**Request/Response Design**

**Common points to be checked before sending a request to client: Profile type – Couple/Single? Categories – Changed? Location – Changed?**

1. **Home Page**
2. **Registered user**

When User Logs in/Changes location: Depending upon his home location and selected favourite categories the Home page should be populated when a user enters.

Assumptions: 1. Default location is stored in DB. 2. Favourite categories are stored in the DB.

Request: HTTP POST/GUID/{Location}

Server Action: Run a SQL to pick up events (for a 7 day period from today)

SQL Criteria: Event Categories = User favourite categories

Even Location = {Location}

Client: Read the results in an Input Stream buffer and use it appropriately.

1. **Event Details:**
2. When user clicks on an event displayed on the Home page under any category.

Client Request: HTTPPOST/EventId/EventDetails

Server Action: Run a SQl to get all the event details for the particular event id.

Client Action: Read the results in an Input Stream buffer and use it to display appropriately.

1. When user clicks on Join button:

Client Request: HTTPPOST/GUID/Eventid/JoinEvent

Server Action:

1. Run a SQL to update the Event Id with the participant GUID, increase the number of participants.
2. Update the participant Events table with the Event Id.
3. Also generate a message to the Event creator (Insert a Row in the Message table) regarding the participation.
4. When User enters a message and clicks on Query button:

Client Request: HTTPPOST/GUID/Eventid/EventQuery

Server Action: 1.Generate a message with the Query description to the Event Generator. 2. Add a row to the Interested Events for the participant?

1. **Profile Page**
2. When user click on the Profile item on the Side menu and enters profile page:

Request: HTTPPOST/GUID/FETCHPROFILE

Server Action: Run a SQL to get the details to populate the profile Page.

Client: Read the results in an Input Stream buffer and use it appropriately.

1. After changes to the profile – User clicks on save

Request: HTTPPOST/{GUID)/UpdateProfile

Server Action: Read the changes sent by the client and run an update SQL to update the changes in the DB where Guid = Guid from Client

1. **Categories Page**
2. When user click on the Categories item on the Side menu and enters profile page:

Assumption: All categories are stored initially in Clients DB when the client logs in.

Request: HTTPPOST/{Guid}/SelectCategories

Server Action: Run a SQL to get the Categories marked by client

SQL Criteria: Categories = Client Categories

Client Response: Read the client’s favourite categories and sub categories, mark them in the Categories page.

1. Changes to categories selection by user and saves:

Request: HTTPPOST/{GUID}/UpdateCategories

Server Action: Read the category changes and update the DB with an Update/Insert SQL accordingly to insert/update categories.

SQL Criteria: GUID = {GUID}

1. **Create Events Page**
2. When user clicks on the create event item on the side menu.

Request: HTTPPOST /GUID/GetCategories

Server Action: Run a SQL to get all the User favourite categories and sub-sub categories for GUID = {GUID}

Client: Read the response and use the data for appropriate fields on the page.

Location field should be populated with the location that’s on the client currently.

1. When user finishes entering details and clicks on the Create event button.

Request: HTTPPOST/GUID/CreateEvent

Server Action: Run an Insert SQL to create an Event

SQL Criteria: Location = {location}, GUID = {GUID}, Category = {Category, Sub category1, Sub Category2..}

Client: Read the Server response code and display ‘Event created successfully’ as per the return code.

C. Update Event: When a user opens the event he created, changes and saves (This leads to the question of List of Events created).

User can change dates, participants, Description, (Location?).

Request: HTTPPOST/EventId/UpdateEvent

Server Action: Run an Update SQL to update the user event with the details sent by client. SQL Criteria – Eventid = {EventId}

1. **Joined Events**
2. When user clicks on the Joined Events from the side menu.

Request: HTTPPOST/GUID/JoinedEvents

Server Action: Run q SQL to get the list of user joined events

Client: Read the Response from the server and list them accordingly.

If there is only one event then it should be displayed like in the 2.Event Details

Otherwise list them on the page and then the user clicks on them it should open the 2.Event Details (the HTTP Post Request is the same)

1. Cancel Participation – (The user should be allowed to cancel the participation until 24 hours before the event starts?) When the user clicks on the cancel participation button

Request: HTTPPOST/GUID/EventId/CancelNomination

Server: Run a SQL to Update the Participation number of the event

Also Run another SQL to remove the Event from the User.

Client Action: Read the Response code from server and update the user accordingly with an ‘Event participation cancelled successfully’ message or other error messages if the return code is not successful.

1. **Inbox:**
2. **Couples Profile:**