



Cloud Computing Assignment 3

Project Title : “List-It: A To-Do List Application”

Deployed Project URL : <http://ebcognitoappenv.eba-mm33x.us-east-1.elasticbeanstalk.com/>



Table of Contents

Deployed Project URL :	0
Signed Contribution Agreement	3
Summary	4
Introduction	5
Related Work	7
Software Architecture	8
• AWS Elastic Beanstalk (EB):	8
• Amazon EC2:	9
• Amazon Cognito:	9
• Amazon Simple Storage Service (S3):	9
• Amazon DynamoDB database:	9
• Amazon Simple Email Service (SES):	9
• Amazon Lambda:	10
• Amazon CloudWatch:	10
• Amazon Kinesis Data Stream and Data Firehose	10
• Amazon IAM:	10
Developer Manual	11
• AWS Elastic Beanstalk	11
• AWS Cognito	12
• AWS DynamoDB	14
• AWS Simple Storage Service S3	15
• AWS Lambda	16
• AWS CloudWatch	17
• AWS Simple Email Service	17
• AWS Kinesis Data Stream and Data Firehose	18
• AWS IAM	20
User Manual	21
• Register	21
• Account Verification	22
• Login	22
• Favourites/Home	23
• All Tasks	23
• Add New Task	24
• Task Details	25
• Subtask Details	26

• Marking Tasks or Subtasks as Completed.....	27
• Emails of the Tasks which are due next day.....	29
• User Profile	30
References	31

Table of Figures

Figure 1: System Architecture	8
Figure 2: Elastic beanstalk Environment	12
Figure 3: Example Users in the Userpool.....	13
Figure 4: General Settings of Userpool	13
Figure 5: Task Table structure and sample entries.....	14
Figure 6: Subtask table structure and sample entries.....	15
Figure 7: S3 bucket for Task Images: cca3imgs.....	15
Figure 8: S3 bucket to store User profile images: cca3users.....	16
Figure 9: AWS Lambda function overview.....	17
Figure 10: AWS CloudWatch event overview.....	17
Figure 11: AWS SES sender email verification.....	18
Figure 12:Kinesis Data Stream	18
Figure 13:Kinesis Delivery Stream	19
Figure 14:Task table Data Stream enabled	19
Figure 15: Subtask table Data Stream enabled	20
Figure 16: IAM rules example	20
Figure 17:Registration Page.....	21
Figure 18: User Email Verification	22
Figure 19: Successful Login of New User.....	22
Figure 20: Favourite Tasks	23
Figure 21: All Tasks List.....	24
Figure 22: Add a new task	25
Figure 23: Task Details	26
Figure 24: Subtask Details	27
Figure 25: Subtask Completion	28
Figure 26: Task Completion	29
Figure 27: Email Alert Notification	30
Figure 28: User Profile	30

Signed Contribution Agreement

Student Name: Sohee Kim	Student Name: Pranamya Korde
Student ID: s3720307	Student ID: s3779009
Contributions: <ol style="list-style-type: none"> 1. Develop project Idea 2. Development of technologies for application 3. Creating database and exporting it to DynamoDB 4. Development and deployment of app 5. Implementation of services 	Contributions: <ol style="list-style-type: none"> 1. Develop project Idea 2. Design of project architecture 3. Implementing CSS into website 4. Writing report 5. Implementing HTML to frontend of the website
Contribution Percentage: 50%	Contribution Percentage: 50%
<p>By signing below, I certify all information is true and correct to the best of my knowledge.</p> <p>Signature: Sohee Kim</p> <p>Date: 13/06/2021</p>	<p>By signing below, I certify all information is true and correct to the best of my knowledge.</p> <p>Signature: Pranamya P Korde</p> <p>Date: 13/06/2021</p>

Summary

The objective of this project was to create a web-based application that would be helpful for users in their daily lives to increase the productivity and efficiency in daily time management. We aimed to use AWS cloud related services to design and implement the highly scalable, robust, and distributed architecture. We have used Boto3 python SDK to access the AWS services. Considering the timeframe and scope of a project, our project application 'List-it' utilizes most of the possible AWS cloud infrastructure and services like compute, storage, database, network and content delivery, security, and Authorization. It allows registered users to create and track their tasks in the form of tasks list. Users can store list of subtasks related to a main task in it. Each subtask may contain different details like Images related to the subtask and URL links related to the subtask. Users can make some of the tasks as their favorites to track them easily. The application also has a functionality to send an email to the users regarding the list of tasks which are due in next few days as an alert notification.

Introduction

The objective of our project is to create AWS powered cloud application that would be helpful for our users in their daily lives to better organize their day-to-day tasks so that they can better manage their time. We aim to build a web-based application which will play as a role of a To-Do list application with the use of cloud-based technology and services. The idea behind this is we all plan a lot of things in our day-to-day life but because of the unimagined circumstances and rush in life, we kind of miss some of these planned tasks. And it is scientifically proven that writing down daily tasks somewhere results in the better management of time and efforts to complete them with efficiency. How big or small the tasks are, if we write them down somewhere, we can keep a track on them, we can better organize time to complete them, we can better organize our daily life around them to make more out of our daily efforts.

For example, consider a life of average computer engineering student named Josh who lives alone at the university campus. He has 4 different subjects each semester which have minimum 3 assignments per subjects, some of those are group assignment so require weekly meetings. He is working at a café as a part-time barista where he needs to keep a track on his weekly shift timings and weekly earnings so that he can calculate the monthly expenditure from it. He assigns a list of whole lot of tasks to himself to complete within a day or a week but can only complete few of them because of the unforeseen circumstances and rush in daily life. But if he can write down the daily/weekly tasks somewhere, it will better help him in managing his time to work on every task in the written list. That's where our application plays an important role in providing the access to create and track the lists of daily tasks, weekly targets, and monthly plans. This application will help him in better organizing his time by giving alerts if the due date of a particular task is within next 3 days. He can categorize the tasks based on their importance and make them favourite to better track them.

The motivation behind choosing this topic for our project is we all lack somewhere in organizing our time in daily lives and an application like this will personally of a great use to organize the time and efforts in more efficient way. This application is powered by many AWS cloud services which are running on the internet and are scalable, flexible, and agile. Cloud related services and cloud computing technologies are in a boom now a days because of many obvious reasons. Cloud Computing is the delivery of on-demand services related to the application infrastructure over the internet and usually on a pay-as-you-go basis [1]. Rather than owning the whole infrastructure, from data centres to the computational servers, cloud computing gives benefit by providing these services on the internet and as per the demand of a user [1]. This reduces the upfront cost, complexity of owning and maintenance of the IT infrastructure for the company. Cloud computing underpins vast variety of services. That includes, consumer services like from basic services like storage, networking, and processing power through to natural language processing, artificial intelligence, and various security and blockchain related services as well [2].

The proposed application makes use of many AWS services to implement functionalities like User management, increase/decrease scalability, store and manage the data, manage connections, handle security of user data, and handle deployment of the application.

The functionalities provided by the application 'List-It' are summarized as follows:

- User can store a task name and list of sub-tasks along with the other details of them in it.
- Whenever all sub-tasks are marked as done, the main task is automatically marked as done.
- For example:

Task 1 : CC Assignment 1 (Done)

sub-task1 : Get access to GCP (Done)

sub-task2 : Decide Language to write a code (PHP or Flask) (Done)

sub-task3 : Register for Demo of Assignment 1 (Done)

sub-task4 : Submit code for the assignment (Done)

sub-task5 : essential readings for the implementation (Done)

Task 2 : CC Assignment 2

sub-task1 : Get access to AWS(Done)

sub-task2 : Decide Language(Done)

sub-task3 : Register for Demo.

sub-task4 : Submit code.

Task 3 : Movies to Watch

sub-task1 : Movie1

sub-task2 : Movie2 (Done)

sub-task3 : Movie3

etc.

- User can store various details for sub-tasks like: Title, Description, Due Date, Images, links, URLs etc.
- Moreover, User can bookmark tasks as important or favourite which he can directly access from the favourites tab instead of going through each of the task in all tasks list.
- User can edit/delete the tasks and sub-tasks at any point of time.
- User will get notification email in the morning if the due date of a particular task is tomorrow.
- User account is created for each user along with his personal details and the profile image which can be accessed and edited at any point of time.

Related Work

The main functionality of 'List-It' is to provide a platform for registered users to list down their daily/weekly/monthly tasks and keep a track on them to complete them efficiently withing the targeted time limit. There are numerous numbers of similar web and mobile applications in terms of the to-do list related functionality. This section mainly addresses one web application and one mobile service application which we think are closer to the actual design of our application.

Firstly, the famous application to create a to do list to organize work and life named **Todoist**. Todoist is a much bigger application which serves variety of different functionalities along with the collaborative organization of tasks within a particular organization or company. The main functional difference between this application and proposed application is the proposed solution is only for the better management of the personal daily work and it won't provide any collaborative features till now. The main similarity between this application and proposed application is that they both gives confidence to users that by organizing the tasks digitally, users can make progress on things which are important for them [3]. Lot functionalities in our applications are designed by getting inspiration of similar functionalities introduced by this application. For example, the functionality having sub-tasks or sections for each created tasks which makes them easier to keep track on.

The next is mobile service application named **WiseList** by the company **Frimus Technology** private limited. This mobile application will help users to better organize the tasks in more graphical way and share them with their family, colleagues, and team members. This application can be used at personal as well as the professional level at a small-scale business. As WiseList is mobile, it will send the time and location-based reminders to its users.

The major similarity of this application with our designed prototype is in functionality of setting reminders before the due date of the task and adding a functionality provide quick creation of the tasks. The major difference between this application and List-It is this application provide a collaboration or sharing of tasks within family or team members of a team. Our application is focused only on the personal use as user may store the personal short-term goals in it.

Software Architecture

The high-level architecture diagram of the List-It application along with communication between different cloud components (AWS services) is presented as follows:

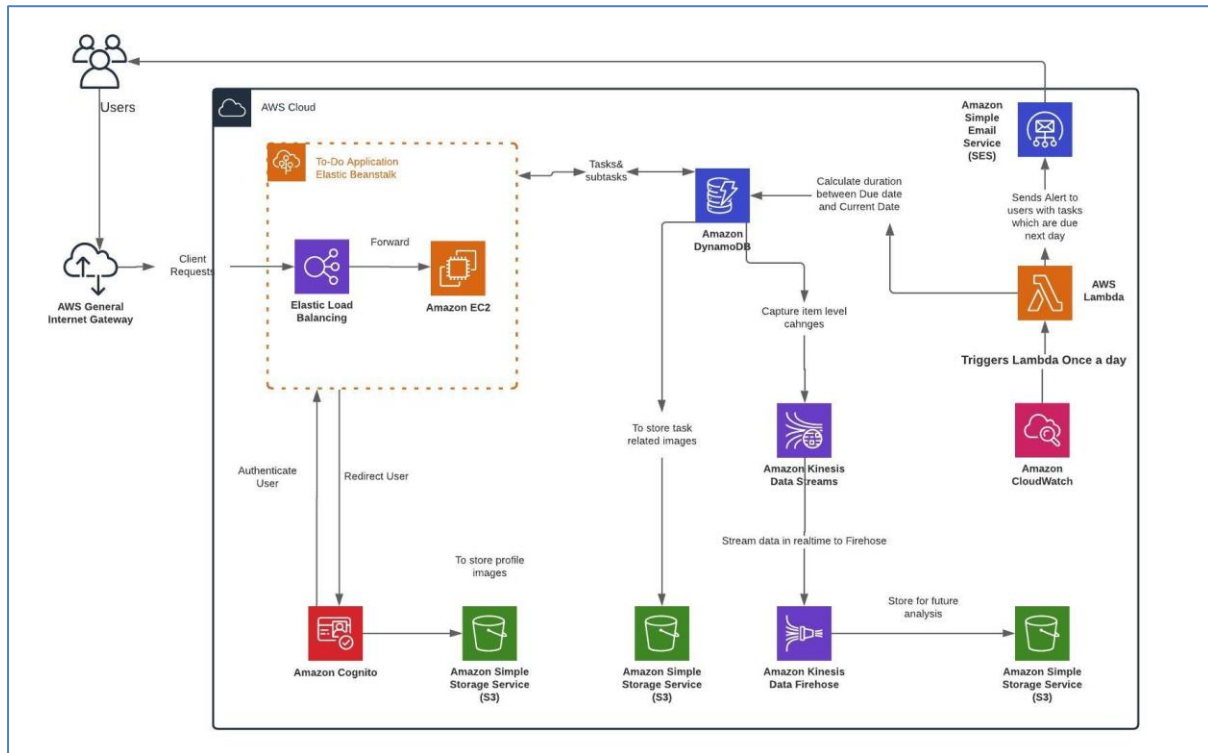


Figure 1: System Architecture

In this architectural diagram, AWS Elastic Beanstalk along with its components like Elastic Load Balancer and AWS EC2 are used for the deployment handling, computation and workflow management tasks, Amazon DynamoDB, and Amazon S3 buckets are used for storing and managing the data of the application, AWS Cognito is used for User Authorization, synchronization and user identity management purpose and AWS Lambda, Amazon SNS (Simple Notification Service) and SES (Simple Email Service) are used for implementing the alert mechanism in the application. Below are the details of each of the cloud component used along with their technical details:

- **AWS Elastic Beanstalk (EB):**

AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling the web applications and services developed with many languages including Java, Python, PHP, Node.js, etc on the familiar servers like Apache, Nginx, Passenger etc [4]. It is fastest and simplest way to deploy a web-application on AWS cloud as we only need to upload the code of the application and EB as a service handles all other deployment configurations automatically. These configurations include capacity provisioning, auto-scaling, load balancing, application health monitoring etc [4]. The List-It application is built using Python

based Flask framework for which the platform to upload the application code is provided by the Elastic Beanstalk.

- **Amazon EC2:**

Amazon EC2 instances are the part of AWS resources provided by the platform of Elastic Beanstalk under web server environment tier which is getting used by the application List-It.

- **Amazon Cognito:**

User Authentication and User data management plays an important role in terms of security, session management, synchronization of our application. AWS Cognito is a service which provide easy way to authenticate and authorize user to the application specific data by providing a JWT tokens after successful logon process. List-It application uses a custom User Pool as a directory to store the users along with their verified email id, username, first name, last name, phone number and profile image name during register and login option.

- **Amazon Simple Storage Service (S3):**

Amazon Simple Storage Service is a highly durable, and scalable data storage for any application. In a S3 bucket, data is stored as an object than as a file [5]. The data storage is highly scalable, and we can store large volume of data in these S3 buckets. Storage and retrieval of data from S3 is also an easy process due to which it is one of the most efficient services provided by the AWS cloud platform. S3 buckets are used in List-It to interact with Cognito and DynamoDB tables to store Task specific images and User Images in two separate buckets.

- **Amazon DynamoDB database:**

AWS DynamoDB is a No-SQL key-value paired database. As it is an unstructured database, it is used in List-It application store Tasks and Sub-Tasks specific data. Two tables are created to store Tasks and Subtasks. Owner of a task is mentioned in the Task table while Parent Task of a subtask is mentioned in subtask table to keep the foreign key constraint active while storing the data. Tasks and subtasks shown on the user platform are accessed directly from the DynamoDB tables.

- **Amazon Simple Email Service (SES):**

Amazon SES is a cloud-based email sending service which allows developers to send emails over a cloud platform to their users [6]. SES is used in our application to email the task details to the users which are due in next 1 day from the current date. Whenever, the user registers himself, he gets added to the SES recipients list later from which user get emails regarding the tasks which are due in next 1 day. This is the important feature, as it helps user to increase the priority of those tasks which are due next day.

- **Amazon Lambda:**

AWS Lambda is a compute service that lets you run your code without provisioning and managing any servers [7]. We can either invoke the lambda functions with lambda API or we can use these functions as a response to the event happened in other AWS services like S3 and DynamoDB. In our application. The benefits of using Lambda are they are serverless functions, developers don't need to pay for the managing servers, resources, maintenance etc. Developer only needs to focus on the code he is writing. We are invoking lambda function to trigger a simple email service (SES) whenever for a particular subtask, the due date is within next 1 day from a current date.

- **Amazon CloudWatch:**

AWS CloudWatch is a monitoring and observability service built for developers, DevOps engineers, IT managers etc which can provide data related insights to monitor the applications and to respond to system-wide performance changes and optimize resource utilization [8]. We used CloudWatch in List-It application to invoke a lambda function across all the tasks of all the users once a day. That lambda function checks if the due date of a particular task is due tomorrow and if so, it will create a use wise list of tasks which are due tomorrow and send an email to them on their registered email id. This event invokes a lambda function once a day at around 4.00 pm.

- **Amazon Kinesis Data Stream and Data Firehose**

Amazon Kinesis Data Streams and Data Firehose (Delivery Streams) are the services under AWS Analytics category. Although the application currently cannot use for the analytics purpose, these services can be used for logging purpose. Amazon Kinesis streams allows streaming of the log files which consists of logs about the item level changes made in DynamoDB tables. Whenever a record is inserted, updated, or deleted from Task or Subtask tables, a log of old image and new image is created into the S3 buckets in a real time for analysis purpose. If the intensive analysis and research is required, these log files data can be stream to the Amazon Redshift or Athena data storages on which intensive analytics is possible. In future, this log files can be used for a functionality to revert back some changes made by the user if user want to undo those changes.

- **Amazon IAM:**

AWS Identity and Access Management service enables the developers to manage access to AWS services and resources in a more secure way [9]. Each user should have a specific IAM role assigned to him to get a particular level of access to a particular service of AWS.

Developer Manual

• AWS Elastic Beanstalk

We have used AWS Elastic Beanstalk to deploy, scale and synchronize our web application. Elastic Beanstalk has been created by following the instructions [10].

Extra steps to take to build an EB environment:

- Select Managed platform. For Platform, select Python, and for Platform branch select Python 3.6 running on 64-bit Amazon Linux.
- Choose Configure more options and in the Instances, section set EC2 instance security group to group we created above.

The Elastic beanstalk can be externally arranged with the AWS CLI. Below are the steps to configure the environment and upload a web-application code file.

1. To check if the project you are building is following the similar project structure.

File structure

```
<Dir-name>
├── application.py
├── requirements.txt
├── <virtual-env-name>/
│   └── // stuff
├── static/
│   ├── static.js
│   └── style.css
└── templates/
    └── // html files
```

2. Then we can follow the below commands to create an Elastic Beanstalk Environment. For that, the awsebcli is required.

Setup

- Install awsebcli
 - o `pip install awsebcli`
- Create a folder and enter:
 - o `mkdir <dir-name>`
 - o `cd <dir-name>`
- Create virtual environment:
 - o `python -m venv <virtual-env-name>`
 - o `source <virtual-env-name>/bin/activate`
- Install Flask:
 - o `pip install flask==1.0.2`
 - o `pip install boto3`
- Write code:
 - o Main python file has to be named as `application.py` and have `application = Flask(__name__)`

3. For deploying the application for the first time, we need to install all the required packages. We have stored those packages in requirements.txt file which we have to run through pip command to install all these packages. We ignored the virtual environment in our source code using .ebignore file. For the first time deployment process we have to initialize the Elastic Beanstalk with eb init command

Steps to deploy for the first time:

- Create a file that tells Elastic Beanstalk to install the libraries during deployment:
 - o `pip freeze > requirements.txt`
- Create .ebignore and add the virtual environment:
 - o `echo <virtual-env-name> > .ebignore`
- Initialise
 - o `eb init -p python-3.6 <app-name> --region us-east-1`
 - o `eb init`
 - o `eb create <env-name>`
- Open the web page.
 - o `eb open`

Steps to redeploy next time onwards:

- Do this if you install something:
 - o `pip freeze > requirements.txt`
- `eb deploy`
- `eb open`
-

Screenshot of the EB environment:

Environment name ▲	Health ▼	Application name ▼	Date created ▼	Last modified ▼	URL ▼	Running versions ▼	Platform ▼	Platform state ▼	Tier name ▼
ebcognitoappenv	OK	ebcognitoapp	2021-06-07 22:00:33 UTC+1000	2021-06-09 04:23:46 UTC+1000	ebcognitoappenv.eba-mmww3m33x.us-east-1.elasticbeanstalk.com	app-fd54-210609_042235	Python 3.6 running on 64bit Amazon Linux	Supported	WebServer

Figure 2: Elastic beanstalk Environment

• AWS Cognito

The User pool with all the default settings has been created to serve the basic purpose of User data management and authorization [11]:

It only requires email of a user to validate a different user and along with their respective passwords to authorize them. On the New User Registration page (Sign Up), user can enter their details as their Username, email, phone number, given name, family name, profile picture etc. Although all these fields except email are not compulsory fields, we made the register form require all the attributes.

The profile images of a verified users are stored into a S3 bucket with username as a key. Boto3 is used to manage the user pool. For accessing the userpool with Boto3, the 3 main keys like USER_POOL_ID, CLIENT_ID and CLIENT_SECRET are required which can be found in the general settings of the Userpool.

Below code is from application.py

```

USER_POOL_ID = ''
CLIENT_ID = ''
CLIENT_SECRET = ''
client = boto3.client('cognito-idp', region_name='us-east-1')

```

Example of users in the user pool:

Username	Enabled	Account status	Email	Email verified	Phone number verified	Updated	Created
abc	Enabled	CONFIRMED	tyw11620@eoopy.com	true	false	Jun 10, 2021 6:24:46 AM	Jun 3, 2021 8:35:31 AM
asdf	Enabled	CONFIRMED	reza90@meadowutilities.com	true	false	Jun 9, 2021 2:15:45 AM	Jun 3, 2021 12:02:47 PM

Figure 3: Example Users in the Userpool

General settings if the Userpool used is shown below:

Pool Id	us-east-1_ML8n8zEda	
Pool ARN	arn:aws:cognito-idp:us-east-1:647399968670:userpool/us-east-1_ML8n8zEda	
Estimated number of users	36	Edit
Required attributes	email	Edit
Alias attributes	none	
Username attributes	none	
Enable case insensitivity?	Yes	
Custom attributes	Choose custom attributes...	
Minimum password length	8	Edit
Password policy	uppercase letters, lowercase letters, special characters, numbers	
User sign ups allowed?	Users can sign themselves up	
FROM email address	Default	Edit
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...	Edit
Verifications	Email	
Advanced security	Enable advanced security...	Edit
Tags	Choose tags for your user pool	Edit
App clients	ebcognitoclient	Edit
Triggers	Add triggers...	Edit

Figure 4: General Settings of Userpool

- **AWS DynamoDB**

Task and Subtask are two main tables required in the project to store the respective data. As the information user enter is can be null or incomplete, it is decided to make use of unstructured No-SQL database like DynamoDB rather than using structured databases like RDS. Task table consist of Owner field which is a sort-key with which tasks are identified for each of the user from Cognito Userpool. Subtask table consists of ParentTask as a sort-key to differentiate the subtasks for each main task.

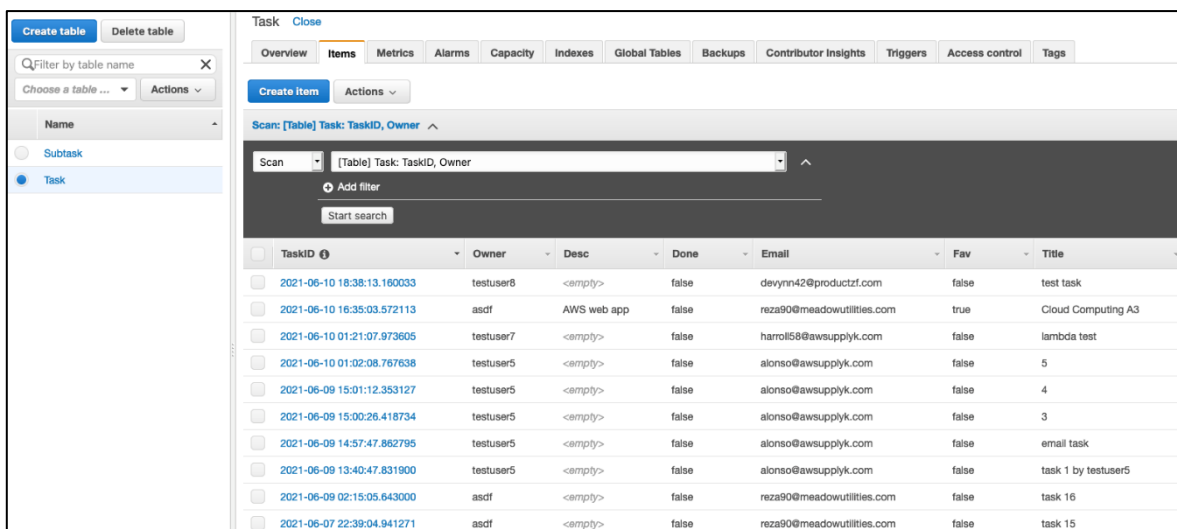
Boto3 is used to manage the tables.

Below code is from application.py

```
dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
tasktable = dynamodb.Table('Task')
subtable = dynamodb.Table('Subtask')
```

For the Task Table Partition key is TaskID and sort key is Owner. Other Fields of the Task table are Task Title, Task Description, Email ID of Owner to send the alert, if that task is marked as Favourite, and If that task is Done .

Some random entries in the Task Table:



The screenshot shows the AWS DynamoDB console interface. On the left, there's a sidebar with 'Create table' and 'Delete table' buttons, a search filter, and a list of tables including 'Subtask' and 'Task'. The main panel shows the 'Task' table details, including its partition key 'TaskID' and sort key 'Owner'. Below this, a table of sample entries is displayed with columns: TaskID, Owner, Desc, Done, Email, Fav, and Title.

TaskID	Owner	Desc	Done	Email	Fav	Title
2021-06-10 18:38:13.160033	testuser8	<empty>	false	devym42@productzf.com	false	test task
2021-06-10 16:35:03.572113	asdf	AWS web app	false	reza90@meadowutilities.com	true	Cloud Computing A3
2021-06-10 01:21:07.973605	testuser7	<empty>	false	harrol58@awsupplyk.com	false	lambda test
2021-06-10 01:02:08.767638	testuser5	<empty>	false	alonso@awsupplyk.com	false	5
2021-06-09 15:01:12.353127	testuser5	<empty>	false	alonso@awsupplyk.com	false	4
2021-06-09 15:00:26.418734	testuser5	<empty>	false	alonso@awsupplyk.com	false	3
2021-06-09 14:57:47.862795	testuser5	<empty>	false	alonso@awsupplyk.com	false	email task
2021-06-09 13:40:47.831900	testuser5	<empty>	false	alonso@awsupplyk.com	false	task 1 by testuser5
2021-06-09 02:15:05.643000	asdf	<empty>	false	reza90@meadowutilities.com	false	task 16
2021-06-07 22:39:04.941271	asdf	<empty>	false	reza90@meadowutilities.com	false	task 15

Figure 5: Task Table structure and sample entries

For the Subtask table Partition key is SubtaskID and sort key is ParentTask. Other Fields in the Subtask table are Subtask Title, Subtask Description, Due Date of a task, If that subtask is Done, Image if required, and URL if required.

Some sample entries in the Subtask table are:

The screenshot shows the AWS Glue console interface. On the left, there's a sidebar with 'Create table' and 'Delete table' buttons, and a search bar. The main area displays the 'Subtask' table details. The table has columns: SubtaskID, ParentTask, Desc, Done, Due, Image, Title, and Url. Below the table, there are sample entries with their respective values.

SubtaskID	ParentTask	Desc	Done	Due	Image	Title	Url
2021-06-10 18:39:09.539935	2021-06-10 18:38:13.160033	<empty>	false	2021-06-11	<empty>	test subtask	<empty>
2021-06-10 16:37:26.065406	2021-06-10 16:35:03.572113	<empty>	false	2021-06-10	<empty>	Take screenshots of AWS consoles	<empty>
2021-06-10 01:21:40.191697	2021-06-10 01:21:07.973605	<empty>	false	2021-06-10	<empty>	test 3	<empty>
2021-06-10 01:21:29.044354	2021-06-10 01:21:07.973605	<empty>	false	<empty>	<empty>	test 2	<empty>
2021-06-10 01:21:21.158442	2021-06-10 01:21:07.973605	<empty>	false	2021-06-10	<empty>	test1	<empty>
2021-06-09 13:43:04.595466	2021-06-09 13:40:47.831900	<empty>	false	2021-06-11	<empty>	3 asdfasdzxcv	<empty>
2021-06-09 13:42:55.081034	2021-06-09 13:40:47.831900	<empty>	false	<empty>	<empty>	2 sdafawefasdcx	<empty>
2021-06-09 13:41:22.522041	2021-06-09 13:40:47.831900	<empty>	false	2021-06-11	<empty>	1 asdfasfasdfasdfasdfas	<empty>
2021-06-09 04:11:56.875261	2021-06-06 17:49:09.020202	<empty>	false	<empty>	<empty>	5-4	<empty>
2021-06-09 04:11:33.511889	2021-06-06 17:49:09.020202	<empty>	false	<empty>	<empty>	5-3	<empty>

Figure 6: Subtask table structure and sample entries

• AWS Simple Storage Service S3

The main purpose of using S3 buckets in the project is to store the images related to the Tasks and Subtasks and profile images of users which are shown in the application at a corresponding User profile and Task description. Two separate S3 buckets are created to store profile images and subtask related images. Boto3 is used for managing the storage and retrieval of these images as per the requirement.

Below code is from application.py

```
s3client = boto3.client('s3', region_name='us-east-1')
```

```
bucketname = 'cca3imgs'
```

```
bucketnameuser = 'cca3users'
```

The first S3 bucket is **cca3imgs** which is used to store images of subtasks.

The screenshot shows the AWS S3 console interface for the 'cca3imgs' bucket. The 'Objects' tab is selected, showing a list of objects. The objects are listed with their names, types, last modified dates, sizes, and storage classes.

Name	Type	Last modified	Size	Storage class
2021-06-06 04:43:56.950265.png	png	June 6, 2021, 04:43:58 (UTC+10:00)	4.2 KB	Standard
2021-06-07 21:41:07.713170.png	png	June 7, 2021, 21:43:55 (UTC+10:00)	3.7 KB	Standard
2021-06-07 21:47:47.347208.jpg	jpg	June 7, 2021, 21:47:49 (UTC+10:00)	5.3 KB	Standard

Figure 7: S3 bucket for Task Images: cca3imgs

The second S3 bucket is **cca3users** which is used to store user profile images.

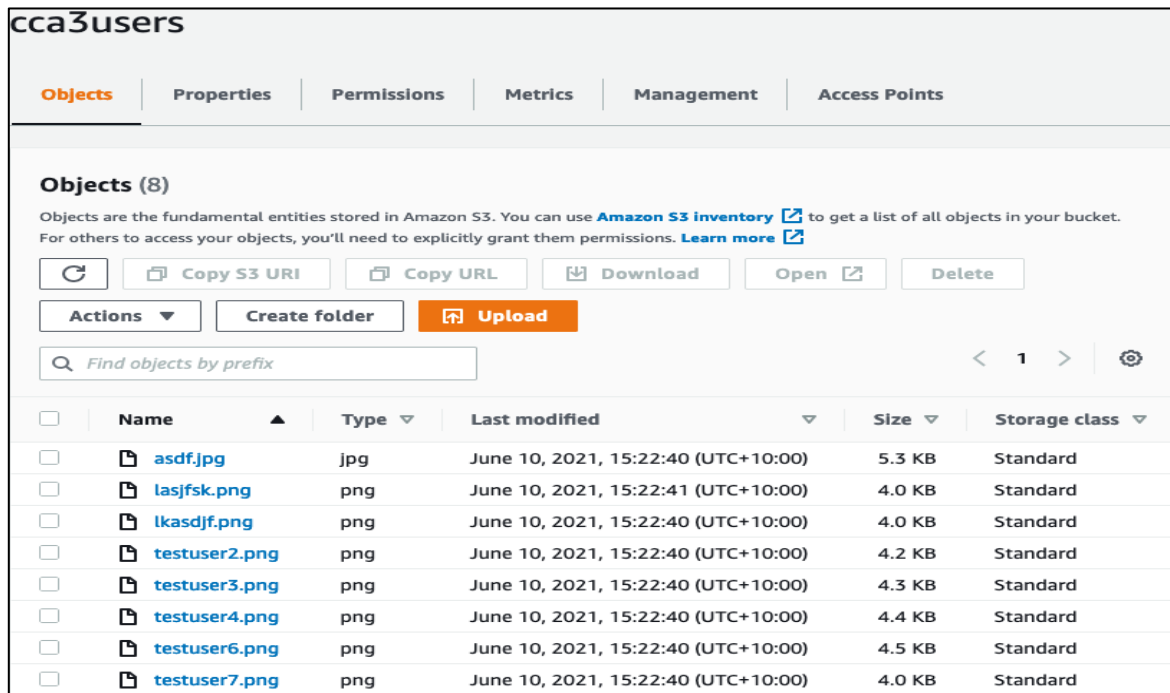


Figure 8: S3 bucket to store User profile images: cca3users

- **AWS Lambda**

AWS Lambda provides serverless computation and scalable functionality.

In the List-It application, a Lambda function is created to trigger an AWS SES (AWS Simple Email Service) service to send emails to users with tasks that are due next day. Due to this functionality, users will get an email notification of the tasks and corresponding subtasks which are due next day and not done. This requires a special IAM policy to be created and attached to add permissions to access Simple Email Service of AWS.

How to send emails using SES on Lambda [12] :

1. Create and attach IAM policy to IAM role
2. Create Lambda function with IAM role
3. Write code, save, and deploy

Below is the screenshot of the Lambda Function Overview:

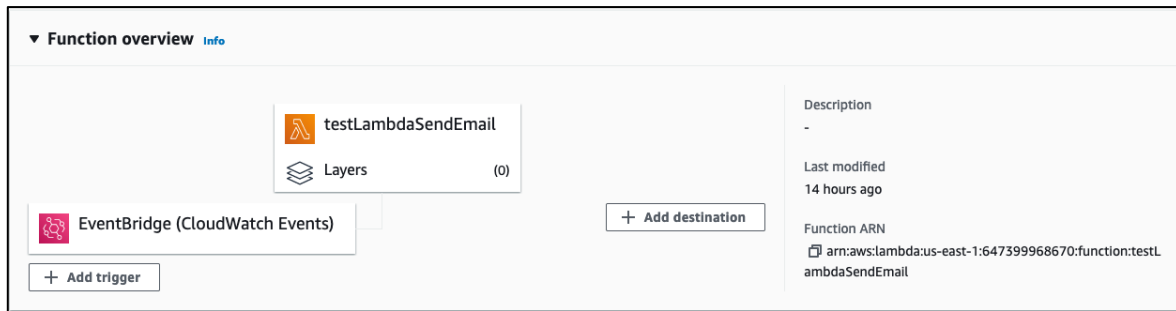


Figure 9: AWS Lambda function overview

• AWS CloudWatch

AWS CloudWatch is a monitoring and observability service which can help developers to set events at different points in the application. In the List-It application, AWS CloudWatch is used to schedule an event to invoke the lambda function once a day. The lambda function will check the due date of tasks and will trigger an email to the owners of the tasks if the due date of a particular subtask is due next day. The event timing is set to 8.30am each day. Below is the screenshot of the event scheduled via CloudWatch:

How to schedule a Lambda function using CloudWatch Event:

1. Create rule
2. Event selector: schedule at fixed rate of 1 day
3. Add target: Lambda function, select your Lambda function

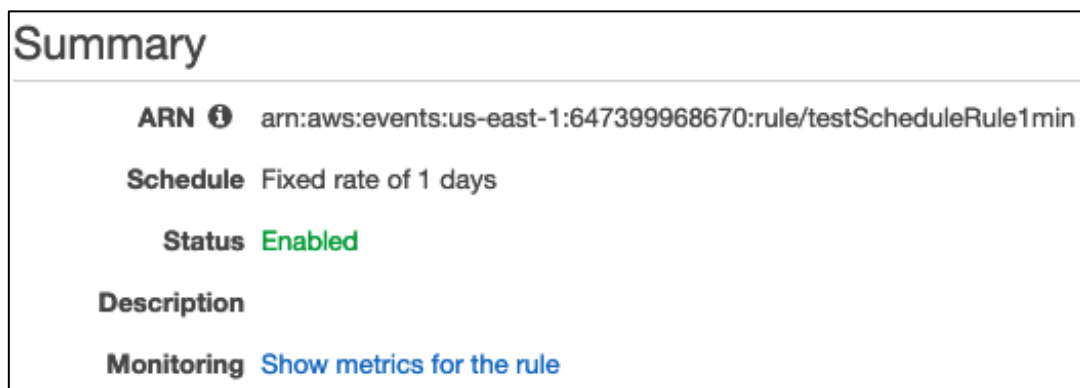


Figure 10: AWS CloudWatch event overview

• AWS Simple Email Service

AWS needs to verify the sender email address to send emails. As the AWS account is in the form of sandbox account, we had to request for production access so we can send emails to users. After getting the production access, we have created a Lambda function which can trigger the SES service to send emails to the registered users who have tasks which are due next day in their account. The email addresses which are used for registration are used to send the email.

The senders email address to send the emails to the users is verified and mentioned below:

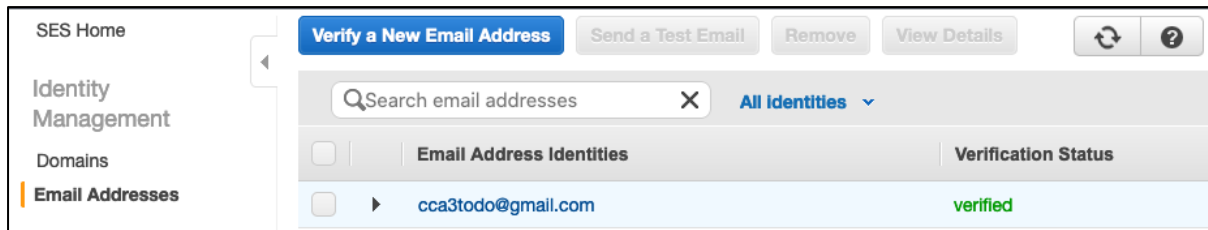


Figure 11: AWS SES sender email verification.

• AWS Kinesis Data Stream and Data Firehose

AWS Kinesis Data Streams and Firehose services are generally used for the analytics and research purpose. Although the application is not made for analytics purpose, these services are used in the application development to create a logs of item level changes made in DynamoDB tables Task and Subtask. Whenever, a record is inserted, updated, or deleted in those tables, a log of the changes is streamed to the S3 buckets in real time with the timestamp associated with it.

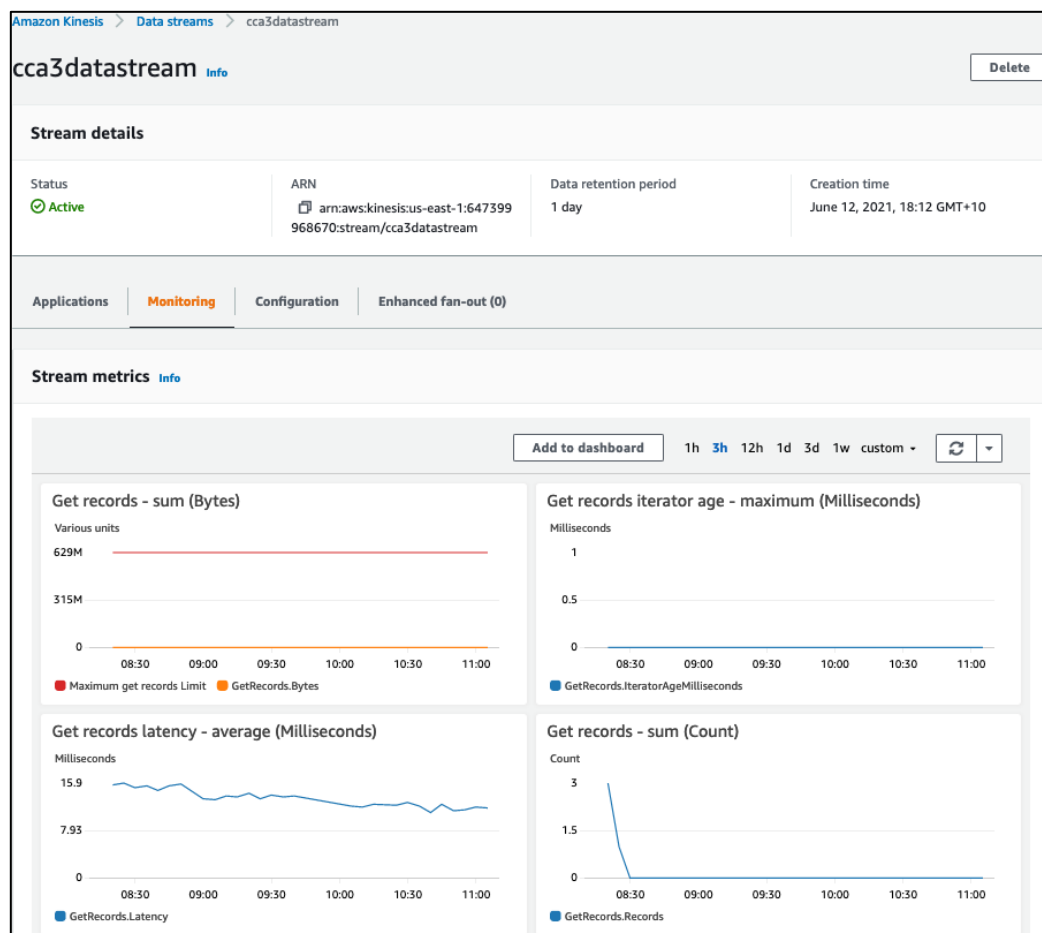


Figure 12:Kinesis Data Stream



Figure 13:Kinesis Delivery Stream

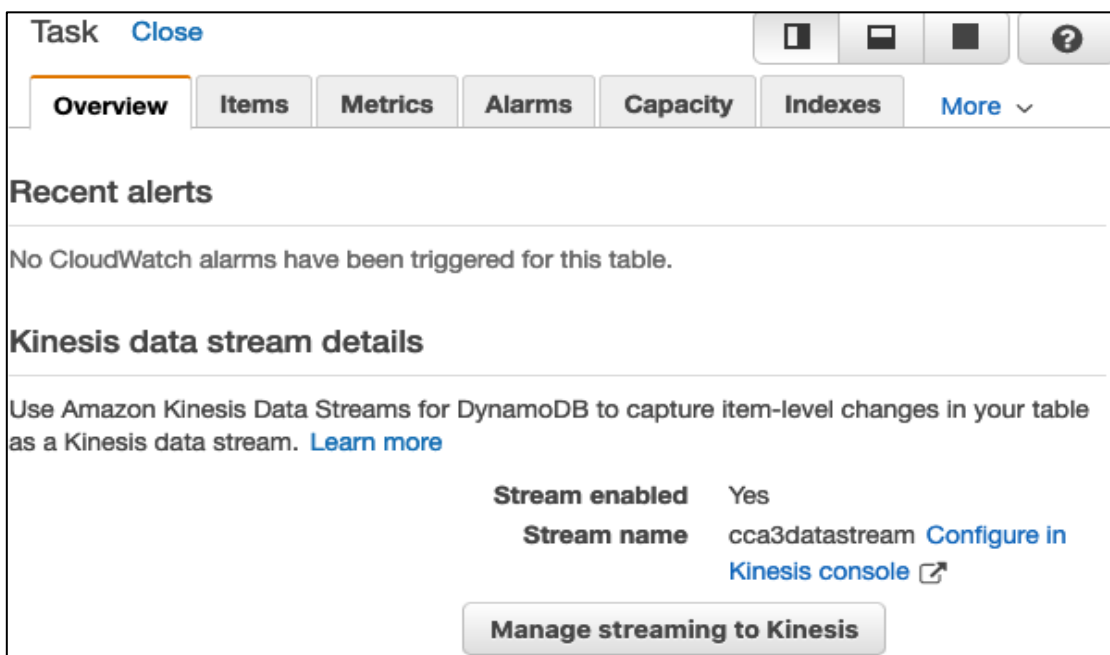


Figure 14:Task table Data Stream enabled

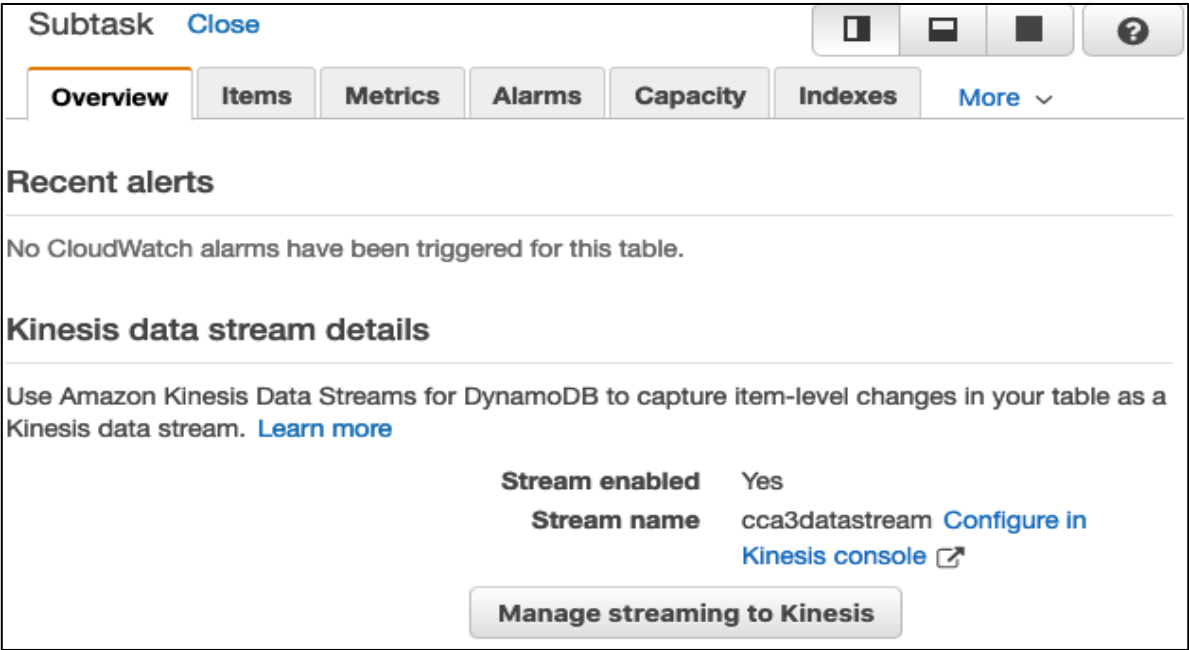


Figure 15: Subtask table Data Stream enabled

• AWS IAM

To use all the above-mentioned services, AWS users’ needs to have a specific level of access to those service. For this purpose, IAM roles are created to access those AWS services more securely. Below is the image of some of the IAM roles used to access the AWS services.

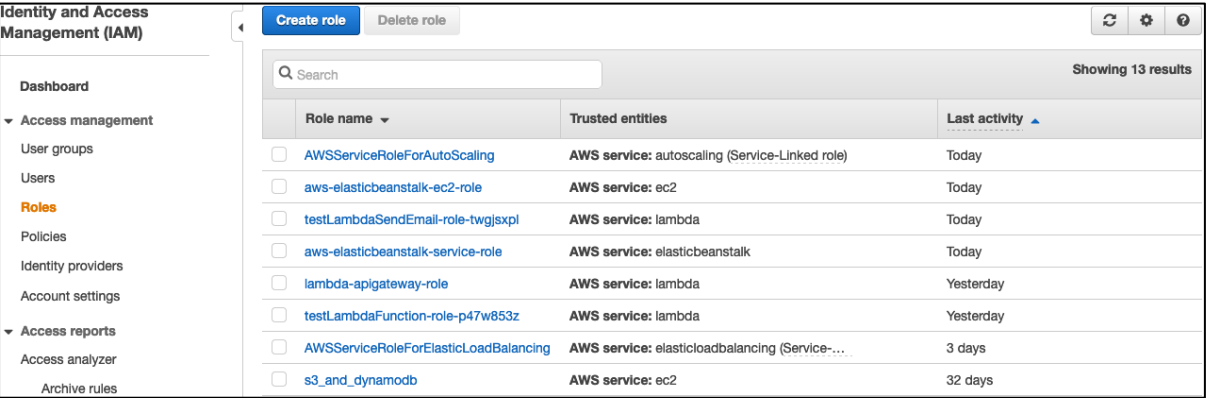
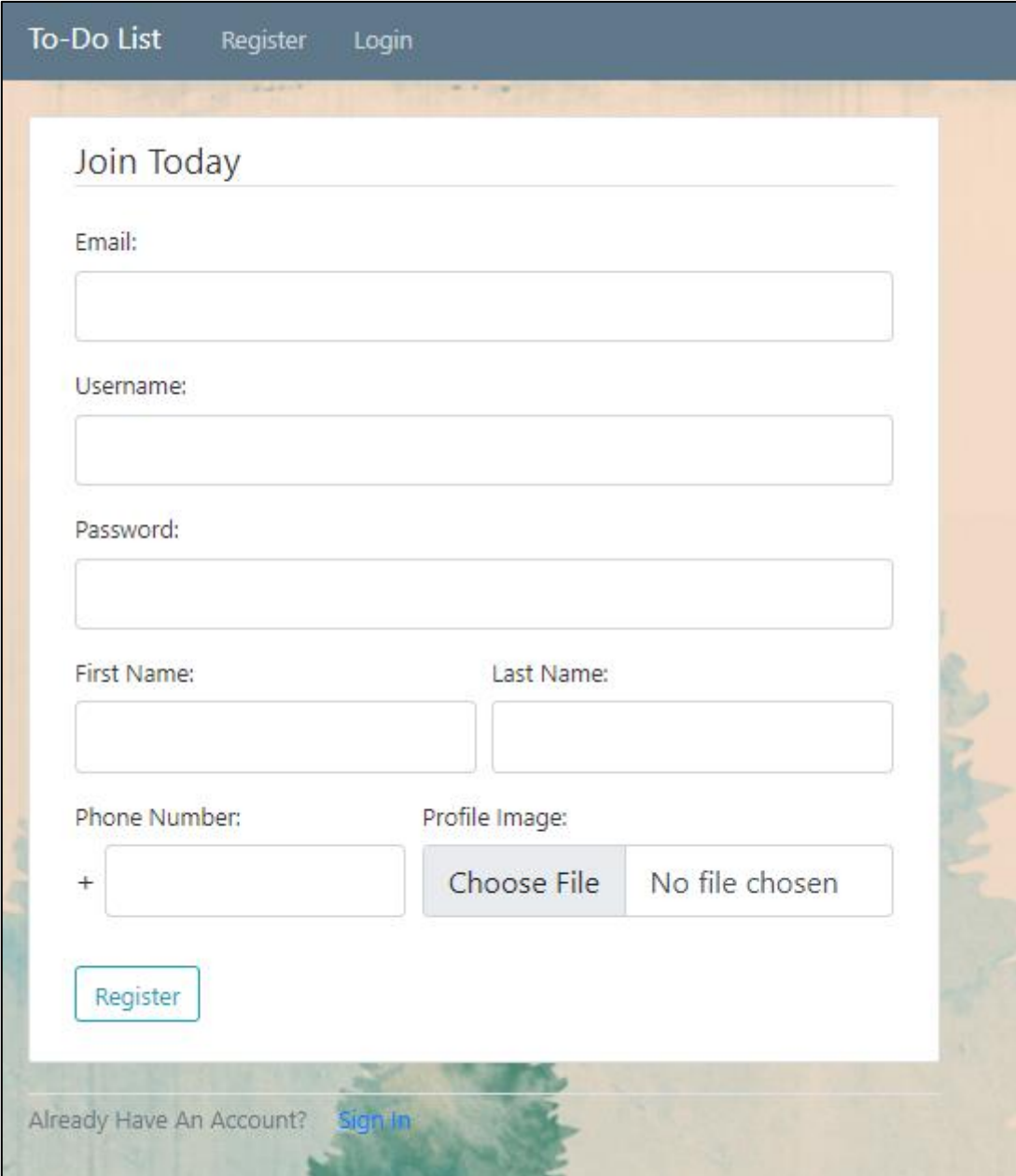


Figure 16: IAM rules example

User Manual

- **Register**

To be able to use the application, User need to register to the application by clicking on the 'Register' button on the top menu bar. For registration, all the fields are required to be filled out including the profile image.

A screenshot of a web application's registration page. At the top, a dark blue navigation bar contains the links 'To-Do List', 'Register', and 'Login'. The main content area has a light beige background with a subtle tree pattern. A white registration form is centered, titled 'Join Today'. It contains input fields for 'Email:', 'Username:', 'Password:', 'First Name:', 'Last Name:', and 'Phone Number:'. The 'Phone Number' field is preceded by a '+' sign. To the right of the phone number is a 'Profile Image:' section with a 'Choose File' button and the text 'No file chosen'. A blue 'Register' button is at the bottom left of the form. Below the form, the text 'Already Have An Account?' is followed by a blue 'Sign In' link.

To-Do List Register Login

Join Today

Email:

Username:

Password:

First Name: Last Name:

Phone Number: Profile Image:

+ Choose File No file chosen

Register

Already Have An Account? [Sign In](#)

Figure 17:Registration Page

- **Account Verification**

Following the registration, an email will be sent to users email address with a 6-digit security code to verify your account. To activate the account user must have to verify his email-id with the application. Once you input this code, your account will be activated, and you will be redirected to the login page.

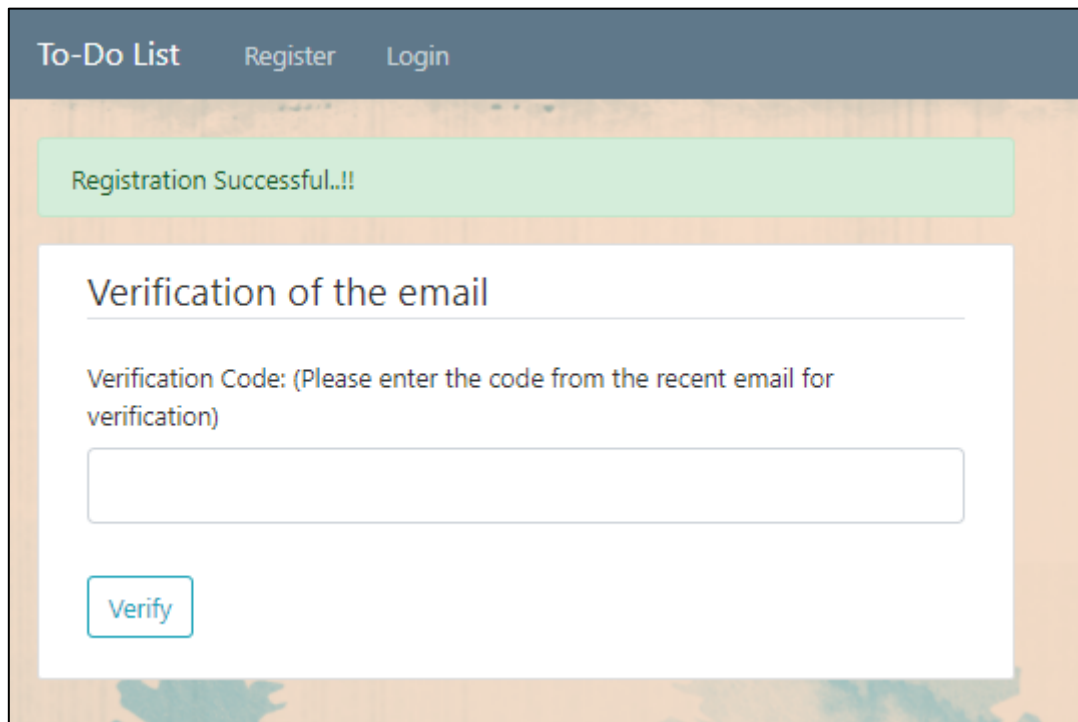


Figure 18: User Email Verification

- **Login**

After registration, user must have to login into a website to access the features and functionalities of the website. To log in, input your username, and password and select the login button. If the user is not the registered user of the application, there is a small link below the login button which will redirect the user to the registration page. After successful login, user will be redirected toward the section where he can see his favourite tasks list.

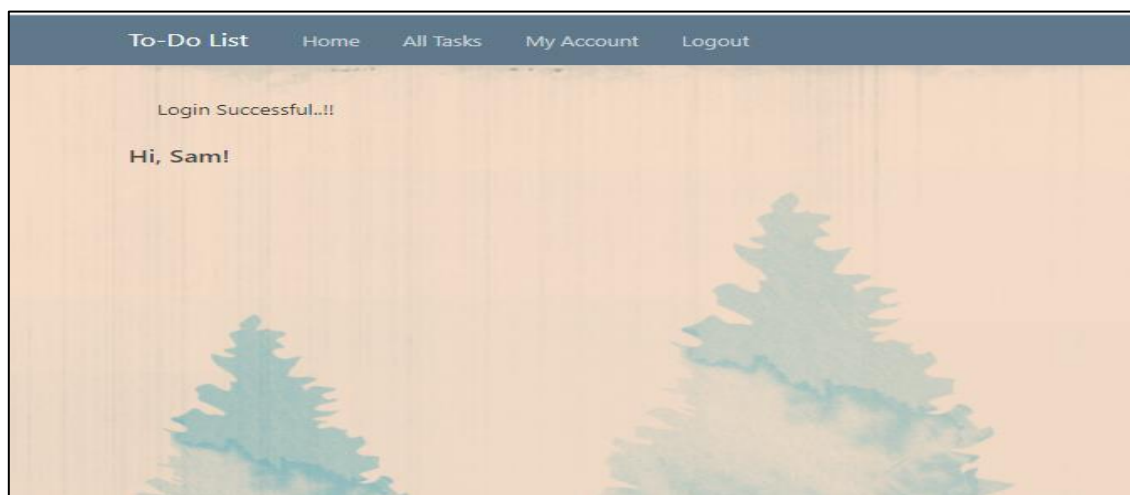


Figure 19: Successful Login of New User

- [Favourites/Home](#)

This is the default page user redirected to after the successful login. It is the first option on the navigation bar on the topside of the application layout. User can see the titles of the tasks which are marked as favourite by the user. To create a new task or mark the existing task as favourite, user need to click on the 'All Tasks' button on the top side of the navigation bar. If there are tasks in the Favourites tab, you can access them directly without going through a list of all tasks. Whenever you click on the particular task from the favourites page, the page will redirect you to the 'All Tasks' task list, showing the favoured task you clicked at the top of the screen so you can easily access it without having to find it in the long list of your tasks.

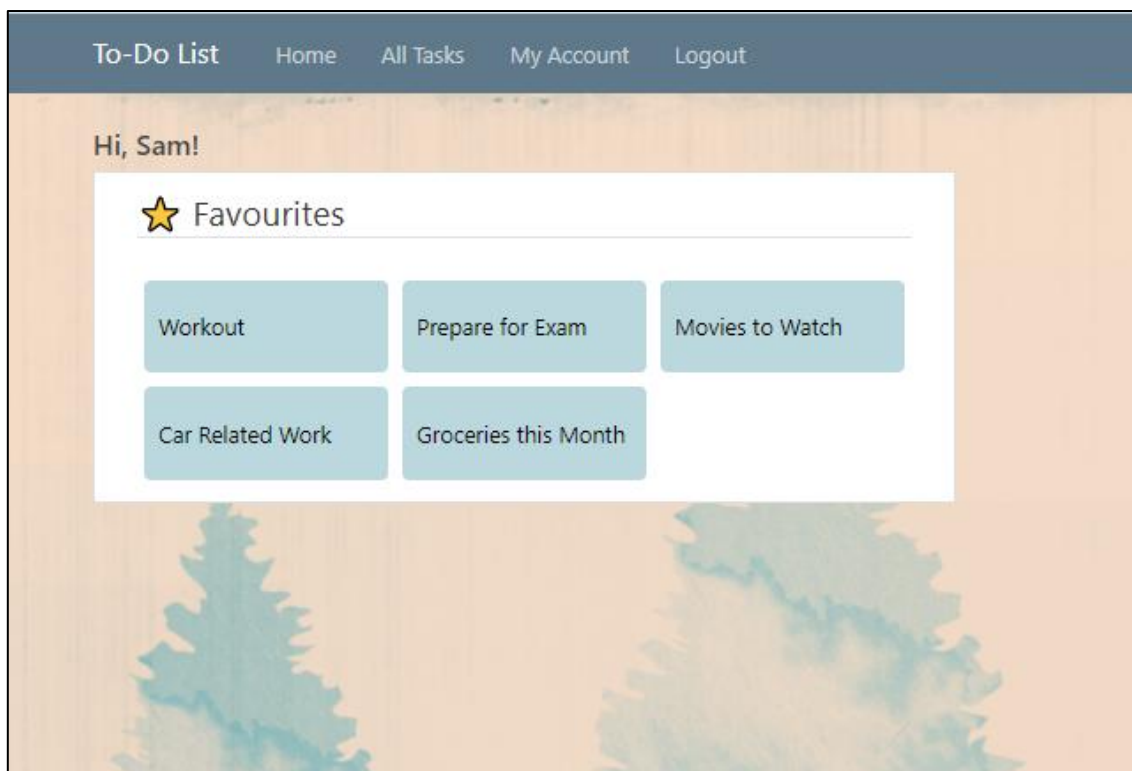


Figure 20: Favourite Tasks

- [All Tasks](#)

If the user is new and they don't have any tasks in their 'All Tasks' list, then a message of "There aren't any existing tasks at the moment. Please make a new task by clicking Add new task button." will show on their 'All tasks' page. User need to create a new task by clicking on the 'Add new task' button.

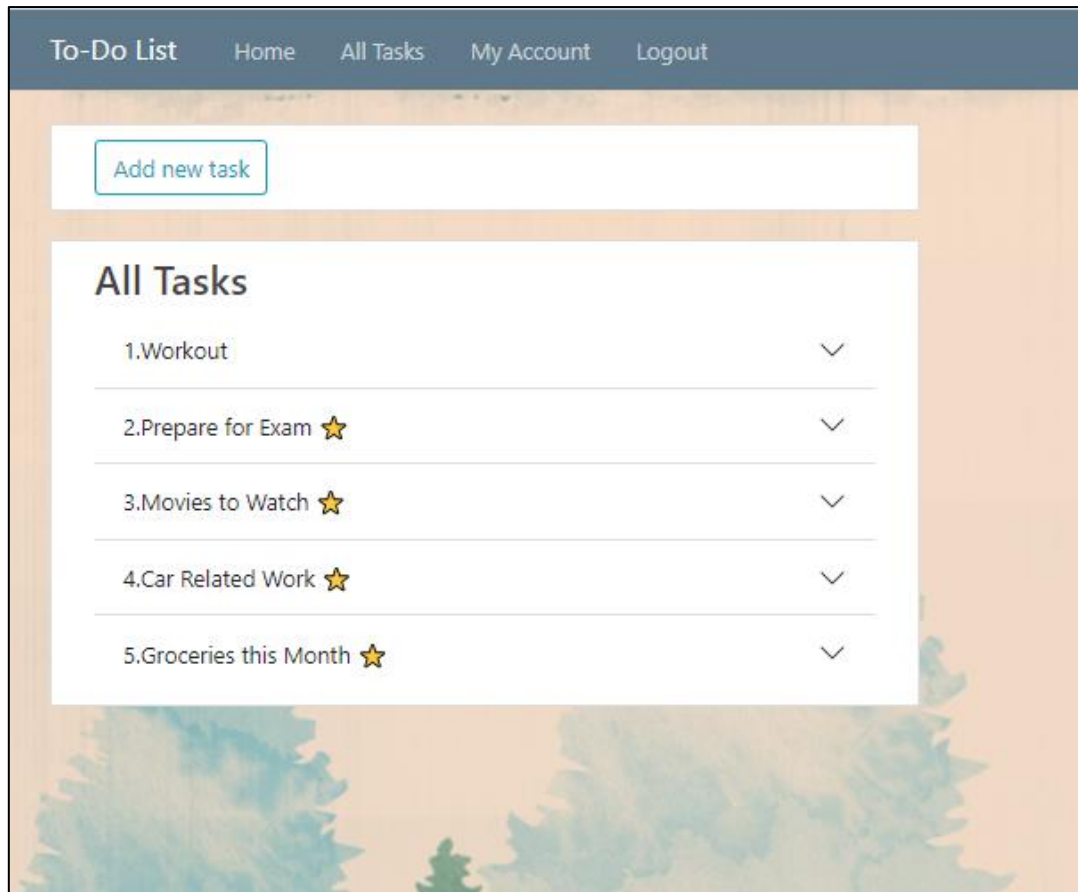


Figure 21: All Tasks List

- **Add New Task**

To add a new task, there is a button on the page of 'All Tasks' names 'Add new task'. After clicking this button, a form will open where user needs to enter some details about the task like, Title of a task, Description of a Task, If the task is marked as Completed and If the task is marked as favourite. Only the title is required to add a task. Once user fill the information and press the 'Add' button they can see the task in their 'All Tasks' list. If the user marked the task as favourite, it would start showing on both 'All Tasks' list as well as on 'favourites' page.

The screenshot shows a web application interface for a to-do list. At the top, there is a navigation bar with links: 'To-Do List', 'Home', 'All Tasks', 'My Account', and 'Logout'. Below this, a modal form titled 'Add new task' is displayed. The form contains the following elements:

- A button labeled 'Add new task' at the top left of the form.
- A text input field labeled 'Title:'.
- A text input field labeled 'Description:'.
- Two checkboxes: 'Completed: ☐' and 'Make it Favourite?: ☐- An 'Add' button at the bottom left of the form.

Figure 22: Add a new task

• Task Details

To see the tasks details, there will be an arrow button present in front of the Task Title on 'All Tasks' page. Click on that button which will drop down into a form containing details and some buttons regarding that task. If the task is marked as favourite, then there will be a ★ (star) symbol present in front of the task title. Task details include of the Task Title, Task Description, 3 buttons of 'Edit the task', 'Delete the task' and 'Add new Subtask' and a list of Subtasks. If the Task do not have any subtasks in it, the Subtasks list will be empty.

a. Editing the Task:

Users can edit the details of the task at any point of time by clicking on the Edit task button present in the drop down of that task. User can mark the task as completed there if it does not have any subtasks.

b. Deleting the Task:

Users can delete the task at any point of time by clicking on the 'delete this task' button present in drop down menu for each task. When user clicks on that button a popup for confirmation for deletion will appear on the screen. If user accepts the popup, the Task and it's any existing subtasks will get deleted permanently from the user's task list.

c. Adding new Subtask

Users can add new subtasks under the same task title multiple times by clicking on Add new subtask button. When user clicks on that button a new form will appear below where users can enter the subtask details which they wanted to enter. After clicking on add button, a new subtask will get added under the parent task. And user can see that task inside the task details in 'Subtask list' section. If a task was marked as complete before, its state will change to incomplete, removing the tick for the task.

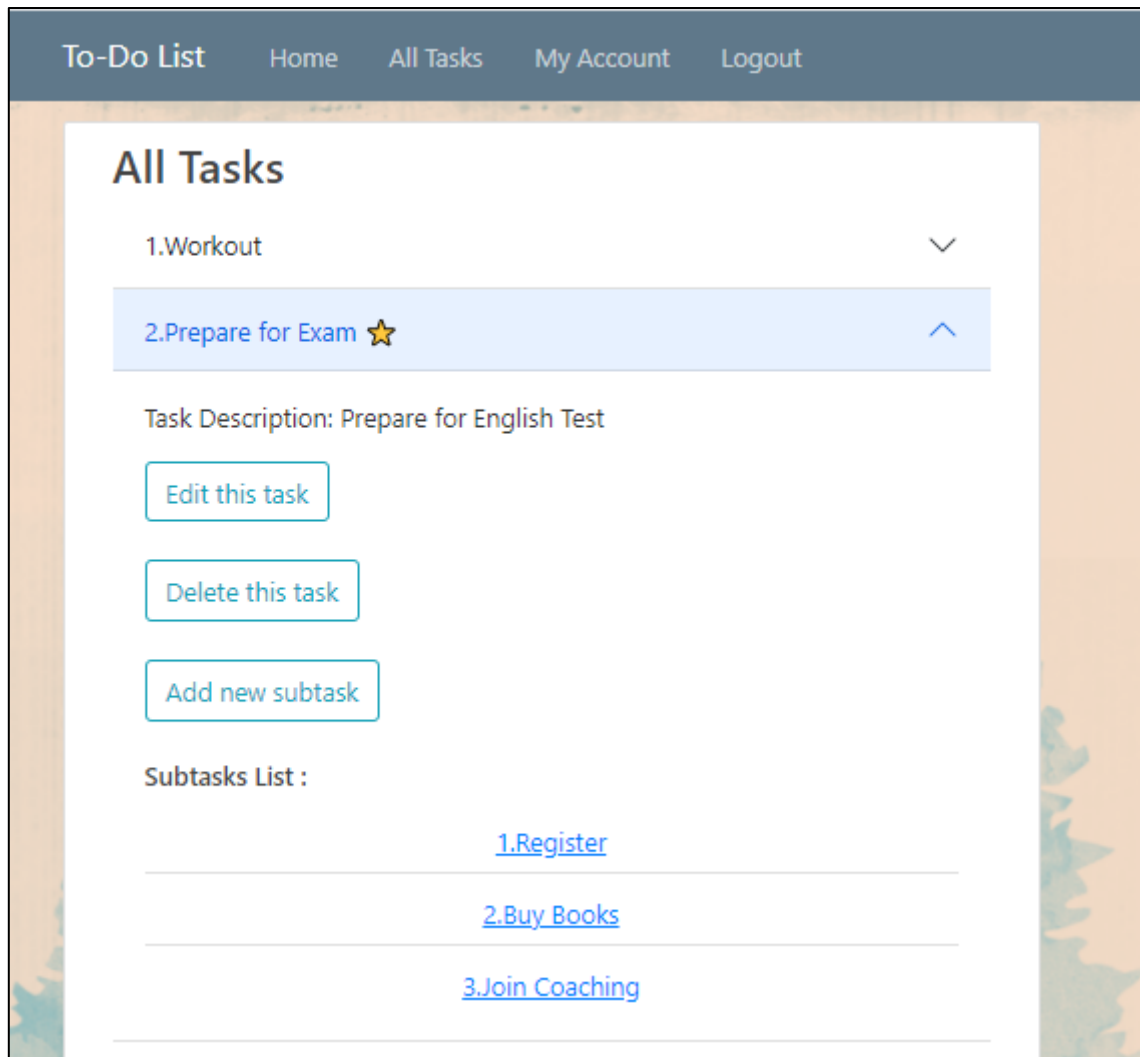


Figure 23: Task Details

• Subtask Details

Whenever user selects option to add subtask in a particular task, a form will appear below the 'Add new subtask' button. User can enter various details to create a subtask including, Subtask Title, Subtask Description, Due Date of a Task, Image if applicable, URL if applicable, if user wanted to mark that as completed while creating etc. Only the title is required to add a subtask. After filling all the details and clicking on the Add button, the subtask will appear on the subtask list section under the parent task.

Users can see the subtask details by clicking on the title of subtask. Subtasks also consists of the similar buttons to Edit and delete the subtask which works in a similar way as those for Tasks. If the subtask image does not update after editing, please do a hard refresh by clicking the refresh button of the browser while holding down the shift button. Users can see the URLs and Images they uploaded for each subtask. Once the subtask is deleted, it will permanently get deleted from the parent task.

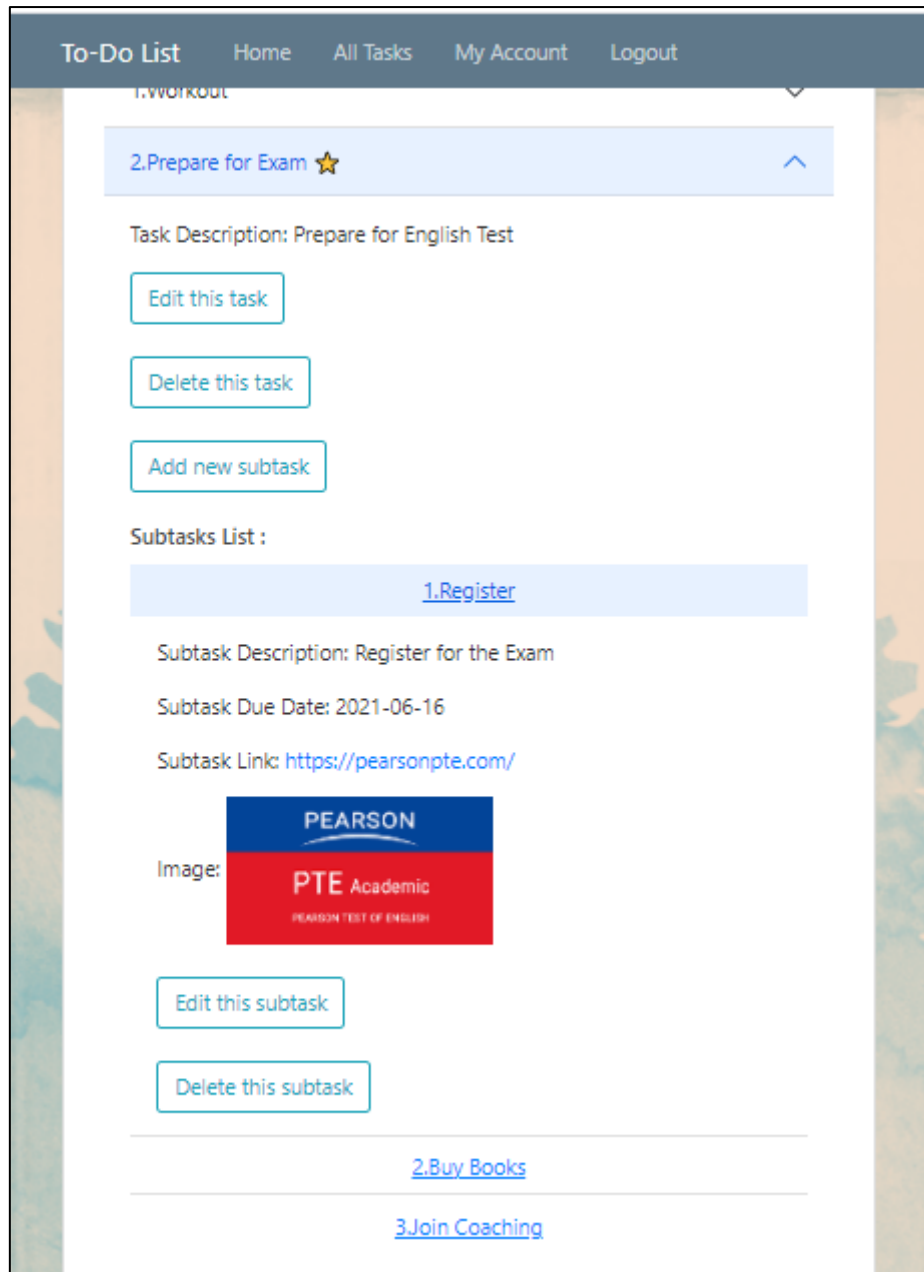


Figure 24: Subtask Details

- **Marking Tasks or Subtasks as Completed.**

Users can mark the subtasks as completed or done after the creation by clicking on the 'Edit this Subtask' button from the subtask which they wanted to mark as completed. After selecting the checkbox of Done, the sign of completion will appear near the Title of the subtask. If all the subtasks are marked as done or completed, the main task will automatically be marked as completed and a sign of completion will appear near the Task title on 'All Tasks' page.

5.Groceries this Month ☆

Task Description: Groceries List

Edit this task

Delete this task

Add new subtask


Subtasks List :

1.Rice ✓

Subtask Description: Rice 5KG

Subtask Due Date: 2021-06-22

Subtask Link:
<https://www.woolworths.com.au/shop/productdetails/75816>

Image:


Edit this subtask

Delete this subtask

Figure 25: Subtask Completion

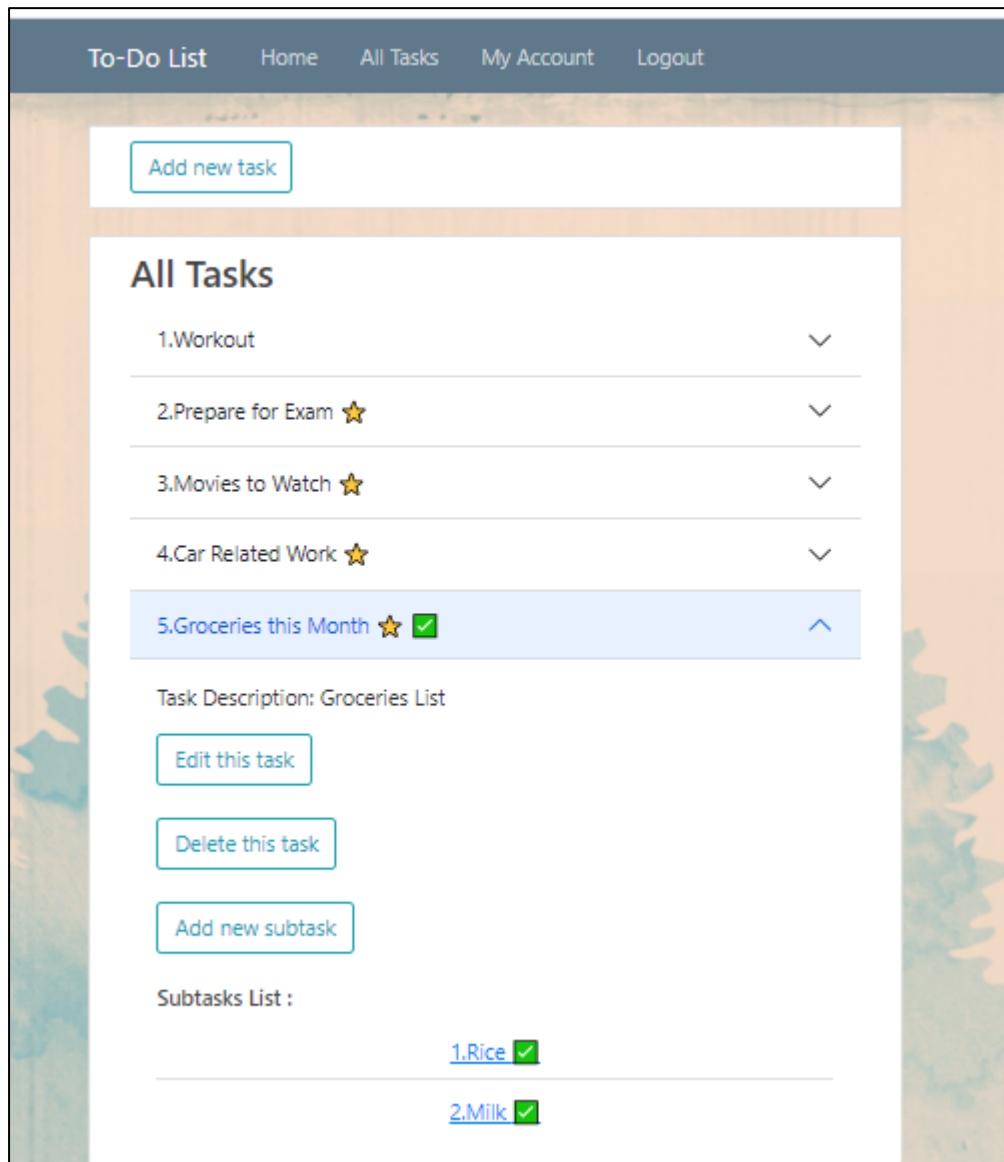


Figure 26: Task Completion

- **Emails of the Tasks which are due next day.**

Users will get an email consisting of the list of subtasks along with their respective parent tasks one day before the due date of the subtasks. The email will come from the official email address of the application "cca3todo@gmail.com" at around 8:30 in the morning.

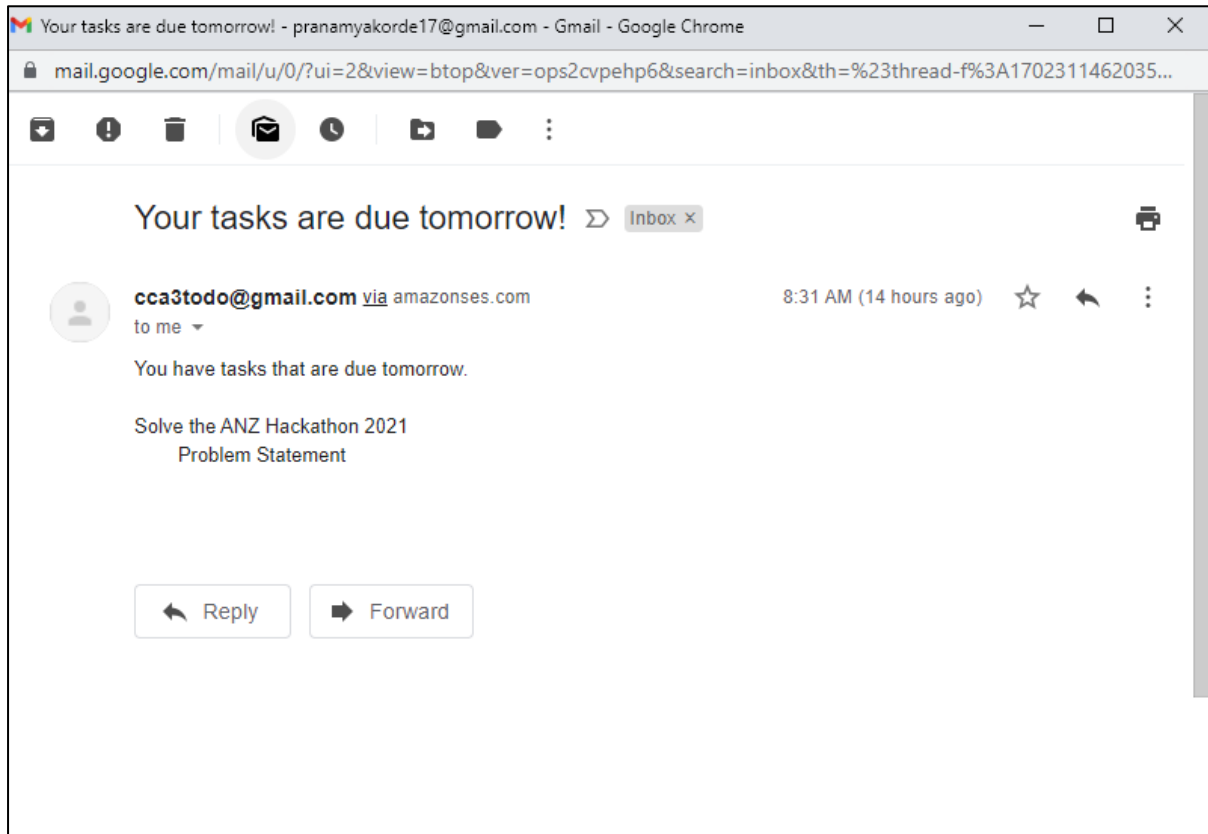


Figure 27: Email Alert Notification

- **User Profile**

Users can see and change their profile details on the My Account tab from the navigation bar which is at the top of the application layout. User can see and edit different details like First Name, Last Name, Phone Number and User profile image. If the profile image does not update after changing, please do a hard refresh by clicking the refresh button of the browser while holding down the shift button.

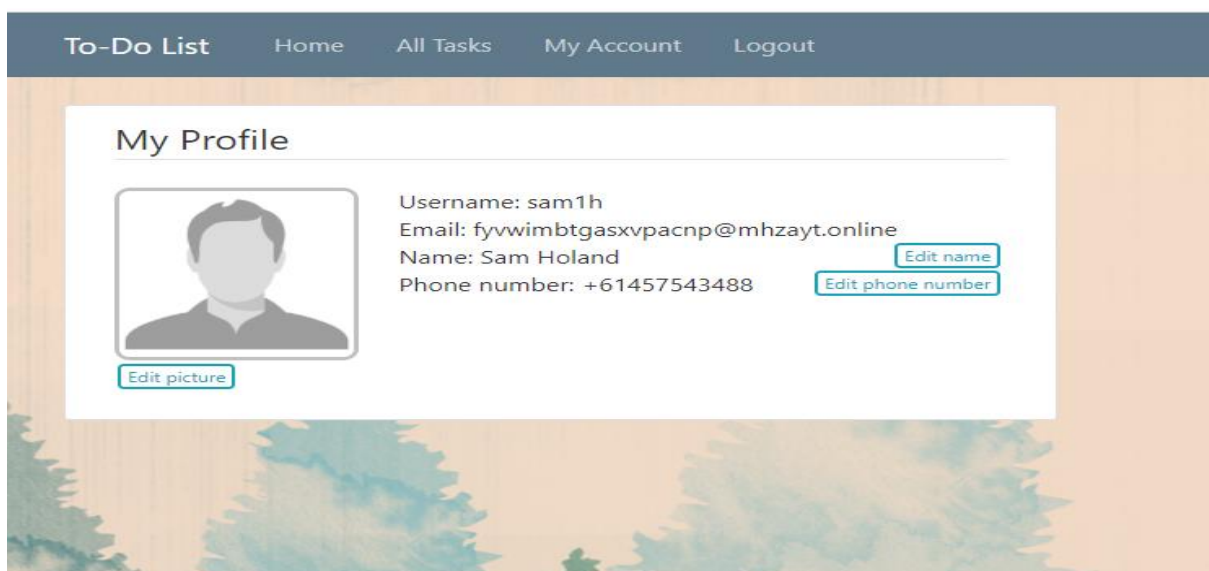


Figure 28: User Profile

References

- [1] aws.amazon.com, "aws.amazon.com," Amazon.com, [Online]. Available: <https://aws.amazon.com/what-is-cloud-computing/>. [Accessed 11 06 2021].
- [2] S. Ranger, "ZD Net," 13 12 2018. [Online]. Available: <https://www.zdnet.com/article/what-is-cloud-computing-everything-you-need-to-know-about-the-cloud/>. [Accessed 11 06 2021].
- [3] todoist, "https://todoist.com/," todoist, [Online]. Available: <https://todoist.com/features>. [Accessed 08 06 2021].
- [4] aws.amazon.com, "https://aws.amazon.com/elasticbeanstalk/," Amazon.com, [Online]. Available: <https://aws.amazon.com/elasticbeanstalk/>. [Accessed 07 06 2021].
- [5] aws.amazon.com, "https://aws.amazon.com/s3/," amazon.com, [Online]. Available: <https://aws.amazon.com/s3/>. [Accessed 05 06 2021].
- [6] aws.amazon.com, "https://aws.amazon.com/ses/," Amazon.com, [Online]. Available: <https://aws.amazon.com/ses/>. [Accessed 07 06 2021].
- [7] aws.amazon.com, "https://aws.amazon.com/lambda/," Amazon.com, [Online]. Available: <https://aws.amazon.com/lambda/>. [Accessed 08 06 2021].
- [8] aws.amazon.com, "https://aws.amazon.com/cloudwatch/," Amazon.com, [Online]. Available: <https://aws.amazon.com/cloudwatch/>. [Accessed 09 06 2021].
- [9] aws.amazon.com, "https://aws.amazon.com/iam/," Amazon.com, [Online]. Available: <https://aws.amazon.com/iam/>. [Accessed 10 06 2021].
- [10] aws.amazon.com, "https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-flask.html," Amazon.com, [Online]. Available: <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-flask.html>. [Accessed 12 06 2021].
- [11] aws.amazon.com, "https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-pool-as-user-directory.html," Amazon.com, [Online]. Available: <https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-pool-as-user-directory.html>. [Accessed 11 06 2021].
- [12] aws.amazon.com, "https://aws.amazon.com/premiumsupport/knowledge-center/lambda-send-email-ses/," Amazon.com, [Online]. Available: <https://aws.amazon.com/premiumsupport/knowledge-center/lambda-send-email-ses/>. [Accessed 12 06 2021].