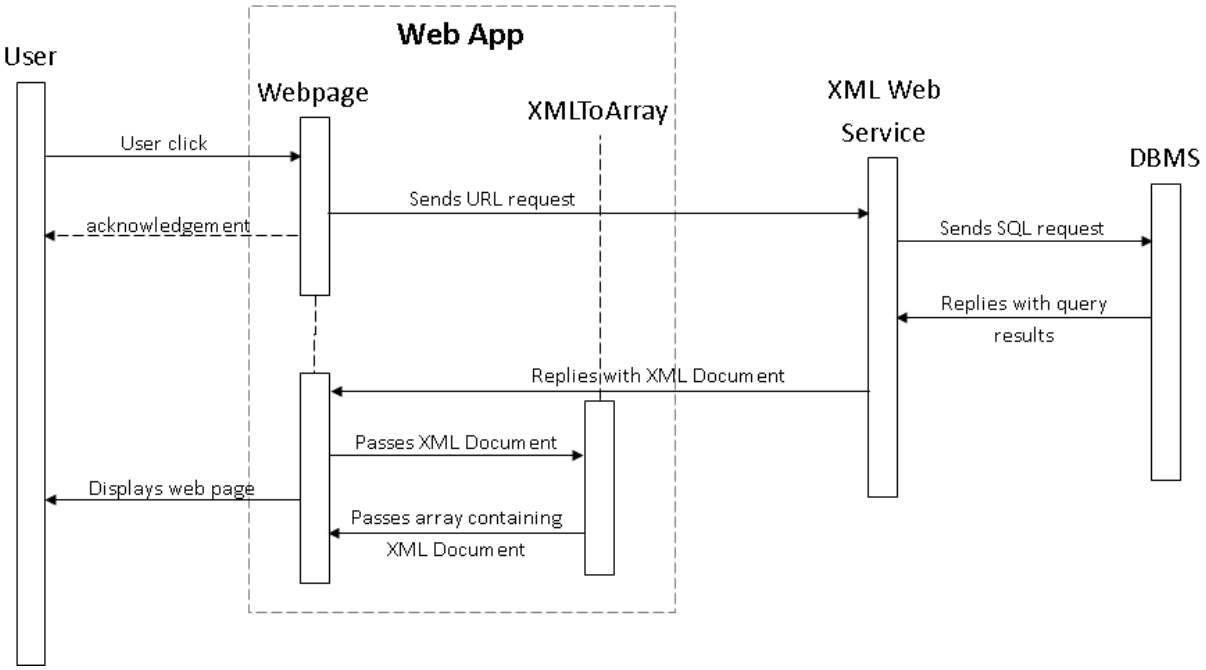


Figure 3.2.1 – Sequence Diagram for the Web App

The webpage takes the inputs of the user which may simply be a click, or parameters entered by the user. The webpage will then either reload, or a new webpage will be opened and the XML Web Service will be used in the process. The webpage will choose from a list of possible requests to send to the XML Web Service in order to perform the task that the user has requested. This request may involve adding, deleting or modifying data from the database. For more details on processes of the XML Web Service please see the corresponding section below.

The XML Web Service produces the XML document that is then returned to the Webpage that requested it. This can return data from the database or just an acknowledgement that a task has been carried out successfully by the database. The webpage then passes the XML document to the 'XMLToArray' file which converts it to an array in the XML structure. The idea behind using the array is that the format is easier to implement as an event feed onto web pages, but can be used to display any data.

The output of the 'XMLToArray' file is then parsed and displayed to the user by the webpage. This may be the loading of a new webpage or delivering feedback to the user to let them know the task they requested has been executed successfully.

BlackBerry App

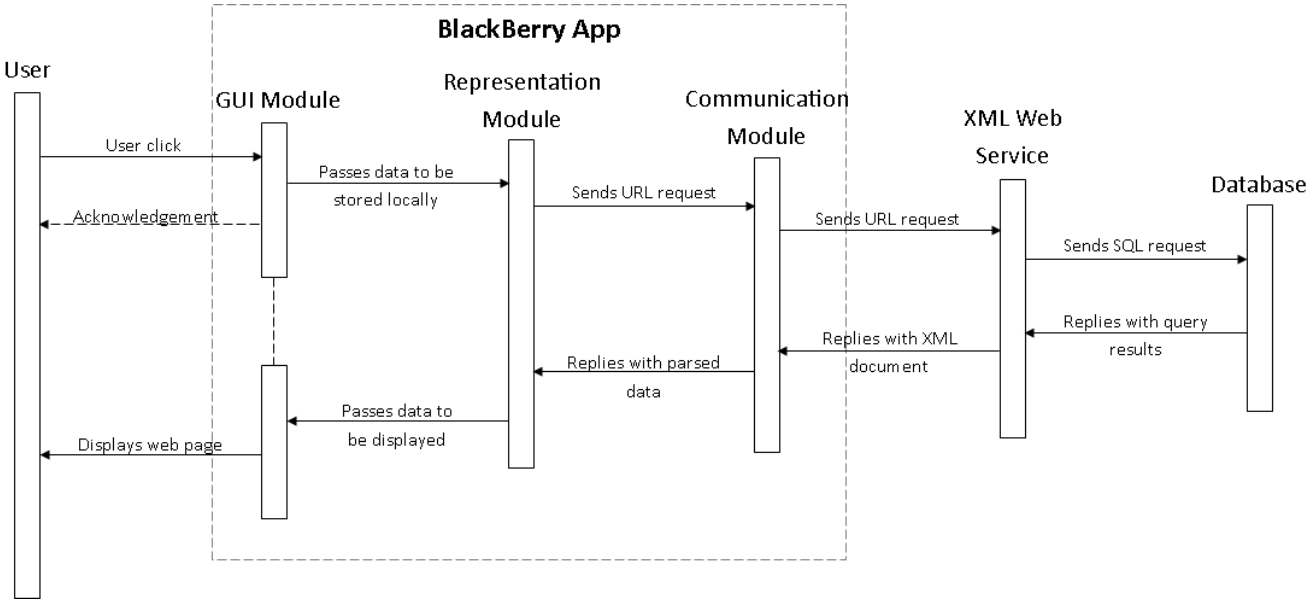


Figure 3.2.2 – Sequence Diagram for the BlackBerry App

THE GUI MODULE

The GUI Module is the class that the user directly interacts with; it controls the user interface. Its purpose is to initiate a sequence of processes based on the user input so that the system performs the task they have requested. Additionally, the user interface must change according between different screens based on user selection. The processes inside this module include allowing users to input data, and then allowing that data to be accessed by the representation module because it is not stored in the GUI module.

THE REPRESENTATION MODULE

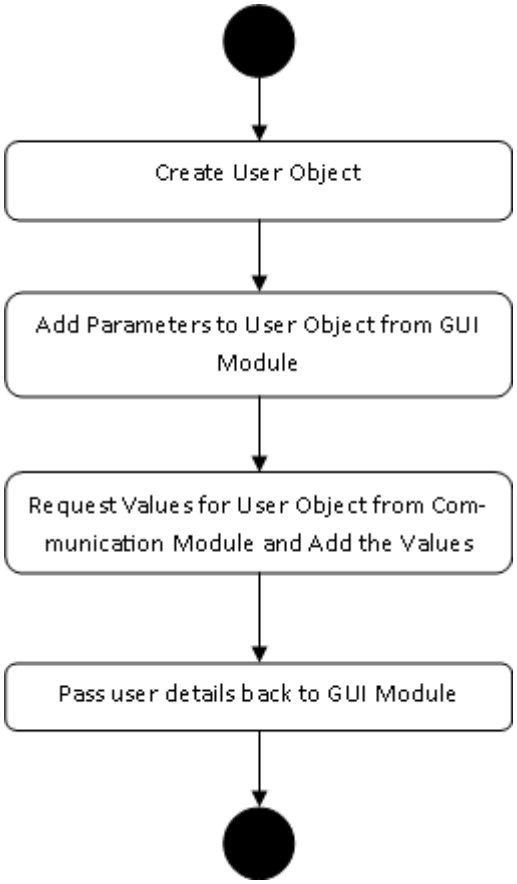


Figure 3.2.3 – Activity Diagram showing the process of when a user logs in

When a user attempts to log in to the system by inputting their user name and password into the GUI, the following process is followed. The user name and password are passed to the representation module, which creates a user object with this information. The user object then requests that the communication module retrieve data about the user; their preferences and details of events they are attending. These details are stored in the user object so that they can be accessed by the GUI module.

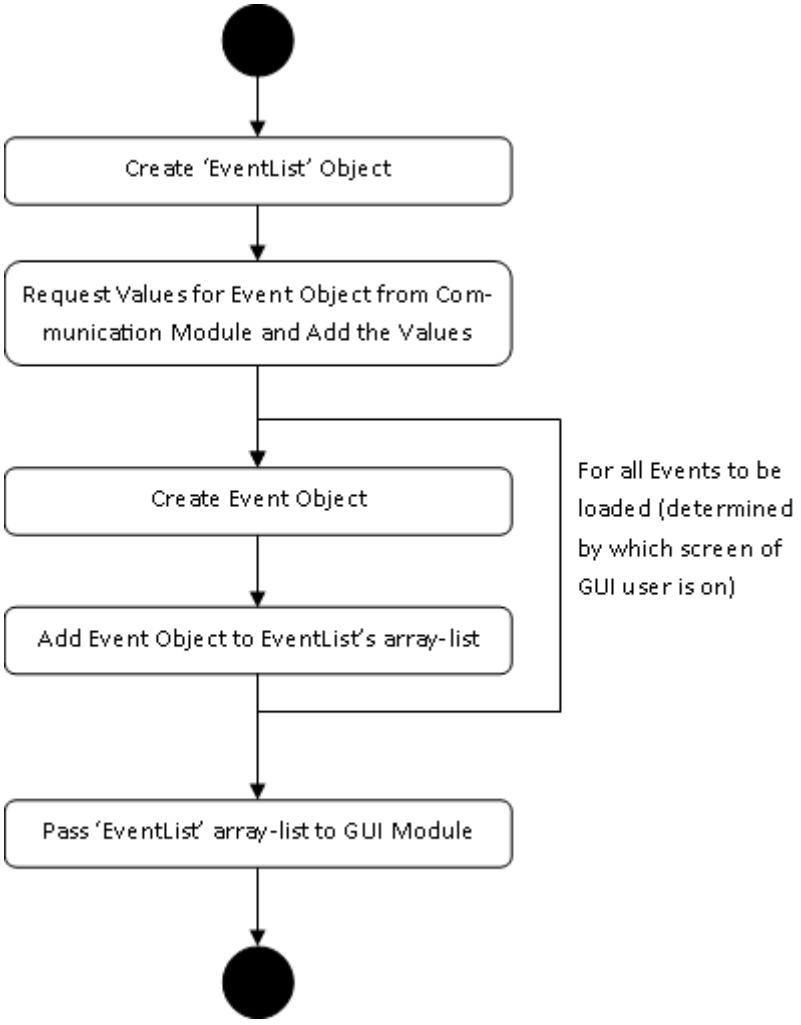


Figure 3.2.4 – Activity Diagram showing the process of when a user access a screen containing event details

The above process takes place when the user navigates to a screen on the BlackBerry app that requires the display of event details; this includes a list of events and a detailed view of one event. When the GUI module opens a new screen on the app, a new 'EventList' object is initialised. A request for the required event data is sent to the communication module – depending on the GUI module, this may be one event or many events. The communication module then replies with the values needed for all the events that are to be displayed. The representation module then recursively places details about each event into a new event object and places it into the 'EventList' array-list. The GUI module can then display this event list.

THE COMMUNICATION MODULE

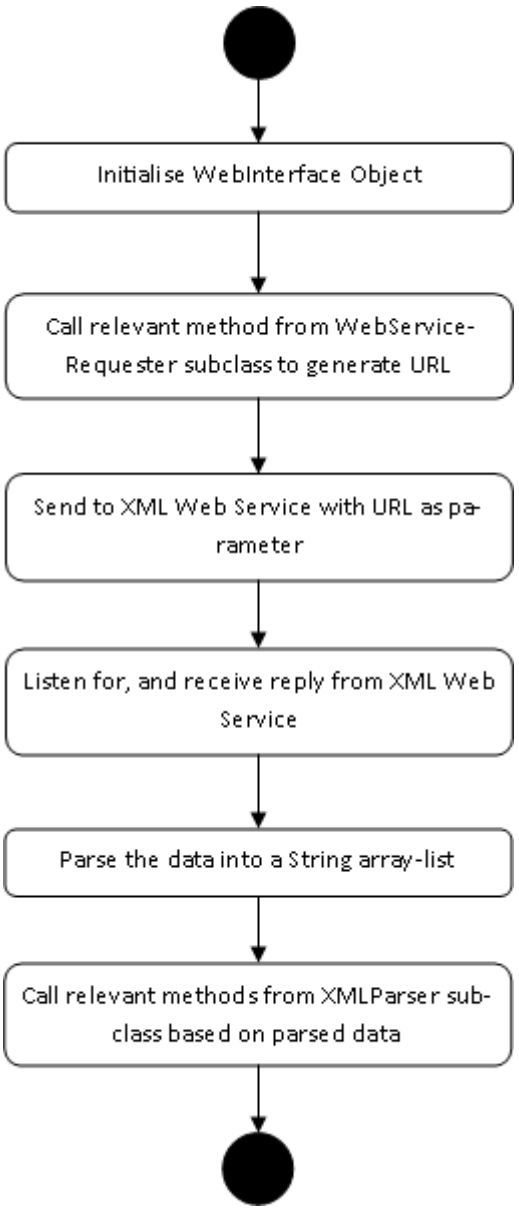


Figure 3.2.5 – Activity Diagram showing the processes of the communication module

Every time the representation module calls the communication module, a new WebInterface object is created. The WebServiceRequester subclass provides methods that create a URL parameter to be sent to the XML web service. The parameter may itself include parameters such as user id, username, password, event comments etc. Where necessary, the methods include these parameters within the URL parameter that they produce (in String format). The WebInterface then sends the URL to the web service and awaits a reply. When it receives a reply, it processes it into a String which is passed to methods in the XMLParser subclass. The data is parsed using a series of if statements, checking for the XML tags and adding a new String object to the stored array-list every time a new tag is found. Based on what is stored in the array-list, the XMLParser subclass will carry out methods to pass the data back to the representation module.

The XML Web Service

The URL requests that are accepted are defined by the functions of the XML Web Service. Depending on the request, the URL will carry any parameters that need to be passed. For example, to return a list of categories based on user id, the URL request would look something like: "[Address of web service]?service=getfeed&user=[UserID]" where the [UserID] is the parameter being passed.

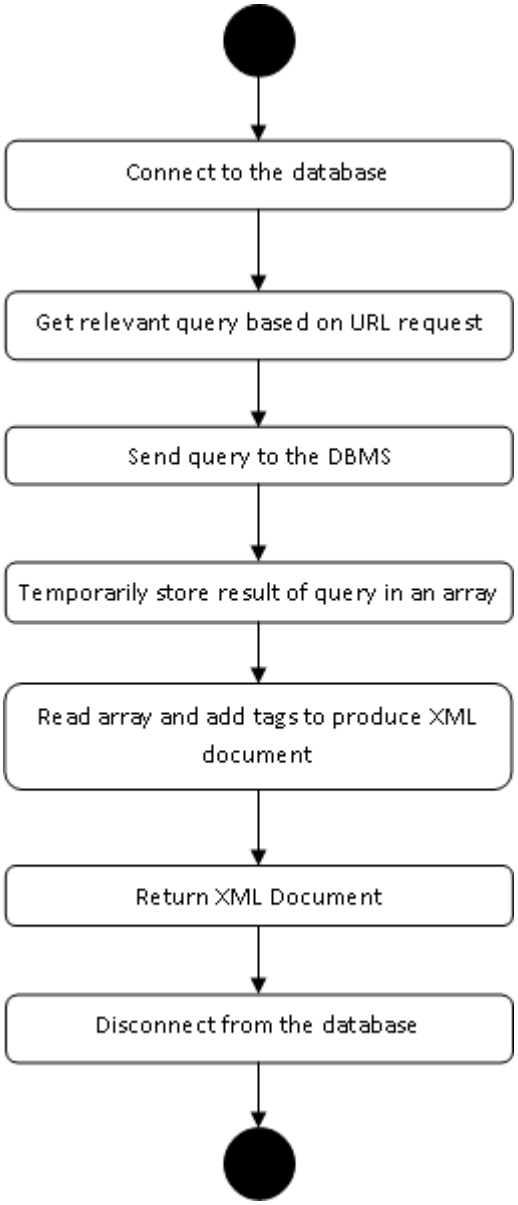


Figure 3.2.6 – Activity Diagram showing the processes of the XML Web Service

The XML Web Service connects to the DBMS using the address of the database and any passwords required to get access to it. The XML web service will store a number of pre-made queries that are selected based on the URL given to the web service. Any parameters needed are added to the query from those provided in the URL. The query is then sent to the DBMS to be executed on the database; the results of the query should then be received by the web service. An array of arrays is created with each record put into a String array. The data is converted to an XML document by adding opening and closing tags to the start and end of each array for each record. The web service then

returns the XML document using PHP's echo function. The XML web service then disconnects from the database, as a new instance of it will be called the next time the clients need to access data from the database.

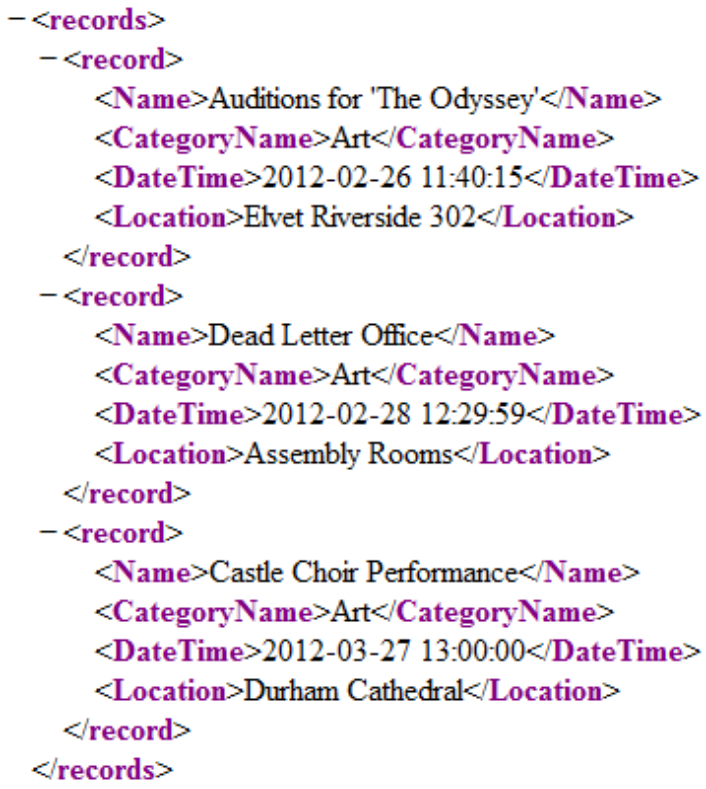


Figure 3.2.7 – Example output of the XML Web Service