**Daily Task :**

* **database closed kar ke test case load** kare to kya test case failed ho jayega hame test case failed na karva na ho to ham MackingAPI ka use karenge
* **iterator is interface** aur interface ka object nahi banta to kese ham collection me object banake uska use karte hai REASON: matlab iterator method koi class me implemented hua hoga to class konsa hai?
* **Servlet** ek **interface**
* **Servlet Properties** :
  + Iinit,
  + doPost or doGet ,
  + Destroy
* **Transaction Properties** : ACID
* Atomity :
  + Fully successful or non
* Consistency :
  + database from one consistence state to other consistence state before or after transaction commint
* Issolation :
  + one transaction commited other on issolate form
* Durability :
  + When transaction commited then database brings to consistence state so when the power lose or system failure but that not effect on consistence state because jevu transaction thay eni pehla database ma data store thai jay 6e
* How to other people know that you perform each test case is sufficient or not so we use **Coverage** load
* **An interface is a contracting between two parties** 1)that implements interface 2)
* **An interface** is a not contracting between two parties is **less straight-forward**
* **Difference between tilde~ and caret^ :** 
  + **Tilde~:** used for approximate equivalent to the version,
    - **Update all future pitches without increasing minor version**
    - It’s a decimal version installation
  + **Caret^:** used for compatibility with version
    - **Update all future pitch/minor without increasing the major version**
    - It’s default version installation
* **What is nodeJs ::** it’s **Javascript runtime on v8 engine** is written in c++
* **NPM :: Node Package Manager**
* **JWT ::** allowing **user to access routes,services,and resources that are permitted with token** and it easily to access any domain . Securely transmitting information between two parties.
* **let/var/const ::** 
  + **let – particular block follows ,updated but not redeclared**

//updated

let greeting = "say Hi";

greeting = "say Hello instead";

//re-declared consist error

let greeting = "say Hi";

let greeting = "say Hello instead"; //errror :: undefined

* + **var – redeclared or updated,**

**if we initialized var in function it can only be accessed for that function only**

**Var initialized top ma karine any where use kari saku until not change or any parameter initiated**

* + **const – can not update or redeclare**

const greeting = {

message: "say Hi",

times: 4

}

//or

greeting.message = “say Hi”

* difference between **append file** and **writefile** is that writefile override the existing contentor append just add an input into file
* Deletefile we used fs.unlink(“filename”,(err)=>{if(err) throw err })

Or it will delete file

* **The querystring.decode()** function is an alias for **querystring.parse().** And The **querystring.encode()** function is an alias for **querystring.stringify()**
* **CAllBackHell** is **a pyramid of doom** means common problem arises in multiple level due to control or access function .it’s harmless when user not expected logic of program **means** callback ke under dusra callback then next then next so its like pyramid and problem yeh hai ki justify the error very difficult so **avoid error used Promise**
* **Async** is send request to server but not wait for response or any other one will be fast to send request then it will run first
* **Sync** send req to server and wait for response stepBystep process
* Async.waterfall is used as centralized for all error handling by only any one conditional on all time
* **Promises** to maintain an application,readable the code,
* ExpressJs : It’s server-side framework.save time
* **MongoDB** :It’s document database designed for ease of development and scaling
* MONGO :: It’s a command shell and use mongoD
* MONGOD :: It’s the database that implement the data and help hardware to store the data, Host process for database
* IN MONGODB :: database:database,table:collection,rows:document,column:field