

**Snap Challenges**

Photography Gamified App

**JACOB ALLEN**

**19003931**

**UXCFXK-30-3**

**Digital Systems Project**

**Text

Description automatically generated**

# **Table of Contents**

# **Abstract**

# **Acknowledgements**

# **Table of Figures**

# **Table of Tables**

# **1 – Introduction**

**1.1 – Project Timeline (Gantt Chart)**

**2021**

**2022**

Table 1- Gantt Chart

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | OCT | NOV | DEC | JAN | FEB | MAR | APR |
| Research |  | |  |  |  |  |  |
| Requirements |  |  |  |  |  |  |  |
| Database Design |  | First Draft Completed – 19/11/21 |  |  |  |  |  |
| API Design |  | First Draft Completed – 20/11/21 |  |  |  |  |  |
| UI Design |  |  |  | |  |  |  |
| API Implementation |  |  | |  |  |  |  |
| Database Implementation |  | Completed - 03/12/21 | |  |  |  |  |
| Front End Implementation |  |  |  | | |  |  |
| Internal Testing (Own) |  |  |  |  |  |  |  |
| External Testing (User) |  |  |  |  |  |  |  |
| Finalise Report |  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| **Legend** | |
| Report Tasks |  |
| Design Tasks |  |
| Development Tasks |  |
| Testing Tasks |  |

**1.2 – The problem**

There are many existing popular photography sharing platforms, Instagram, 500px and Flicker to name a few. However, none of these sites offer an engaging way to get more images shared and often do not include detailed information about the photographs on the platform. Snap Challenges will attempt to resolve the gaps left by these sites by offering an engaging and informative photo sharing experience.

Photographers often experience burn out from their art, this can be for a variety of reasons, such as:

* Not being able to reach new locations.
* Not experimenting with styles outside of their comfort zone.
* High cost of new gear.
* A lack of engagement with the content they share.

Whilst Snap Challenges will not claim to be able to entirely eliminate these burn out factors, Snap challenges will attempt to mitigate this via the use of gamification. The app will provide users with challenges created by users and admins. These challenges will be regular and achievable yet challenging. This approach should result in more engagement with the platform than a typical photography sharing platform.

**1.3 – Why it’s important**

One of the main ways to improve at photography is to just get out there in the field, and practice. Snap Challenges gives the user a reason to get out and do this practice. This will hopefully lead to an improvement in their photography which benefits the whole community as we get to see better photos.

The app will show camera settings such as:

* Make
* Model
* Aperture
* Shutter Speed
* ISO
* Focal Length

Providing users with this level of detail about a photograph provides a good insight in to how the photo was taken. This will mean that the community can collaboratively improve their photography simply by observing others work.

**1.4 – Aims and Objectives**

AIMS:

1. To develop a platform independent app to gamify photography via the use of challenges.
2. To minimise burn out amongst photographers using these challenges to engage them with the app and their hobby.
3. To educate the users of the app by providing information about the camera settings used to achieve the resulting photo.

OBJECTIVES:

1. Research gamification benefits and drawbacks, and other times it has been used in the photography field.
2. Research React Native in comparison to Progressive Web Apps (PWAs).
3. Research data storage solutions MYSQL vs NOSQL.
4. Design and develop the database schema.
5. Design and develop the Python Flask API.
6. Design and develop the front-end client as either a React Native app or PWA.
7. Test the system from both a technical and user perspective.
8. Produce a report to summarise findings, research, implementation, and testing stages of development.

**1.5 – Potential ethical and legal issues**

The system will handle user data therefore it will need to conform to the General Data Protection Regulation (**GDPR**). Passwords will be stored in the database, this could be a security concern, to minimise risk passwords will be hashed using the SHA256 algorithm and salted. This will be done via werkzeug.security a python module which is already a requirement of flask so using it doesn’t increase the number of dependencies used. The only person who will know the plain text password will be the user.

# **2 – Literature Review**

**2.1 Gamification**

(Hamari and Huotari, 2012) Define gamification as “a process of enhancing a service with affordances for gameful experiences in order to support user’s overall value creation.” This implies that utilising gamification can offer higher engagement levels from users of a service and therefore making each user more valuable to that service. This definition is backed up by a survey completed by (Wang, et al, 2017) who state, “With regard to users’ perceptions, participants found that the gamified HCS (Kpoprally) provided a better engagement experience compared to the non-gamified one.”

**What roles does gamification have to play in an app?**

Gamification can be applied via a wide variety of unique approaches, some of which have likely yet to be conceptualized. Examples of such gamification practices are:

* Rewards
* Challenges / Quests
* Achievements
* Leaderboards
* Progress Indicators

A journal article (Hence, et al, 2017) states “badges, leaderboards, and performance graphs also seemed to contribute to an increase in perceived task mean-ingfulness.” This shows that the tried-and-true methods of gamification work to ensure the user has a more engaging and positive outlook on the app in general.

Examples of such gamification can be seen throughout a wide range of applications both web and mobile, in both massively popular and small niche apps, some of which can be seen in figure 1.

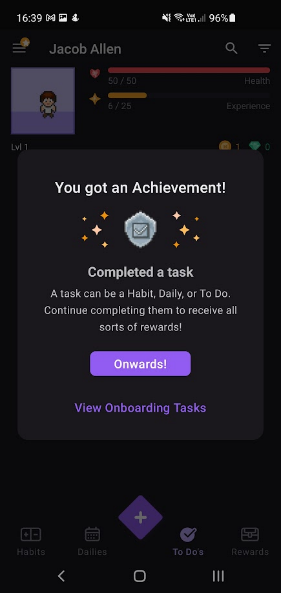
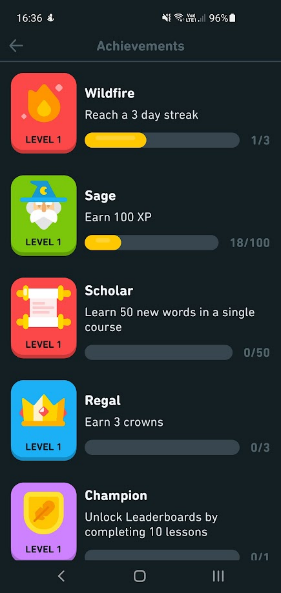


Figure 1 - Examples of Gamification in Duolingo and Habitica

**Gamification’s previous use in photography:**

There have been a few previous use cases of gamification in the photography field. One of which is PhotoTrip a website for helping tourists find less considered cultural/interesting sites, whose creators state “In the gamified version of PhotoTrip, we adopt three most common gamification strategies—points, badges, and leaderboards (PBL). Points are provided as scores, corresponding to the measure of each quest the user is involved in and badges are awarded whenever certain thresholds are met.”

A screenshot of a computer

Description automatically generated with medium confidence

Figure 2 - Gamification being used via use of a voting system in PhotoTrip

(Bujari, et al, 2016).

The implementation of gamification in this website is a starting point however it is quite basic only utilising three gamification techniques. And whilst gamification might aid its user’s engagement, it isn’t the only factor, and unfortunately this website lacks other engaging factors such as visual design. The app feels like it was designed with minimal thought towards aesthetic and a full focus on functionality. Also, the website appears to be designed purely for desktop use, which although is fine, reduces its usability by the very demographic it tries to serve, as tourists would likely want to spontaneously check the site during their trip not only use it for preplanning.

Another use of Gamification in relation to photography was the Narva application which was used to find the location and vantage point from which historical photographs were taken as shown in figure 3.

A picture containing grass, outdoor, sky, residential

Description automatically generated

Figure 3 - An example of image positioning with Narva (Liestøl, 2018)

The Narva app’s approach to gamification was not to follow traditional implementations such as badges, quests, e.t.c. But instead, was to use traditional games such as Hot & Cold and Jigsaw and to modify their rules to fit the context of the app as is shown by the following quotes from the conference paper:

* “The project or quest in the Hot & Cold game is comparable to that of the rephotographer who seeks to discover the original photograph’s vantage point by comparing the current view with the old photograph.”
* “the relationship between individual photos, and the space they record and depict, as well as their positions, quickly pointed to the jigsaw puzzle”

(Liestøl, 2018)

However, later in the paper it is made clear that this approach wasn’t as beneficial as they initially thought. They state, “Instead of searching among traditional analogue games for defining rules comparable to and shared by the activity in question (photography), we could have adapted more superficial strategies for the implementation of game elements, using extrinsic rewards like achievements and badges”. This is likely due to the superficial nature of most humans, we see value in gaining something that is ours to keep and to show off, in this case badges or achievements. We see little motivation to do something we are not getting any value or reward out of.

**Final Findings:**

There are many varied and unique approaches to gamification, and it can be applied in a wide range of different forms, however, some forms are tried and tested and are almost guaranteed to be more engaging. These are badges/achievements, challenges, and leaderboards. This is likely due to a human desire to be rewarded and compete, people are more likely to stick with something if there is a clear and well-defined goal at the end. Completing a task or earning an achievement provides a boost of dopamine, the effect of which is “well known for their strong responses to rewards and their critical role in positive motivation.” (Bromberg-Martin, 2010)

**2.2 HTTP Protocol**

The system will partially consist of a backend API, this will be developed from scratch using Python and Flask. To successfully develop this an understanding of the HTTP Protocol will be required. In particular, an understanding of the HTTP verbs will be required.

These HTTP Verbs are defined by the HTTP Protocol (Request for comments, 1999) and the definitions are as follows:

**GET:**

“The GET method means retrieve whatever information (in the form of an entity) is identified by the Request-URI. If the Request-URI refers to a data-producing process, it is the produced data which shall be returned as the entity in the response and not the source text of the process, unless that text happens to be the output of the process.”

**POST:**

“The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line.”

**PUT:**

“The PUT method requests that the enclosed entity be stored under the supplied Request-URI. If the Request-URI refers to an already existing resource, the enclosed entity SHOULD be considered as a modified version of the one residing on the origin server. If the Request-URI does not point to an existing resource, and that URI is capable of being defined as a new resource by the requesting user agent, the origin server can create the resource with that URI. If a new resource is created, the origin server MUST inform the user agent via the 201 (Created) response. If an existing resource is modified, either the 200 (OK) or 204 (No Content) response codes SHOULD be sent to indicate successful completion of the request. If the resource could not be created or modified with the Request-URI, an appropriate error response SHOULD be given that reflects the nature of the problem. The recipient of the entity MUST NOT ignore any Content-\* (e.g. Content-Range) headers that it does not understand or implement and MUST return a 501 (Not Implemented) response in such cases.”

**DELETE:**

“The DELETE method requests that the origin server delete the resource identified by the Request-URI. This method MAY be overridden by human intervention (or other means) on the origin server. The client cannot be guaranteed that the operation has been carried out, even if the status code returned from the origin server indicates that the action has been completed successfully. However, the server SHOULD NOT indicate success unless, at the time the response is given, it intends to delete the resource or move it to an inaccessible location.”

The API the system will use will be a CRUD style REST API. This means that each of the CRUD operations can be mapped to different HTTP verbs as follows.

Table 2 - CRUD to HTTP Mappings

|  |  |
| --- | --- |
| CRUD | HTTP |
| CREATE | POST |
| READ | GET |
| UPDATE | PUT |
| DELETE | DELETE |

These mappings fit the definitions of the HTTP verbs provided above.

**2.3 Database Choice**

There are many different options to choose from when it comes to databases. The main comparison to look into will be between SQL vs NOSQL based databases.

**SQL:**

SQL stands for Structured Query Language and acts as the main interface between the database and the client. It was initially conceived in 1974 by IBM Researchers and was first standardized in 1986 Most SQL based database software are relational database management systems (RDBMS). These databases consist of records, fields, relations and derived relvars. These can also be described using SQL terms:

Table 3 - RDBMS vs SQL Terms

|  |  |
| --- | --- |
| Relation DB Term | SQL |
| Record | Row |
| Field | Column |
| Relation | Table |
| Derived Relvar | View |

Groff (2002) States “SQL has become the standard database management language across a broad range of computer systems and application areas, including mainframes, workstations, personal computers, OLTP systems, client/server systems, data warehousing, and the Internet.” This shows the importance of SQLs role in the relational database.

SQL is hugely successful due to its early support from IBM, throughout the early 80s IBM pushed for the mainstream adoption of SQL, via the use of a commercialized product, SQL/Data System which was announced in 1981. In 1983 IBM announced a version of this system to run on their mainframe operating system VM/CMS. Later in same year they announced DB2, which began shipping in 1985, DB2 ran on IBM’s MVS operating system, which was used by large data centres. Then in 1986 the first SQL standard ANSI SQL1 is ratified. In 1987 ISO follows and ratifies ISO SQL1. However, despite the existence of these standards many versions of SQL have small variations between each other.

**NOSQL:**

NOSQL is a “Next Generation Database Management Systems mostly addressing some of the points: being non-relational, distributed, open-source and horizontally scalable.

The original intention has been modern web-scale database management systems. The movement began early 2009 and is growing rapidly. Often more characteristics apply such as: schema-free, easy replication support, simple API, eventually consistent / BASE (not ACID), a huge amount of data and more. So the misleading term "nosql" (the community now translates it mostly with "not only sql") should be seen as an alias to something like the definition above.” (NOSQL, 2009).

NOSQL DBs address some of the limitations of more traditional relational databases, mainly scalability, complexity, complex querying, and feature bloat.

**Scalability:**

Traditional relational databases can be hard to scale due to their design. The only ways to scale relational databases are to upgrade the hardware it is running on or distribute the database across multiple servers. However, “relational databases aren't designed to function with data partitioning, so distributing their functionality is a chore” (Leavitt, 2010)

**Complexity:**

Traditional DBs require all data to be put into tables however, not all data can be fit into tables. Because this data doesn’t fit well with the relational DB structure, it can be hard to force unsuitable data to work with this format.

**Complex Querying:**

Traditional relational DBs utilise SQL which although very strong for querying structured data, it struggles with other types as it is not designed to handle that type of data. Utilising SQL also requires a significant amount of code making it more complex. “SQL can entail large amounts of complex code and doesn't work well with modern, agile development” (Leavitt, 2010).

**Feature bloat:**

“Relational data-bases offer a big feature set and data integrity. But NoSQL proponents say database users often don't need all the features, as well as the cost and complexity they add.” (Leavitt, 2010).

**Final Verdict:**

The data that snap-challenges will be handling is very structured and lends itself well to and SQL based server as the data can nicely fit into tables. For this reason, snap-challenges will use MYSQL for its database server. However, if demand requires it a hybrid database architecture could be used by integrating a in-memory based key value store such as Redis as a cache. “Redis is an in-memory remote database that offers high performance, replication, and a unique data model to produce a platform for solving problems.” (Carlson, 2013). Redis will only be integrated into snap-challenges if demand requires it as having an in-memory cache is redundant and adds extra complexity if the app still performs well when utilising the standard MYSQL server.

**2.4 React Native vs Progressive Web App**

**2.5 Existing Solutions**

There are a few different existing apps the main ones being GuruShots and ViewBug. However, these apps both seem to suffer from similar flaws. The main flaw being and extreme focus on a pay to win style of gamification and at times the apps almost feel like they solely exist as a quick cash and data grab scheme. Figure 4 shows some examples of features where these apps ask for payment:

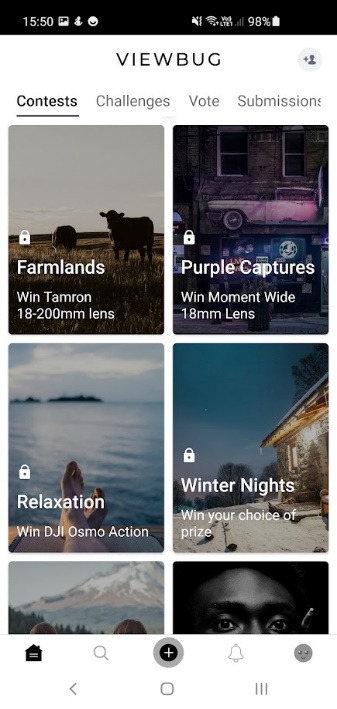
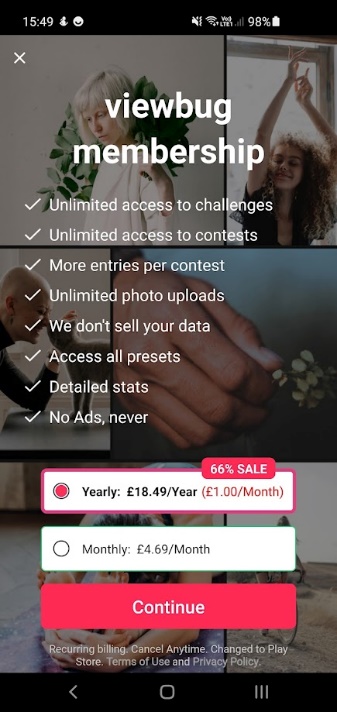
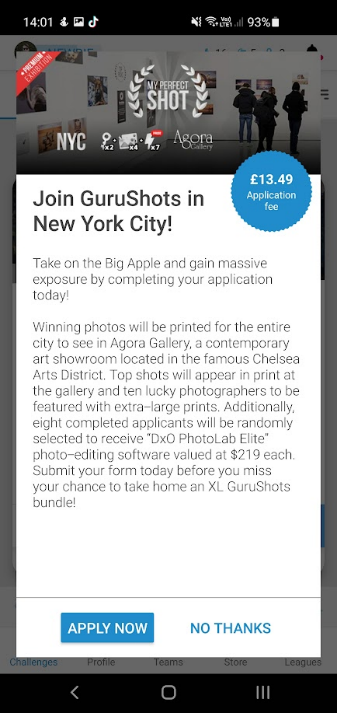
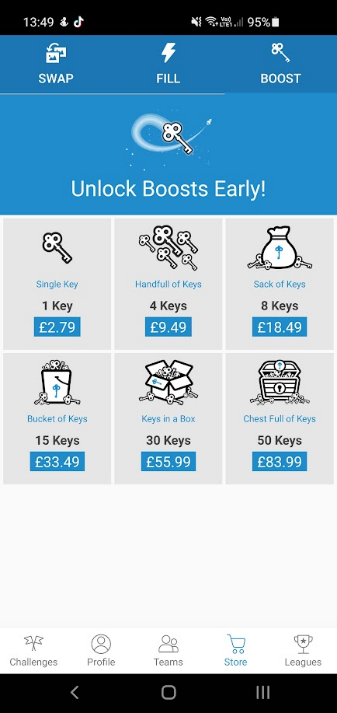


Figure 4 - Examples of Gurushots (left 2 images) and ViewBug (right 2 images) asking for payment.

Overall, Gurushots is less obtrusive with its microtransactions and membership, however ViewBug is almost unusable without a membership which costs 18.49 GBP per year. The rightmost image in figure 4 shows different contests, the small padlock above the contest name indicates that it requires this membership. As you can see there are no contents in the screenshot which can be freely accessed, with a total of 21 free contests out of a total of 78 available contests at the time of writing. This means that roughly 73% of contests at the time of writing required membership. This does somewhat make sense as these contests do offer expensive prizes, however this practice is misleading as ads for ViewBug such as the one shown in figure 5 do not show any of these locked challenges and do not mention the membership cost.

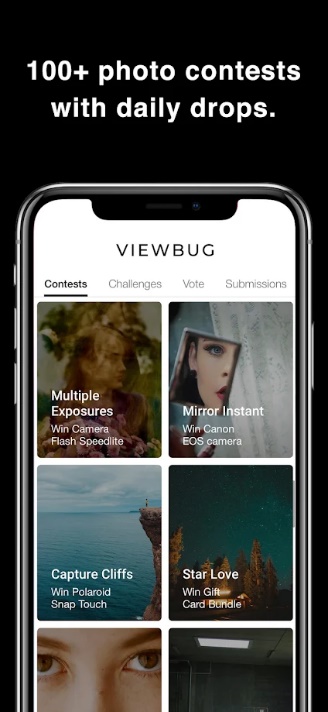


Figure 5 - ViewBug ads shown on Google Play (Google Play, 2013)

Both Gurushots and ViewBug do list camera metadata alongside images. However, this is very hidden, deep in menus, of which most users wouldn’t think to ever look through. This means it is likely ignored by all but the most curious of users and therefore is almost redundant to include.

On top of that public opinion on these apps seems to mostly be mixed. One user on the r/photography subreddit posted the question “Your Take on Apps Like GuruShots and Other Photo Challenge Apps” (Anon, 2018) to which another user submitted the following response

“

I like the 'idea' of gurushots and I like that you can see your ranking after the challenge is over.

But it encourages people to just vote on every image they see to vote more/faster. (Or to vote on sh\*\*\*y images because, maybe, they're less likely to get other votes)

I do think that truly 'great' images rise to the top, but I know when I decide to 'grind it out' and vote, I'm just voting yes as fast as possible - because NOT voting doesn't do s\*\*t for me. Scrolling past an image is a waste of effort.

If it was a tinder-like yes/no vote, and you got 'rewarded' for each one, I'd be more thoughtful with my votes.

” (Anon, 2018)

From this quote it is clear to see that Photographers do enjoy the concept of a challenge-based photography app, however, it is also clear that they feel Gurushot’s implementation requires too much dedication in order too succeed.

# **3 – Requirements**

**3.1 MoSCoW:**

**3.2 Functional Requirements:**

**API:**

|  |  |  |
| --- | --- | --- |
| **ID** | **DESCRIPTION** | **MoSCoW** |
| FR1 | A GET request sent to the Photo API endpoint must return the json data for the specified ID as well as the 200 HTTP response code. | M |
| FR2 | A POST request sent to the Photo API endpoint must create a record of the data on the database and return the json data of that record as well as the 201 HTTP response code. | M |
| FR3 | A PUT request sent to the Photo API endpoint must update an existing record on the database and return the HTTP response code 204. | M |
| FR4 | A DELETE request sent to the User API endpoint must DELETE a record form the database if it exists and return a 204 HTTP response code. | M |
| FR5 | A GET request sent to the User API endpoint must return the json data for the specified ID as well as the 200 HTTP response code. | M |
| FR6 | A POST request sent to the User API endpoint must create a record of the data on the database and return the json data of that record as well as the 201 HTTP response code. | M |
| FR7 | A PUT request sent to the User API endpoint must update an existing record on the database and return the HTTP response code 204. | M |
| FR8 | A DELETE request sent to the User API endpoint must DELETE a record form the database if it exists and return a 204 HTTP response code. | M |
| FR9 | A GET request sent to the Country API endpoint must return the json data for the specified ID as well as the 200 HTTP response code. | M |
| FR10 | A GET request sent to the Challenge API endpoint must return the json data for the specified ID as well as the 200 HTTP response code. | M |
| FR11 | A POST request sent to the Challenge API endpoint must create a record of the data on the database and return the json data of that record as well as the 201 HTTP response code. | M |
| FR12 | A PUT request sent to the Challenge API endpoint must update an existing record on the database and return the HTTP response code 204. | M |
| FR13 | A DELETE request sent to the Challenge API endpoint must DELETE a record form the database if it exists and return a 204 HTTP response code. | M |
| FR14 | A GET request sent to the Post API endpoint must return the json data for the specified ID as well as the 200 HTTP response code. | M |
| FR15 | A POST request sent to the Post API endpoint must create a record of the data on the database and return the json data of that record as well as the 201 HTTP response code. | M |
| FR16 | A PUT request sent to the Post API endpoint must update an existing record on the database and return the HTTP response code 204. | M |
| FR17 | A DELETE request sent to the Post API endpoint must DELETE a record form the database if it exists and return a 204 HTTP response code. | M |
| FR18 | A GET request sent to the Badge API endpoint must return the json data for the specified ID as well as the 200 HTTP response code. | M |
| FR19 | A POST request sent to the Badge API endpoint must create a record of the data on the database and return the json data of that record as well as the 201 HTTP response code. | M |
| FR20 | A PUT request sent to the Badge API endpoint must update an existing record on the database and return the HTTP response code 204. | M |
| FR21 | A DELETE request sent to the Badge API endpoint must DELETE a record form the database if it exists and return a 204 HTTP response code. | M |
| FR22 | A GET request sent to the ChallengePosts API endpoint must return the json data for the specified ID as well as the 200 HTTP response code. | M |
| FR23 | A GET request sent to the UserPosts API endpoint must return the json data for the specified ID as well as the 200 HTTP response code. | M |
| FR24 | A GET request sent to the UserChallenges API endpoint must return the json data for the specified ID as well as the 200 HTTP response code. | M |
| FR25 | A GET request sent to the UserBadges API endpoint must return the json data for the specified ID as well as the 200 HTTP response code. | M |
| FR26 | Certain information may only be retrieved from the API if an authorization token is provided alongside the initial request. | S |

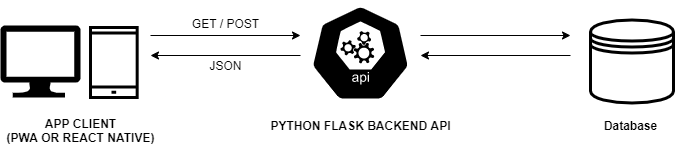
**3.3 Non-Functional Requirements**

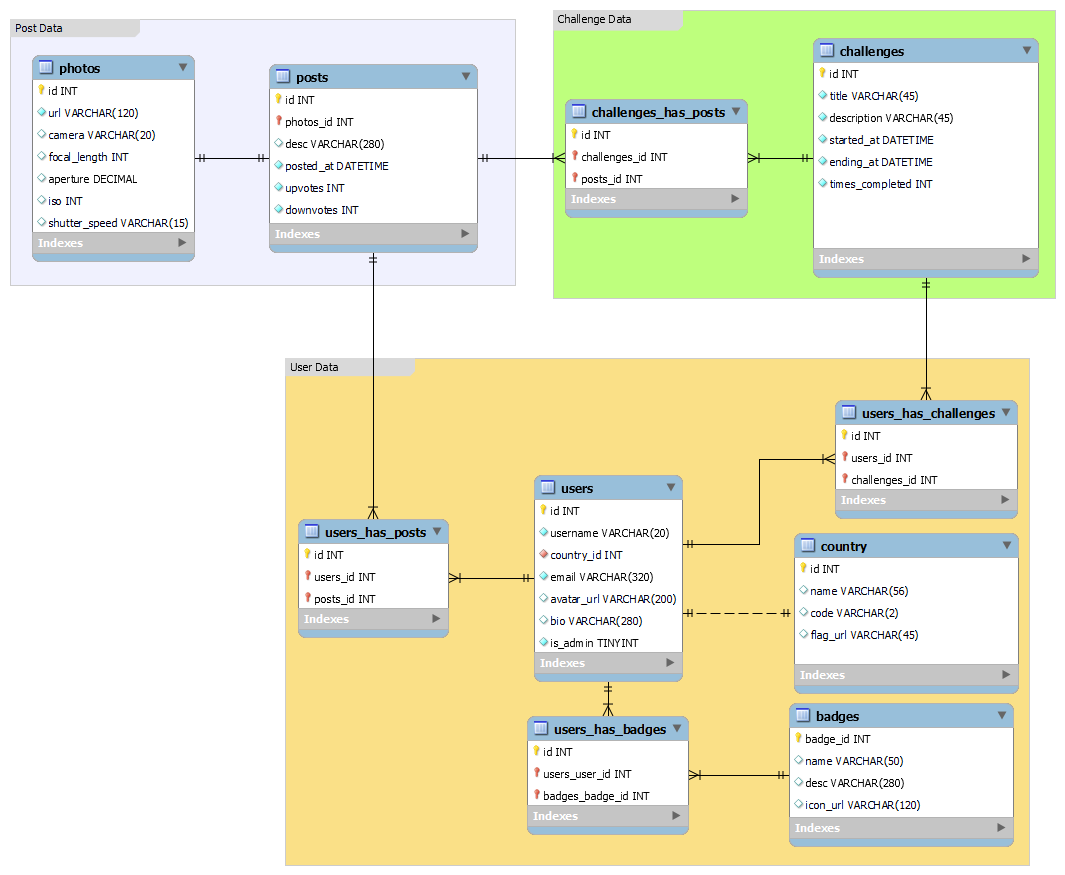
**API:**

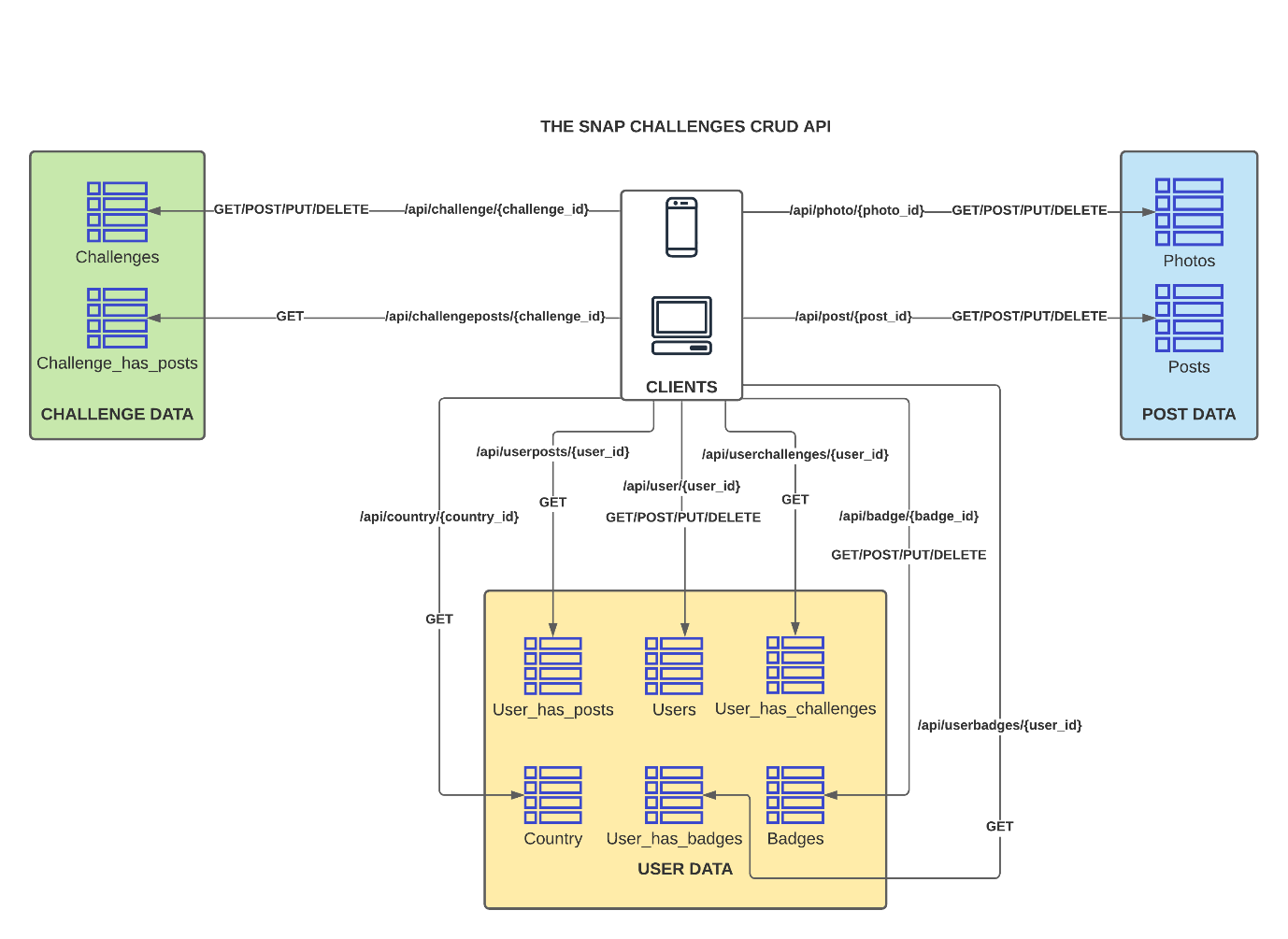
|  |  |  |
| --- | --- | --- |
| **ID** | **DESCRIPTION** | **MoSCoW** |
| NFR1 | Any API call should respond within 3 seconds. | M |

# **4 – Methodology**

# **5 – Design**







# **6 – Implementation**

# **7 – Evaluation**

# **8 – Conclusion**

# **9 – References**

* Request for Comments (1999) *2616 Hypertext Transfer Protocol -- HTTP/1.1* [online]. IETF Datatracker. (no place): RFC Editor. Available from: <https://datatracker.ietf.org/doc/html/rfc2616>
* Groff, J. R. and Weinberg, P. N. (2002) *SQL : The Complete Reference.* New York: McGraw-Hill Professional. Available at: <https://search-ebscohost-com.ezproxy.uwe.ac.uk/login.aspx?direct=true&db=nlebk&AN=80523&site=ehost-live> [Accessed: 21 December 2021].
* NOSQL (2009) *NOSQL Database.* Available From: <https://hostingdata.co.uk/nosql-database/> [Accessed: 22 December 2021].
* Leavitt, N. (2010) Will NoSQL Databases Live Up to Their Promise?. *Computer* [Online]. 43 (2), pp. 12-14. [Accessed 27 December 2021].
* Carlson, J. (2013) *Redis in Action* [online] New York: Manning Publications Co. [Accessed 28 December 2021]
* Hamari, J., Huotari, K. (2012) 16th International Academic Mindtrek Conference [online], Tampere, Finland, 3-5 October 2012. Association for Computing Machinery. Available From: <https://dl-acm-org.ezproxy.uwe.ac.uk/doi/10.1145/2393132.2393137> [Accessed 05 January 2022]
* Wang, X., et al (2017) Examining the Effectiveness of Gamification in Human Computation. *International Journal of Human–Computer Interaction* [online] 33 (10), pp.813-821. [Accessed 11 January 2022]
* Hence, J., et al (2017) How gamification motivates: An experimental study of the effects of specific game design elements on psychological need satisfaction. *Computers in Human Behavior* [online] 69 (no part/issue), pp.271-380. [Accessed 12 January 2022]
* Bujari, A., et al (2016) Using gamification to discover cultural heritage locations from geo-tagged photos. *Personal and Ubiquitous Computing* [online]. 21 (no part/issue), pp.235-252. [Accessed 16 January 2022]
* Liestøl, G. (2018). The Photo Positioning Puzzle : Creating Engaging Applications for Historical Photographs by Combining Mobile Augmented Reality and Gamification. In: 2018 3rd Digital Heritage International Congress (DigitalHERITAGE) held jointly with 2018 24th International Conference on Virtual Systems & Multimedia (VSMM 2018). [online] Digital Heritage International Congress (DigitalHeritage). Available from: [https://ieeexplore-ieee-org.ezproxy.uwe.ac.uk/document/8810038/](https://ieeexplore-ieee-org.ezproxy.uwe.ac.uk/document/8810038/%20) [Accessed 16 January 2022].
* Bromberg-Martin, E., S., et al (2010) Dopamine in Motivational Control: Rewarding, Aversive, and Alerting. *Neuron* [online]. 68(5), pp.815-834. [Accessed 16 January 2022]
* Google Play (2013) ViewBug - Photography. Available from: [https://play.google.com/store/apps/details?id=com.viewbug.viewbug&hl=en\_GB&gl=US](https://play.google.com/store/apps/details?id=com.viewbug.viewbug&hl=en_GB&gl=US%20) [Accessed 16 January 2022].
* Anon (2018) r/photography - Your Take on Apps Like GuruShots and Other Photo Challenge Apps. Available from: https://www.reddit.com/r/photography/comments/9vnjvm/your\_take\_on\_apps\_like\_gurushots\_and\_other\_photo/ [Accessed 16 January 2022].

# **10 – Appendices**