Functional Specification

PROJECT BACKGROUND

"Design a web application that uses the Google Maps API to show the distribution of Computer Lab Ring members around the world. Include a secure protocol such that (with mutual consent) two members can exchange contact details when they live within a specified travelling distance, or are visiting the same location. All personal data must otherwise be securely protected, under the control of a database administrator."

Before attempting to implement a solution to the above problem, it is important to conduct research to see whether similar problems have been solved before, and what form the solution to these problems took. Investigation into the Google Maps API was also required to determine how easily it would support our solution to this problem.

Initial investigation into the Google Maps API showed that it provided a number of useful techniques which would aid construction of the system. Specifically, it is possible to draw a marker at any point on the map, and it is possible to convert an address into latitude and longitude, techniques which should enable the location of a Computer Lab Ring member to be entered and displayed with ease.

Showing the distribution of people around the world on a map is a problem that has been tackled before by other websites, for example 'ClustrMaps', which generates maps to show the geographical distribution of hits received by a given website. It would seem sensible to tackle this problem in a similar manner, and display the rough location of each user as a marker on the map in order to show the distribution of the Computer Lab Ring members around the world.

With regard to the ability of members to conduct a secure exchange of details, research into this also brought up websites where a similar scheme had already been implemented. Notably, social networking site Facebook has a system whereby members can choose, on a fine-grained level of detail, which parts of their contact details are viewable by which other members of the site. The ability to exchange contact details with other users is also a feature of websites such as FriendsReunited, and the protection of personal data is a vital part of all internet commerce and banking.

FACILITIES TO BE PROVIDED

The facilities to be provided are outlined in the document 'Requirements Specification', attached to or delivered with this document (and should be considered a subset of this document). Following the paradigm of 'Feature

Driven Development (FDD)', the document takes the form of a list of features, broken down into feature sets which are ranked among three categories:

- *Mandatory* – These features in their entirety constitute the ***acceptance criteria*** for the final product. All must be implemented in order for the product to qualify as complete. Note that under the 'Spiral Model' development process, the specification is revised during each iteration and therefore new mandatory features may be added or old ones removed (with suitable justification).
- *Preferable* – These features would add client-value functionality to the system but are not necessary for it to be usable.
- *Possible* – These features could add some client-value functionality but are of limited use.

In the case of the 'Login' module, it is assumed that the groups database table for usernames, passwords etc is that of the Ring thus simplifying the task of authentication and registration. However, should this be unacceptable to the client, a 'possible' feature set has been specified ('AUTHENTICATION/REGISTRATION') to offer the functionality required to authenticated users against the Ring and for 'Hotel' to exists as a separate entity.

MAJOR SYSTEM COMPONENTS

It is proposed that the system should be split into the following modules:

- Login
- Detail exchange
- Profile
- Event
- Maps
- Friends
- Layout

The web pages that users view will interact with one (or more) of these modules. In turn, the modules will interact with each other, and also with the database. This design leads itself to a three-layered view of how the system will be constructed.

The feature sets (heading from the Requirements Specification document) that each module will be responsible for are outlined below.

Login

The login module provides three basic features:

- Functionality to log a user in.
- Functionality to check if a valid user is logged in.
- Functionality to get the current logged in user.

The login module will often be used by web pages and other modules to check that a valid user is logged in, and if they have permission to access the requested page. It may also be called on to get the current logged in user for a session (eg: for viewing a user's own profile).
This module is not dependent on any other modules.

Detail Exchange

The detail exchange module provides features to facilitate the exchange of user's details:

- Functionality to initiate, complete, reject, or cancel a detail exchange request.
- Functionality to get a list of all pending and sent requests

The detail exchange module handles actions regarding contact detail exchange between two users, and provides information about the current state of any such exchanges in progress.
This module is dependent on the login module.

Profile

The profile module provides the following features:

- Functionality to get any user's limited profile, full profile, or to select one of the two profiles depending on whether the two users are friends or not.
- Functionality to update a user's own profile.
- Functionality to get any user's limited location, full location, or to select one of the two locations as appropriate.

The profile module handles all the necessary functions needed to use profiles. It allows web pages to display relevant profile information and retrieve information from the database. It also provides functions needed by other modules (eg: the Maps module needs to be able to get a user's location to plot a point on the map.)
This module is dependent on the maps module, which is used to discover a user's exact location, and the login module.

Event

The events module provides event related features:

- Functionality to retrieve information about a specific event.
- Functionality to filter events using a specified filter.
- Functionality to add and delete events.

The events module allows web pages to retrieve a (filtered) list of events as required. It also allows the user to interact with events through web pages. The events module is utilised by other modules to provide listings of events (eg: the profiles module may need to find all of a user's events).

This module is dependent on the maps module, which is used to confirm or enter an event's location, the friends module, which is used to find all of a user's friends' events, and the login module.

Map

The maps module provides mapping functions:

- Functionality to create a map, and set its location and zoom level.
- Functionality to create a marker at a specified point on the map.

The maps module is utilised by many of the web pages to display a map in the page's current context. The maps module is mostly passive; it does not need to call functions provided by other modules. It is the responsibility of other modules to get the maps module to plot the correct markers.
This module is partly dependent on the login module (for secure access), but is also used in the publicly available World Map.

Friends – *Feature set 'FRIENDS DIRECTORY'*

The Friends module provides friend-related features:

- Functionality to determine if two users are friends
- List all the friends of user X (by a filter, if requested)
- Search user X's friends by
    - Any accessible (by user X) *Profile* detail
    - Radius of a location (e.g. an event or home location)
- Links to other modules to provide the ability to view a friend of a user's:
    - *Profile*
    - *Events*
- Remove a friend

The Friends module is utilised by many of the other modules to establish the relation between two users. The Friends module is mostly passive; it does not call methods from any other modules (apart from links to view *Profile*s and *Events*). It has dependencies on the following modules:

- Profile
- Events
- Detail Exchange

Layout – *Feature set 'WEB PAGE LAYOUT'*

The layout module provides no functionality, but instead defines how the web pages are laid out. The intention is to use CSS to define behaviours for the HTML tags used in the construction of the website. By using a modular stylesheet system, the design becomes much easier to modify if necessary, and the design is consistent across the whole of the website.

MANAGEMENT STRATEGY

At the outset of the project, the group nominated a ‘Project Manager’ to be the spokesperson and manager of the group. Ostensibly, the group will follow a traditional Spiral Model development process and thus the management strategy will reflect the paradigms of ‘agile teams’.

Rather than nominate group members for certain tasks that they then own to completion, agile teams self-organize based on competencies and skill sets of team members, with work products attributable to the team, not to individuals (collective ownership). The team developing a particular feature (set) will have significant autonomy to make the project management and technical decisions required to get the job done. Planning is kept to a minimum, constrained only by the milestone goals outlined by the group project administrators and Hotel’s own acceptance criteria for the final product.

The team will meet for at least an hour every Monday, Wednesday and Friday (with the potential for more at the request of a group member) for the duration of the project and will utilise IRC interactive messaging and e-mail to ensure rapid communications.

Based on information obtained during these meetings, the team adapts its approach in a way that accomplishes an increment of work. As each day passes, continual self-organization and collaboration move the team toward a completed software increment.

It should be noted that whilst the Project Plan specifies suggested tools for use in unit testing and code documentation, the only mandatory obligations it lays out is that comprehensive unit testing and code documentation occur. The agile nature of the team and its sub-teams allow autonomy in using the best tools for the particular job at a particular time.

Further information concerning the team’s adopted software engineering process can be found in the ‘Process’ section of the ‘Project Plan’.