ME3405

Semester Internship Report

Name: Abhishek Ghosh

Roll No: me21btech11001

Institute: Indian Institute of Technology Hyderabad

Department: Mechanical Engineering

Company Name: Circle Security (GoCircle AI India Pvt. Ltd.)

Role: Software Engineer Intern

Duration: 8th January 2024 – 1st June 2024

Introduction

This internship report provides an overview of my experience at Circle Security, a tech company specializing in Credential Free Authentication (CFA) and related security solutions. During my internship, I focused on backend implementation, API integration, and automation tasks, leveraging various tools and technologies to enhance the company's authentication and security processes.

About the Company

Circle Security (GoCircle AI India Pvt. Ltd.) is committed to revolutionizing the authentication process through its Credential Free Authentication (CFA) technology. The company's innovative solutions aim to streamline authentication while ensuring high security and ease of use for users. Circle Security operates primarily in the B2B domain, offering robust security solutions to its clients.

Website Link : https://circlesecurity.ai/

Tasks Accomplished during the Internship:

1. Understanding Credential Free Authentication & Setting Up Circle locally:

- Learned about Credential Free Authentication, which eliminates the need for traditional passwords, enhancing security.
- Installed and set up Circle on desktop and mobile devices, using email-based authentication and QR code scanning for login.
- Replicated and built functions for validating user sessions, including getSession, getUserSession, and expireUserSession.
- Implemented functions using NodeJS and Postman to interact with Circle Access APIs for session validation.
- Utilized cryptographic methods to generate and validate signatures for secure API requests.

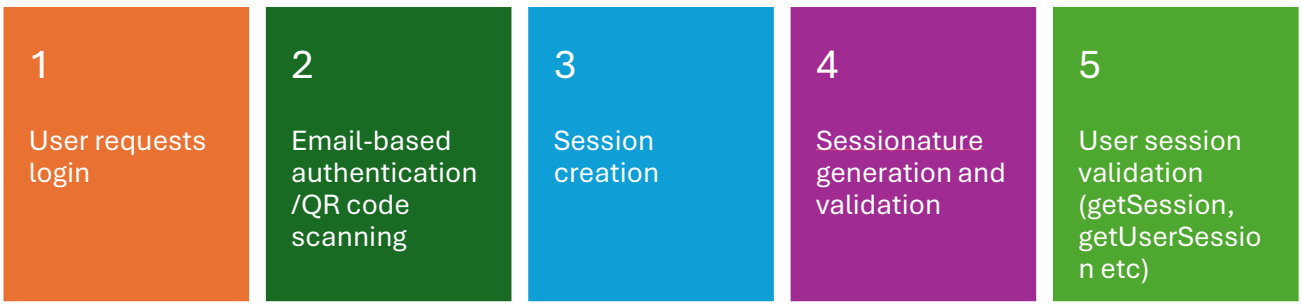


Fig: Steps involved in user login via Circle

2. User Acceptance Testing (UAT):

- To understand the product better performed testing ensuring functionality of the following products
    i. Circle Data Protection (CDP)
    ii. Circle Password Manager

3. API Integration and Development:

- **Signature Generation**: Explored Circle API documentation to understand signature creation and validation. Using the data in body, created signatures that fetched correct response on hitting the API using Postman (app key is passed as headers and while generating signatures, app key should be correct for authorization)
- **API Functions**: Developed the following functions to interact with Circle's APIs using NodeJS and .NET frameworks.
    i. GetSessionAsync
    ii. GetUserSessionAsync
    iii. CreateAuthorizationAsync
    iv. GetAuthorizationContract
    v. ExpireSessionAsync
    vi. ComputeSignature
- General overview of functions:
    i. The functions used fetch APIs to hit the url (GET/POST).
    ii. To check validity, signatures are matched.
    iii. A body along with a signature created using the input parameters and write key is sent to the APIs.
    iv. In response a JSON object is returned which contains mainly two things: data and signature.
    v. Using the data from the response a signature is created using the read key and matched with the signature received from the response for validation.
    vi. Different functions take in different parameters.
    vii. getAuthorizationContract takes authID as parameter which is only valid if createAuthorization (returns data which contains authID) is called first.

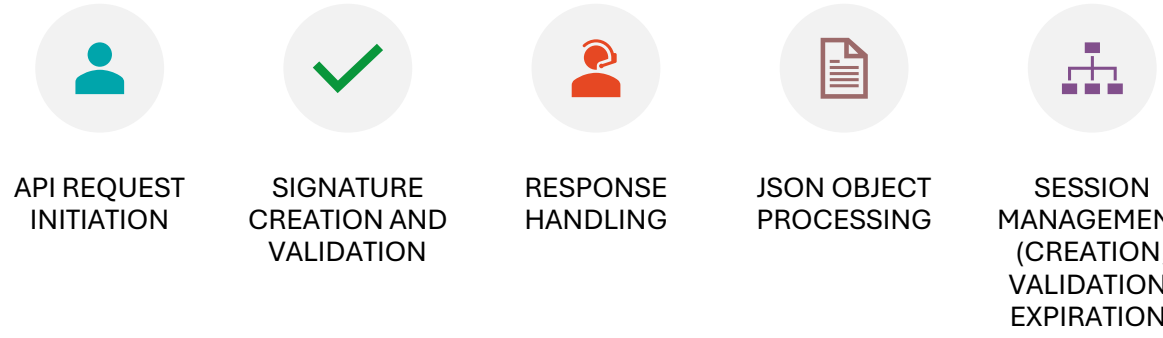
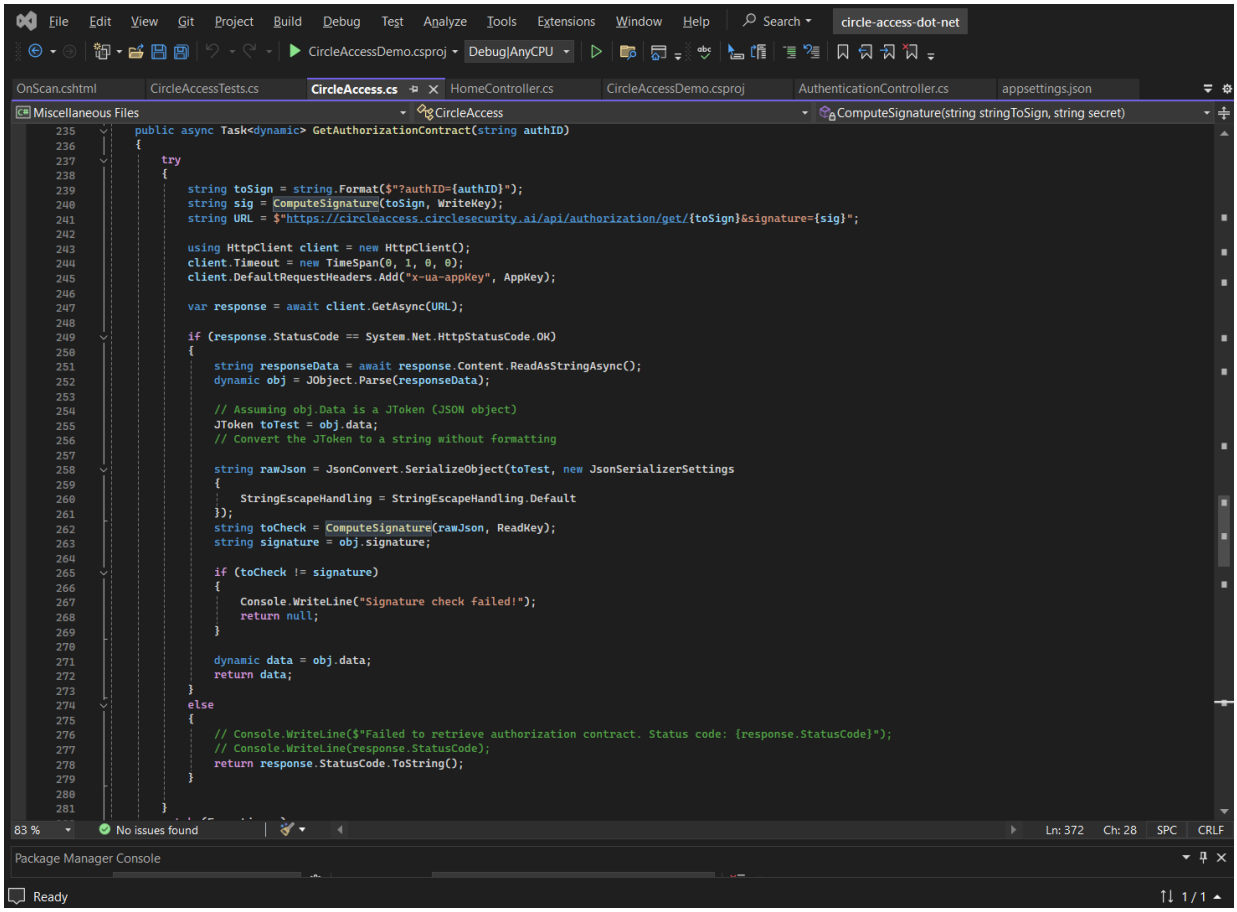


Fig: General overview of functions



```
public async Task<dynamic> GetAuthorizationContract(string authID)
{
    try
    {
        string toSign = string.Format($"?authID={authID}");
        string sig = ComputeSignature(toSign, WriteKey);
        string URL = $"https://circleaccess.circlesecurity.ai/api/authorization/get/{toSign}&signature={sig}";

        using HttpClient client = new HttpClient();
        client.Timeout = new TimeSpan(0, 1, 0, 0);
        client.DefaultRequestHeaders.Add("x-ua-appKey", AppKey);

        var response = await client.GetAsync(URL);

        if (response.StatusCode == System.Net.HttpStatusCode.OK)
        {
            string responseData = await response.Content.ReadAsStringAsync();
            dynamic obj = JObject.Parse(responseData);

            // Assuming obj.Data is a JToken (JSON object)
            JToken toTest = obj.data;
            // Convert the JToken to a string without formatting

            string rawJson = JsonConvert.SerializeObject(toTest, new JsonSerializerSettings
            {
                StringEscapeHandling = StringEscapeHandling.Default
            });
            string toCheck = ComputeSignature(rawJson, ReadKey);
            string signature = obj.signature;

            if (toCheck != signature)
            {
                Console.WriteLine("Signature check failed!");
                return null;
            }

            dynamic data = obj.data;
            return data;
        }
        else
        {
            // Console.WriteLine($"Failed to retrieve authorization contract. Status code: {response.StatusCode}");
            // Console.WriteLine(response.StatusCode);
            return response.StatusCode.ToString();
        }

    }
```

Fig: Snippet of the GetAuthorizationContract function

(Note: All the code has been made open source on the website)

- Parameters of different functions:
    i. **GetSessionAsync** : *sessionID*
    ii. **GetUserSessionAsync** : *sessionID, userdID*
    iii. **CreateAuthorizationAsync** : *returnURL, question, customID, approvals (array)*

iv. **GetAuthorizationContract** : *authID*
v. **ExpireSessionAsync** : *sessionID, userID*
vi. **ComputeSignature** : *string (object whose signature needs to be created), secret (Write/Read Key)*

- **Unit Testing:** Conducted headless testing using NUnit framework and System.net.http to test individual components of the software against various edge cases.
    i. **Headless Testing:** Performed end-to-end tests where the browser does not load the application's user interface.
    ii. **NUnit Framework:** Focused on testing individual software components.
    iii. **System.net.http:** Provided a programming interface for sending HTTP requests and receiving HTTP responses from a resource identified by a URL.

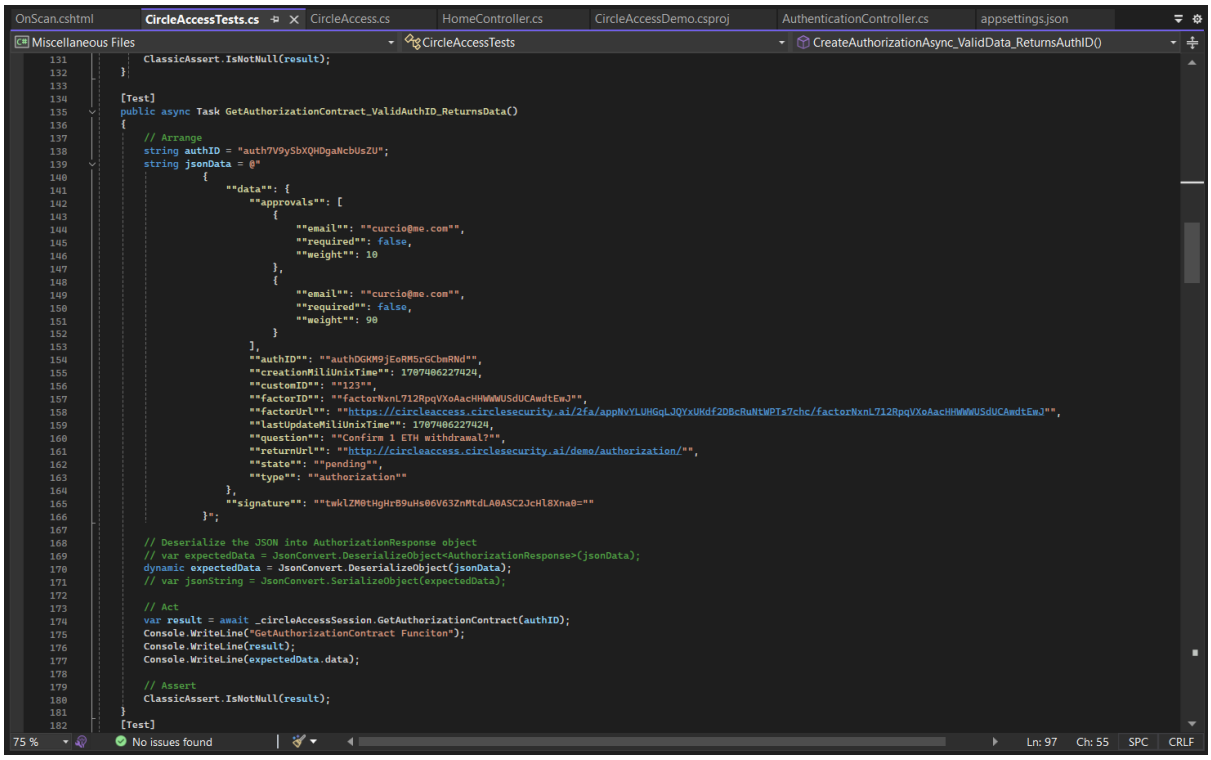


Fig: Snippet of one of the various test cases

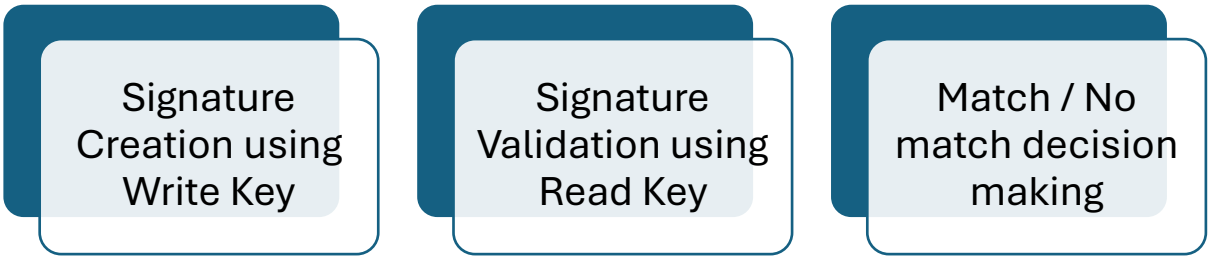


Fig: Signature Verification Process

4. Research and Data Collection:

- Researched authentication procedures used by brokerage firms like Zerodha and Upstox.
- Identified and collected contact information for over 800 VCs interested in cybersecurity and data security, using LinkedIn as a primary resource.

5. Automation with PowerShell and Microsoft Graph:

- **General overview:**
  i. **Script Creation:** Developed a PowerShell script to automate the creation and configuration of applications in Azure AD. (The script automated a 28 page Circle Manual, easing the workflow of the setup)
  ii. **Microsoft Graph Integration**: Integrated Microsoft Graph SDK to add API permissions and client secrets.
- **Detailed overview:**
  i. The script automates the following tasks
    1. Create Azure AD App Registration
    2. Add Application Permissions
    3. Grant Admin Consent
    4. Add Client Secret
    5. Enable Public Client Flows
    6. Add Redirect URI
    7. Add Custom Domain
    8. Save Configuration to JSON

9. Configure Tenant Settings
10. Verify Custom Domain
11. Merge SSO Configuration
12. Set Unique IDs for Users

- **Script development timeline:**
  i. Created a PowerShell script that uses Microsoft graph sdk to create a new application in Azure Ad.
  ii. After creation of the application, it adds API permissions to it and then adds client secret to the application. The following permissions are added:
    1. Directory.AccessAsUser.All
    2. Group.Read.All
    3. User.Read
    4. User.Read.All
  iii. Added a method in the script which upon accepted by the user grants admin consent to the API permissions.
  iv. Added code for redirect URL and custom domain.
  v. After the script is run all important records such as Client ID, Tenant ID, custom domain, Client secret and TXT record are converted to JSON file and is downloaded in the same directory of the user (with a unique name).
  vi. Added a http request (POST) in the script that would configure the tenant on the Client's side (add up details such as ClientID, TenantID, domain & secret) using the JSON returned previously.
  vii. Added domain verification using *Microsoft.Graph.Identity.DirectoryManagement*
  viii. Added a GET request that would use the AppKey and would return XML response.
  ix. Using the XML response and domainName merged a previous script (adds unique ids as well) that would set up the SSO.
  x. Fixed some authentication error due to wrong permissions granted in graph.

xi. Improved the code by adding user interface at stages of the script (y/n) and breakdown the code into small chunks.
xii. Added a Delete App registration upon user input.

References: Microsoft Graph Documentation

- **Script Documentation**:
  i. Created a comprehensive README.md file documenting the PowerShell script, including prerequisites, installation instructions, and workflow.
  ii. Included all the references used in making of the script.
- **Workflow Design**: Designed and proposed a workflow for the script from the user perspective, including end-to-end configuration and execution.
- **Video Demonstration**: Worked on a demonstration video showcasing the automation task and its functionalities.

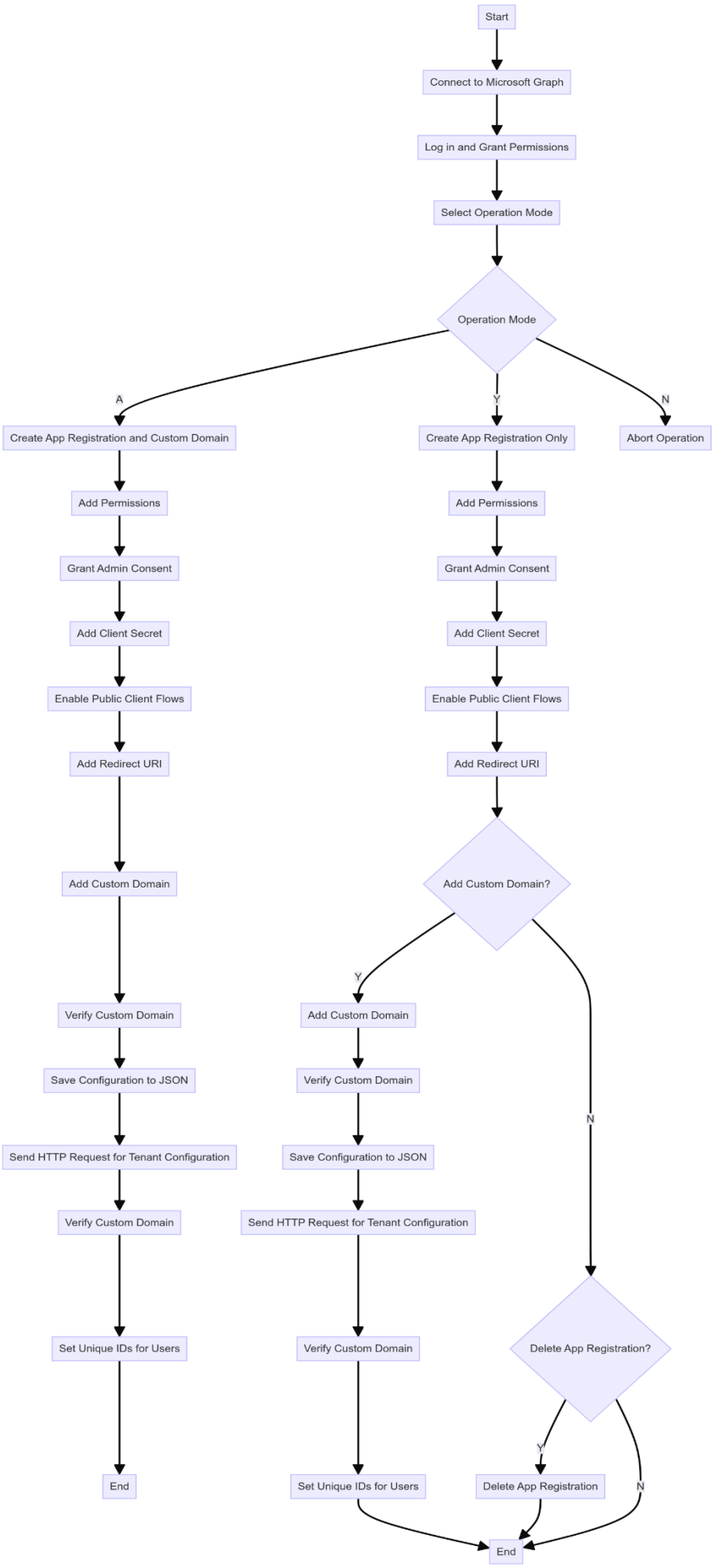


Fig: Flowchart of the Script

Skills Acquired

1. Technical Skills:

a. **NodeJS and .NET Development**: Gained hands-on experience in developing and converting API functions between NodeJS and .NET frameworks.
b. **API Integration**: Enhanced skills in integrating and interacting with APIs using cryptographic techniques for secure communication.
c. **PowerShell Scripting:** Developed and optimized automation scripts using PowerShell and Microsoft Graph SDK.

2. Testing and Validation:

a. **UAT and Unit Testing**: Conducted extensive User Acceptance Testing and unit testing using NUnit framework.
b. **Headless Testing**: Implemented and ran headless tests to validate functionality without loading the application's user interface.

3. Research and Analysis:

a. **Industry Research**: Researched authentication procedures and collected data on VCs, enhancing understanding of industry practices and investment interests.

4. Documentation and Communication:

a. **Technical Writing**: Created detailed documentation for automation scripts and developed high-level designs and workflows.
b. **Presentation Skills**: Prepared and presented a demonstration video, effectively communicating technical processes and solutions.

Learnings from the Internship

1. Technical Proficiency

a. **Backend Development**: Improved proficiency in backend development with a focus on security and authentication solutions.
b. **API Security**: Gained a deeper understanding of cryptographic techniques and secure API integration.

2. Problem-Solving and Adaptability

a. **Creative Solutions**: Developed problem-solving skills by addressing technical challenges and optimizing automation scripts.
b. **Adaptability**: Adapted to different technologies and frameworks, enhancing versatility as a developer.

3. Time Management and Collaboration

a. **Project Management**: Managed multiple projects and tasks efficiently, ensuring timely completion without compromising quality.
b. **Collaboration**: Worked collaboratively with team members and stakeholders, enhancing communication and teamwork skills.

Conclusion

My internship at Circle Security has been a transformative experience, providing me with valuable insights into the field of security and authentication. The diverse projects and tasks have not only sharpened my technical abilities but also honed my problem-solving and communication skills. As I move forward in my career, I am confident that the knowledge and experience gained during this internship will serve as a strong foundation for future opportunities in the tech industry. I am grateful for the practical experience and the valuable connections made during my time at Circle Security.