



A TECHNICAL PROJECT REPORT ON
“Hotel Management Systems”
TEAM MEMBERS

Sandeep Anumula	- S02057678
Srinivas Madala	- S02060306
Sai Prashanth Reddy Dyapa	- S02060447
Ruthik Juvva	- S02060044
Akash Badavath	- S02059991
Purushottam Reddy Tippani	- S02058159

SOUTHEAST MISSOURI STATE UNIVERSITY
INSTRUCTOR NAME: ZIPPING LU
COURSE: DISTRIBUTED CLOUD COMPUTING
SPRING - 2024

Introduction:

The goal of the Hotel Management System is to make managing a hotel easier while still meeting the needs of both employees and visitors. Without the limitations of conventional manual methods, this system provides a digital platform that facilitates the effective handling of reservations, guest services, and administrative tasks.

Objective:

The Hotel Management System's three primary objectives are to enhance guest experiences, expedite administrative work, and manage hotel operations. In addition to improving overall guest experience, this project aims to increase operational efficiency and optimize revenue potential for the hotel by developing a user-friendly platform.

Project Analysis:

Functional Requirements:

Accommodations can be provided to guests by registering their details in advance.

Check-in and check-out times are displayed in real time for guests, making reservations simple.

Access to the Blog: For news on events, local attractions, and hotel updates, visitors can visit the blog section.

Integration of Dashboard: Guests can effectively manage reservations, requests, and preferences with the integration of an extensive dashboard system.

Non-Functional Requirements: -

Security: Mandatory password changes improve the security of the initial login.

Portability: The system is made to function flawlessly on a variety of hardware and operating systems.

Audit Trail: An audit trail records every login attempt, whether successful or not, to preserve data integrity and keep an eye on system access.

SDLC:

Planning, designing, implementing, testing, and maintaining software applications are all part of the Software Development Life Cycle (SDLC), which is an organized method. The goal of the SDLC process is to guarantee that software projects are finished on schedule, within budget, and without errors.

Software developers can achieve the best possible results in software development by adhering to the SDLC methodology.

The project follows a structured SDLC approach: -

1. Planning
2. Design
3. Implementation
4. Development
5. Testing
6. Maintenance

Modules:

- Room Management: Manages room inventory, descriptions, and pricing.
- Reservation System: Facilitates room booking, modification, and cancellation.

- Payment Gateway: Ensures secure online transactions for room bookings.
- Guest Services: Manages requests such as room service, housekeeping, and concierge.
- Customer Management: Manages guest accounts, preferences, and loyalty programs.
- Content Management: Handles website content, including room details, amenities, and local attractions.
- Staff Management: Manages staff assignments, schedules, and performance tracking.
- Cloud Technologies and Services Used:
- The hotel management system utilized several cloud technologies and services for reliability and scalability:

Amazon Web Services (AWS) for hosting and server management.

Google Cloud Platform (GCP) for data storage and processing.

Firebase for real-time guest interactions and notifications.

Key services included:

- The application servers were hosted on AWS Elastic Compute Cloud (EC2), which was one of the key services.
- The Relational Database Service (RDS) offered by AWS facilitates the effective management of MySQL databases.
- To secure access to AWS services, use AWS Identity and Access Management (IAM).
- Static files and media can be stored on Amazon S3.

- Guni corn: Used to administer and execute the Python application as a WSGI server.
- Nginx: A reverse proxy that improves security, controls SSL/TLS, and boosts capacity to handle loads.

Implementation:

The hotel management system was developed and deployed using a combination of technologies. Python and Django were employed for the back-end functionalities, while HTML5, CSS3, JavaScript, and Bootstrap were utilized for the front-end design.

Key implementation steps included:

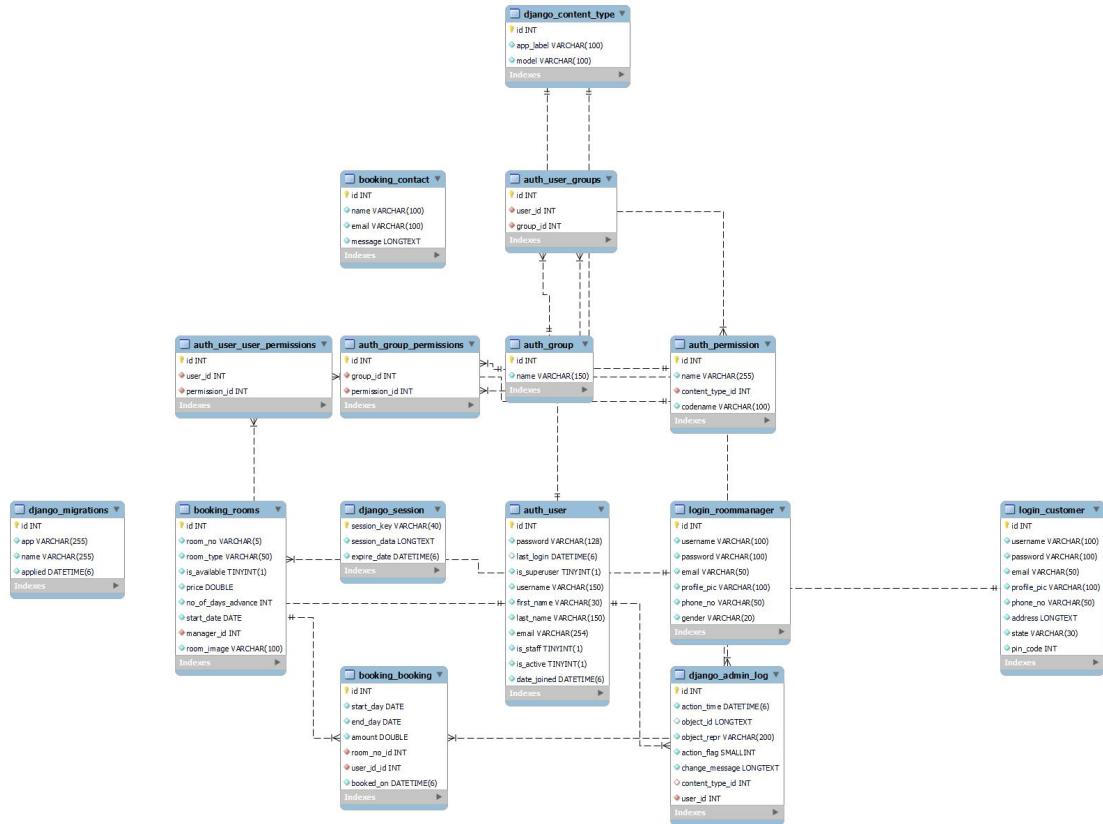
Server Configuration: Configured EC2 instances on AWS running Ubuntu, installed necessary software packages, and set up Guni corn with Nginx to serve the Django application, ensuring optimal performance and scalability.

Database Setup: Utilized AWS RDS for MySQL to host the hotel management system's database, ensuring data security, reliability, and scalability to accommodate the system's requirements.

Deployment: Implemented continuous integration and deployment processes using GitHub Actions. Automated testing and deployment procedures were established to streamline the deployment of updates and ensure the system's stability on AWS EC2 instances.

Tools and Technologies Used for Application Development:

UML diagrams:



Front-End:

HTML5: Utilized for structuring the web content and defining the layout.

CSS3: Applied for styling and enhancing the presentation of web pages.

JavaScript: Implemented to add dynamic behavior and interactivity to the user interface.

Bootstrap: Employed as a front-end framework for creating responsive and mobile-friendly designs.

Back-End:

Python: Chosen as the primary programming language for implementing server-side logic.

Django: Utilized as a high-level Python web framework to expedite development and ensure a clean architecture.

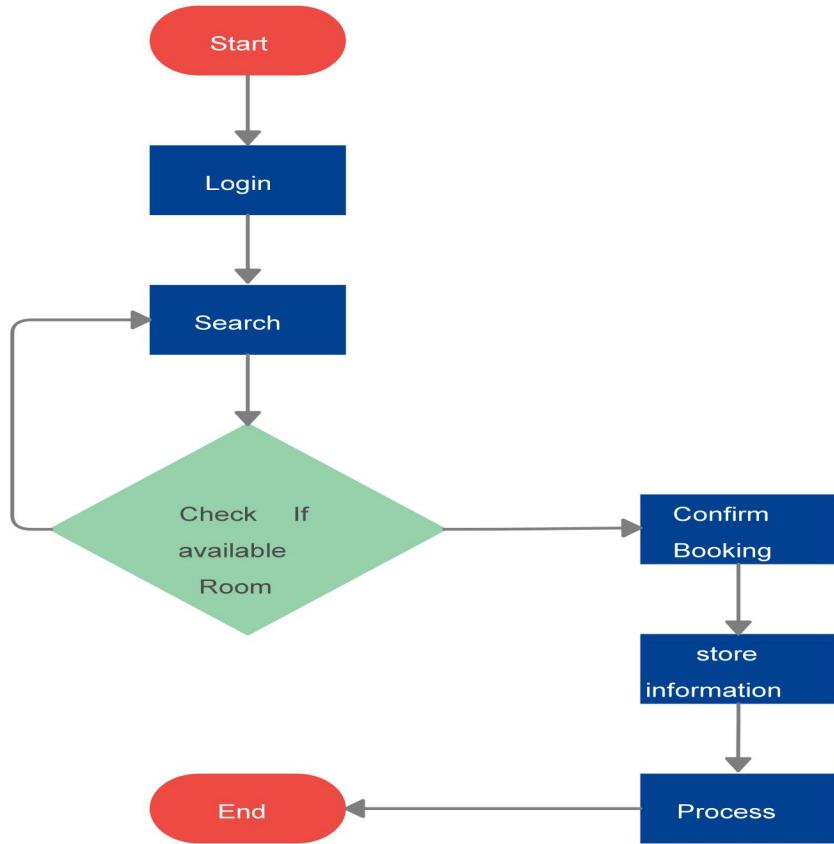
Pandas: Used for efficient data manipulation and analysis, particularly beneficial for handling extensive data sets.

Database:

AWS RDS for MySQL: Deployed as a managed relational database service to store and manage hotel data securely and efficiently.

MySQL Workbench: Utilized as a visual tool for database development and administration, offering features for designing, modeling, and querying the database schema.

Block Diagram:



The Hotel Management System is designed to ensure high availability and scalability, utilizing AWS EC2 instances for hosting and AWS RDS for database management. The system is built using Django, Gunicorn, and Nginx to handle server-side logic and client requests efficiently. The front-end interface, developed with HTML5, CSS3, JavaScript, and Bootstrap, provides a user-friendly and interactive experience.

Challenges and Solutions:

Scalability: Addressed by leveraging cloud-based services such as AWS EC2 and RDS, allowing for automatic scaling of

resources based on demand. This ensures that the hotel management system can handle varying levels of traffic without experiencing performance issues.

Security: Managed by integrating AWS security features like security groups and IAM roles to protect both the application and its data. Encryption protocols and access control mechanisms were implemented to safeguard sensitive information and prevent unauthorized access.

Deployment Complexity: Handled by utilizing tools like Gunicorn and Nginx, which simplify the deployment process and enhance the system's performance. Automated deployment pipelines, configured through services like AWS Code Pipeline, streamline the deployment process and reduce the risk of errors.

Results and Outcomes:

The deployment of the Hotel Management System on AWS yielded significant improvements in handling user requests and data processing speeds. The system demonstrated robustness in managing simultaneous connections, showcasing the effectiveness of the cloud infrastructure utilized.

Insights gained from the deployment include the importance of proper initial configuration and the benefits of cloud scalability in adapting to changing demands. The hotel management system's performance enhancements and increased reliability contribute to improved guest experiences and operational efficiency.

Implementation:

The implementation of a hotel management system follows a similar process to that of an e-commerce platform, with adjustments tailored to the hospitality industry. Here's a breakdown of the implementation process:

Planning:

Define the hotel management system's architecture, identifying key features such as room management, reservation system, guest services, and staff management tools.

Design the user interface and layout, considering factors like room booking process, guest check-in/check-out flow, and staff management interfaces.

Outline the integration of necessary functionalities such as payment processing, guest communication tools, and third-party services for booking management.

Platform Selection:

Choose a suitable technology stack for developing the hotel management system, considering factors like scalability, security, and ease of integration.

Select a hotel management software or framework that aligns with the system's requirements and budget, ensuring it provides essential features for hotel operations.

Development:

Develop the user interface for the hotel management system, focusing on intuitive design and seamless navigation for guests and staff.

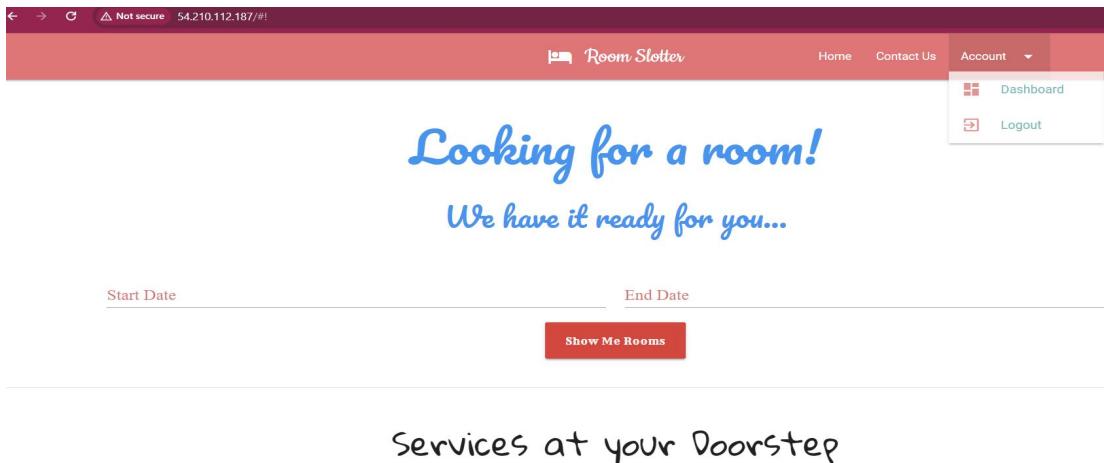
Implement backend functionalities using chosen technologies like Python and Django, ensuring efficient management of rooms, reservations, and guest services.

Integrate necessary features such as payment gateways, communication tools, and database management for storing guest information and transaction records.

Testing:

Main page:

Dashboard:

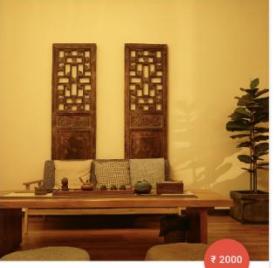


← → ⌂ Not secure 54.210.112.187



₹ 2500

Room with View



₹ 2000

Double room with working tables



₹ 5000

Sea facing rooms

Room Slots
Get your room booked in seconds with amazing offers.

Important Links
[Home](#) [Contact Us](#) [Logout](#)

Signup as room manager:

← → ⌂ Not secure 54.210.112.187/manager/login1

Room Sletter

Home Contact Us Login/Register

For Customer Signup/Login [Click Here](#)

LOGIN SIGN UP

Signup as Room Manager

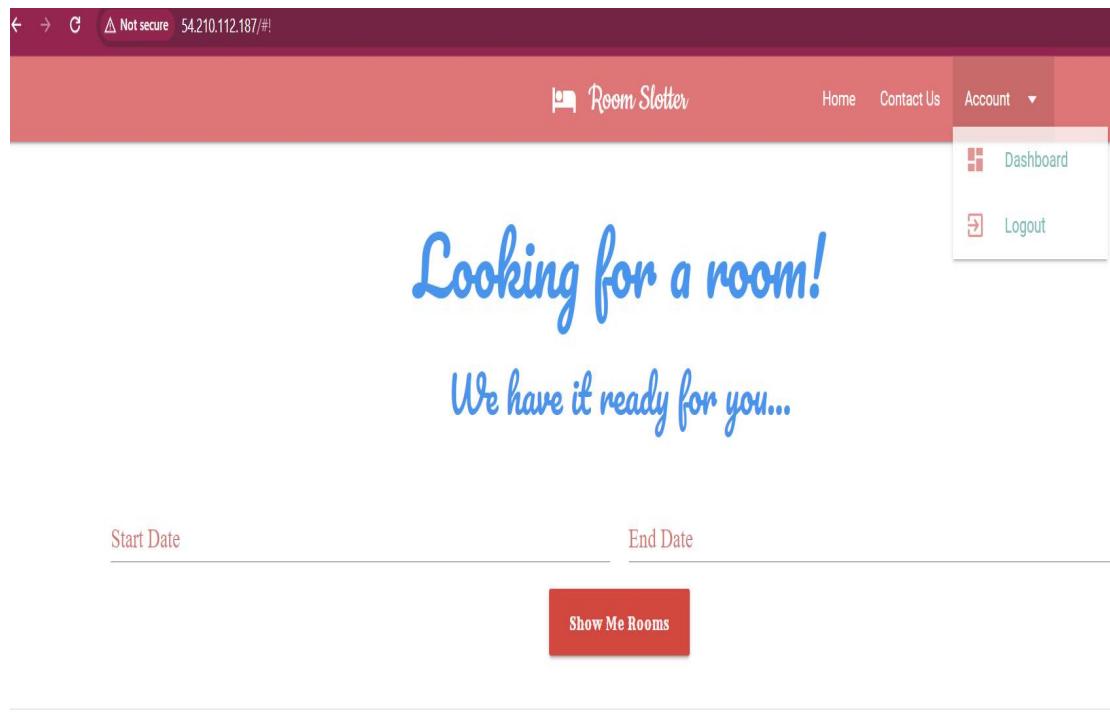
Email

Username

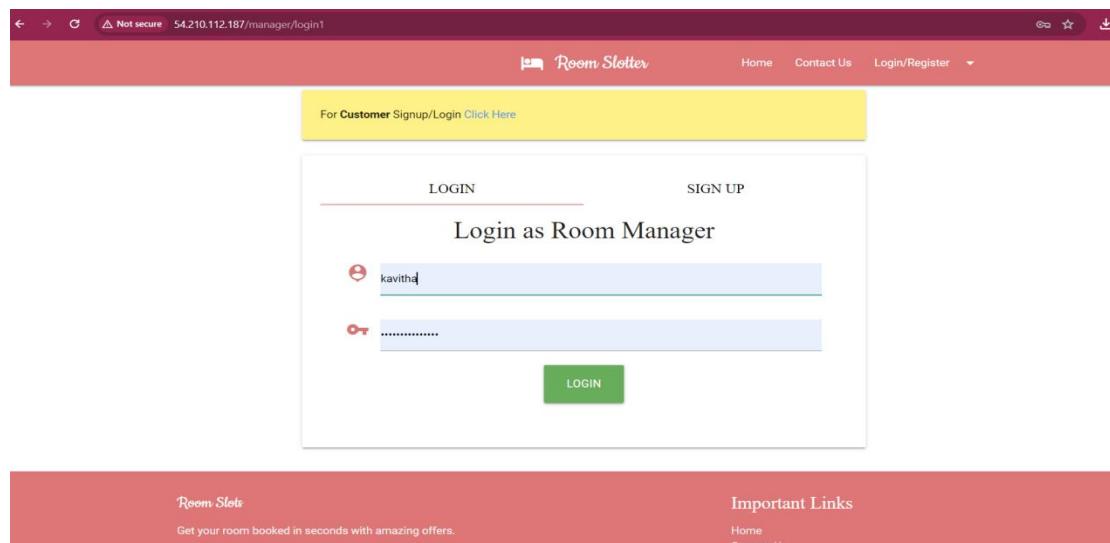
Password

Male Female Others

10 digit phone no



Sign in or login page



Login successful:

The screenshot shows a web browser window with the URL <http://192.168.1.101/rslotter>. The page has a red header bar with the logo "Room Slotter" and navigation links for "Home", "Contact Us", and "Login/Register". Below the header, a message says "You can been sucessfully logout." followed by two login options: "Login as Customer" and "Login as Room Manager".

You can been sucessfully logout.

Login as Customer

Login as Room Manager

Room Slots

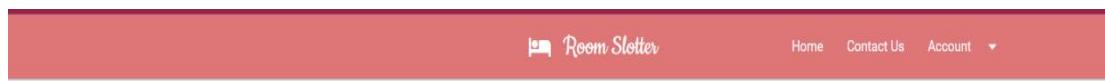
Get your room booked in seconds with amazing offers.

Important Links

Home
Contact Us
Room Manager

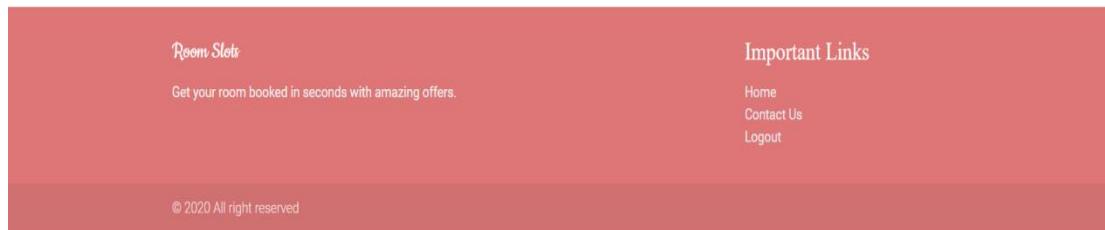
© 2020 All right reserved

Before adding the room page;



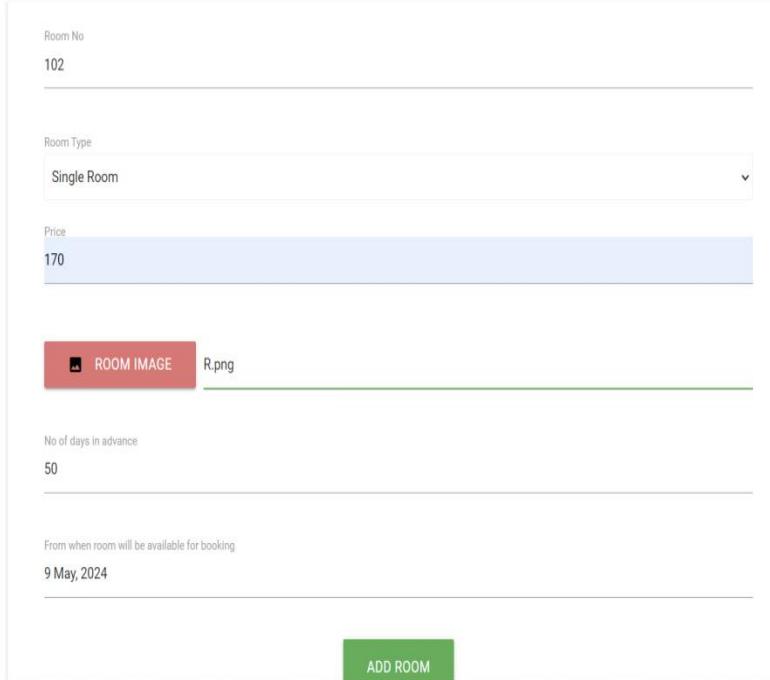
Booking History

Sorry You haven't booked any room so far.



Add the room details:

← → C Not secure 54.210.112.187/user/add-room/new/ ☆



The form consists of several input fields and a file upload area. The fields are: Room No (102), Room Type (Single Room), Price (170), Room Image (R.png), No of days in advance (50), and From when room will be available for booking (9 May, 2024). A green "ADD ROOM" button is at the bottom right.

Room No	102
Room Type	Single Room
Price	170
ROOM IMAGE	R.png
No of days in advance	50
From when room will be available for booking	9 May, 2024

ADD ROOM

← → C Not secure 54.210.112.187/manager/dashboard/1 ☆

 Room Slotted Home Contact Us Account ▾

Room Added Successfully

OCCUPIED ROOMS STATUS

0

TOTAL ROOMS TO MANAGE

2

[ADD ROOMS](#)

Room Status

Room No	Room Type	Price	Customer Name	Booked On	Booking Date	Edit	Delete
101	Single Room	200.0			Available	EDIT	DELETE
102	Single Room	170.0			Available	EDIT	DELETE

Confirm booking your room

← → C Not secure 54.210.112.187/book-now/110 ☆

 Room Slotted Home Contact Us Account ▾

Click on the Confirm Booking to book your room...



Double Room

Stay Duration: 18/May/2024 - 24/May/2024
 Room No: 110
 Total Bill: 6*200.0 = 1200.0 rupees
 Room Manager: kavitha

[CONFIRM BOOKING](#)

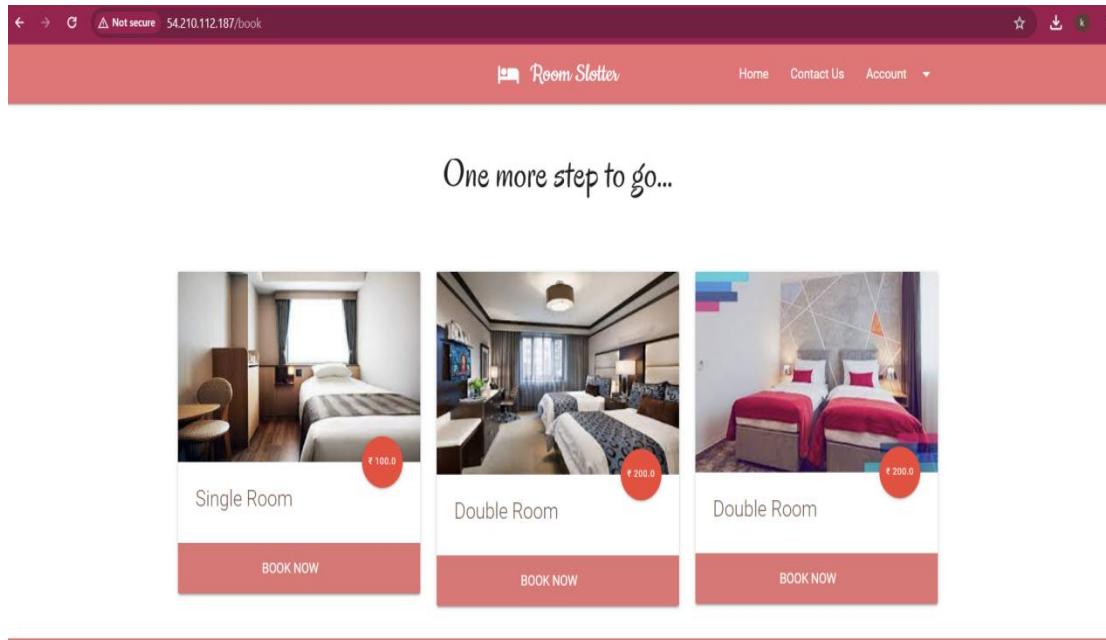
Room Slotted

Get your room booked in seconds with amazing offers.

Important Links

- [Home](#)
- [Contact Us](#)
- [Logout](#)

© 2020 All right reserved



Booking room details:

The screenshot shows the 'Booking History' section of the 'Room Slotter' website. It features two boxes: 'No of active room booking' (1) and 'Rooms booked in past' (0). A 'BOOK MORE ROOMS' button is located above the first box. Below this, a table lists a single booking entry. At the bottom, there are sections for 'Room Slots' (with a note about amazing offers) and 'Important Links' (including Home, Contact Us, and Logout).

Booking ID	Booked On	Start Date	End Date	Billing	Room Manager	Action
1	May 4, 2024, 10:47 a.m.	May 10, 2024	May 12, 2024	400.0	kavitha	CANCEL BOOKING

Click on Book now

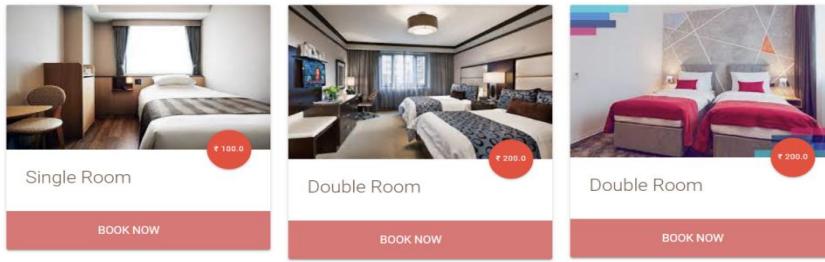
Conformation for booking page

The screenshot shows a booking confirmation for a room. At the top, there's a red header bar with the logo 'Room Slotter' and navigation links for 'Home', 'Contact Us', and 'Account'. A green success message 'Room has been successfully booked.' is displayed above a red button labeled 'BOOK MORE ROOMS'. Below this, two boxes show 'No of active room booking' (1) and 'Rooms booked in past' (0). A section titled 'Booking History' lists one booking entry:

Booking ID	Booked On	Start Date	End Date	Billing	Room Manager	Action
2	May 4, 2024, 11:06 a.m.	May 18, 2024	May 24, 2024	1200.0	kavitha	CANCEL BOOKING

At the bottom, there's a footer with 'Room Slots' and 'Important Links' (Home, Contact Us, Logout), a browser address bar showing '54.210.112.187/book', and the 'Room Slotter' logo again.

One more step to go...



Conduct thorough testing of the hotel management system to identify and address any bugs, errors, or usability issues.

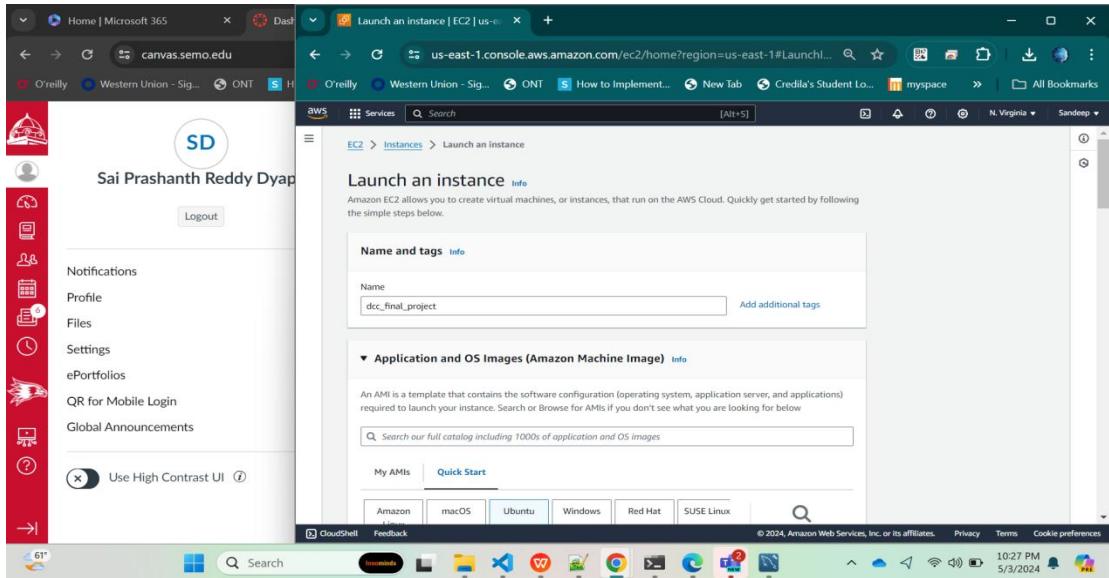
Perform functional testing to ensure all features work as intended, including room booking, reservation management, and staff tools.

Test the system's performance under various conditions to optimize speed and responsiveness.

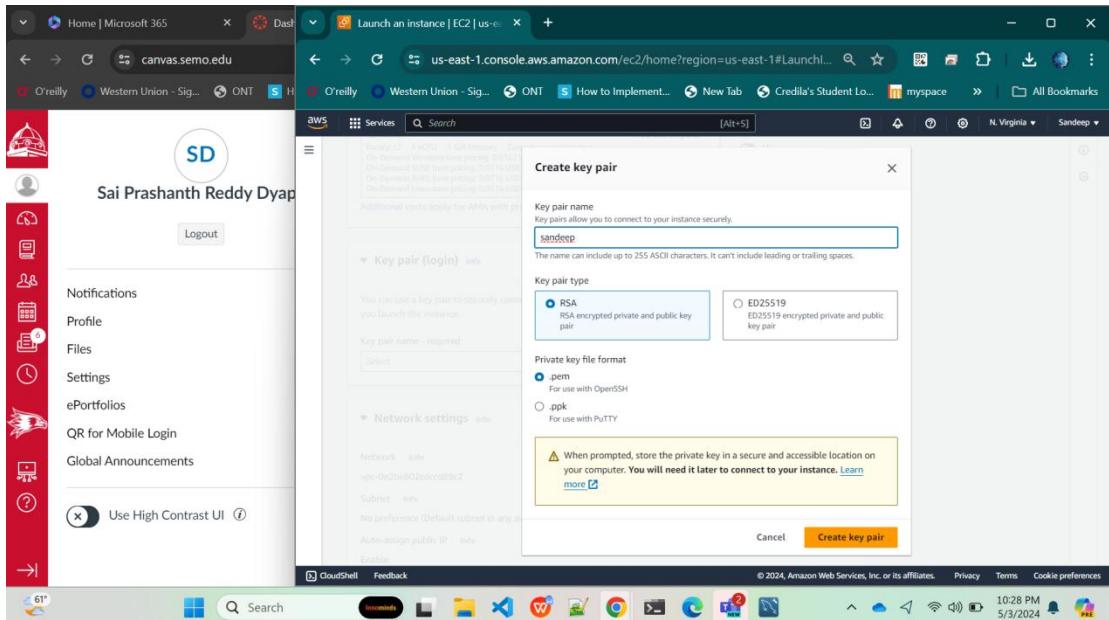
Conduct security testing to identify vulnerabilities and implement measures to protect guest data and system integrity.

Deployment steps:

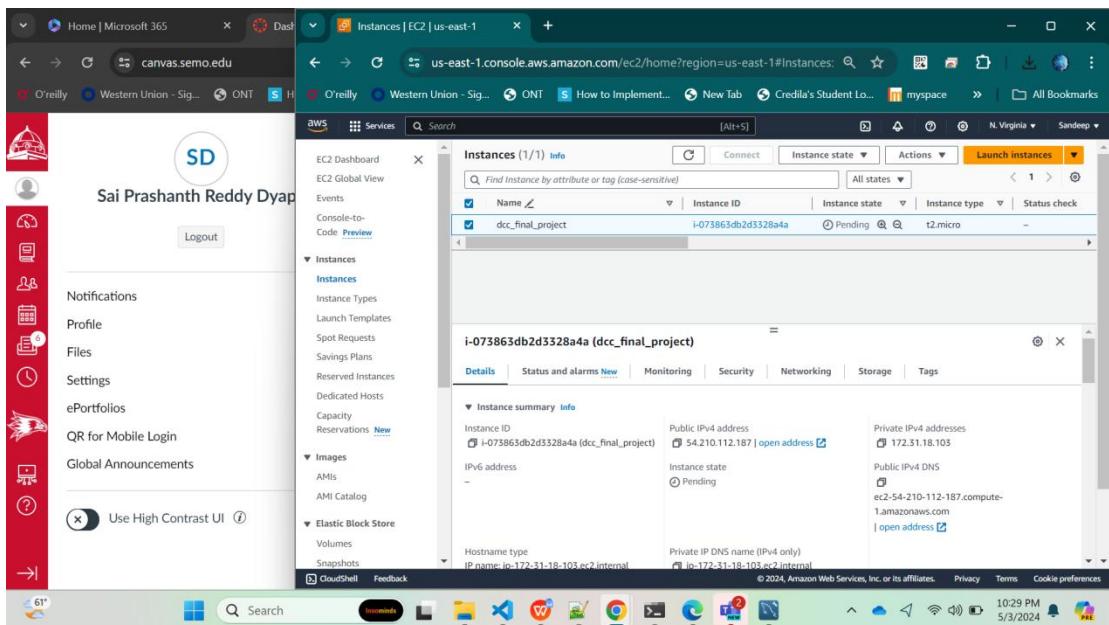
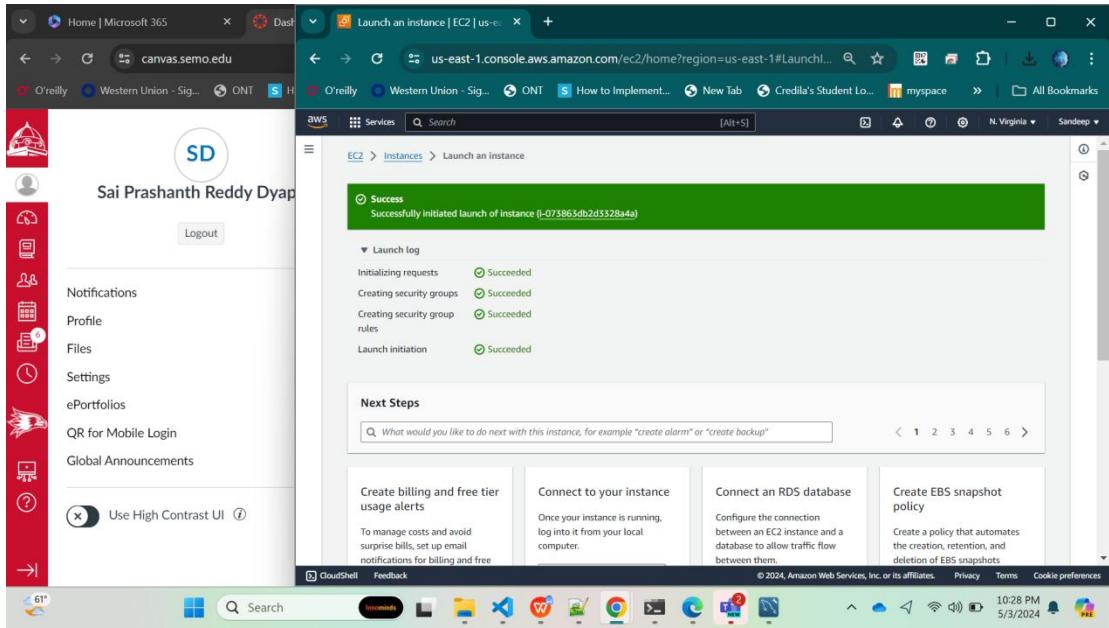
1.Launch Instance



2. create key pair and rsa and .pem select then click on create key pair



Instance sources successfully.



Click on inbound rules

Select SSH, ANY and save rules.

Check inbound rules

The screenshot shows the AWS CloudShell interface with two tabs open. The left tab is titled "ModifyInboundSecurityGroup" and displays the "Inbound rules" configuration for a security group. The right tab shows the AWS Services navigation bar. The CloudShell interface includes a sidebar with user profile information and a taskbar at the bottom.

Inbound rules

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-00345d80b190fbda	SSH	TCP	22	Cu... 0.0.0.0/0	Delete
-	HTTP	TCP	80	An... 0.0.0.0/0	Delete
-	HTTPS	TCP	443	An... 0.0.0.0/0	Delete

Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Buttons: Cancel, Preview changes, Save rules

Instance group rules successfully added.

The screenshot shows the AWS CloudShell interface with two tabs open. The left tab is titled "SecurityGroup | EC2 | us-east-1" and displays the "Details" page for a security group named "sg-0ab88201ef67e6239 - launch-wizard-1". The right tab shows the AWS Services navigation bar. The CloudShell interface includes a sidebar with user profile information and a taskbar at the bottom.

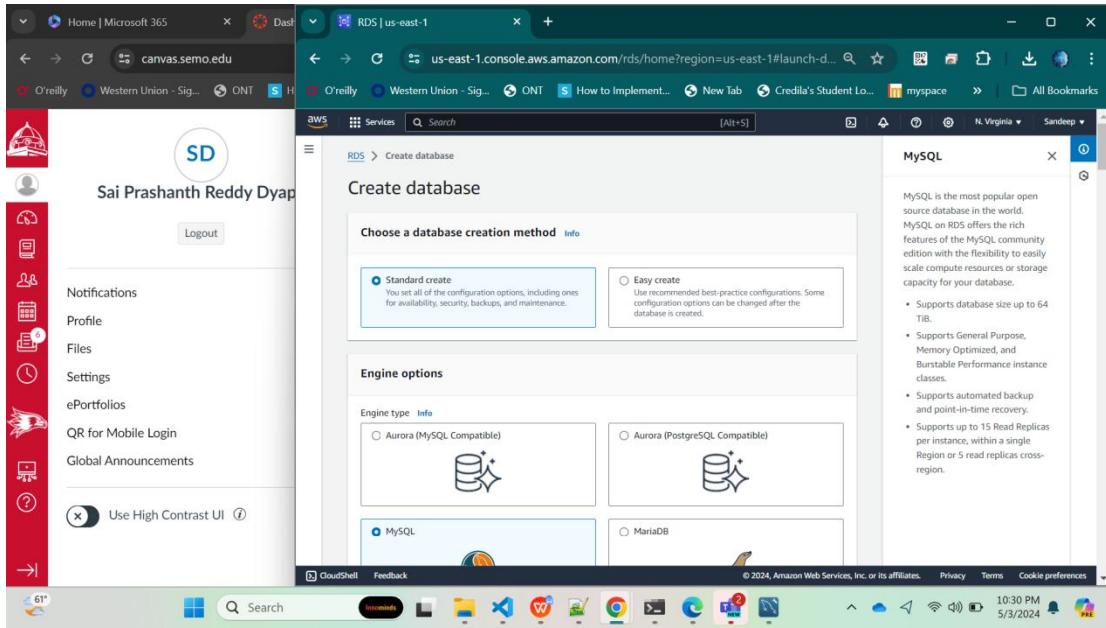
Details

Security group name	Security group ID	Description	VPC ID
launch-wizard-1	sg-0ab88201ef67e6239	launch-wizard-1 created 2024-05-04T02:27:13.431Z	vpc-0e2ba802cdccdb9c2

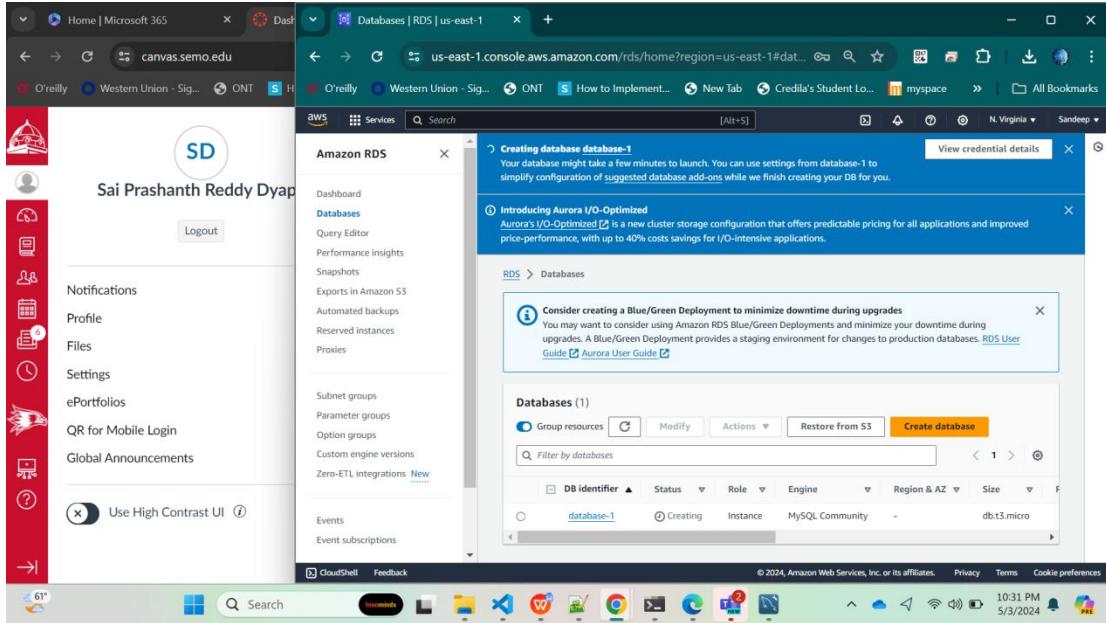
Inbound rules

Source	Protocol	Port range	Action
0.0.0.0/0	TCP	22	Allow
0.0.0.0/0	TCP	80	Allow
0.0.0.0/0	TCP	443	Allow

Database connection to RDS:



Creating database database-1.



Creating database database-1.

Lets check community & security.

The screenshot shows a web browser window with two tabs open. The left tab is "Home | Microsoft 365" and the right tab is "RDS | us-east-1". The main content area displays the Amazon RDS console for a database named "database-1".

Summary:

DB identifier	Status	Role	Engine	Recommendations
database-1	Backing-up	Instance	MySQL Community	Region & AZ
CPU	Class	Current activity		us-east-1b

Connectivity & security:

Endpoint & port	Networking	Security
Endpoint: database-1.crm4siai7.us-east-1.rds.amazonaws.com Port: 3306	Availability Zone: us-east-1b Subnet group: subnet-00000000	VPC security groups: default (sg-072b7537137af9e44) VPC: vpc-0e2ba802cdcc89c2 Publicly accessible: Active

Workbench:

The screenshot shows the MySQL Workbench application window. A dialog box titled "Setup New Connection" is open, prompting for connection details.

Setup New Connection

Connection Name: Sandeep - Dcc Project - Spring 2024

Connection Method: Standard (TCP/IP)

Parameters: SSL Advanced

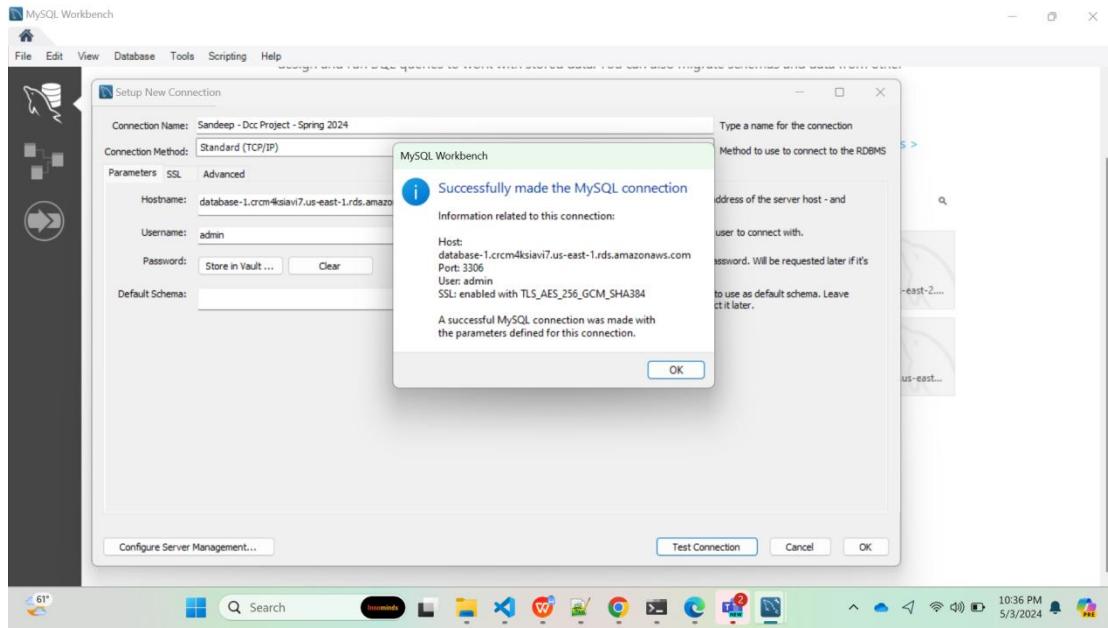
Hostname: database-1.crm4siai7.us-east-1.rds.amazonaws.com Port: 3306

Username: admin

Password: Store in Vault ... Clear

Default Schema:

Test Connection Cancel OK



Databases:

```

settings.py
# Default primary key field type
# https://docs.djangoproject.com/en/3.0/ref/settings/#db-primary-key
default_auto_field = 'django.db.models.AutoField'

WSGI_APPLICATION = 'room_slot.wsgi.application'

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

'default': {
    'ENGINE': 'django.db.backends.mysql',
    'NAME': 'hotel_management_system',
    'USER': 'admin',
    'PASSWORD': 'Python123$',
    'HOST': 'database-1.crm4ksiai7.us-east-1.rds.amazonaws.com',
    'PORT': 3306,
}
# Password validation
# https://docs.djangoproject.com/en/3.0/ref/settings/#auth-password-validation
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
]

```

The screenshot shows an IDE (IntelliJ IDEA) displaying the `settings.py` file of a Django project named "HOTEL-MANAGEMENT-M...". The code defines the database configuration, specifically switching from SQLite to MySQL with the provided connection parameters. The IDE interface includes toolbars, a search bar, and a status bar at the bottom.

File directory cmd open

```
(hotel-env-lat) C:\DCC_05_04\Hotel-Management-master\Hotel-Management-master\room_slot>python manage.py makemigrations
No changes detected

(hotel-env-lat) C:\DCC_05_04\Hotel-Management-master\Hotel-Management-master\room_slot>python manage.py migrate
System check identified some issues:

WARNINGS:
?: (mysql.W002) MySQL Strict Mode is not set for database connection 'default'
  HINT: MySQL's Strict Mode fixes many data integrity problems in MySQL, such as data truncation upon insertion, by escalating warnings into errors. It is strongly recommended you activate it. See: https://docs.djangoproject.com/en/3.0/ref/databases/#mysql-sql-mode
Operations to perform:
  Apply all migrations: admin, auth, booking, contenttypes, login, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
```

Python manage.py run server

```
Applying auth.0003_alter_user_email_max_length... OK
Applying auth.0004_alter_user_username_opts... OK
Applying auth.0005_alter_user_last_login_null... OK
Applying auth.0006_require_contenttypes_0002... OK
Applying auth.0007_alter_validators_add_error_messages... OK
Applying auth.0008_alter_user_username_max_length... OK
Applying auth.0009_alter_user_last_name_max_length... OK
Applying auth.0010_alter_group_name_max_length... OK
Applying auth.0011_update_proxy_permissions... OK
Applying login.0001_initial... OK
Applying login.0002_auto_20200304_1307... OK
Applying booking.0001_initial... OK
Applying booking.0002_booking_roomimage_rooms... OK
Applying booking.0003_auto_20200304_1726... OK
Applying booking.0004_remove_rooms_room_image... OK
Applying booking.0005_rooms_room_image... OK
Applying booking.0006_booking_booked_on... OK
Applying booking.0007_remove_rooms_status... OK
Applying login.0003_auto_20200314_1916... OK
Applying login.0004_auto_20200314_1916... OK
Applying sessions.0001_initial... OK

(hotel-env-lat) C:\DCC_05_04\Hotel-Management-master\Hotel-Management-master\room_slot>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
May 03, 2024 - 22:39:47
Django version 3.0.3, using settings 'room_slot.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

UBUNTU deployed:

A screenshot of a Windows desktop environment. On the left, there's a vertical sidebar with icons for Home, Notifications, Profile, Files, Settings, ePortfolios, QR for Mobile Login, Global Announcements, and a checked 'Use High Contrast UI' option. The main area shows a browser window with the URL 'canvas.semo.edu' and a terminal window titled 'Command Prompt - python n'. The terminal output is as follows:

```
ubuntu@DESKTOP-D7G3HKE:~/spring2024/sandeep$ ssh -i "sandeep.pem" ubuntu@ec2-54-210-112-187.compute-1.amazonaws.com
The authenticity of host 'ec2-54-210-112-187.compute-1.amazonaws.com (54.210.112.187)' can't be established.
ECDSA key fingerprint is SHA256:SUIdkEZekQI34DT2t+M52Q709hIZLpgMJfpmf7NEL0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-54-210-112-187.compute-1.amazonaws.com,54.210.112.187' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1017-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

 System information as of Sat May 4 02:42:58 UTC 2024

 System load: 0.0      Processes:         96
 Usage of /: 20.5% of 7.57GB   Users logged in:     0
 Memory usage: 21%          IPv4 address for eth0: 172.31.18.103
 Swap usage:  0%

 Expanded Security Maintenance for Applications is not enabled.

 0 updates can be applied immediately.

 Enable ESM Apps to receive additional future security updates.
 See https://ubuntu.com/esm or run: sudo pro status

 The list of available updates is more than a week old.
```

Command :sudo apt-get install

Sudo apt-get update

A screenshot of a Windows desktop environment. On the left, there's a vertical sidebar with icons for Home, Notifications, Profile, Files, Settings, ePortfolios, QR for Mobile Login, Global Announcements, and a checked 'Use High Contrast UI' option. The main area shows a browser window with the URL 'canvas.semo.edu' and a terminal window titled 'Command Prompt - python n'. The terminal output is as follows:

```
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

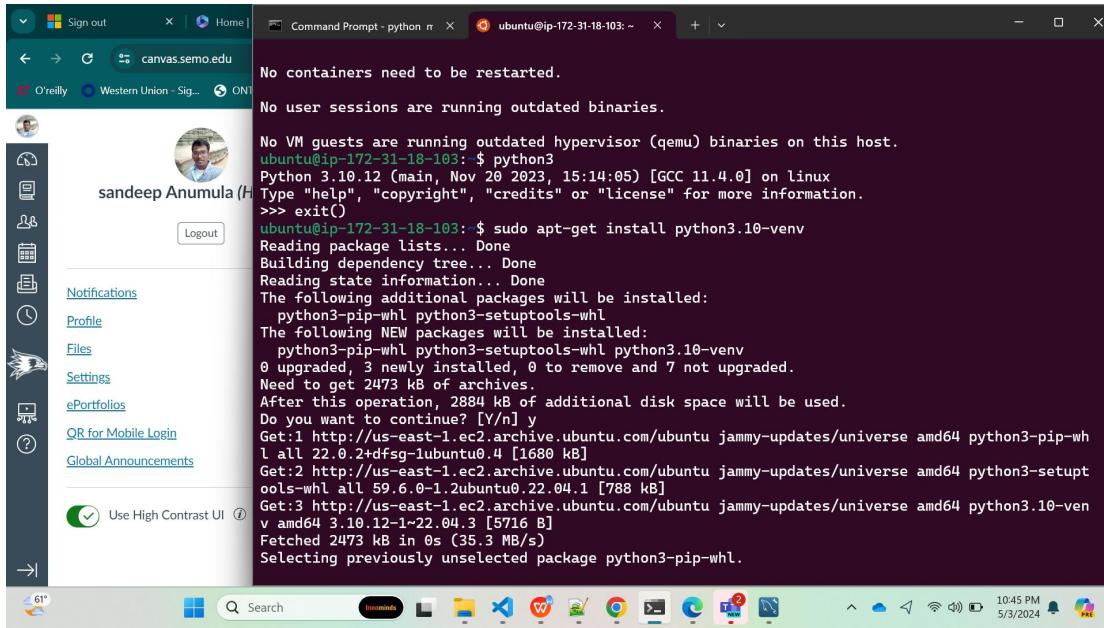
ubuntu@ip-172-31-18-103: $ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
...
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1612 kB]
...
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [304 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [1830 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [311 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1072 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [245 kB]
```

Next command sudo apt-get upgrade

Fetched 31.1 MB in 6s (5481 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages have been kept back:
 distro-info-data landscape-common linux-aws linux-headers-aws linux-image-aws
 python3-update-manager update-manager-core
The following packages will be upgraded:
 cloud-init cpio klibc-utils less libc-bin libcurl3 libklibc libnghttp2-14 libnss3
 locales openssh-client openssh-server openssh-sftp-server ubuntu-advantage-tools
 ubuntu-pro-client ubuntu-pro-client-l10n
17 upgraded, 0 newly installed, 0 to remove and 7 not upgraded.
Need to get 13.1 MB of archives.
After this operation, 60.4 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libc6 amd64 2.35-0
ubuntu3.7 [3235 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libc-bin amd64 2.3
5-0ubuntu3.7 [706 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 openssh-sftp-serv
er amd64 1:8.9p1-3ubuntu0.7 [38.9 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 openssh-server amd
64 1:8.9p1-3ubuntu0.7 [435 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 openssh-client amd
64 1:8.9p1-3ubuntu0.7 [906 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgnutls30 amd64
3.7.3-4ubuntu1.5 [966 kB]

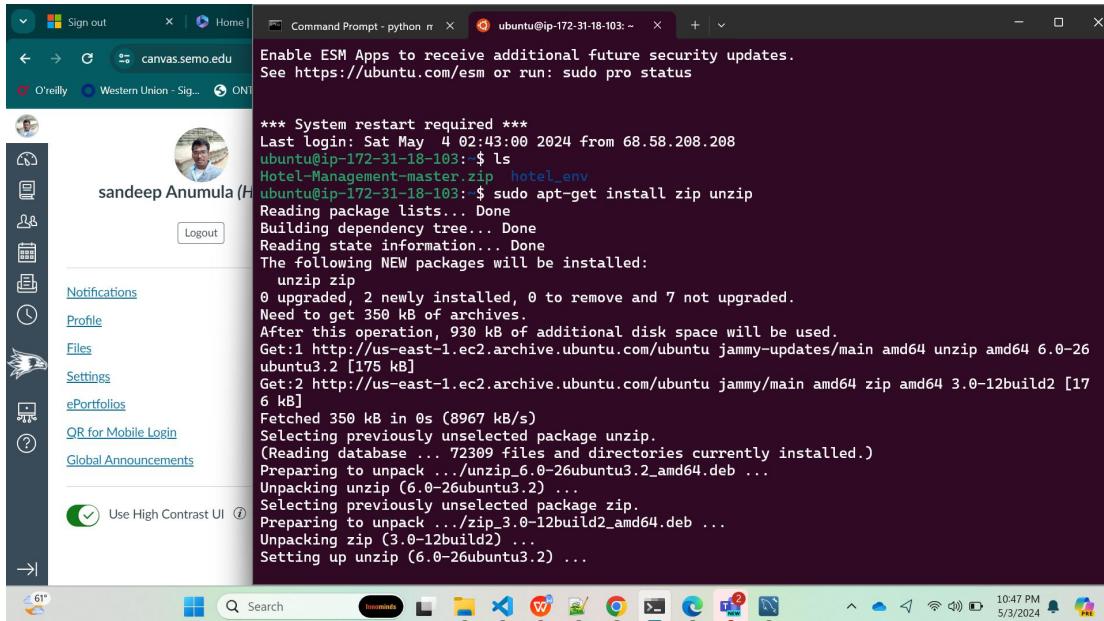
Command: sudo apt-get install supervisor nginx

No containers need to be restarted.
No user sessions are running outdated binaries.
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-18-103: \$ sudo apt-get install python3-dev python3-pip nginx supervisor
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
 build-essential bzip2 cpp cpp-11 dpkg-dev fakeroot fontconfig-config fonts-dejavu-core g++
 g++-11 gcc gcc-11-base javascript-common libalgorithm-diff-perl
 libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan6 libatomic1 libc-dev-bin
 libc-devtools libcc1-0 libcrypt-dev libdeflate0 libdpkg-perl libexpat1-dev
 libfakeroot libfile-fcntllock-perl libfontconfig1 libgcc-11-dev libgd3 libgomp1 libisl23
 libitm1 libjbig0 libjpeg-turbo8 libjpeg libjs-jquery libjs-sphinxdoc libjs-underscore
 liblsan0 libmpc3 libnginx-mod-http-geoip2 libnginx-mod-http-image-filter
 libnginx-mod-http-xslt-filter libnginx-mod-mail libnginx-mod-stream
 libnginx-mod-stream-geoip2 libnsl-dev libpython3-dev libpython3.10-dev libquadmath0
 libstdc++-11-dev libtiff5 libtirpc-dev libtsan0 libubsan1 libwebp7 libxml2 linux-libc-dev
 lto-disabled-list make manpages-dev nginx-common nginx-core python3-wheel python3.10-dev
 rpcsvc-proto zlib1g-dev
Suggested packages:
 bzip2-doc cpp-doc gcc-11-locales debian-keyring g++-multilib g++-11-multilib gcc-11-doc
 gcc-multilib autoconf automake libtool flex bison gdb gcc-doc gcc-11-multilib glibc-doc bzip2
 libgd-tools libstdc++-11-doc make-doc fcgiwrap nginx-doc ssl-cert supervisor-doc
The following NEW packages will be installed:
 build-essential bzip2 cpp cpp-11 dpkg-dev fakeroot fontconfig-config fonts-dejavu-core g++
 g++-11 gcc gcc-11-base javascript-common libalgorithm-diff-perl



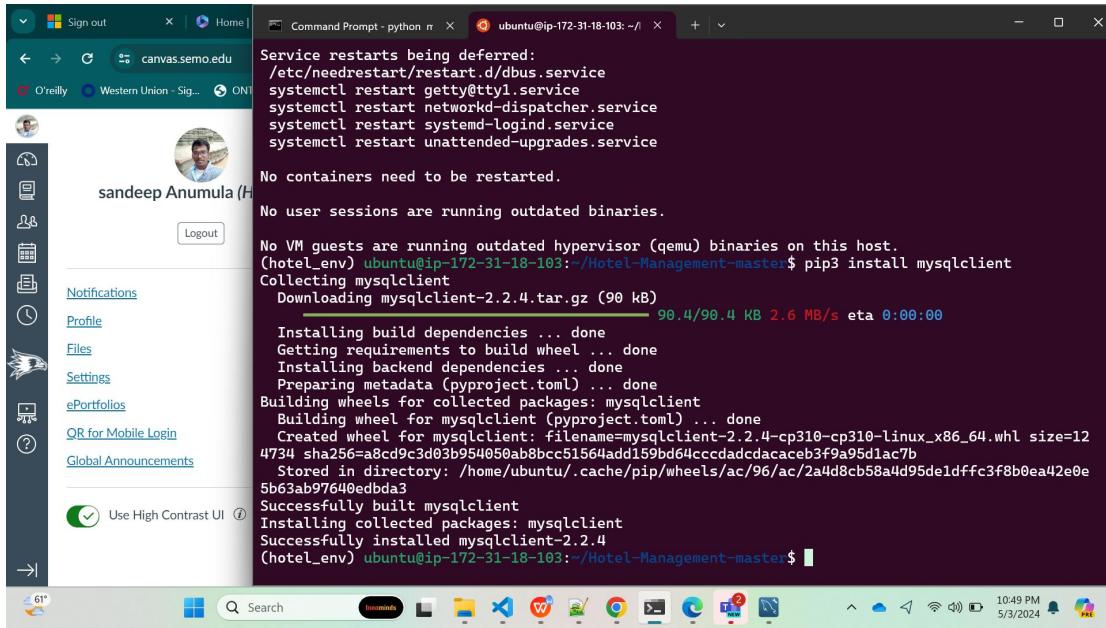
No containers need to be restarted.
No user sessions are running outdated binaries.
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-18-103: ~ \$ python3
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
ubuntu@ip-172-31-18-103: ~ \$ sudo apt-get install python3.10-venv
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
 python3-pip-whl python3-setuptools-whl
The following NEW packages will be installed:
 python3-pip-whl python3-setuptools-whl python3.10-venv
0 upgraded, 3 newly installed, 0 to remove and 7 not upgraded.
Need to get 2473 kB of archives.
After this operation, 2884 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 python3-pip-whl all 22.0.2+dfsg-lubuntu0.4.1 [1680 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 python3-setuptools-whl all 59.6.0-1.2ubuntu0.22.04.1 [788 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 python3.10-venv amd64 3.10.12-1~22.04.3 [5716 B]
Fetched 2473 kB in 0s (35.3 MB/s)
Selecting previously unselected package python3-pip-whl.

Sudo apt-get install zip unzip

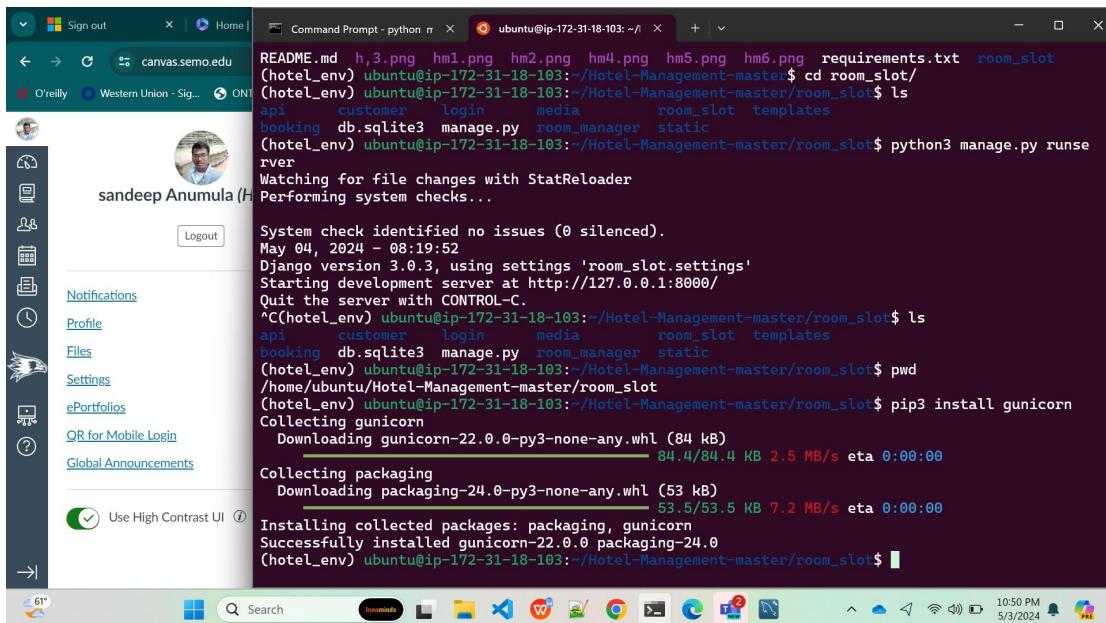


Enable ESM Apps to receive additional future security updates.
See <https://ubuntu.com/esm> or run: sudo pro status
*** System restart required ***
Last login: Sat May 4 02:43:00 2024 from 68.58.208.208
ubuntu@ip-172-31-18-103: ~ \$ ls
Hotel-Management-master.zip hotel.env
ubuntu@ip-172-31-18-103: ~ \$ sudo apt-get install zip unzip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
 unzip zip
0 upgraded, 2 newly installed, 0 to remove and 7 not upgraded.
Need to get 350 kB of archives.
After this operation, 930 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 unzip amd64 6.0-26
ubuntu3.2 [175 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 zip amd64 3.0-12build2 [17
6 kB]
Fetched 350 kB in 0s (8967 kB/s)
Selecting previously unselected package unzip.
(Reading database ... 72309 files and directories currently installed.)
Preparing to unpack .../unzip_6.0-26ubuntu3.2_amd64.deb ...
Unpacking unzip (6.0-26ubuntu3.2) ...
Selecting previously unselected package zip.
Preparing to unpack .../zip_3.0-12build2_amd64.deb ...
Unpacking zip (3.0-12build2) ...
Setting up unzip (6.0-26ubuntu3.2) ...

Pip3 install mysqlclient



```
Service restarts being deferred:  
/etc/needrestart/restart.d/dbus.service  
systemctl restart getty@tty1.service  
systemctl restart networkd-dispatcher.service  
systemctl restart systemd-logind.service  
systemctl restart unattended-upgrades.service  
  
No containers need to be restarted.  
  
No user sessions are running outdated binaries.  
  
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
(hotel_env) ubuntu@ip-172-31-18-103:~/Hotel-Management-master$ pip3 install mysqlclient  
Collecting mysqlclient  
  Downloading mysqlclient-2.2.4.tar.gz (90 kB) 90.4/90.4 KB 2.6 MB/s eta 0:00:00  
    Installing build dependencies ... done  
      Getting requirements to build wheel ... done  
      Installing backend dependencies ... done  
      Preparing metadata (pyproject.toml) ... done  
    Building wheels for collected packages: mysqlclient  
      Building wheel for mysqlclient (pyproject.toml) ... done  
        Created wheel for mysqlclient: filename=mysqlclient-2.2.4-cp310-cp310-linux_x86_64.whl size=12  
4734 sha256=a8cd9c3d03b954050ab8bcc51564add159bd64cccdadcacceb3f9a95d1ac7b  
        Stored in directory: /home/ubuntu/.cache/pip/wheels/ac/96/ac/2a4d8cb58a4d95de1dfffc3f8b0ea42e0e  
5b63ab97640edbda3  
Successfully built mysqlclient  
Installing collected packages: mysqlclient  
Successfully installed mysqlclient-2.2.4  
(hotel_env) ubuntu@ip-172-31-18-103:~/Hotel-Management-master$
```



```
README.md h_3.png hml.png hm2.png hm4.png hm5.png hm6.png requirements.txt room_slot  
(hotel_env) ubuntu@ip-172-31-18-103:~/Hotel-Management-master$ cd room_slot/  
(hotel_env) ubuntu@ip-172-31-18-103:~/Hotel-Management-master/room_slot$ ls  
api customer login media room_slot templates  
booking db.sqlite3 manage.py room_manager static  
(hotel_env) ubuntu@ip-172-31-18-103:~/Hotel-Management-master/room_slot$ python3 manage.py runserver  
Watching for file changes with StatReloader  
Performing system checks...  
  
System check identified no issues (0 silenced).  
May 04, 2024 - 08:19:52  
Django version 3.0.3, using settings 'room_slot.settings'  
Starting development server at http://127.0.0.1:8000/  
Quit the server with CONTROL-C.  
C(hotel_env) ubuntu@ip-172-31-18-103:~/Hotel-Management-master/room_slot$ ls  
api customer login media room_slot templates  
booking db.sqlite3 manage.py room_manager static  
(hotel_env) ubuntu@ip-172-31-18-103:~/Hotel-Management-master/room_slot$ pwd  
/home/ubuntu/Hotel-Management-master/room_slot  
(hotel_env) ubuntu@ip-172-31-18-103:~/Hotel-Management-master/room_slot$ pip3 install gunicorn  
Collecting gunicorn  
  Downloading gunicorn-22.0.0-py3-none-any.whl (84 kB) 84.4/84.4 KB 2.5 MB/s eta 0:00:00  
Collecting packaging  
  Downloading packaging-24.0-py3-none-any.whl (53 kB) 53.5/53.5 KB 7.2 MB/s eta 0:00:00  
Installing collected packages: packaging, gunicorn  
Successfully installed gunicorn-22.0.0 packaging-24.0  
(hotel_env) ubuntu@ip-172-31-18-103:~/Hotel-Management-master/room_slot$
```

The screenshot shows a Windows desktop environment. On the left is a vertical sidebar with various icons and links such as Notifications, Profile, Files, Settings, ePortfolios, QR for Mobile Login, and Global Announcements. A user profile for 'sandeep Anumula' is displayed. The main area has two windows: a browser window for 'canvas.semo.edu' and a terminal window for 'ubuntu@ip-172-31-18-103:/etc'. The terminal window displays the following text:

```
Performing system checks...
System check identified no issues (0 silenced).
May 04, 2024 - 08:19:52
Django version 3.0.3, using settings 'room_slot.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
'C(hotel_env) ubuntu@ip-172-31-18-103:~/Hotel-Management-master/room_slot$ ls
api    customer  login   media   room_slot templates
booking db.sqlite3 manage.py room_manager static
(hotel_env) ubuntu@ip-172-31-18-103:~/Hotel-Management-master/room_slot$ pwd
/home/ubuntu/Hotel-Management-master/room_slot
(hotel_env) ubuntu@ip-172-31-18-103:~/Hotel-Management-master/room_slot$ pip3 install gunicorn
Collecting gunicorn
  Downloading gunicorn-22.0.0-py3-none-any.whl (84 kB)
     84.4/84.4 KB 2.5 MB/s eta 0:00:00
Collecting packaging
  Downloading packaging-24.0-py3-none-any.whl (53 kB)
     53.5/53.5 KB 7.2 MB/s eta 0:00:00
Installing collected packages: packaging, gunicorn
Successfully installed gunicorn-22.0.0 packaging-24.0
(hotel_env) ubuntu@ip-172-31-18-103:~/Hotel-Management-master/room_slot$ whereis gunicorn
gunicorn: /home/ubuntu/hotel_env/bin/gunicorn
(hotel_env) ubuntu@ip-172-31-18-103:~/Hotel-Management-master/room_slot$ cd /etc/supervisor/conf.d/
(hotel_env) ubuntu@ip-172-31-18-103:~/etc/supervisor/conf.d$ ls
(hotel_env) ubuntu@ip-172-31-18-103:~/etc/supervisor/conf.d$ sudo touch gunicorn.conf
(hotel_env) ubuntu@ip-172-31-18-103:~/etc/supervisor/conf.d$ sudo nano gunicorn.conf
(hotel_env) ubuntu@ip-172-31-18-103:~/etc/supervisor/conf.d$ sudo mkdir /var/log/gunicorn
(hotel_env) ubuntu@ip-172-31-18-103:~/etc/supervisor/conf.d$ '
```

Server deployment started:

The screenshot shows a Windows desktop environment. On the left is a vertical sidebar with various icons and links such as Notifications, Profile, Files, Settings, ePortfolios, QR for Mobile Login, and Global Announcements. A user profile for 'sandeep Anumula' is displayed. The main area has two windows: a browser window for 'canvas.semo.edu' and a terminal window for 'ubuntu@ip-172-31-18-103:/etc'. The terminal window displays the same log output as the previous screenshot, indicating the deployment process is still ongoing.

Url: <http://127.0.0:8000/>

Deployment:

Set up hosting for the hotel management system, choosing a reliable and scalable hosting provider.

Configure the domain name and ensure proper DNS settings for the website's accessibility.

Deploy the system to the hosting environment, ensuring all components are properly configured and functioning.

Maintenance and Updates:

Regularly maintain and update the hotel management system to address any issues, enhance functionality, and improve user experience.

Manage room inventory, update pricing, and handle guest requests through the system's administrative interface.

Monitor system performance and security, implementing necessary updates and patches to ensure smooth operation.

By following these steps, a hotel management system can be effectively implemented to streamline hotel operations and enhance guest satisfaction.

Evaluation:

PROs:

Global Reach: Using assistance of hotel management systems, lodging establishments can reach a wider audience outside of their immediate neighborhood.

Convenience: By enabling guests to make reservations at any time and from any location without having to physically visit the hotel, these systems provide a convenient booking experience.

Cost-effective: By automating procedures like guest services and reservation administration, hotel management systems help lower operating expenses.

More Insights: Hotels can customize their services and marketing plans by gaining important insights into the behavior and preferences of their guests.

Personalized experiences from booking to check-out are made possible by these systems, which improves overall satisfaction.

24/7 Availability: Visitors can make reservations and enquiries at any time thanks to the hotel management systems' round-the-clock accessibility.

Scalability: These systems have the capacity to expand to CONs:

Security Risks: Hotel management systems are vulnerable to cybersecurity threats such as data breaches and hacking, putting guest information at risk.

Customer Service Challenges: Providing quality customer service can be challenging, particularly with inquiries related to reservations, room preferences, and billing.

Operational Complexity: Managing a hotel management system requires dealing with various operational tasks, including room inventory, housekeeping schedules, and staff management.

Payment Processing Fees: Hotels incur fees for processing credit card transactions, which can impact profitability.

Lack of Personal Touch: Despite providing convenience, hotel management systems may lack the personal touch of traditional hospitality, affecting guest relationships and loyalty.

Competitive Market: The hotel industry is highly competitive, requiring hotels to continually innovate and adapt to stay ahead of competitors.

Future Works:

The submission of individualized experiences catered to the tastes and requirements of every visitor is expected to become a greater emphasis for hotel management systems.

Mobile-Friendly Solutions: To accommodate the increasing number of tourists who use smartphones, a bigger focus will be placed on mobile-friendly interfaces and mobile booking capabilities.

Integration with Emerging Technologies: Operational efficiency and guest experiences will be improved through integration with technologies like voice assistants, blockchain, and augmented reality.

Sustainability Initiatives: To satisfy the rising demand for environmentally friendly travel options, hotels will give priority

to sustainability initiatives, such as eco-friendly operations and ethical sourcing.

Upgraded Delivery Services: To provide guests with same-day service and amenity delivery, hotels can investigate forming alliances with nearby couriers and deploying technology.

Constant Innovation: To stay competitive, hotels must make constant investments in new ideas and adjust meet changing demands from visitors and market trends.

Conclusion: -

To sum up, the Hotel Management System is a solid solution created as a component of the curriculum for Distributed Cloud Computing. A scalable, secure, and high-performing hotel management platform is created by this project, which showcases the effective use of modern web development techniques and cloud technologies. The system's goals are to increase operational effectiveness, boost guest satisfaction, and satisfy the changing demands of the hospitality sector by integrating cutting-edge features and seamless user experiences.