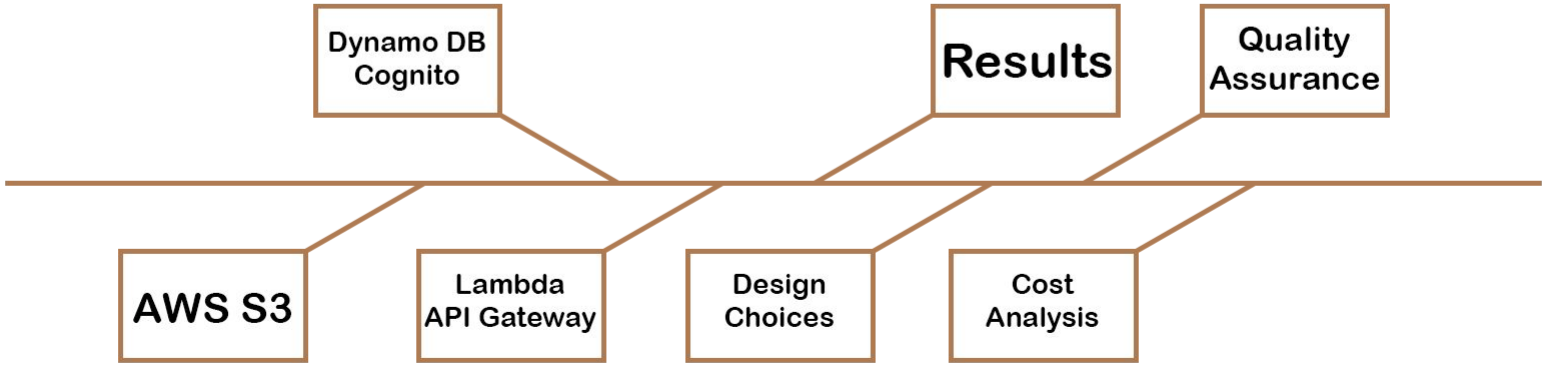
CS55A

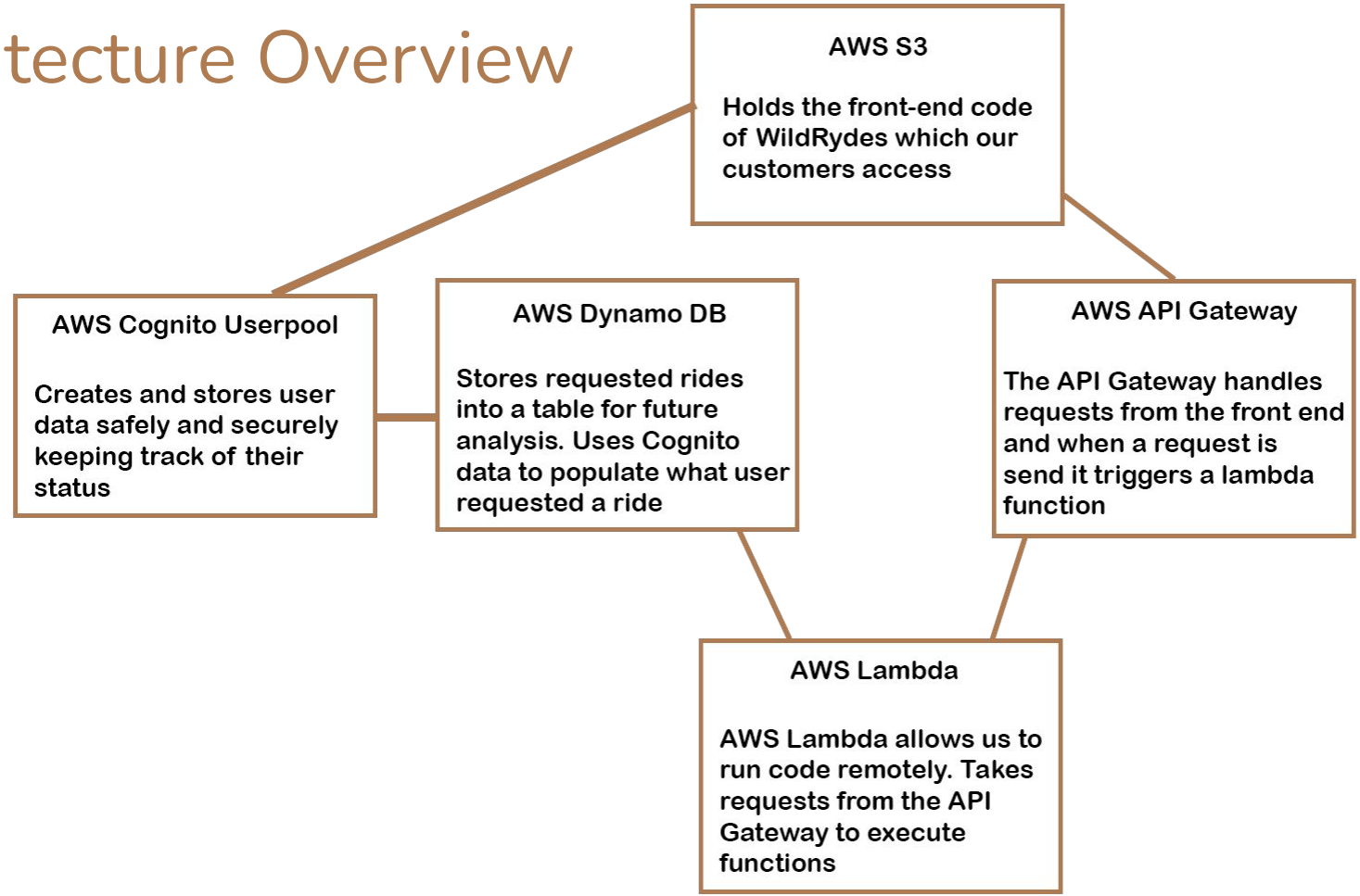
Jason Palmeri

Today’s Presentation

In this presentation, we will take an in-depth look at WildRydes’ first production build. We will explore the system architecture that powers out backend and enables our customers to access our services. We will discuss the various components of our system and how they work together to provide a seamless experience for our users. Join me as we delve into the inner workings of WildRydes and discover what makes our service unique.



Architecture Overview



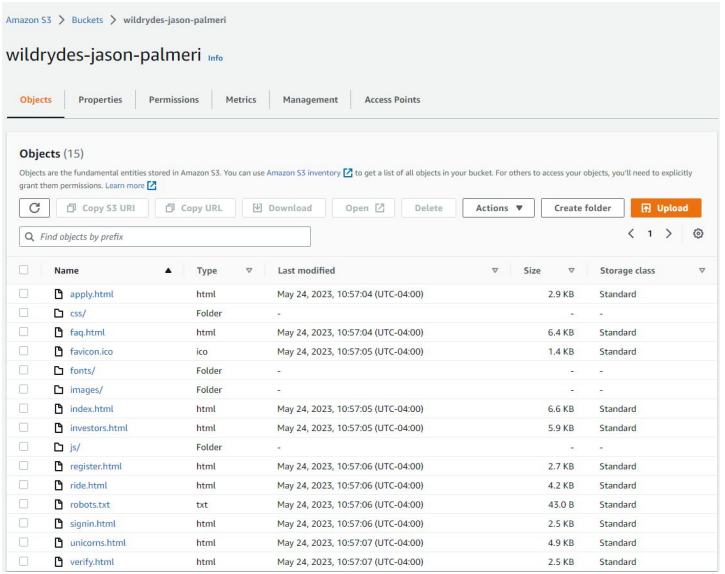
Building the Platform - AWS S3

Amazon Simple Storage Service (S3) is a scalable, durable, and highly available storage service that allows us to store files on the cloud, and enable static website hosting.

S3 provides us with all the tools we need to upload, download and configure our web hosting to get WildRydes up and running within minutes.

With S3's scalability and durability we should expect smooth transitions to larger data sets and more users, while keeping costs low on a pay-as-you-go model

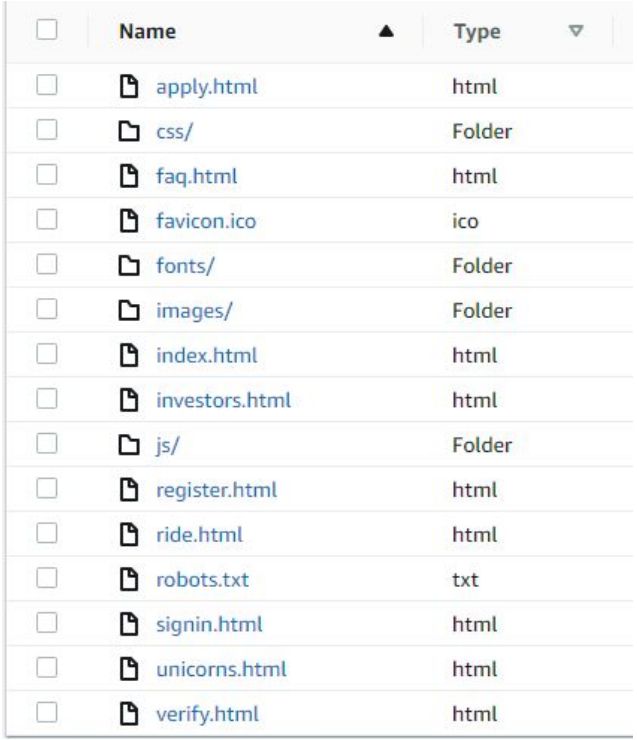
http://wildrydes-jason-palmeri.s3-website-us-east-1.amazonaws.com/



AWS S3 - In-Depth

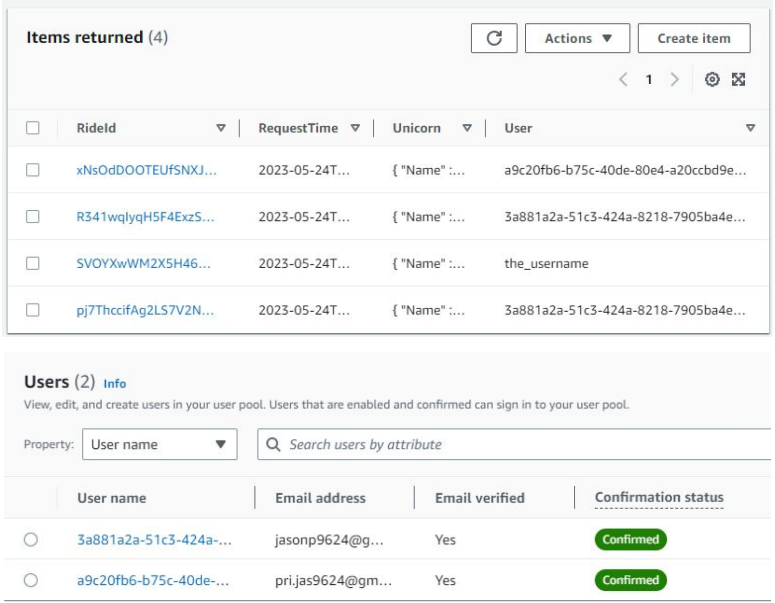
AWS S3 is one of many ways to host and deploy a website or web application. For WildRydes we use S3 as our core hosting service because WildRydes front-end is static, meaning there isn't any server-side code being ran.

AWS S3 hosts our web applications file on the cloud, this includes all images, fonts, layout styles, and web pages. Having our files hosted on the cloud allows our developers to easily access code from all around the world.



Name	Type
apply.html	html
css/	Folder
faq.html	html
favicon.ico	ico
fonts/	Folder
images/	Folder
index.html	html
investors.html	html
js/	Folder
register.html	html
ride.html	html
robots.txt	txt
signin.html	html
unicorns.html	html
verify.html	html

Building the Platform - DynamoDB and Cognito Users



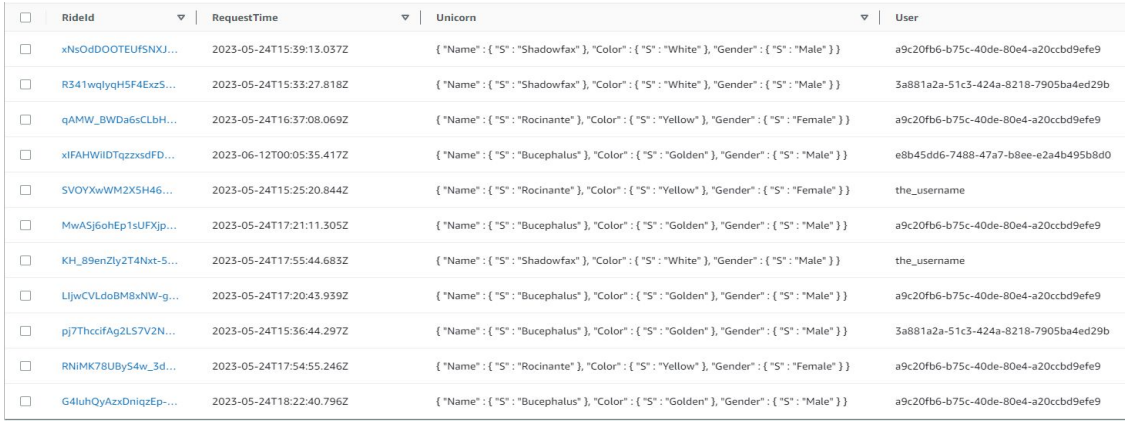
DynamoDB is one of AWS's database services that allows us to store Ride data for future analysis and support for users. Within our Rides database we store all user transactions, detailing the request time, unicorn information and the user that requested the ride.

User data stored in the Ride database comes from AWS's Cognito User Pool, which allows us to setup user registration and sign-in seamlessly into the WildRydes application, with built in confirmation and email services.

Together these services allow us to track users who request rides, as well as allow new users to register.

AWS DynamoDB In-Depth

DynamoDB stores every ride that is requested by users. It stores the information into the Rides table. Within the Rides Table we have a certain structure that is used to keep things uniform. First we store a RideId, which is a unique identifier for each requested ride. Next we store a RequestTime, which is helpful for our support team to see when a ride was requested. Next we have the Unicorn field, this stores the unicorn that responded to the request, again great for the support team. Finally we store the User's username, which comes from Cognito. In our case the username is similar to the RideId where it is just a unique identifier, unlike some companies that allow users to choose a custom username



RideId	RequestTime	Unicorn	User
xNsOdDOOTEUfSNXJ...	2023-05-24T15:39:13.037Z	{ "Name" : { "S" : "Shadowfax" }, "Color" : { "S" : "White" }, "Gender" : { "S" : "Male" } }	a9c20fb6-b75c-40de-80e4-a20ccbd9efe9
R341wqlyqH5F4ExzS...	2023-05-24T15:33:27.818Z	{ "Name" : { "S" : "Shadowfax" }, "Color" : { "S" : "White" }, "Gender" : { "S" : "Male" } }	3a881a2a-51c3-424a-8218-7905ba4ed29b
qAMW_BWDa6sCLbH...	2023-05-24T16:37:08.069Z	{ "Name" : { "S" : "Rocinante" }, "Color" : { "S" : "Yellow" }, "Gender" : { "S" : "Female" } }	a9c20fb6-b75c-40de-80e4-a20ccbd9efe9
xIFAHWiIDTqzzxsdFD...	2023-06-12T00:05:35.417Z	{ "Name" : { "S" : "Bucephalus" }, "Color" : { "S" : "Golden" }, "Gender" : { "S" : "Male" } }	e8b45dd6-7488-47a7-b8ee-e2a4b495b8d0
SVOYXwWM2X5H46...	2023-05-24T15:25:20.844Z	{ "Name" : { "S" : "Rocinante" }, "Color" : { "S" : "Yellow" }, "Gender" : { "S" : "Female" } }	the_username
MwASj6ohEp1sUFXjp...	2023-05-24T17:21:11.305Z	{ "Name" : { "S" : "Bucephalus" }, "Color" : { "S" : "Golden" }, "Gender" : { "S" : "Male" } }	a9c20fb6-b75c-40de-80e4-a20ccbd9efe9
KH_89enZly2T4Nxt-5...	2023-05-24T17:55:44.683Z	{ "Name" : { "S" : "Shadowfax" }, "Color" : { "S" : "White" }, "Gender" : { "S" : "Male" } }	the_username
LIjwCVLdoBM8xNW-g...	2023-05-24T17:20:43.939Z	{ "Name" : { "S" : "Bucephalus" }, "Color" : { "S" : "Golden" }, "Gender" : { "S" : "Male" } }	a9c20fb6-b75c-40de-80e4-a20ccbd9efe9
pj7ThccifAg2LS7V2N...	2023-05-24T15:36:44.297Z	{ "Name" : { "S" : "Bucephalus" }, "Color" : { "S" : "Golden" }, "Gender" : { "S" : "Male" } }	3a881a2a-51c3-424a-8218-7905ba4ed29b
RNiMK78UByS4w_3d...	2023-05-24T17:54:55.246Z	{ "Name" : { "S" : "Rocinante" }, "Color" : { "S" : "Yellow" }, "Gender" : { "S" : "Female" } }	a9c20fb6-b75c-40de-80e4-a20ccbd9efe9
G4IuhQyAzxDniqzEp-...	2023-05-24T18:22:40.796Z	{ "Name" : { "S" : "Bucephalus" }, "Color" : { "S" : "Golden" }, "Gender" : { "S" : "Male" } }	a9c20fb6-b75c-40de-80e4-a20ccbd9efe9

AWS Cognito In-Depth

AWS Cognito is our all in one user management system, and handles all of our needs for the time being. Cognito is built into our application using their JavaScript libraries to provide user registration, user logins, and user authentication.

With AWS Cognito we can see a list of our users, and their statuses, only allowing logged in, and confirmed users access the application.

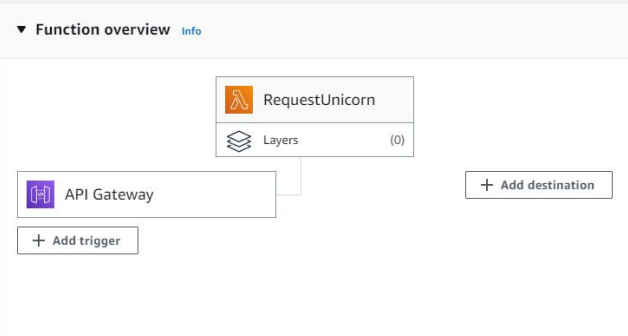
Cognito will help us to support our customers

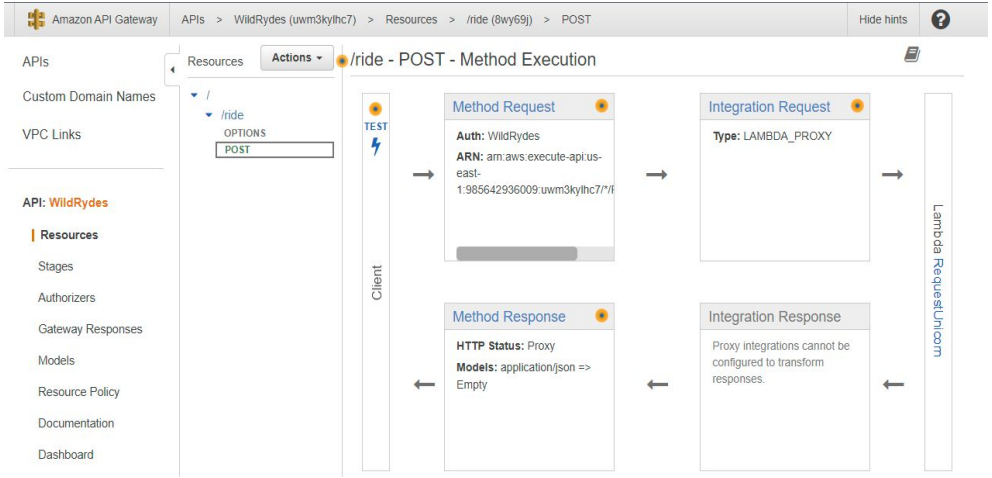
Who are having issues by allowing our team to see

Their current status and other user information

Building the Platform - Lambda and the API Gateway

AWS Lambda is a service that allows us to run code based on certain actions, or triggers that occur. With the AWS API Gateway we can send ride requests from the front-end via JavaScript. The API then triggers which calls to the Lambda functions, which will generate ride data for our users.





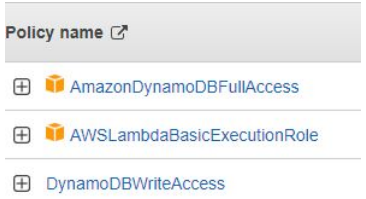
AWS Lambda - In-Depth

AWS Lambda is an important platform for WildRydes as it handles our database writing, and also sends data to our web application based on API triggers.

Currently our Lambda platform holds the RequestUnicorn function, which when requested by our API, generates a random unicorn from the fleet, and writes the ride request data to our database, and finally sends the user the information on their requested ride.

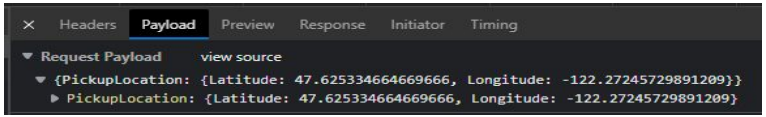
```
function recordRide(rideId, username, unicorn) {
    return ddb.put({
        TableName: 'Rides',
        Item: {
            RideId: rideId,
            User: username,
            Unicorn: unicorn,
            RequestTime: new Date().toISOString(),
        },
    }).promise();
}
```

Within AWS Lambda we use a custom IAM role for our Lambda: WildRydesLambda. This role allows us to access other AWS services such as DynamoDB giving us full access to create, read, update, and delete entries as needed. Additionally this role allows us to execute and write to CloudWatch for logging



AWS API Gateway - In-Depth

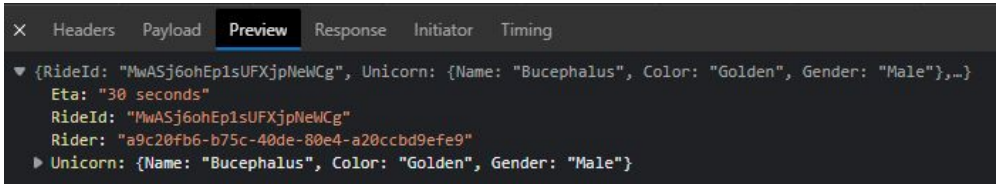
The AWS API Gateway allows us to make calls to other services and functions to do certain tasks. In WildRydes we use the /ride route to make a request to our lambda function. This goes through a few steps to get the requested data:



1. Make a POST request to /ride with PickupLocation
2. The POST request then sends this data to our findUnicorn function within our Lambda to find a unicorn

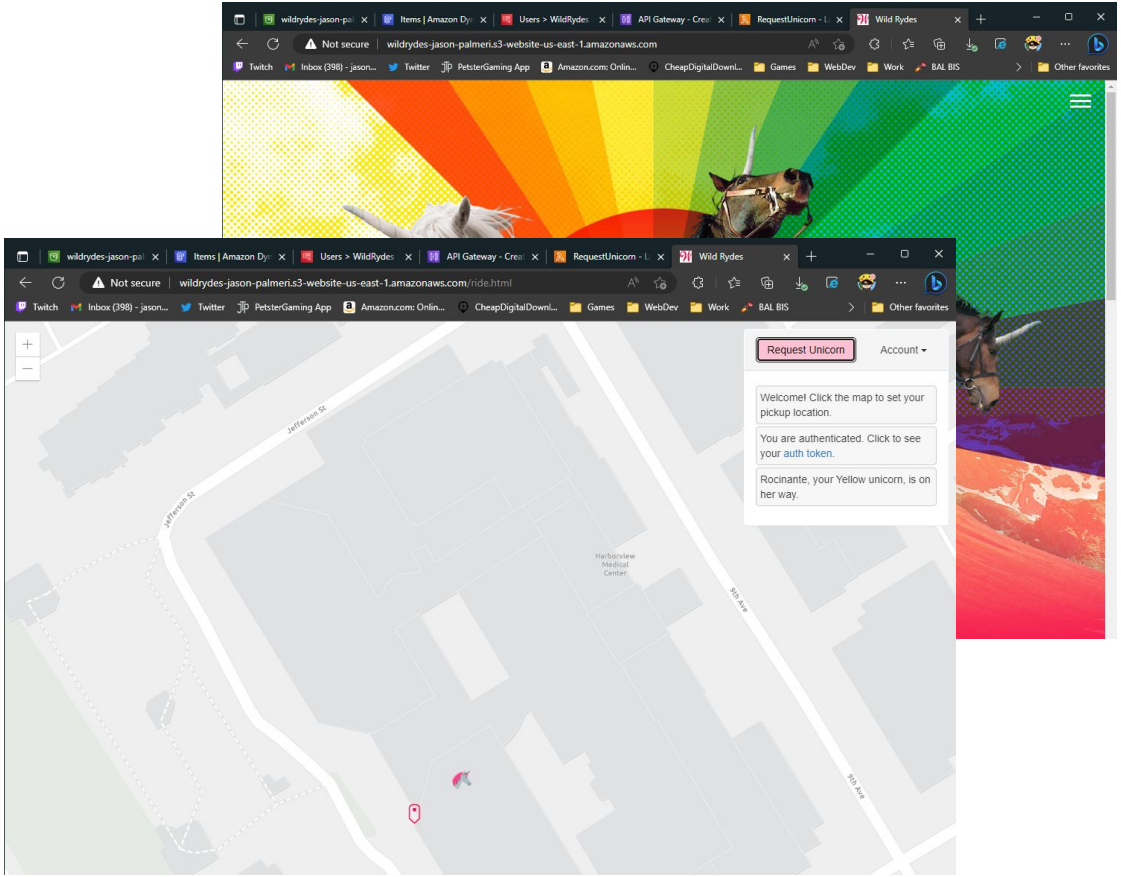
```
function findUnicorn(pickupLocation) {
    console.log('Finding unicorn for ', pickupLocation.Latitude, ', ', pickupLocation.Longitude);
    return fleet[Math.floor(Math.random() * fleet.length)];
}
```

3. Finally we send the data back to the client and update the page using javascript



The Result

With these AWS services working together we can deliver a polished web application with user authentication, API gateways and database storage, all with pay-as-you-go pricing that will allow us to expand as needed without extra hassle on the development team.



WildRydes UI

Meet the Unicorns

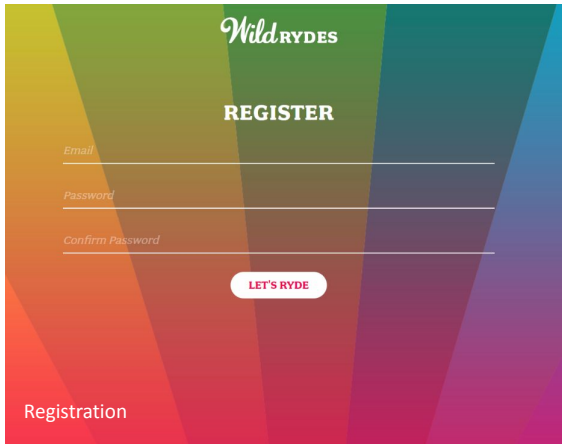
Q: Why should I use this app?
A: Unicorns are faster, safer, and more reliable. In recent times, their numbers have grown significantly, reaching a scale that makes it finally possible to harness them for mass transportation at an affordable cost.

Q: How do you recruit the unicorns? How can I know that my unicorn is trustworthy?
A: Our unicorns are recruited from only the most humane and highest standard unicorn farms. Our unicorns are grass-fed, free range creatures raised on vegan, non-GMO diets. These unicorns are also completely safe because unicorns have infallible morality and judgment.

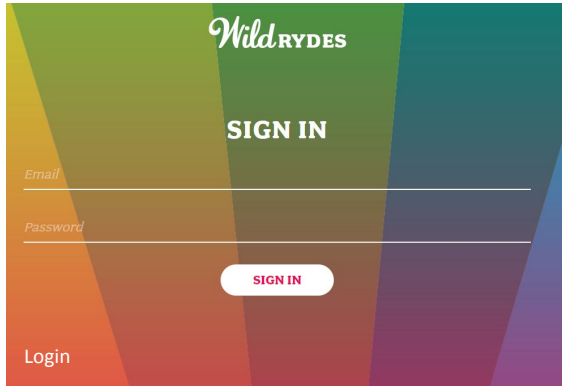
Q: How do I request a unicorn?
A: Simply download our app, then tap a button to begin. Your unicorn will arrive shortly.

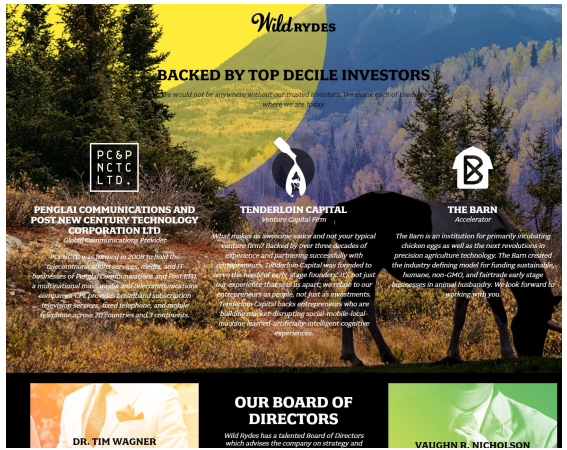
Q: How much does it cost?
A: Since Wild Rydes is a marketplace for flight-based transportation, the price you pay is based on factors such as distance and availability of unicorns. You set the maximum price you're willing to pay for a given ryde and then Wild Rydes matches you with a unicorn that's willing to accept your price.

Q: How does it work?
A: Our product is powered by a complex algorithm which efficiently matches idle unicorns with ryders based on factors such as proximity and shortest time-to-destination. The system is built on a serverless architecture, which



Registration





Investors

Architecture Design Choices

Most of the choices made for WildRydes is based around the reliability and scalability of the platform. We chose AWS S3 to host our website and it's files because it is easily accessible using the cloud, and can also be scaled up or down based on the needs WildRydes has and will have.

All systems were built on the us-east-1 region. This choice was made for the development phase of the project, as our developers worked remotely from the East coast. When we go live we can switch to any region available to AWS that will serve our needs, which is a part of why we chose AWS as the infrastructure.

We made the choice to use the combination of DynamoDB and Cognito because of it's easy setup, and scalability. Cognito made it simple to setup a user management system. Similarly DynamoDB allowed us to create database tables easily to view our ride information.

Finally we used Lambda and the API Gateway to create functions and routes to these functions. The API Gateway makes it easy to create, secure, and monitor API Gateways, as having these gateways unsecure could be troublesome. Our Lambda service was built on a smaller scale to start, as we see ourselves not needing much space or memory currently. As we expand we can easily expand the memory of our Lambda with AWS's pay as you need platform.

All choices were made with the scale of the current company, and the future company in mind, as well a pay-as-you-go mindset to help get the startup going.

Quality Assurance Test Cases

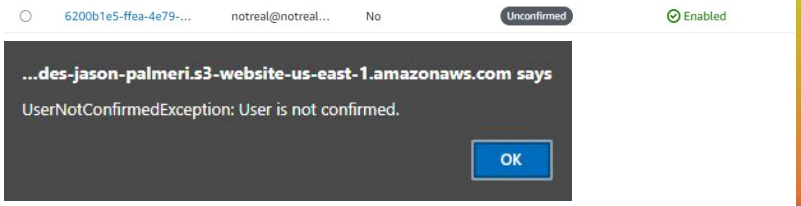
The team ran through several Test Cases to ensure that the product functions properly on all ends.

A few tests that were ran:

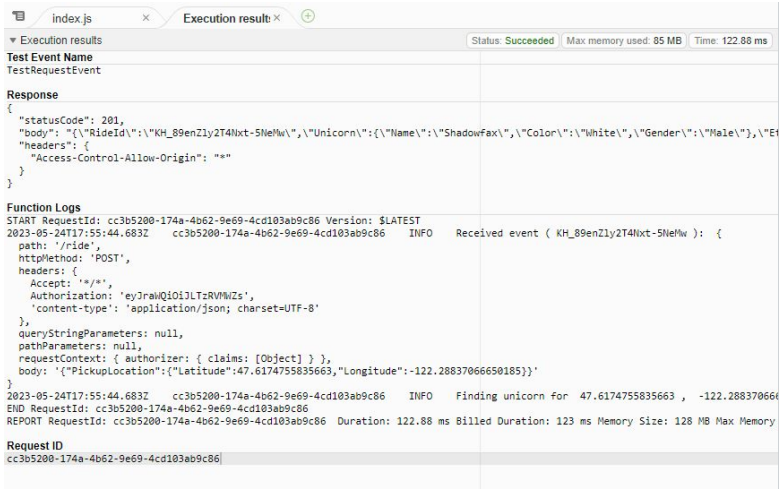
1. Validated/Unvalidated user access
2. RequestUnicorn functionality test
3. Registration, Login, Logout functionality test
4. Web app access tests, page loading tests
5. Verify users is authenticated on the front-end

Quality Assurance Test Cases Visualized

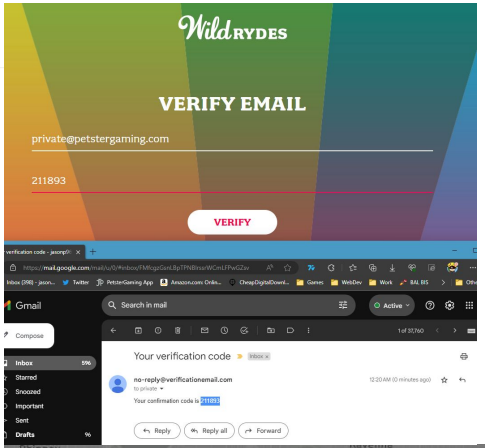
1



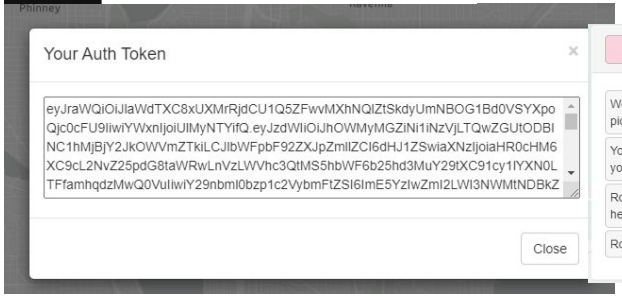
2



3



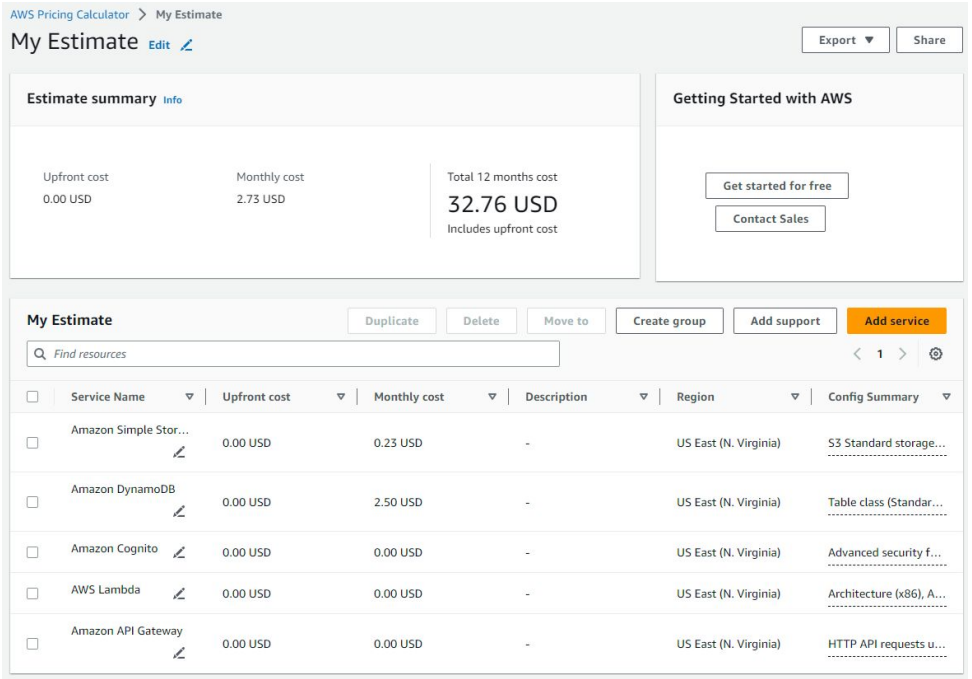
4



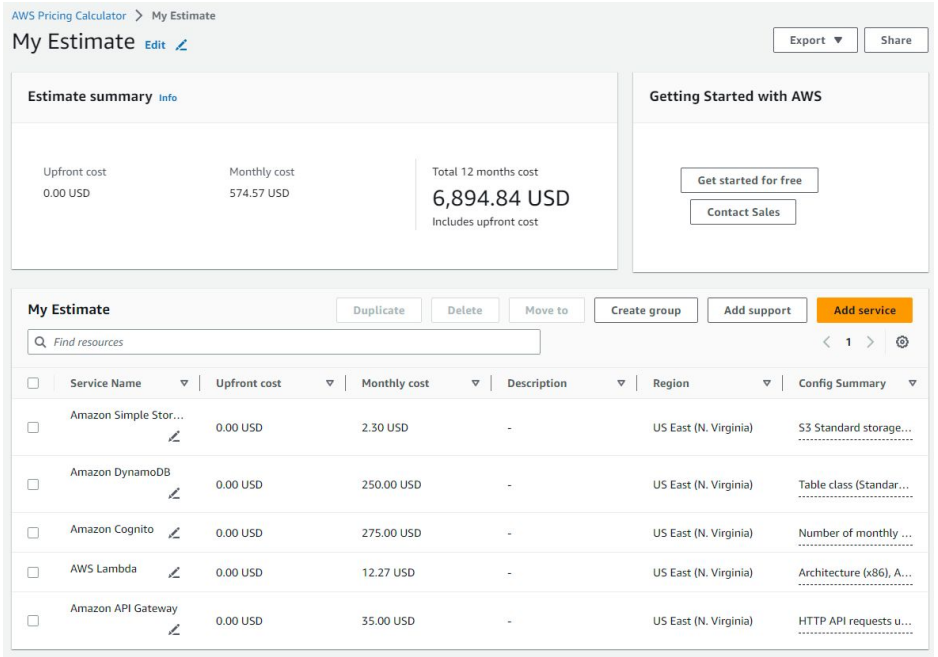
5

Cost Analysis and Breakdown - Startup

For WildRydes startup costs we are looking at around $2.73 per month, with 100 monthly users requesting 10 rides each. As you can see the cost is minimal for the year at $32.76. The biggest cost is in the DynamoDB where we are storing our customer transaction data. The next biggest cost is the S3 service, hosting our web application and it's files.



Cost Analysis and Breakdown - Success



As WildRydes expands and sees over 1 million monthly active users, requesting at least 10 rides per month we can see over a 20000% increase in yearly costs, at $6894.84. With a larger user set and more rides being requested we have to increase the size of both our DynamoDB and Cognito systems, which both consume around $250 per month. Some services stay at a lower price such as the S3, which doesn't need to be expanded as much as the other services, and the Lambda service, which is only handling a small function.