**app.py**

|  |  |
| --- | --- |
| **Technique** | **What it does (line number)** |
| Complex data model in database, several tables.  Cross-table parameterised SQL  Aggregate SQL functions  User/CASE-generated DDL script | 224 – gets the line username, password, and privilege from the credentials table for the user, searched based on the hash id generated  274 – gets the username from the credentials table that matches the hash that is generated  302 – gets the username and password from the credentials where the hash id matches the hash generated  520 – gets all the id of the files in the files table to check if the randomly generated id is in there  550 – inserts into the files table the id of the file, username of person submitting it, file name, file data, file tree, and date.  568 – gets all the data from the files data from the files table  635 – Gets all the file data from the files table, where the id matches.  670 – Deletes the files depending on the id  736 – Selects everything from the general room table which holds chats  764 – Inserts the username, message, and date into the chat room table |
| List operations  List | 32 – Stores the credentials into a queue list  54 – Stores the data into the data in order list  89 – Peeks into register queue  94 – Enqueues to the register queue  107 – Dequeues from the register queue  232 – appends new username into the username list  237 – appends new password into the password list  242 – appends new privilege to the privilege list  422 – Stores the banned characters in register credentials  531 – Adds to the id list  639 – appends to the downloaded data  748 – Adds to the messages list |
| Pattern matching (regular expression) | 606 – Searches the files with search query using regular expression. |
| Circular queue with queue operations | 27 – class for the circular queue that stores the register requests  36 – checks if the queue is full  44 – Checks if the queue is empty  52 – Gets the queue in order  89 – Peeks into the queue  94 – Enqueue into the queue  107 – Dequeues from the queue |
| Hash table | 255 – Hash function that creates hash value from the username  222, 272,300, 346 – Generates the hash using the hash function from the username |
| Files organised for direct access | 125 – Reads the register request from JSON file  172 – Updates the register request JSON file |
| Dynamic generation of object based on user defined use of OOP model | 466, 687, 809– Creates new user based on the client-side sessions credentials. |
| Server-side scripting using request and response.  Calling parameterised Web service APIs and parsing JSON/XML to service a complex client-server model.  JSON.  Complex client server model | 843 – Runs the server at port 8000  382 – On the handle credentials event, gets the JSON data for credentials  400 – emits the username and privilege using JSON  416 – On the handle register event, takes the data in JSON format and credentials  496 – Gets the next value in the register queue when the get\_requests event is sent from the client  561 – get\_html\_data event gets all the files data  618 – emits JSON data to the client side  622 – On download or delete event, does command based on id. Uses JSON  684 – Sends the files to the client through JSON  711 – Emits the file data to the file page  776 – Sends the encryption keys to the client side in JSON format  782 – Gets all the chat page messages  806 -Takes messages from the client and stores it into the database. Takes it in the JSON format. |
| Complex user-defined use of object-orientated programming (OOP) model, eg classes, inheritance, composition, polymorphism, interfaces | 176 – Makes general user  196 – Admin user will inherit from the general class  723 – Encapsulates the room name and room content |

**compression.py**

|  |  |
| --- | --- |
| **Technique** | **What it does (line number)** |
| Complex user-defined use of object-orientated programming (OOP) model, eg classes, inheritance, composition, polymorphism, interfaces | 3 – Encapsulates the characters into a class by storing the frequency, character, and the code. |
| Tree traversal | 40 – Does post order traversal through the tree and gets the leaf nodes of the tree which are the characters |
| Tree structure | 221 – Creates a tree by taking the nodes and then appending them to one node |
| Recursion | 33 – Recursively calculates the codes in the tree  90 – Merge sort splits list recursively |
| Dictionary | 36 – Stores the codes  75 – Stores the frequency |
| Complex mathematical modelling |  |
| List operations and lists | 100, 101 – Splits list  199 – Holds all the decoded data  223 – Holds the nodes  165, 166 – Stores the node list |
| Merge sort | 90 – Sorts the characters based on the characters frequency. |
| Exception handling | 20 – Checks if the node is a leaf node |

**encryption.py**

|  |  |
| --- | --- |
| **Technique** | **What it does (line number)** |
| Complex mathematical modelling | 3 – Checks if prime number is prime using the Miller-Rabin method  61 – Calculates the greatest common divisor of two number  71 – Extended Euclidian algorithm to calculate the greatest common divisor efficiently  83 – Finds the modular inverse of two values to get the private key  90 – gets the totient of two values  124 – Encrypts the data  139 – Decrypts the data |
| Recursion | 71 – Recursively calculates the greatest common divisor |
| List and list operations | 43 – Iterates over the low primes list |