

USING CLOUD COMPUTING TO IMPLEMENT BIDDING WEBSITE APPLICATION



BY –

Shashiraj Waleshwar – ve8664

Rishabh Parekh – hr8687

Siddhant Shah – pc3460

Table of contents

Chapter 1: Introduction	3
1.1 Problem summary	3
1.2 Solution	3
Chapter 2: Implementation Workflow	4
2.1 Architectural overview	5
2.1.1 Architecture of the system	5
2.1.2 Relation between services	5
Chapter 3: Development Process	7
3.1 Description of the development process	7
3.2 Challenges team faced during building this project	19
Chapter 4: Code & Technology	20
4.1 Programming language	20
4.2 Cloud services	20
Chapter 5: Git Access & Source Code	21
Chapter 6: Demo Screenshots from Website	25
Chapter 7: Outstanding Features of this Project	29
Chapter 8: Future Enhancements	29
Chapter 9: References	29

CHAPTER 1: INTRODUCTION

1.1 Problem Summary

An organization or company wants an open platform for the bidding projects for their own organization. So, an open bidding platform is created where the organizations can come together and put a bidding requests for their projects. The reason why we created this is that these companies don't want to put upfront investment on resources like hardware, servers as they don't know about the company size in future.

The company does not want to rely on primitive technology such as manual storage or local server storage which creates many problems while the company grows and it's also very slow while retrieving the employee information from the database and they don't map according to all their databases. So, the company requirements are to provide a system where large data can be stored based on size and storing employee information and helping in retrieving the data with the help of the employee ID which we can use as the primary key where this overall process is done in a fast and efficient manner.

1.2 Solution

To solve the above mentioned problem, our proposed system will eliminate all the above issues and provide efficient results by using cloud computing technology, this system is implemented as a web application on AWS.

The web application we developed is deployed on AWS cloud platform where EC2 is used for launching instances S3 bucket is used for storing for example images and Relational Database Services for connecting to databases, the whole idea is to deploy web application on an EC2 instance which acts as Infrastructure as a Service (IaaS). In this web application we are providing search functionality to retrieve stored information about employees very quickly and efficiently.

So, by this system the company or organization will be more benefited in numerous ways like companies don't need to put upfront investment on purchasing resources as scaling up and scaling down is done whenever it's needed automatically and pay as per usage of the resources only.

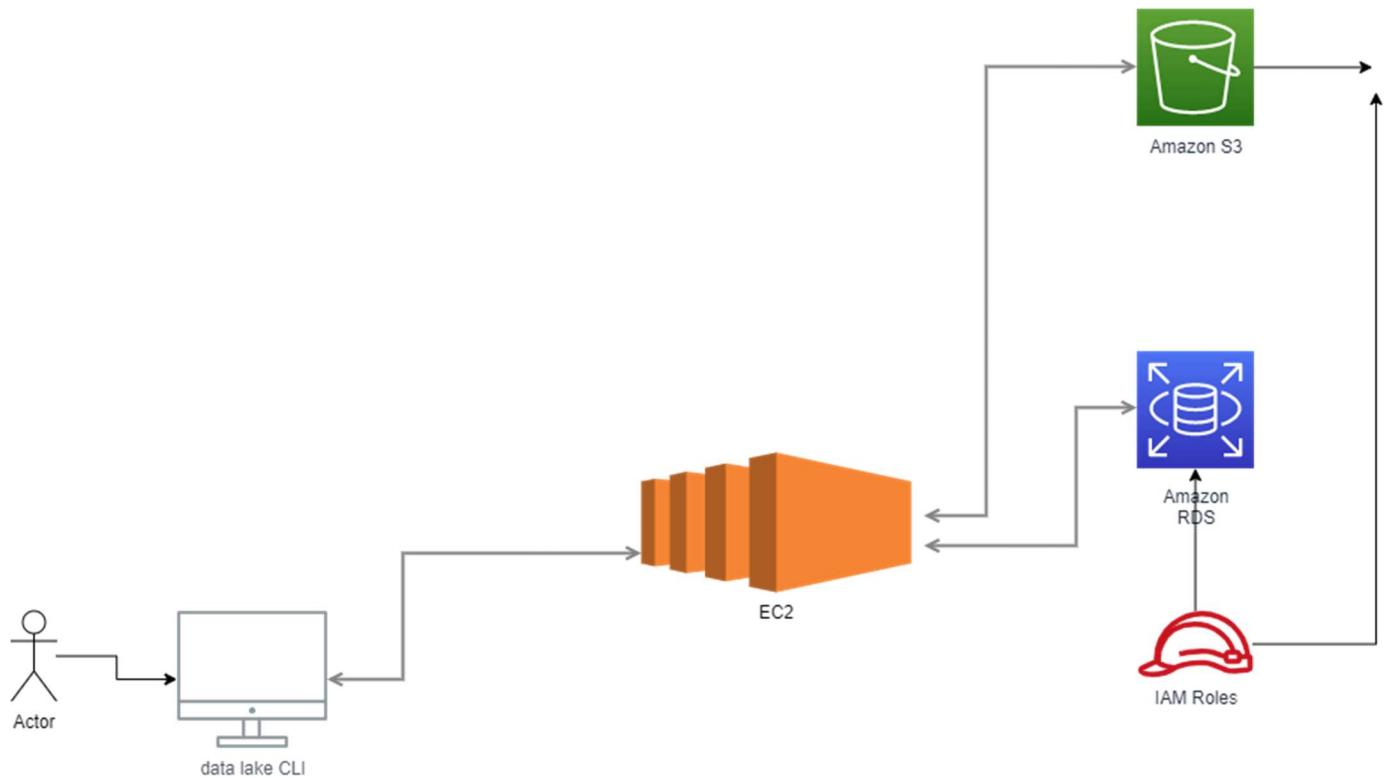
We are using many AWS services to implement this system like RDS, S3, EC2 and Python for programming. The cloud platform offers more storage space, so large data can be stored, the computation speed is very fast so that administrators can retrieve or analyze employee data very efficiently.

CHAPTER 2: IMPLEMENTATION WORKFLOW

2.1 Architectural Overview

Here we are going to discuss the Architecture of the system and Relation between services.

2.1.1 Architecture of the system



The overall architecture of the system can be roughly divided into three components:

1. Selecting and configuring Amazon services
2. Building client-side application
3. Connecting to S3 and RDS databases and IAM for giving permissions to our databases.

In selecting and configuring Amazon services we are using three AWS namely -

1. RDS
2. S3
3. EC2
4. IAM

We need to select these services one by one From AWS dashboard and click appropriate options to set it up.

In building client-side applications, we are using Python and Html for the coding part.

We then deployed our application on EC2 instance which is getting connected to RDS and S3 bucket.

2.1.2 Relation between services

- We are using Amazon **RDS** service, which is used to operate and scale a relational database in the cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks. In this we are using MySQL database and this is used to store the entries. This RDS service will be connected by the EC2 server, this relation is used to access the database.
- We are using Amazon **S3** service, which is a Storage for the internet. We can use it to store and retrieve any amount of data at any time, from anywhere on the web. We should give EC2 instance access to S3, basically to upload the data. To make this relation we will give an IAM role to our instance.
- We are using Amazon **EC2** service, it is a web service for launching and managing Linux/UNIX and Windows Server instances in Amazon's data centers. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Here EC2 should be connected to RDS, S3 as described above, and the website is also deployed in EC2 server.
- We are using an **IAM** service, which is used when we connect with our RDS. We should give authentication to our S3 service, so we need to provide an IAM role for our instance.

CHAPTER 3: DEVELOPMENT PROCESS

Here we are going to discuss the Description of the development process and challenges the team faced during this project building.

3.1 Description of the development process

1) The first phase is to select and configure the Amazon services.

We are using three Amazon service initially -

1. RDS
2. S3

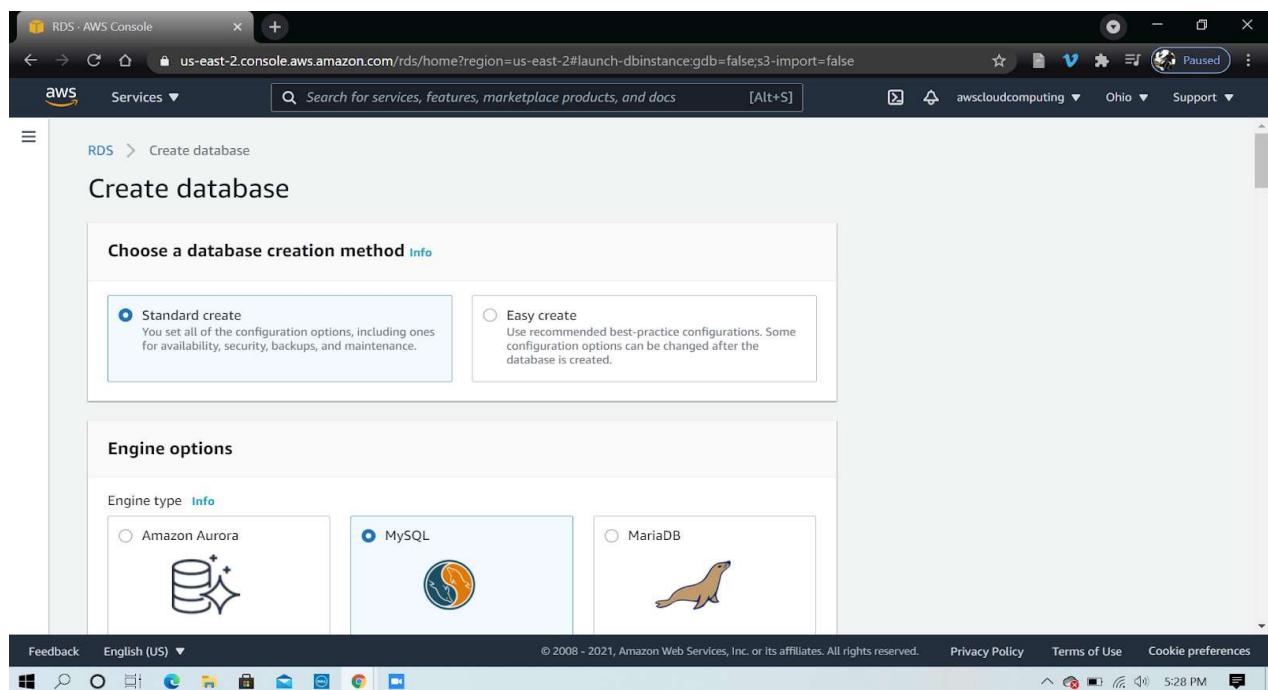
3. EC2

- Deploy MySQL database on AWS

It is done by using Amazon RDS, for this we should select RDS service from AWS management console dashboard and then select MySQL database engine. There are many engines for this project. We are using MySQL engine.

There are many advantages of using RDS, we don't need to launch a server, install MySQL on it and configure it manually instead of that we are just using AWS dashboard and select right options which will be managed automatically by AWS.

Some key advantages are easy to administer, highly scalable, available & durable, fast, secure and inexpensive.



The screenshot shows the 'Database options' configuration page for creating a new MySQL database instance. The 'Database name' is set to 'mydatabase'. The 'Port' is set to '3306'. The 'DB parameter group' is 'default.mysql8.0'. The 'Option group' is 'default:mysql-8-0'. Under 'IAM DB authentication', the 'Disable' option is selected. A note at the bottom states: 'Note: if no database name is specified then no initial MySQL database will be created on the DB Instance.'

Database options

Database name [Info](#)
mydatabase

Note: if no database name is specified then no initial MySQL database will be created on the DB Instance.

Port [Info](#)
TCP/IP port the DB instance will use for application connections.
3306

DB parameter group [Info](#)
default.mysql8.0

Option group [Info](#)
default:mysql-8-0

IAM DB authentication [Info](#)
 Disable
Manage your database user credentials through AWS IAM users and roles.

Feedback Looking for language selection? Find it in the new [Unified Settings](#)

© 2022, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

A blue banner at the top informs the user: '⚠ This database creation flow will be removed starting on June 30, 2021. Starting on June 30, 2021, you won't be able to use this database creation flow. The new database creation flow supports all features. No data loss will occur due to this change. [Switch to the new database creation flow.](#)'

Step 1 Select engine

Step 2 Specify DB details

Step 3 Configure advanced settings

RDS > Create database

Specify DB details

Instance specifications

Estimate your monthly costs for the DB Instance using the [Amazon Web Service Simple Monthly Calculator](#)

DB engine
MySQL Community Edition

License model [Info](#)
general-public-license

DB engine version [Info](#)
MySQL 8.0.28

Feedback Looking for language selection? Find it in the new [Unified Settings](#)

© 2022, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

The screenshot shows the Amazon RDS Management console with the URL us-east-1.console.aws.amazon.com/rds/home?region=us-east-1&lw=classic#databases. The left sidebar has 'Amazon RDS' selected under 'Services'. The main area shows the 'Databases' section with a table listing one database named 'bidding'. The table columns are DB identifier, Instance, Role, Engine, Region & AZ, and Size. The database details are: bidding, MySQL Community, us-east-1b, db.t2.micro.

DB identifier	Instance	Role	Engine	Region & AZ	Size
bidding	MySQL Community		us-east-1b	db.t2.micro	

- Create S3 bucket

It is done by using Amazon S3, for this we should select S3 service from AWS management console dashboard and create it. Check whether your bucket has been deployed in the correct region. Below screenshot shows the configuration that we have done for the S3 bucket creation as we are fetching the images from the S3 bucket.

Bidding System | Add placement | Django site admin | S3 bucket

s3.console.aws.amazon.com/s3/bucket/create?region=us-west-1

Services Search for services, features, blogs, docs, and more [Alt+S]

We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose Provide feedback.

Amazon S3 > Buckets > Create bucket

Create bucket Info

Buckets are containers for data stored in S3. [Learn more](#)

General configuration

Bucket name Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

AWS Region

Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.
[Choose bucket](#)

Feedback Looking for language selection? Find it in the new [Unified Settings](#) © 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Bidding System | Add placement | Django site admin | S3 Management Console

s3.console.aws.amazon.com/s3/buckets?region=us-east-1

Services Search for services, features, blogs, docs, and more [Alt+S]

We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose Provide feedback.

Follow security best practices for S3. [Learn more](#)

Amazon S3 > Buckets

Account snapshot

Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

Buckets (1) Info

Buckets are containers for data stored in S3. [Learn more](#)

[Create bucket](#)

Find buckets by name

Name	AWS Region	Access	Creation date
biddingbucket	US East (N. Virginia) us-east-1	Bucket and objects not public	May 6, 2022, 14:52:38 (UTC-07:00)

Feedback Looking for language selection? Find it in the new [Unified Settings](#) © 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Bidding System | Add placement | Django site admin | S3 Management Console

s3.console.aws.amazon.com/s3/upload/biddingbucket?region=us-east-1

Services Search for services, features, blogs, docs, and more [Alt+S]

We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose Provide feedback.

Upload succeeded View details below.

Upload: status

The information below will no longer be available after you navigate away from this page.

Summary

Destination	Succeeded	Failed
s3://biddingbucket	4 files, 1.8 MB (100.00%)	0 files, 0 B (0%)

Files and folders Configuration

Feedback Looking for language selection? Find it in the new Unified Settings © 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Bidding System | Add placement | Django site admin | S3 Management Console

s3.console.aws.amazon.com/s3/upload/biddingbucket?region=us-east-1

Services Search for services, features, blogs, docs, and more [Alt+S]

We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose Provide feedback.

Upload succeeded View details below.

Files and folders (4 Total, 1.8 MB)

Name	Folder	Type	Size	Status	Error
bid.jpg	-	image/jpeg	119.7 KB	✓ Succeeded	-
dashboard.jpg	-	image/jpeg	1014.3 KB	✓ Succeeded	-
news-2.jpg	-	image/jpeg	491.8 KB	✓ Succeeded	-
news.jpg	-	image/jpeg	247.2 KB	✓ Succeeded	-

Feedback Looking for language selection? Find it in the new Unified Settings © 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Some key advantages are Cost effective storage classes, easily managed data and access controls, query-in-place and process on-request, performance, scalability, availability and durability

The screenshot shows the AWS S3 console interface. At the top, there are three tabs: 'Bidding System', 'Add placement | Django site admin', and 'biddingbucket - S3 bucket'. The 'biddingbucket - S3 bucket' tab is active. Below the tabs, the URL is s3.console.aws.amazon.com/s3/buckets/biddingbucket?region=us-east-1&tab=permissions. The main content area has a blue header with a message: 'We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose Provide feedback.' Below this is a green success message: 'Successfully edited Block Public Access settings for this bucket.' Another message below it says: 'Follow security best practices for S3.' with a 'Learn more' link. The navigation bar at the bottom includes tabs for 'Objects', 'Properties', 'Permissions' (which is highlighted in orange), 'Metrics', 'Management', and 'Access Points'. A 'Permissions overview' section is visible, followed by a 'Block public access (bucket settings)' section which contains a detailed description of how public access is granted through various AWS services. At the bottom of the page, there is a footer with links for 'Feedback', 'Looking for language selection? Find it in the new Unified Settings', '© 2022, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

The screenshot shows the AWS S3 Bucket Policy editor. At the top, there are three tabs: "Bidding System", "Add placement | Django site admin", and "biddingbucket - S3 bucket". The "biddingbucket - S3 bucket" tab is active. Below the tabs, there's a search bar with the placeholder "Search for services, features, blogs, docs, and more" and a keyboard shortcut "[Alt+S]". A blue banner at the top says: "We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose Provide feedback." On the right side of the banner is a "Provide feedback" button. The main area is titled "Edit bucket policy" with an "Info" link. It has two buttons: "Policy examples" and "Policy generator". Below these buttons, the "Bucket ARN" is listed as "arn:aws:s3:::biddingbucket". The "Policy" section contains a JSON code editor. The code is as follows:

```
1  {
2      "Version": "2012-10-17",
3      "Statement": [
4          {
5              "Sid": "Statement1",
6              "Principal": {},
7              "Effect": "Allow",
8              "Action": []
9          }
10     ]
11 }
```

A modal window titled "Edit statement Statement1" is open on the right. It has a "Remove" button. Under "1. Add actions", it says "Choose a service" and has a "Filter services" input field. Below the modal, the status bar shows "Feedback Looking for language selection? Find it in the new Unified Settings" and "© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".

The screenshot shows the AWS Policy Generator. The browser tabs are "Bidding System", "Add placement | Django site admin", "biddingbucket - S3 bucket", and "AWS Policy Generator". The "AWS Policy Generator" tab is active. The URL is "awspolicygen.s3.amazonaws.com/policygen.html".
Step 2: Add Statement(s)
A statement is the formal description of a single permission. See a [description of elements](#) that you can use in statements.
Effect: Allow Deny
Principal:
AWS Service: All Services ('*')
Actions: All Actions ('*')
Amazon Resource Name (ARN):
ARN should follow the following format: arn:aws:s3:::\${BucketName}/\${KeyName}. Use a comma to separate multiple values.
Add Conditions (Optional)
Add Statement: You must enter one or more Principals.
Step 3: Generate Policy
A *policy* is a document (written in the [Access Policy Language](#)) that acts as a container for one or more statements.
Add one or more statements above to generate a policy.

Bidding System | Add placement | Django site admin | biddingbucket - S3 bucket | AWS Policy Generator

awspolicygen.s3.amazonaws.com/policygen.html

Amazon Resource Name (ARN)

ARN should follow the following format: arn:aws:s3:::\${BucketName}/\${KeyName}.
Use a comma to separate multiple values.

Add Conditions (Optional)

Add Statement

You added the following statements. Click the button below to Generate a policy.

Principal(s)	Effect	Action	Resource	Conditions
• *	Allow	• s3:GetObject	arn:aws:s3:::biddingbucket/*	None

Step 3: Generate Policy

A *policy* is a document (written in the [Access Policy Language](#)) that acts as a container for one or more statements.

Generate Policy **Start Over**

This AWS Policy Generator is provided for informational purposes only, you are still responsible for your use of Amazon Web Services technologies and ensuring that your use is in compliance with all applicable terms and conditions. This AWS Policy Generator is provided **as is** without warranty of any kind, whether express, implied, or statutory. This AWS Policy Generator does not modify the applicable terms and conditions governing your use of Amazon Web Services technologies.

©2010, Amazon Web Services LLC or its affiliates. All rights reserved.
An [amazon.com](#) company

Bidding System | Add placement | Django site admin | biddingbucket - S3 bucket | AWS Policy Generator

s3.console.aws.amazon.com/s3/buckets/biddingbucket?region=us-east-1&tab=permissions

Services Search for services, features, blogs, docs, and more [Alt+S]

We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose Provide feedback. [Provide feedback](#)

Successfully edited bucket policy. [Learn more](#)

Follow security best practices for S3. [Learn more](#)

Objects Properties Permissions Metrics Management Access Points

Permissions overview

Access [Public](#)

Block public access (bucket settings)

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Feedback Looking for language selection? Find it in the new [Unified Settings](#) © 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

The screenshot shows the AWS S3 console with the object 'bid.jpg' selected. The top navigation bar includes tabs for 'Bidding System', 'Add placement | Django site ad...', 'biddingbucket - S3 bucket', and 'AWS Policy Generator'. A search bar at the top right contains the placeholder 'Search for services, features, blogs, docs, and more [Alt+S]'. Below the search bar, a message encourages users to provide feedback on the updated S3 console experience. The main content area is titled 'Object overview' and displays the following details:

Attribute	Value
Owner	dshah38
AWS Region	US East (N. Virginia) us-east-1
Last modified	May 6, 2022, 14:54:05 (UTC-07:00)
Size	119.7 KB
Type	jpg
Key	bid.jpg
S3 URI	s3://biddingbucket/bid.jpg
Amazon Resource Name (ARN)	arn:aws:s3:::biddingbucket/bid.jpg
Entity tag (Etag)	2462796f4f9dbb9d499007516eee8795
Object URL	https://biddingbucket.s3.amazonaws.com/bid.jpg

At the bottom of the page, there are links for 'Feedback', 'Looking for language selection? Find it in the new Unified Settings', '© 2022, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

- Deploy a EC2 server

It is done by using Amazon EC2 service, for this we should select EC2 service from AWS management console dashboard. Then launch an instance, where the ubuntu server is used. Select all the desirable options, configure security groups, create a new key pair or select existing one if you have one, and then review and launch. Some key advantages are reliable, scalable and infrastructure on demand.

Launch instance wizard | EC2 Main

us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#LaunchInstanceWizard:

Services ▾ Search for services, features, marketplace products, and docs [Alt+S]

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 1: Choose an Amazon Machine Image (AMI)

SUSE Linux
Free tier eligible

Ubuntu Server 20.04 LTS (HVM), SSD Volume Type - ami-08962a4068733a2b6 (64-bit x86) / ami-06446ad1d755489e (64-bit Arm)
Free tier eligible

Microsoft Windows Server 2019 Base - ami-0b697c4ae566cad55
Windows Free tier eligible

Microsoft Windows Server 2019 Base with Containers - ami-0235468fa1249482b

Select Select Select

Cancel and Exit

64-bit (x86) 64-bit (Arm)

Feedback English (US) ▾

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

Windows Search Task View Taskbar 7:05 PM

Launch instance wizard | EC2 Main

us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#LaunchInstanceWizard:

Services ▾ Search for services, features, marketplace products, and docs [Alt+S]

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below.

Learn more about Amazon EC2 security groups.

Assign a security group: Create a new security group Select an existing security group

Security Group ID	Name	Description	Actions
sg-38960c4c	default	default VPC security group	Copy to new

Inbound rules for sg-38960c4c (Selected security groups: sg-38960c4c)

Type	Protocol	Port Range	Source	Description
All traffic	All	All	sg-38960c4c (default)	
SSH	TCP	22	0.0.0.0/0	

Cancel Previous Review and Launch

Feedback English (US) ▾

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

Windows Search Task View Taskbar 7:06 PM

Launch instance wizard | EC2 Manager

us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#LaunchInstanceWizard:

Services ▾ Search for services, features, marketplace products, and docs [Alt+S]

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 7: Review Instance Launch Details

Please review your instance launch details. You can always change them later.

AMI Details

Ubuntu Server 20.04 LTS (HVM, SSD Volume Type) Free tier eligible Root Device Type: ebs Virtualization Type: KVM

Instance Type

Instance Type ECU

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Create a new key pair Key pair name cloud

Download Key Pair

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location**. You will not be able to download the file again after it's created.

Edit AMI Edit instance type Network Performance Cancel Previous Launch

Feedback English (US) ▾ © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

Instance details | EC2 Manager

us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#InstanceDetails:instanceId=i-0cbfbbed141dedb4b3

New EC2 Experience Tell us what you think

EC2 Instances i-0cbfbbed141dedb4b3

Instance summary for i-0cbfbbed141dedb4b3 (cloud-pro) Info

Updated less than a minute ago

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0cbfbbed141dedb4b3 (cloud-pro)	3.20.221.160 open address	172.31.44.222

Instance state	Public IPv4 DNS	Private IPv4 DNS
Running	ec2-3-20-221-160.us-east-2.compute.amazonaws.com open address	ip-172-31-44-222.us-east-2.compute.internal

Instance type	Elastic IP addresses	VPC ID
t2.micro	-	vpc-0847d263

AWS Compute Optimizer finding	IAM Role	Subnet ID
Opt-in to AWS Compute Optimizer for recommendations. Learn more	-	subnet-9d590bd1

Feedback English (US) ▾ © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

Next connect the server to our machine.

To connect from windows to server one can use Putty or Git bash or Super Putty which is a premium tool.

To connect from a mac one can use the terminal itself, this will connect to an instance deployed in AWS.

Next connect to our RDS

First deploy MySQL client on the server which helps to connect to our RDS machine, for this install MySQL client.

```
ubuntu@ip-172-31-23-106: ~
ubuntu@ip-172-31-23-106:~$ mysql -u admin -p -h employee.ccrxw9mefb.us-east-1.rds.amazonaws.com
Enter password: |
```

```
ubuntu@ip-172-31-23-106: ~
ubuntu@ip-172-31-23-106:~$ mysql -u admin -p -h employee.ccrxw9mefb.us-east-1.rds.amazonaws.com
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3036
Server version: 8.0.20 Source distribution

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> |
```

After installing MySQL client software now we connect to the RDS, so that now we will be in a MySQL shell.

Next deploy database on server

For this first create a database and then create a table in that database for the information to be stored.

// commands

After logging into databases:

Show databases; // gives list of Databases present

```
1 • show databases;
```

Result Grid | Filter Rows: Export: | Result Grid

Database
biddingdatabase
information_schema
mysql
performance_schema
sys

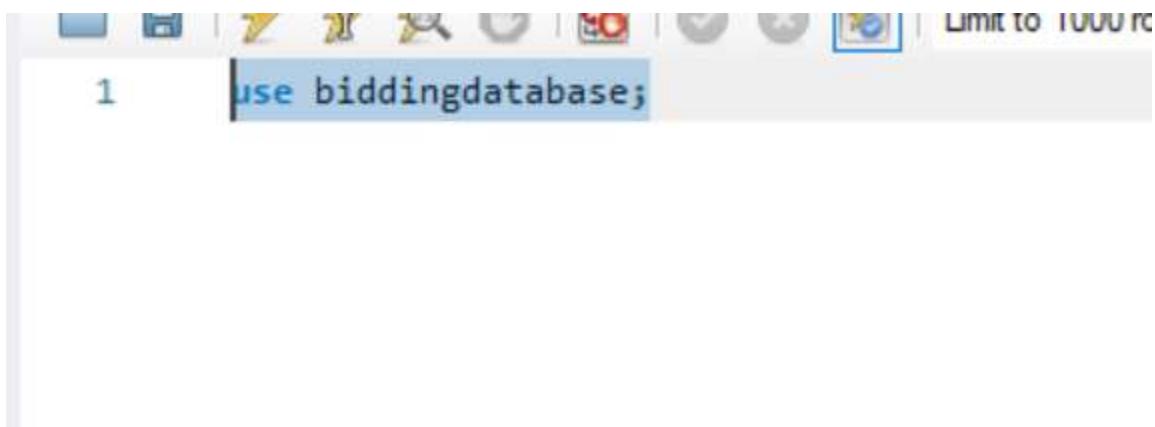
Result 1 × Read Only Co

Output

create database biddingdatabase; // will create a database with the given name.

need to switch to created database)

Use biddingdatabase;



A screenshot of the MySQL Workbench interface. The top menu bar includes 'File', 'Edit', 'View', 'Tools', 'Help', and a 'Session' dropdown. Below the menu is a toolbar with icons for file operations like 'New', 'Open', 'Save', 'Print', 'Copy', 'Paste', 'Find', 'Replace', 'Search', 'Import', 'Export', 'Properties', and 'Help'. A status bar at the bottom right shows 'Limit to 1000 rows'. The main area is a SQL editor window with a light gray background. In the top left of the editor, the number '1' is displayed. To its right, the text 'use biddingdatabase;' is written in blue, indicating it is a MySQL command. The rest of the editor is empty.

(

(now we can create tables in the created database)

These are the tables that we created in our project :

Query 1 × SQL

show tables;

Result Grid | Filter Rows: Export: Result Grid Form Editor Field Types

	Tables_in_biddingdatabase
▶	app_bid
	app_company
	app_placement
	app_placementbid
	auth_group
	auth_group_permissions
	auth_permission
	auth_user
	auth_user_groups
	auth_user_user_permissions
	django_admin_log
	django_content_type

Result 2 × Read Only Output

```
Tables_in_biddingdatabase
app_bid
app_company
app_placement
app_placementbid
auth_group
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissions
django_admin_log
django_content_type
```

2) The second phase is to build the website, configure the website code and deploy the website on EC2 server.

- Build the website

Here to build the website we have used the Python Flask framework. Django is known

asa simple but extensible framework.

- Configure the website code

- 1) Provide Host name of RDS in host region of config code.
- 2) Provide username, password and database name in config code.
- 3) Provide bucket name and region in the configuration code.

- Deploy website on EC2 server

Make the website available on EC2 server.

After connecting to EC2 instance we need to clone a git repository:

```
⚡ ubuntu@ip-172-31-23-106: ~  
ubuntu@ip-172-31-23-106:~$ git clone "https://github.com/mkodali37/cs_623_proj.git"
```

Then navigate to the project folder and install the required packages:

Execute all the below commands on the server to install them on server.

- sudo apt-get install python3

```
ubuntu@ip-172-31-23-106: ~/CS_623_proj
ubuntu@ip-172-31-23-106:~$ ls
CS_623_proj
ubuntu@ip-172-31-23-106:~$ cd CS_623_proj
ubuntu@ip-172-31-23-106:~/CS_623_proj$ sudo apt-get install python3 |
```

Python3 is the language that is coded for the website, we must install it on an existing system where we are running the website.

- sudo apt-get install python3-django

```
ubuntu@ip-172-31-23-106: ~/CS_623_proj
ubuntu@ip-172-31-23-106:~$ ls
CS_623_proj
ubuntu@ip-172-31-23-106:~$ cd CS_623_proj
ubuntu@ip-172-31-23-106:~/CS_623_proj$ sudo apt-get install python3-flask |
```

Flask helps to go ahead and publish a website on to that server.

- sudo apt-get install python3-pymysql

```
ubuntu@ip-172-31-23-106: ~/CS_623_proj
ubuntu@ip-172-31-23-106:~$ ls
CS_623_proj
ubuntu@ip-172-31-23-106:~$ cd CS_623_proj
ubuntu@ip-172-31-23-106:~/CS_623_proj$ sudo apt-get install python3-pymysql |
```

It is a library which helps the website to connect to the MySQL server.

- sudo apt-get install python3-boto

```
ubuntu@ip-172-31-23-106: ~/CS_623_proj
ubuntu@ip-172-31-23-106:~$ ls
CS_623_proj
ubuntu@ip-172-31-23-106:~$ cd CS_623_proj
ubuntu@ip-172-31-23-106:~/CS_623_proj$ sudo apt-get install python3-boto|
```

It is an SDK which helps to connect to S3 service where we upload files.

Then give EC2 instance or server access to basically upload data to S3. We need to do this because on the server we are running some code and that code is trying to interact with the AWS S3 service, when we connect with our RDS we should give authentication to our S3 service, so for that we must give the IAM role to our instance.

To give IAM role to our instance, follow the below steps -

- Select IAM service from AWS management console dashboard.
- Then create an admin role for our AWS account.
- Then select roles for our EC2 instance.

Then we should go to our EC2 service and click on the attach/replace IAM role and select the role which we created from it. Now EC2 is connected to RDS, EC2 is connected to S3, and website deployed in EC2 server.

3.1 Challenges team faced during building this project

- Different Amazon services being used in this project and connection to RDS from Git bash/ Putty is one of the challenges we faced.
- All the commands to install flask, boto and pymysql in the server.
- In the second phase initially we didn't give EC2 instance access to upload data to S3, after that we made a research on IAM roles and used IAM service to upload pictures to S3 bucket.
- Not able to decide whether to use elastic beanstalk or AWS EC2 for deploying our application.

CHAPTER 4: CODE & TECHNOLOGY

4.1 Programming language

1) Frontend

In frontend we used Python and Html languages.

2) Backend

In backend we used RDS (MySQL Database)

4.2 Cloud services

1) Amazon RDS:

We are using Amazon RDS service, which is used to operate and scale a relational database in the cloud. In this we are using MySQL database and this is used to store the entries. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

2) Amazon S3:

We are using Amazon S3 service, which is a Storage for the internet. We can use it to store and retrieve any amount of data at any time, from anywhere on the web.

3) Amazon EC2 :

We are using Amazon EC2 service, it is a web service for launching and managing Linux/UNIX and Windows Server instances in Amazon's data centers.

4) Amazon IAM

AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. When we connect with our RDS we should give authentication to our S3 service, so that we must give the IAM role to our instance.

CHAPTER 5: GIT ACCESS & SOURCE CODE & Youtube URL

Below is the GitHub Link for our project:

<https://github.com/rishabparekh14/CS-632-CloudComputing-OnlineBiddingSystem>

You tube URL :-

Please refer to below screenshot of our code for the modules of the page that we have shown below:

Python Code:

Settings.py :- It contains the code for connecting our application to any Database. May it be RDS or even local database

```
≡ Release Notes: 1.67.0      models.py 4      settings.py X
bidding_system > settings.py > ...
1  # import django_heroku
2  import os
3
4  # Build paths inside the project like this: os.path.join(BASE_DIR, ...)
5  BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
6
7
8  # Quick-start development settings - unsuitable for production
9  # See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/
10
11 # SECURITY WARNING: keep the secret key used in production secret!
12 SECRET_KEY = 'u&u)*c5cl@^syz^*5kq0^em#+3fu7(!=n1mjpoh$i5heoc_t2'
13
14 # SECURITY WARNING: don't run with debug turned on in production!
15 if os.getenv('DJANGO_ENVIRONMENT') == 'PRODUCTION':
16     DEBUG = False
17 else:
18     DEBUG = True
19
20
21 ALLOWED_HOSTS = ["ec2-54-242-209-160.compute-1.amazonaws.com", "54.242.209.160"]
22
23
24 # Application definition
25
26 INSTALLED_APPS = [
27     'django.contrib.admin',
28     'django.contrib.auth',
29     'django.contrib.contenttypes',
30     'django.contrib.sessions',
31     'django.contrib.messages',
32     'django.contrib.staticfiles',
33     'app'
34 ]
```

ⓘ You have Docker installed on your system. Do you want to install the recommended extensions for it?

Install

```

# DATABASES = {
#     'default': {
#         'ENGINE': 'django.db.backends.postgresql_psycopg2',
#         'NAME': 'my_db',
#         'USER': 'hero',
#         'PASSWORD': 'my_db@123',
#         'HOST': 'localhost',
#         'PORT': '5432',
#     }
# }

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}

# Password validation
# https://docs.djangoproject.com/en/2.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
]

```

The above database shown in screenshot is dummy connection but we have actually connected to RDS. But, we have changed the database config and uploaded the code to EC2.

Models.py :- This contains information about the tables that we have created. The tables would be created based on the models mentioned in the below code.

```
app > 📄 models.py > 📁 Company
1  from django.db import models
2  from django.conf import settings
3  from django.contrib.auth.models import User
4  # Create your models here.
5
6  class Company(models.Model):
7      """
8          Simply contains company details, referenced by Placement model
9      """
10
11     company_name = models.CharField(max_length=255)
12     company_address = models.CharField(max_length=255)
13     company_description = models.TextField(default="There is currently no description available for this company")
14
15     company_created = models.DateTimeField(auto_now_add=True)
16     company_modified = models.DateTimeField(auto_now=True)
17
18     def __str__(self):
19         return self.company_name
20
21
22 class Placement(models.Model):
23     """
24         A placement allows investors to bid on company capital raise
25     """
26
27     placement_title = models.CharField(max_length=255)
28     placement_slug = models.SlugField()
29     placement_company = models.ForeignKey(Company, on_delete=models.CASCADE)
30
31
32     placement_created = models.DateTimeField(auto_now_add=True)
33     placement_modified = models.DateTimeField(auto_now=True)
34
```

The screenshot shows a code editor interface with two tabs: 'models.py 4' and 'forms.py 4'. The 'models.py 4' tab is active, displaying Python code for a Django model named 'PlacementBid'. The code defines fields for bid status, creation and modification dates, and user. It also includes a class Meta and __str__ methods. The code is annotated with line numbers from 44 to 72.

```
44     bid_status = models.BooleanField(default=False)
45
46     bid_created = models.DateTimeField(auto_now_add=True)
47     bid_modified = models.DateTimeField(auto_now=True)
48
49     def __str__(self):
50         return '{} - {}'.format(self.user, self.bid_status)
51
52
53     class PlacementBid(models.Model):
54         """
55             The junction table for placement and bid models/tables. Contains every instance of a bid for a pl
56         """
57
58         user = models.ForeignKey(User, on_delete=models.CASCADE)
59         placement = models.ForeignKey(Placement, on_delete=models.CASCADE)
60         bid = models.ForeignKey(Bid, on_delete=models.CASCADE)
61         offer = models.IntegerField()
62         shares = models.IntegerField()
63         confirmed = models.BooleanField(default=False)
64
65         placementbid_created = models.DateTimeField(auto_now_add=True)
66         placementbid_modified = models.DateTimeField(auto_now=True)
67
68     class Meta:
69         ordering = ['-placementbid_modified']
70
71     def __str__(self):
72         return '{} - {} - {}'.format(self.shares, self.offer, self.user)
```

URLS.py :- This contains the url routes that are present in our application

The screenshot shows a code editor interface with two tabs open: 'models.py' and 'urls.py'. The 'urls.py' tab is active and displays the following Python code:

```
1  from django.urls import path
2  from . import views
3  from django.contrib import admin
4
5
6  app_name = 'app'
7
8  urlpatterns = [
9      path('register/', views.register, name='register'),
10     path('login/', views.login, name='login'),
11     path('logout/', views.logout, name='logout'),
12
13     path('', views.home, name='home'),
14
15     path('marketplace/', views.placements, name='placements'),
16     path('marketplace/<placement_slug>/', views.placement_detail, name='placement-detail'),
17
18     path('my-bids/', views.bid_summary, name='bid-summary' ),
19     path('confirm-bids/', views.confirm_bids, name='confirm-bids'),
20
21     path('dashboard/', views.dashboard, name='dashboard'),
22     path('about/', views.about, name='about'),
23
24     path('admin/', admin.site.urls)
25 ]
26
```

Views.py :- It contains the information of the API that we are using as well as contains the information about of the routing that what should be response after the page that we are passing. Also, it contains the logic of the html responses of the page i.e. it connects the html with the business logic as well as the database

```
Ξ Release Notes: 1.67.0 ✎ models.py 4 ✎ views.py 8 ✎
app > ✎ views.py > ...
  1  from django.shortcuts import render, redirect, get_object_or_404
  2  from .forms import CustomRegisterForm
  3  from .models import Placement, PlacementBid, Bid, Company
  4  from django.contrib.auth.models import User
  5  from django.contrib.auth import login as auth_login
  6  from django.contrib.auth import logout as auth_logout
  7  from django.contrib.auth import authenticate
  8  from django.contrib.auth.decorators import login_required
  9  from django.core.paginator import Paginator
10  from django.db.models import Sum, Avg, Count
11
12
13 def register(request):
14     if request.method == 'POST':
15         form = CustomRegisterForm(request.POST)
16         if form.is_valid():
17             form.save()
18             return redirect('app:login')
19     else:
20         form = CustomRegisterForm()
21
22     context = {'form': form}
23
24     return render(request, 'register.html', context)
25
26
27 def login(request):
28     if request.method == 'POST':
29
30         username = request.POST['username']
31         password = request.POST['password']
32
33         user = authenticate(request, username=username, password=password)
34
```

In 1 Col 1 Spaces: 4 LITE 8 LF Python 3.9.9 64-bit @ Coliive 0

Template folder:- The files highlighted on the left are all the html files which contains the code related to views of every page.

```
File Edit Selection View Go Run Terminal Help
login.html - Cloud_Application - Visual Studio Code
EXPLORER ... Release Notes: 1.67.0 models.py 4 login.html
CLOUD_APPLICATION ...
tests.py
urls.py
views.py
bidding_system ...
__pycache__
_init_.py
settings.py
urls.py
wsgi.py
env ...
static ...
templates ...
about.html
base.html
bid-summary.html
dashboard.html
footer.html
home.html
login.html
nav.html
placement_detail.html
placements.html
register.html
.gitattributes
db.sqlite3
manage.py
Procfile
README.md ...
OUTLINE ...
TIMELINE ...
templates > login.html > ...
1  {% extends 'base.html' %} {% block content %}
2  <main>
3      <section class="section">
4          <section class="container">
5              <div class="row">
6                  <div class="col 18 m12 s12">
7                      <div class="card">
8                          <div class="card-content">
9                              <form action="" method="POST" class="form">
10                                 {% csrf_token %}
11                                     <h5>Login page</h5>
12                                     <div class="input-field">
13                                         <label for="" class="label">
14                                             Username
15                                         </label>
16                                         <input type="text" name="username">
17                                     </div>
18                                     <div class="input-field">
19                                         <label for="" class="label">
20                                             Password
21                                         </label>
22                                         <input type="password" name="password">
23                                     </div>
24                                     <div class="input-field">
25                                         <button style="margin-top: 1rem;" class="btn waves-effect waves-light">
26                                             Don't have an account? click <a href="{% url 'app:register' %}" st
27                                             p>Don't have time to sign up? Use the guest account credentials belo
28                                         </button>
29                                     </div>
30                                     </form>
31                                 </div>
32                             </div>
33                         </div>
34                     </div>
```

Requirement.txt : It contains the configuration needed for your application.

≡ Release Notes: 1.67.0 ×

models.py 4

≡ requirements.txt ×

≡ requirements.txt

```
1 dj-database-url==0.5.0
2 Django
3 mysqlclient
4 django-heroku==0.3.1
5 gunicorn==19.9.0
6 pytz==2019.1
7 sqlparse==0.3.0
8 whitenoise==4.1.2
9
```

CHAPTER 6: DEMO SCREENSHOTS FROM WEBSITE

Home Page:

The screenshot shows the homepage of a website. At the top, there is a dark blue header bar with several icons: a house, a document, a grid, a question mark, and a magnifying glass. On the right side of the header are two red buttons labeled "REGISTER" and "LOGIN". Below the header, there is a white box containing a section titled "Notice" with the text: "You must have a registered account and be logged in, in order to view and place bids within our marketplace." Further down, another white box contains a section titled "Bidding Place" with the text: "Welcome to the Bidding System. Here you can find freelancing projects hosted by well-known organizations." Below this text is a pink button with the text "VISIT THE MARKET". At the bottom of the page is a large image showing the interior of a building under construction or renovation, featuring exposed wooden beams and large metal ducts.

Register as a bidder Page:

Registration page

Username:

Required. 150 characters or fewer. Letters, digits and @./+/-/_ only.

Email:

Password:

Your password can't be too similar to your other personal information.

Your password must contain at least 8 characters.

Your password can't be a commonly used password.

Your password can't be entirely numeric.

Password confirmation:

Enter the same password as before, for verification.

CREATE ACCOUNT

[Go back to home page.](#)

Login Page:-

⚠ Not secure | 54.242.209.160:8000/login/

Login page

Username

Password

LOGIN

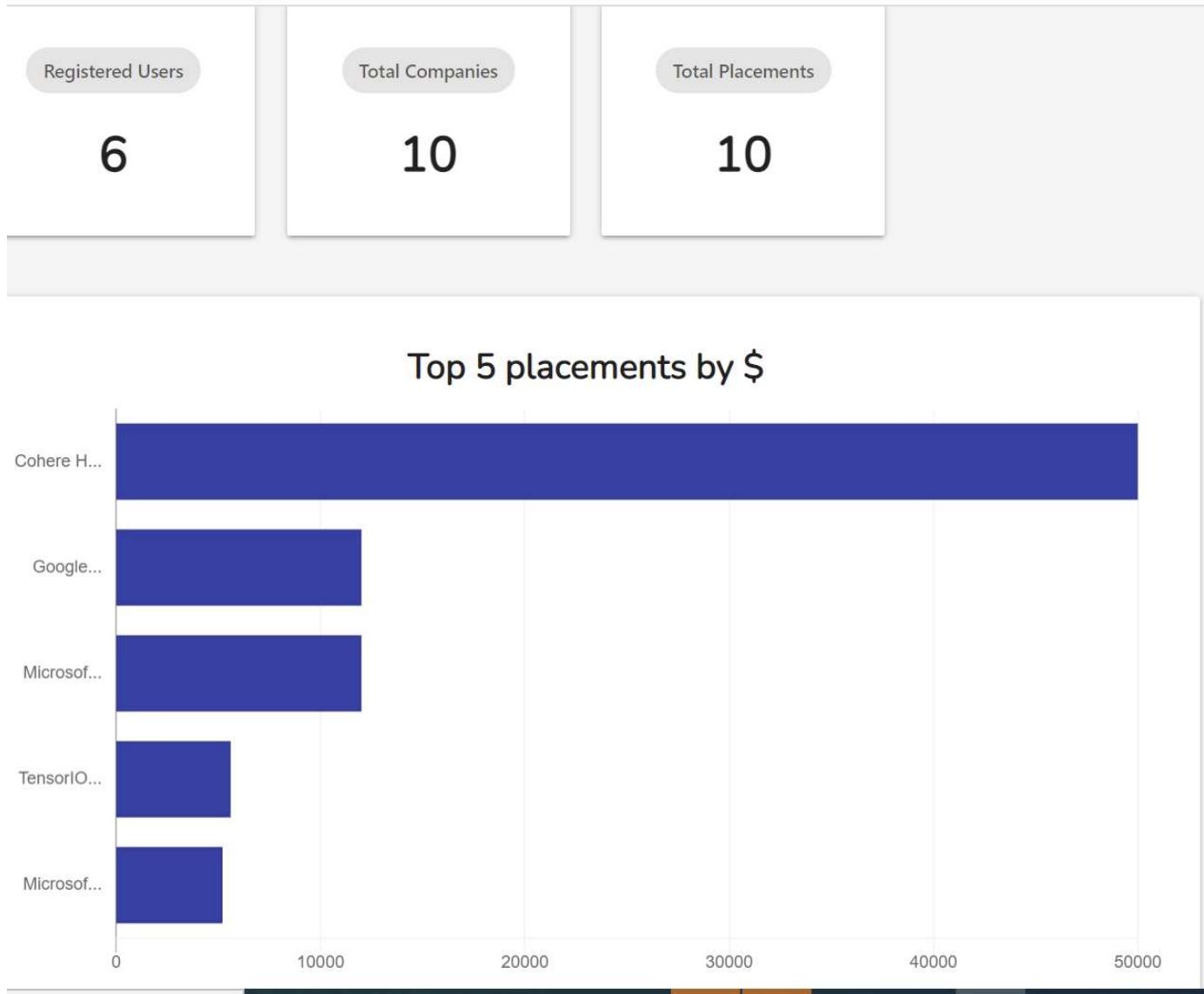
Don't have an account? click [here](#) to register.

Don't have time to sign up? Use the guest account credentials below.

username: 'dc' password: '1'

Statistics Page:-

This page shows the statistics of the amount of bids made by a certain Company.



About us Page which contains the details of all the routes:

Available Routes

/ The home page. Contains news.

about/ Holds details about the application.

register/ Allows users to register an account

login/ Allows users to login to an account

logout/ Allows users to logout and clear their session

marketplace/ Listing of placements from various companies.

marketplace/slug/ Detailed view for a placement within the marketplace.

my-bids/ User scope bid summary for available placements.

dashboard/ Basic analytics and animated charts for the database.

Bids page :

Bids made by us for the project posted by a company.

Your bid summary					
#	Bid status	Company Name	Timestamp	\$ Bidden	Engineers Involved
15	Unconfirmed	Microsoft	May 8, 2022, 6:27 a.m.	5200	4
14	Unconfirmed	Google	May 8, 2022, 6:15 a.m.	12000	3
13	Unconfirmed	Meta	May 8, 2022, 5:27 a.m.	3400	12
10	Unconfirmed	Cohere Health	May 8, 2022, 5:24 a.m.	50000	20
9	Unconfirmed	Cohere Health	May 8, 2022, 5:24 a.m.	1980	2
7	Unconfirmed	Google	May 8, 2022, 5:24 a.m.	3200	3
6	Unconfirmed	Microsoft	May 8, 2022, 5:23 a.m.	12000	8
5	Unconfirmed	TensorIOT	May 8, 2022, 5:23 a.m.	1234	3

[CONFIRM BIDS](#)

2:29 PM

Bids Proposal Page:-

This page contains the bids placed by company which can be looked by everyone who logs in.

Google API Project		May 8, 2022, 4:24 a.m.	May 8, 2022, 4:24 a.m.	VIEW DETAILS
Full-Stack Development		May 8, 2022, 5:14 a.m.	May 8, 2022, 5:14 a.m.	VIEW DETAILS
Front-End Development		May 8, 2022, 5:14 a.m.	May 8, 2022, 5:14 a.m.	VIEW DETAILS
Hardware Optimization		May 8, 2022, 5:15 a.m.	May 8, 2022, 5:15 a.m.	VIEW DETAILS
Data Analysis		May 8, 2022, 5:16 a.m.	May 8, 2022, 5:16 a.m.	VIEW DETAILS
UI Development		May 8, 2022, 5:22 a.m.	May 8, 2022, 5:22 a.m.	VIEW DETAILS
Machine Learning Development		May 8, 2022, 5:22 a.m.	May 8, 2022, 5:22 a.m.	VIEW DETAILS
Bug Tracking Tool		May 8, 2022, 5:23 a.m.	May 8, 2022, 5:23 a.m.	VIEW DETAILS
Hardware Centralization		May 8, 2022, 5:24 a.m.	May 8, 2022, 5:24 a.m.	VIEW DETAILS

CHAPTER 7: OUTSTANDING FEATURES OF THIS PROJECT

1. Usage of different cloud services viz, EC2, S3, RDS , IAM
2. Elimination of Infrastructure, and upfront investment.
3. Enhanced level of flexibility and scalability in comparison to traditional datacentres.
4. Storing and retrieving of data in a fast and efficient manner.
5. Simple Project management.
6. Accessible from anywhere in the world and Increased productivity.

CHAPTER 8: FUTURE ENHANCEMENTS

1. Implementation of AWS DynamoDB instead of MYSQL database, as AWS DynamoDB is a key-value and document database that delivers single-digit millisecond performance at any scale.DynamoDB can handle more than 10 trillion requests per day and can support peaks of more than 20 million requests per second. So, AWS DynamoDB can be one of our future enhancements.

CHAPTER 9: REFERENCES

1. <https://docs.aws.amazon.com/>
2. <https://docs.aws.amazon.com/s3/index.html>
3. <https://docs.aws.amazon.com/pythonsdk/>
4. [2] "Introducing Python: modern computing in simple packages (Second edition.)", Bill Lubanovic, O'Reilly Media, November, 2019., Chapter 18. This is a book that discusses the basic usage of Python programming language within various topics, including the web systems. Bill Lubanovic is a senior software engineer currently working for Internet Archive. O'Reilly is a popular publisher that has published various books related to software development.