# Taxulator – The Smart Way to File Taxes

Elisha Coad
Department of Computer Science
University of Idaho
Moscow, USA
coad1223@vandals.uidaho.edu

Cade Disselkoen
Department of Computer Science
University of Idaho
Moscow, USA
diss7248@vandals.uidaho.edu

Ryan Rapier
Department of Computer Science
University of Idaho
Moscow, USA
rapi3147@vandals.uidaho.edu

Hasan Jamil
Department of Computer Science
University of Idaho
Moscow, USA
jamil@uidaho.edu

## ABSTRACT

For decades, taxes have remained the same. People are expected to keep meticulous record of all their income and spending over the year and then spend hours filling out a tax report. While the effort put into this is commendable, what if there's a better way? In this paper, we describe the implementation of Taxulator [13], a quicker and easier way to file yearly taxes. Through collecting forms from the appropriate parties beforehand, the individual taxpayer is able to file taxes in a matter of minutes.

## CCS CONCEPTS

• **Information systems → Data management design and models**; • **Computing methodologies → Information extraction**; PDF parsing.

## KEYWORDS

tax reports, query language, congregate user system, information extraction, user profile management system

**ACM Reference Format:**
Elisha Coad, Cade Disselkoen, Ryan Rapier, and Hasan Jamil. 2018. Taxulator – The Smart Way to File Taxes. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

## 1 INTRODUCTION

### 1.1 Motivation

In the United States of America, taxes have always been a complex system. It requires much leg work on the individual taxpayer and because of this the process is often dreaded. This is the motivation behind the Taxulator program, to streamline the tax filing process. Our aim is to make so that once the appropriate information is in place, a user can submit their tax report with the click of a button.


*Conference'17, July 2017, Washington, DC, USA*

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

An additional motivation for Taxulator is to bring the myriad of benefits of technology to tax filing. Such benefits include security, clarity, and easy record keeping. By receiving private financial information directly from the source, without the possibility of interception, we provide security. By providing additional information throughout the tax filing process, so that fewer mistakes are made overall, we provide clarity. And we also bring better record keeping, for a digital report is much easier to find and pull up than searching through boxes of old files.

### 1.2 Testing Resources

If you would like to dig deeper into the demo of taxulator yourself, you can find all the code on our GitHub at https://github.com/elishacoad/taxulator, including the latex code of this report. The live prototype can also be found at http://mytax1.kb-projects.com/index.html. Some testing data that might be useful is as follows:

**Table 1: Company Sample Data**

| Name | Email | Password |
|---|---|---|
| University of Idaho | campus@uidaho.edu | uofirocks! |
| Future Design | future@design.com | futureishere77 |
| SEL | company@sel.com | a43cx32gsf |

**Table 2: Tax Payer Sample Data**

| Name | Email | Password |
|---|---|---|
| Henry Jamison | Henry@gmail.com | whataday43 |
| Susie Queen | susie@gmail.com | susieiscool |
| Stewart Rowe | nfay@example.net | fx38f183 |

**Table 3: Tax Payer IDs**

| Name | ID |
|---|---|
| Henry Jamison | 60004 |
| Susie Queen | 60006 |
| Stewart Rowe | 50034 |

## 1.3 Research

Previous research shows there are already a few tax solutions on the market. One can hire a professional tax accountant to do their taxes for them, however this method requires handing off private information and can be a bit costly. There also are also several tax websites which aim to make tax filing an easier process, such as TurboTax and FreeTaxUSA. However, there are very few that use a centralized user filing scheme, meaning that most websites still require the individual taxpayer to upload all of their employer forms, investment interest forms, rental income forms, and the list goes on. What we propose is that these forms are submitted directly by their respective parties, helping reduce the overall amount of confusion and drudgery for the everyday user.

## 1.4 Challenges

As one might expect, a collaborative tax filing process such as the one Taxulator is looking to employ has its challenges. For one, it's a scheme that requires the cooperation of multiple parties to work. For a single tax filing, employers must report salaries, banks must report investment interests, rentals must be reported by the renters and so on. To help speed up its adoption rate, Taxulator would be deployed in controlled areas where the companies and banks in that area are encouraged to adopt Taxulator as part of their process for the yearly tax season. If we can get a few cities fully on-board with the Taxulator system, the hope is that its usefulness will catch on and it will spread outward to other cities.

Another challenge with creating any digital system that deals with finances is security. We will be dealing with highly private information and as such will need highly secure systems. To do this, our goal is to limit our access to the customer's info and documents as much as possible, so that only the parties who need the information will have access. We will also be utilizing top-of-the-line encryption methods so that in the unlikely event there is a security breach, all data will be unrecognizable.

The last challenge that will be discussed is the difficulty of collating data from multiple different sources and documents into one place. There will be multiple forms from multiple enterprises, each requiring different treatment. Not only will the documents need to be scrapped, each type of document will require a custom way to scrap the data. This challenge though can be overcome through simply brute force. We will ensure we have methods of handling each unique form. In the odd case of a form being needed that we don't have, we will give the user an option to flag the case so that we can add the case for future users.

# 2 DESIGN AND ARCHITECTURE

Taxulator will need to collect data from both the individual taxpayer as well as from organizations. Because of this, our website implements two ways to use the website, as an individual and as an organization. The individual's side of the website will give options to file a new tax return, resubmit a tax return, and to review previous filings. The organization side of the website will allow organizations to manage an employee/customer database and to submit forms for specific employees/customers. Both will be assigned a unique 20-digit identifier upon registration.

When a user lands on the site, they will be asked to create an account. This will set up some of the basic information for the user that can be used for the tax filing process and this is also when a unique 20-digit identifier will be assigned. When the user goes to file their taxes, Taxulator will automatically link all appropriate forms (provided by other organizations) to the taxpayer's session. The user can verify that all the right forms were included before carrying on. If anything is missing, the user will have to option to upload the form manually. The user will also have the option to report their own expenses.

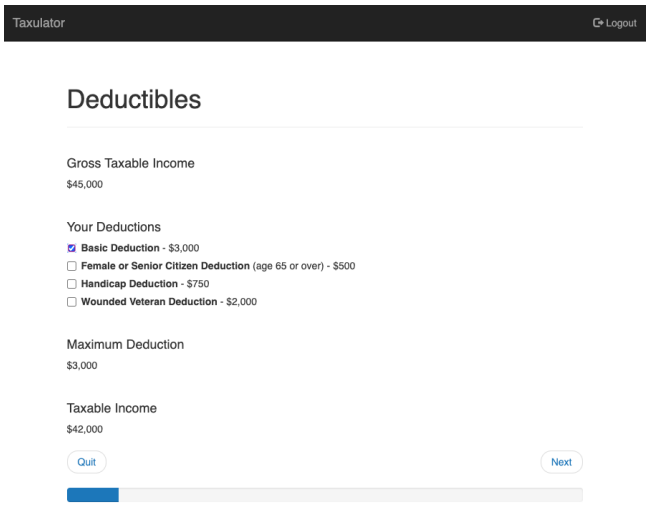


**Figure 1: Tax Payer Deductibles Form**

The next step is to calculate the users adjusted taxable income. To do this, the user will check off which deductions they are eligible for. Then student loans, home mortgage interests, job-related expenses, etc. will be taken into effect producing the user's adjusted taxable income. The adjusted taxable income is then taxed using a graduated scale, and the user is notified of which percentage rate will be applied.

Then the minimum annual city tax is ensured using a database of the city tax minimums in the USA. After this, the user reports their employers, businesses or farms owned, and any houses they own. The user also must report home rentals. The first time any of these are entered they will get added to a database for future use. Once added to the database, the user will be able to select them from a drop-down list.

Lastly, Taxulator will prepare the tax return. It will attach all necessary documents, cross reference incomes and deductions, and verify all reports. Taxulator also will show the taxpayer if they are due for a return and if so of how much. At this point the user can sign and submit the report. If the user wishes to modify the report, they can create a copy of it and then resubmit a new report. All submitted reports are retained and viewable from the dashboard of the website.

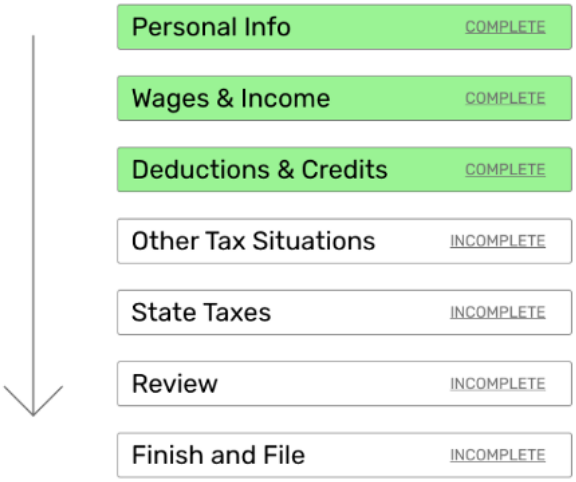


**Figure 2: Tax Report Session Flow**

## 2.1 Database Scheme

Taxulator uses a session-centric database scheme. What this means is that as a user files a tax report, all correlated information will be stored in a session. Upon submission, this session will be added to the session database for future reference and retrieval. Refer to Figure 3 for the database scheme we decided to use.

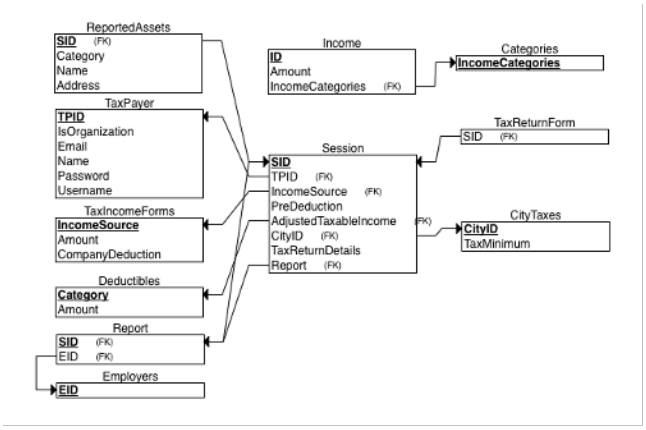


**Figure 3: Session Database Scheme**

We used this scheme because it allows for each category to be compartmentalized, and then retrieved when needed for the session. Since the user will be filing a new tax report every year, every report can just be saved as a session. Now something important to note is that as a user is going through submitting their tax return, all of that session's information will be stored client-side. This achieves multiple ends. First and foremost, if the user decides to quite filing midway, we wanted a way to store the data without entering an incomplete entry into our database. Storing it client-side allows that session to be temporarily suspended until the user decides to continue where they left off or discarded. Upon completion of filing taxes, the client-side session will be submitted as an entry to the database. This minimizes the need to place unstable data within our database.

**Table 4: Tax Payer Table Example**

| PID | IsOrg... | Name | Email | Username | Password |
|---|---|---|---|---|---|
| 232134... | 1 | Payton Mann | yledne... | wilder... | 43d3!S... |
| 412312... | 0 | Juston Schuppe | kutchm... | emardr... | d1560a... |
| 122134... | 0 | Pat Hood | vstant... | kasand... | 66cf0a... |
| 774103... | 1 | Dwight Adams | lauren... | eeichm... | d51347... |

## 2.2 Stack

Our stack implements two core layers, the client and the server. The client can be broken down further into the frontend and the backend. The server manages our database.

On the client side, we used HTML, CSS, and JavaScript to handle the frontend. We also implemented the CSS framework Bootstrap to handle much of the styling. We used PHP on the backend to interface with our MySQL database system.

On the server side, Apache was used for hosting our web server. Using Apache paired with the MySQL database on phpMyAdmin, we were able to get a fairly functional interface between the client and the server.

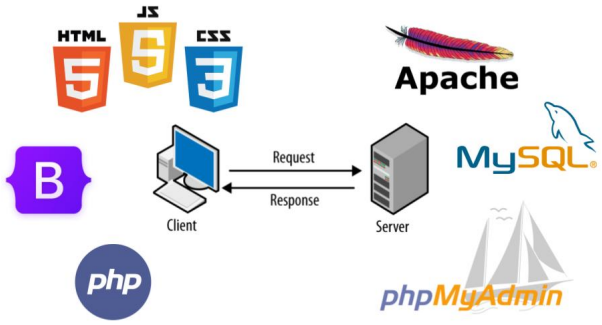


**Figure 4: The Client Server Stack**

To help us get familiar working with such a stack, we created a sandbox page [12], based off of Joel's code. We used this for testing out queries with the data. The sandbox allows for selecting a database to work with, typing in a query, and then submitting and getting the output. This was useful during the development stage of our site as it allowed for quick testing of queries to ensure they worked. This also helped us get a better understanding of how to work with multiple layers of the stack. By using PHP as the middle man, we were able to send queries to our database and then output the results.

## 2.3 Functionality

Taxulator's core functionality is to allow users to file taxes quickly and easily. We have two sides to the website, the individual tax payer side and the company side. Both sides can sign up and log in, and each have their own capabilities. Here are Taxulator's primary features:

- Landing Page
- User Profile System
- User Dashboard
- Business Employee Form
- Data Forms
- Records of Previous Sessions
- Database of Tax-related Information

## 2.4 Challenges

While working on Taxulator, we came across many challenges. These included data generation, connecting to the server, and learning how to work with PHP. Many of the team hadn't had any experience with web programming, so we conducted much research on the subject to help us get started. While the challenges we faced did slow down our development, they also helped us learn how to overcome difficulties and how to be adaptable.

*Data generation.* During the early stages of the project, one of the pressing issues was how we could fill the database with dummy data to test with. After hearing in class that dummy data could be generated with a python package named *Faker*, we began our research. After a couple weeks of research and little progress, it was decided to discard the python package and find a dummy data generator. We tested a few more different generators and settled on one called *FillDB* [6]. The challenge with data generation mainly came from having to spend many hours researching the python program and trying to get it to work. Ultimately, the decision to redirect our course was more beneficial, as it would have likely taken a considerable amount of more time to try and get *Faker* working. After switching to the online data generator *FillDB*, we were able to generate the data within only a couple hours and fill our database with dummy data.

*Connecting to the server.* Later in the project, after setting up the database scheme and filling it with dummy data, our next task was to write PHP and HTML code to connect our database to the user interface. We downloaded FileZilla and attempted to login but got an error that said credentials were incorrect. After attempting to login on multiple different FTP programs and reentering the credentials, we were still rejected. We emailed the professor and subsequently the TA, though communication was slow. Unfortunately, while we tried to get as much as we could done during the meantime, all of our work having to do with the PHP code was practically on hold because we needed access to the server in order to set things up correctly. This slowed down our work considerably, but we tried to do what we could. Finally, a couple of weeks and emails later, we retested the credentials and were able to gain access. We are still unsure as to the reason for what was causing the error, but were thankful it got figured out.

*Lack of sufficient knowledge in PHP.* Our last task was to connect the backend to the frontend through PHP code. None of the team members had had previous knowledge of PHP, so there was a steep learning curve we all went through. We spent many hours with PHP to try to get a sandbox up and running. This unfortunately ended up taking longer than expected, which made it so that we did not have time to implement many of the features we originally planned on completing. However this sandbox [12] was extremely helpful for us to get a grasp of PHP and also helped us write queries for the Taxulator site.

*Generating PDFs.* We started out with the goal to generate a filled out PDF the user can download. We first tried getting to get PDFtk for PHP working. We had some trouble getting installed, and after a couple hours decided to find another PDF generator. We found PDF-LIB, a JavaScript library for generating PDFs. After a bit of work, we were able to generate downloadable PDFs. We have to declare the exact location we want each line of text to appear, but we were glad to get it working. It is possible to use PDF-LIB to markup existing documents, and if we had more time we would have it fill out a Form 1040 with the user's data.

## 2.5 Design Choices

Filing taxes is a complicated system. That's why we wanted Taxulator to be as intuitive as possible. When landing on the Taxulator homepage, the user will have the option to sign up or log in. Right underneath these options will also be a prompt for businesses. That will lead to a similar homepage, but with a sign up and log in for businesses. We chose this design because we realize the bulk of our users will be individual tax payers, so we want it to be quick and easy for them to know what to do. If a business happens upon the website though, we want them to find their portal. Alternatively we also will provide a *taxulator.com/business* url for business to go directly to their side of the website.

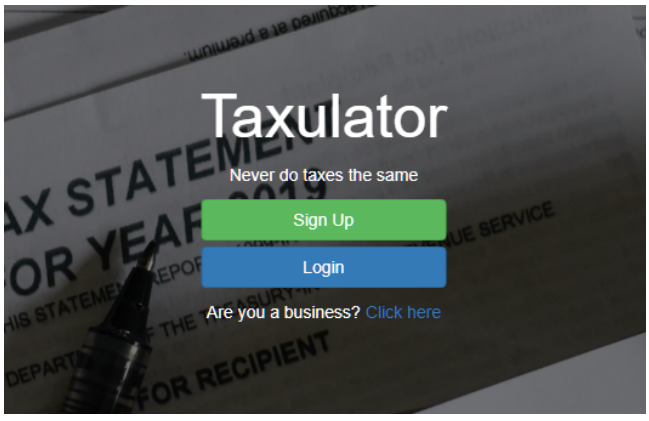


**Figure 5: Taxulator Landing Page**

Once logged in, the user will be taken to their dashboard. This is their personal home page. From here they can start a new tax report or view previous reports. They can also download previous reports in the form of a PDF if needed. We chose a dashboard design

because we needed a place that the user can feel grounded to as they go about filing taxes.

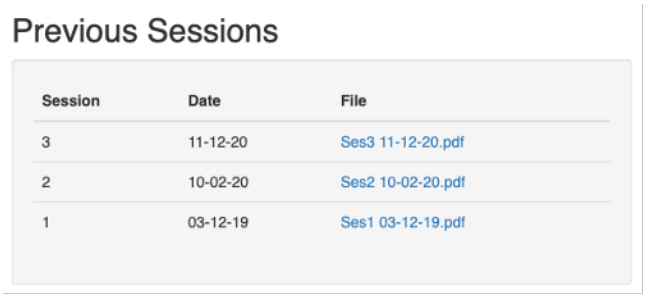


**Figure 6: Dashboard View of User's Previous Sessions**

For the design language, we decided to go with a clean and minimal style. We used an open-source CSS framework called Bootstrap [1] to quickly and efficiently get our designs up and running. Using Bootstrap helped us stay consistent in much of our styling. However, we did still use some CSS for spacing and such where needed.

## 2.6 Platform Selection

When we first started Taxulator, we conducted much research to decide which platforms we were going to be using. With so many options, it was hard to narrow down the best ones. However, we made some choices, so here are the main platforms we decided to go with.

*ERDPlus.* We used a software tool called ERDPlus to generate our Relational Schema diagrams. It's a free tool online which allows for easy database modeling using a cloud-based system. ERDPlus also allowed us to brainstorm out what our primary keys were going to be and how we wanted to structure our database [5].

*FillDB.* FillDB allowed us to be able to quickly generate a large volume of data in MySQL format in order to populate our database. Most of our data was generated this way, entries such as taxpayers, employers, city tax rates, etc [6].

*Trello.* To keep track of who was working on what, and what still needed to get done, we used a team management tool called Trello. Trello made it easy to see who was working on what and also to see what we still had left to get done. We could tag the importance of each task as well as link to useful resources for future reference. This tool helped streamline the development process and provided clarity when there was a plethora of moving parts [14].

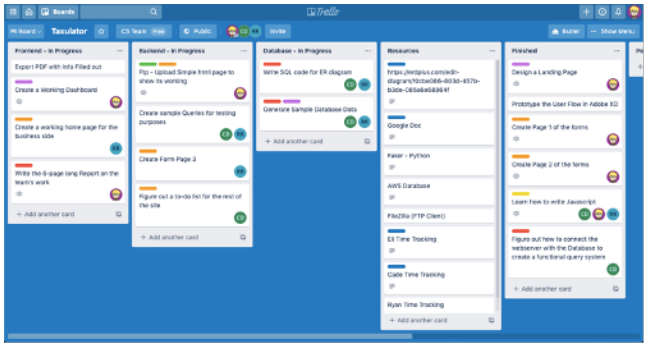

**Figure 7: Snapshot of Our Team's Trello Page**

*Adobe XD.* Adobe XD is a powerful prototyping tool that we used to get an idea of where we were going. View Figure 8 to see the screens made. The blue arrows show the flow of the screens. This allowed us to step through the pages in a prototype before we even got to programming, so we could have a clear picture of where we were headed [16].


**Figure 8: Adobe XD Taxulator Prototype**

*GitHub.* We tried to use the version control software GitHub to track changes made to the code base. Unfortunately, we did not get to implement its use as much as we would have liked, but nonetheless it is still a valuable tool. We added our final code to GitHub for sharing [2]. If we were to carry this project further, we would like to transition to a purely GitHub workflow to keep track of changes made and to have backups of our code base [7].

*phpMyAdmin.* phpMyAdmin is an online software tool we used to handle the administration over our MySQL database. After generating sample data with FillDB, we imported this data into phpMyAdmin to use with our website [11].

*PDF-LIB.* PDF-LIB is a JavaScript library used for generating PDFs. We used it to generate the final tax report and to populate the report with information. It is light weight and was fairly intuitive to use [10].

*Other Tools.* We also used other tools such as Adobe Illustrator [8] and Adobe InDesign [9] to draft up user flows and for making presentations. Along the way, we also used Google Docs [4] for shared notes, MySQL Workbench [15] for testing queries, and Visual Studio Code [3] as our editor.

# 3 CONCLUSION

Taxulator has come a long way. That being said, there's still a lot of work ahead of us to make a final product. There are many limitations to our current design, and many improvements that are in order.

## 3.1 Current Limitations

Currently there are many features that need to be refined in the project. Specifically, connecting the backend to the frontend has been a challenge and has still not been fully implemented. One of the main limitations currently is that the site doesn't have any PDF info extracting capabilities, which would be key for moving forward.

## 3.2 Future Improvements

In order to improve, the first thing we would do is implement more backend to frontend PHP code. We would build out the forms and make sure they are working with the database. We could then work on a PDF generator to output a final PDF tax report when the user finishes their session. Additionally, we could refine the overall interface of the website, ensuring a consistent design language.

Other improvements would include executing a more comprehensive business side to the site. While with a limited time frame we had to prioritize the user side of the site, we recognize that the business side is also an integral part and would like to have built that out more. Also one other large part of the site that we weren't able to get to was the security side of the site. Finances require a high level of security, so we would have liked to spent a fair amount of time on securing users' information in the form of end-to-end data encryption.

We also wanted to add more complementary material to the tax filing process in the form of info pages and tool-tips. With taxes, there are a lot of terms that aren't always intuitive. We want to make these convoluted ideas more clear by giving quick access to definitions and explanations where applicable. This would hopefully take out much of the confusion that comes with filing taxes, especially for new tax filers.

If we were to redo the project, there are a number of things we would have done differently. For one, we could have spent less time trying to get Faker working and just have gone with FillDB from the start. Or at least find someone who could have helped with Faker earlier on to get us started. Additionally, a more base knowledge with PHP would have been helpful, so setting aside time for working on PHP tutorials would have sped up the web building process.

## 3.3 Final Thoughts

Overall, considering where we started, we feel we've made considerable progress on Taxulator. There were many hurdles we had to jump over, including learning new languages, error solving, and working with new software. However, considering how much we learned and the experience we gained through this process, we consider it a success. Each of us now know much more about setting up a full stack database system and we feel we have a solid beginning understanding of web programming as a whole.

# REFERENCES


[1] Bootstrap. [n.d.]. *CSS Framework.* https://getbootstrap.com/
[2] Taxulator Code. [n.d.]. *GitHub Repository.* https://github.com/elishacoad/taxulator
[3] Visual Studio Code. [n.d.]. *Editor.* https://code.visualstudio.com/
[4] Google Docs. [n.d.]. *Collaborative Documentation Software.* https://www.google.com/docs/about/
[5] ERDPlus. [n.d.]. *Schema Generator.* https://erdplus.com/
[6] FillDB. [n.d.]. *Data Generation Software.* http://filldb.info/
[7] GitHub. [n.d.]. *Version Control Software.* https://github.com/
[8] Illustrator. [n.d.]. . https://www.adobe.com/products/illustrator.html
[9] InDesign. [n.d.]. *Design and Presentation Software.* https://www.adobe.com/products/indesign.html
[10] PDF-LIB. [n.d.]. *Javascript PDF Generator.* https://pdf-lib.js.org/
[11] phpMyAdmin. [n.d.]. . https://www.phpmyadmin.net/
[12] Sandbox. [n.d.]. *Query Testing Website.* http://mytax1.kb-projects.com/Sandbox/sandbox.php
[13] Taxulator. [n.d.]. *Project Demo Website.* http://mytax1.kb-projects.com/index.html
[14] Trello. [n.d.]. *Project Management Software.* https://trello.com/
[15] MySQL Workbench. [n.d.]. *Placeholder.* https://www.mysql.com/products/workbench/
[16] Adobe XD. [n.d.]. *Application Prototyping Software.* https://www.adobe.com/products/xd.html