Documentation

Web Application Introduction

Welcome to the documentation of our Time Management web application.

Time Management is an app that enables you to enjoy many time management services in just one app. We all know that there are many efficiency apps in the market, but many of them are just focusing on one functionality, for example, Todo List, Tomato Timer, and White sound noise. Users have to download many apps and use them together if they want to use the compound functionalities. An application that can integrate these functionalities all together and provide users a better environment for work or study is needed. Thus, we release our brand new and powerful time-management web application. Whether you work from home or in the office, you see what your task schedule is, how you spent your time and how productive you were. We believe that in this special time, self-isolated people need this convenient tool to keep them effective at home to work or study!

This web application **enables** users to:

- **Registration and login:** Create a personal account and login. Log out if they want.
- **Todo List:** A logged-in user can operate on his Todo list. Create a task, finish a task or delete a task.
- **Pomodoro:** A logged-in user can set up a Pomodoro timer to start timing. A user can set the time he wants and focus on this period.
- **White noise sound:** A logged-in user can enjoy the white noise sound we provided on our website to help them focus during the Pomodoro time.
- **Personal report:** Our website will record your focusing time automatically. A user can view his personal report: the focusing time duration within a day. They can learn about his work or study habit by looking over his personal statics.
- **Rank report:** Our website will record all the users' focusing time automatically. A user can view the rank report produced by our website: the top-three users who have the longest using time a day and their total focusing time a day. A user can be encouraged to focus longer and improve his efficiency by looking over top-three users.

Agenda of the documentation

The structure of the documentation is basically in three parts: **Users guide**, **Developers reference** and **AWS cost model**.

- **Users guide:** instruction for users to use the web application with its functionality of Time Management.
- **Developers reference:** instruction for developers to understand the architecture of the application. In addition to the background process.
- **AWS cost model:** design a cost model of the team's application, predict the costs for 10, 1000 and 1000,000 users respectively.

Users guide

This guide will give the user a tutorial on how to use this application.

Login page

Open the browser and enter the URL of the homepage of the website: https://a3ye3991k1.execute-api.us-east-1.amazonaws.com/dev to access the login page which also is the homepage of the website. The login page contains a login block which consists of two inputs: Username and Password. Username and password are required to log in.

Login process

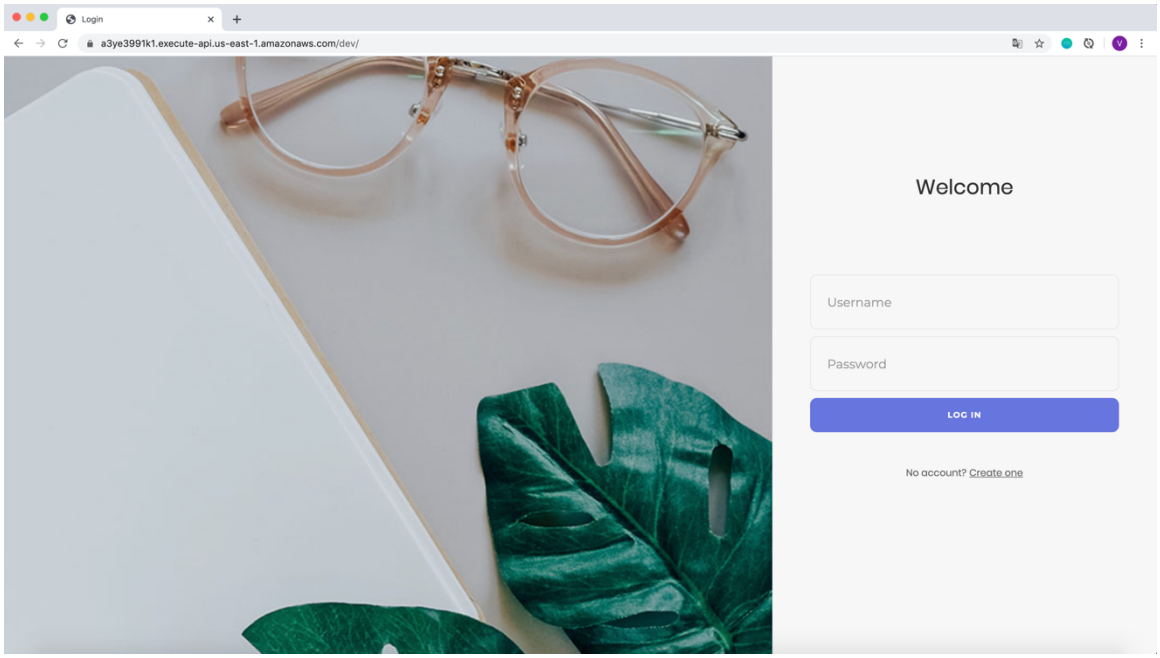
You should enter your registered username and correct password in the inputs of Username and Password and click on the **LOG IN** button to finish the login process and then you will jump to your personal page. If you do not have an account, please click on **Create one** to jump to the register page.

Error messages

- You will get the reminder of "Username is required" or "Password is required" if one of them is empty.
- You will get the reminder of "The username does not exist. Please try again" if the username you entered does not exist.
- You will get the reminder of "Password error" if you enter an incorrect password.

Success messages

- You will see the message of "Welcome back + your username" on the top of your personal page to show that you have successfully logged in.



Login Page

Register page

You can access to register page when you click on the **Create one** button in the Login page or directly visit it by entering the URL of https://a3ye3991k1.execute-api.us-east-1.amazonaws.com/dev/register/.Register page contains a register block that consists of three inputs: Username, Password and Confirm password. All the inputs are required.

Register process

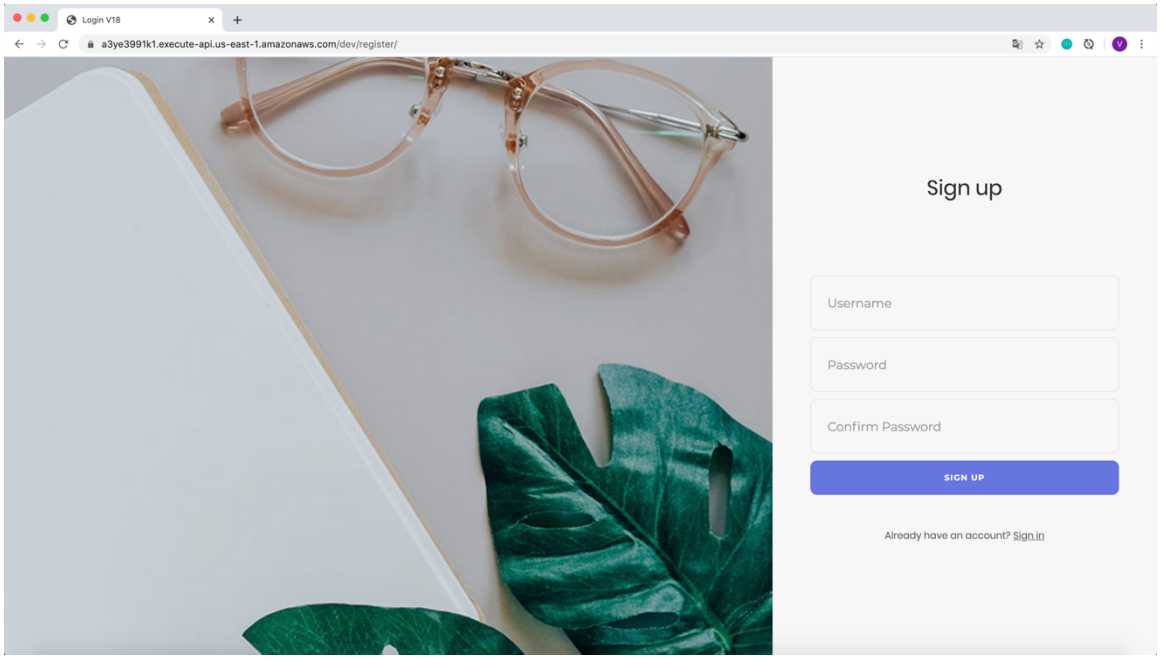
You should enter a legal username, password and confirmation password in three inputs and click on the **SIGN UP** button to finish the registration process and then you will jump to the Login page to login. If you already have an account, please click on **Sign in** to jump to the Login page.

Error messages

- You will get the reminder of “Username is required”, “Password is required” and “Please enter your password” if one of the inputs is empty.
- You will get a reminder of “The length of the username is illegal!” if your username is more than 100 characters.
- You will get a reminder of “The name already exists!” if your username is the same as someone else.
- You will get a reminder of “Two passwords are inconsistent!” if your original password and confirmation password are inconsistent.

Success messages

- You will get a reminder of “You have successfully registered!” if you have done the registration process correctly.



Register Page

Personal page

You can access to your personal page when you have successfully logged in. You will see the message of “Welcome back + your username” on the top of your personal page to show that you have successfully logged in.

The personal page consists of four parts: A **clock** showing the current time, **Todo List**, two buttons for **Personal report** and **Rank report** and **Log out** button.

Clock: You can see the current dynamic time on the top of your personal page.

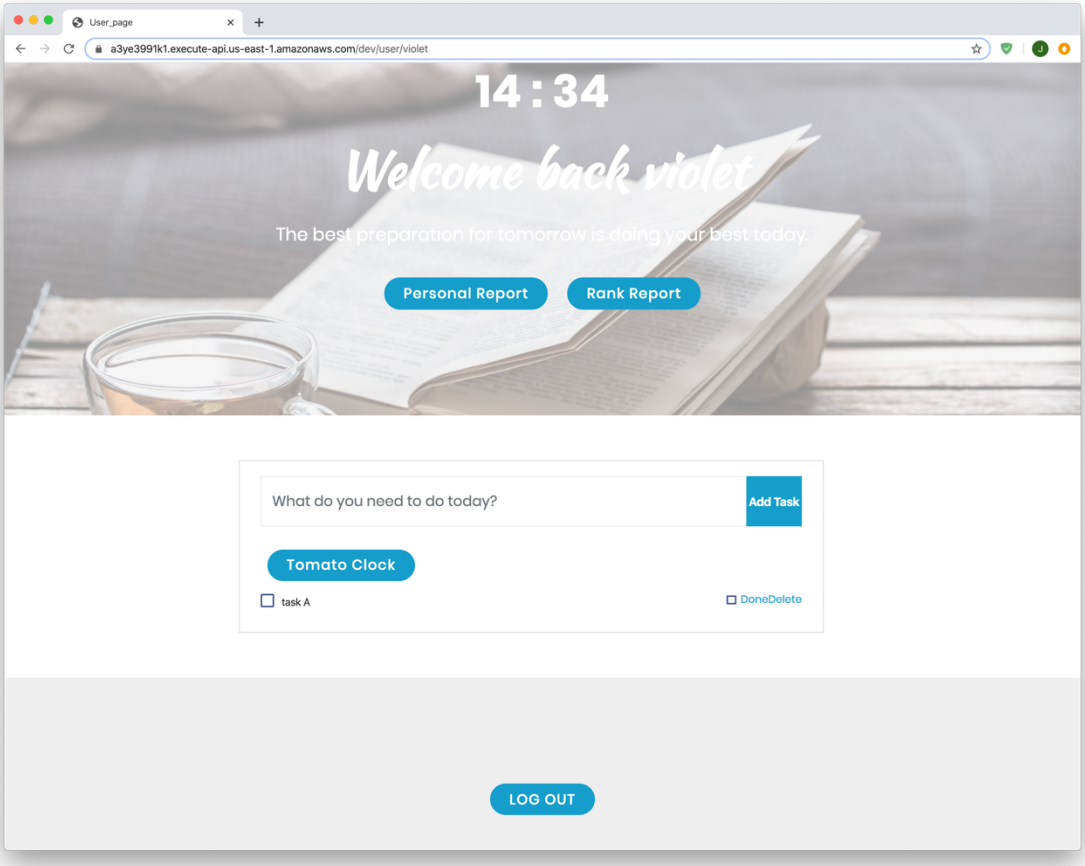
Todo List: You can use the Todo List to organize your tasks neatly. Todo List contains three parts:

- **An input box** which enables you to enter your task content with a submit button to add a task to your task list.
- **A task list** with the **Done** and **Delete** button. If you have finished the task, you can click on the Done button of this task and there will be a strikethrough on the task content to show that you have finished. If you want to delete the task, you can click on the Delete button of this task and this task will disappear in your task list.

- **Tomato Clock button:** If you click on the Tomato Clock button, you will jump to the Tomato Timer page.

Buttons for Personal report and Rank report: Click on the Personal report button, you will jump to the personal report page which enables you to view your personal report which is the focusing time duration within a day. Click on the Rank report button, you will jump to the rank report page which enables you to view the rank report which is the top-three users who has the longest using time a day and their statics.

Log out button: Click on the Log out button, you will directly log out your current account and back to the homepage of our website which is the Login page also.



Personal Page

Tomato Clock page

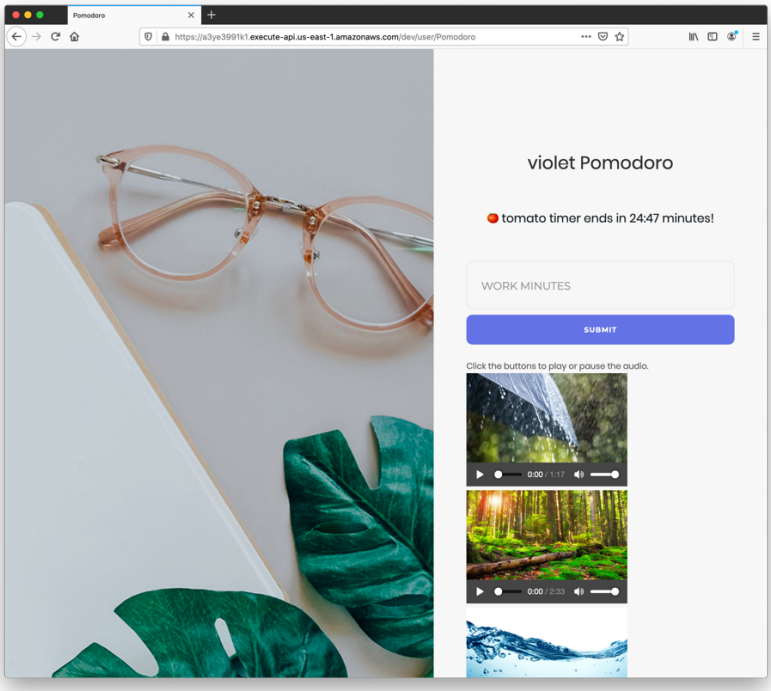
You can access to Tomato Clock page by click on the **Tomato Clock button** in Todolist block in your personal page. Tomato Clock page consists of three parts: **tomato clock**, **white sound noise** and **back button**.

Tomato Clock: You can use the Tomato Clock to organize your time neatly. Tomato Clock contains three parts:

- **An input box** which enables you to enter the time you want to focus in this tomato timer.
- **A dynamic counting down timer:** this dynamic timer counts down dynamically and you can see how much time left to finish this Tomato Timer.

White noise sound: Our website provides 3 different high-quality white noise audios: the sound of rainfall, the sound of forest and the sound of water. Each sound is composed of a picture to show the type of the sound and a player to help you to control the sound. You could choose one of them to keep you accompany when you are working or study with a Tomato Timer.

You can click on the **Back button** to go back to the personal page.



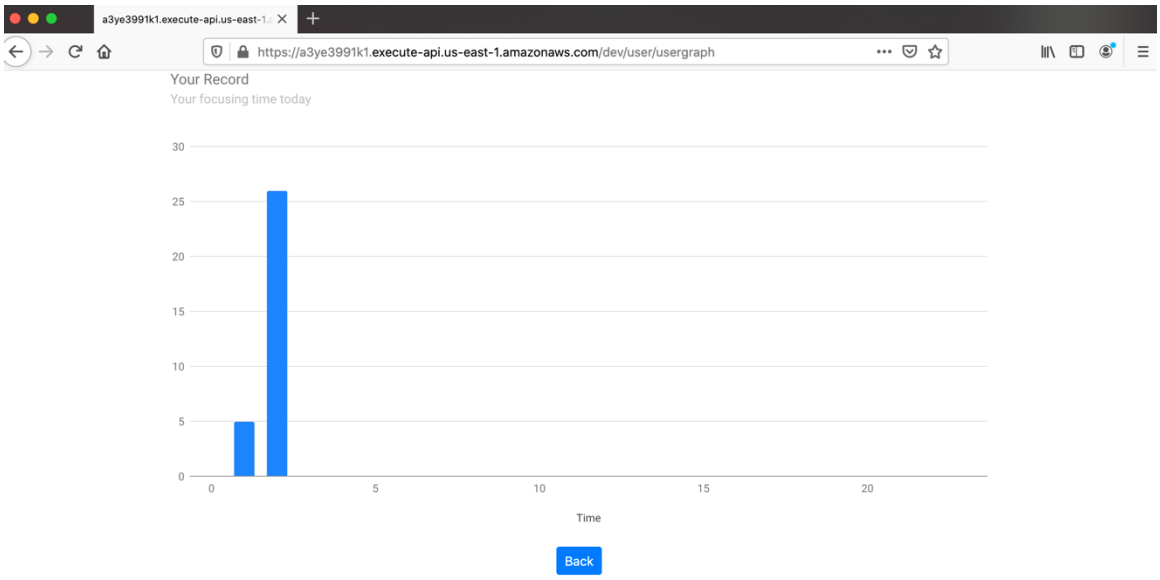
Tomato Clock page

Personal Report page

You can access to Personal report page by click on the **Personal Report button** on the personal page. You can see your focusing time distribution in 24 hours from the **bar chart** in the Personal report page.

Bar chart of personal report: the x axis of the bar chart is the 24 hours in the day and the y axis of the bar chart is the user's focusing duration in each interval. Our users can see their time distribution in each hour to observe their work or study habit like what time they always like to work or study during the day.

You can click on the **Back button** to go back to the personal page.



Personal Report page

Rank Report page

You can access to the **Rank Report page** by click on the **Rank Report button** in the personal page. You can see the top-three users who has the longest using time a day and their statics from the **bar chart** in the Rank report page.

Bar chart of personal report: the x axis of the bar chart is the username of the top-three users and y axis of the bar chart is the user's total focusing duration in the day. Our users can be encouraged to focus longer and improve his efficiency by looking over top-three users.

You can click on the **Back button** to go back to the personal page.

Rank Report page

Developer's Reference

Programming environment
Language: Python 3.7
Framework: Flask
IDE: Pycharm
Database: DynamoDB
Program structure:

```
ece1779_a3                                  # Directory for the whole project
├ templates                                          # Webpage file directory
│ ├ login.html
│ ├ register.html
│ ├ user_page.html
│ ├ user_data.html
│ ├ user_rank.html
├ venv                                       # Virtual environment directory
├ app.py                                                    # Main function
├ model.py                                          # Create class for app.py
├ create_data.py                    # Functions to create tables in DynamoDB
├ uploadfiletoS3.py               # Functions to upload static files to S3 bucket
├ zappa_settings.json                # Set up the correct environment for Zappa
```

We use Zappa to deploy our application.
Folder **templates** contains all the .html files which are used to build the web pages. The **app.py** is the function that implements most of the functionalities. Inside the file app.py, the main implementations are as following:

Main program

Open Module

flask	# a micro web framework based on the Werkzeug and Jinja2.
flask_wtf&wtforms	# Operator that reads the user's input from the web page
hashlib	# The module provides algorithms to implement the hashes for passwords

Customized Module

model	# The function that create class for app.py
create_data	# Functions to create tables in DynamoDB
uploadfiletoS3	# Functions to upload static files to S3 bucket

Customized Class

User( )	# Define a Class for users to support the table *Users* and operations of this table in the database

Customized Function

create_table1( )	# The function that create table *Users* in DynamoDB
create_table2( )	# The function that create table *Userdata* in DynamoDB
create_table3( )	# The function that create table *Userrank* in DynamoDB
insert_user_to_table( )	# The function that insert the user's account information to the table *Users* in DynamoDB
insert_task_to_db ( )	# The function that insert the user's tasks information to the table *Users* in DynamoDB
insert_userdata_to_db ( )	# The function that insert the user's tomato timer information to the table *Userdata* in DynamoDB

query_user_by_name ( )	# The function that query the user's information in the table *Users* in DynamoDB
query_max_taskid ( )	# The function that query the user's max task id in the table *Users* in DynamoDB
query_usertask_by_taskid ( )	# The function that query the user's task information(task content and if it is done) in the table *Users* in DynamoDB
delete_task_by_taskid( )	# The function that delete the all the information of user's tasks in the table *Users* in DynamoDB
update_done_by_taskid( )	# The function that update the done information of user's tasks in the table *Users* in DynamoDB
hash()	# The function that hash the password concatenated with a per-user salt value

Rooting Function

user_register( )	#The rooting function that renders the register page and operates the input from the register page
user_login( )	#The rooting function that renders the login page and operates the input from login page
user_visit( )	#The rooting function that renders the personal page and operates the input from login page
add_task( )	#The rooting function that delete tasks to the user's task list and renders the personal page
delete_task( )	#The rooting function that add tasks to the user's task list and renders the personal page
resolve_task( )	#The rooting function that update the status of the user's tasks and renders the personal page
draw_graph( )	#The rooting function that render the personal report page
collect_data( )	#The rooting function that render the rank report page

user_logout( )	#The rooting function that enables users to log out and back to the login page
add_pomodoro( )	#The rooting function that render the tomato timer page and operates the input from the register page

Templates

login	#Display the login page (the home page of the website)
register	#Display the register page
user_page	#Display the user personal page
user_data	#Display the personal report page with user's focusing durations
user_rank	#Display the rank report page with the top-three users
tomato	#Display the tomato timer page with the audio play list

Database

Database name: Dynamodb

Table name	Userdata	Userrank	Users
Description	Store the user's usage information	Store data for top 3 user	Store the user's information
Primary partition key	UserName (String)	Rank (Number)	UserName(String)
Primary sort key	StartTime (String)	-	TaskId (Number)

Background process

In addition to the main web functionality, our application also includes a separate process that runs in the background and does something useful for our application. This separate background process works directly with **Lambda functions**.

The functionality of the background process: one of the functionalities of our website to provide each user a rank report of the top-three users who have the longest using time a day and their total focusing time a day. We have to scan the table *userdata* which records

the users' start time and duration in a day and calculate their total duration in a day to produce the top three users of our website. If we put this process in the execution of the app, if a user clicks on the Rank Report button to see the rank we have to scan the whole table and do the calculation. Image we have 1,000,000 users and they want to see the rank, 1,000,000 reads of the table is a large pressure for the database and website. So, we use the background process to help us to collect data and do the calculation. The background process scans the table of *userdata,* calculate the total duration of all the users in the database, find out the top-three users and insert the rank, top-three usernames and their total durations to another table *userrank*. Thus, when a user wants to see the Rank Report, we only have to scan the table *userrank* with only three rows which is too much smaller than the table *userdata*, making the time and cost smaller a lot than before. The background process will execute **once every 15 minutes**.

The mechanism of the scheduled execution of background process: we use the services of AWS: **SNS** and **CloudWatch Alarm**. The Lambda function of the background process is triggered by a SNS whose topic is Update. We set up a cloud watch to count the number of messages which is released by this specific topic. If 15 minutes passes and there is no message datapoint in the threshold we set, cloud watch will alarm and publish a message through SNS whose topic is Update, this topic of SNS will trigger the Lambda function of the background process. That is to say, the Lambda function will be triggered once every 15 minutes.

Interactions between the background process and the app: the background process scan the table of *userdata,* calculate the total duration of all the users in the database, find out the top-three users and insert the rank, top-three usernames and their total durations to another table *userrank*. Main program of the app only has to scan the table *userrank* and pass the values to the front side when it needs.

AWS Costs Model

The AWS services and their costs we are currently using are as follows:

- Lambda
- Dynamodb
- S3
- SNS

We will choose to predict the pricing for provisioned capacity, with the number of users at 10, 1000 and 1000,000 respectively

Lambda

13.5 MB of memory is allocated to the lambda function, we can assume one user will execute it 300 times in one month and it ran for 1 second each time. The AWS charges $0.20 per 1M requests with a free tier provides 1M requests per month, and $0.0000166667 for every GB-second with a free tier provides 400,000 GB-s. The monthly charges for a single user would be calculated as follows:

Monthly compute charges:
Total compute (seconds) = 300* (1s) = 300 seconds

Total compute (GB-s) = 300 * 13.5 MB/1024 = 3.955 GB-s

We can observe that both 10,1000, and 1000,000 user's total compute is less than free tier compute, then the Monthly compute charges is 0.

Monthly request charges:
Total requests – Free tier requests = Monthly billable requests

10 users total requests=3000 < 1M, free

1000 users total requests=300,000 < 1M, free

1000,000 users: total requests=300,000,000

Monthly billable requests=299M, Cost= $59.8

Lambda Monthly Costs (10 users)	0
Monthly Costs (1000 users)	0
Monthly Costs(1000,000 users)	$59.8

Dynamodb

We assume that each user will use our application for 4 hours on average per day. Each user consumes 10 write capacity units, 20 write capacity units per day on average. We have three tables in total and assume the average storage for one user is increasing at a speed of 0.6 KB per day.

Services	Prices	Planning usage for 10 users/day	Planning usage for 1000 users/day	Planning usage for 1000,000 users/day	Costs (10 users)	Costs (1000 users)	Costs(1000,000 users)
Write capacity unit	0.000725 USD/h	100	10,000	10,000,000	1.74/day	174/day	174000/day
Read capacity unit	0.000145 USD/h	200	20,000	20,000,000	0.696/ day	69.6/day	69600/day
Data storage	25GB free/month $0.28 /GB-month	6KB	600 KB	600,000 KB	0	0	4914.875 /month

Dynamodb Monthly Costs (10 users)	(1.74+0.696)*30=$73.08
Monthly Costs (1000 users)	(174+69.6)*30=$7308
Monthly Costs(1000,000 users)	(174000+69600)*30+4914.875 =$77994.87

S3

S3 has a fixed size of 14.1 MB. The price of the S3 Standard is $0.026 per GB. Thus, the total monthly cost for S3 is $0.000358.

SNS

We use SNS to send an email notification to the application developer and Lambda Function for background process every 15min, the price for email SNS Notification is $2.00 per 100,000. Thus, the total monthly cost is $2.

Total cost for six month

In summary, the total cost for six months is as follows:

	Lambda	Dynamodb	S3	SNS	Total Costs
Costs (10 users)	0	$73.08*6	$0.000358	$2	$440.48
Costs (1000 users)	0	$7308*6	$0.000358	$2	$43850
Costs(1000,000 users)	$59.8*6	$77994.87*6	$0.000358	$2	$468,330.02