CS3216 Assignment Report – Group 5

Milestone 0: Describe the problem that your application solves.

We want to make it easy for people to maintain and keep ahead of their daily routines and to-do tasks. There are many people who have busy schedules with many to-do tasks to track, and often use sticky notes to help manage these tasks.

Although sticky notes work very well as to-do task planners and reminders, they aren't conveniently portable. You can't exactly carry your physical sticky notes board around with you. Hence we wanted to create an app that makes sticky notes mobile.

We also wanted to make the app a personal to-do tracker for a party of one, because very often, you just want to in one glance know what you need to do, instead of using some app where it is cluttered with other people's to-dos or some app where you have to click through numerous places just to see what you need to do.

Milestone 1: Describe your application and explain how you intend to exploit the characteristics of mobile cloud computing to achieve your application's objectives, i.e. why does it make most sense to implement your application as a mobile cloud application?

It should be a mobile application because you need to be able to refer to your to-do task list on the go, anytime anywhere.

It should use the cloud because many interactions happen online and you need to be able to sync those online interactions and to-do tasks into the app. Another reason is that you should be able to access your data on any device, without having to manually transfer your data, since this syncing can be better done via the cloud.

Milestone 2: Describe your target users. Explain how you plan to promote your application to attract your target users.

Our target users are university students and young working adults who have busy schedules with many to-do tasks to track, and wished that their to-do list could manage itself, without much manual intervention or rearrangement.

There are 2 main ways that we can promote the app.

- We can allow users to share their schedules and allow other people to add tasks to their schedules. This increases exposure of the app to other people.
- Our app supports Facebook, so we can also promote our app on the Facebook social platform.

Milestone 3: Pick a name for your mobile cloud application. (Not graded).
We have named our app as Taak ("Task" in Dutch).

Milestone 4: Draw the database schema of your application.

Entries	Users
id VARCHAR(255) user VARCHAR(255) value TEXT	id VARCHAR(255) realid TEXT token TEXT

The entries table stores entries corresponding to a sticky note. The users table is used for verifying the authenticity of the user (specifically Facebook users). The tables are not directly related to each other since anonymous users are not stored in the users table. However, the realid field in the Users table corresponds to the user field in the Entries table if the user is a Facebook user.

Milestone 5: Design and document (at least 3) most prominent requests of your REST API. The documentation should describe the requests in terms of the triplet mentioned above. Do provide us with a brief explanation on the purpose of each request for reference. Also, explain how your API conforms to the REST principles and why you have chosen to ignore certain practices (if any).

This REST API represents a collection of entries.
`api/entry/[user-id]/`
This represents a single entry.
`api/entry/[user-id]/[entry-id]`

Getting collection of task entries (collection)
Request: GET
Parameters: token (string) The token is used to uniquely identify a user session, which can be translated by the server to a user ID. While we can also identify the user via the id parameter, the token is for additional security. The server would then query the database for all entries belonging to the user.
Response: entries (Entry[]) The response is an array of the Entry data object, which is compatible with the one used by the client. An entry can contain a varying number of parameters as defined by the client, since the data contained within the entries are created by the client, rather than by the server. However, each entry must have an id that is unique (this id is generated by client).

The response is in JSON.

Updating task entries (collection)
Request: PUT
Parameters: token (string), entries (Entry[] in JSON), time (time stamp in milliseconds) The token is used to identify the user. The entries represent an array of entries that should be modified on the server side. The time stamp is used by the client to identify the message. Multiple entries can be modified via this request.
Response: code (string), message (string), time (time stamp in milliseconds) The code is used to identify the response type. "200" means the transaction is completed successfully. The message will inform the client of which entries have actually been updated on the server. The time stamp is an echo of the time value sent in the parameters - this is used by the client to identify the transaction. The response is in JSON.

Deleting task entries (collection)
Request: DELETE
Parameters: token (string), entries (Entry[] in JSON), time (time stamp in milliseconds) These parameters are similar to the PUT request, except that the Entry object can simply just have the "id" value.
Response: code (string), message (string), time (time stamp in milliseconds) The response is the same as PUT.

Get a single task entry
Request: GET
Parameters: token (string) The token is the same as the one used for the GET request for collections.
Response: Entry in JSON If no such entry exists or if the authentication fails, null is returned.

Modify/add a single entry
Request: PUT
Parameters: token (string), Entry, time (milliseconds) The token is the same as the one used for the GET request for collections. The Entry parameter represents the object that is to be inserted or modified on the server. The time parameter is echoed back to the client, used as a transaction id by the client.
Response: code (string), message (string), time (milliseconds) Similar to the Collections version, except for only one entry.

Delete a single entry
Request: DELETE
Parameters: token (string), Entry, time (milliseconds) Similar to the format used in PUT for single entry.
Response: code, message, time (milliseconds) Similar to the Collections version, except for only one entry.

There are 5 main principles for REST, which are identification of resources, manipulation of resources, self-descriptive messages, hypermedia, and statelessness.

1. Identification of resources: Our REST API returns results as a JSON string, which itself contains a representation of an object that contains either a collection of the resource, or one resource. The API focuses on the handling of a resource known as an Entry. Each Entry is uniquely identified by an id as well as the user who made the entry. Both IDs are generated by the client and server.
2. Manipulation of resources: Our REST API supports the manipulation of the stored resources via the PUT and DELETE requests. The API uses the standard GET, PUT, DELETE requests. POST is not used in this context because each user only has access to one collection (POST in REST is usually used for creating a new collection).
3. Self-descriptive messages: The REST API behaves differently based on the URL specified. The URL used will hint as to what object is being handled.
4. Hypermedia: Our REST api does not use hypermedia (i.e. collections having “pointers” to entries, entries having “pointers” to other metadata). This is because the API is designed for synchronization rather than general purpose querying; hence generally entries are handled in bulk via 1-3 main transactions rather than through multiple transactions (saving bandwidth for the client, which is critical for mobile applications). The client will need to have access to all the entries at all times rather than a subset, thus our API does not follow this principle.
5. Statelessness: The API is semi-stateless. Suppose a request fails, the client can retry it without any consequence. The server however is not truly stateless, since it does check if the user is authenticated (which itself involves a state), but the transactions/requests themselves are stateless.

Milestone 6: Tell us some of the more interesting queries (at least 3) in your application that requires database access. Provide the actual SQL queries you used.

1. Putting entries into the database

```
INSERT INTO entries(id, user, value) VALUES($id, $user, $value)
ON DUPLICATE KEY UPDATE id=$id, user=$user, value=$value
```

This query updates an existing value or adds the entry if it does not exist. It is more efficient than deletion followed by insertion, or checking if the entry exists then choosing update or insertion.

2. "Deletion" of entries

```
UPDATE entries SET value='' WHERE id=$id
```

This query sets the value of an entry to an empty string. The value is typically a JSON string. By setting it to an empty string, we know that the entry is "deleted" by the user. The reason why we are doing this form of "deletion" instead of actual deletion is because we need to perform synchronization of entries that were deleted online, but not purged on devices where the app is used but not yet synchronized.

3. Selecting all entries belonging to a user

```
SELECT * FROM entries WHERE user=$user
```

This query returns all information about all entries belonging to a user.

Milestone 7: Find out and explain what [QSA,L] means. Tell us about your most interesting rewrite rule.

QSA means "Query Statement Append". Any GET parameters behind the URL will be transferred to the rewritten URL if the rewrite rule has this flag.
L means "Last". If this rewrite rule with this flag is matched, the rewrite rule mechanism will stop at this rule and no longer check for other rules.
[QSA,L] means both the "QSA" and "L" flags are set for this rule.

RewriteRule

```
^api/entry(?:/([^/]+))?(?:/([^/]+))?/?$ api.php?action=entry&id=
$1&entryid=$2 [QSA,L]
```

This rule rewrites all requests to
"[APPURL]/api/entry/[PARAMETER]/[PARAMETER2]" to
"[APPURL]/api.php?action=entry&id=[PARAMETER1]&entryid=[PARAMETER2]".
[PARAMETER1] is the ID of the user, while [PARAMETER2] is an optional parameter used for modifying a single entry.

Since the "QSA" flag is used, we can append additional parameters to the pre-rewritten URL and it'll be passed to the API. The L flag is used for stopping the rewrite matching,

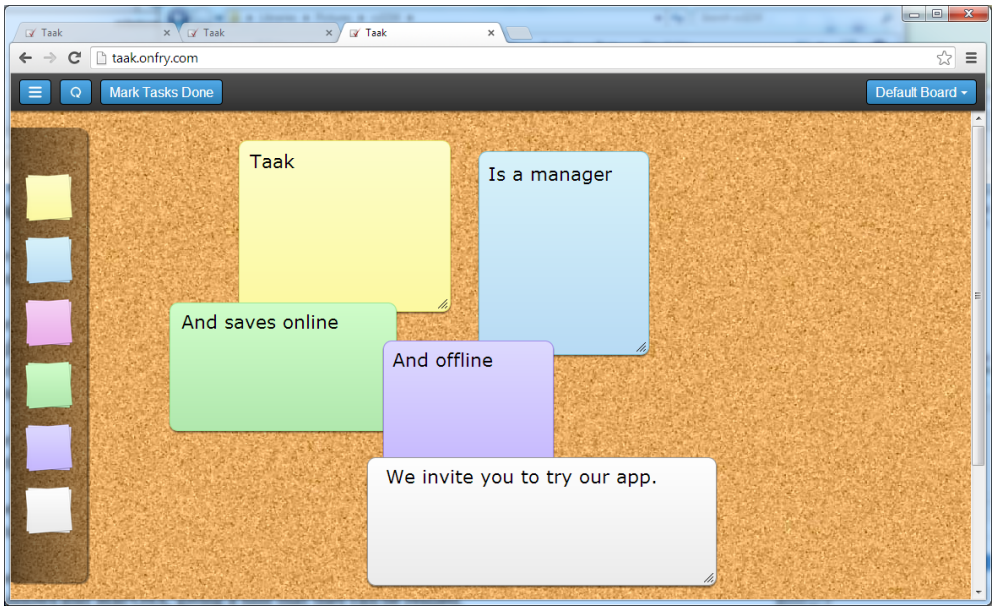
otherwise the other rewrite rules that occur after this rule may interfere with this particular rule.

Milestone 8: Create an attractive icon and splash screen for your application. Try adding your application to the home screen to make sure that they are working properly.

The icons (shown below) would also be used as the icon for our application.

The one on the left is the main icon used as the FavIcon, while the icon on the right is used for the home screen on iPhones and iPads.

Milestone 9: Style different UI components within the application using CSS in a structured way (i.e. marks will be deducted if you submit messy code). Explain why your UI design is the best possible UI for your application.



We have made our UI resemble the good old sticky note board, to give users a sense of familiarity on the first time that they use our app.

For the background of the sticky notes, we chose colors that are lower in saturation and higher in lightness, to once again give familiarity through resembling the colors of real

sticky notes and to also make it easier for users to see the text written on their sticky notes.

For the font of the sticky note text, we chose sans-serif fonts such as Helvetica to utilize native fonts used in mobile devices such as the iPhone and iPad.

There are also some little nuances in the UI that provides hints and makes it more intuitive.

The notes can be resized. The resize handle used resembles typical handles seen in Windows and Mac OSX, giving a hint that they can be resized.

We deliberately chose images of a pile of sticky notes for the icons where you drag and drop sticky notes from. This is a replication of real life where you tear and get your sticky notes from a pile.

Milestone 10: Implement and explain briefly the offline functionality of your application. Make sure that you are able to run and use the application from the home screen without any internet connection. State if you have used Web Storage, Web SQL Database or both in your application. Explain your choice.

Application Cache is implemented on the application. The application can be used offline as its assets such as images, JavaScript and HTML files are stored on the device in the application-cache.

The application will store task entries in Web Storage so that it can be read and modified offline. We are using Web Storage because it is widely supported across many browsers compared to other alternatives like WebSQL.

Milestone 11: Explain how you will keep your client in sync with the server. Elaborate on the cases you have taken into consideration and how it will be handled.

Answer:

The client will first perform a GET request from the server (assuming already authenticated earlier via POST request) to retrieve the full list of entries. The server will also keep deleted entries for the purposes of synchronization, so that these entries will not be accidentally re-added by unsynchronized clients.

The client will then compare the entries from the server and the ones from its local copy, and attempts to merge them. All entries will have a “last modified” parameter that is used to determine which entry is to be kept. The merged copy is then compared with the server copy to determine which entries should be updated or deleted on the server. The

client will then follow up with the PUT and DELETE requests to modify the state on the server so that both the client and server are now synchronized.

"Deleted" entries on the server are set to use an empty string. On the client side, deleted entries that are not yet synchronized with the server will also use an empty string, but can be removed after synchronization.

Milestone 12: Compare the pros and cons of basic access authentication to other schemes such as using cookies, digest access authentication or even OAuth. Justify your choice of authentication scheme.

Among the possible authentication schemes, we evaluated the pros and cons of each and decided on OAuth.

Cookies
Pros: Easy to implement and use.
Cons: Cookies that can be accessed by JavaScript (i.e. not HTTP-only cookies) can be stolen by Cross-Site Scripting or Man-in-the-middle attacks for use in replay attacks. In the case of HTML5 mobile apps, if cookies are used for authentication, they are likely not going to be HTTP-only because the JavaScript side of the client may need access to information to authenticate the client (especially for REST requests). In the example provided in the assignment document, the cookie sends the password in clear, which poses a large security risk.

Digest Access Authentication (DAA)
Pros: More secure than cookies because the password is not sent in clear.
Cons: Still vulnerable to man-in-the-middle attack because authenticity of the server is not verified.

OAuth
Pros: More secure than cookies and DAA because server authenticity can be verified. User does not share credentials like passwords with the app.
Cons: Much harder to implement. User must leave the app to visit the login page on the actual site to authenticate (otherwise it'll violate the "not share credentials" part). In OAuth 2.0 authentication cannot be done via AJAX because the user must visit the site in the intermediate step.

We are using Facebook OAuth because of the following reasons:
1. OAuth is generally considered secure by the experts and practitioners. OAuth outsources authentication to a authorized third-party such as Google, Facebook etc., thus leverages established authentication process of those organizations. It does not

involving storing user credential or password in our database, but only an authentication token provided by the third-party of choice.
For the initial login, one-time code provided by Facebook is used for instead of token, which has an expiry time of several thousand minutes. Also, we are storing both the user id and the code into database and local storage. In subsequent authentications, we compare the user id and token inside local storage with the pair inside database to ensure authenticity and consistency. These methods improve the security. They also provide cross-device usability since the third-party we are using, Facebook, is accessible in many devices and provides the same methods.

2. OAuth is convenient for the user. If the user has logged into the third-party website already (in the case of Facebook, the chance is high), the authentication process will be as easy as a single click and a few automatic redirection for the user. This is the case even for the initial login. As such, OAuth ensures a smooth authentication process for the user and saves the hassle of typing username and password.

Milestone 13: Describe 1-3 user interactions within the application and explain why those interactions help make it better.

For the Taak app, we have made our UI feel right as if you were managing real sticky notes.

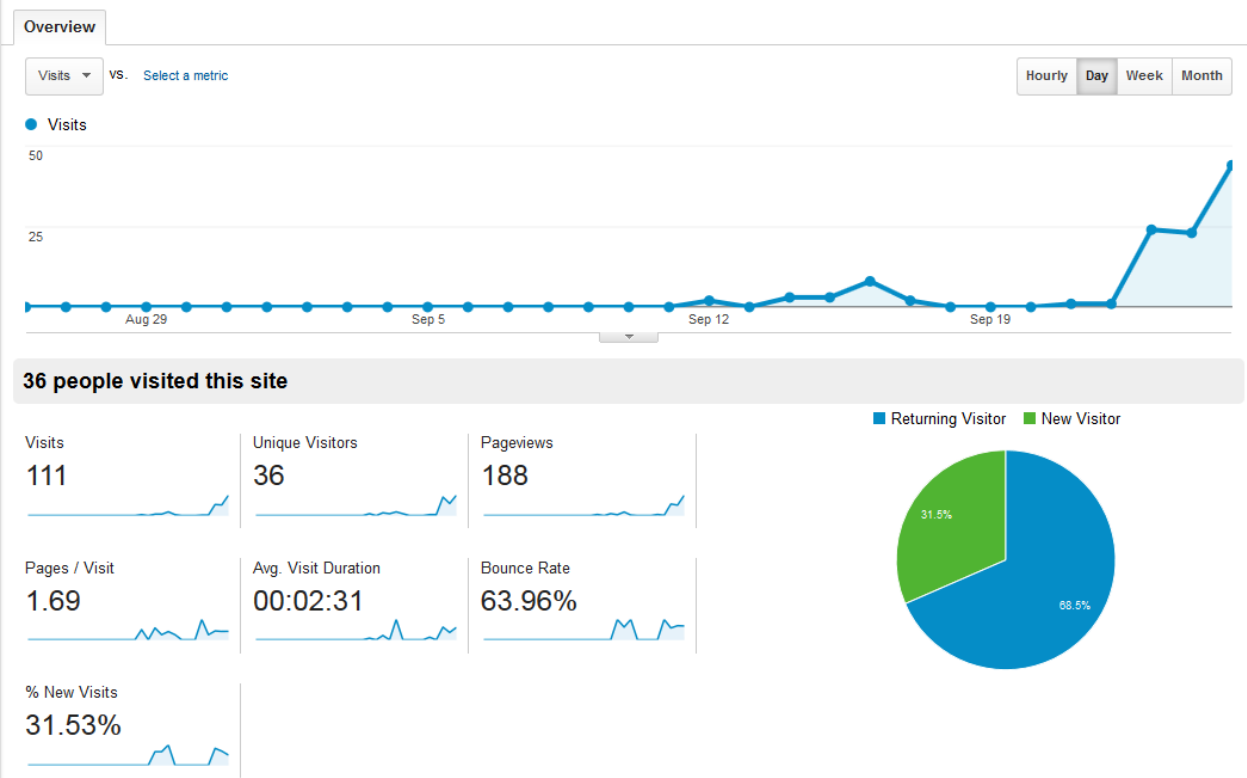
- Drag and drop a sticky note from the sticky note pile to create (paste) one. This is a replication of real life where you tear your sticky notes from a pile and paste them on a surface. Drag a sticky note to the bin to delete (peel in real life) it.
- Sticky notes can be placed anywhere on the board as you desire, and you can move them around subsequently. This allows you to group tasks together, for easier reference, just like in real life.

The similarities in the user experience reduces learning curve required for users, and makes the app feel like “it just works” so that anyone can easily pick the app up. On top of that, we also implemented features that provide convenience for the users.

- When you tap on a sticky note, you are selecting the sticky note. The selected sticky note is brought to the top layer. It enables you to see the select sticky note clearly, over other surrounding sticky notes that might have been previously covering and obscuring important parts of it.
- On that same selection, a bar also appears to allow you to change the color of the sticky note, so that you can change it's color scheme and "affiliation", without having to rewrite everything on it. Something that you can't do in real life.

Milestone 14: Embed Google Analytics in your application and give us a screenshot of the report. Make sure you embed the tracker at least 48 hours before submission deadline as updates are reported once per day.

Analytics is installed. The image shown below is the screenshot of the traffic statistics.



Milestone 15: Identify and integrate any social network(s) with users belonging to your target group. State the social plugins you have used. Explain your choice of social network(s) and plugins. (Optional)

While Taak doesn't use social plugins or even the Facebook SDK, we did actually have a login system that uses the Facebook platform. We chose Facebook because it is a well-established OAuth authentication provider. Facebook login provides secure, trusted and convenient authentication service.

Milestone 16: Make use of the Geolocation API in your application. Plot it with Bing or Google Maps or even draw out possible routes for the convenience of your user. (Optional)

The use cases of our app do not depend on location. Hence this milestone is unsuitable for our app. It would be redundant to add geolocation in, so we left it out.

Milestone 17: Justify your choice of framework/library by comparing it against others. Explain why the one you have chosen best fulfils your needs. (Optional)

We have used jQuery TouchPunch and jQuery UI to allow us to support drag-and-drop operations on touch devices.
jQuery UI is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library. It provides features that are much needed in highly interactive web applications. Specifically, we make use of its draggables, droppables and resizables interaction sets to achieve the desired manipulations in our app.
The TouchPunch library is a helper library for the popular jQuery UI that allows us to use jQuery UI features such as draggables and resizables in touch devices, without needing to add additional code. The library allows us to provide an optimized experience for touch screen users.