COSC349 Assignment 2 - Your Software in the Cloud

Max Stewart (1086706) & Meg Starnes (1667581)

WebApp link: http://ec2-18-234-154-37.compute-1.amazonaws.com/index.php

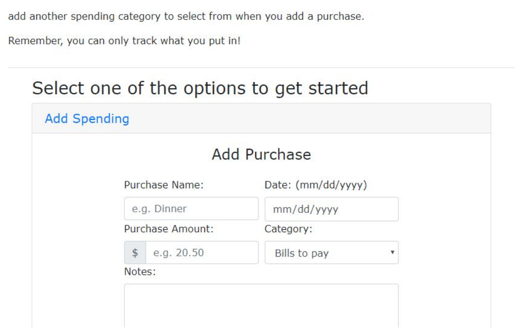
We have built a budgeting and spending tracker which we deployed to the web using AWS services. We deployed our application by following the tutorial linked below. We have a VPC that the Webserver and database run in which allows for a robust, easy connection between the EC2 and RDS. First we created a VPC using an EC2 instance that our Webserver and Database could run in, this had public and private subnets. The webserver is the only thing we needed the public to access and so this was hosted on the public subnet so a user could access it from their own browser. Since our database was only going to be accessed from requests from our web server it could be hosted on the private subnet and the web server was able to access this database as it was in the same VPC, this also meant we had extra security since the public was not able to access the database. We added a security group for the web server that allowed ssh and http access from anywhere, this would allow our users to access the web server from any browser. Our database was given a different security group that allowed MySQL access and so we could query this database from our webpage. After these groups were set up we created an RDS instance, this used the MySQL engine and was connected to our subnets and security groups described above.
We then created an EC2 instance that our web server would run on, this used the Amazon Linux AMI and a t2.small instance type (both were free tier eligible and so were suitable). This EC2 was connected to the VPC, subnets and security groups described above. We installed httpd24, php and MySQL modules and started the webserver, we were now able to deploy our web app on this EC2.
Tutorial link: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide//TUT_WebAppWithRDS.html

We decided to use 2 EC2's as one of our cloud-based services as it allows us to effectively rent resizable units of compute power and networking space within the AWS infrastructure to build our software into the cloud. It avoids the need to invest in hardware so we can develop and deploy our web applications faster and more cost efficiently and removes the need to predict and prepare for high spikes in web traffic by auto scaling up and down when necessary. One of our EC2 instances is the webserver and the other instance runs our VPC.

We also used an RDS to deploy our database into the cloud which ensures that the security and management of the database are automated by AWS. Our RDS was read and written to by our EC2 and stores all of the user's data submitted through the EC2 in the database to access when required as well as managing common database administration tasks.

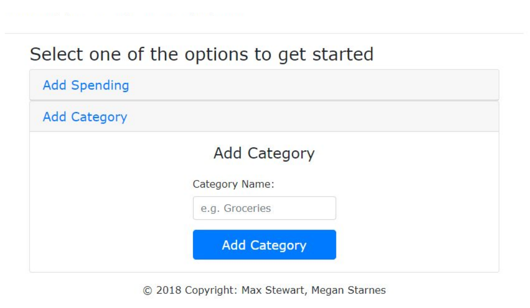
Users can utilize our web application as a money spending resource (top left screenshot) that provides graphs that track their spending over time. They will start on the homepage and by following the instructions on the page they will fill in the forms that will ask for information about their spending such as money spent, date, name, etc. This information will be stored and will be used later to remind the user of what their money was spent on. They can then go to the "Spending Graph" link on the navigation bar to view their spending as different graphs. If they want to see all the transactions as complete lists they can go to the "Database" link on the navigation bar. This page prints the whole database and information

stored within as a table, the user can order this in different ways by using the options in the top right, a user can also delete records off the database if they choose. Our app strives to make it easy to track spending and be easily accessible. Users can access our app using the following link: http://ec2-18-234-154-37.compute-1.amazonaws.com/index.php
They are able to add categories (top right screenshot) for their spending such as bills to pay, school fees, etc, as well as adding individual transactions (top left screenshot) that can be assigned to one of these categories. There is also a graph made from these transactions that users can view to visually see how their spending is changing over time and where their money is being spent (bottom right screenshot). The user can also access all their spending records printed as a table and they can delete specific spending records in the case they have added a wrong record or wrong information in the record (bottom left screenshot).

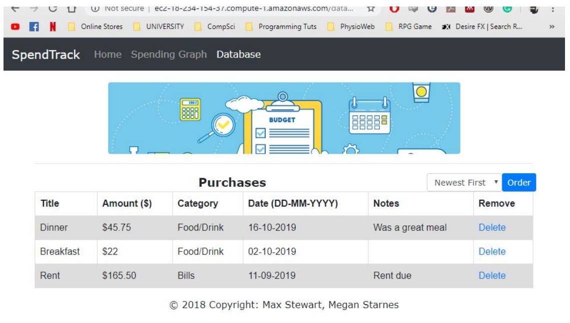

add another spending category to select from when you add a purchase.

Remember, you can only track what you put in!

Select one of the options to get started


Select one of the options to get started

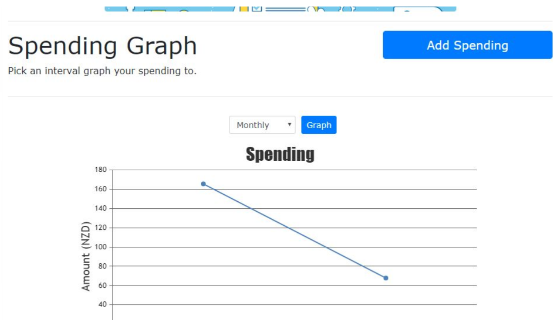
© 2018 Copyright: Max Stewart, Megan Starnes



Purchases

Title	Amount ($)	Category	Date (DD-MM-YYYY)	Notes	Remove
Dinner	$45.75	Food/Drink	16-10-2019	Was a great meal	Delete
Breakfast	$22	Food/Drink	02-10-2019		Delete
Rent	$165.50	Bills	11-09-2019	Rent due	Delete

© 2018 Copyright: Max Stewart, Megan Starnes


Spending Graph

Pick an interval graph your spending to.

These screenshots show our project deployed and active within the cloud.

Our assignment built on top of a web app that was developed for assignment 1. Therefore we used the existing vagrant environment to locally develop the new features. We were able to do this as the database from our local machine use the MySQL engine and therefore the same as the RDS on AWS. This development process was much quicker as we were able to quickly access and modify the files on our local machine. When we were happy with our program we simply transferred the files via SFTP to our Amazon EC2 web server. We simply had to change the credentials for connecting to our RDS and our application was live and ready for testing.