**Gift Manager Module:**

* On search request, first need to check if user uploaded at least three gifts.
* Searching implementation:
  + Client sends following parameters:
    - age (integer)
    - **gender (‘M’ or ‘F’)**
    - **relationship (**
    - **price range**
  + First search DB by gender, age (filter by age ranges) and price range (optional). Retrieve 50 results ordered by relevance according to relationship matrix and then by the number of likes.
  + Response to client:
    - json of object of gifts. each gift will contain:
    - description (gift name etc.)
    - **uploading user (user\_id??)**
    - **age (integer)**
    - **relationship**

**choices:**

**‘Parent’**

**‘Grandparent’**

**‘Sibling’**

**‘Cousin’**

**‘Parent in low’**

**‘Nephew’**

**‘Friend’**

**‘Partner’**

**‘Child’**

**‘Child\_in\_law’**

**‘Grandparent\_in\_law’**

**‘Sibling\_in\_Law’**

**‘Acquaintant’**

**‘Colleage’**

**‘Grandson’**

**‘Uncle/Aunt’**

* + - **price**
    - **image\_url (optional)**
    - **uploading time**
    - **gift rank**
* Upload a new gift:
  + Client sends following parameters:
    - age (integer)
    - **gender (‘M’ or ‘F’)**
    - **relationship**

**choices:**

**‘Parent’**

**‘Grandparent’**

**‘Sibling’**

**‘Cousin’**

**‘Parent in low’**

**‘Nephew’**

**‘Friend’**

**‘Partner’**

**‘Child’**

**‘Child\_in\_law’**

**‘Grandparent\_in\_law’**

**‘Sibling\_in\_Law’**

**‘Acquaintant’**

**‘College’**

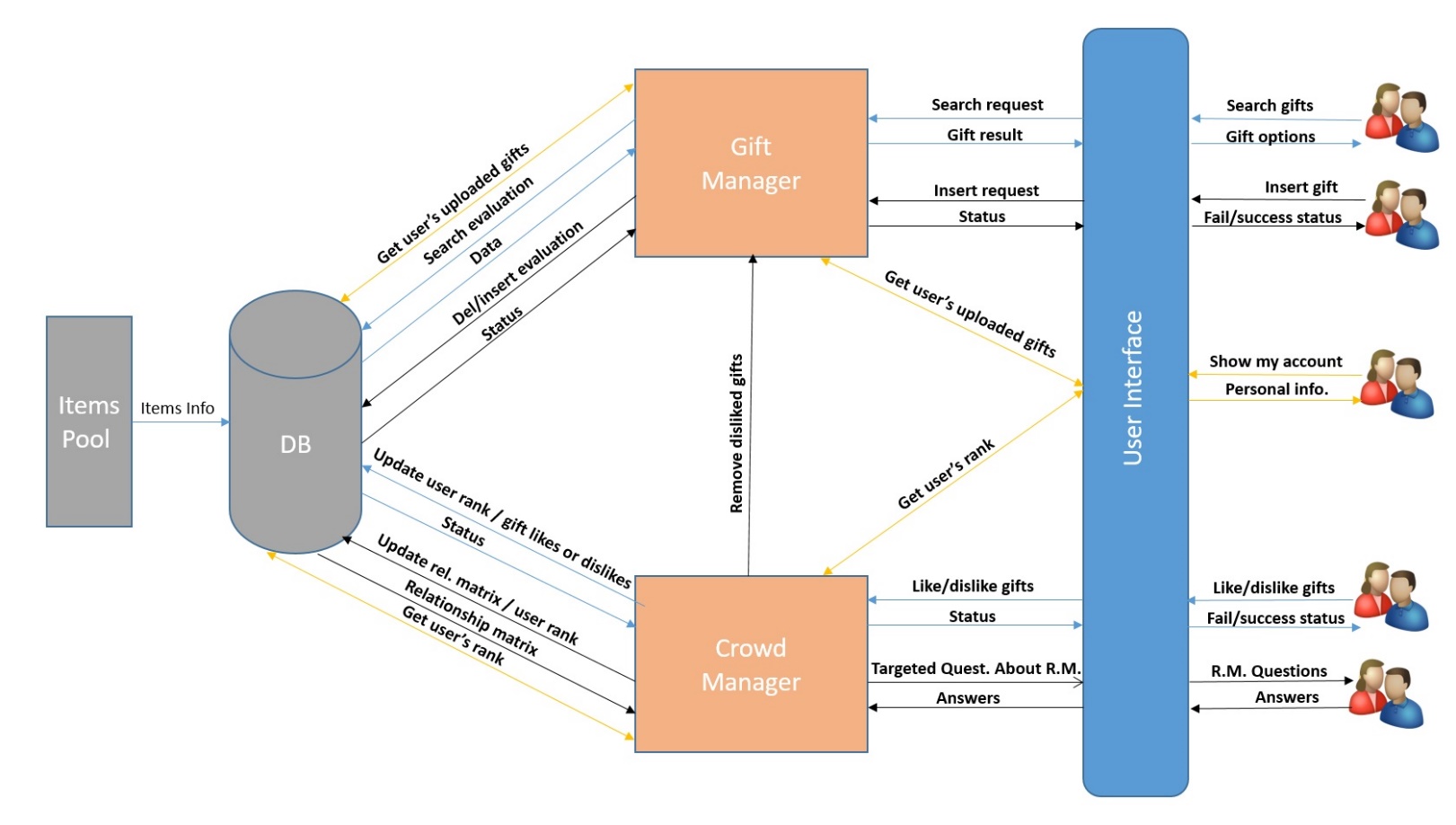
**‘Grandson’**

**‘Uncle/Aunt’**

* + - **price**
    - **image\_url (optional)**
    - **user id**
    - **description**
    - **relationship2 (for crowdsourcing question)**
    - **connection strength (between relationships)**
  + Server Response to client:
    - Status: Fail/success?
  + Insert relevant information that the user supplied to the DB.
  + Increase user rank (add 3 points).
  + Return status to the user (Fail if not logged in for example or Success).
* Edit gift:
  + Let user edit his gift if and only if the gift was added at most 1 hour ago (save timestamp for each gift).
  + Send an edited query to the DB.
  + Return status to the user (Fail if too much time passed since uploading for example).
* **Logging Module:**
* Check if user is not banned.
* Implement logging via Google API (?)
* **Crowd Manager Module:**
* User may like/dislike other gifts if his rank is high enough (5 points). It affects the uploader rank appropriately, for every like user gets 1 point.
* Spammer identification:
  + Under gift rank of -5, the gift will be removed from the DB.
  + If more than 3 gifts of a user were removed he will be blocked for 48 hours.
  + When the user will be presented with R.M. questions we may ask questions that we know the answer for and check if the user's answers are credible (optional).
  + Add spam button for every gift (optional).
* Present user with about 10 R.M. questions (for example would you buy a specific gift to your mom) and grant him 1 point for every answer, save counter for answered questions per user.
* Retrieves user rank when needed.
* Process answers for R.M. questions and strengthen the relations in R.M. when possible.
* **Liking a gift:**
  + Client sends following parameters:
    - user\_id
    - like: (1\-1)
    - gift\_id
  + Response to client:
    - Status: Fail/Success
* **Profile Page:**
  + Client sends following parameters:
    - user\_id
  + Response to client:
    - Json object of gifts objects

Each gift contains:

* + - * + **price (integer)**
        + **image\_url (optional)**
        + **user id**
        + **description (string)**
        + **gift rank (integer)**
    - Status: Fail/Success

****