**DATA STRUCTURE**

**mongoDB Schema**

1. What is the \_\_v key that exists in every collection?

[GVB]: “\_\_v” (version number key) is attribute added by mongo driver by default in the document. It is required for maintaining the versions of a document so that mongoose internaly can prevent the consistency of document data in concurrent requests.

1. What is the purpose of a counter or index in addition to \_id that is the unique primary key for the document autogenerated by mongodb? Is this typical? (applies to all keys with the id first letter of the customer/brand/location/employee collection as the first letter and id as the suffix: cid, bid, lid, eid, uid, fid) How does this counter get set and updated? Noted that the identitycounters collection stores current count for each of these fields. Also noted that it is not an index within a particular customer, or brand e.g. bid does not get reset to 1 for each new customer, but instead bid is unique across all customers; how does it get modified when one of the brands is removed? If we need to keep this id, should we call it index to distinguish from the mongodb generated \_id?

[GVB]: This id I added it earlier.. but I guess this is not required now. I added this earlier with intention to have traditional id pattern followed in relational database. However, I have not used it further execpt storing them and instead using mongo defined ID for optimized indexing performance. I guess this Id’s are not required now, which we can remove now.

* 1. In identitycounters, there are two models called “User” and “users”. Assuming “User” with field=”uid” and count=1 is old data – to be confirmed

[GVB]: Since, we will be anyways removing id’s now. And, yes User was old data.

1. Can we have consistent naming? Is the convention to use firstword\_secondword… for collection names that have two or more words? Inside a collection, is the first letter of second word capitalized, and so on, for multiword keys? Also, typically the full word is spelled out vs. abbreviations where possible unless it is convention e.g use “id” for identifier. Assuming these conventions, requesting the changes below.
   1. Change add to address
   2. Change contactNo to phone
   3. Some collections with two words have underscore separating first and second word e.g. feedback\_histories; some don’t e.g. identitycounters; change identitycounters to identity\_counters
   4. Change feedbacks collection to say feedback as feedback plural is feedback
   5. In collection employees for fullName key, change fname to fullName
   6. In collection employees for fullName key, change lname to lastName
   7. In collection employees, change startDt to startDate
   8. In collection employees, change phoneNo to phone
   9. In collection employee\_histories, change empUniqId to empUniqueID *(or see point (j)(v) below for alternative approach)*, and change newVal to newValue, oldVal to oldValue
   10. In the following collections where the key refers to the \_id of the parent that the document is child of, should we rename the key to have Id suffix?
       1. In brands collection, change customer to customerId
       2. In locations collection, change brand to brandId
       3. In employees collection, brandId and locationId used correctly (no change)
       4. To prevent confusion with mongodb generated Ids, and employer provided Id, in employees collection, should we rename employeeId to be employerIssuedId?
       5. In employee\_histories collection, change empUniqId to employeeId, and change the current employeeId to be employerIssuedId (see above point iv. for rationale)
       6. In beacons collection, change customer to customerid, brand to brandId, location to locationId, and employeeid to employerIssuedId. Is there a reason why we also don’t include the employeeId i.e. the \_id generated for the employee linked to this beacon?
       7. In brand\_roles collection, brandId used correctly (no change)
       8. In users and guest\_users collections, change role\_id to roleId
   11. In users collection, should we rename fbUniqId to facebookProfileId or facebookUsername? Not sure which information from facebook are we provided through the login API (<https://www.facebook.com/facebookUsername> or <https://www.facebook.com/profile/picture/view/?profile_id=facebookProfileId)>
   12. In users collection, should we rename fbUniqId to facebookProfileId or facebookUsername?
   13. In users collection, change phoneNo to phone, userName to username
   14. Should the collections guest\_users and users be renamed to guest\_consumers and consumers? It is confusing to know who are we referring to, by users? Because the database stores many different users’ information – could be admins with varying level of access, consumers, employees, third-party apps that will have different credentials
   15. In user\_roles, change role\_type to roleType
   16. In brands collection, change ratingImgId to ratingId as it points to the rating document inside ratings collection, not a particular image
   17. In feedbacks collection, change ratingVal to ratingValue

[GVB] We just need to accept this and do the changes.

1. What are isActive and isDeleted keys? Would like to understand how these work when say a brand is deleted; can we make it inactive but keep in database? Is this how it works if an employee leaves the location (and brand/customer)?

[GVB]: Yes mythilli. We have isActive and isDeleted attributes for maintaining the state of the entities. Like, if we employee leaves the job; he will be marked isDeleted:1 but the record will persist. Same for all other entitites.

IsActive is used for temporarily deactivating the profile and still be visible deactivated profile but not deleting it.

1. In employee collection, for each employee document, we already have brandId and locationId employee belongs to. Can we also add the customerId? Just wondering what happens if a brand gets sold to a new customer.

[GVB]: Already Done..I too thought the same and so all the new employees will contain the data.

1. In employee\_histories collection, why do you have both empAttribute and key? For e.g. empAttribute stored “fname” and key stored “First Name”. Is it necessary to have the latter key inside this document?

[GVB]: key suggests what exactly that attribute corresponds to in a readable format and we display key in the table on UI. So, key and empAttribute both are needed.

1. In employee\_histories collection, what do createdAt and updatedAt timestamps hold? If empAttribute of the empUniqId is edited first time, assume document gets created first time and createdAt is filled in. Then if same empAttribute of the same empUniqId is edited again, does (a) the same document get edited with a new value, and the updatedAt timestamp is added, or (b) a new document gets created?

[GVB]: In case of same empAttribute of the same empUniqId is edited, then a new document will be added and not updated since we are maintaining history records. And, I think updatedAt are not really required because history records are read-only and will not be modified.

1. How are edits handled vs. role changes, for employee? Should we add actions for the following, to the admin portal? *Note: information below is on portal functionality. It may also have a bearing on the underlying data structure i.e. what to capture in employee history.*
   1. “New role” *(helper text – click here if employee’s role has changed, such as when the employee is promoted or moved into a different role within the same location)*. This will gather inputs when last role ended (defaults to today), when new role started (defaults to today), and new role details

[SM]: This is already built in a current implementation.

* 1. “New location” *(helper text – click here if employee is moving to a different location. You will be able to update the employee’s details e.g. role, beacon Id etc. by going to that location’s employees list in the portal. Note: location should already be in the database. The admin for the location will be prompted to add the employee to their list of employees)* This will ask if they’ve moved locations; only if the user has access to the new location in the db, can they enter the particulars into the new location db. Otherwise, they would simply mark the employee as handed off to a new location and that employee should no longer show up in this location’s employees list. If the user has access to the new location in the db, there should be a notification in the new location list that an employee has appeared but not added to the list for this location yet. It’s like they are in a “parking lot” of sorts. In order to add them to this new location, the user who has access to this location has to “add” the employee and confirm the details in the employee record (may verify role, start date, beaconId etc.)

[SM]: Understood the requirement. Currently, it is not supported in the portal and it is an involving task that can have an impact on delivery dates. Can we do it post pilot? When we are building a transfer of an employee, probably we should also build functionality that allow an employee crossing over a brand or a customer.

* 1. “Terminate position” *(helper text – click here if the employee is no longer with the location and the company)* This will gather end date and take off the employee from list of employees from this location. But employee will still be in employees database. If the employee is recreated at a later stage at a same or different customer/brand/location, they can be linked together into one employee based on their email or phone (unique ID for employee). That will then link the history to the employee record. However, the new customer/brand/location cannot have access to the old records of the employee if they were part of a different customer/brand/location.

[SM]: It can be built along with 'b'. Again, I will suggest we take it up post pilot.

* 1. “Deactivate” *(helper text – click here if the employee is no longer part of the Applause feedback program, but is still an employee at your company)* –This means employee still there, but no longer in front-of-house role and not part of applause feedback system.

[SM]: This is not currently available, but can be built easily.

Unclear how the above states are being stored in employee\_histories, and how these actual changes are being tracked vs. changes that are profile edits (e.g. updated misspelled name, updated last name because they got married, changed profile image, changed personalization custom text). Do we need histories for the latter? (think not)

[GVB]: Currently, I have captured details for profile edits. However, if you think it is not required for application perspective we will clear capturing of those information and will think on how to capture following events - ‘relocation’, ‘termination’, ‘deactivation’, ‘deletion’. If you think more, you can please append in the list or mail it seperately.

1. What is captured in a guest\_user document? What are appVer (version of Applause app they are using?) and role\_id?. Is there a unique device ID handle that the OS can provide so that we can trace multiple guest user feedbacks to a single device?

[GVB]: ‘guest\_user’ document captures all the guest user details; users who are using our app without login. ‘appVer’ corresponds to applause application version. ‘role\_id’ corresponds to id in user\_roles collection created to define system roles for users.

1. What are appVer (version of Applause app they are using?) and role\_id? Is userName required? Is it not sufficient to look at regType and look at the email, phone or facebookId in the same document for access? If a user changes their email or phone number, can we still keep their old feedback tied to their account (assume it is tied by user’s \_id and not the email or phone, but want to check how it works).

[GVB]: ‘appVer’ corresponds to applause application version. ‘role\_id’ corresponds to id in user\_roles collection created to define system roles for users. Umm, username not really required and yes we can lookup over regType and email/ phoneNo. Old feedbacks are tied to user ids and not to usernames; so, yes feedback will still be maintained even if user has changed their email / phone numbers.

1. Explain how user\_roles collection is used. What is meant to be included in accesslist?

[GVB]: ‘user\_roles’ collections captures the different roles who will be accessing our system or application. For example, user role ‘END\_CONSUMER’ is used by the third party applications for accessing our API’s. Access-List will define the paths accessible to this user role.

1. How is the img key used in ratings collection? Where do the images come for the rating scale (1 to 5)? And how is the logic determined whether giving a rating of x (on a scale of 1 to count) means showing selected images for 1 to x, or only showing the selected image for x. For example, if stars -- you highlight 1, 2, and 3 stars for a rating of 3; if smiley – you highlight only the 3rd image for a rating of 3 ? can we put a flag to say cumulative (1 or 0)? cumulative=1 would be any rating that has a behavior similar to stars; cumulative=0 would be any rating that has a behavior similar to smileys. So that way if stars changed to flowers, it won’t matter and it is not tied to a particular image. Also, if count changed to 7 (because someone decided they want a 7-pt scale vs. 5) it would still work.

[SM]: Currently, images are stored in the application bundle and are not supplied dynamically. So in a way, they are hardcoded. Let's discuss if you want to add these.

1. Can the consumer edit the feedback they give to an employee, after it is submitted? Would not want that functionality. In the feedbacks collection there is a updatedAt key with value as timestamp. How would that be used?

[SM]: No, consumer can not edit the feedback they give to an employee. updatedAt timestamp is currently used on "My feedback history" screen. Currently we show only one time which indicated last feedback given for any employee of a given location.

1. In the feedbacks collection, there is a user.role\_id key inside user object key. Is this the role\_id from the document in the users collection? How is it used?

[GVB]: Actually, we have defined this role\_id for a purpose needed earlier and I guess we have sorted out byusing another approach. So, this wont be required now. However, the ‘role\_id’ refers to ‘user\_roles’ collection.

1. Since feedbackReasons is captured for each of the brand’s roles in brand\_roles collection, and defaults to whatever is in defaultReasons available for the brand in brands collection, for any role that does not appear in the brand\_roles collection, assume we don’t need the collection feedbackreasons anymore? If so, latter to be deleted.

[[GVB]: For any role that does not appear in the brand\_roles collection, it will default to defaultReasons.

Also, the collection ‘feedbackReasons’ was used earlier and can be safely removed now.

1. VERY IMPORTANT: We need to capture data on interactions – how does this/can this get sent to the server periodically? How do we know which user or guest\_user captured interactions, and what were they?
   1. Each beaconID that resolves to an employeeID
   2. Timestamp of interaction
   3. Signal strength of beaconID at the time of interaction (power levels)
   4. Location of interaction – geolocation?
   5. Length of interaction (I know we are only checking for 10 seconds, but how about checking for the persistence of the signal and measure the total dwell time?)

[SM]: Yes, it will be stored in the DB. Currently, it is a WIP on app. Need to see how it can be retrieved in report/analytics etc.

1. We are also missing additional fields in the feedbacks collection - what device/OS/app version did the feedback come through? Because consumer may not always login through same device that they created account with, would they?

[SM]: Yes, app version should be stored in a service other than login. Thinking aloud - I think best place is when we send interaction data.

1. Is otps collection for the mobile Nexmo verification? Noted that “otp” key has a 4-digit number as value.

[GVB]: “otp’ key stores four-digit value for email based otp confirmation and for mobile number verification we will using Nexmo services only.

**SECURITY AND ACCESS CONTROLS**

Categories

* + 1. Database data
    2. Server data (that is not in database); any config scripts/variables
    3. Consumer app data
    4. Beacon nearables data

States

* + 1. Data at rest
    2. Data in transit
       1. Between beacon and consumer app
       2. Between consumer app and server
       3. Between server and database (they could be on two different machines)
       4. Between server and API caller (third party apps)
       5. Between database and other entity that is not the server (do we make this possible, or how do we turn it off?)

**Database data: mongoDB Security and Access control**

1. I was able to connect and run commands (e.g. show collections) to the remote database running at 54.173.120.23:27017/applause without any authentication credentials; would expect that the database access takes in access credentials
2. How are we restricting access to collections and/or documents (i.e. records) inside collections based on the role of the db user. For example, if I’m a brand admin of a particular customer, I should only get access to my brand document in the brands collection, and similarly, only get access to the documents pertaining to my brand’s locations in the locations collection
3. Does the primary contact in customer, brand, location have any access to the database? What parts of the database would they have access to? I think the role of primary is for reviewing feedback data (that will be presented in visual charts/graphs at a later date, not part of this release) and for sending notifications, alerts based on feedback being received. So if they have access, it would be read-only access.
4. Where are the user’s roles being created, and their associated permissions? How do we set this up/change? For example, Meimodo will have a super user/admin that has access to everything. If we were to create separate admins by customer, for example, where would we configure that?
5. How does DB access work for a customer’s customer admin/brand admin/location admin as they traverse the employee’s history? If employee was at a different Applause location that did not belong to the customer, we cannot show that data to the customer/brand/location.
6. We are required to encrypt databases that store employee names and IDs as such data is considered PII (Personally Identifiable Information) by U.S. Govt. (<http://www.gsa.gov/portal/content/104256)> and as such, both data at rest and data in transit has to be encrypted. So can we set up MongoDB with TLS/SSL? Should we encrypt all databases or only some databases?
7. Should the employee’s password be stored inside employees collection, and user’s password inside users collection? If this collection is encrypted, then maybe okay. What is typical approach for storing passwords?
8. For security (and speed) purposes, is it better to create separate feedback and employee collections by customer? *[Note: speed not issue for pilot, with sharding etc. to be dealt later]*
9. Do we need to get certificates from certification authority for server portal and consumer app(s)?
10. Passwords should not be retrievable, and can only be reset. [Note: Two factor authentication may be overkill at this stage]
11. Database IP address to be blocked from outside so only App Engine or a hosted server has access to database in production environment. For UAT and development, external access needed for development purposes (as it is today)

**CONSUMER PRIVACY**

**Information being captured about the consumer when account is created**

* *[Guest and registered user]* Device name, version; OS name, version
* *[Opt-in, for registered user]* email and/or phone; first one provided during sign up will be used for login purposes
  + Password associated with email or phone
* *or [Opt-in, for registered user]* FB login credentials (if consumer chooses to authenticate with FB vs. with email or phone)
  + Minimum FB account credentials required
    - TBD
  + Optional *[user can opt-out by deselecting attributes from profile other than the minimum required, during first login/authentication with Facebook account]*
    - TBD
* *[For registered user]* Geolocation latitude and longitude – do we need this?

[GVB]: We are storing this for analytics purpose and let us know that from where the maximum registrations happening and such analytical queries.

* *[Optional, for registered user]* profile image, full name

**Information being captured about consumer, when app is downloaded but no interactions are captured (at non-Applause enabled locations)**

* TBD

**Information being captured about the consumer, when app captures interactions at an Applause-enabled location**

* TBD

**Information being captured about the consumer, when consumer provides feedback on an Applause-detected interaction or feedback on a profile selected from the all\_profiles tab of the consumer app**

* TBD

**EMPLOYEE PRIVACY**

TBD

**Beacons’ questions (to ask developer/other startup contacts as well as Estimote)**

* Pros/cons of using nearable identifier vs. UUID
* Bulk setting of UUIDs to one number, changing power level and frequency?
* Nearable security – ensure no spoofing or modification of nearable ID by a malicious application (e.g. can someone use the Estimote app by being next to an employee wearing a nearable and change the ID?)
* Cost of beacon nearables at bulk – 100, 1000, 10000
* Nearables form factor – can we get in one standard small version with a peelable sticker? or without a case? Price impact for those different form factors?

**Frameworks notes**

**Server**

1. Express.js | body-parser: Parse incoming request bodies in a middleware before your handlers, available under the req.body property
2. Express.js | compression: for deflate and gzip (handle bulk upload of image files)
3. jsonwebtoken: means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JSON object that is digitally signed using JSON Web Signature (JWS) and/or encrypted using JSON Web Encryption (JWE)
4. node-geocoder: node wrapper around Google’s geocoder API – for geocoding and reverse geocoding
5. nodemailer: send emails
6. q: promise library; promises are abstractions to solve for complicated asynchronous code — what to do when action finishes or fails
7. mongoose: mongodb object modeling
8. fs: file I/O
9. winston: logging library

**Front-end/ web framework (supported in browser)**

1. AngularJS: enhanced HTML for web apps
2. jQuery: HTML document traversal, manipulation, eventhandling, animation and simplifies Ajax (XMLHttpRequest…) – why do we need jquery when you are using Angular? [per MD’s research, “jQlite already included in Angular and this should be all the jQuery that is necessary”]
3. Bootstrap: HTML and CSS based design templates for typography, forms, buttons, navigation and other interface components – are you using UI Bootstrap or that which relies on jQuery?
4. Sequence: responsive CSS animation framework