StoreAssist

Project Plan

**Team Name: TeamSaaS Akanksha Jaiswal**

**Jyothi H R**

**Manasi Sadanand Pai**

**Prarthana Hemanth**

Version 1.4

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 04/22/2020 | 1.0 | Initial Draft | Prarthana Hemanth |
| 04/23/2020 | 1.1 | Added updates regarding Chatbot | Akanksha Jaiswal |
| 04/24/2020 | 1.2 | Added architecture diagram and other minor updates | Prarthana Hemanth |
| 04/24/2020 | 1.3 | Added updates regarding TwitterBot | Manasi Sadanand Pai |
| 04/29/2020 | 1.4 | Updates regarding Chatbot and references | Akanksha Jaiswal |

Table of Contents

1. Introduction 4

1.1 Purpose of this document 4

1.2 Intended Audience 4

1.3 Scope 4

1.4 Definitions and acronyms 4

1.4.1 Definitions 4

2. Background and Objectives 5

3. Architecture & High Level Design 5

4. Organization 5

*4.1* Project group 5

4.2 Customer 6

5. Development process 6

6. Project risks 6

7. Communication 7

7.1 Collaboration 7

7.2 Git 7

8. Project plan 7

8.1 Time schedule 7

8.2 Test plan 8

9. References 10

# Introduction

## Purpose of this document

The purpose of this document is to provide a detailed project description of the application called StoreAssist, which is designed to help employees working at large supermarkets or grocery shops in their day to day activities. This document includes details about organization, roles, deliverables, project risks, time plans and financial plans.

## Intended Audience

This document shall be used in all phases of the project as a guideline. Intended audiences of this project are all project stakeholders:

* project supervisor
* project leader
* team members

## Scope

This document defines the project plan of the StoreAssist application. The overview includes objectives of the project, organization of the project team, development process that is going to be used during the project, assessment of possible risks, communication used between project stakeholders and project plan that includes time schedule and activity plan.

## Definitions and acronyms

### Definitions

|  |  |
| --- | --- |
| **Keyword** | **Definitions** |
| StoreAssist | The name of the project |
| Project Supervisor | A person in charge of supervising the project |
| Project Leader | A person in charge of organizing the team and communicating with the project supervisor |
| Team Member | An active member of the team responsible for making the job done |
| Milestone | A time in a project that marks the end of a project phase or the completion of an important deliverable. |
| Git | Version control system that will be used in this project |
| Scrum | An iterative and incremental agile software development method for managing software projects and product or application development |
| Kunagi | Web-based tool for integrated agile project management and collaboration based on Scrum |
| Scrum sprint | The basic unit of development in Scrum |
| Scrum master | Ensures the smooth working of the Scrum team and enforces Scrum practices |
| Product owner | Responsible for product management and its quality |

# Background and Objectives

StoreAssist is an app built on AWS which would be a one stop portal for employees working at a large Store/Super Market say Target, Walmart etc.

The main features of this app would be as follows:

•Capability to add in customers to the store’s rewards program. Here along with the basic details of the customer like Name, Email ID, Phone No., we would also take a picture of the customer and store it.

•Whenever a registered customer comes to the store, at the checkout counter, we would have the ability to recognize the person by clicking a picture and retrieving their reward points details. So this gives the customer the ability to redeem rewards even when they have not carried their Rewards card.

•Capability to post upcoming offers to Twitter, using a scheduled process. The employee would only have to update an excel sheet with the tweet and when it should be posted, and the rest would be handled by the system.

•A chatbot which would help the employee answer customer concerns and queries.

**The app is accessible at :**

<https://storeassist.auth.us-east-2.amazoncognito.com/login?response_type=token&client_id=2t1kmm2o49pa0148dd1hopfslo&redirect_uri=https://storeassist.manasipai.com/StoreAssist.html>

Test Credentials:

Username: prarthana

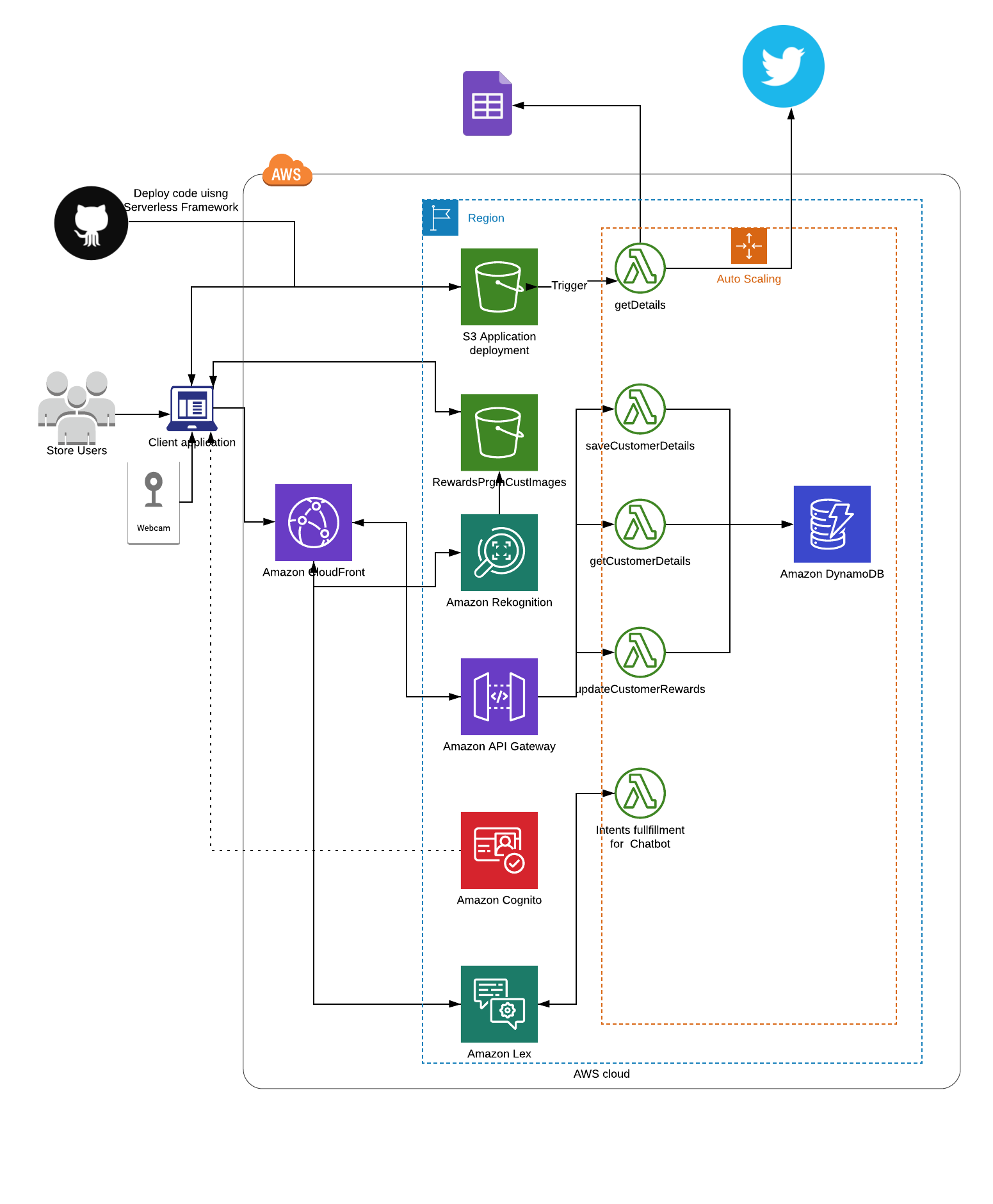
Password: Prar#1612

# Architecture & High Level Design

The app is built on the AWS Cloud with a Node JS backend and is completely serverless. Below is the design description of each of the modules:

* **Adding a new member:** For this functionality we are storing the basic details of the new member in DynamoDB and the corresponding image that is clicked in an S3 bucket. We are using AWS API Gateway to store the details in DynamoDB.
* **Retrieving a member:** For this functionality we are making use of AWS Rekognition to compare the image clicked with the image in S3. Based on that, we are retrieving details of the user from Dynamo DB using API Gateway.
* **Posting Offers to Twitter:** For this functionality we are making use of the serverless framework provided by from serverless.com. We create yaml file for creating the resources like the Lambda function and the cron job and deploy the files using serverless framework to AWS. The serverless framework deploys the files to S3 bucket, creates the lambda function that will be triggered according to the the cron job.
* **Chatbot:** We have built chatbot using Amazon Lex and lambda functions which fulfills the intents of the user. We have predefined some utterances which user can enter and based on those utterances, suitable lambda will be triggered and give the response back on chatbot UI.

Below is the architecture diagram for our app:



# Organization

## Project group

|  |  |  |
| --- | --- | --- |
| **Name** | **Initials** | **Responsibility (roles)** |
| Akanksha Jaiswal | AJ | Worked on building the chatbot using AWS LEX. |
| Jyothi HR | JH | Worked on retrieving member functionality and designing the overall architecture diagram. |
| Manasi Pai | MP | Worked on posting offers to twitter functionality and also deploying the app to AWS using AWS CodePipeline and Github webhooks. |
| Prarthana Hemanth | PH | Worked on Cognito login and adding a new member functionality. Also worked on integrating all the modules. |

## Customer

The target customers are employees of any large supermarket or retail stores. This app makes their day to tasks a lot easier. They can add in new reward program members, retrieve members, also answer customer queries using the chatbot and easily post new offers to social media.

# Development process

The app is built on AWS Cloud. Below is the development process followed:

* We divided the app into 4 individual modules, that each team member could work on independently.
* We laid out a design by deciding on the different AWS Services that would be leveraged.
* Once each of us completed the individual modules, we worked on merging the code and did a thorough end to end testing.
* Then we hosted the app on S3 and created a CloudFront distribution and hosted it using R53.
* We have used AWS Code Pipeline for CICD. The code is uploaded to git repository from where the Github webhook is created between the git repository and AWS Codepipeline and CodePipeline deploys the application code into S3 bucket created for deployment. CloudFront distribution is created for the S3 bucket and the CNAME for the disctribution is used in Route53 DNS. Whenever the code is updated in git automatically the AWS Codepipeline gets triggered and the web application is updated.

# Project risks

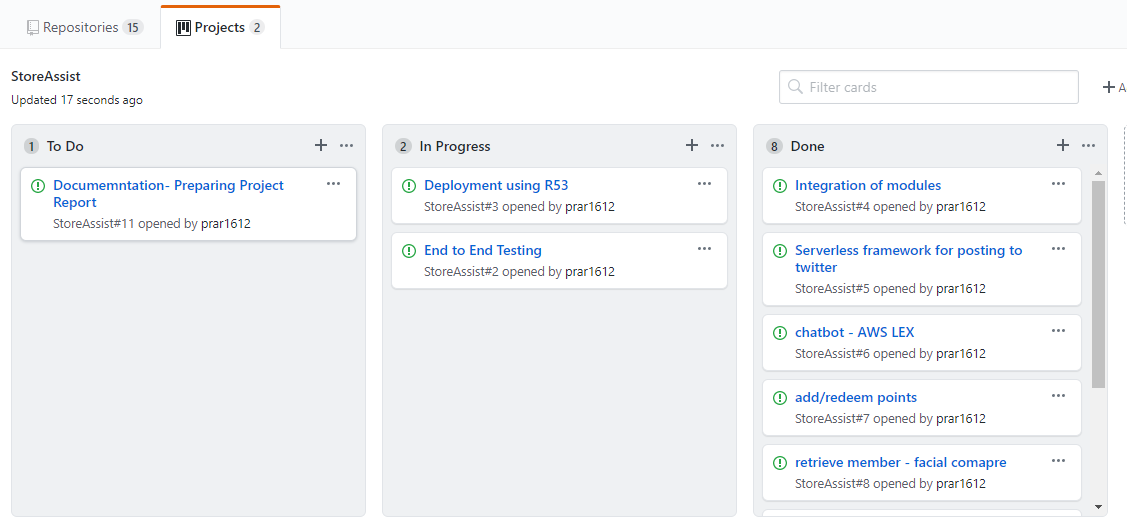
|  |  |  |
| --- | --- | --- |
| **Possibility** | **Risk** | **Preventive action** |
|  |  |  |

# Communication

We set up weekly calls to track progress of each module to ensure that we could deliver the project within timelines.

## Collaboration

We used GIT Project Board to create User Stories, so that we could track the progress. Below is the screenshot of the same:



## Git

Link to the Github Repo : <https://github.com/prar1612/StoreAssist.git>

# Project plan

## Time schedule

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **Milestone**  **Description** | **Responsible Dept./Initials** | **Finished week** | **Forecast Week** |
|  | High level design | All | Feb 5th | Feb 5th |
|  | Division of project into independent modules | All | Feb 12th | Feb 10th |
| 3. | Add a new member- store in dynamoDB | PH | Mar 5th | Mar 1st |
| 4. | Add a new member – store image in s3 | PH | Mar 20th | Mar 20th |
| 5. | Image comparison for retrieving member | JH | Mar 12th | Mar 10th |
| 6. | Retrieveing details from DynamoDB | JH | Apr 5th | Mar 30th |
| 7. | Built various Intents and lambdas on Lex to resolve employee’s general queries | AJ | Mar 15th | Mar 15th |
| 8. | Integration of Lex chatbot into website | AJ | Apr 10th | Apr 5th |
| 9. | Created Lambda function for business logic of the twitter bot | MP | Mar 15th | Mar 15th |
| 10. | Using serverless framework deployed the lambda and cron job | MP | Apr 7th | Apr 1st |
| 11. | Integration of Modules | PH | Apr 15th | Apr 10th |
| 12. | End to end testing | All | Apr 17th | Apr 15th |
| 13. | Deploying the app | MP | Apr 22nd | Apr 20th |

## Test plan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No. | 001 | Phase: | 1 | Author: | PH | Date: 03/22 |
| Test Category: | | **Unit Testing** | | |  |  |
| Software Product: | | Add a new member module | | | |  |
| Test Title: | | Testing the ability to add new member | | | | |
| Test Purpose: | | Testing the ability to add new member | | | | |
| Test Setup: | | User should be able to login through cognito and view the main add a member screen. | | | | |
| Prerequisites: | | NA | | | | |
| Procedure: | | Fill in the add a new member form and click submit. In the next screen, click an image using the webcam and click Take Snapshot. | | | | |
| Checks: | | User should be able to fill in the form and webcam should be operational. | | | | |
| Expected Results: | | Image of user should be saved in S3 bucket with email id of the user and the details filled in the form should be in DynamoDB. | | | | |
| Result: | | Image of user is saved in S3 bucket with email id of the user and the details filled in the form are in DynamoDB table. | | | | |
| Reason for Failure: | | NA | | | | |
| Remarks: | | NA | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No. | 002 | Phase: | 1 | Author: | JH | Date: 04/10 |
| Test Category: | | **Unit Testing** | | |  |  |
| Software Product: | | Retrieve a member | | | |  |
| Test Title: | | Retrieve a member | | | | |
| Test Purpose: | | Check if member is retrieved correctly | | | | |
| Test Setup: | | A customer should be registered initially which we will try to retrieve. | | | | |
| Prerequisites: | | A customer should be registered initially which we will try to retrieve. | | | | |
| Procedure: | | Go to Retrieve Member tab and click an image using webcam and click the authenticate button. | | | | |
| Checks: | | The webcam should be functional. The image and user details should be retrieved. | | | | |
| Expected Results: | | The user’s image is retrieved from s3 along with details from dynamoDB | | | | |
| Result: | | The user’s image is retrieved from s3 along with details from dynamoDB | | | | |
| Reason for Failure: | | NA | | | | |
| Remarks: | | NA | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No. | 003 | Phase: | 1 | Author: | JH | Date: 04/10 |
| Test Category: | | **Unit Testing** | | |  |  |
| Software Product: | | Retrieve a member | | | |  |
| Test Title: | | Retrieve a member and add/retrieve points | | | | |
| Test Purpose: | | Check if you are able to add points to retrieved user | | | | |
| Test Setup: | | A customer should be registered initially which we will try to retrieve. | | | | |
| Prerequisites: | | A customer should be registered initially which we will try to retrieve. | | | | |
| Procedure: | | Go to Retrieve Member tab and click an image using webcam and click the authenticate button. Once user is retrieved enter the number of points to be added as 10 and click Add button | | | | |
| Checks: | | The webcam should be functional. The image and user details should be retrieved., and the points should get added. | | | | |
| Expected Results: | | 10 points gets added to the user’s reward points in DynamoDB. | | | | |
| Result: | | 10 points gets added to the user’s reward points in DynamoDB. | | | | |
| Reason for Failure: | | NA | | | | |
| Remarks: | | NA | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No. | 004 | Phase: | 1 | Author: | AJ | Date: 04/12 |
| Test Category: | | **Unit Testing** | | |  |  |
| Software Product: | | Test the chatbot | | | |  |
| Test Title: | | Testing the chatbot | | | | |
| Test Purpose: | | Testing the chatbot | | | | |
| Test Setup: | | User should be able to login and click on the chat button. | | | | |
| Prerequisites: | | User should be able to login and click on the chat button. | | | | |
| Procedure: | | Click on the chat button and request for reward program details. | | | | |
| Checks: | | The chat button should be visible and user should be allowed to enter in the chat window. | | | | |
| Expected Results: | | On requesting reward program details, the available reward program options should be displayed. | | | | |
| Result: | | On requesting reward program details, the available reward program options are displayed. | | | | |
| Reason for Failure: | | NA | | | | |
| Remarks: | | NA | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No. | 005 | Phase: | 1 | Author: | MP | Date: 04/12 |
| Test Category: | | **Unit Testing** | | |  |  |
| Software Product: | | Test the twitter Bot | | | |  |
| Test Title: | | Testing the twitter Bot | | | | |
| Test Purpose: | | Testing the twitter Bot | | | | |
| Test Setup: | | User should be able to edit Google Spreadsheet. cron job is updated to 5 mins | | | | |
| Prerequisites: | | Google Spreadsheet should be editable. | | | | |
| Procedure: | | User updates row in Google Spreadsheet with offer details. | | | | |
| Checks: | | The Google Spreadsheet should be editable | | | | |
| Expected Results: | | The Twitter account should see the only the new offer details tweet after 5 mins. | | | | |
| Result: | | The Twitter account should see the only the new offer details tweet after 5 mins. | | | | |
| Reason for Failure: | | NA | | | | |
| Remarks: | | NA | | | | |

# References

<https://medium.com/@sjehutch/amazon-facial-recognition-using-rekognition-ba01510085b2>

<https://www.youtube.com/watch?v=P8okmPWIAcQ>

<https://codeburst.io/develop-server-less-chat-bot-in-minutes-for-your-resume-7bd7725d8e77>

<https://serverless.com/>

<https://dev.to/lorenzotenti/how-to-build-a-serverless-twitter-bot-lph>

Images:

Banner: <https://www.freepik.com/free-vector/shopping-banners-pack_808595.htm>

Google Sheets Image: <https://www.cleanpng.com/png-google-docs-online-spreadsheet-google-analytics-4599956/> -

Logo: <https://londontheinside.com/wp-content/uploads/2019/07/CGAP1009_SKU_2.jpg>