Software Requirements Specification

for

Yummy Tummy

Version 1.0 approved

Prepared by Ivan Teo, Joel Lim & Daniel Yang

Super Smash Bros

25 Aug 2021

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

Grace Wong	21 Oct 2021	Maximum number of participants in an event is limited to 5 instead of 10	2

1. Introduction

1.1 Purpose

<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.>

1.2 Document Conventions

<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>

1.3 Intended Audience and Reading Suggestions

<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>

1.4 Product Scope

<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>

1.5 References

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

2. Overall Description

2.1 Product Perspective

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram

that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

2.2 Product Functions

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>

2.3 User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

2.4 Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

2.5 Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

2.6 User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

2.7 Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such

as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

3. External Interface Requirements

3.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

3.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

3.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

3.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

4. System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1 Authentication

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

User will sign in and initialize a Yummy Tummy account through Google authentication. This use case is of medium priority.

The website will implement the "Firebase Authentication" using **"Sign In with Google"** to create an account and the website's access features.

Firebase Authentication is used in conjunction with Sign In with Google. This will allow for more versatile modes of registration in the event the website would require more ways of logging in, eg. Facebook or Twitter login; these modes can be implemented quickly, developed and launched in future.

Benefit: 5
Penalty: 5
Cost: 1
Risk: 1

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

1. **User** selects the "*Sign In with Gmail"* button.
2. This opens up a popup where the user chooses the Gmail account they wish to use with the website.
3. On selecting a valid account, the tab is closed and the website is automatically redirected into the private route, allowing the user to accept invites **(Invitee)** and create **(Host)** events.
4. Their user details will also be stored in a **User Document** in **Firebase**, as well as returned to the website for use eg. the displayed name.

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

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

REQ-1: Website must inform **User** in the event that an unexpected error occurs with signing in. (ie. poor internet connection, inability or unsuccessful connection to Google's account database.)

REQ-2: On successful sign in, the system must determine if the gmail account is a first time user. If so, the system shall create a user document with the email address as the identifier. Otherwise, the system will sign the user in as per normal.

4.2 Create event

4.2.1 Description and Priority

The website **Dashboard** will allow **User (Host)** to create an event. **User** will be able to set parameters such as maximum number of participants, the date and time of the event and the title of the event. After which, upon successful creation, the user will be provided with an event link which can be copied and shared through other platforms for dissemination. This use case is of high priority.

Benefit: 10
Penalty: 10
Cost: 1
Risk: 1

4.2.2 Stimulus/Response Sequences

1. User selects the option to "*jio your friends!"*.
2. User enters the title of **Event**.
3. User selects the date and time of **Event**.
4. User chooses the maximum number of participants he is allowing for **Event**.
5. User selects the create option to formally create **Event**.
6. Firebase generates a unique URL for the **User (Host)** to copy to clipboard and share with other participants.
7. Website redirects **User (Host)** to the unique URL to enter the **Event Page**.

4.2.3 Functional Requirements

REQ-1: **User** must be able to input the following fields during an **Event** creation: 1. Title of event (Open ended field) 2. Time & Date (drop-down menu) 3. Maximum number of participants expected (+ button to increment the maximum number and - to decrement maximum number).

REQ-2: The system must display and accept dates only in dd/mm/yyyy hh:mm format.

REQ-3: "*Create*" button will be initially disabled. System checks if all above required fields are filled before enabling the button.
REQ-4: A unique link will be generated for the host to share.
REQ-5: The user must be allowed to create multiple events.
REQ-6: If the **Firebase** is inaccessible (post), the system must display an error message "Server offline" and return the user to the homepage.
REQ-7: The system shall accept up to 5 participants to each event.
REQ-8: "*Save to clipboard"* functionality shall be implemented to save the link onto a local clipboard for future dissemination.

4.3 Accept Event with Location

4.3.1 Description and Priority

The website will allow the **User (Invitee)** to join an **Event**. **User** will have two options to input his location which will be fed into an algorithm to generate the optimal Recommended Eateries: Requesting "Current Location" of user or manually keying user's current postal code. By entering a valid location, the system will update the **Invitee's** status to a **Participant**. This use case is of high priority.

Benefit: 10
Penalty: 10
Cost: 1
Risk: 5

4.3.2 Stimulus/Response Sequences

1. A logged-in **User (Invitee)** accesses the unique URL generated by the **Event** host.
2. **User (Invitee)** is able to view the **Event** Details such as title, date and time, and location (where applicable)
3. **User (Invitee)** will also be given the option to invite more friends with the invite link.
4. **User (Invitee)** selects the option to input location information and join
    a. *"Join with Current Location"*
    b. *"Join with your postal code"*
5. **Invitee** status is updated to **Participant** upon successful entry of their location into the system.

4.3.3 Functional Requirements

REQ-1: The system must be able to prompt users to enable location settings if the *"Join with Current Location"* option is selected.
REQ-2: *"Join with your postal code"* button shall be initially disabled. Upon detecting that the postal code field has been filled with exactly 6 digits, postal code button will be enabled.
REQ-3: Upon clicking the respective *"Join with your postal code"* button, the system must first verify that the postal code is valid (using Google Maps API) and retrieve the longitude and latitude. Else, error message to be displayed "Location Invalid".

REQ-4: Upon clicking the *"Join with your postal code"* button, the system must verify that the current location has been collected successfully. Else, error message to be displayed "Location Invalid".

REQ-5: *"Join Event"* must add the name of the participant to the **Event's** attendance, increments the count of total participating members, and adds the participant's longitude and latitude to the **Event**'s "total longitude" and "total latitude" fields respectively, to the database.

REQ-6: Upon joining the room successfully, the participant's user interface must update accordingly, including removing the join buttons and adding their name to the list of attendees.

REQ-7: If maximum capacity for an **Event** has been reached, a pop-up error message is returned "This event has reached the maximum capacity! Please contact the host!"

REQ-8: If the user is not logged in, a pop-over modal will be shown, prompting the user to sign in with Google before he can join the **Event**.

REQ-9: If the **Event State (expired)**, the user will be redirected to "**Link Expiration Page**".

4.4 Location Based Search for Recommended Eateries

4.4.1 Description and Priority

Returns a list of recommended food locations based on the locations of all participants. An algorithm is used to triangulate and generate the optimal location for all participants. This use case is of the highest priority.

Benefit: 10
Penalty: 10
Cost: 5
Risk: 9

4.4.2 Stimulus/Response Sequences

1. System takes into account the locations of all the participants once the host decides to close the room.
2. Loading screen is displayed as the algorithm triangulates all participants' locations.
3. System returns a list of recommended food locations based on triangulated locations.

4.4.3 Functional Requirements

REQ-1: The host has the ability to kick uninvited users from the **Event**, in the unlikely chance that an unknown user joins the room by randomly guessing the unique URL.

REQ-2: Upon the host clicking search, the system must request the **Event**'s "total latitude", "total longitude fields", and "number of participants" from the database, and display an error if it cannot retrieve this data.

REQ-3: Develop the algorithm to obtain the optimal location between all users. This would be achieved by dividing "total longitude" and "total latitude" by "number of participants". If no food stores are found within a 50m radius of the calculated optimal location, the algorithm will widen the search area with 50m radial increments until 5 food places are found. If more than 5 food places are found within the search radius, randomly return 5 out of the list to the frontend.

REQ-4: Connect the frontend to Google Maps API which will take in the optimal location in format of longitude and latitude.

REQ-5: Specify API request parameters to obtain all the nearest food and beverage locations near the optimal location and return them as a JSON response. (ie. store name, longitude, latitude, review stars, operating hours). Operating hours of food location will be checked if it coincides with the **Event** start time before returning a list of options to the front-end.

4.5 Selecting Location and Confirmation of Event

4.5.1 Description and Priority

User is shown the list of recommended Recommended Eateries based on the locations of all participants. Recommended food locations will show their location, rating, opening hours as well as number of reviews. **User** will also be able to access the food location on Google Maps by clicking on the image. This use case is of high priority.

Benefit: 10
Penalty: 10
Cost: 3
Risk: 6

4.5.2 Stimulus/Response Sequences

1. **User (Host)** selects option to initiate the *search*
2. Upon completion of the algorithm, the system returns a list of recommended food locations based on all the participants' locations.
3. **User (Host)** can view location, rating and operating hours of the food location.
4. **User (Host)** will confirm the location which is to their liking.
5. The **Event Page** will be updated. Users will be shown the host's selection and be able to open in Google Maps.

4.5.3 Functional Requirements

REQ-1: JSON response to be passed to the frontend and content is displayed in a formatted manner for the host to peruse and select the optimal location.

REQ-2: Backend updates that the location of the **Event**has been selected upon confirmation.

REQ-3: Option to redirect users to Google Maps upon **Host's** decision on the meeting point.

4.6 Edit Event

4.1.1 Description and Priority

User (Host) is allowed to edit the **Event**name, starting time, and the maximum number of participants allowed, or delete the entire **Event**. This feature is of medium priority.

Benefit: 6
Penalty: 1
Cost: 1
Risk: 1

4.1.2 Stimulus/Response Sequences

1. **User (Host)** clicks the Edit **Event**icon to reveal a modal to edit the **Event**.
2. If the host wishes to edit the value,
    a. Host will input the updated information into the text fields.
    b. When all changes are made, the host clicks the *"Save"* button and the modal will close when the data is updated successfully.
3. If the host wishes to delete the **Event**, they click the *"Delete Event"* button.
    a. A dialog will prompt them to confirm or cancel their decision.
    b. If confirmed, the modal will close when the **Event**is deleted successfully.
4. If the host does not want to edit the **Event**anymore, they click the cancel button to close the *"Edit Event"* overlay.

4.1.3 Functional Requirements

REQ-1: **User (Host)** must be able to edit the following fields during an **Event** editing: 1. Title of event (Open ended field) 2. Time & Date (drop-down menu) 3. Maximum number of participants expected (drop-down menu).
REQ-2: The system must display and accept dates only in dd/mm/yyyy hh:mm format.
REQ-3: *"Save"* button will be initially disabled. System checks if all above required fields are filled before enabling the button.
REQ-4: If the **Firebase** is inaccessible or empty, the system must display an error message "Server offline" and return the user to the homepage.
REQ-5: The system should accept up to 5 participants to each **Event**.
REQ-6: In the case of an error with updating **Firebase**, a dialog will be shown.
REQ-7: Only **User (Host)** is able to edit any fields after **Event** creation.

4.7 Sign Out

4.1.1 Description and Priority

Signs the current **User** out of the system. This use case is of low priority.
Benefit: 1
Penalty: 4
Cost: 1
Risk: 1

4.1.2 Stimulus/Response Sequences

1. User clicks the sign out button.
2. User is redirected to the Landing Page on successful log out.

4.1.3 Functional Requirements

REQ-1: System should automatically redirect users to Landing Page.
REQ-2: Frontend should recognize there is no user logged in and deny access to any private routes.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

5.1.1 The website user interface should provide a response within 3 seconds on clicking the “Sign In with Google” button to ensure this step does not hinder the user’s adoption of the website.

5.1.2 Location retrieval should happen within 5 seconds from clicking.

5.1.3 Search results for location based restaurant recommendations should appear within 10 seconds.

5.1.4 Clears unconfirmed events automatically in the **Events Document** and **Dashboard** (ie. if the host does not initiate “*Search”* before the event timing) to free up **Firebase** and declutter the **Dashboard**.

5.2 Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>

5.2.1 System will update the maximum size of gatherings according to the latest COVID-19 restrictions.

5.2.2 The Curated Locations page that displays the location the host decided on, can only be accessed with the accounts invited to the **Event**.

5.2.3 A notification to request for the user's permission to use their current location before any location detail is retrieved via Location Services when *"Join With Current Location"* is selected.

5.3 Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

5.3.1 User's email and location will not be revealed between users for privacy concerns.

5.3.2 Location data will only be retrieved once, when clicking on *"Join With Current Location"*.

5.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, ***reusability****, robustness, testability, and* ***usability****. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>*

5.4.1 Using a navigation bar in the website with dedicated space for at least 5 tabs to allow for better scalability in the event more features are added.

5.4.2 Users must be signed into an account to interact with the system. This is to prevent a single user from joining an **Event**more than once, affecting the midpoint of all locations.

5.4.3 Upon logging in, the user's account and credentials should be saved in the browser's local storage to prevent unnecessary logging in every time the user wishes to use the website.

5.4.4 The website should have a responsive design, so as to accommodate for use on phones and tablets, which would be quite common given the purpose of the app.

5.4.5 At least 95% of users should understand how to create and join an **Event** within 2 minutes on their first visit to the website.

5.5 Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

5.5.1 All users need a valid Google Account and be signed in while using the app.
5.5.2 Actors can perform the following functions based on their assignment.

User	**Actions**
Host	- Create **Event** - Edit **Event** - View **Event** Page - Account Authentication - Location Based Search for Recommended Eateries - Selecting Location and Confirmation of **Event** - Sign Out
Invitee	- Account Authentication - Accept **Event** with Location - View **Event** Page - Sign Out
Participant	- Account Authentication - View **Event** Page - Sign Out

6. Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Data DictionaryEv

Term	Definition
Actors	
User	A host or participant or invitee who uses the website with an account.
Host	A person who creates an event with their account and invites their friends.
Invitee	A participant who has been invited to the event
Participant	A person who joins and accepts an event that was not created by themselves.
Firebase	The backend database and server that will be used in this project to create, read, update and delete data.
Event Descriptions	
Event	The plans which a user can create and invite friends to on the website. It includes a title, starting time, eating place, and names of attendees.
Event Page	The page that displays all relevant Event Information.
Location	The location data which stores the longitude, latitude and geohash of the user or eating place.
Recommended Eateries	A restaurant, food stall or hawker centre that users will be recommended upon search.
Initiate Location Search	Host has finalized the user list, invitees can no longer join the event once the search function has been initiated.
Event State	The event state post search initialisation - 1. The host has begun the search but yet to select the finalised location. 2. The host has selected and finalised the location.
Event State (Locked)	Once host has begun the search - invitees can no longer join the event
Event State (Open)	Event location and participants have been confirmed and all participants are awaiting for the event to occur.
Event State (Expired)	If the event's scheduled time has passed, the event is moved to Previous Events
User Document	JSON file storing the latest instance of the user’s events and actions on the website.

Event Document	JSON file storing the latest instance of the events and event details (ie. users, time/date, location) on the website.
Private Route	Route that requires Authentication Token to access.
Public Route	Route that anyone can access.
Dashboard	Main Page that displayed Upcoming Events, Previous Events and "*jio your friends!"* button
Link Expiration Page	Page that will be displayed when a User tries to access the Event URL of an expired event.

Team	**Term**	**Data type**	**Dimension / Measure**	**Example**	**Source of truth**	**Notes**
marketing	total_spend	number	measure	1,200,000	ERP	
marketing	utm_campaign	string	dimension	"Social buttons"	GA	https://analytics.google.com/analytics/web/#/report-home/1234567890
...						

Name	Data Type	Format	Definition
User Collection	Collection	NIL	The collection of User objects (documents) to be stored on Firebase.
User ID	String	NIL	A randomly generated unique identifier for each User object.

User	Document / JSON Object	User ID : {
		+ name(String)
		- eventsID([String])
		}
			A host or participant or invitee who uses the website with an account, to be stored on Firebase as a document within User collections.

Event Collection	Collection	NIL	The collection of Event objects (documents) to be stored on Firebase.
Event ID	String	"<Adjective>-<Noun>-<2 Digit Number>"	A randomly generated unique identifier for each Event object that is human readable.
Events ID	[String]	NIL	A list of Event IDs that a user has

			joined or created, regardless of Event Status, to be displayed on the landing page.

Event	Document / JSON Object	Event ID : {
		- Total Coordinate (Location)
		+ Status (Int)
		+ Start Time (Date)
		+ Event Title (String)
		+ Max number of participants (Int)
		+ Host ID (String)
		+ Participants ID ([String])
		+ Selected Eatery(String)
		+ Recommended Eateries([String])
		}
			The event document that will be stored on Firebase within the Event Collection.

Start Time	Date	HH:MM DD/MM/YYYY	The time in which the event is set to start. If a location has not been selected past this time, the event will be deleted. If a location has been decided, the event's status will be set to 2 (expired).
Location	Object	- lng (double) - lat (double)	'lat' and 'lng' are the variable names for the latitude and longitude of the location coordinates respectively.
Status	Int	0 == Open 1 == Locked 2 == Expired	The status property for the event's state.
Place ID	String	NIL	The unique identifier returned by Google Places API which corresponds to a single eatery.
Recommended Eateries	[String]	NIL	String of Place ID's corresponding to the eateries recommended to the host.
Selected Eatery	String	NIL	Place ID corresponding to the eatery that the host selected.
Host ID	String	NIL	The User ID which belongs to the host of the event.
Participants ID	[String]	NIL	A list of User IDs corresponding to the participants who joined the

			room.

Eatery View Model	JSON Object	{
		rating: Double,
		address: String
		reviews: Int,
		opening hours: String,
		location: Location,
		postal code: Int,
		photo reference: String,
		}
			The JSON Object that contains only the required information to be displayed on the webpage. The data retrieved from Google Places API will be transformed into this View Model for more efficient use of space.
			
			Reference: https://developers.google.com/maps/documentation/places/web-service/details#PlaceDetailsResponses

Photo Reference	String	NIL	The unique identifier of a photo of the Eatery which can be queried using Google Places Photos API to retrieve the image. Reference: https://developers.google.com/maps/documentation/places/web-service/photos

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>

1. Allow users to vote for the place to eat, rather than let the host choose solely.
2. Allow random users to join the room, to meet new friends.

Source: http://www.frontiernet.net/~kwiegers/process_assets/srs_template.doc