DESIGN AND IMPLEMENTATION OF HOTEL RESERVATION USING PYTHON

A Project report submitted to

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

In partial fulfilment of the requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

By

BHUMIREDDY SUSMITA (Y19CS1211)

KAPA SUPRIYA (Y19CS1271)

AMBATI DEEPAK (Y19CS1205)

Under the Guidance of

Mr. K. MADHU KIRAN

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CHALAPATHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Accredited by NAAC with 'A' grade, Accredited by NBA, Approved by A.I.C.T.E, Affiliated To Acharya Nagarjuna University)

Guntur-522034

2022-2023

CHALAPATHI INSTITUTE OF ENGINEERING AND TECHNOLOGY (AUTONOMOUS)

(Accredited by NAAC with 'A' grade, Accredited by NBA, Approved by A.I.C.T.E, Affiliated To Acharya Nagarjuna University)

CHALAPATHI NAGAR, LAM, GUNTUR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project work entitled as **"DESIGN AND IMPLEMENTATION OF HOTEL RESERVATION USING PYTHON"** submitted by **B. SUSMITA, K. SUPRIYA and A. DEEPAK** in partial fulfilment for the award of the Degree of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** is a record of bonafied work carried out under my guidance and supervision.

GUIDE	**HEAD OF THE DEPARTMENT**
Mr. K. MADHU KIRAN	Dr K.KIRAN KUMAR,Ph.D
Assistant Professor	Vice Principal and HOD

ACKNOWLEDGEMENT

We express my sincere thanks to our beloved Chairman **Sri. Y.V. ANJANEYULU** for providing support and stimulating environment for developing the project.

We express deep sense of reverence and profound gratitude to **Dr. M. CHANDRA SEKHAR, Ph.D,** Principal for providing me the great support in carrying out the project.

It plunges us in exhilaration in taking privilege in expressing our heartfelt gratitude to **Dr. K. KIRAN KUMAR,** Vice-Principal and HOD of Department of CSE for providing every facility and for constant supervision.

We are thankful to our guide **Mr. K. MADHU KIRAN**, Assistant Professor, Department of CSE for his constant encouragement, suggestions, constant supervision, and abundant support throughout our project.

Thanks to all the teaching and non-teaching staff and lab technicians for their support andalso to our Team-mates for their valuable Co-operation.

BY

BHUMIREDDY SUSMITA (Y19CS1211)

KAPA SUPRIYA (Y19CS1271)

AMBATI DEEPAK (Y19CS1205)

CONTENTS

ABSTRACT

The design and implementation of a hotel reservation system is a complex task that requires careful planning and execution. This system allows customers to make reservations for hotel rooms through an online platform, enabling them to choose their preferred dates and room types. The system includes various features such as user registration, room availability check, room selection, and payment processing. The design of the system involves identifying the key functionalities, data requirements, and user interfaces. The implementation of the system involves developing the database schema, coding the user interface, and integrating the various components. Successful implementation of the hotel reservation system can significantly enhance the efficiency of hotel operations, increase customer satisfaction, and ultimately drive revenue growth. The system allows customers to make reservations for hotel rooms through a decentralized platform, where the transaction history is stored on the blockchain network. The system includes various features such as user registration, room availability check, room selection, and payment processing using cryptocurrencies. The design of the system involves identifying the key functionalities, data requirements, and smart contract development. The implementation of the system involves deploying the smart contracts on the Ganache blockchain, coding the user interface, and integrating the various components. Successful implementation of the hotel reservation system using Ganache can significantly enhance the security of the system, increase customer trust, and ultimately drive revenue growth.

CHAPTER-I

INTRODUCTION

Hotel reservation system is a software application that allows users to book hotel rooms and manage reservations online. The system makes it easy for customers to browse available rooms, select dates and make reservations from the comfort of their own homes. In this project, we will design and implement a hotel reservation system using the Python programming language.

The system will consist of two main components: a user interface and a database. The user interface will allow customers to search for available rooms based on various criteria such as room type, price, location and dates. Once the customer has found a suitable room, they can make a reservation by providing their personal information and payment details.

The database will store information about the hotel rooms, their availability, pricing, and customer reservations. We will use SQLite, a lightweight database management system, to store and manage this data. We will also use various Python libraries such as Tkinter for the user interface and Pandas for data management.

Our goal is to create a system that is easy to use, fast, and reliable. We will achieve this by designing the system with a focus on user experience, using clean code practices, and thoroughly testing the system before release. Overall, this project will provide a valuable learning experience for anyone interested in software development using Python.

Design and implementation of hotel reservation using Ganache involves creating a decentralized application (DApp) for managing hotel reservations on a blockchain network. Ganache is a local blockchain network emulator that allows developers to test and deploy smart contracts before deploying them on a live blockchain network.

The DApp will allow customers to search and book hotel rooms, while the hotel owners can manage their inventory, pricing, and availability. The application will utilize the Ethereum blockchain network, which allows for secure and transparent transactions, as well as smart contracts for automated execution of business logic.

The first step in designing the application will be to create the smart contracts for hotel rooms, bookings, and payments. These smart contracts will define the logic for creating and managing hotel rooms, booking reservations, and handling payments. Once the smart contracts are created, they will be deployed to the Ganache blockchain network for testing.

The DApp will be developed using web3.js, a JavaScript library that allows interaction with the Ethereum network. The user interface will be developed using HTML, CSS, and JavaScript, and will be integrated with the smart contracts via web3.js. The UI will allow customers to search for hotel rooms, view room details, and make reservations. Hotel owners will be able to manage their inventory, set pricing, and view bookings.

Once the DApp is developed and tested on the Ganache network, it will be deployed to the Ethereum mainnet for live use. Customers will be able to book hotel rooms using cryptocurrency, and payments will be automatically processed via the smart contracts.

In summary, the design and implementation of hotel reservation using Ganache involves creating a DApp that utilizes smart contracts on the Ethereum blockchain network for secure and transparent hotel room reservations and payments. Ganache is used as a testing environment for smart contracts before deployment to the live blockchain network.

Hotel reservation systems are essential tools for hotel management and booking operations. These systems help hotel staff to manage room inventory, pricing, availability, and reservations more efficiently. Python is a popular programming language used for developing web applications and software systems. In this project, we will design and implement a hotel reservation system using Python programming language.

The hotel reservation system will allow users to search for available rooms, view room details, make reservations, and cancel reservations. The system will also allow hotel staff to manage room inventory, view booking details, and update room availability. The application will be built using a web-based user interface, making it accessible from anywhere with an internet connection.

The system will be designed using the Model-View-Controller (MVC) architecture. The Model will manage data storage and retrieval, including room availability, booking details, and user information. The View will be responsible for rendering the user interface, allowing users to interact with the system. The Controller will act as an intermediary between the Model and View, handling user requests and updating the system accordingly.

To implement the hotel reservation system, we will use several Python libraries, including Flask for building the web application, SQLAlchemy for database management, and Jinja2 for rendering HTML templates. We will also use Bootstrap for styling the user interface and integrating responsive design.

In summary, the hotel reservation system using Python will be a web-based application that allows users to search for available rooms, view room details, make reservations, and cancel reservations. The system will be built using the MVC architecture and will use several Python libraries for web application development.

What is a blockchain?

A blockchain is a decentralized application comprising data records or “blocks” that cannot be modified by a single actor. In the blockchain network, each block in the chain contains many transaction records. When new transactions are added to the blockchain, those transactions are recorded automatically once the block as a whole is verified and added to the chain.

The recorded transactions are then distributed to every participant’s ledger. That’s why if someone were to change one block in a chain, it would become immediately obvious that the system was compromised. It’s highly infeasible for malicious parties to corrupt a blockchain network. In theory, feasible attacks such as the 51% can be carried out, where a majority of the network is owned and commanded by malicious actors. Running such an attack operation would become increasingly costly very quickly, however, since the cost would increase with every block that is added. Because the malicious actors would own the majority of the network, attacking it would mean acting against its own interests.

Why should hotel reservation system use blockchain technology?

There are several potential benefits to using blockchain technology for a hotel reservation system, including:

1. Enhanced security: Blockchain technology is known for its high level of security due to its decentralized nature and use of cryptography. By using blockchain, hotel reservation systems can offer a more secure environment for storing and sharing sensitive data such as user information, booking details, and payment information.

2. Improved transparency: Blockchain technology offers a transparent and tamper-proof system for storing and sharing data. This means that all transactions and changes to the

reservation system can be easily tracked and verified, reducing the risk of fraud and errors.

3. Reduced intermediaries: A blockchain-based reservation system can eliminate the need for intermediaries such as travel agencies or online booking platforms. This can help hotels reduce costs and increase profits by allowing them to have direct relationships with their customers.

4. Faster transactions: Blockchain technology enables near-instantaneous transactions, which can improve the speed and efficiency of the reservation process. This can result in a better customer experience and increased customer satisfaction.

5. Smart contracts: Blockchain technology allows for the use of smart contracts, which are self-executing contracts with the terms of the agreement written into code. This can automate many aspects of the reservation process, such as payment processing and room availability, reducing the need for human intervention.

Overall, while blockchain technology is still relatively new and developing, it offers several potential benefits for hotel reservation systems, including increased security, transparency, and efficiency.

Yes, blockchain technology uses cryptography to secure and authenticate transactions. Cryptography is the practice of using mathematical algorithms to encrypt data and ensure its confidentiality, integrity, and authenticity. In a blockchain, each block contains a cryptographic hash of the previous block, which creates a chain of blocks that are linked together in a tamper-proof manner.

Cryptography is used in blockchain technology to:

1. Secure transactions: Transactions are encrypted using cryptographic algorithms to ensure that they cannot be altered or tampered with by unauthorized users.

2. Authenticate users: Cryptography is used to create digital signatures that authenticate the identity of users and ensure that only authorized users can access and make changes to the blockchain.

3. Protect privacy: Cryptography can be used to encrypt user data and ensure that it remains confidential.

4. Ensure immutability: Cryptography is used to create a digital fingerprint or hash of each block, which is unique and cannot be changed without invalidating the entire blockchain.

Overall, cryptography plays a crucial role in the security and authenticity of blockchain technology, making it a highly secure and trustworthy system for storing and sharing data.

SYSTEM SETUP

REQUIREMENTS

- Python3 Interpreter
- Visual Studio Code
- Visual Studio Code with Python Extension
- Ganache
- Truffle Suite
- Visual Studio C++ Tools
- Solidity
- web3.js

INSTALLING REQUIREMENTS

- Installing Python on Windows
- Installing VS Code on Windows
- Installing Python Extension on Windows
- Installing git on Windows
- Installing Ganache on Windows
- Installing C++ Tools with Visual Studio

PROJECT SETUP

- Installing web3
- Installing truffle
- Installing Flask

EXISTING SYSTEM

The existing system of hotel reservation can vary widely depending on the specific hotel or hospitality business. However, many hotels use some type of property management system (PMS) to manage their reservations, room inventory, and customer information.

These PMSs often integrate with online booking platforms such as Expedia, Booking.com, or Airbnb to manage reservations made through these platforms. Hotel staff can view and manage room availability, rates, and bookings through a centralized dashboard. Guests can also make reservations directly through the hotel's website or by calling the hotel.

In addition to the PMS, hotels may also use other software systems such as customer relationship management (CRM) software or revenue management software to manage their operations and maximize their profits.

Some hotels also use loyalty programs to incentivize customers to make repeat bookings or to offer special deals and discounts. These loyalty programs often integrate with the hotel's reservation system to track customer activity and offer personalized promotions.

Overall, the existing system of hotel reservation can be complex and varied depending on the specific hotel or hospitality business. However, most hotels use some type of software system to manage their reservations and operations, and many also offer loyalty programs or partnerships with online booking platforms to attract customers.

There are various hotel reservation systems used in different countries around the world, some of which are:

1. Amadeus: Amadeus is a popular hotel reservation system used in many countries around the world, including the United States, Europe, Asia, and South America. It offers a wide range of features such as online booking, room inventory management, and pricing optimization.
2. Sabre: Sabre is another widely used hotel reservation system, primarily used in the United States, but also in other countries such as Canada, Mexico, and the Caribbean. It offers a range of features such as online booking, revenue management, and customer relationship management.

3. Galileo: Galileo is a hotel reservation system used primarily in Europe, Asia, and Africa. It offers features such as online booking, travel agent management, and booking management.
4. Booking.com: Booking.com is an online booking platform used globally that offers a wide range of accommodation options, including hotels, apartments, and vacation rentals. It has a user-friendly interface and offers a range of features such as price comparison, customer reviews, and instant booking.
5. Ctrip: Ctrip is a hotel reservation system primarily used in China, offering a wide range of travel services including hotel reservations, flights, and vacation packages. It offers a user-friendly interface and a range of features such as customer reviews and price comparison.

These are just a few examples of the various hotel reservation systems used in different countries around the world. The specific systems used by hotels may depend on factors such as location, target customer base, and available resources.

PROPOSED SYSTEM

A proposed system of hotel reservation using blockchain in India could offer several benefits to both customers and hotel businesses. Here is a basic outline of a proposed system:

1. User interface: The system would have a user-friendly interface for customers to search for available rooms, view pricing and amenities, and make reservations. The interface would also allow customers to view their booking history and make changes to existing reservations.
2. Blockchain technology: The system would utilize blockchain technology to ensure secure and transparent transactions. This would prevent any unauthorized changes to booking data, provide real-time updates to all parties involved, and improve the security of customer data.
3. Smart contracts: Smart contracts could be used to automate various processes, such as payment processing and room inventory management. This would enable faster and more efficient booking processes, reduce errors, and improve the overall customer experience.

4. Loyalty programs: Blockchain-based loyalty programs could be used to incentivize customers to make repeat bookings. These programs could use smart contracts to track customer activity and offer personalized promotions.
5. Reporting and analytics: The system would provide real-time reporting and analytics on hotel occupancy, revenue, and other key metrics. This would allow hotels to make data-driven decisions to optimize their operations and increase profitability.
6. Integration with other systems: The system would integrate with other systems such as property management systems (PMS), customer relationship management (CRM) software, and revenue management software to manage hotel operations.
7. Security: The system would have strong security features, such as encryption and multi-factor authentication, to protect customer data and prevent fraud.

Overall, a proposed system of hotel reservation using blockchain in India would offer a more secure, efficient, and transparent booking process for customers while also enabling hotels to manage their operations and improve profitability.

CHAPTER-II

METHODOLOGY

2.1. FLOW

The flow of hotel reservation using blockchain could include the following steps:

1. Search for available rooms: The customer would search for available rooms using the hotel reservation platform's user interface.
2. Select a room and make a reservation: Once the customer finds a suitable room, they would select it and make a reservation. At this point, a smart contract would be created, detailing the terms of the reservation and the payment amount.
3. Payment processing: The customer would make a payment using a cryptocurrency or traditional payment method, which would be processed securely using blockchain technology.
4. Verification and confirmation: Once the payment is processed, the blockchain network would verify the transaction and confirm the reservation details.
5. Room inventory management: The hotel staff would have access to a central database of all available rooms and their availability. This would allow them to manage room inventory and avoid overbooking or double booking.
6. Smart contract execution: As the reservation date approaches, the smart contract would automatically execute, releasing the payment to the hotel and updating the room availability status.
7. Customer check-in: When the customer arrives at the hotel, they would check in as usual, and the hotel staff would confirm the reservation details using the blockchain platform.
8. Check-out and payment release: When the customer checks out, the hotel staff would confirm that the room was left in good condition and that there were no extra charges. Once confirmed, the smart contract would automatically release the payment to the hotel.

9. Reporting and analytics: The blockchain platform would provide real-time reporting and analytics on hotel occupancy, revenue, and other key metrics. This would allow hotels to make data-driven decisions to optimize their operations and increase profitability.

Overall, using blockchain technology for hotel reservation would provide a secure, transparent, and efficient booking process for customers while also improving the operational efficiency and profitability of hotels.

2.2 BLOCKCHAIN

Blockchain is a distributed digital ledger technology that allows for secure and transparent transactions and record-keeping. It consists of a chain of blocks that contain information about transactions, with each block containing a unique cryptographic code that links it to the previous block in the chain.

2.2.1 How does blockchain work ?

There are three basic components of a blockchain: blocks, miners, and nodes.

Block: Every blockchain is made up of several blocks, and each block includes data, which is a record of transactions. The crucial point is that the chain is not owned by a single individual or organization.

Miners: Miners are assigned with adding new blocks to the chain via a process known as mining. Miners must solve complicated mathematical problems in order to add data to the block. When a block is efficiently mined, the miner is financially rewarded.

Nodes: A node connects each block to another block, forming the chain. A node is basically that keeps the copies of the ledger and keeps the network operational.

2.2.2 How does blockchain provides security?

Most blockchains arrange data into blocks, with each block containing a transaction or set of transactions. Each new block in a cryptographic chain connects to all the blocks before it in such a way that it is very hard to tamper with. Consensus processes ensure that each transaction within the block is truthful and accurate by validating and agreeing on all transactions within the block. Blockchain technology provides decentralization by allowing members of a distributed network to participate. There seems to be no single point of failure, and an individual user cannot manipulate the transaction record.

2.2.3 How is immutability achieved by blockchain?

A hash value is a unique value that identifies a single block. As hash values are determined by the content of each block, each block is uniquely identifiable by its own hash value. As a result, each block can refer to or point to the one before it, such that the fourth block refers to the third, which refers to the second, and so on. As a result, the hash value serves as a reference.

For example, in our system, Admin (Government) sends the funds requested by the user. After that, the Government determines whether the transaction being conducted is legal. As the iterations continue, a chain builds, demonstrating transaction transparency. Lastly, the fund is transferred from Admin to the User, and the transaction gets completed. The immutable feature of blockchain is seen in the previous example, making it immaculate. It is immutable due to the combination of validations given by the blockchain hashing procedure and cryptography.

2.2.4 How will a transaction enter into the blockchain?

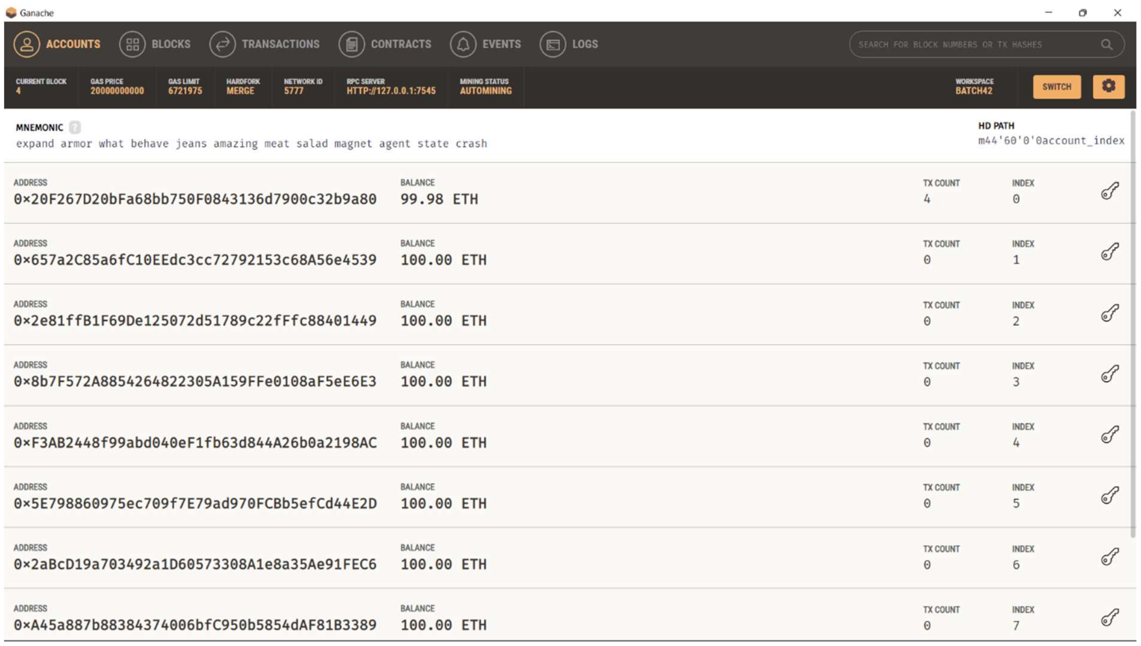
It is necessary to approve and validate a transaction before it can be added to the blockchain. Before a transaction can be added to the blockchain, it must go through several important phases. Now, we'll take a look at cryptographic key authentication, and proof of work authorization protocols in subsequent blockchain networks.

2.2.5 Authentication: Although the original blockchain was supposed to function without a central authority, transactions must still be validated. Accessing a person's "account" or "wallet" of value requires the identification of cryptographic keys. A string of data (similar to a password) serves as a cryptographic key. Using a secured digital identity created by these two keys - a private one that is visible only to a user, and a public one that everyone can see - users can authenticate themselves via signatures and 'unlock' transactions.

2.2.6 Authorization: After the users have agreed on the transaction, it must be authorized, before it can be added to a chain of blocks. The decision to bring a transaction to the chain on a public blockchain is decided by consensus. The majority of "nodes" (or computers) must accept the transaction for it to be valid. The people who own machines in the network are rewarded for confirming transactions. This method is referred to as 'proof of work'.

2.2.7 Understanding Libra: Learn how Facebook used certain parts of blockchain technology to establish Libra, a new cryptocurrency that has the potential to impact the banking and finance industries.

2.2.8 Proof of Work: To add a block to the chain, Proof of Work asks the individuals who own the machines in the network to solve a difficult mathematical problem. Mining is the process of resolving an issue, and 'miners' are generally rewarded in bitcoin. However, mining is a difficult task. The mathematical problem could only be resolved via trial and error, with a 1 in 5.9 trillion chance of succeeding. The process requires significant amounts of computational power, and this consumes considerable amounts of energy. This means that the benefits of mining must surpass the expense of the computers and the electricity used to power them because a single computer would take years to solve a mathematical problem.



CURRENT BLOCK 4 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MERGE | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE BATCH42

MNEMONIC
expand armor what behave jeans amazing meat salad magnet agent state crash

HD PATH
m44'60'0'0account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×20F267D20bFa68bb750F0843136d7900c32b9a80	99.98 ETH	4	0
0×657a2C85a6fC10EEdc3cc72792153c68A56e4539	100.00 ETH	0	1
0×2e81ffB1F69De125072d51789c22fFfc88401449	100.00 ETH	0	2
0×8b7F572A8854264822305A159FFe0108aF5eE6E3	100.00 ETH	0	3
0×F3AB2448f99abd040eF1fb63d844A26b0a2198AC	100.00 ETH	0	4
0×5E798860975ec709f7E79ad970FCBb5efCd44E2D	100.00 ETH	0	5
0×2aBcD19a703492a1D60573308A1e8a35Ae91FEC6	100.00 ETH	0	6
0×A45a887b88384374006bfC950b5854dAF81B3389	100.00 ETH	0	7

Ganache is a software which is used for blockchain network. Ethereum block chain is used in this project as it is a decentralized blockchain platform that establishes a peer-to-peer network that securely executes and verifies application code, called smart contracts. Smart contracts allow participants to transact with each other without a trusted central authority. Ethereum is also a bitcoin which worth nearly 1 Ethereum is equal to Rs.1,15,224.51 at present it is used for non-financial transactions also. In the above picture the interface of the Ganache blockchain is shown firstly they provide 100.00 ETH when we do any transaction the amount will be debited from the account. The address is used to identify the user. The data is stored in the blockchain in the form of blocks that to the data is encrypted using asymmetric key cryptography. So, no one can access the data without the access key.

2.3 SMART CONTRACTS

A smart contract is a self-executing contract with the terms of the agreement directly written into computer code. These contracts are built on top of blockchain technology, which allows them to be executed automatically without the need for intermediaries such as lawyers or banks.

Smart contracts are transparent, secure, and immutable, meaning that once they are deployed on a blockchain network, they cannot be altered. They enable the automation of business processes and transactions, as well as the creation of decentralized applications (dApps) and decentralized autonomous organizations (DAOs).

Some popular blockchain platforms that support smart contracts include Ethereum, Binance Smart Chain, Polkadot, and Cardano.

Smart contracts are code written into a blockchain that executes the terms of an agreement or contract from outside the chain. It automates the actions that would otherwise be completed by the parties in the agreement, which removes the need for both parties to trust each other.

Solidity programming is used to write the smart contract with the extinction .sol is is a object oriented programming language same as C++, Java.

First step is developing smart contract by using solidity as per our requirements after testing the smart contract and then migrating the smart contract.

2.3.1 SOLIDITY FILES

Solidity is a high-level programming language used for writing smart contracts on the Ethereum blockchain. Smart contracts are self-executing contracts with the terms of the agreement between buyer and seller being directly written into code. They are stored on a decentralized blockchain network, which makes them tamper-proof and transparent.

Solidity is an object-oriented language that is similar to JavaScript in syntax. It supports a wide range of features, including inheritance, libraries, and user-defined types. Solidity is compiled to bytecode, which is then executed by the Ethereum Virtual Machine (EVM).

Some of the applications of Solidity include creating tokens, managing digital assets, and creating decentralized applications (dApps). Solidity is one of the most popular programming languages for smart contracts on the Ethereum blockchain and has a large developer community.

In this project I have used two solidity files funds. Sol and register. Sol these are the two files used to build contract with blockchain in my project.

2.3.1.1 register. Sol

This register. Sol file is used to store all the information regarding registration of user and and that information is used while logging in with our username and password, if the information is not stored we cannot perform any activity so the management cannot allocate any rooms to the customers.

This file establishes the connection between registration page and Ethereum blockchain so that information is stored in the form of blocks. Here accounts are created by using address in the Ethereum blockchain account.

The data is stored in the encrypted format so that no one can access the data. This blockchain network asymmetric key cryptography to encrypt the data so without public key or private key they can access the data.

Source Code:

```
// SPDX-License-Identifier: MIT

pragma solidity >=0.4.22 <0.9.0;

contract register {

 address admin;

 uint password;

  address[] _customers;

 string[] _names;

 string[] _emails;

 string[] _mobiles;
 uint[] _passwords;

 mapping(address=>bool) users;

 constructor() public {

   admin=msg.sender;

   password=1234;

 }

 function loginadmin(address a,uint p) public view returns(bool) {

   if(a==admin && p==password){

     return true;

   }

   else{

     return false;

   }

 }

 function registeruser(address customer,string memory name,string memory email,string
memory mobile,uint password1) public {

     require(!users[customer]);

     users[customer]=true;

     _customers.push(customer);

     _names.push(name);

     _emails.push(email);

     _mobiles.push(mobile);

     _passwords.push(password1);

 }
 function viewusers() public view returns(address[] memory,string[] memory,string[]
memory,string[] memory,uint[] memory){
   return(_customers,_names,_emails,_mobiles,_passwords);
 }
 function loginuser(address username,uint password2) public view returns(bool){
   uint i;
   for(i=0;i<_customers.length;i++){
     if(username==_customers[i] && _passwords[i]==password2){
       return true;
     }
   }
   return false;
 }
}
```

2.3.1.2 rooms. Sol

The rooms. Sol file is used to store all the remaining information like room details, Allocated rooms, room id requests, view requests, view room status, room vacated details, customer information and remaining activities you performed in the application except registration details.

This file establishes the smart connection with blockchain to store the data in Ethereum blockchain at our work space and that data is stored in the form of blocks. Here rooms are allocated to the customers after creating the account to the admin using the address in the Ethereum blockchain.

The data is stored in the encrypted format so that no one can access the data. This blockchain network asymmetric key cryptography to encrypt the data so without public key or private key they can access the data.

Source Code:

```
// SPDX-License-Identifier: MIT

pragma solidity >=0.4.22 <0.9.0;

contract rooms {

  uint reqcount=0;

  address[] _customers;

  string[] _aadhars;

  string[] _city;

  uint[] _noofrooms;

  uint[] _noofdays;

  string[] _dates;

  string[] _noofadults;

  uint[][] _roomids;

  uint[] _roomreq;

  uint totalrooms=10;

  uint[] status=[0,0,0,0,0,0,0,0,0,0];

    function roomrequest(address customer,string memory aadhar,string memory city,uint noofrooms,uint noofdays,string memory date,string memory noofadults) public{

  reqcount+=1;

    _roomreq.push(reqcount);

    _customers.push(customer);

    _aadhars.push(aadhar);

    _city.push(city);

    _noofrooms.push(noofrooms);
    _noofdays.push(noofdays);

    _dates.push(date);

    _noofadults.push(noofadults);

    _roomids.push([0]);

  }


  function viewrequests() public view returns(uint[] memory,address[] memory,string[] memory,string[] memory,uint[] memory,uint[] memory,string[] memory,string[] memory,uint[][] memory){


return(_roomreq,_customers,_aadhars,_city,_noofrooms,_noofdays,_dates,_noofadults,_roomids);

  }

  function allocateroom(uint roomreq,uint[] memory id) public{

    uint i;

    uint j;

    for(i=0;i<_roomreq.length;i++) {

      if(roomreq==_roomreq[i]) {

        _roomids[i]=id;

      }

    }

    for(j=0;j<id.length;j++){

      status[id[j]-1]=1;

    }

  }
```

```
 function viewroomstatus() public view returns(uint[] memory){
  return(status);
 }
 function vacateroom(uint roomreq) public{
  uint i;
  uint j;
  uint[] memory id;

  for(i=0;i<_roomreq.length;i++){
    if(roomreq==_roomreq[i]){
        id=_roomids[i];
        _roomids[i]=[100];
    }
  }
  for(j=0;j<id.length;j++){
    status[id[j]-1]=0;
  }
 }
}
```

2.4 DAPP

A DApp (decentralized application) is a type of software application that runs on a decentralized blockchain network. DApps are designed to be transparent, secure, and resistant to censorship or manipulation. They are decentralized because they run on a blockchain network, which means that they are not controlled by a central authority or third party.

In a DApp blockchain, the application logic is stored on the blockchain and executed by the nodes in the network. The blockchain serves as a distributed ledger that records all transactions and data in a tamper-proof way. The decentralized nature of the blockchain ensures that no single entity has complete control over the network, making it resistant to attacks and manipulation.

DApps can be used for a variety of purposes, including decentralized finance (DeFi), gaming, social networks, and more. Some popular DApps include Uniswap, Compound, and CryptoKitties.

Developing a DApp requires expertise in blockchain technology and programming languages such as Solidity (for Ethereum) or Rust (for Polkadot). DApp development involves creating smart contracts, which are self-executing contracts that automate the process of verifying and executing transactions on the blockchain.

2.5 WEB3

Web3, also known as Web3.0 or the decentralized web, refers to the next generation of the internet that aims to give users greater control over their data and online identities. It is a term used to describe a set of emerging technologies and protocols that are built on top of blockchain, the distributed ledger technology that underpins cryptocurrencies like Bitcoin and Ethereum.

Web3 technology enables the creation of decentralized applications (dApps) that run on a decentralized network of computers rather than a centralized server. This means that users can interact with applications without relying on a central authority to store and manage their data.

Web3 technologies include blockchain protocols such as Ethereum, IPFS (InterPlanetary File System) for decentralized storage, and smart contracts, which are self-executing contracts with the terms of the agreement between buyer and seller being directly written into lines of code.

These technologies enable the creation of decentralized applications that are more secure, transparent, and resistant to censorship than traditional web applications.

```
>>>installing web3

>>>open command prompt

>>>pip install web3
```

Web3 will be installed into your pc.

2.5.1 web3.py

Web3.py is a Python library for interacting with Ethereum blockchain. It allows you to connect to an Ethereum node and perform various operations, such as reading and writing data to the blockchain, deploying smart contracts, and sending transactions.

Here's a simple example of how to use Web3.py to read the balance of an Ethereum account:

```
from web3 import Web3


# Connect to a local Ethereum node
w3 = Web3(Web3.HTTPProvider('http://localhost:8545'))
# Get the balance of an Ethereum account
balance = w3.eth.get_balance('0x1234567890123456789012345678901234567890')
# Print the balance in wei
print(f"Balance: {balance}")
```

In this example, we first connect to a local Ethereum node using the **Web3.HTTPProvider** class. Then, we use the **w3.eth.get_balance** method to retrieve the balance of an Ethereum account (replace the account address with a real one). Finally, we print the balance in wei.

Web3.py also provides many other features for working with Ethereum, such as interacting with smart contracts and listening for events.

2.5.2 WEB3.JS

web3.js is a JavaScript library that provides a way to interact with the Ethereum blockchain through a web browser or a Node.js environment. It is a part of the larger web3.js ecosystem, which includes other libraries and tools that help developers build decentralized applications (dApps) on the Ethereum network.

With web3.js, developers can write code to:

- Query information from the blockchain, such as account balances, transaction details, and contract data.
- Send transactions to the blockchain, such as transferring Ether or executing smart contract functions.
- Listen for events emitted by the blockchain, such as when a new block is mined or when a smart contract emits an event.

Web3.js supports both the traditional, proof-of-work Ethereum blockchain as well as newer, proof-of-stake networks like Ethereum 2.0. It also has a modular architecture that allows developers to easily extend its functionality or use only the parts that they need.

Overall, web3.js is an essential tool for building decentralized applications on the Ethereum network.

2.6 TRUFFLE SUITE

The Truffle Suite is a development framework for Ethereum-based decentralized applications (dapps). It provides a suite of tools that make it easier for developers to build, test, and deploy smart contracts on the Ethereum blockchain.

The Truffle Suite includes the following components:

1. Truffle Framework: A development framework that provides tools for writing, compiling, and deploying smart contracts on the Ethereum blockchain.
2. Ganache: A personal blockchain for Ethereum development that allows developers to test their smart contracts in a simulated environment.

3. Drizzle: A collection of front-end libraries for building dapps that interact with smart contracts.
4. Truffle Boxes: Pre-built project templates that provide a starting point for building dapps.
5. Truffle Teams: A collaboration and project management platform for teams working on dapp development.

Overall, the Truffle Suite is a comprehensive set of tools that simplifies the development process for Ethereum-based dapps and allows developers to focus on building innovative applications.

To install the Truffle suite, follow these steps:

1. Install Node.js: Truffle is built using Node.js, so you'll need to install it first if you haven't already. You can download and install Node.js from the official website: https://nodejs.org/.
2. Install Truffle: Once you have Node.js installed, open up a terminal (or command prompt on Windows) and run the following command:

```
>>> npm install -g truffle
```

This will install Truffle globally on your machine, which means you can use it from anywhere.

3. Verify the installation: After the installation is complete, run the following command to verify that Truffle is installed correctly:

```
>>> truffle version
```

This should display the version number of Truffle that you just installed.

That's it! You're now ready to start using the Truffle suite for Ethereum development.

CHAPTER-III

FRAMEWORK

3.1 FRAMEWORK TEMPLATES

In computer science and software development, a framework is a pre-designed and pre-built structure or set of software tools and libraries that provides developers with a foundation on which to build applications. Frameworks are used to standardize the development process, promote code reuse, and simplify application development.

Frameworks can be general-purpose, such as web application frameworks like Django, Ruby on Rails, or Flask, or they can be specialized for specific industries or applications, such as machine learning frameworks like TensorFlow, PyTorch, or Keras.

A typical framework may provide developers with:

- A set of pre-defined classes and functions for common tasks
- Rules and guidelines for coding practices
- Pre-built templates for creating user interfaces
- Support for database connectivity and data access
- A testing framework for automated testing and debugging
- Security features and guidelines
- Documentation and examples to help developers get started

Frameworks are designed to save time and increase productivity by eliminating the need to start from scratch every time an application is developed. Instead, developers can use the framework's pre-built components and tools to quickly develop applications that meet the requirements of their clients or customers.

3.1.1 LANGUAGES USED IN FRAMEWORK

- HTML (Hypertext Markup Language)
- CSS (Cascading Style Sheet)
- JAVASCRIPT
- BOOTSTRAP

3.1.2 Name of the templates included in this application

These pages are linked in the application using flask framework to python code (app.py). This is the execution file of this application with the help of these templates we are providing input and output is shown.

- Home.html
- Head.html
- Admindashboard.html
- adminlogin.html
- allocateroom.html
- customer.html
- customers.html
- dashboard.html
- head.html
- home.html
- login.html
- main1.html
- registration.html
- requestroom.html
- roomstatus.html
- vacateroom.html

CHAPTER-IV

PYTHON PROGRAMMING

>>>Python programming using flask framework

Python Flask is a micro web framework for building web applications using the Python programming language. Flask is designed to be lightweight, flexible, and easy to use, making it a popular choice for developers who want to create web applications quickly and efficiently.

With Flask, you can create web applications that respond to HTTP requests, render templates, handle database connections, and more. Flask also includes support for testing and debugging your applications, making it easy to catch errors and improve the quality of your code.

One of the key features of Flask is its use of routes to handle incoming requests. Routes are defined using the @app.route() decorator, and can be used to handle requests for specific URLs or URL patterns. For example, a route might handle requests for the root URL ("/"), or for a specific page ("/about").

Flask also supports the use of templates to generate dynamic content for your web pages. Templates are written using Jinja2, a popular templating engine for Python. With Jinja2, you can include variables, conditionals, and loops in your templates, making it easy to generate dynamic content based on user input or database queries.

Overall, Flask is a powerful and flexible web framework that is easy to learn and use. Whether you're building a small web application or a large-scale project, Flask provides the tools you need to create robust and scalable web applications using Python.

4.1.1 Importing Libraries

You can also import specific functions or classes from a library using the from keyword.

To import libraries in Python, you can use the **import** statement followed by the name of the library you want to import.

```
from flask import Flask,render_template,request,redirect,session
from web3 import Web3,HTTPProvider
import json
```

>>> Here we imported flask framework from flask library flask, render_template, request, redirect, session functions are imported.

>>> From web3 we imported web3, HTTPPrivider functions.

HTTPProvider is a class in web3.js library, which is a collection of libraries that allows interaction with an Ethereum node using JavaScript. HTTPProvider is responsible for establishing a connection with an Ethereum node over HTTP protocol.

When creating an instance of HTTPProvider, you need to provide the URL of the Ethereum node that you want to connect to. This URL is typically the endpoint of an Ethereum JSON-RPC API, which allows you to interact with the Ethereum network using JSON-RPC protocol over HTTP.

>>> Json is imported

JSON stands for JavaScript Object Notation. It is a lightweight data interchange format that is easy for humans to read and write and easy for machines to parse and generate. JSON is a text format that is completely language-independent but it is widely used in web applications and APIs for data exchange between the server and client.

JSON syntax is based on a subset of JavaScript's syntax and it consists of two data structures:

1. Key-value pairs: It consists of a key, which is always a string, followed by a colon, and then a value. The value can be a string, number, boolean, array, object, or null.
2. Arrays: It is an ordered list of values, where each value can be a string, number, boolean, array, object, or null.

JSON is widely used in web development for transmitting data between a server and a client or between different systems, such as microservices. It is also used for storing and exchanging data in many other applications.

4.1.3 connect_blockchain_register

>>>Source Code

```
def connect_with_register(acc): #connecting to the register contract
    blockchain='http://127.0.0.1:7545'
    web3=Web3(HTTPProvider(blockchain))
    if acc==0:
        acc=web3.eth.accounts[0]
    web3.eth.defaultAccount=acc
    artifact_path='../build/contracts/register.json' #loading artifact
    contract_address=register_contract_address
    with open(artifact_path) as f:
        contract_json=json.load(f)
        contract_abi=contract_json['abi'] #extracting abi
    contract=web3.eth.contract(address=contract_address,abi=contract_abi) # type: ignore #passing contract
address,abi
    return(contract,web3)
```

This python function is used to build connection with Ethereum blockchain and the application. By using this function, we register in the blockchain after creating workspace in our Ganache blockchain. To do this registration some amount of ETH will be debited from our account at the particular address, Initially we will have 100 ETH for every connection 0.01 ETH will be debited. This process will occur only ones in one workspace.

Truffle suite command is used to build the connection between these two,

>>> In visual studio code run

>>> truffle compile

This is used to compile the truffle in our environment

>>> truffle migrate

This command is used to migrate the blockchain address to our application means establish the connection between these two by using Ethereum for this process some ether will be debited from our account.

4.1.4 connect_blockchain_room

>>> Code

```
def connect_with_rooms(acc): #connecting to the room contract
    blockchain='http://127.0.0.1:7545'
    web3=Web3(HTTPProvider(blockchain))
    if acc==0:
        acc=web3.eth.accounts[0]
    web3.eth.defaultAccount=acc
    artifact_path='../build/contracts/rooms.json' #loading artifact
    contract_address=rooms_contract_address
    with open(artifact_path) as f:
        contract_json=json.load(f)
        contract_abi=contract_json['abi'] #extracting abi
    contract=web3.eth.contract(address=contract_address,abi=contract_abi) # type: ignore #passing contract
address,abi
    return(contract,web3)
```

This function is used to store all the details regarding rooms. This function establishes the connection between rooms and Ethereum block chain, if it does not store this information we cannot grant the room through the request.

4.1.5 All the templates are linked to the Flask application and the python functions are declared for each template.

The working of the application will start from here each function is having the separate work of process to implement based on the user request. Each function and code are explained below with images and examples, detailed explanation of each is described below separately,

Secret Key for Flask App

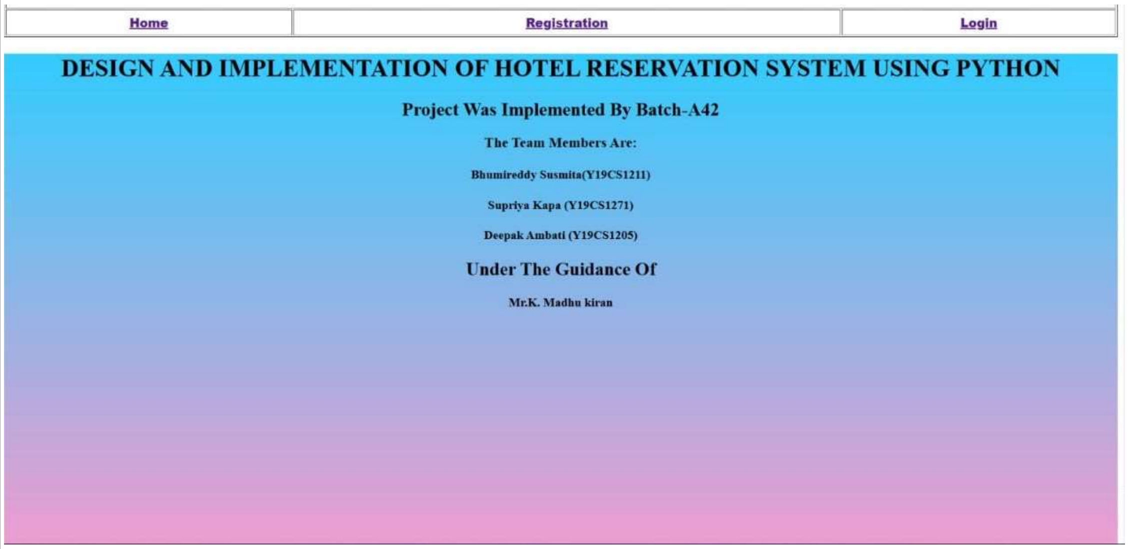
```
app=Flask(__name__)
app.secret_key='batch 42'
```

Home Page

Home page shows all the required information and this is the first page in the application from here we can access all the pages. The user registration and login is also accessed from this page. This is the common page that displays to all the users even for public.

```
@app.route('/')
def homepage():
    return render_template('Home.html')
```

Home page view:



DESIGN AND IMPLEMENTATION OF HOTEL RESERVATION SYSTEM USING PYTHON

Project Was Implemented By Batch-A42

The Team Members Are:

Bhumireddy Susmita(Y19CS1211)

Supriya Kapa (Y19CS1271)

Deepak Ambati (Y19CS1205)

Under The Guidance Of

Mr.K. Madhu kiran

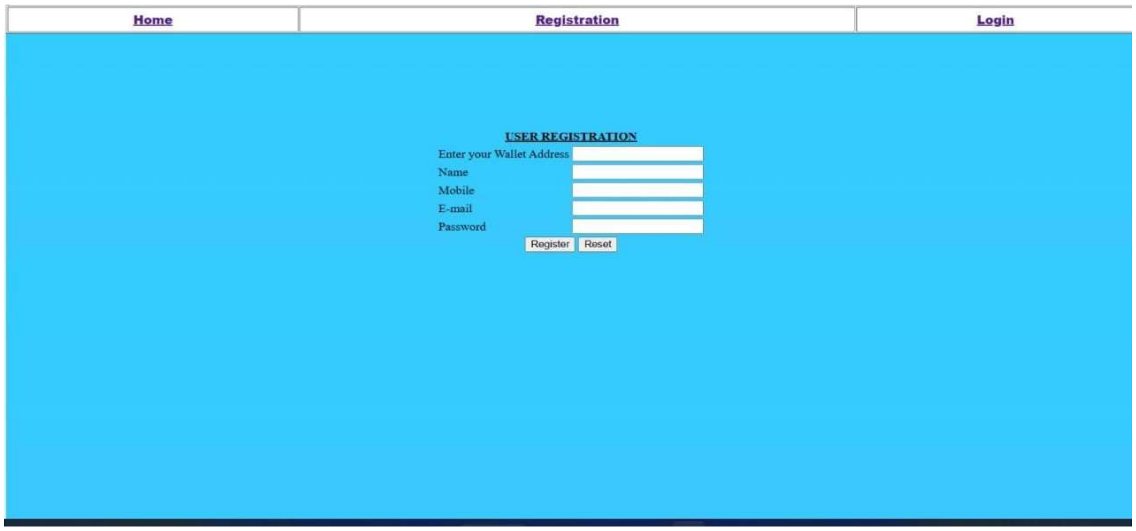
4.1.a.home page

Registration page

This is the registration page of the application from here we should create the account by using the address in the Ethereum blockchain so that all the information is store in the application. Using solidity file (register. Sol) this information is stored in the blockchain.

```
@app.route('/Registration')
def registrationpage():
    return render_template('Registration.html')
```

REGISTRATION PAGE VIEW :



4.1.b.Registration page

Username: It should be the address from the blockchain.

Name: Name can be your name or any other.

Email ID: Mail ID should be given.

Phone Number: Phone Number should of 10 digits.

Password: Password can be of any king but keep it securely.

Register: After filling all the required details please click on the register button to register then it is redirected to the login page.

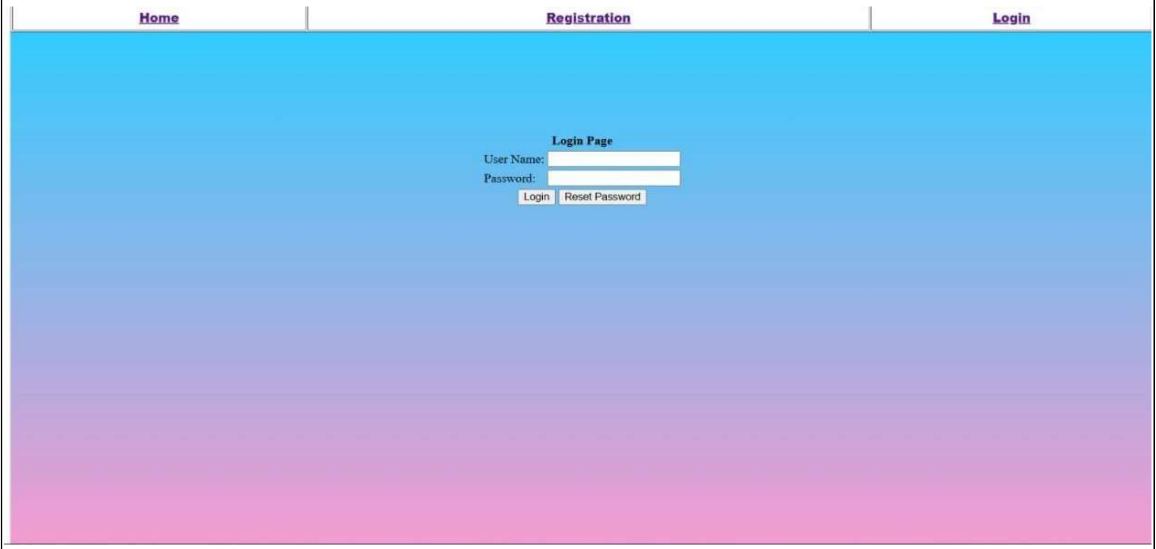
These are the details that should be filled while registration. All the details must be filled correctly. Always the username should be the address, address should be taken from the Ethereum blockchain. Admin can only allocate the room to the customer using address. If you use the address that is already used while registration you will face an error.

User Login Page:

```
@app.route('/Login')
def loginpage():
    return render_template('Login.html')
```

Login Page View:

```
@app.route('/loginuser',methods=['post','get'])
def loginuser():
    username=request.form['username']
    password=request.form['password']
    print(username,password)
    contract,web3=connect_with_register(0) #connect to blockchain
    state=contract.functions.loginuser(username,int(password)).call() # view permission
    print(state)
    if(state==True):
        session['username']=username
        return (redirect('/dashboard'))
    else:
        return(render_template('Login.html',res='Invalid Credentials'))
```



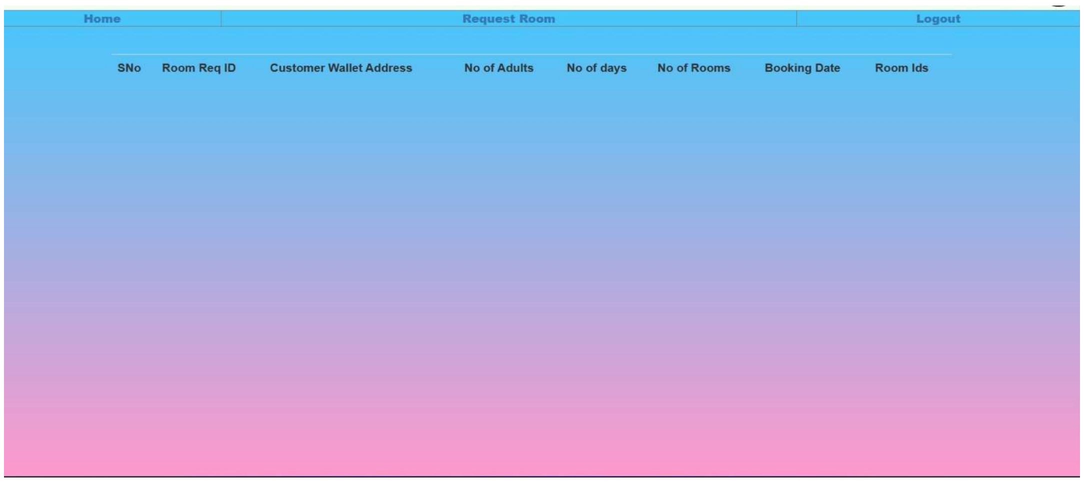
4.1.c.Login page

If you enter incorrect password or if you are not authorized to login then you will be shown an error only senders are having access to login

Dashboard page :

```
@app.route('/dashboard')
def dashboardpage():
   data=[]
   walletaddr=session['username']
   contract,web3=connect_with_rooms(0)
   _roomreq,_customers,_aadhars,_city,_noofrooms,_noofdays,_dates,_noofadults,_roomids=contract.func
tions.viewrequests().call()
   for i in range(0,len(_customers)):
      if(_customers[i]==walletaddr):
         dummy=[]
         dummy.append(_roomreq[i])
         dummy.append(walletaddr)
         dummy.append(_noofadults[i])
         dummy.append(_noofdays[i])
         dummy.append(_noofrooms[i])
         dummy.append(_dates[i])
         dummy.append(_roomids[i])
         data.append(dummy)
   return render_template('dashboard.html',dashboard_data=data,l=len(data))
```

After the completion of user login a dashboard will appeared to the user with the details of room request id, customer wallet address, no of adults, no of days, no of days, no of rooms, booking date and the room ids.



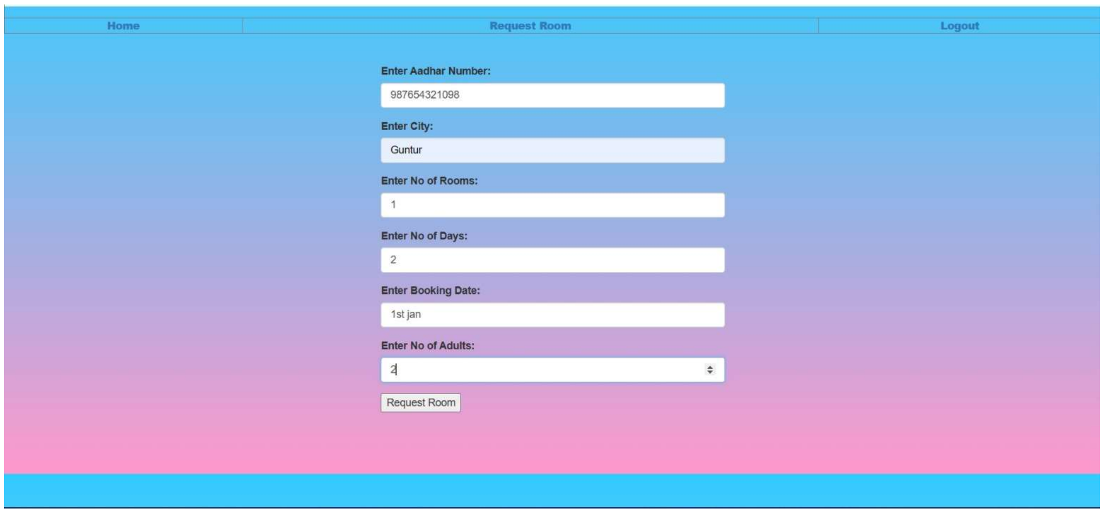
SNo	Room Req ID	Customer Wallet Address	No of Adults	No of days	No of Rooms	Booking Date	Room Ids

4.1.d.Dashboard

Request room page:

```
@app.route('/requestroom')
def requestroompage():
    return render_template('requestroom.html')
```

In this request room page the user will have the details of aadhar number, city, no of rooms, no of days, booking date, no of adults.

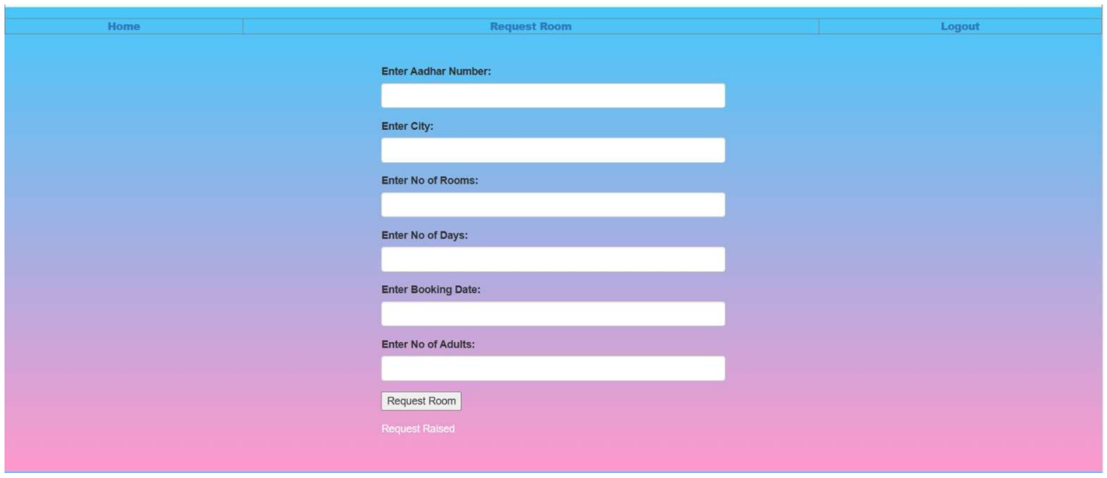


Enter Aadhar Number: 987654321098
Enter City: Guntur
Enter No of Rooms: 1
Enter No of Days: 2
Enter Booking Date: 1st jan
Enter No of Adults: 2

4.1.e.Request room

Request raised:

```
@app.route('/requestroomform',methods=['post','get'])
def requestroomformpage():
    walletaddr=session['username']
    aadhar=request.form['aadhar']
    city=request.form['city']
    noofrooms=request.form['noofrooms']
    noofdays=request.form['noofdays']
    date=request.form['date']
    noofadults=request.form['noofadults']
    print(walletaddr,aadhar,city,noofrooms,noofdays,date,noofadults)
    contract,web3=connect_with_rooms(0)
     hash=contract.functions.roomrequest(walletaddr,aadhar,city,int(noofrooms),int(noofdays),date,noofadul
ts).transact()
    web3.eth.waitForTransactionReceipt(hash) #storing data in the blockchain
    return (render_template('requestroom.html',res='Request Raised'))
```



4.1.f.Request raised

The user needs to fill the application in the request room page. After filling the details the request for room will raised.

User Logout page:

```
@app.route('/logout')
def logoutpage():
    session['username']=None
    return redirect('/')
```

After the request for room raised then the user should logout from the page.

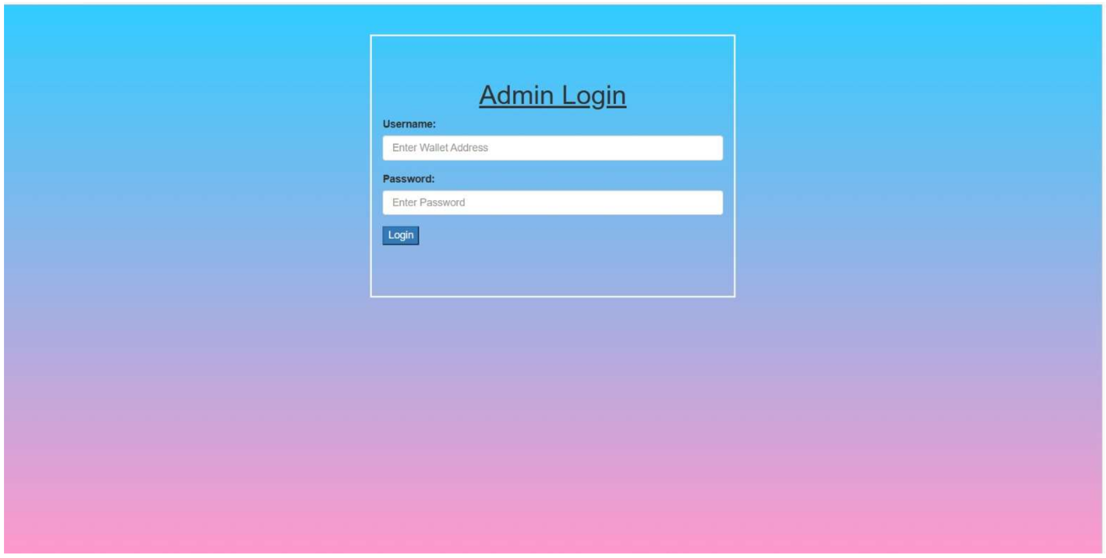
Admin login page:

```
@app.route('/admin')
def adminloginpage():
    return render_template('adminlogin.html')
```

This is the admin login page with the username and password.

Admin login user :

```
@app.route('/adminloginuser',methods=['POST'])
def adminloginuser():
    username=request.form['username']
    password=request.form['password']
    print(username,password)
    contract,web3=connect_with_register(0)
    state=contract.functions.loginadmin(username,int(password)).call()
    if state==True:
        return (redirect('/admindashboard'))
    else:
        return(render_template('adminlogin.html',err='invalid details'))
```

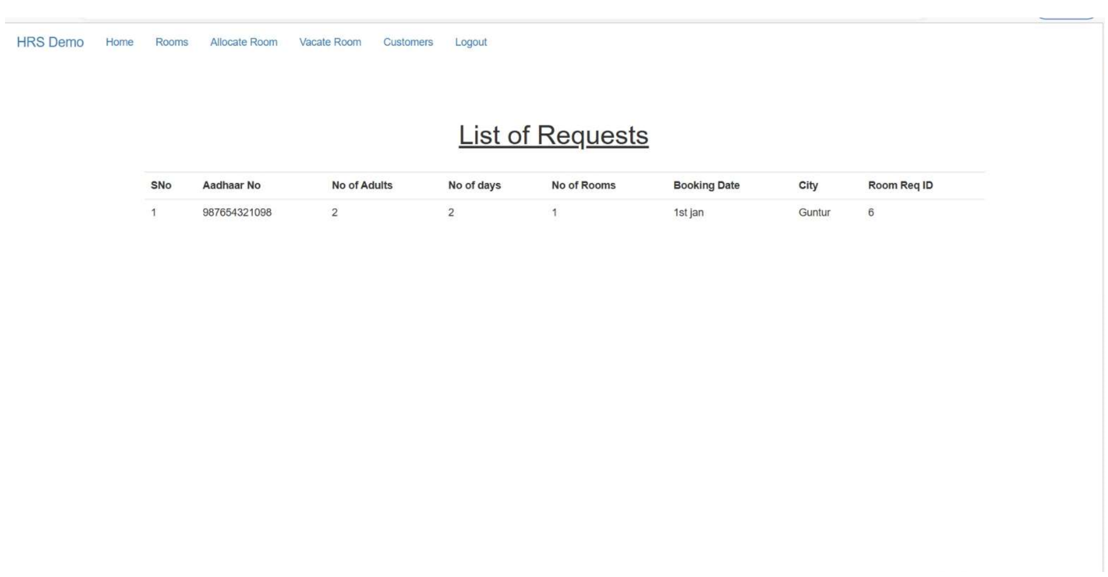


4.1.g.Admin Login

Admin login with username and password. If the username and password are correct then the admin can get the dashboard.

Admin dashboard page :

```
@app.route('/admindashboard') #display all the request raised by the customer
def admindashboard():
    data=[]
    contract,web3=connect_with_rooms(0)
    _roomreq,_customers,_aadhars,_city,_noofrooms,_noofdays,_dates,_noofadults,_roomids=contract.func
tions.viewrequests().call()
    for i in range(0,len(_customers)):
        if(_roomids[i][0]==0):
            dummy=[]
            dummy.append(_roomreq[i])
            dummy.append(_aadhars[i])
            dummy.append(_noofadults[i])
            dummy.append(_noofdays[i])
            dummy.append(_noofrooms[i])
            dummy.append(_dates[i])
            dummy.append(_city[i])
            dummy.append(_customers[i])
            data.append(dummy)
    return render_template('admindashboard.html',dashboard_data=data,l=len(data))
```



List of Requests

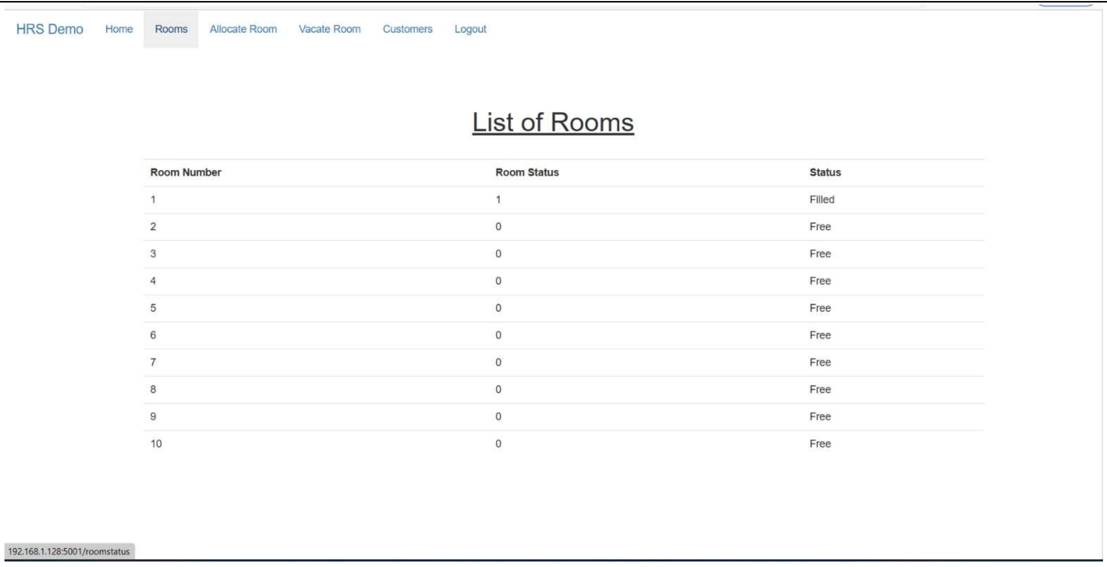
SNo	Aadhaar No	No of Adults	No of days	No of Rooms	Booking Date	City	Room Req ID
1	987654321098	2	2	1	1st jan	Guntur	6

4.1.h.Admin dashboard

This is the admin dashboard page. Admin will get the list of requests like aadhar number, no of adults, no of days, no of rooms, booking date and city with room request id.

Room status page :

```
@app.route('/roomstatus',methods=['get','post']) #display room status
def roomstatus():
    data=[]
    contract,web3=connect_with_rooms(0)
    status=contract.functions.viewroomstatus().call()
    for i in status:
        dummy=[]
        dummy.append(i)
        if i==0:
            dummy.append('Free')
        else:
            dummy.append('Filled')
        data.append(dummy)
    return(render_template('roomstatus.html',dashboard_data=data,l=len(data)))
```



List of Rooms

Room Number	Room Status	Status
1	1	Filled
2	0	Free
3	0	Free
4	0	Free
5	0	Free
6	0	Free
7	0	Free
8	0	Free
9	0	Free
10	0	Free

192.168.1.128:5001/roomstatus

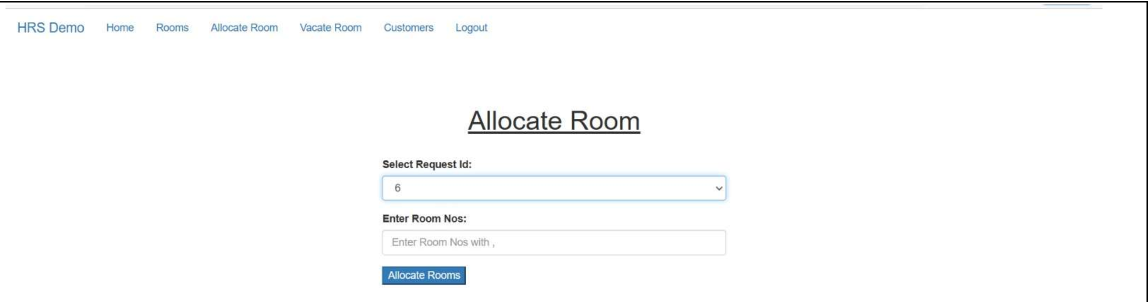
4.1.i.Room status

In this room status page the admin will get to know that the status of the rooms whether the room is free or it is filled.

Allocate room page :

```
@app.route('/allocateroom',methods=['get','post']) #allocating rooms
def allocateroom():
    data=[]
    contract,web3=connect_with_rooms(0)
    _roomreq,_customers,_aadhars,_city,_noofrooms,_noofdays,_dates,_noofadults,_roomids=contract.func
tions.viewrequests().call()
    for i in range(0,len(_roomreq)):
        if _roomids[i][0]==0:
            dummy=[]
            dummy.append(_roomreq[i])
            data.append(dummy)

    return render_template('allocateroom.html',dashboard_data=data,l=len(data))
```



Allocate Room

Select Request Id:

6

Enter Room Nos:

Enter Room Nos with ,

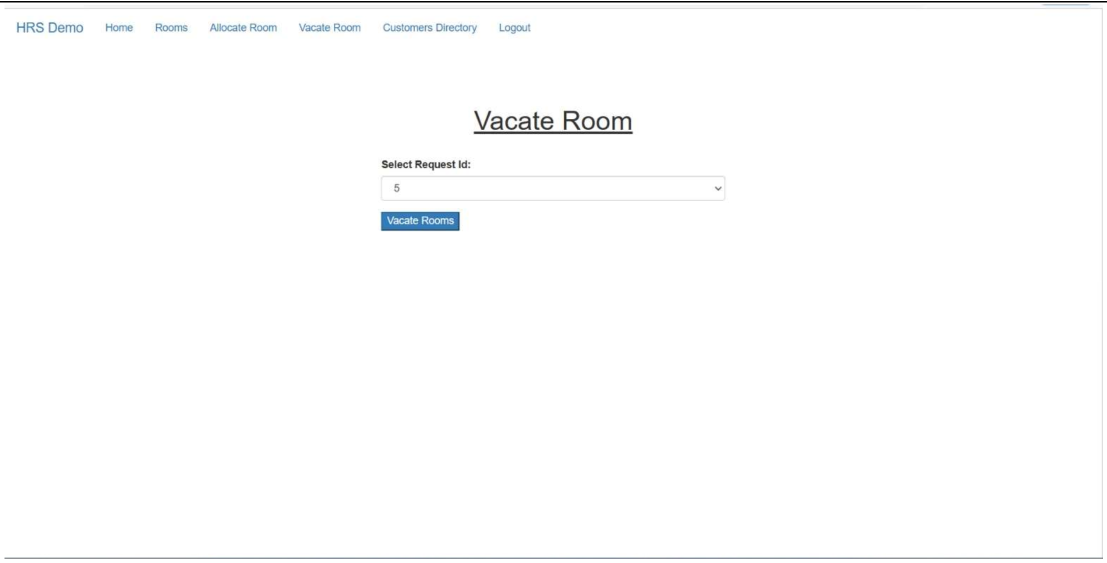
4.1.j.Allocate room

The admin can allocate the room to the customer with the request id and gives the room.

Vacate room page :

```
 @app.route('/vacateroom')

def vacateroom():
   data=[]
   contract,web3=connect_with_rooms(0)
   _roomreq,_customers,_aadhars,_city,_noofrooms,_noofdays,_dates,_noofadults,_roomids=contract.func
tions.viewrequests().call()
   for i in range(len(_roomreq)):
       if(_roomids[i][0]!=0 and _roomids[i][0]!=100):
           dummy=[]
           dummy.append(_roomreq[i])
           data.append(dummy)
   return render_template('vacateroom.html',dashboard_data=data,l=len(data))
```



4.1.k.Vacate room

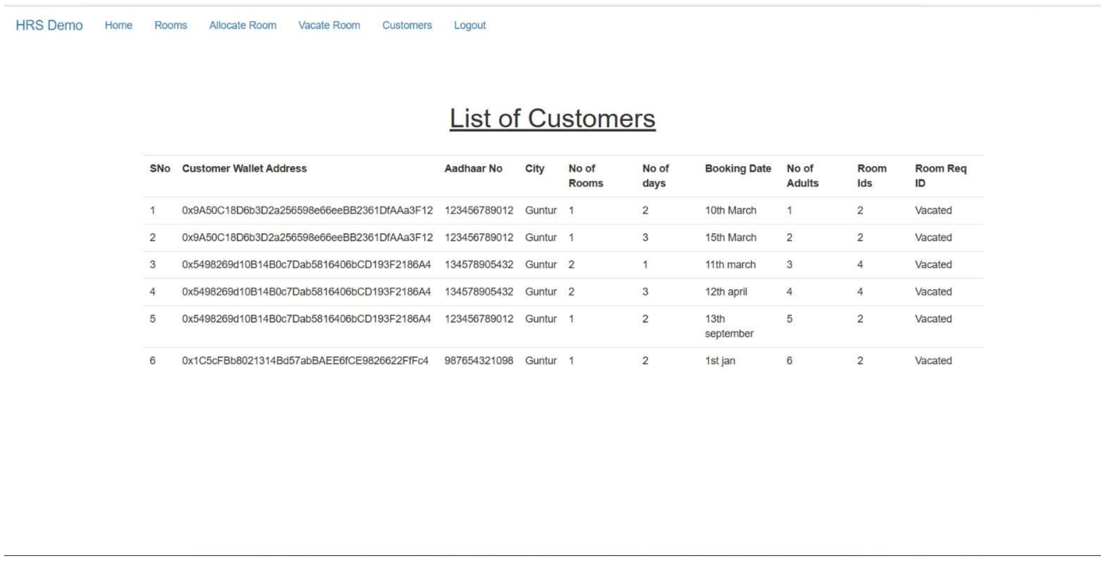
Based on the request id room vacate should be done by the admin.

Customer directory :

```
@app.route('/customersdirectory',methods=['post','get']) #display all customers
def customersdirectory():
   contract,web3=connect_with_rooms(0)
   _roomreq,_customers,_aadhars,_city,_noofrooms,_noofdays,_dates,_noofadults,_roomids=contract.func
tions.viewrequests().call()
   data=[]
   for i in range(len(_roomreq)):

      dummy=[]
      dummy.append(_roomreq[i])
      dummy.append(_customers[i])
      dummy.append(_aadhars[i])
      dummy.append(_city[i])
      dummy.append(_noofrooms[i])
      dummy.append(_noofdays[i])
      dummy.append(_dates[i])
      dummy.append(_noofadults[i])
      if(_roomids[i][0]==100):
         dummy.append('Vacated')
      elif(_roomids[i][0]==0):
         dummy.append('Request Not Confirmed')
      else:
         dummy.append(_roomids[i])
      data.append(dummy)

   return render_template('customers.html',dashboard_data=data,l=len(data))
```



List of Customers

SNo	Customer Wallet Address	Aadhaar No	City	No of Rooms	No of days	Booking Date	No of Adults	Room Ids	Room Req ID
1	0x9A50C18D6b3D2a256598e66eeBB2361DfAAa3F12	123456789012	Guntur	1	2	10th March	1	2	Vacated
2	0x9A50C18D6b3D2a256598e66eeBB2361DfAAa3F12	123456789012	Guntur	1	3	15th March	2	2	Vacated
3	0x5498269d10B14B0c7Dab5816406bCD193F2186A4	134578905432	Guntur	2	1	11th march	3	4	Vacated
4	0x5498269d10B14B0c7Dab5816406bCD193F2186A4	134578905432	Guntur	2	3	12th april	4	4	Vacated
5	0x5498269d10B14B0c7Dab5816406bCD193F2186A4	123456789012	Guntur	1	2	13th september	5	2	Vacated
6	0x1C5cFBb8021314Bd57abBAEE6fCE9826622FfFc4	987654321098	Guntur	1	2	1st jan	6	2	Vacated

4.1.l.Customer directory

Customer directory shows all the list of customer details contains customer wallet address, aadhar number, city, no of rooms, no of days, booking date, no of adults, room ids, room request id.

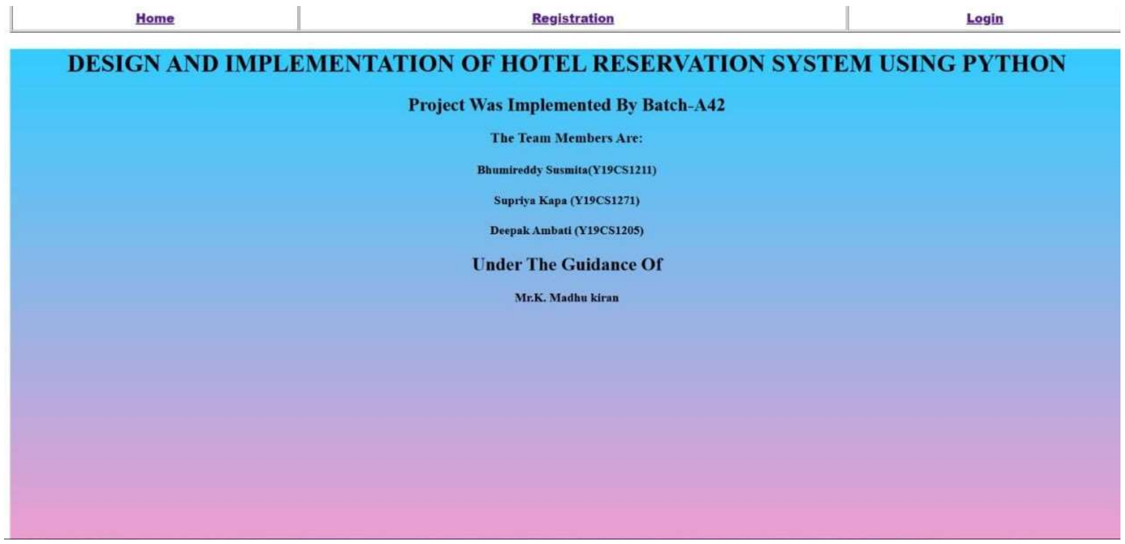
Admin logout :

```
@app.route('/adminlogout')
def adminlogout():
    return(redirect('/admin'))
```

After the completion of process then the admin can logout from the page.

4.2 STEP BY STEP EXECUTION

STEP 1: Open application



DESIGN AND IMPLEMENTATION OF HOTEL RESERVATION SYSTEM USING PYTHON

Project Was Implemented By Batch-A42

The Team Members Are:

Bhumireddy Susmita(Y19CS1211)

Supriya Kapa (Y19CS1271)

Deepak Ambati (Y19CS1205)

Under The Guidance Of

Mr.K. Madhu kiran

Fig : Home page displays for all any one can access this.

STEP 2: User registration

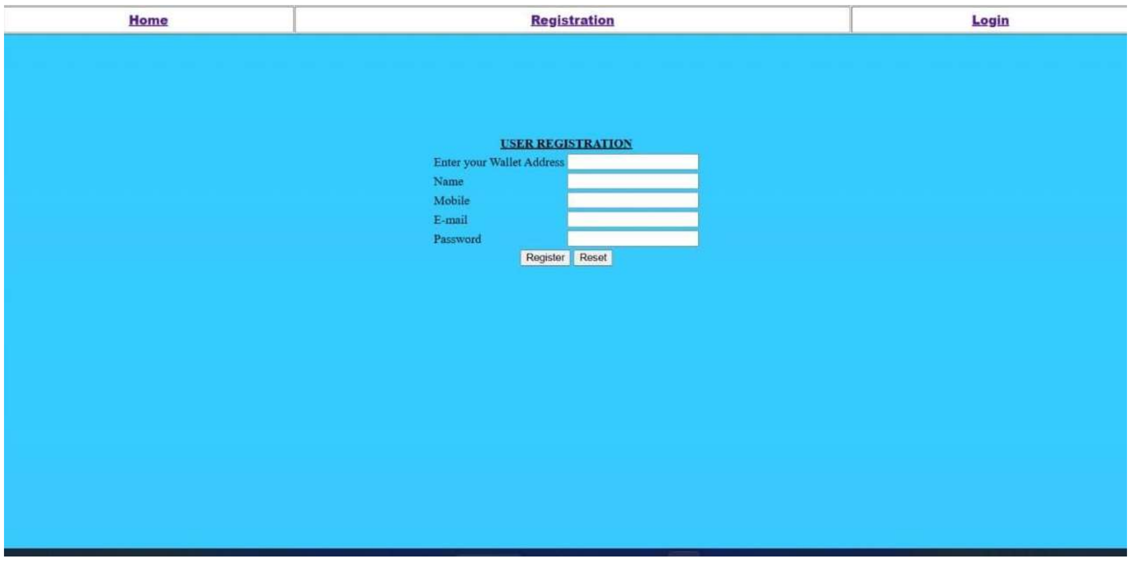


Fig :All the required details are filed accordingly.

STEP 3: User Login

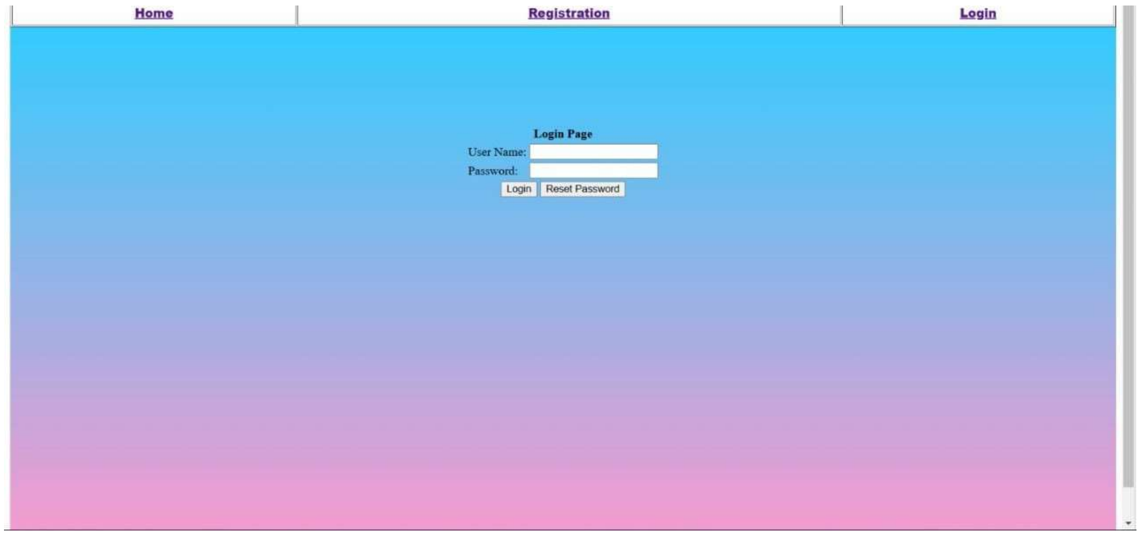
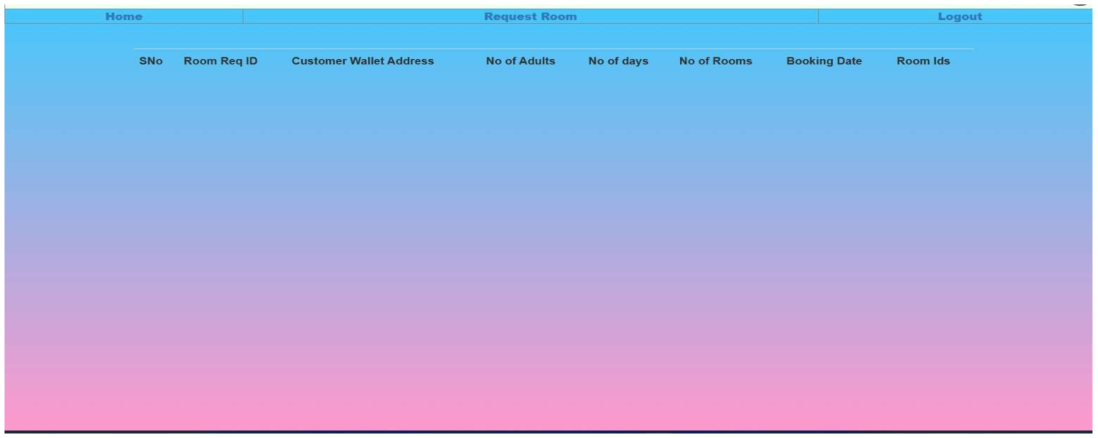


Fig : User login page

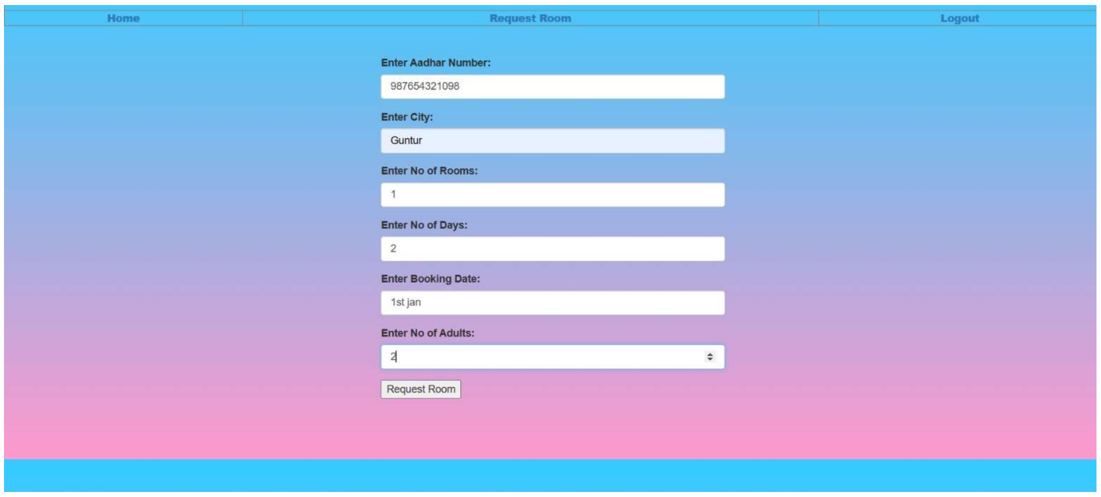
STEP 4: User Dashboard



SNo	Room Req ID	Customer Wallet Address	No of Adults	No of days	No of Rooms	Booking Date	Room Ids

Fig : User Dashboaard page

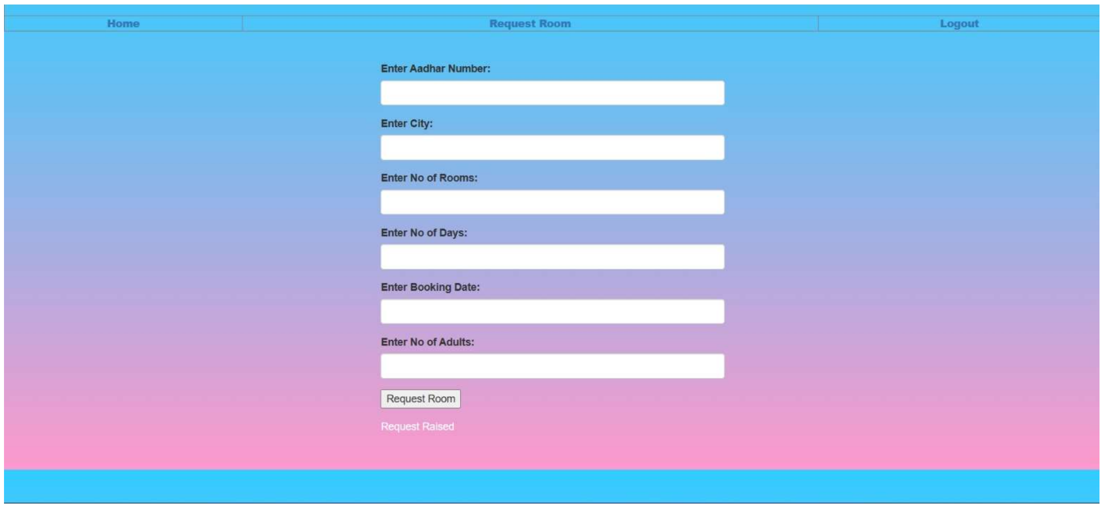
STEP 5:Room request



Enter Aadhar Number:
987654321098
Enter City:
Guntur
Enter No of Rooms:
1
Enter No of Days:
2
Enter Booking Date:
1st jan
Enter No of Adults:
2

Fig : Room request page

STEP 6: Request raised



Enter Aadhar Number:
Enter City:
Enter No of Rooms:
Enter No of Days:
Enter Booking Date:
Enter No of Adults:

Request Raised

Fig : Request raised page

STEP 7: Admin login

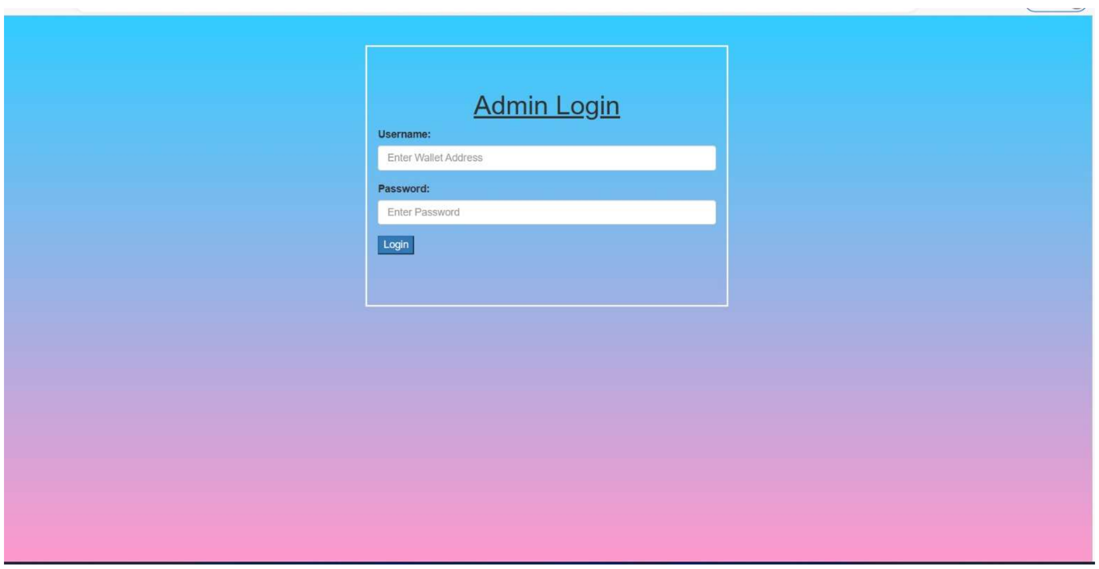
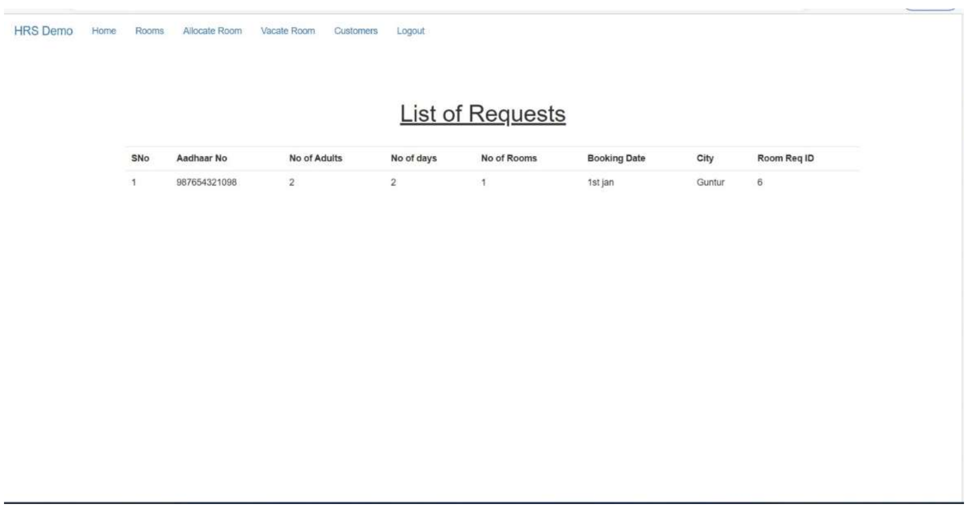


Fig : Admin login page

STEP 8: Admin dashboard

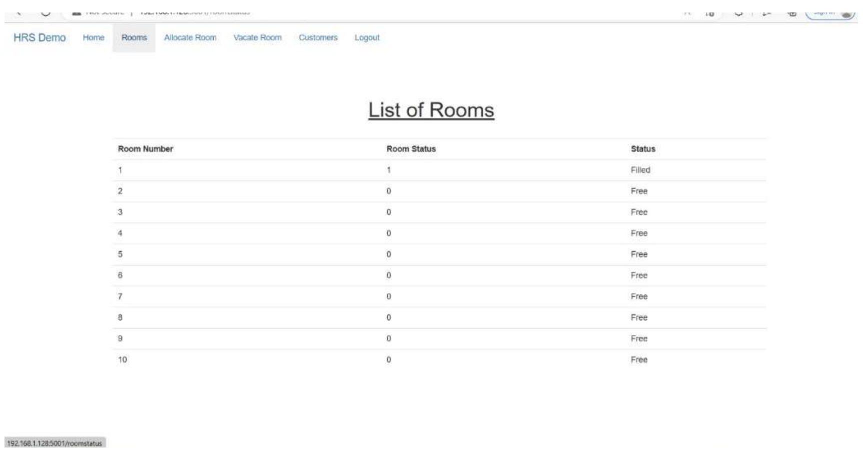


List of Requests

SNo	Aadhaar No	No of Adults	No of days	No of Rooms	Booking Date	City	Room Req ID
1	987654321098	2	2	1	1st jan	Guntur	6

Fig : Admin dashboard page

STEP 9: Room Status



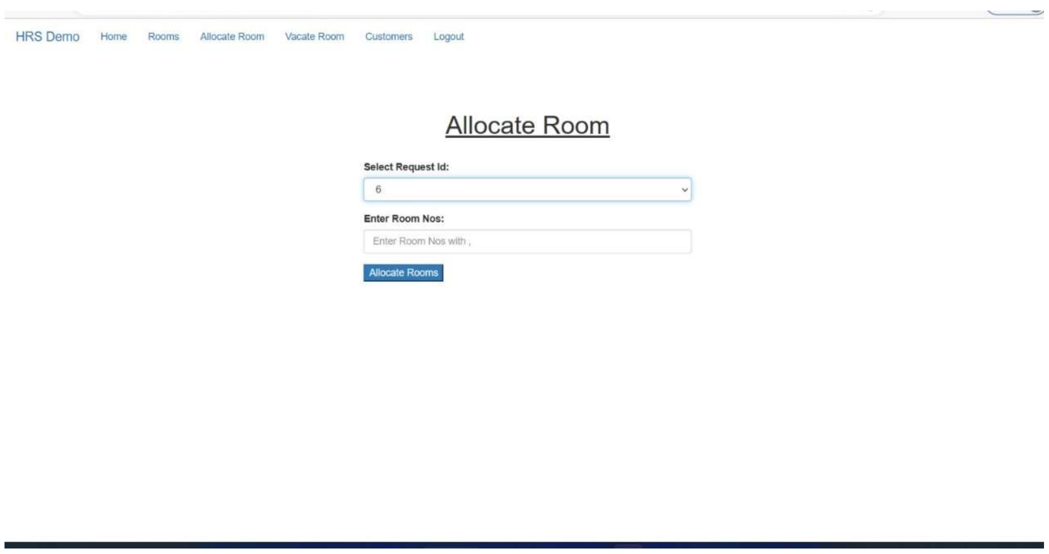
List of Rooms

Room Number	Room Status	Status
1	1	Filled
2	0	Free
3	0	Free
4	0	Free
5	0	Free
6	0	Free
7	0	Free
8	0	Free
9	0	Free
10	0	Free

192.168.1.128:5001/roomstatus

Fig : Room status page

STEP 10: Allocate room



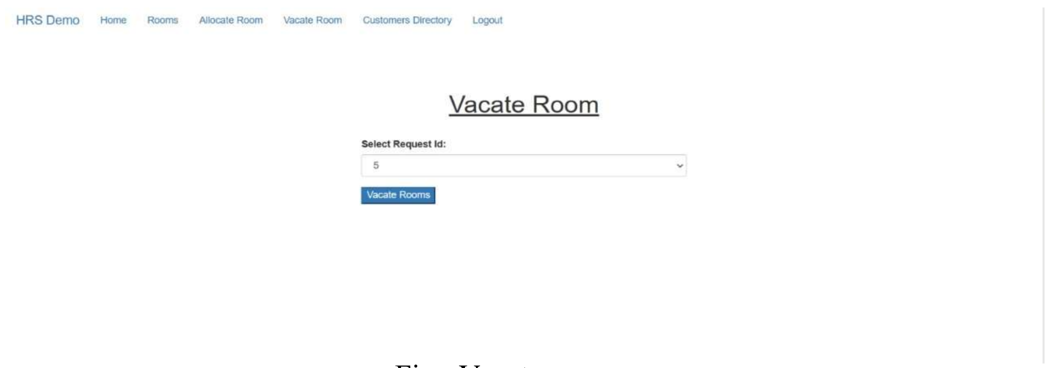
Allocate Room

Select Request Id: 6

Enter Room Nos: Enter Room Nos with ,

Fig : Allocate room page

STEP 11: Vacate room



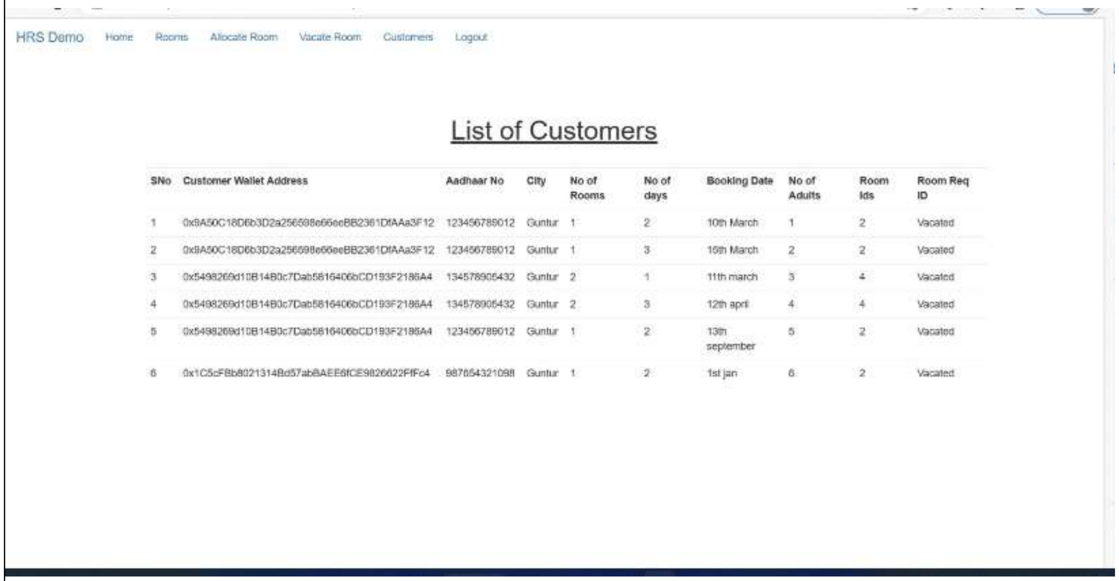
Vacate Room

Select Request Id:

5

Fig : Vacate room page

STEP 12: Customer directory



List of Customers

SNo	Customer Wallet Address	Aadhaar No	City	No of Rooms	No of days	Booking Date	No of Adults	Room Ids	Room Req ID
1	0x9A50C18D6b3D2a256598e66eeBB2361DfAAa3F12	123456789012	Guntur	1	2	10th March	1	2	Vacated
2	0x9A50C18D6b3D2a256598e66eeBB2361DfAAa3F12	123456789012	Guntur	1	3	15th March	2	2	Vacated
3	0x5498269d10B14B0c7Dab5816406bCD193F2186A4	134578905432	Guntur	2	1	11th march	3	4	Vacated
4	0x5498269d10B14B0c7Dab5816406bCD193F2186A4	134578905432	Guntur	2	3	12th april	4	4	Vacated
5	0x5498269d10B14B0c7Dab5816406bCD193F2186A4	123456789012	Guntur	1	2	13th september	5	2	Vacated
6	0x1C5cFBb8021314Bd57abBAEE6fCE9826622FfFc4	987654321098	Guntur	1	2	1st jan	6	2	Vacated

Fig : Customer directory

CHAPTER-V

CONCLUSION AND REFERENCES

CONCLUSION

In conclusion, hotel reservation and booking can be implemented using Python by building a web application or using an existing API. There are several libraries and frameworks available in Python that can be used to create such an application, such as Flask, Django, and Pyramid.

To build a hotel reservation and booking system, the application needs to have the following features:

1. User authentication and registration
2. A search engine that allows users to search for hotels based on various criteria such as location, price, and amenities.
3. A booking system that allows users to book a hotel room by selecting a room type and providing their personal information and payment details.
4. A database to store information about hotels, rooms, users, and bookings.

The application should also have features for hotel management, such as the ability to add or remove hotels, manage room availability, and view bookings.

Overall, building a hotel reservation and booking system using Python can be a complex task, but it can be made easier with the use of existing libraries and frameworks. With the right development team and resources, it is possible to create a reliable and user-friendly hotel booking application.

REFERENCES

- Visual studio code (https://code.visualstudio.com/download)
- Ganache (https://trufflesuite.com/ganache/)
- Python 3 (https://www.python.org/downloads/)
- Visual studio (https://code.visualstudio.com/download)
- Nodejs (https://nodejs.org/en/download/)