Tryl: Don’t Pay For What You Don’t Use

An exploration of broken free trial tactics and a consumer protection solution

Carina Daidone, Steven Flores, Jonathan Grossman

Table of Contents

I. Current Landscape

Two thirds of Americans use a subscription service for everyday goods, which has created its own economy valued at $1.5 trillion (perkinscoie.com, 2022). Subscription services have become more than just a business model—they're now an entrenched part of American consumer culture. At this point, 75% of direct-to-consumer companies offer these services (Eyal, 2022). At their best, they make life easy through auto payment and recurring delivery or content. They excite evolutionary pleasure points by triggering variability when the box that arrives each month contains a new, special item. 51% of consumers listed their leading reason for purchasing a subscription box was to try new products (Tighe, 2022). Even better, products that used to be bought outright have an additional lure—they're mostly now free, at least in the beginning. Free trials have proliferated and, when done correctly, benefit both the consumer and the company. In the past, a bad product meant wasted money. Now consumers can try before they buy and cancel if the product is not up to snuff.

Companies are no longer able to get away with hoping to pawn off a bad product on an unsuspecting consumer and have been forced to increase the quality of their offerings, knowing that with the costs of acquisition, they will lose money by losing a consumer after the free trial. This results in shady industry practices that keep the consumer around for as long as possible—hiding cancel buttons deep in a website, doing away with cancel buttons altogether and making the person call in. In some cases, write in. The internet is riddled with horror stories of subscriptions that don't have a way of getting canceled. The sneakiness is not from an underbelly of shady websites, it includes upstanding, well-funded, generally rules following companies, from the New York Times to Fabletics. Only an upswell of angry consumers and bad publicity seems to get an otherwise well-intentioned company to change their unctuous subscription practices. The result is that Americans pay for far more subscriptions than they are aware of—a study found that the average consumer thinks they're spending $86 per month on subscriptions, when it's $219 (Waldman, 2022). A study done in Washington projected that 59% of their population had mistakenly enrolled in a subscription plan, and that 100,000 were unable to cancel plans they wanted out of because it was "too difficult" (Takahama, 2022).

Political reaction to one of the largest consumer outrages of our time has been slow. California was early to regulate subscription businesses in 2010, but their response was tempered, and other states have been slow to follow. It's only as of this year that New Jersey has taken the lead on cracking down on the deception tactics subscription companies utilize. Their law that just took effect mandates that companies have online, phone and mail cancellation methods, give clear notice of those, and acknowledge the cancellation. Financially incentivized companies move much faster than politics though, so while this legislation is a good first step, it is more pyrrhic victory than robust consumer protection. Even with stronger legislation though, consumers will still find themselves spending on unwanted subscriptions due to simply forgetting when a free trial ends. We've all done that and realized too late—one of the authors didn't notice as a free month trial for Hulu turned into a $55 per month charge for over a year, a charge that was not wanted nor caught until too late. The problem is so ubiquitous at this point that whether fraud or forgetfulness, most likely you and those around you have experienced a forgotten or nearly impossible to cancel subscription.

There has been some legal remediation recently. Amazon was sued by the European Union for its "dark patterns" in canceling Prime and now has to adhere to a two-click rule for cancellations (Lomas, 2022). The education company Age of Learning is paying $10 million to settle a suit for not properly disclosing a renewing membership fee and forcing consumers to navigate byzantine rules (Lazarus, 2020). But these are small wins in an ocean of losses for consumers. There is an opportunity to pass laws that could restrict companies' abilities to undermine cancellation attempts—for instance, limiting the number of clicks to cancellation or a parity law that mandates online cancellation if the sign-up initially occurs online. However, legislatures, for the most part, are not keen on starting down that road of regulation. People take to the internet to vent their grievances and experiences. Websites dedicated to the topic are a veritable collection of frustrations. The world's "facepalm" hotline, Reddit, has a thread detailing babysitting apps that charge monthly without a booking, children's book apps that continue sending books and charging after cancellation, six months free of a magazine that never arrived while the charges did.

II. Competition

The competitive space has become more crowded over the years, reflecting the increasing problem that uncanceled trials and subscriptions pose and the variety of ways to address it. We can think of the various solutions as functionally covering different corner cases that evolve to evade existing free trial tactics. Think of these approaches as somewhere along the spectrum of escalating functions that play to different parts of one's wallet, whether it's the cards in it or the code that reminds of an impending payment deadline. Something that should be noted is that in the spectrum of solutions, no existing solution automates subscription cancellation.

DoNotPay — Free Trial Surfing

On one side of that spectrum, oddly developed by a software engineer, is Free Trial Surfing. Joshua Browder, a trailblazer in the consumer advocacy space (his original company, DoNotPay, is an AI system that gets people out of unfairly given tickets or fines), started Free Trial Surfing to address the functional gaps of the code-only subscription cancellers. The idea behind Free Trial Surfing is to stop the payment at its source and not deal with the layers built over payments. The company issues a virtual credit card number and a fake name that one uses to sign up for their free trials. Then, when the website attempts to start charging after the free trial ends, the card gets rejected because it has no money on it. Given that most free trials test for sufficient funds when signing up, the company had to pair up with a bank to avoid crashing into the first hurdle (this is not a finance class, though, so we won't go into the financial and banking dimension of this mechanism). Such a technique for this function should incite a cynical reaction—if this prodigy coder needed to go as deep into a bank's portfolio as issuing fake credit cards, does that say something about the utility of companies that rely on slick code to prevent payment?

Mint Premium

Mint uses AI to recognize and inform someone of their subscriptions but stops short of canceling them for the user. Originally a banking app in the money management space, their approach leverages the credit or debit charges a person has already given them access to for their banking needs. Given that they already have all their customers' receipts, this is a product extension of their primary offering. One that required the building of an AI filter that could match recurring dates, company names, and charges to form a high probability guess that the corresponding commerce represents a subscription. An extra layer of complexity is that the AI is not confused by recurring charges and can separate those from subscriptions. Recurring charges are usually necessary, like a heating bill or other utility, or an insurance premium. The product wisely cannot confuse those with a Hulu subscription, as it might render itself useless or at least unusable to weed through.

Bobby subscription tracker

Bobby is a subscription tracker that relies on users to input their various trials and payments. The log is entirely user-generated and allows someone to put in the length of the trial, company, category, and description. It has the added feature of adding up all the monthly payments to give the user a better idea of their cumulative spending. They also notify when a bill is soon to be charged. More of a notification service than actionable features, it demands the user to do much of the initial work for subscription recognition and only gives notifications as a result.

PocketGuard

PocketGuard functionally has subscription tracking features but is primarily an app for people seeking general budgeting help. PocketGuard has the user group their spending into categories, so they can more easily see where their money is going for the purpose of being more aware of the breakdown of their spending. Their core value-add is revealing habits that people don't realize consumes a lot of their income—something as simple as their coffee every morning becomes a large expense in the long run. PocketGuard exists to reveal those budget-busting but overlooked charges. It has the added feature of attempting to lower one's bills and can negotiate cable charges on one's behalf. With respect to subscriptions, it will flag recurring charges in case the user is unaware of that regular spend.

RocketMoney (Formerly TrueBill)

A consummate banking app, RocketMoney has a feature-rich subscription utility. It organizes subscriptions and notifies a user when payments are approaching. It uses AI to recognize recurring payments even if the user didn’t input them. Their primary advantage compared to competitors is that they have a concierge that enables the user to cancel unwanted subscriptions or free trials run wild. An essential element of the subscription canceller is that it uses real people who cancel the subscription, which shows how tricky automated technology is to accomplish that. Considering that RocketMoney is a holistic banking app, the existence of a concierge fits better with their feature set and banking model. It is not compatible with a standalone subscription canceller that doesn't have a multitude of income streams and services. They offer credit scores, mortgages, and personal loans. Their concierges are likely operating at full capacity across all their services, making their subscription cancellation services more of a marginal service. They claim to have canceled over $155 million of subscriptions for their 3.4 million customers. It is a paid service and costs between $3 to $12 per month, depending on the plan.

III. Current Technological Barriers

One of the most significant hindrances to automating subscription cancellation is CAPTCHA. A CAPTCHA is a protocol that protects websites from bots by producing and evaluating tests that humans can pass but current computer programs cannot. For example, a test may include a user choosing images containing boats or reading distorted text and typing the words into a dialogue box, which currently computer programs cannot replicate. Through the process of unsubscribing, a majority of subscription cancellations utilize CAPTCHAs for human authentication. Therefore, without that validation, a computer program could not move forward to finalize or complete the unsubscription.

Another barrier exists around the Login Service API. This API "...provides endpoints to manage the User Session in Conversational Cloud, such as User Login, Application Login, Logout and Refresh." If subscription services were aware that a software as a service application was being created to unsubscribe users from their service, they would not be willing to create partnerships. They would build blockers, use preventative techniques to make it difficult for easy login, and try to create friction in the process.

Another impediment is data privacy. This is becoming an increasing concern for end users across all web-based applications. For one, a person may want to keep their passwords private from another website in case of data breach. However, without passwords being saved, the login process could not happen automatically, and therefore the process would not be able to be completed end-to-end. In addition, users may not want a company to know all of their subscription information. They may be private about their transactions or be apprehensive that their information can be collected as data and sold to third parties for marketing purposes.

IV. Our product: Tryl

The first stage of our product will be a repository of all subscriptions a user has. Once they log in, they can add new subscriptions, including company name, trial duration, and the date the trial ends. After saving this information, this will allow users to see all of their subscriptions. They will also be able to see what trials are active, what trials are inactive, and what trials have become subscriptions. Our software will send out reminder emails when the trial ends so that users can decide if they want to subscribe to the service or terminate the trial before their paid subscription starts.

The next phase would be to automate filling in the subscription form. The ideal situation is if a user signs up for a trial and all of the information is stored to Tryl automatically. Two different approaches could accomplish this. One option would be to have an integration with websites allowing users to check a box, and all of their information would be migrated to Tryl. The other option would be for users to forward their trial subscription confirmation email to a general inbox, and Tryl could scrub all the pertinent information into the database.

The last phase would be to automate the unsubscription process. This is where we find most of the challenges requiring the most work. We would need to solve the human authentication issue that CAPTCHA is currently working to protect. In the final stage of Tryl, users could sign up for trials, have all of their subscription details stored automatically, and then decide whether or not they would like Tryl to cancel their trial for them automatically before the paid subscription starts. This would be a carefree seamless process for the user that is concerned that they sign up for trials and end up being subscribed to services for an indefinite period before realizing how much money they have been paying.

V. Product Development

Phase 1: Design and Proof of Concept

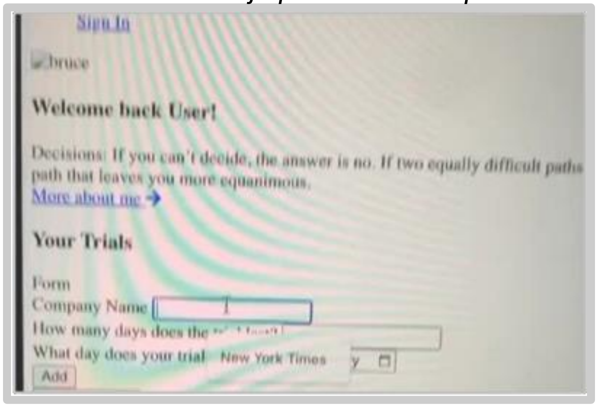
The beginning of the development of this project included three parts:

- The creation of the front end
- The creation of the back end
- Connection to MongoDB

Front-End Creation

After brainstorming potential product features and creating a mock-up of Tryl, we choose a free template design from Creative Tim that utilizes Bootstrap. Bootstrap allowed us to manage our time efficiently by spending less time on HTML/CSS details and more time on the JavaScript elements the app would require. The first page built was index.html, which contains a form to collect data and a table to display that data.

Index.html/index.ejs proof of concept:



Sign In

bruce

Welcome back User!

Decisions: If you can't decide, the answer is no. If two equally difficult paths path that leaves you more equanimous.

More about me →

Your Trials

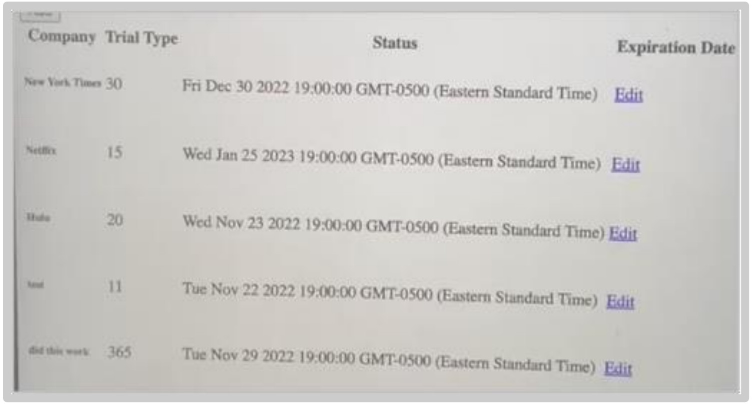
Form

Company Name

How many days does the

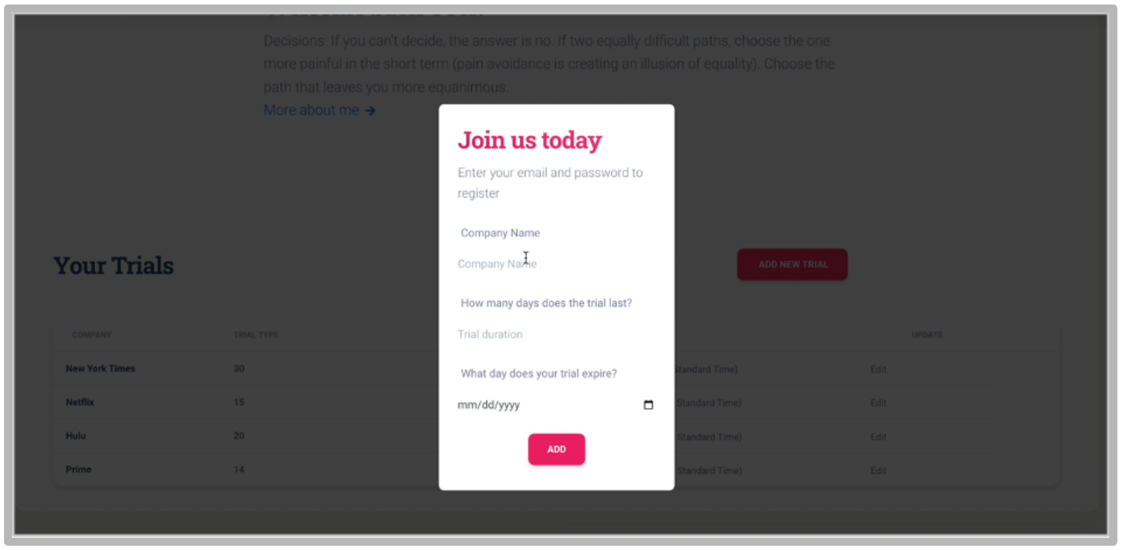
What day does your trial New York Times

Add



Company	Trial Type	Status	Expiration Date
New York Times	30	Fri Dec 30 2022 19:00:00 GMT-0500 (Eastern Standard Time)	Edit
Netflix	15	Wed Jan 25 2023 19:00:00 GMT-0500 (Eastern Standard Time)	Edit
Hulu	20	Wed Nov 23 2022 19:00:00 GMT-0500 (Eastern Standard Time)	Edit
test	11	Tue Nov 22 2022 19:00:00 GMT-0500 (Eastern Standard Time)	Edit
did this work	365	Tue Nov 29 2022 19:00:00 GMT-0500 (Eastern Standard Time)	Edit

Index.html/index.ejs with CSS applied:



Decisions: If you can't decide, the answer is no. If two equally difficult paths, choose the one more painful in the short term (pain avoidance is creating an illusion of equality). Choose the path that leaves you more equanimous.

More about me →

Your Trials

ADD NEW TRIAL

COMPANY	TRIAL TYPE		UPDATE
New York Times	30	tandard Time)	Edit
Netflix	15	Standard Time)	Edit
Hulu	20	Standard Time)	Edit
Prime	14	Standard Time)	Edit

Join us today

Enter your email and password to register

Company Name

Company Name

How many days does the trial last?

Trial duration

What day does your trial expire?

mm/dd/yyyy

ADD

Back-End Creation

The first goal was to create basic functionality that provided a foundation for the concept of Tryl, including the form and table created within index.html. We identified that we needed the ability to process data and identified potential JavaScript libraries we could use, such as Node, React, and Ruby on Rails. Based on research, we narrowed down the options to Node due to the number of tutorials that were available to reference. We initially targeted MySQL to host the database; however, we ran into issues getting a connection and establishing a connection through the console. We came across a tutorial that used Node, Express, and MongoDB. This tutorial influenced our decision to use MongoDB instead of MySQL. From this tutorial, we established the following:

- Installing and utilizing Homebrew in the console helped install the necessary dependencies without issue.
- A challenge at this phase included re-learning how to use the console, such as installing dependencies, connecting to the server, and other commands.
- Walked through creating a free cluster on MongoDB and creating the Tryl collection.
- Established connection with MongoDB collection by defining the collection URI in a config file called keys.js (config/keys.js). Called the URI in server.js and created a connection using mongoose.
    - Created the Data Schema for trial data collection that defined the fields and the data type.
    - Established routes using **app.post** to send data to MongoDB and **app.get** to retrieve data from MongoDB.

```
app.get("/", function(req, res) {
  Tryl.find({}, function(err, tryls) {
    res.render('index', {
      trialList: tryls
})
  })
});

app.post("/", async function(req, res) {
  let newTryl = new Tryl({
    companyname: req.body.companyname,
    trialduration: req.body.trialduration,
    trialexpiration: req.body.trialexpiration
  });
  newTryl.save();
  res.redirect('/');
});
```

- This tutorial introduced EJS (Embedded JavaScript Templates), and we changed all .html files to *.ejs*. Set the app engine in *server.js* to EJS to enable the use of *.ejs* files. Then used EJS tags to display data from MongoDB within the table in *index.ejs*. [code image]
- Utilizing **console.log** helped with troubleshooting errors. It often resulted in Google and Stack Overflow searches. ChatGPT was also helpful at times. For instance, I added part of the code from server.js that was resulting in an error into the search bar of ChatGPT. ChatGPT returned an explanation and the suggestions for how to resolve the error. It did not always provide a great solution, but often helped point me in the right direction as topics to search for on Stack Overflow.

Connecting to MongoDB

Establishing the connection to MongoDB was challenging, it took three days of finding the best-fitting tutorial to result in a functioning connection and flowing data from form to database to table. We learned that sending data to and from MongoDB requires:

- Mongoose is a JavaScript library that facilitates a connection between MongoDB and a Node.js app.
- Establishing a data schema that can be defined in server.js or within a models folder as a JavaScript file. The fields created in the schema must match the name defined in the fields of a form.
- App.get and app.post – both mentioned earlier – were key to the transfer of data to and from the app.

- Establishing the URI can be done within server.js, but it makes it easier to read and find if stored in a separate JavaScript file. We organized it where keys.js lived under a config folder.

Keys.js

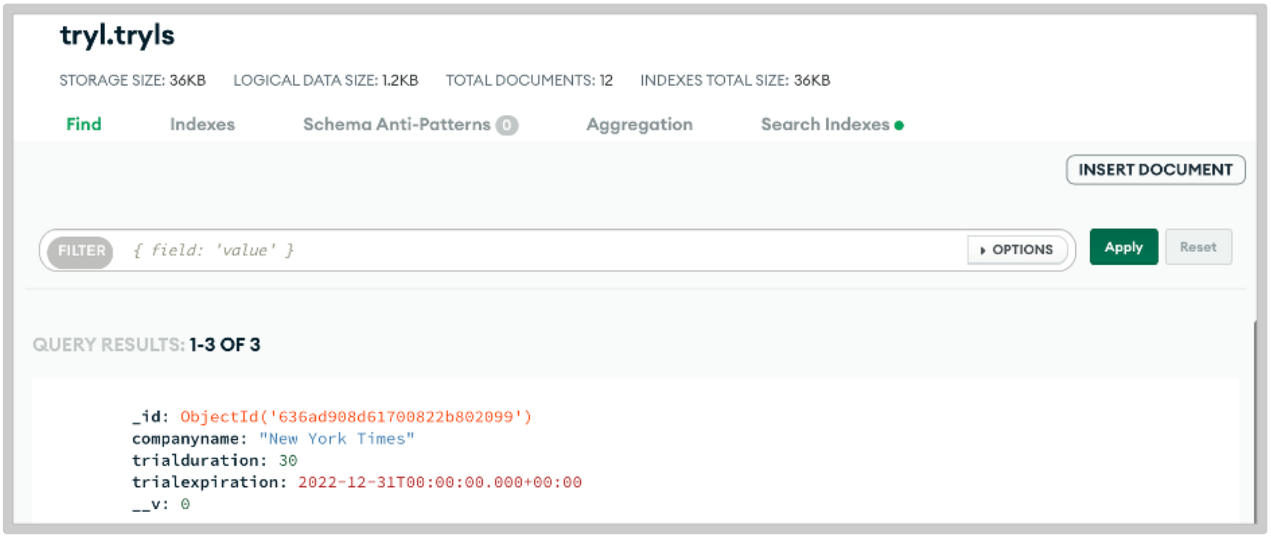
```
module.exports = {
   MongoURI:
'mongodb+srv://cdaidone:4RMb0merkMaqs0Et@cluster0.rdjhtko.mongodb.net/tryl'
}
```

Calling URI into server.js and creating connection:

```
const db = require('./config/keys').MongoURI;

// NEW Connect to MongoDB Database
mongoose.connect(db, { useNewUrlParser: true, useUnifiedTopology: true })
   .then(() => console.log('MongoDB Connected'))
   .catch(err => console.log(err));
```

Example of what a form submission looks like in the tryls collection:



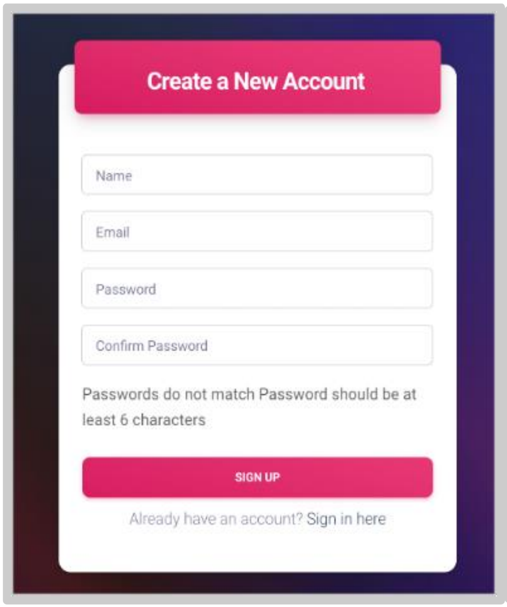
```
_id: ObjectId('636ad908d61700822b802099')
companyname: "New York Times"
trialduration: 30
trialexpiration: 2022-12-31T00:00:00.000+00:00
__v: 0
```

Phase 2: User Creation, Log in, and Logout

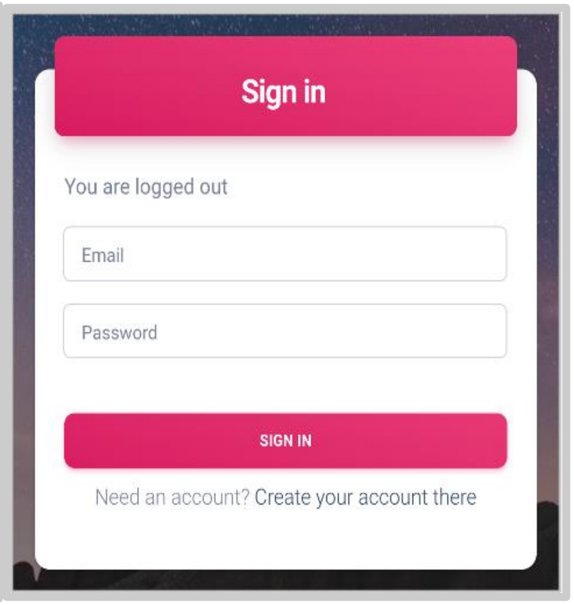
With the foundation of the app functioning, we built the remaining pages: home, login, and register. To facilitate the creation of a user and enable the ability to log in and out, we needed a tutorial that included the components already utilized: node, express, and MongoDB. Like in phase one, this process took a few days to find the correct tutorial, and the Passport.js documentation did not include guidance on setting up authentication with Passport and MongoDB. However, we did find a great YouTube tutorial that helped tremendously with phases 2 and 3. With this guidance, we created the following:

- Model for user data created under models/User.js.
- Utilized Passport.js to set up multiple components for creating a new user and user log in. The users.js file was created within the routes folder to define routes related to both functions. Error messages were also defined:
  - Email field must require an email address containing @ symbol, and the email address must not already exist within the database.
  - Password and Password2 fields must match and be at least six characters.
  - Utilized bcrypt to encrypt passwords and display them as a hash when entered in the register or log in forms.
  - Created login and logout handle. Logout took troubleshooting, but we identified that it needed redirecting to /users/login instead of /logout.
  - Config/passport.js contains the definition of the log in rules to check if an email was registered and the entered password was correct.
  - Used Connect-Flash to display the error/success messages created. We initially tried to style these messages with Bootstrap alerts; however, while the appropriate error message pulled through the EJS tag, it also showed all HTML code around it meant for styling. We tried to find a solution and proceeded without the Bootstrap alert styling.

Flash messages



Passwords do not match Password should be at least 6 characters



You are logged out

With the ability to create a user and log in completed, the app needed authentication established to require a logged-in user to access index.ejs. Config/auth.js defines ensureAuthenticated and forwardAuthenticated. Auth.js is called into the necessary JavaScript files to enable the assignment of the authentication variables to the appropriate routes.

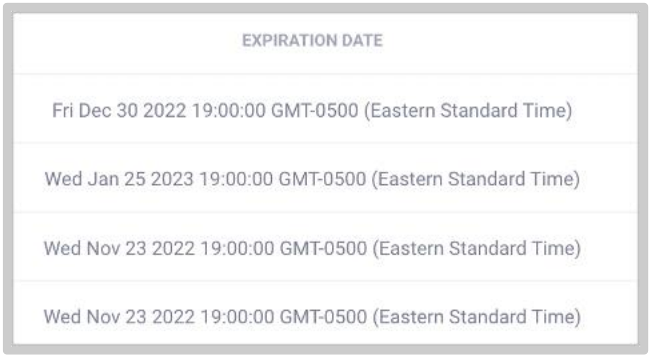
Authentication

```
module.exports = {
   ensureAuthenticated: function(req, res, next) {
      if(req.isAuthenticated()) {
         return next();
      }
      req.flash('error_msg', 'Please log in to view this resource');
      res.redirect('/users/login');
   },
   forwardAuthenticated: function(req, res, next) {
      if (!req.isAuthenticated()) {
            return next();
         }
         res.redirect('/index');
      }
   };
```

The tutorial used express layouts with a master layout.ejs file in the views folder and utilized a tag - <% body %> - to pull in content from other .ejs files like home and login. This update required consolidating all CSS and JavaScript files in the layout.ejs file. However, because the CSS and JS files were stored locally, the other .ejs files outside of layout.ejs could not pull in these files, resulting in broken pages. To resolve this, we uploaded all images, CSS, and JS files to Google Drive and referenced the share link for each file in the code. After implementing, we noticed random occurrences where the CSS and JS scripts would stop working, and an error would display saying that trialList was not defined. At this point, this issue only occurred randomly, and we continued using Google Drive links.

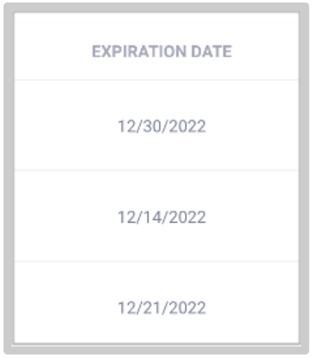
Installed moment.js to format the expiration date in the trial data table in index.js. This library changed the date from displaying as date/time/timezone to just date (mm/dd/yyyy). However, we could not resolve an issue where the date in the table was displayed one day before the actual date submitted to and stored in MongoDB. JavaScript does not treat the date as a calendar date, and the fix requires using .toISOString. Due to time, we left the date as-is.

Before Moment.js:



EXPIRATION DATE
Fri Dec 30 2022 19:00:00 GMT-0500 (Eastern Standard Time)
Wed Jan 25 2023 19:00:00 GMT-0500 (Eastern Standard Time)
Wed Nov 23 2022 19:00:00 GMT-0500 (Eastern Standard Time)
Wed Nov 23 2022 19:00:00 GMT-0500 (Eastern Standard Time)

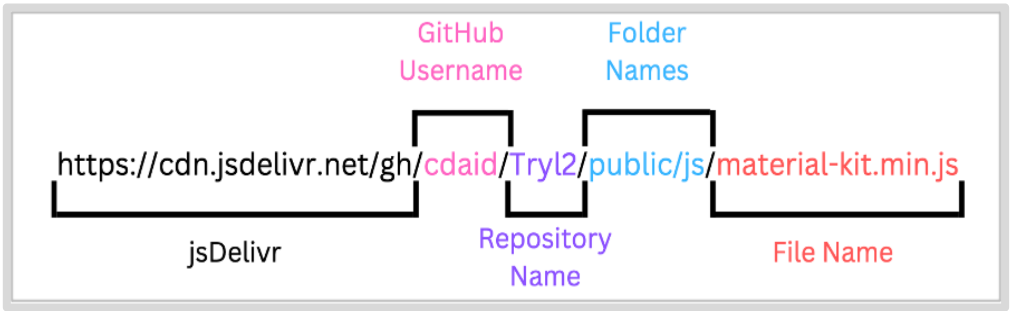
After Moment.js:



EXPIRATION DATE
12/30/2022
12/14/2022
12/21/2022

Phase 3: User-Specific Data Tables and App Fine Tuning

The next step for Tryl included associating trial form submissions with the signed-in user. However, the Google Drive link issue in Phase 2 continued and increased in occurrence, causing significant delays. We needed to find a better solution and one that was free. Research for free CDN options provided many options, and we chose JSDELIVR due to its popularity within the software development industry and integration with GitHub. Creating a JSDELIVR link requires including the GitHub username, repository name, and file path. Using JSDELIVR as the CDN provided stability for the app, stopped recurring errors, and led to all images loading faster.

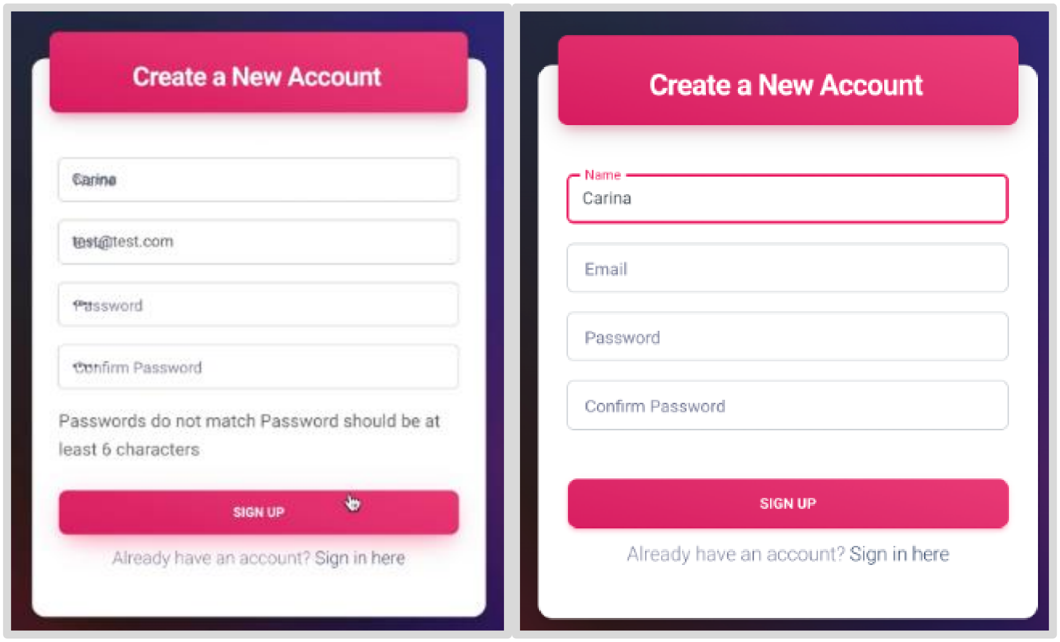


https://cdn.jsdelivr.net/gh/cdaid/Tryl2/public/js/material-kit.min.js

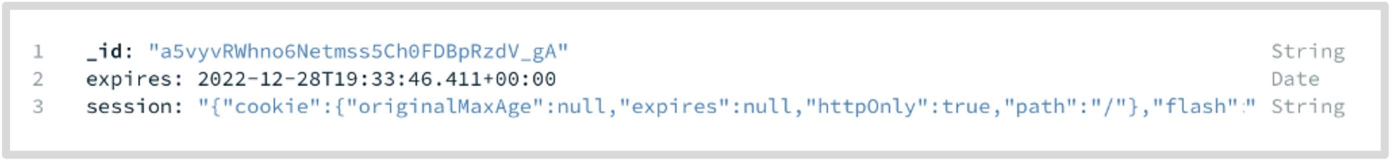
Updating all image, CSS, and JS link sources in the ejs files led to discovering a missing JS file. Resolving this led to fixing the signup form where the placeholder text was not disappearing whenever text was entered into a field.

With the link issue resolved, we leveraged connect-mongo and sessions to create user sessions (cookies), requiring a user.id to be passed with a user log in and trial data submissions.

Resolved form placeholder text issue:

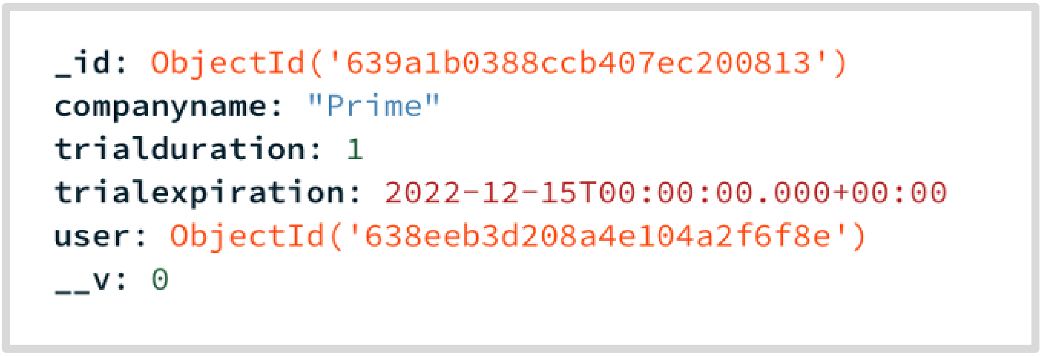

Create a New Account
Carina
test@test.com
Password
Confirm Password
Passwords do not match Password should be at least 6 characters
SIGN UP
Already have an account? Sign in here
Create a New Account
Name
Carina
Email
Password
Confirm Password
SIGN UP
Already have an account? Sign in here

Session stored in MongoDB:



```
_id: "a5vyvRWhno6Netmss5Ch0FDBpRzdV_gA"                                          String
expires: 2022-12-28T19:33:46.411+00:00                                            Date
session: "{"cookie":{"originalMaxAge":null,"expires":null,"httpOnly":true,"path":"/"},"flash":"  String
```

User ID passed with form submission in MongoDB:



```
_id: ObjectId('639a1b0388ccb407ec200813')
companyname: "Prime"
trialduration: 1
trialexpiration: 2022-12-15T00:00:00.000+00:00
user: ObjectId('638eeb3d208a4e104a2f6f8e')
__v: 0
```

We faced multiple challenges at this stage as the tutorial we followed was two years old and included code for connect-mongo that has since been depreciated. We were able to reference documentation to include the correct code.

The hours lost due to the Google Drive link issue were critical; that time could have been used to create the if-else EJS statement needed in index.ejs to filter the data table to only show results for the signed-in user. Email notifications could have followed with a SendGrid integration. However, as it stands today, Tryl allows for users to create a new account, log in, submit trial information, view trial information, and logout.

File Directory

Folder: Tryl

- **Config**
    - auth.js
    - Keys.js
    - Passport.js
- **Models**
    - User.js
    - userModel.js
- **Public**
    - Css
    - Fonts
    - Img
    - Js
    - Scss
- **Routes**
    - Index2.js
    - Users.js
- **Views**
    - **Error**
        - 404.js
        - 500.js
    - **Partials**
        - Messages.js
    - Home.ejs
    - Index.ejs
    - Layout.ejs
    - Login.ejs
    - Register.ejs
    - Package.json
    - server.js

GitHub Link: https://github.com/cdaid/Tryl2

VI. Conclusion

For this project, we built a product that can be one piece of the larger correction we hope to see in the subscription tactics realm. Legislation tends to play catch up while consumers are left high and dry. So we see our competitors as compatriots in the battle for consumer protection, all operating at different access points. The ultimate goal is to create a product that autonomously cancels subscriptions for the consumer. Still, as outlined, some details currently hamper that process, and other technology needs to be explored to reach that level of responsiveness. In the meantime, we see awareness coupled with the various products currently on the market as important steps forward, with many to follow.

Bibliography

1. Perkins Coie LLP. “New Jersey to Regulate Automatically Renewing Subscription Services.” Perkins Coie LLP. Retrieved on October 24, 2022 from https://www.perkinscoie.com/en/news-insights/new-jersey-to-regulate-automatically-renewing-subscription-services.html, October 13 2022.

2. Eyal, Nir. “3 Reasons Subscription Services Fail.” Harvard Business Review. Retrived on October 22, 2022 from https://hbr.org/2022/10/3-reasons-subscription-services-fail, October 11 2022.

3. Tighe, D. “Main Reasons Why Consumers Used Subscription Boxes in 2022.” Statista. Retrieved on October 27 2022 from https://www.statista.com/statistics/1004469/main-reasons-why-us-consumers-used-subscription-boxes/. August 2, 2022.

4. Waldman, Ed. “6 Services to Help You Cancel Your Subscriptions.” Aarp.org. Retrieved on October 26, 2022 from https://www.aarp.org/home-family/personal-technology/info-2022/subscription-cancel-services.html, June 29 2022.

5. Takahama, Elise. “Millions in WA May Have Enrolled in a Subscription Service By Accident.” Seattle Times. Retrieved on October 26, 2022 from https://www.seattletimes.com/seattle-news/millions-in-wa-may-have-enrolled-in-a-subscription-service-on-accident/, October 12, 2022.

6. Lomas, Natasha. “Amazon Agrees to Drop Prime Cancellation ‘Dark Patterns’ in Europe.” Retrieved on October 24, 2022 from https://techcrunch.com/2022/07/01/amazon-ends-prime-cancellation-dark-patterns-europe, July 1 2022.

7. Lazarus, David. “Canceling Subscriptions Shouldn’t Be Like Running an Obstacle Course.” LA Times. Retrieved on October 29, 2022 from https://www.latimes.com/business/story/2020-10-27/column-canceling-subscriptions, October 27, 2020.