Proceedings of The National Conference
On Undergraduate Research (NCUR) 2016
University of North Carolina Asheville
Asheville, North Carolina
April 7-9, 2016

Electronic Bill (eBill) Collector with Single Sign-On

Konstantin Kazantsev
Computer Science - Information
The University of North Carolina at Asheville
One University Heights
Asheville, North Carolina 28804 USA

Faculty Advisors: Dean Brock, Rebecca Bruce, Marietta Cameron, Susan Reiser, Kevin Sanft, Charles Sheaffer, Adam Whitley

Abstract

Electronic Bill (eBill) Collector is a web application designed to make bill paying easier, by finding and organizing a client's electronic bills. The application filters the client's emails looking for specific senders; then parses selected emails to extract certain fields, e.g. due date and due amount; and, finally, archives the information in a database for easy sorting and retrieval. To keep the clients' information secure and private, authorization and access restrictions are imposed. Authorization security, as in OAuth, secures clients' data from outside threats and builds a layer of security within to ensure the client only sees information pertaining to them. There will also be access restriction at the SQL level to create another layer of security. Developed in Brackets using JavaScript, the web app implements the Gmail API, Google Cloud SQL, and OAuth 2.0. Gmail API is used because clients may receive emails from other email service providers. In order to have a good user experience (UX), the web application is easy to use and easy to learn. The accessible user interface (UI) I is accomplished by using user's language, having clear and easy to use controls, and have clear ways to undo unwanted actions. Five representative users tested the eBill Collector and their feedback was used to iteratively improve the application's functionality and its user experience.

Keywords: Gmail, Web Application, Single Sign-On

1. Introduction

Paper bills are becoming a thing of the past. People are switching to electronic bills for environmental reasons and expediency. With more than just bills flooding a user's inbox, chances are high that a bill will be overlooked or forgotten. Users need a service that will decrease the time spent gathering and viewing current bills. Yet, users do not need to be burdened with an additional sign-on to access the service.

In this paper the author will summarize the current research on single sign-on systems, explain the architecture of the eBill Collector web application from logging in to logging out, and conclude with the features to be implemented in eBill Collector Version 2.0.

2. Literature Review

Single sign-on (SSO) is an authentication protocol that allows users to sign onto multiple client websites using only one username and password. Suoranta and Sun explain that most SSO systems depend on redirecting to authenticate the user before allowing user access to the client website[5,4]. While OAuth is an authorization protocol, some providers have implemented OAuth to be deployed as SSO. Leiba explained that while SSO systems just authenticate users, in OAuth users can grant limited access to their information in client websites without giving the client their username and password[2]. For this reason, SSO and OAuth are becoming more popular among users, clients, and providers. Darwish and Sun describe how the implementation of OAuth SSO allows users to sign-on and give limited permissions to clients[1,4]. In this vein, the eBill Collector web application uses OAuth SSO. When the user starts the login process, the user are asked to first log into their Gmail account if they are not already logged in. Then the user is asked if they wish to give eBill Collector limited access to their information. Once the user approves, the user will be redirected to the web application. No additional information is asked of the user because Sun has found that many users find it confusing when clients request more information right after login. Once the user is done, the user has the option to logout just from the eBill Collector application or logout from Gmail as well. Suoranta and Sun[3,5] suggest that client websites should give users the option to logout of the entire SSO system in one click.

3. Methodology

3.1 Software Used

Brackets was used to develop the web application, because it is a modern text editor with a blend of visual tools. OAuth 2.0 is used to access the Google APIs and as a sign-in option to reduce the number of user ids and passwords a person needs to remember. Also, eBill Collector uses the Gmail API to retrieve emails. The reason for selecting Gmail is that it has the ability to receive and send other types of emails: e.g., Hotmail and Yahoo. For storing the collection of electronic bills and their history, MySQL is used because it is relatively light-weight and can be extremely fast when applications leverage its architecture. Speed is important when constructing a web application. The email parser was written in Python. Its function is to parse the user emails to retrieve electronic bill information. Python's email package provides a standard parser that understands most email document structures.

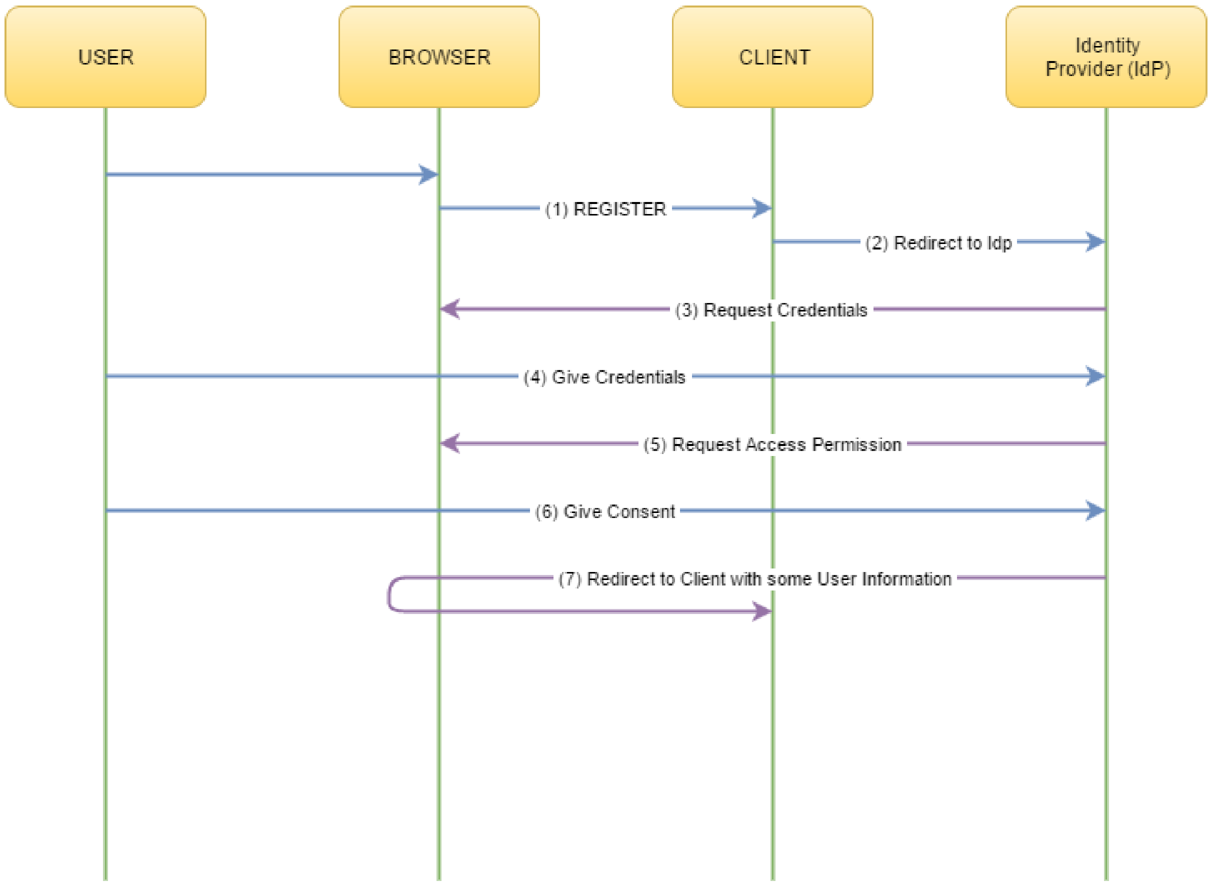

Figure 1: Typical Single Sign-On (SSO) process

3.2 SSO

To reduce the number of usernames and passwords a user needs to remember, the web application uses OAuth 2.0 protocol to sign up and use eBill Collector using the user's Gmail account. Even though OAuth 2.0 is an authorization protocol, the eBill Collector implements Google OAuth 2.0 as a SSO. What is the difference between typical SSO and OAuth SSO? A typical SSO only authenticates a user when logging into a website.

Figure 1 shows the process of a typical SSO process when a user signups for a client website [3,5]. A user first accesses the client on the browser. (1) Then, the user selects an Identity Provider (IdP), such as Google, Facebook, or other to use for authentication. (2) The client redirects the user to the chosen IdP for the authentication process. (3) The IdP will ask the user for username and password. This step may be skipped if user has already logging into the IdP inside the browser. (4) Otherwise, the user will provide the IdP with their username and password. At this point, the user is signed into the IdP as well. (5) After verifying the user against the IdP's database, IdP requests the release of the users profile information. This step may be skipped if the user is returning to the client website. At step (6), the user can review the information the client is requesting and decide to share this information or deny it. If the user denies access, the client will not be able to access user's resources. (7) Once the user gives consent, the IdP will redirect the user to the client with the user's profile information that the client has requested.

The client receives only a copy of requested user profile information. However, the client does not receive the user's username and password. Therefore, when the user returns to the client in the future, the user will need to repeat steps 1 through 4. Google OAuth is the same for the first 6 steps, similar in 7th step and has more steps as Figure 2 illustrates [1,2,3,4]. The difference in the 7th step is that the IdP returns an authorization code through the browser to the client. In step (8), the client uses the authorization code to request a token. (9) After the IdP authenticates the client and validates the authorization code, it issues an access token. This token contains the scopes and durations of the user granted access. (10) With the token, the client can retrieve the information needed and (11) provides the user desired service.

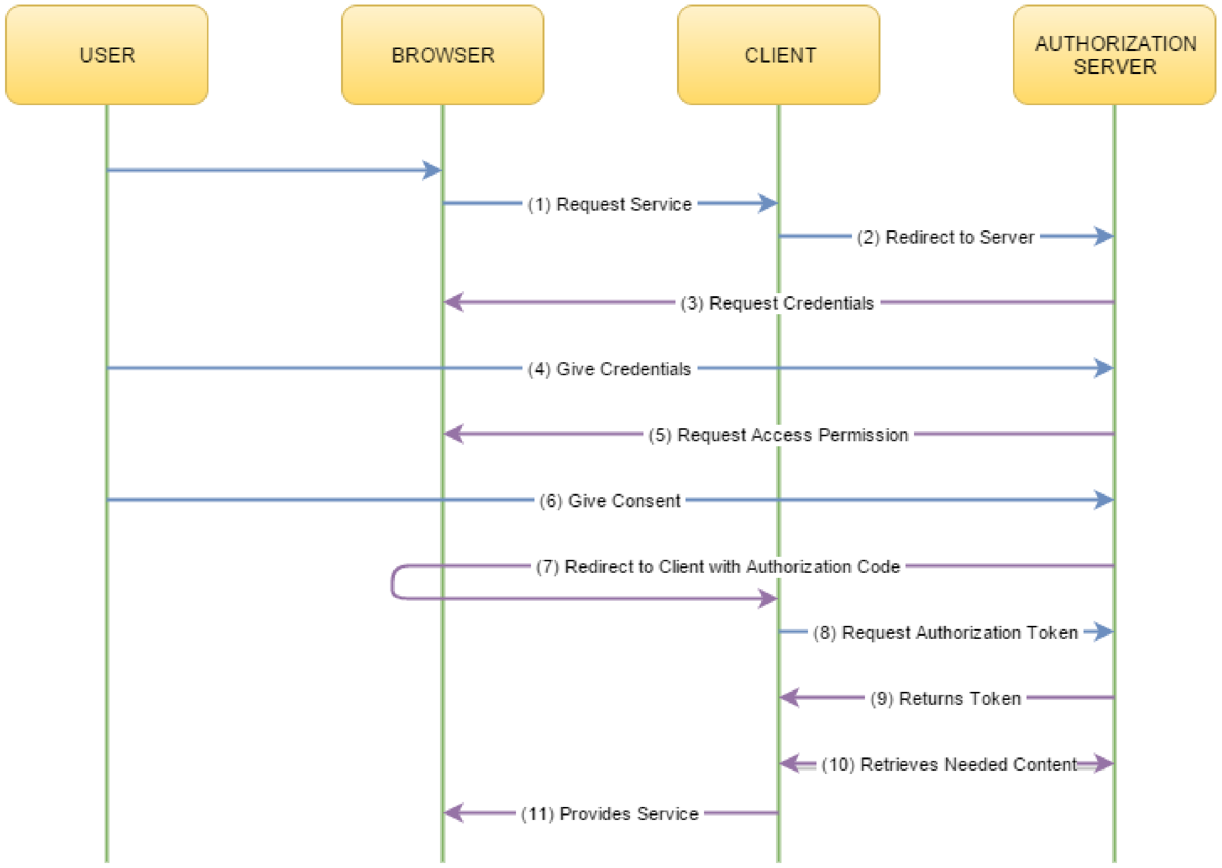

Figure 2: Google OAuth 2.0 process

3.3 eBill Collector web application

eBill Collector uses the Google OAuth 2.0 SSO to authenticate the user at each login. If no error occurs at login, the web application checks if this is the user's first time logging into the web application. Figure 3 illustrates the process of entering the web application. If the user is new to the web application, the user's email is stored in a log and a table is created. This table stores the user's bill due date, merchant name, last four digits of account number if provided by the email, the amount due and date when user paid the bill. All information is collected from the email that a merchant sends to the user except for the date paid. All this happens automatically.

At the home page, the user sees a table containing outstanding bills. The user can enter the date when the bill is paid and the bill will be removed from the table. The user may enter the merchant's email address at the home page. Once a merchant's email address is entered, the web application will access the user's Gmail and search for bills corresponding to the merchant's email. If any bills are found, they are added to users table and will show up on home page sorted by due date. Otherwise, nothing new happens from user's perspective.

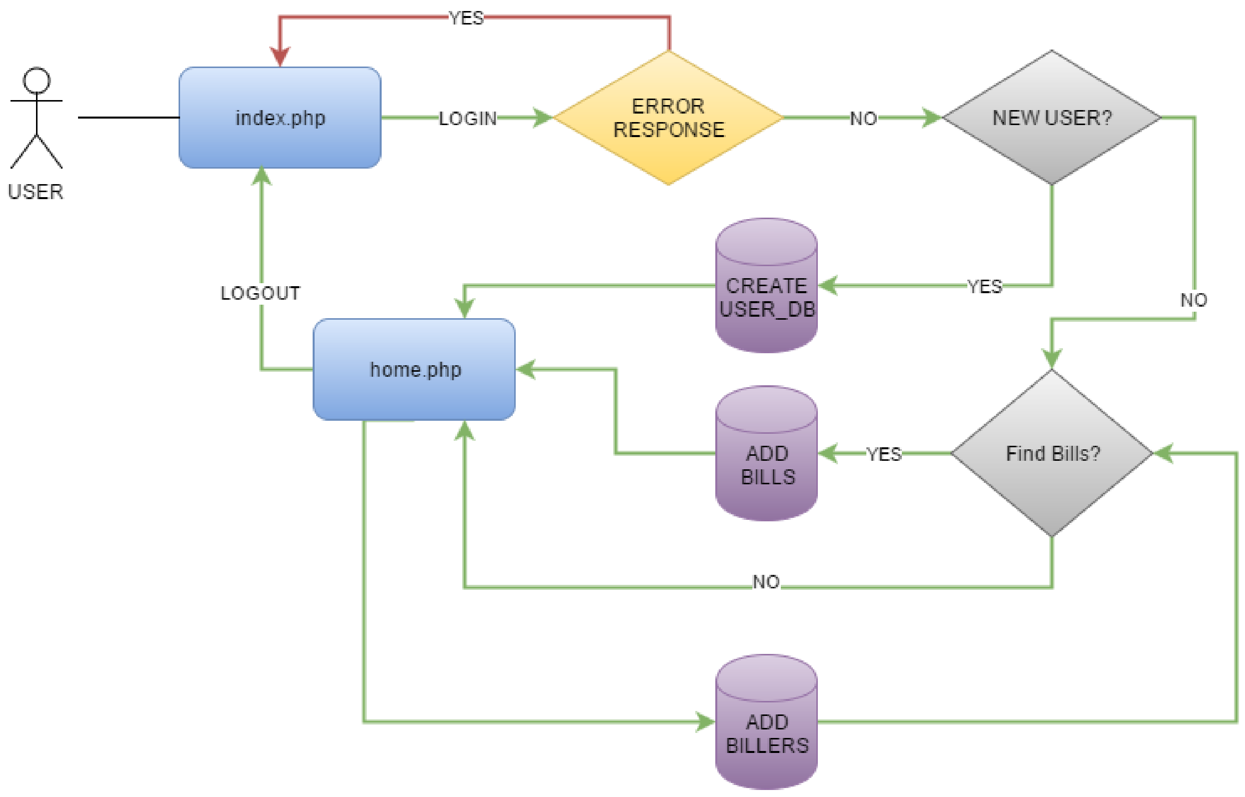

Figure 3: eBill Collector web application main process

3.4 Logout

Once the user is finished reviewing bills, the user has two options to logout. One of the logout options only logs the user out of eBill Collector, while the second option also logs the user out of the IdP. A small study found that many users are unaware that logging into any client using any IdP, logs them into the IdP as well[3]. Creating a single logout (SLO) to end the IdP session and client session[5] in which the SLO is executed should be the minimum expectation from a client that uses SSO login system. When the user logs out by using either SLO or client only logout, the user is redirected back to the welcoming page.

4. Conclusion

Everyone wants to be more efficient, and efficiency is the main goal of eBill Collector. In its current condition, eBill Collector reduces the time spent on gathering and evaluating bills. Using a Google login, eBill Collector reduces the burden of tracking and memorizing usernames and passwords. To give a sense of trustworthiness to potential users and an atmosphere of not being intrusive[3], eBill Collector does not redirect user to the homepage when user access the web application, even if they have already logged in to Google. Instead, eBill Collector only requests access to information needed to accomplish its service. Future developments will allow eBill Collector to reduce dependency on the user to gather electronic bills.

5. Future Work

While the current version of the eBill Collector web application decreases time and headaches in reviewing bills, features and enhancements have been identified for inclusion in the next version. The highest priority enhancement to eBill Collector is to allow it to have access to its users' Gmail accounts when they are logged off. This will permit eBill Collector to add new bills as soon as they are received by the user. The next highest priority enhancement is to have eBill Collector to parse all users' emails to search for merchants that have not been entered and their corresponding bills and add both to users' tables. This will streamline the time it takes for bills to appear in users' tables. A new optional feature, suggested by testers of the current version, is the ability for the user to elect to receive SMS notifications when eBill Collector finds a new bill or bills. Allowing the users to approve or reject this feature is optimal because carrier charges may apply.

Several other features will be added to eBill Collector. First on the list is the ability to view paid bills from the last two years. The user will be able to sort bills by date due or date paid, merchant, and amount paid. An option will be given to calculate the amount that was paid in that time frame or by a particular merchant. The second feature is the ability to export paid and/or unpaid bills in a particular time frame and/or by merchant to an excel sheet or pdf file. Thus the user may store a digital record of bills paid and, if desired, print them. The third feature to be identified is the ability to allow eBill Collector to fully manage bills for a user, by providing a process for the manual entry of bills from merchants that do not send electronic bills. The final planned feature is to have a desktop version of eBill Collector. Allowing users to use a desktop version will ensure that users will be able to use eBill Collector even when the user is offline.

6. References

1. Darwish, Marwan; Ouda, Abdelkader, "Evaluation of an OAuth 2.0 protocol implementation for web server applications," in Computing and Communication (IEMCON), 2015 International Conference and Workshop on , vol., no., pp.1-4, 15-17 Oct. 2015. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7344461&isnumber=7344420 (accessed February 13, 2016).

2. Leiba, Barry. "OAuth Web Authorization Protocol." IEEE Internet Computing 16, no. 1 (01, 2012): 74-77, http://wncln.wncln.org/docview/914302361?accountid=8388 (accessed February 13, 2016).

3. Sun, San-Tsai, Eric Pospisil, Ildar Muslukhov, Nuray Dindar, Kirstie Hawkey, and Konstantin Beznosov. "Investigating Users' Perspectives of Web Single Sign-On: Conceptual Gaps and Acceptance Model." ACM Trans. Internet Technol.13. (11, 20013): 2:1-2:35. doi: http://0-doi.acm.org.wncln.wncln.org/10.1145/2532639 (accessed February 10, 2016).

4. Sun, San-Tsai, and Konstantin Beznosov. "The Devil is in the (Implementation) Details: An Empirical Analysis of OAuth SSO Systems." In Proceedings of the 2012 ACM conference on Computer and communications security, pp. 378-390. ACM, 2012. doi = 10.1145/2382196.2382238. (accessed February 14, 2016).

5. Suoranta, Sanna, Asko Tontti, Joonas Ruuskanen, and Tuomas Aura. "Logout in single sign-on systems." In Policies and Research in Identity Management, pp. 147-160. Springer Berlin Heidelberg, 2013. DOI 10.1007/978-3-642-37282-7_14 (accessed February 19, 2016).