

WILD RYDES

(THE SERVERLESS WEB APPLICATION)

A

PROJECT REPORT

**In partial fulfillment for the award of
Industrial Training**

Of

BACHELOR OF COMPUTER APPLICATIONS

**Under the guidance of MR. SHOUVIK SARKAR Project
carried out at**



ARDENT COMPUTECH PVT LTD (AN ISO 9001:2015 CERTIFIED)

SDF Bldg, Module 132, GP Block, Sector V, Bidhanagar, Kolkata- 700091

• **Submitted by**

- 1. AKSHAY MISHRA**
- 2. RAJAT SINGH**
- 3. PRIYAM BHAUMIK**
- 4. TANISHQ TARIQUE**
- 5. ROSHAN JAISWAL**

THE HERITAGE ACADEMY CHOWBAGA RD, MUNDAPARA,

WESTBENGAL, INDIA

AUGUST-SEPTEMBER-2021

In association with



1. Title of the Project: **WILD RYDES (THE SERVERLESS WEB APPLICATION)**
2. Project Members: **1. AKSHAY MISHRA; 2. RAJAT SINGH 3. PRIYAM BHAUMIK 4. TANISHQ TARIQUE 5. ROSHAN JAISWAL**
3. Name and Address of the Guide: **Mr. Shouvik Sarkar, Technical Head (Core Domain)**, Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified), Module 132, SDF Bldg, Sector V, Bidhanagar, Kolkata-91.
4. Working / Training experience of the Guide: **8 Years**
5. Project Version Control History:-

Version	Primary Authors	Description of Version	Date Completed
Final	AKSHAY MISHRA RAJAT SINGH PRIYAM BHAUMIK TANISHQ TARIQUE ROSHAN JAISWAL	Project Report	10 th September, 2021

6. Signatures of Team Members:-

- Akshay Mishra
- Rajat Singh
- Priyam Bhaumik
- Tanishq Tarique
- Roshan Jaiswal

Date :- 10th September, 2021

7. Signature of Approval of Project Proposal Evaluator :-

MR. SHOUVIK SARKAR

Date :-

PROJECT RESPONSIBILITY FORM

“WILD RYDES” (THE SERVERLESS WEB APPLICATION)

SL NO.	NAME OF MEMBER	RESPONSIBILITY GIVEN
1.	AKSHAY MISHRA	AWS amplify, AWS lambda, project-report, presentation
2.	RAJAT SINGH	AWS cognito, front-end coding, project-report
3.	PRIYAM BHAUMIK	Backend coding, code commit, presentation, video-editing
4.	TANISHQ TARIQUE	AWS API gateways, testing, project-report
5.	ROSHAN JAISWAL	Testing, project-report, AWS dynamoDB

Each group member must participate in project development and developing the ideas for the required elements. Individual group members will be responsible for completing tasks that help to finalize the project and the performance. All group members must be assigned a task.

• Name of the Students :-

1. AKSHAY MISHRA
2. RAJAT SINGH
3. PRIYAM BHAUMIK
4. TANISHQ TARIQUE
5. ROSHAN JAISWAL

• Signatures of the Students :-

- Akshay Mishra
- Rajat Singh
- Priyam Bhaumik
- Tanishq Tarique
- Roshan Jaiswal

Date :- 10th September, 2021

DECLARATION

We hereby declare that the project work being presented in the project proposal entitled “**WILD RYDES**” (THE SERVERLESS WEB APPLICATION) is in partial fulfillment of the requirements for the award of industrial training in cloud computing with Amazon Web Services at ARDENT COMPUTECH PVT. LTD, BIDHANAGAR, KOLKATA, WEST BENGAL, is an authentic work carried out under the guidance of **MR. SHOUVIK SARKAR**. The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

Date : 10th September, 2021

Name of the Students:-

1. AKSHAY MISHRA
2. RAJAT SINGH
3. PRIYAM BHAUMIK
4. TANISHQ TARIQUE
5. ROSHAN JAISWAL



ARDENT COMPUTECH PVT LTD (AN ISO 9001:2015 CERTIFIED)

SDF Bldg, Module 132, GP Block, Sector V, Bidhanagar, Kolkata- 700091

CERTIFICATE

This is to certify that this proposal of the project entitled “WILD RYDES” (THE SERVERLESS WEB APPLICATION) is a record of bonafide work, carried out by 1. AKSHAY MISHRA 2. RAJAT SINGH 3. PRIYAM BHAUMIK 4. TANISHQ TARIQUE 5. ROSHAN JAISWAL under my guidance at Ardent Computech Pvt. Ltd. In my opinion, the report in its present form is in partial fulfillment of the requirements for the award of industrial training in cloud computing with Amazon Web Services and as per the regulations of the Ardent Computech Pvt. Ltd. To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report.

Guide / Supervisor

MR. SHOUVIK SARKAR

Technical Head (Core Domain)



ARDENT COMPUTECH PVT LTD (AN ISO 9001:2015 CERTIFIED)

SDF Bldg, Module 132, GP Block, Sector V, Bidhanagar, Kolkata- 700091

ACKNOWLEDGEMENT

The success of any project depends largely on the encouragement and guidelines of many others. We take this sincere opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project work.

Our heartfelt thanks to Mr. Sanjoy Banerjee, for providing us the opportunity to develop the project at Ardent Computech Pvt. Ltd.

We would like to show our greatest appreciation to Mr. Shouvik Sarkar, Project Manager at Ardent, Kolkata. We always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance, this project would not have materialized.

We wish to express our deep sense of gratitude to our Head of the Department of our college, THE HERITAGE ACADEMY, for his able guidance and useful suggestions, which helped us in completing the project work in time.

Words are inadequate in offering our thanks to the other trainees, project assistants and other members at Ardent Computech Pvt. Ltd. for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.



ARDENT COMPUTECH PVT LTD (AN ISO 9001:2015 CERTIFIED)

SDF Bldg, Module 132, GP Block, Sector V, Bidhanagar, Kolkata- 700091

TABLE OF CONTENTS

SL NO.	NAME OF THE MAIN TOPIC	NAME OF THE SUB-TOPICS	PAGE NOS.
1.	Introduction	1.1. What Is serverless web application	8
		1.2. Objectives of the project	10
		1.3. Scope of the project	11
2.	System Design	2.1. Data flow diagram (DFD)	12
		2.2. AWS Amplify	13
		2.3. Amazon Cognito	14
		2.4. Amazon DynamoDB	15
		2.5. AWS Lambda	16
		2.6. Amazon API Gateway	17
		2.7. AWS Lamba Architectural Diagram	18
		2.8. AWS Amplify Architectural Diagram	19
		2.9 AWS Cognito Architectural Diagram	20
		2.10. Functional Requirements	21
3.	Basic Coding	3.1. codes for front end and backend	22 - 31
4.	Implementation and Testing	4.1. Introduction	32 - 33
		4.2. Objectives of Testing	34
		4.3. Test Cases	35 - 37
5.	Snapshots	5.1. Images of pages taken from the website	38 - 50
6.	Conclusion	6.1. Inferences drawn from the project	51
7.	Bibliography / References	7.1. Some sites used to make the project	52
8.	Thank You		53

1.1 What Is serverless web application?

The term “serverless” is weird because you know you need a server to host anything online. What’s really happening is kind of a distributed server thing. So you’ll have the code for your web app (HTML, CSS, JS) hosted somewhere and then all of your back-end stuff will be taken care of elsewhere. You’ll make API calls to other services and use tools like Azure Functions or AWS Lambda to handle the data.

You can think of a serverless app as a front-end that calls 3rd party APIs when an event happens. That means nothing is sitting there waiting for an event to happen or a call to come through. Unlike with normal web apps, you don’t have to handle the back-end at all. Everything happens dynamically which frees up a lot of resources for your application.

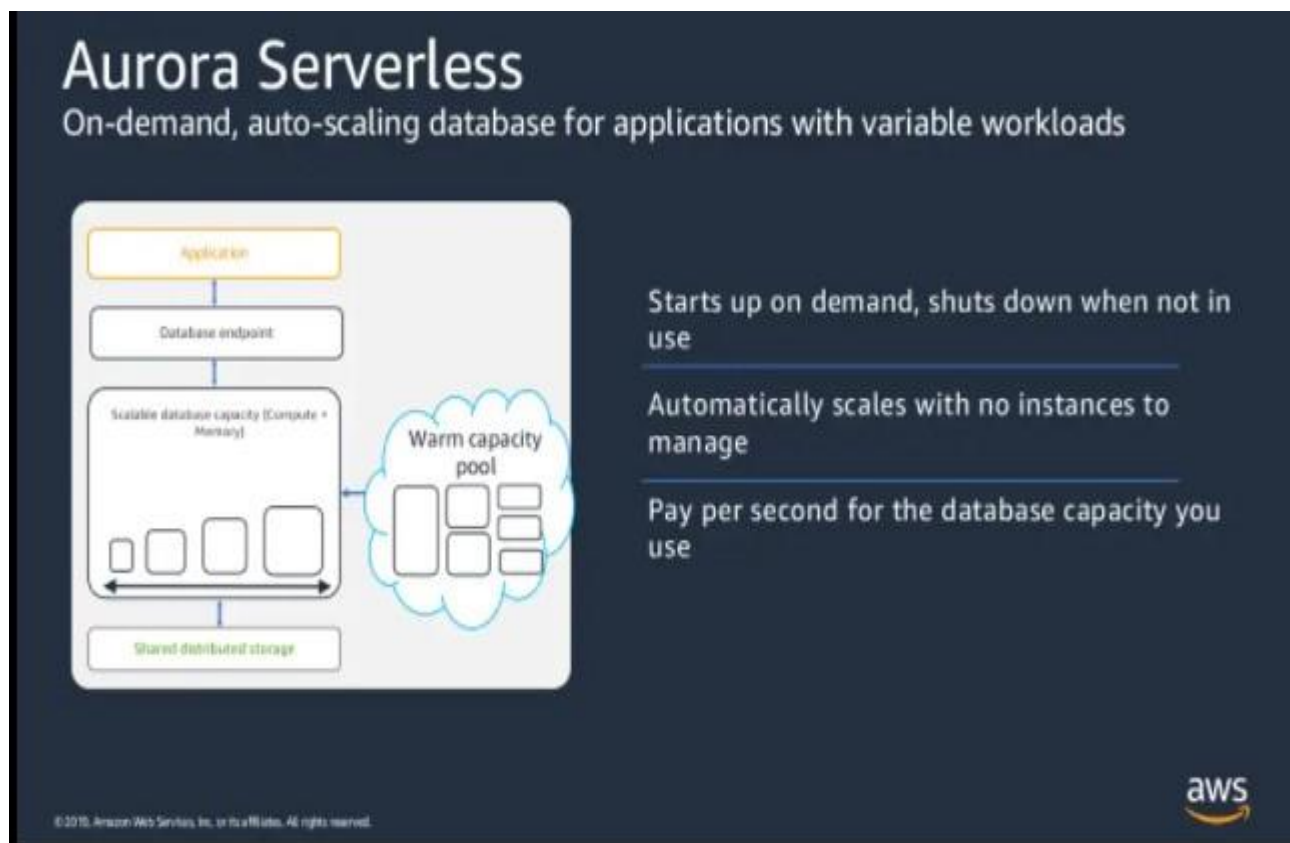
Advantages of serverless web applications :

- You don’t have to worry about your entire back-end crashing because of one bug.
- No more big deploys because your entire back-end is distributed.
- You don’t have to know a specific back-end language because you don’t have to write any code.
- Incredibly fast to set up.
- Scalable from the start because you don’t have to worry about how many requests are hitting your server.

Why Serverless Apps?

Everyone is trying to make their web applications faster, more scalable, and easier to maintain. There's a huge push moving us more and more to the cloud for our hosting meaning there are fewer shops that have the cold server closest onsite. With everything moving towards the cloud, we aren't always sure what OS our "server" is using. So we might as well take it a step further and get rid of the server altogether.

That's what serverless apps are trying to do. The concept has been around for a few years, but it's starting to pick up speed. Companies are starting to experiment with serverless applications because of all the benefits they provide, but just as many companies are hesitating because of the potential issues.



1.1. OBJECTIVES OF THE PROJECT

In this project we have created a simple serverless web application that enables users to request 'unicorn' rides from the 'Wild Rydes' fleet. The application will present a simple user interface . The application will also provide facilities for users to register with the service and log in before requesting rides. The 'Unicorn' rides mentioned frequently is nothing but a bike ride that transports a customer from one place to another within a small area. This is equivalent to Rapido rides which are very famous in India. Because this is a project, the radius of operation is being kept limited to a small city, this same concept and idea can be used to deploy this system on a large scale.

The webapp is a serverless web application which implies that the infrastructure required to run the website or operations page hosted on the internet is not maintained by the owners of the web app. For this we have the different services from AWS like AWS amplify , AWS Cognito, AWS lambda, AWS dynamodb, and AWS API gateways. These different services help us to maintain different aspects of our web app without having an in-house infrastructure for it.

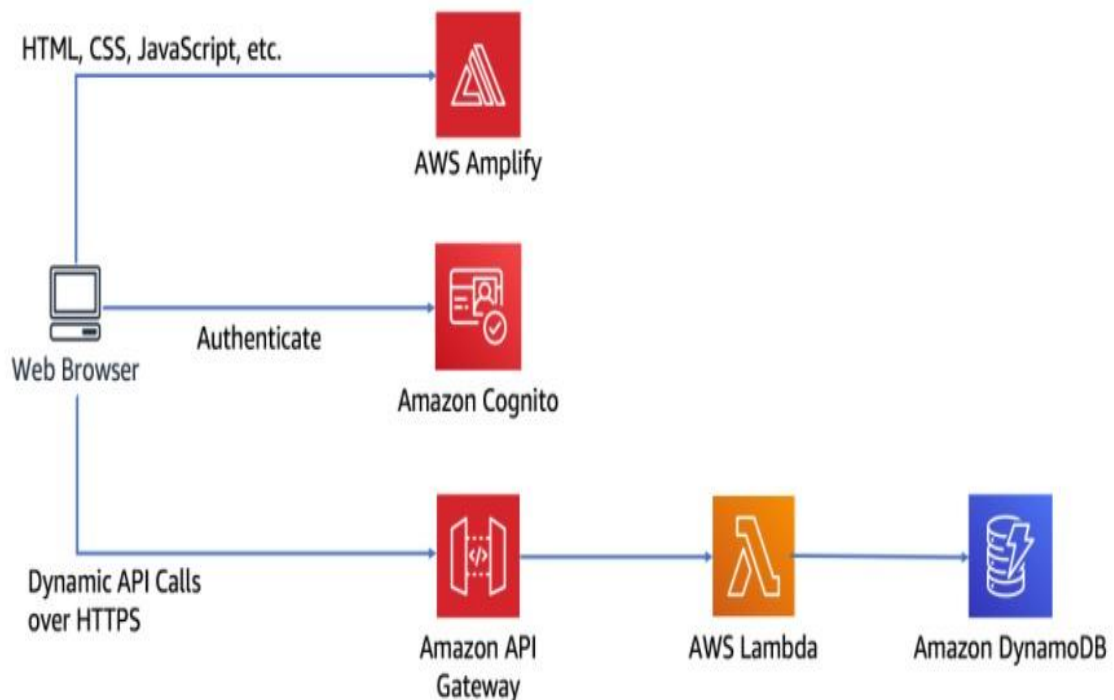
1.2. SCOPE OF THIS PROJECT

The web app created in this project uses a lot of web technologies like html, css, javascript for the frontend, node.js for the backend(Rest API), and a no SQL database for storage of user information.

All these aspects are taken care off through the services provided by AWS, in other words it is fully dependent on AWS. As of now(project-phase) on the website has been thought off and implemented. The scope of this project only deals with the digital part of the service provided(unicorn), the actual operation of the unicorn is out of the scope of this project.

2.1. DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a diagram that describes the flow of the data and the processes that changes data throughout a system. A structured analysis and design tool that can be used for flowcharting in place of or in association with information. Oriented and process oriented system flowcharts. When analysts prepare the Data Flow Diagram , they specify the user's needs at a level of detail that virtually determines the information flow into and out of the system and the required data resources. This network is constructed by using a set of symbols that do not imply physical implementation.



2.2. AWS AMPLIFY

Amazon Web Services are some of the most useful products we have access to. One such service that is becoming increasingly popular as days go by is **AWS Amplify**. It was released in 2018 and it runs on Amazon's cloud infrastructure. It is in direct competition with **Firebase**, but there are features that set them apart.

Why is it needed?

User experience on any application is the most important aspect that needs to be taken care of. **AWS Amplify** helps unify the user experience across platforms such as web and mobile. This makes it easier for a user to choose which one would they be more comfortable with.

It is useful in the case of front-end development as it helps in building and deployment. Many who use it claim that it actually makes full-stack development a lot easier with its scalability.

Main features:

- Can be used for authenticating users which are powered by Amazon Cognito.
- With help from Amazon AppSync and Amazon S3, it can securely store and sync data seamlessly between applications.
- As it is serverless, making changes to any back-end-related cases has become simpler. Hence, less time is spent on maintaining and configuring back-end features.
- It also allows for offline synchronization.
- It promotes faster app development.
- It is useful for implementing Machine Learning and AI-related requirements as it is powered by Amazon Machine learning services.
- It is useful for continuous deployment.

Various AWS services are used for various functionalities. **AWS Amplify** offers. The main components are **libraries**, **UI components**, and the **CLI toolchain**. It also provides **static web hosting** using **AWS Amplify Console**.

Let's have a brief look at some of these components:

Libraries:

The libraries are powered by AWS services. The help in the production of applications immensely, especially in the case of the back-end.

2.3. AWS COGNITO

A **User pool** in **AWS Cognito** is a user directory, which helps users to sign in to your web or mobile app through AWS Cognito. Users can also sign in through other social platforms like Google, Facebook, Amazon, or Apple.

It doesn't matter users can directly sign in or use a third-party authentication, all these users in the User pool have a profile directory that you can access through Software Development Kit (**SDK**).

Services provided by User pool :

User pool provides sign-in and sign-up services

- It also gives you a built-in web UI so that users can sign-in. This web UI is customizable too.
- It manages the user directory and user profile.
- Facilitates you to sign-in with Facebook, Google, Apple, log in with Amazon, and also sign-in with SAML identities providers from your user pool. SAML (Security Assertion Markup Language 2.0) is a service that helps identity providers (IdP) to authenticate the user and pass identity and security information to the service provider(SP).
- It provides Multi-Factor Authentication (MFA) for more secure sign-in by phone and email verification.
- It also gives you a workflow that can be customized and user migration through AWS Lambda triggers.

2.4. AWS DYNAMODB

DynamoDB allows users to create databases capable of storing and retrieving any amount of data and comes in handy while serving any amount of traffic. It dynamically manages each customer's requests and provides high performance by automatically distributing data and traffic over servers.

It is a fully managed NoSQL database service that is fast, predictable in terms of performance, and seamlessly scalable. It relieves the user from the administrative burdens of operating and scaling a distributed database as the user doesn't have to worry about hardware provisioning, patching Software, or cluster scaling. It also eliminates the operational burden and complexity involved in protecting sensitive data by providing encryption at REST.

Advantage of DynamoDB:

The main advantages of opting for Dynamodb are listed below:

- It has fast and predictable performance.
- It is highly scalable.
- It offloads the administrative burden operation and scaling.
- It offers encryption at REST for data protection.
- Its scalability is highly flexible.
- AWS Management Console can be used to monitor resource utilization and performance metrics.
- It provides on-demand backups.
- It enables point-in-time recovery for your Amazon DynamoDB tables. Point-in-time recovery helps protect your tables from accidental write or delete operations. With point-in-time recovery, you can restore that table to any point in time during the last 35 days.
- It can be highly automated.

2.5. AWS LAMBDA

1. What is AWS Lambda?

AWS Lambda is an Amazon serverless computing system that runs code and automatically manages the underlying computing resources. It lets a person automatically run code in response to many types of events, such as HTTP requests from Amazon API gateway, table updates in Amazon DynamoDB, and state transitions. It also enables the person to extend to other AWS services with custom logic, and even create its own back-end services.

The service works by running code on high-availability computer infrastructure.

It then performs all the administrative duties of that compute resource, like:

- providing maintenance on server and operating system,
- automatically scaling and managing the person's capacity provisions,
- handling security patch deployment
- code monitoring
- logging

The only work required on the user's end is supplying code for it to run.

2. How Does AWS Lambda Work?

While AWS Lambda may seem confusing at first, it really isn't. In fact, it's actually just a simple process:

- Start off by uploading the code to AWS Lambda.
- From there, set up the code to trigger from other AWS services, HTTP endpoints, or mobile apps. AWS Lambda will only run the code when it's triggered and will also only use the computing resources needed to run it. The user have to pay only for the compute time used.

3. What are the Advantages of AWS Lambda?

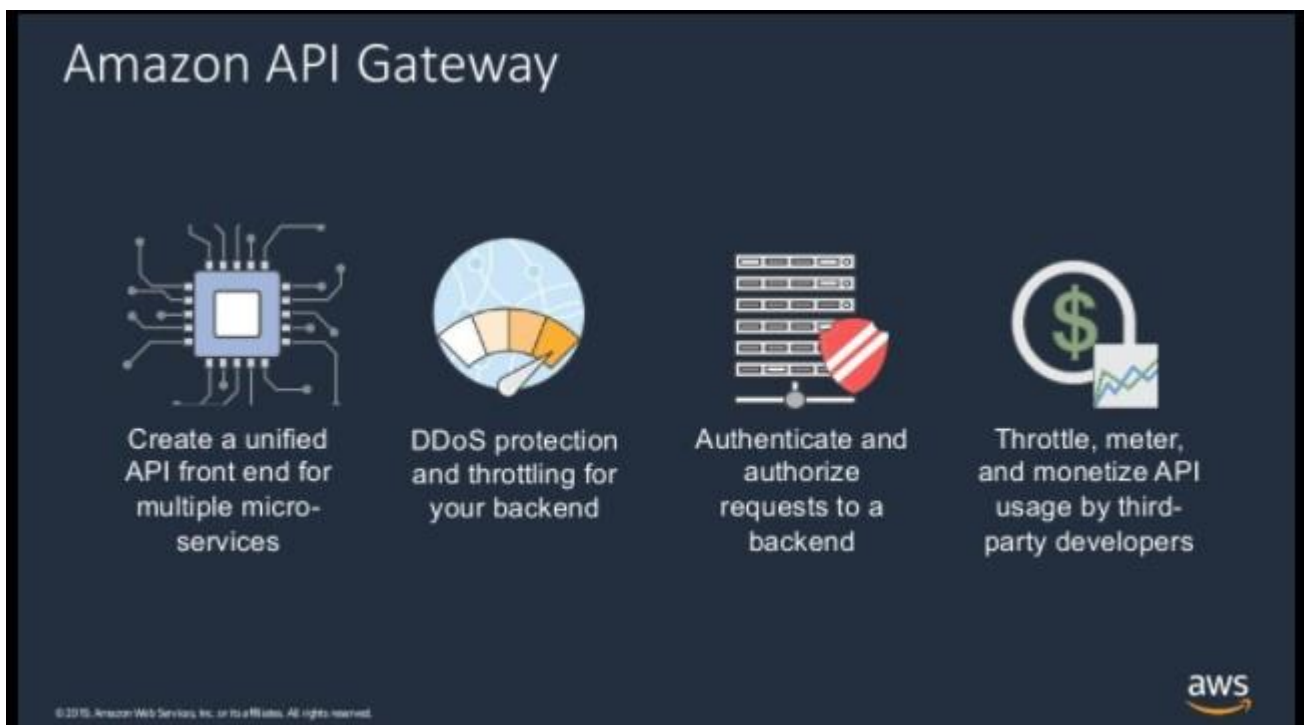
AWS Lambda offers a variety of benefits to its users. Here are 3 of the biggest benefits one should know about when deciding if AWS Lambda is right choice:

- **It doesn't require the user to manage any servers.** Since AWS Lambda automatically runs user's code, there's no need for the user to manage the server. Simply write the code and upload it to Lambda.
- **It empowers the user to easily scale.** AWS Lambda runs code in response to each trigger, so user's application is automatically scaled. The code also runs in parallel processes, each triggered individually, so scaling is done precisely with the size of the workload.

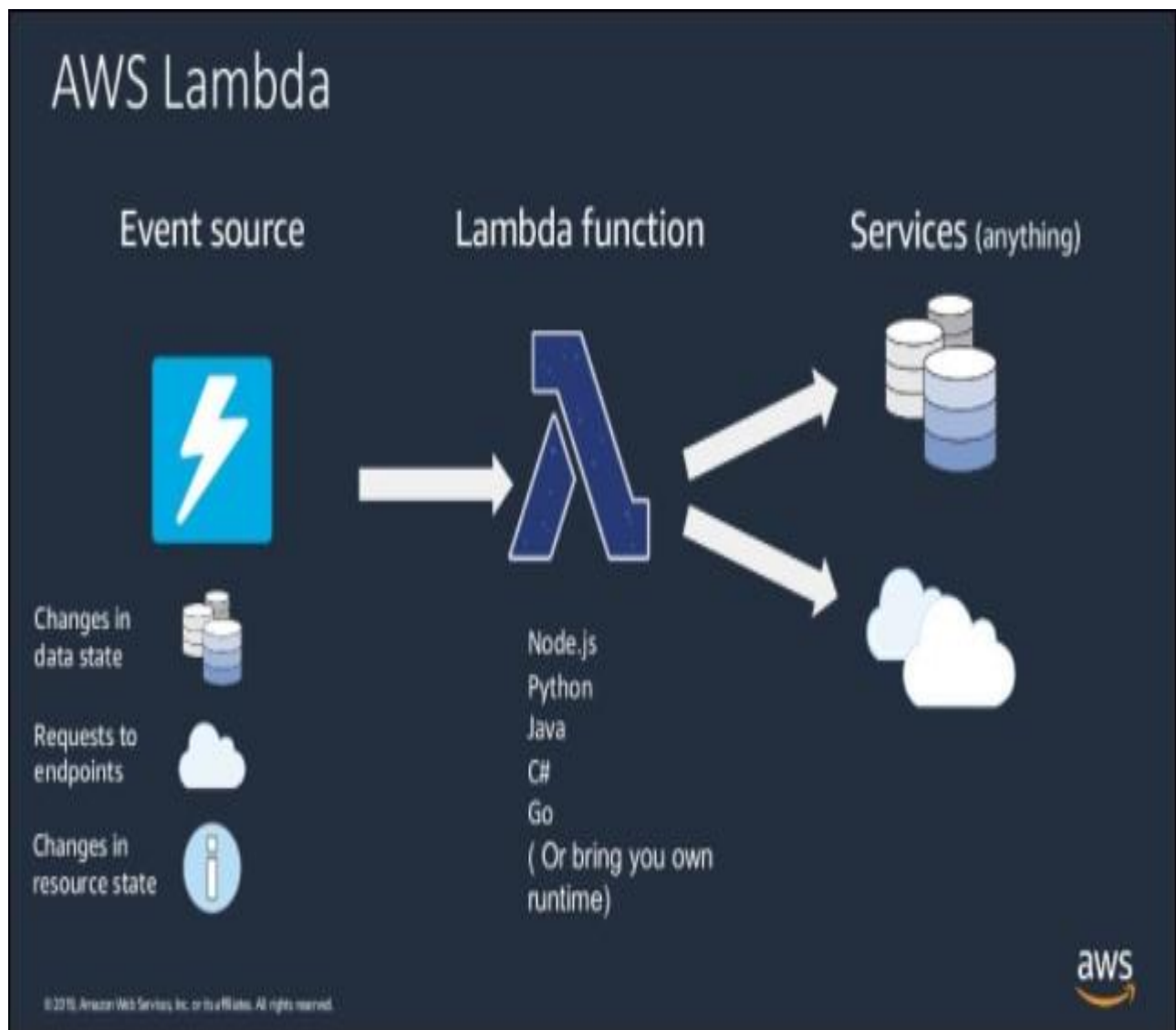
2.6 AWS API GATEWAYS

API – It stands for “Application Programming Interface”. In general APIs are built-in libraries or collections of libraries that perform some specific task or function. In general, API to get connected with other web applications. We generally connect with other web applications by following certain protocols, methods. We can also use the data through the API that is used by another web application or software through the use of API of that software. In other words, we can exchange our data through API with other web applications/software.

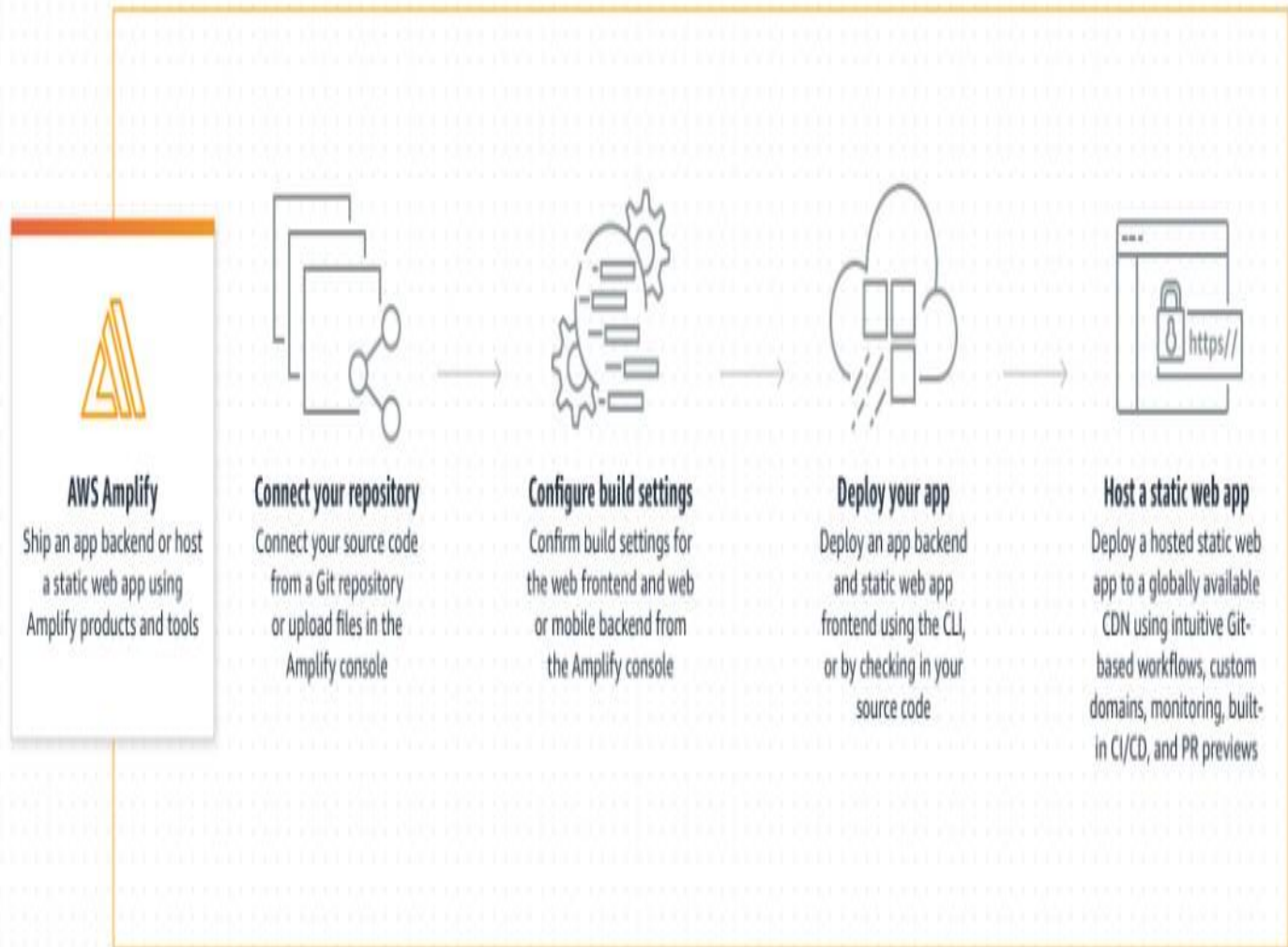
What is API Gateway Pattern: The API Gateway Pattern in some cases stands for “Backend for frontend”. It basically the entry gate for taking entry in any application by an external source. The pattern which is going on in a programmer’s mind while they are making the client’s application. It acts as a medium between the client applications and microservices. For example-Netflix is the most famous example of an API gateway.



2.7 AWS LAMBDA ARCHITECTURAL DIAGRAM



2.8 AWS AMPLIFY CONSOLE:



2.9 AWS COGNITO ARCHITECTURAL DIAGRAM



2.10 FUNCTIONAL REQUIREMENTS OF THIS WEBSITE

MODULES

The Modules used in this website are as follows :-

1. **Home**:- This page is the first page we see when we open the website and it has all the details of the website.
2. **Registration**:- in this page the signup process takes place as the user is required to provide email id , give a password and confirm the password for registration.
3. **Verification**:- this page verifies the user by sending a verification code on the email provided in registration page. After this user needs to provide the email and the verification code to move to the next process.
4. **Sign in**:- This module is for registered users to Login. The user needs to provide the email and password for logging in to there accounts.
5. **Set pickup and requesting unicorn**:- In this page the user needs to click on the map for there location and click on request unicorn for booking the unicorn.
6. **Information**:- This page includes all the information about the website in detail.
7. **Meet the unicorns**:- This page contain detailed information the unicorns like there names experience etc.

3.1. CODES FOR THE WEBSITE

CONFIG.JS

```
window._config = {  
  cognito: {  
    userPoolId: 'us-west-2_uXboG5pAb',  
    userPoolClientId: '25ddkmj4v6hfsfvruhpf7n4hv',  
  },  
  api: {  
    invokeUrl: 'https://rc7nyt4tql.execute-api.us-west-2.amazonaws.com/prod',  
  }  
};
```

COGNITO -AUTH.JS

```
/*global WildRydes _config AmazonCognitoIdentity AWSCognito*/  
  
var WildRydes = window.WildRydes || {};  
  
(function scopeWrapper($) {  
  var signInUrl = '/signin.html';  
  
  var poolData = {  
    UserPoolId: _config.cognito.userPoolId,  
    ClientId: _config.cognito.userPoolClientId  
  };  
};
```

```

var userPool;
    if (!(_config.cognito.userPoolId &&
        _config.cognito.userPoolClientId &&
        _config.cognito.region)) {
        $('#noCognitoMessage').show();
        return;
    }

userPool = new AmazonCognitoIdentity.CognitoUserPool(poolData);

if (typeof AWSCognito !== 'undefined') {
    AWSCognito.config.region = _config.cognito.region;
}

WildRydes.signOut = function signOut() {
    userPool.getCurrentUser().signOut();
};

WildRydes.authToken = new Promise(function fetchCurrentAuthToken(resolve,
reject) {
    var cognitoUser = userPool.getCurrentUser();

    if (cognitoUser) {
        cognitoUser.getSession(function sessionCallback(err, session) {
            if (err) {
                reject(err);
            } else if (!session.isValid()) {
                resolve(null);
            } else {
                resolve(session.getIdToken().getJwtToken());
            }
        });
    } else {
        resolve(null);
    }
});

```

```

/*
 * Cognito User Pool functions
 */
function register(email, password, onSuccess, onFailure) {
    var dataEmail = {
        Name: 'email',
        Value: email
    };
    var attributeEmail = new AmazonCognitoIdentity.CognitoUserAttribute(dataEmail);

    userPool.signUp(toUsername(email), password, [attributeEmail], null,
        function signUpCallback(err, result) {
            if (!err) {
                onSuccess(result);
            } else {
                onFailure(err);
            }
        }
    );
}

function signin(email, password, onSuccess, onFailure) {
    var authenticationDetails = new AmazonCognitoIdentity.AuthenticationDetails({
        Username: toUsername(email),
        Password: password
    });

    var cognitoUser = createCognitoUser(email);
    cognitoUser.authenticateUser(authenticationDetails, {
        onSuccess: onSuccess,
        onFailure: onFailure
    });
}

```



```

function verify(email, code, onSuccess, onFailure) {
    createCognitoUser(email).confirmRegistration(code, true, function
confirmCallback(err, result) {
    if (!err) {
        onSuccess(result);
    } else {
        onFailure(err);
    }
    });
}

```

```

function createCognitoUser(email) {
    return new AmazonCognitoIdentity.CognitoUser({
        Username: toUsername(email),
        Pool: userPool
    });
}

```

```

function toUsername(email) {
    return email.replace('@', '-at-');
}

```

```

/*
 * Event Handlers
 */

```

```

$(function onDocReady() {
    $('#signinForm').submit(handleSignin);
    $('#registrationForm').submit(handleRegister);
    $('#verifyForm').submit(handleVerify);
});

```

```

function handleSignin(event) {
  var email = $('#emailInputSignin').val();
  var password = $('#passwordInputSignin').val();
  event.preventDefault();
  signin(email, password,
    function signinSuccess() {
      console.log('Successfully Logged In');
      window.location.href = 'ride.html';
    },
    function signinError(err) {
      alert(err);
    }
  );
}

```

```

function handleRegister(event) {
  var email = $('#emailInputRegister').val();
  var password = $('#passwordInputRegister').val();
  var password2 = $('#password2InputRegister').val();

  var onSuccess = function registerSuccess(result) {
    var cognitoUser = result.user;
    console.log('user name is ' + cognitoUser.getUsername());
    var confirmation = ('Registration successful. Please check your email inbox or
spam folder for your verification code. ');
    if (confirmation) {
      window.location.href = 'verify.html';
    }
  };
  var onFailure = function registerFailure(err) {
    alert(err);
  };
  event.preventDefault();
}

```

```
    if (password === password2) {
        register(email, password, onSuccess, onFailure);
    } else {
        alert('Passwords do not match');
    }
}

function handleVerify(event) {
    var email = $('#emailInputVerify').val();
    var code = $('#codeInputVerify').val();
    event.preventDefault();
    verify(email, code,
        function verifySuccess(result) {
            console.log('call result: ' + result);
            console.log('Successfully verified');
            alert('Verification successful. You will now be redirected to the login page.');
```

window.location.href = signInUrl;

```
        },
        function verifyError(err) {
            alert(err);
        }
    );
}
})(jQuery));
```

RIDE.JS

```
/*global WildRydes _config*/
```

```
var WildRydes = window.WildRydes || {};
```

```
WildRydes.map = WildRydes.map || {};
```

```
(function rideScopeWrapper($) {
```

```
    var authToken;
```

```
    WildRydes.authToken.then(function setAuthToken(token) {
```

```
        if (token) {
```

```
            authToken = token;
```

```
        } else {
```

```
            window.location.href = '/signin.html';
```

```
        }
```

```
    }).catch(function handleTokenError(error) {
```

```
        alert(error);
```

```
        window.location.href = '/signin.html';
```

```
    });
```

```
    function requestUnicorn(pickupLocation) {
```

```
        $.ajax({
```

```
            method: 'POST',
```

```
            url: _config.api.invokeUrl + '/ride',
```

```
            headers: {
```

```
                Authorization: authToken
```

```
            },
```

```
            data: JSON.stringify({
```

```
                PickupLocation: {
```

```
                    Latitude: pickupLocation.latitude,
```

```
                    Longitude: pickupLocation.longitude
```

```
                }
```

```
            })),
```

```

contentType: 'application/json',
  success: completeRequest,
  error: function ajaxError(jqXHR, textStatus, errorThrown) {
    console.error('Error requesting ride: ', textStatus, ', Details: ', errorThrown);
    console.error('Response: ', jqXHR.responseText);
    alert('An error occured when requesting your unicorn:\n' +
jqXHR.responseText);
  }
});
}

```

```

function completeRequest(result) {
  var unicorn;
  var pronoun;
  console.log('Response received from API: ', result);
  unicorn = result.Unicorn;
  pronoun = unicorn.Gender === 'Male' ? 'his' : 'her';
  displayUpdate(unicorn.Name + ', your ' + unicorn.Color + ' unicorn, is on ' +
pronoun + ' way. ');
  animateArrival(function animateCallback() {
    displayUpdate(unicorn.Name + ' has arrived. Giddy up!');
    WildRydes.map.unsetLocation();
    $('#request').prop('disabled', 'disabled');
    $('#request').text('Set Pickup');
  });
}

```

```

// Register click handler for #request button
$(function onDocReady() {
  $('#request').click(handleRequestClick);
  $(WildRydes.map).on('pickupChange', handlePickupChanged);

```

```

WildRydes.authToken.then(function updateAuthMessage(token) {
  if (token) {
    displayUpdate('You are authenticated. Click to see your <a
href="#authTokenModal" data-toggle="modal">auth token</a>.'.);
    $('#authToken').text(token);
  }
});

if (!_config.api.invokeUrl) {
  $('#noApiMessage').show();
}
});

function handlePickupChanged() {
  var requestButton = $('#request');
  requestButton.text('Request Unicorn');
  requestButton.prop('disabled', false);
}

function handleRequestClick(event) {
  var pickupLocation = WildRydes.map.selectedPoint;
  event.preventDefault();
  requestUnicorn(pickupLocation);
}

function animateArrival(callback) {
  var dest = WildRydes.map.selectedPoint;
  var origin = {};

  if (dest.latitude > WildRydes.map.center.latitude) {
    origin.latitude = WildRydes.map.extent.minLat;
  } else {
    origin.latitude = WildRydes.map.extent.maxLat;
  }
}

```

```
if (dest.longitude > WildRydes.map.center.longitude) {
    origin.longitude = WildRydes.map.extent.minLng;
} else {
    origin.longitude = WildRydes.map.extent.maxLng;
}

WildRydes.map.animate(origin, dest, callback);
}

function displayUpdate(text) {
    $('#updates').append($('- ' + text + '</li>'));
}
}(jQuery));

```

4. IMPLEMENTATION AND TESTING

A Software System Test Plan is a document that describes the objectives , scope , approach and focus of software testing effort. The process of preparing a test plan is a usual way to think the efforts needed to validate the acceptability of a software product. The complete document will help people outside the test group , understand the “WHY” and “HOW” product validation. It should be thorough enough to be useful but not so thorough that no one outside the test group will read it.

4.1. INTRODUCTION

Testing is the process of running a system with the intention of finding errors. Testing enhances the integrity of a system by detecting deviations in design and errors in the system.

Testing aims at detecting error-prone areas. This helps in the prevention of the errors in a system. Testing also adds value to the product by conforming to the user's requirements.

The main purpose of Testing is to detect errors and error-prone areas in a system. Testing must be thorough and well planned. A partially tested system is as bad as an untested system. And the price of an untested and under-tested system is high.

The implementation is the final and important phase. It involves user-training, system testing in order to ensure successful running of the proposed system. The user tests the system and changes are made according to their needs. The Testing involves the Testing of the developed system using various kinds of data. While testing, errors are noted and correctness is the mode.

➤ STEPS FOR SYSTEM TESTING :-

1. Test Environment Setup :- Create testing environment for the better quality of testing.
2. Create Test Case :- Generate test case for the testing process.
3. Create Test Data
4. Execute Test Case
5. Defect Reporting
6. Regression Testing
7. Log Defects
8. Retest

4.2. OBJECTIVES OF TESTING

The objective of our test plan is to find and report as many bugs as possible to improve the integrity of our program. Although exhaustive testing is not possible, we will exercise a broad range of tests to achieve our goal. Our user interface to utilize these functions, is designed to be user-friendly and provide an easy manipulation of the tree. The application will only be used as a demonstration tool, but we would like to ensure that it could be run from a variety of platforms with little impact on performance and usability.

PROCESS OVERVIEW :-

The following represents the overall flow of the Testing Process :-

1. Identify the requirements to be tested. All Test Cases shall be derived using the current Program Specifications.
2. Identify which particular Test(s) will be used to test each module.
3. Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.
4. Identify the expected results for each test.
5. Document the test case configuration, test data, and expected results.

6. Perform the test(s).
7. Document the test data , test cases , and test configuration used during the testing process. The information shall be submitted via the Unit/System Test Report (STR).
8. Successful unit testing is required before the unit is eligible for component integration/system testing.
9. Unsuccessful testing requires a Bug Report Form to be generated. This document shall describe the test case , the problem encountered , its possible cause , and the sequence of events that led to the problem. It shall be used as a basis for the later technical analysis.
10. Test Documents and reports shall be submitted. Any specification to be reviewed , revised , or updated shall be handled immediately.

4.3. TEST CASES

A Test case is a document that describes an input , action , or event and expected response , to determine if a feature of an application is working correctly. A Test case should contain particular such as test case identifier , test condition , test condition , input data.

Requirement expected results. The process of developing test cases can help find problems in the requirement or design of an application , since it requires completely thinking through the operation of the application.

6. TESTING STEPS :-

1. UNIT TESTING :-

Unit Testing focuses efforts on the smallest unit of software design. This is known as Module Testing. The modules are tested separately. The Test is carried out during programming stage itself. In this step , each module is found to be working satisfactorily as regards to the expected output from the module.

2. INTEGRATION TESTING :-

Data can be lost across an interface. One module can have an adverse effect on another , sub functions , when combined , may not be linked in desired manner in major functions. Integration Testing is a systematic approach for constructing the program structure , while at the same time conducting test to uncover errors associated within the interface. The objective is to take unit tested modules and build program structure. All the modules are combined and tested as a whole.

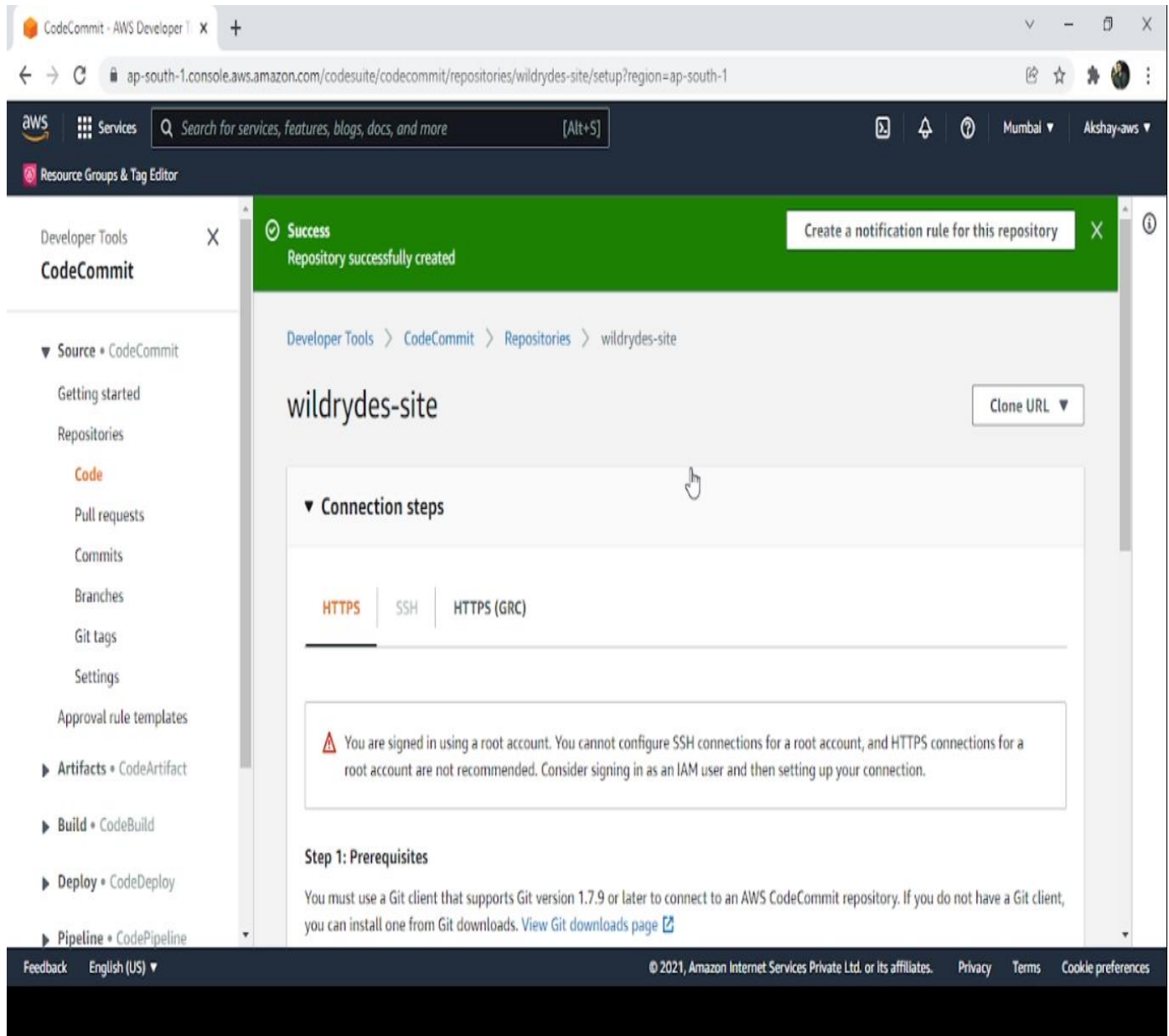
3. VALIDATION :-

At the culmination of the integration testing , software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test begins in Validation Testing. Validation Testing can be defined in many ways. But a simple definition is that the validation succeeds when the software functions in a manner that is expected by the customer. After validation test has been conducted , one of the three possible conditions exists.

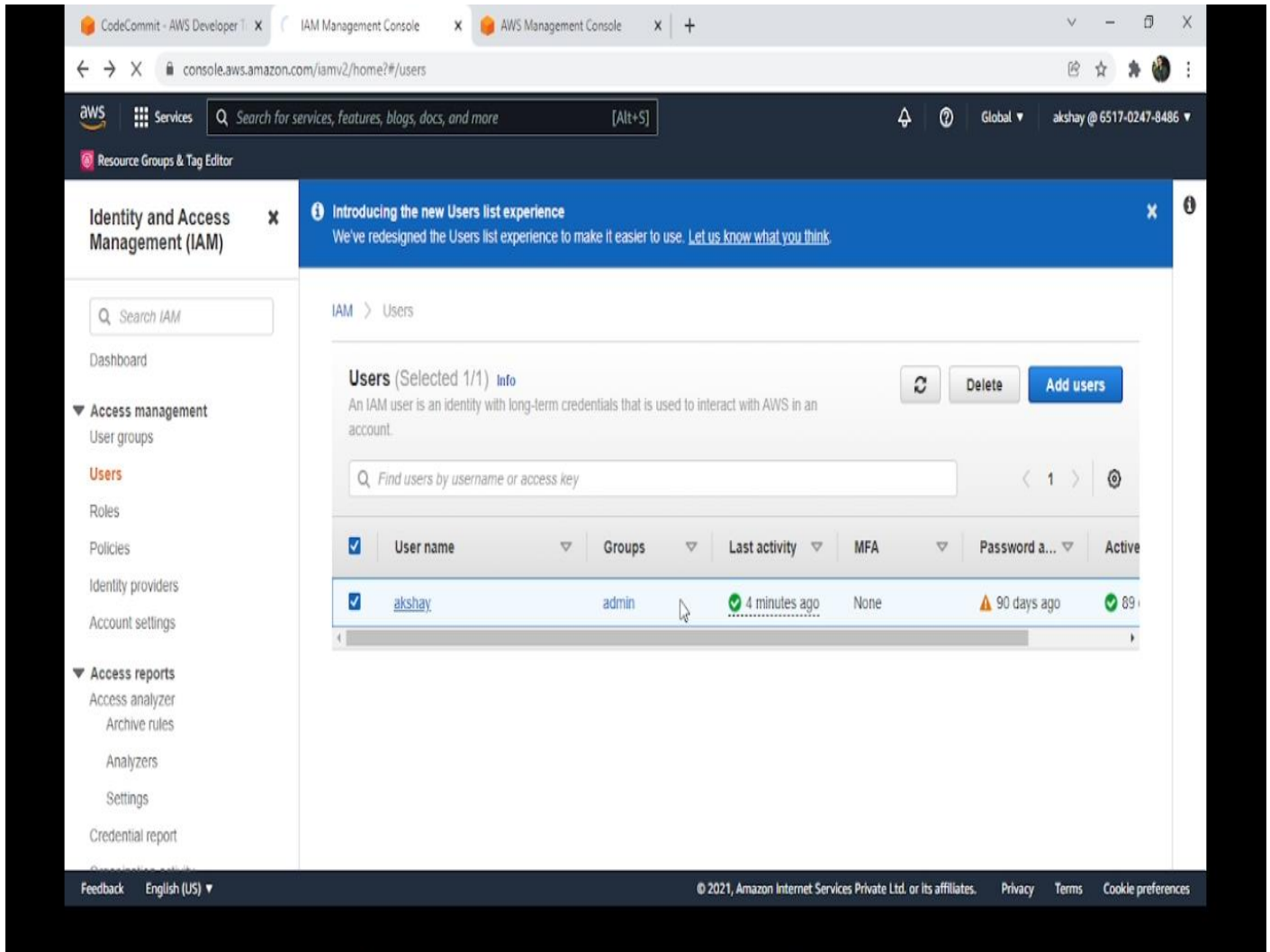
- a) The function or performance characteristics confirms to specification and are accepted.
- b) A deviation from specification is uncovered and a deficiency list is created.
- c) Proposed system under consideration has been tested by using validation test and found to be working satisfactorily.

5.1. SNAPSHOTS

1. AWS CODE COMMIT :-



2. GIVING ROLES TO IAM USER :-



3. DEPLOYING WEBSITE AWS AMPLIFY :-

The screenshot displays the AWS Amplify console interface. The left sidebar shows the 'App settings' menu with options like General, Admin UI management, Domain management, Build settings, Previews, Notifications, Environment variables, Access control, Monitoring, and Rewrites and redirects. The main content area is titled 'Frontend environments' and shows the 'master' branch with 'Continuous deploys set up (Edit)'. A diagram illustrates the deployment pipeline: Provision (green checkmark) - Build (green checkmark) - Deploy (green checkmark) - Verify (green checkmark). Below this, a table provides details for the last deployment:

Last deployment	Last commit	Previews
12/2/2021, 8:19:55 PM	This is an autogenerated message Auto-build AWS CodeCommit - master	Disabled

The bottom of the console shows a footer with 'Feedback', 'English (US)', '© 2021, Amazon Internet Services Private Ltd. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'. A notification bar at the bottom right says 'Show all'.

4. MANAGING USERS WITH AWS COGNITO:

The screenshot shows the AWS Cognito User Pools console. The browser tabs include 'AWS Amplify Console', 'Wild Rydes-Rydes of future', 'Wild Rydes', 'User Pools - Amazon Cognito', and 'AWS Management Console'. The URL is 'ap-south-1.console.aws.amazon.com/cognito/users/?region=ap-south-1#/pool/ap-south-1_IPxbgAJJR/details?n=1&k=8oslfy'. The page title is 'User Pools | Federated Identities WildRydes'. A notification bar at the top states: 'The new design for Cognito User Pools console is now available. [Try out the new interface](#)'. The left sidebar shows the 'General settings' menu with options: Users and groups, Attributes, Policies, MFA and verifications, Advanced security, Message customizations, Tags, Devices, App clients, Triggers, Analytics, App integration, App client settings, Domain name, and UI customization. The main content area displays the following settings:

- Pool Id:** ap-south-1_IPxbgAJJR
- Pool ARN:** arn:aws:cognito-idp:ap-south-1:651702478486:userpool/ap-south-1_IPxbgAJJR
- Estimated number of users:** 0
- Required attributes:** email
- Alias attributes:** none
- Username attributes:** none
- Enable case insensitivity?** Yes
- Custom attributes:** [Choose custom attributes...](#)

The footer includes 'Feedback', 'English (US)', '© 2021, Amazon Internet Services Private Ltd. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

The screenshot shows the 'Advanced security' page of the AWS Cognito User Pools console for the same user pool. The left sidebar shows the 'Advanced security' menu with options: Triggers, Analytics, App integration, App client settings, Domain name, UI customization, Resource servers, Federation, Identity providers, and Attribute mapping. The main content area displays the following settings:

- Enable case insensitivity?** Yes
- Custom attributes:** [Choose custom attributes...](#)
- Minimum password length:** 8
- Password policy:** uppercase letters, lowercase letters, special characters, numbers
- User sign ups allowed?** Users can sign themselves up
- FROM email address:** Default
- Email Delivery through Amazon SES:** No
- Note:** You have chosen to have Cognito send emails on your behalf. Best practices suggest that customers send emails through Amazon SES for production User Pools due to a daily email limit. [Learn more about email best practices.](#)
- MFA:** [Enable MFA...](#)
- Verifications:** Email
- Advanced security:** [Enable advanced security...](#)

The footer is identical to the previous screenshot.

5. AWS DYNAMODB :-

The screenshot displays the AWS Management Console interface for the Amazon DynamoDB service. The browser tabs at the top include 'CodeCommit - AWS Develop', 'Wild Rydes', 'View table | Amazon Dynam...', and 'AWS Management Console'. The address bar shows the URL: 'ap-south-1.console.aws.amazon.com/dynamodbv2/home?region=ap-south-1#table?initialTagKey=&name=Rides&tab=overview'.

The console header features the AWS logo, a search bar, and the user's profile 'akshay @ 6517-0247-8486'. A blue notification banner at the top states: 'The new DynamoDB console is now complete, and becomes your default experience. Following the preview phase in which we analyzed and incorporated your feedback, we have completed the new DynamoDB console, making it even easier for you to manage your data and resources. Let us know what you think. You can still choose to return to the previous console from the navigation pane.'

The left-hand navigation pane lists various services: Dashboard, Tables (highlighted), Items, PartiQL editor, Backups, Exports to S3, Reserved capacity, DAX, Clusters, Subnet groups, Parameter groups, and Events. Below these are links for 'Tell us what you think' and 'Return to the previous console experience'.

The main content area is titled 'Rides' and shows a list of tables with one table, 'Rides', selected. The table details are displayed in the 'Overview' tab, including:

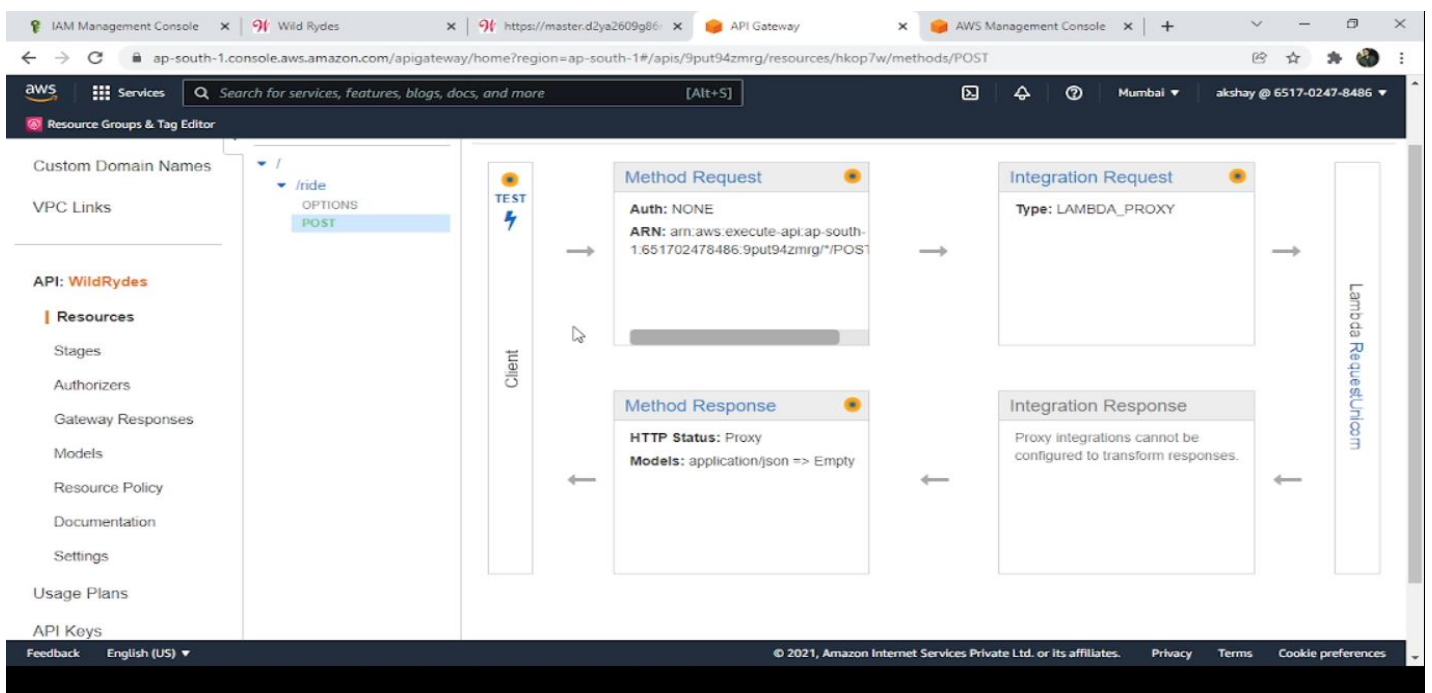
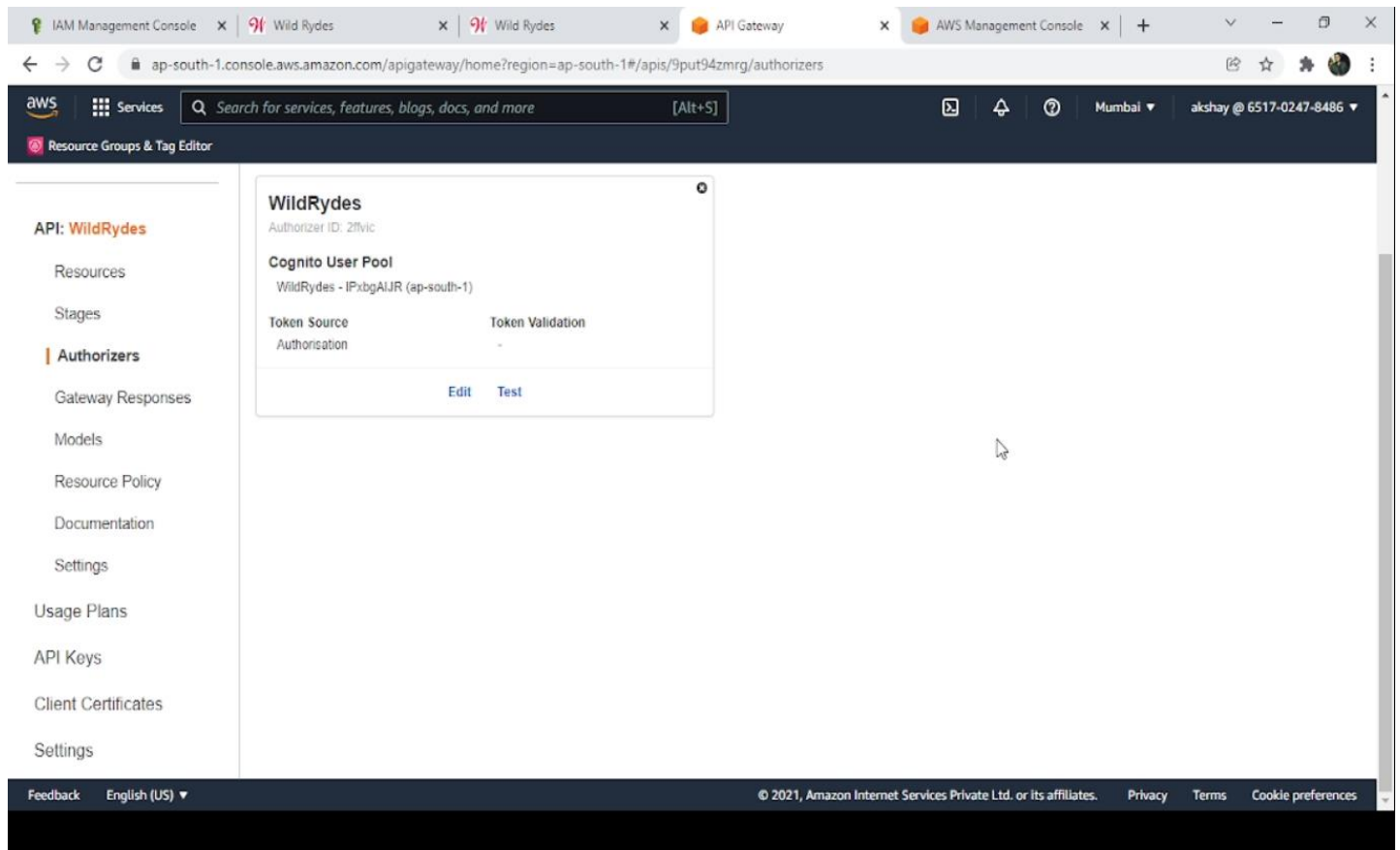
- Partition key: Rideld (String)
- Sort key: -
- Capacity mode: Provisioned
- Table status: Active (indicated by a green checkmark)
- No active alarms (indicated by a green checkmark)

The bottom of the console shows the footer with 'Feedback', 'English (US)', and copyright information: '© 2021, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences'.

6. AWS LAMBDA :-

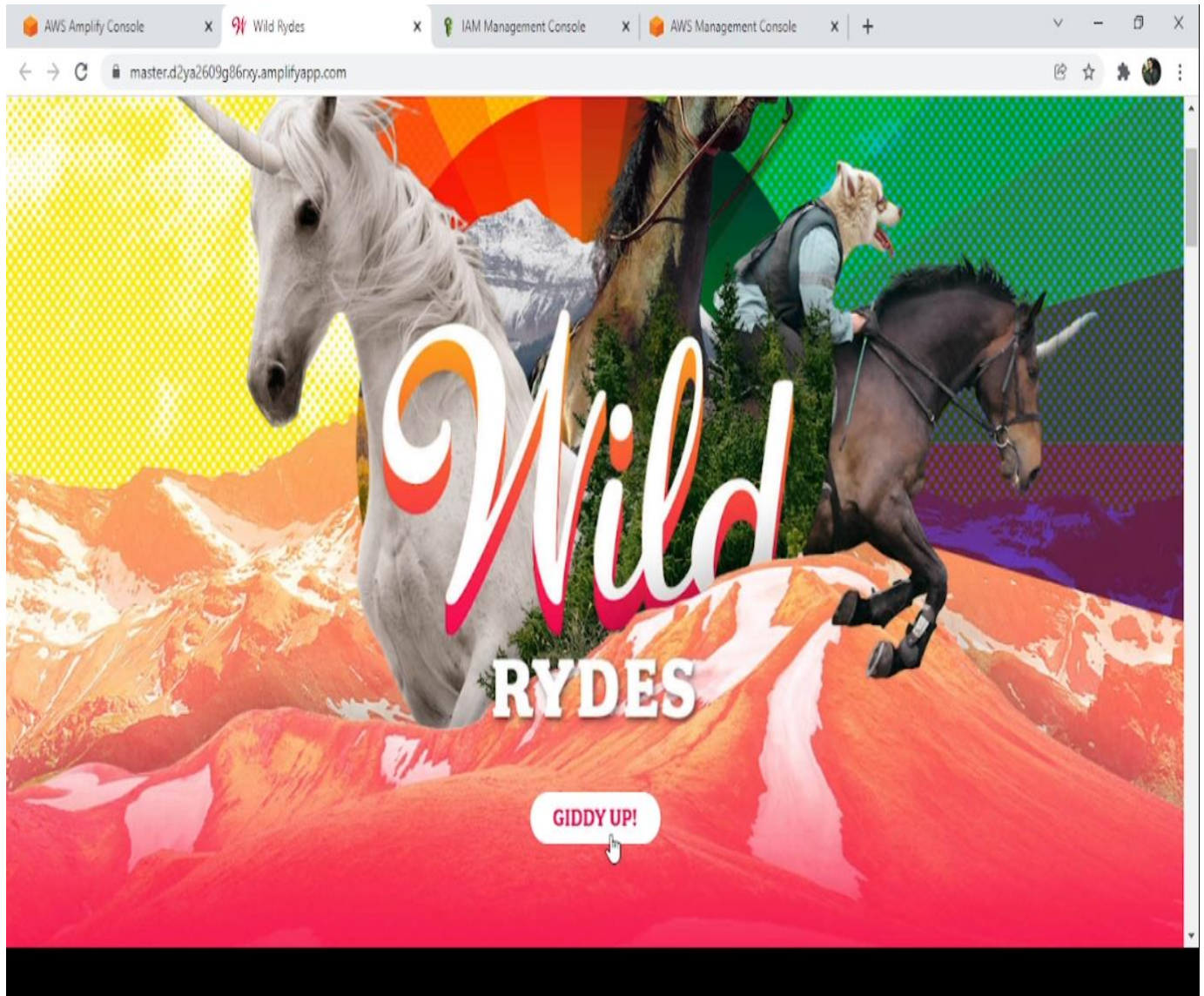
The screenshot displays the AWS Lambda console interface. At the top, a green notification bar states: "Successfully created the function RequestUnicorn. You can now change its code and configuration. To invoke your function with a test event, choose 'Test'." The breadcrumb navigation shows "Lambda > Functions > RequestUnicorn". The function name "RequestUnicorn" is prominently displayed, with buttons for "Throttle", "Copy ARN", and "Actions". Below this, the "Function overview" section includes a card for "RequestUnicorn" with a "Layers (0)" section and buttons for "+ Add trigger" and "+ Add destination". To the right, a details panel shows "Description" as "-", "Last modified" as "6 seconds ago", and "Function ARN" as "arn:aws:lambda:ap-south-1:651702478486:function:RequestUnicorn". At the bottom, a tabbed interface includes "Code", "Test", "Monitor", "Configuration", "Aliases", and "Versions". The footer contains "Feedback", "English (US)", "© 2021, Amazon Internet Services Private Ltd. or its affiliates.", "Privacy", "Terms", and "Cookie preferences".

7. AWS API GATEWAY:-

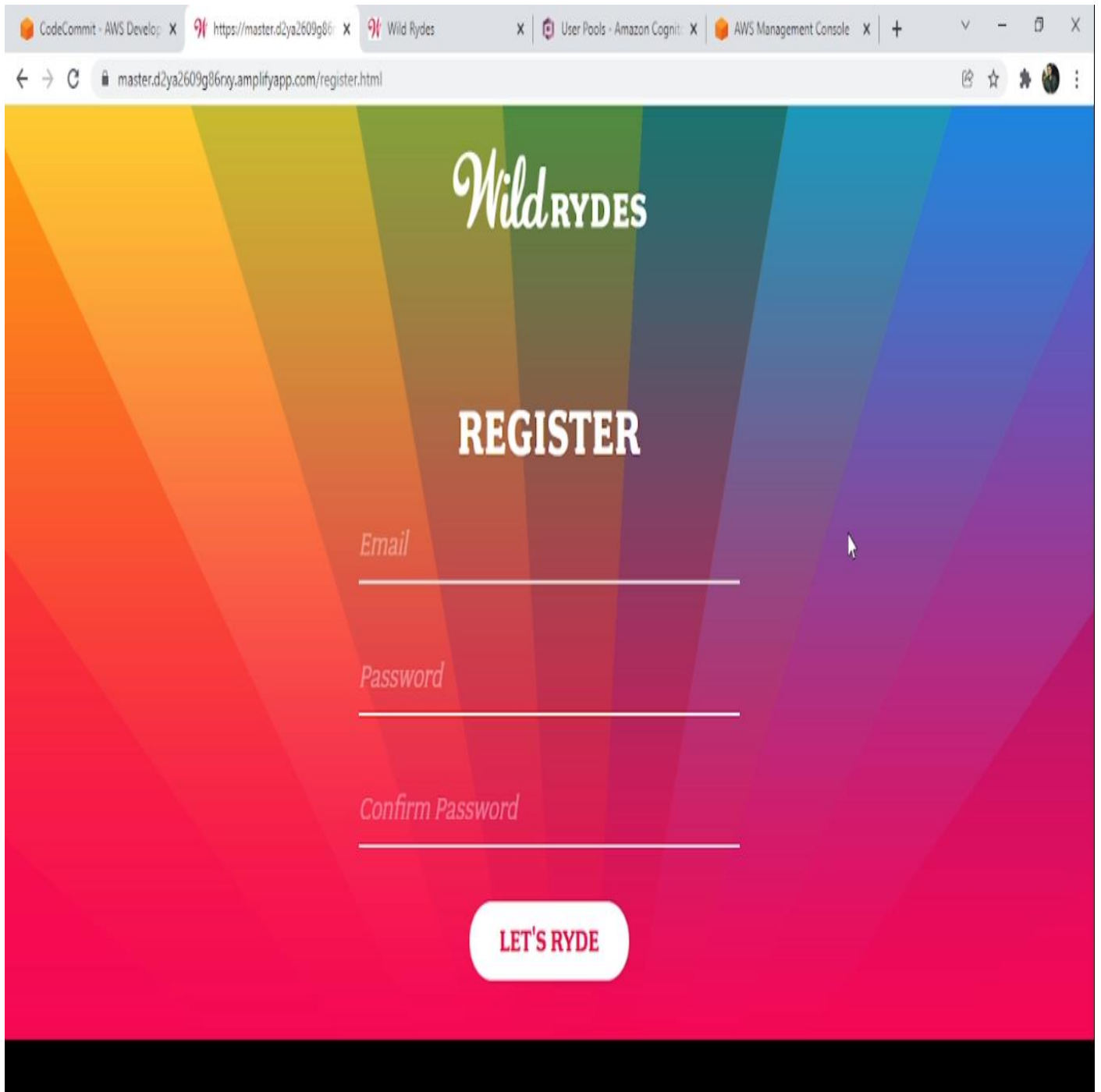


PAGES FROM THE WEBSITE

1. HOME PAGE 1 :-

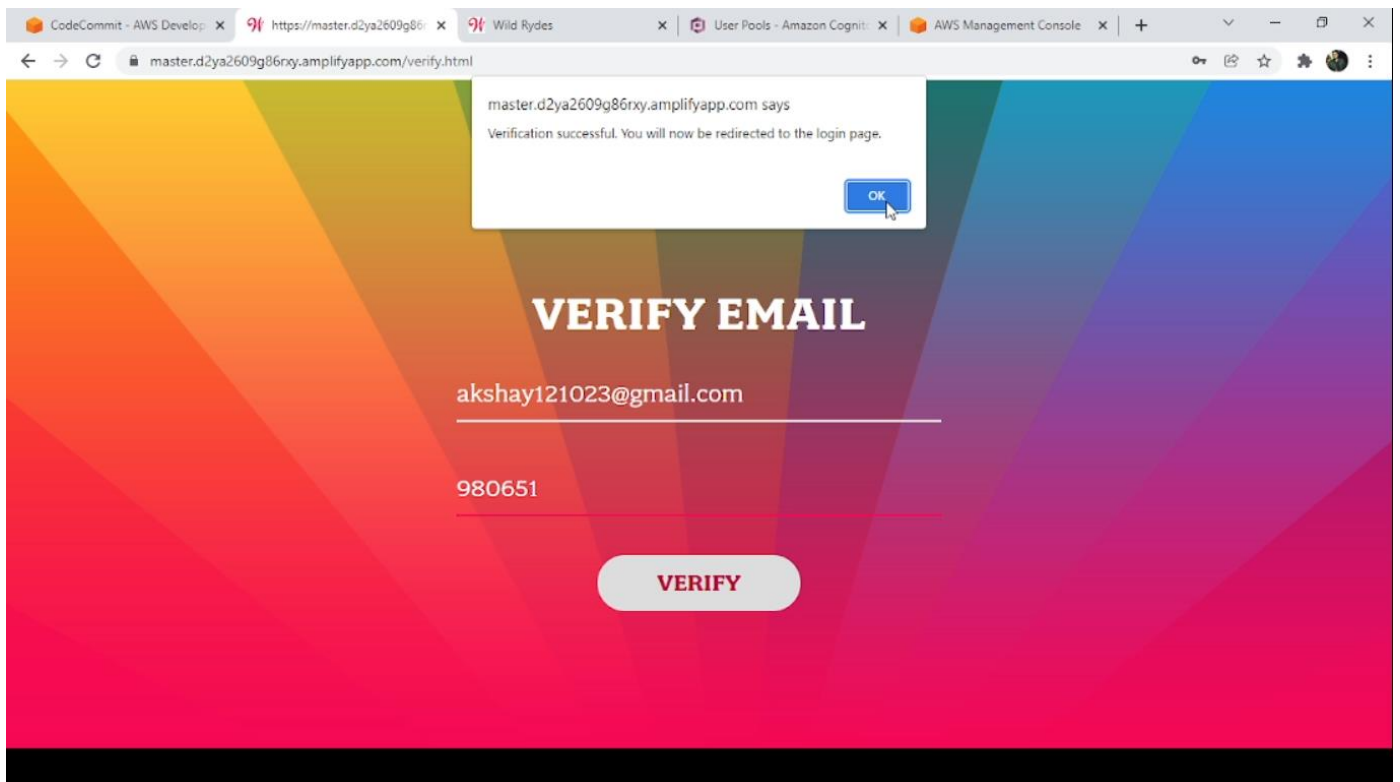
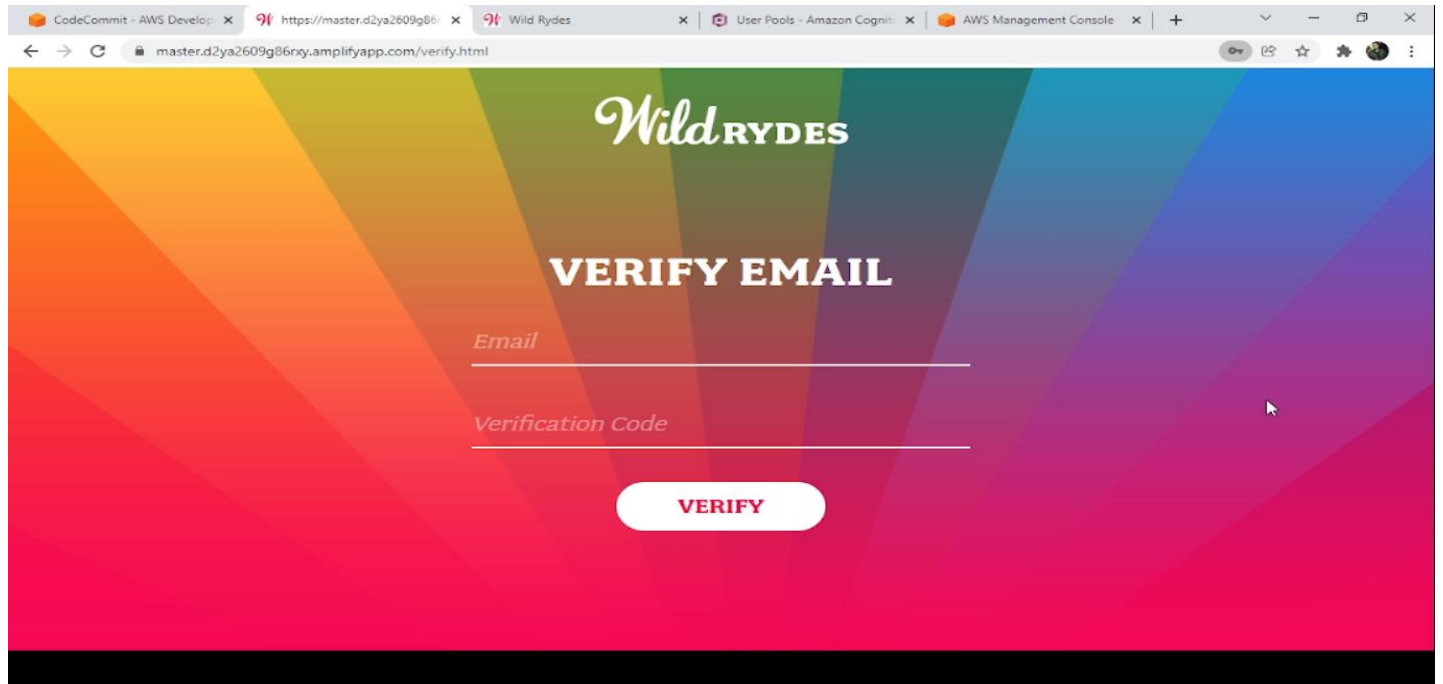


2. REGISTRATION PAGE:-

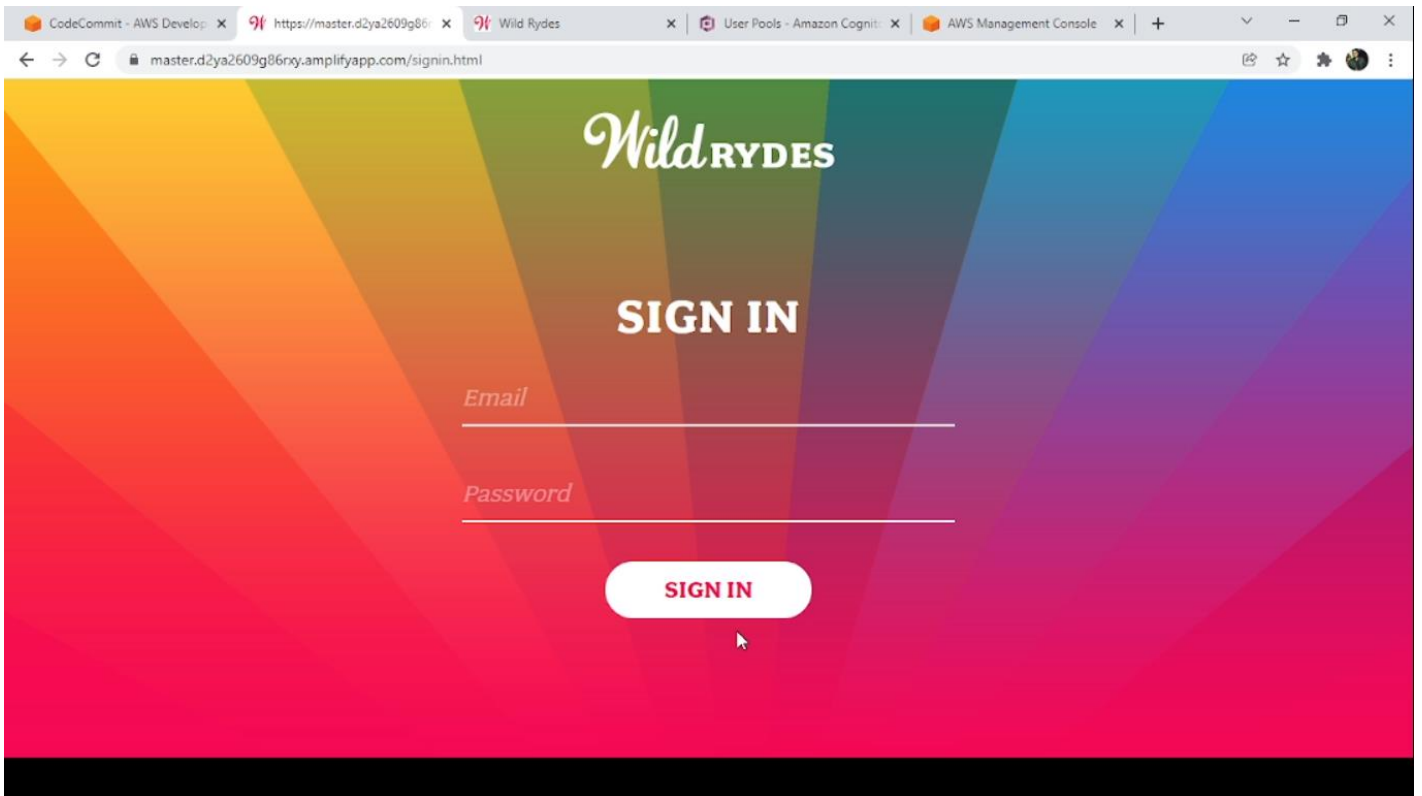


The screenshot shows a web browser window with the URL `master.d2ya2609g86nx.amplifyapp.com/register.html`. The page features a vibrant, multi-colored background with diagonal stripes in shades of yellow, orange, red, green, blue, and purple. At the top center, the logo "WildRYDES" is displayed in a white, stylized font. Below the logo, the word "REGISTER" is written in a large, bold, white sans-serif font. Underneath, there are three input fields for registration: "Email", "Password", and "Confirm Password", each with a white underline. At the bottom center, there is a white rounded rectangular button with the text "LET'S RYDE" in red. The browser's tab bar at the top shows several open tabs, including "CodeCommit - AWS Develo...", "https://master.d2ya2609g86...", "Wild Rydes", "User Pools - Amazon Cognit...", and "AWS Management Console".

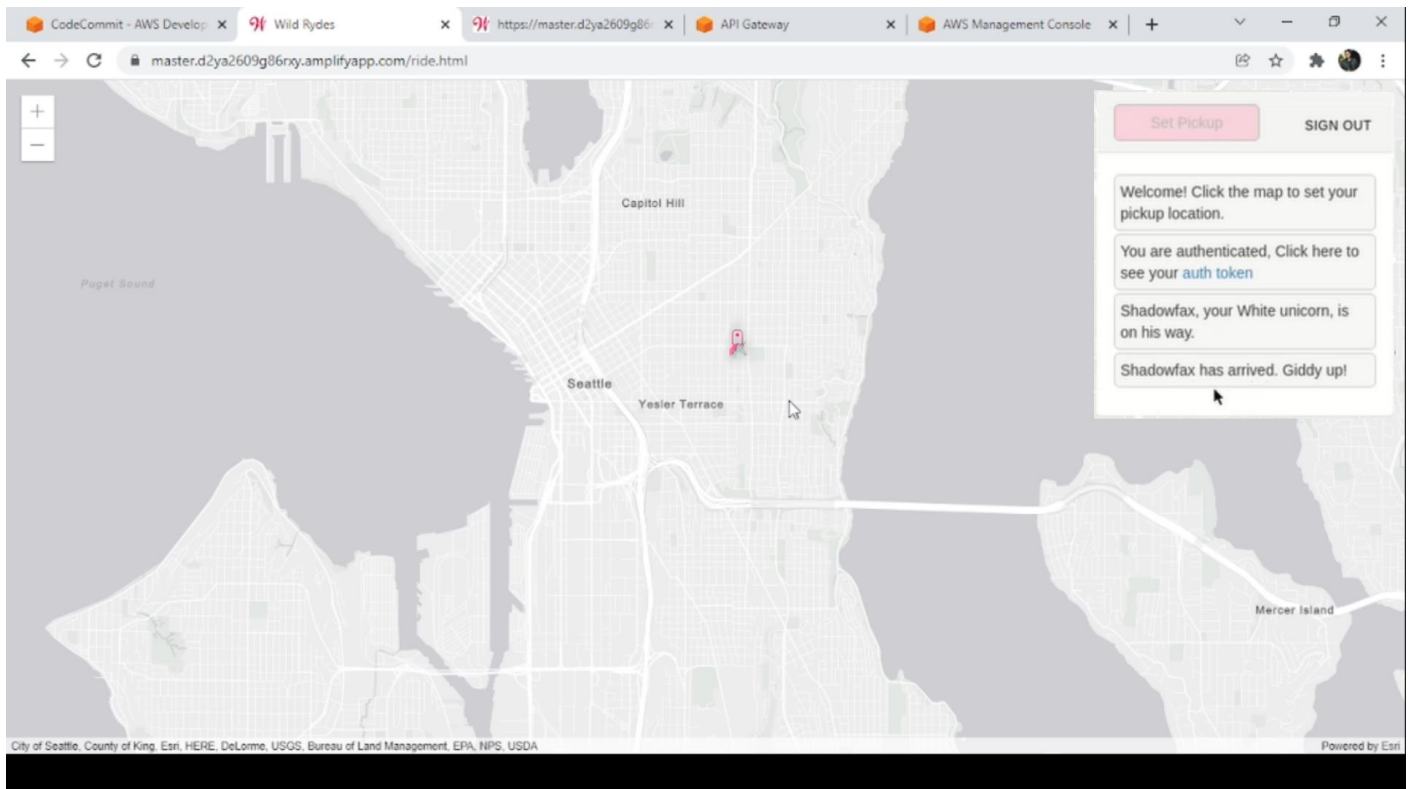
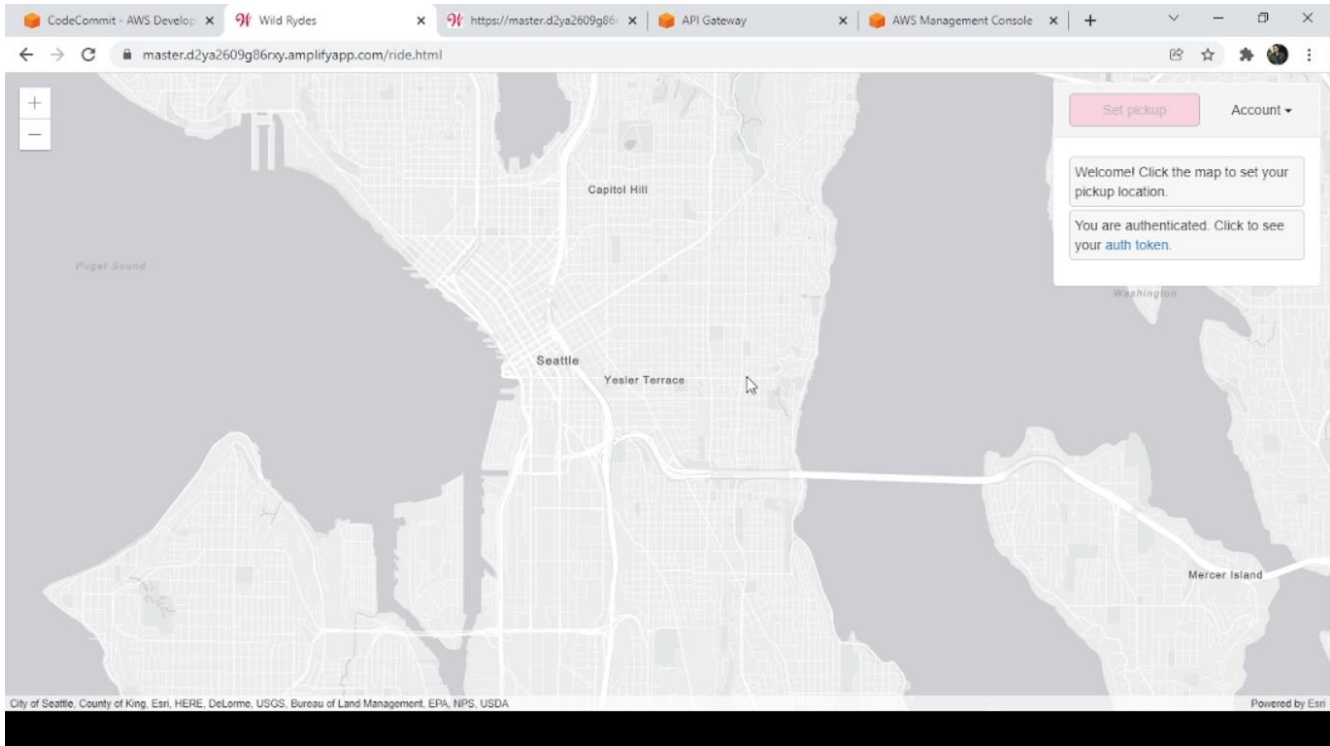
3. VERIFICATION PAGE:-



4. SIGN-IN PAGE:-



5. SET PICKUP AND REQUESTING UNICORN :-



6. DETAILS OF HOME PAGE AND INFORMATION OF UNICORNS:-

HOW DOES THIS WORK?

In today's fast paced world, you've got places you need to be but not enough time in your jam packed schedule. Wouldn't it be nice if there were a transportation service that changed the way you get around daily? Introducing Wild Rydes, an innovative transportation service that helps people get to their destination faster and hassle-free. Getting started is as easy as tapping a button in our app.



DOWNLOAD THE APP

Head over to the app store and download the Wild Rydes app. You're just a few taps away from getting your ryde.



REQUEST A UNICORN

We can get you there. Simply request a ryde on the app and we'll connect you with a unicorn immediately.



PICK A PRICE

Pick the valuation you're willing to pay and your ryde is set up. The only surge is the acceleration you get when taking off.



RIDE OFF TO SUCCESS!

After matching with your unicorn and agreeing to its terms, you'll be all set. Your unicorn will arrive shortly to pick you up.



BUCEPHALUS

Golden Swiss

Bucephalus joined Wild Rydes in February 2016 and has been giving rydes almost daily. He says he most enjoys getting to know each of his ryders, which makes the job more interesting for him. In his spare time, Bucephalus enjoys watching sunsets and playing Pokemon Go.

SHADOWFOX

Brown Jersey

Shadowfox joined Wild Rydes after completing a distinguished career in the military, where he toured the world in many critical missions. Shadowfox enjoys impressing his ryders with magic tricks that he learned from his previous owner.



5.1. CONCLUSION

The project has been appreciated by all the users in the organization. It is easy to use, since it uses the AWS provided in the user dialog. User-friendly screens are provided. The usage of software increases the efficiency, decreases the effort. It has been efficiently employed as a Site Management mechanism. It has been thoroughly tested and implemented.

The project “Wild-Rydes the serverless website” is the ideal place for every person who has go to a place urgently and not want to be stuck in traffic. The application will also provide facilities for users to register with the service and log in before requesting rides. The ‘Unicorn’ rides mentioned frequently is nothing but a bike ride that transports a customer from one place to another within a small area. This is equivalent to Rapido rides which are very famous in India. Because this is a project, the radius of operation is being kept limited to a small city, this same concept and idea can be used to deploy this system on a large scale

The software collects all the contact information of the user to provide asecure gateway for the user’s identity. The software provides a reliable platform for keeping all sensitive information.

For this kind of Online Business, special software must be installed on the server which host the site , or on a secure server that receives all the sensitive data.

The project was made keeping in mind both types of business ventures that can be undertaken. The first kind is profit-generating business, wherein the cost of the whole infrastructure will be paid by the fare rides around the city. The second type of venture can be a non-profit where in the service can be offered for free to the citizens by the government, the cost of infrastructure shall be paid by the government or by donations. The backend code used is open source, so that the same project can be implemented by other individuals or group of individuals for academic or service use cases.

7.1. REFERENCES

1. <https://www.w3schools.com>
2. <https://www.slideshare.com>
3. <https://www.tutorialspoint.com>
4. <https://www.youtube.com>

THANK YOU