
# Backend Technical Document

MITCHELL ETZEL(PO), DARIUS SAKHAPOUR(SM), AUSTIN SHELTON, CEDRIC LINARES

| | |
|---|---|
| Product Name: | URL Logger |
| Release Name: | Alpha |
| Release Date: | 02/2/2017 |
| Revision Number: | 2 |
| Revision Date: | 02/2/2017 |

PRODUCED BY: AUSTIN SHELTON

This document contains the initial requirements for the backend system, and what is needed to meet those requirements. It does not cover all possible future requirements, but provides a foundation on which additional features may be added.

*This document will be modified as requirements are changed and further elaborated on.*

**Legend**

R This should be addressed by the RPMS project team.

U This should be addressed by the URL-Log project (our) team.

**User Database**

- We have no outstanding requirements of the User Database.

**Request Server**

The request server is the primary interface between the client and the databases. The server will be remotely hosted using a service such as Amazon AWS, and will likely use HTTP/S protocol to handle requests.
Additional security will be required to ensure that unintended connections do not interfere with the server's operations.

The server has three primary functions:

- User login
- Checking available surveys
- Data collection

**Collection Database**

The collection database will be hosted alongside the request server, and will be used to store collection information. We will plan to use InnoDB MySQL[3], but this may change if a better solution is discovered.

The database will have two primary sets of data: a list of available surveys and all of the information collected by the client.

The list of surveys will maintain keys and information about each survey that is made available, and will include start and end times.

There are a number of ways to manage the data collected by the client. The primary options are to store all of the data in one table, where each entry stores the key of its associated survey. Another option is to store the collected data for each survey in its own table. Other options are available, but would require a specific need for them to be implemented.

- [U] We need a formal list of what information is going to be collected.
  Currently, we will have separate fields for the user ID, full URL, reference URL, page title, and timestamp. Depending on the storage method, we may also include a survey ID.
- Would it be preferable for all information to be stored in separate fields, or should information, such as web page content, be stored in a block? More specifically, how much preprocessing should be expected, and how much do we expect extracted content to vary?
- Database access permissions will be split between different users to minimize potential damage in the event of unintended access by another party.

**Client Application**

The client application, from the perspective of the back-end, needs to implement the functions addressed in this document my means of web request. The client needs to be able to compile a request, send it to the server, and process the result of each request.

**Back-end Functions**

- Client logs in via user database, receives PID.
  The system should provide the PID on successful login, or indicate a failure.
  <u>What we need:</u>
  - How do we connect to the existing system?
  - What information does the user input to perform a login?
  - What protocol, if any, is used to perform a login[1]?
  - For testing purposes, we would like to have access to a sample account.
  - In what format is the PID stored internally (32-bit, 64-bit, unsigned)?
- Client checks for current surveys to opt-in to or -out of.
- Client sends data to the database, stored with associated PID[2].

- What we need:
    - Where should (will) this be located, and how should it be accessed?
      *This is addressed under the Request Server and Collection Database sections.*
    - What software do the other databases run on?
      *It is easier to manage multiple instances of the same type of database.*
    - Are there any specific or unusual security requirements for the database?
    - Who will need access to the database, and through what methods will data be extracted?

**Google Analytics**

The possibility of using Google Analytics to collect data has been brought up. Google Analytics is built to track user presence on websites and mobile applications. While Google Analytics can be used with Chrome extensions, it is made to collect data specific to the extension, and not pages that are visited while the extension is enabled. It may be possible to inject code into the webpage, but

**Notes and Remarks**

1. From a security standpoint, the client should not access the user database to perform a login. Instead, the client should send a request to a remote interface, which will access the database and confirm that the password is correct.

   Ideally, the connections should be encrypted. The remote interface should also take measures to prevent misuse, such as preventing continuous login attempts.

   We do not suggest allowing the client to perform the login check, as this would leave password structure and database interface vulnerable to reverse engineering.
2. Similar to login, we suggest using a proxy interface for interacting with the database.
   By default, the client should only have permission to write to the database, so the risk is lower than that of the login system. However, it would still be preferable to not have the client directly interact with the database.
3. InnoDB MySQL has good performance with large data sets, and is among the most commonly used database systems. As our team lacks sufficient knowledge of other database platforms, we cannot give a reasonable estimate as to how MySQL will fare against other platforms, given our expected data set.

**Other Considerations** *(also includes some Client feature considerations)*

- R Should the client include a method for obtaining forgotten login information, or will this be covered by another interface?
- Should the client be able to specify specific domains that should not be tracked by the client?