

Class - 01

What is testing?

Why you want to be QA?

⇒ Software that does not work correctly can lead to many problems including loss of money, time or business reputation, even injury or death.

Software testing is executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual desire or requirements.

Why software testing is necessary?

Bug detection: requirement অনুসারী behavior করলে bug আছে।
QA: requirement অনুসারী system develop কোথায় হয়েছে তা? Quality : measurement degree of excellence. → system কম্পিউট আলো → depend on budget.

Customer satisfaction: thorough testing (end to end testing)

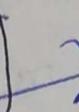
Compliance: Govt. rule

Testing objective

Authentication: system থেকে login করার পর access আছে কি? যদি আছে তাহলে আছে, তবে authorization.

service level: system কতভাবে service দিচ্ছে? গু

Performance test (load test, responsive test...) 
ইয়ুনিভার্সিটি

Verification, validation  Are we building the product right?

verification: student exam করে \rightarrow Preparation করে \rightarrow Preparation করার process কিমা আছে কিমা (পরীক্ষা
সম্বন্ধে প্রত্যেক, উপরিকৃতি প্রত্যেক কিমা, যব কৈশীর
কামানটি - প্রত্যেক কিমা) \rightarrow গ্রহণ করে ensure কৈশীর
verification অর্থাৎ যারা মন্তব্য করে কিমার plan ক্ষেত্রে কৈশীর
কৈশীর plan কিমা আছে কিমা তা ensure কৈশীর verification.

validation: expectation meet করে কিমা (Are we building the
system development process is verification right product?)
development ক্ষেত্র হওয়ার পর ultimate result check
হলো validation.

Testing principle

1. Defect আছে কিমা?

\Rightarrow defect আছে, তো একার পর পূর্ণাঙ্গ হলো a test case
হলো execute ক্ষেত্র হবে।

class 02

STLC

Testing activities

1. Test planning: for test क्या है, कि कि test आइए ?
overall idea: आपके कि क्या है ? functionality कि है
परंतु, काम किसी विकास एवं रोल, उत्तम तरह task आपके हैं
2. monitor and control

test manager

QA →
employee
काम
जूझें

3. test analysis: planning दे रखने की task आइए, लगातार
priority कौन? requirement अनुसारी acceptance criteria
किया है। (what to test?)

customer need

अनुसारी (manager
किया जाएगा
किया जाएगा)

4. test design: how to test? test case किया जूँह है।

5. Test implementation: Prioritize and schedule test procedures
for efficient execution. (Priority अनुसारी test case
यानी ready की 23) execution के लिए,
setup and verify the test environment

6. Test execution: execute लाए bug मैंने bug
report दिया, bug का परिवर्तन test case passed दिया,
actual result किये execution के, expected result
किये design phase.

7. test completion activities: test done होने की
message दियो→ good to go for release.

Testing role

Management role: test manager in org (test planning, monitor and control, completion activities)

-Testing take: analysis, design, implementation, execution

Types of SDLC model

sequential: waterfall, v-model

Iterative: for example 2 weeks ଦେଇ ଏକାଟି iteration ହାବି,
2 weeks ରୁ ମଧ୍ୟ ଫେରି କର ଏଥାରେ ହସି, ପୁନଃ 2
weeks ରୁ ନାହାଲା କରିବାରେ ହସି।

Incremental; iterative approach. Prototype 50%
RAD 6 convert 25%

MVPG iterative কো পর্যায়ে পড়ে, নবম বি increment
বন্ধুর লক্ষ্যে কর একটি দক্ষতা

TPD: Test Driven Development. Developers কোরা
খ্যান feature develop কৰা আবু কৰাৰে, at the

same time QA will test once (प्रारंभिक विकास के समानांतर) developer को क्या करने की ओर आयेगा तो उसके बाहर complete developer को चाहता (specific feature के लिए test case) · developer आये तो उसके test case नहीं पास हो जाता क्योंकि वे अच्छे नहीं होते।

As a QA, developer को दिए गए test case पर पास (most important) करना आवश्यक, फिर उसका एक preventive approach.

ATDD: अक्षय acceptance test (feature को कैसे पास करें) के लिए इसका एक form जिसके माध्यम से minimum घटकों का निश्चय किया जाता है।

BDD: user flow wise defined form format we have इसका एक Given, when, then का नियम है,

Given: user visit website

when: user input email and P/W } every feature

then: user successfully logged in. } given, when, then का नियम

frame इस मात्र non technical person software का flow बनाता है और उसके लिए user को बहुत ज्ञान की ज़रूरत नहीं।

जल्दी QA का तरीका होना: preventive approach & test case failure then corrective approach & check के लिए expected & actual result same किए

Static testing: feature under development. ক্ষেত্র QA
এবং ইস্ট কাম নাই।

Dynamic testing: feature develop complete. QA start
testing.

High independence: QA do test

Scrum: 2 weeks এর কাছে predefined সময়।

Kanban: continuously কাছে চলতি থাকে। 2 weeks এর
মধ্যে মনোযোগ রাখে। ফিল্টেড release করতে পারে।
এখন আজ আছে predefined সময়।

class - 3

Software testing classification

Black box: QA engineer

Based on approach)

1) Static testing: system রাতে আসা আজ ও বিভিন্ন
activity execute করে। (requirement analysis,
test condition, test criteria ফলো, test case লেখা,
test implementation এবং ready for execution
করে। এশে static testing করে।)

2) Dynamic testing, test execution করে। যাকি কর
কী শুনে আছে সেখনে dynamic testing করে।

Dynamic testing

~~1) Functional: feature or requirement for user first?
(Reg → P/W condition, all field, email verification, profile information...)~~

~~2) Non functional: different different browser & test environment, memory leak etc in system use etc
different system then behave etc~~

Based on techniques

1) Black box: it don't have to know the internal architecture of a system but you will execute the system like as an end user. In here you are like a representative of end user. You have to think deeply what an end user can do.

प्रोजेक्ट इंजिनियर एवं उपयोगकर्ता के विचार से अनुसारे एक एंड यूजर के लिए एक सिस्टम का अधिकारी के रूप में काम करना। अनुसारे एक सिस्टम का अधिकारी के रूप में काम करना। अनुसारे एक सिस्टम का अधिकारी के रूप में काम करना। अनुसारे एक सिस्टम का अधिकारी के रूप में काम करना।

2) Whitebox: ~~an engineer~~ engineer ম্যান system রে backend
সম্পর্ক তাঁর এমন ফিল্ট API আছে, তেন্তের এমন API
call রয়ে, এম action রয়ে against তাঁর API call
রয়ে, authorization process, এম platform র রূপ
কর্মসূচি, also db knowledge, integrity, 3rd party
integrity রয়ে code level.

- In code র অসেস করা পাবলো:
- Particularly কোন নতুন feature ডেভেলপ কৰা হচ্ছে, উকিম ইম্পেক কোথায় পড়ত আহো বা শিল্প কৰা পদ্ধতি কোথায় আবশ্যিক test execute কৰত পাৰিব,
- একজুলী statement পাবলো execute কৰ ও ফিল্ড test case prepare কৰা পড়ে আবশ্যিক পদ্ধতি ১০০%.
- test case cover কৰা পড়ে,

3. Experienced based: কোনো system কৰাবলৈ যদি minimum level কো' knowledge পাবলো, তবে knowledge কো' উপর ভাসে এবং নতুন system explore কৰা পাবলো

2 types of dynamic testing

1. Functional: specific feature র ক্ষেত্ৰে feature কি' requirement কি' আছে কিনা? ex: (user reg → all field filling কৰা, means only mandatory field filling কৰা, * কোনো নথি required field filling না কৰা, P/w condition কো' হুয়ে apply কৰা/না কৰা ---)

2. Non functional: browser related. (different browser কে কৰা কৰা), mobile view, এস্যুট. ১০০/১০০ user reg এক্সেন্স কোম্পানি

unnecessary keyword → naughty string.

behavior, naughty string, blank string, double space কোর্ট কুজনি সুব্রহ্মণ্য গোপনী হৈমেতে non-functional test কোর্ট প্রতিবাদ,

Static testing vs dynamic testing

static	Dynamic
Finds defect directly.	cause failure, and defects are determined through subsequent analysis.
Doesn't require executing the code.	Requires the execution of code.
can be applied to non executable work products (e.g., documents).	can only be applied to executable work products (e.g., running software).

Functional testing strategy

Sanity testing: specific feature test কোর্ট প্রতি, একটি feature related কোর্ট কোর্ট আছে অবস্থা কোর্ট feature কে impact কোর্ট কোর্ট feature কে আছে এবং একটি module কে আছে, দুটি module কে আছে এবং connected আছে কোর্ট module কে কোর্ট impact কোর্ট আছে, আবার একটি কোর্ট feature B. module test কোর্ট,

ex: login → login is related with registration and profile.
login, forgot p/w হলো feature testing.
reg, profile হলো sanity testing.

Regression testing: নতুন কোনো feature আসার পরে, অব্য কোনো ক্ষেত্রে impact পড়তে বিনা দিলি ensure কৰা।
(multiple feature add করলে whole system কে ক্ষেত্রে test কৰা হবে)

Smoke testing: system কে basic flow দিকে আকুশিতা, crucial ক্ষেত্রে functionality broken হচ্ছে কিনা, system stable আছে কিনা দেখলো ensure কৰা।
Smoke testing. e.g. 100 test case, some are high priority, some are not. ৫৫ test case দিলো এবং important ক্ষেত্রেই সুস্থিত execute কৰাবো, তে ক্ষেত্রে system কে confidence কৰতে, deep testing কৰিবো।

এখনি আরেক feature আলে, regression testing কৰি মুক্ত কৰি তাই minimum হলো smoke test কৰি এবং then feature হলো related sanity test complete কৰা হবে।

confidence test gain কৱার এবং smoke test কৰা হবে।

Exploratory testing: from experience based testing, intention
of the tester to gain domain knowledge while learning

Monkey testing; Random testing.

Geometric testing: Cross feature multiple time execute হয়। Login button প্রতিবার press করলে, ফলে double একজন টেস্ট করে failed হয় এবং login মিসেস টেস্ট করে টেস্ট করে তার পরে অপেক্ষা করে আছে।

Non-functional

Benchmark testing: System ~~is~~ ~~in~~ standard set
कम्प्यूटर (Particular OS ग्राहक नहीं, android OS
minimum 10 version, 6 टेक्स्ट, minimum 2GB RAM.)

Shift left = test early principle

Test level

white box

1. Unit testing: Done by developers. particular কোডের
function test করে ফলিষ্ঠু parameter pass করে.

2. Integration testing: system টেস্ট করে তিনি মার্ফত data
flow properly করে নিয়ন্ত্রণ করে এবং ensure করে,
feature: send money

Reg → KYC (NID, necessary document verify করা) →
Deposit → send money → balance → statement →
Received sms → rec (receive money),

API প্রযোজন - test করা হবে। (black + white box)

3. System testing: system level টেস্ট করা. Blackbox
testing. Regression testing system টেস্ট করে রয়েছে
(অবশ্য functional testing system testing নয় কিন্তু)

4. Acceptance testing: Release মাত্রার আগে client
কাছে নিয়ে যাও test করে,

Component integration testing:
O বা API

System integration testing: third party app এবং
API (Google, microsoft etc)

multiple unit testing
component testing

Difference between component and unit testing

login component

API 1 \Rightarrow email, P/W \Rightarrow send OTP (1st unit test)

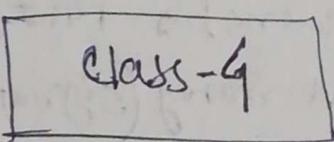
API 2 \Rightarrow verify OTP, if successful \Rightarrow give auth access (component testing)

Test Quadrant

Q1: API testing automatically \rightarrow Technology layer

Q2: API " manually \rightarrow Business "

critique: Product द्वारा शुलगत माल का उपयोग करता है



Test technique

1. Blackbox / specification

2. whitebox / clearbox / structural

3. Experienced box / glassbox

Blackbox \rightarrow UI level G testing करती।

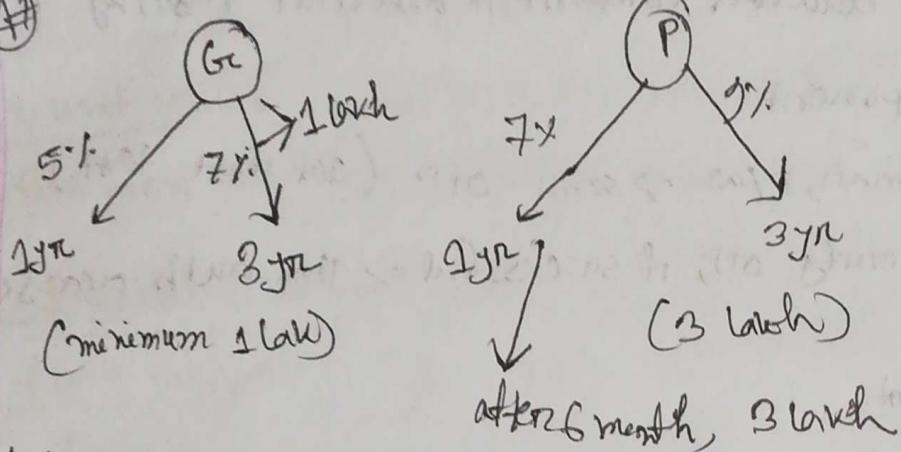
Blackbox: Requirement अनुसार इस तरीके base परे आवश्यक
for test case टिक्का करके (specification). UI based
testing also

1. equivalent partitioning \rightarrow used it more

2. BVA: i) 2BVA ii) 3BVA

ep scenario

(A)



test case covered : 50%, rest of the 50% should be write

state transition : system টে যখন multiple input আসে

no of test হবে তখন ফিল্টে input দ্বাৰা অধিক নথী পাইলে একটি result show
cost - $\frac{1}{2} \times (\text{state}) \rightarrow \text{bckh}$ send money (signup \rightarrow login \rightarrow send
money \rightarrow select amount)

(2)

Decision table testing : based on the condition you have
to find out how much test cases you have to write,

Whitebox testing technique

Code level টে test কোৱা : developer দ্বাৰা code উপর
base আসি আলাকে ফোফা test case ফুঁড়ি দেব

- Bocken দ্বাৰা knowledge & গবেষণা কোৱা দুবে
- indirectly sanity testing কোৱা help কৰিব
- (চেনজ চেনজ কৰে পৰামৰ্শ দেব)
- change কৰে পৰামৰ্শ দেব
- effect কৰে কৰিব

each line of code

i) statement coverage : $\frac{1}{1}$ 100% statement coverage can not guarantee bug free software. parameter, condition statements analyze কোরি স্টেটমেন্ট

decision/branch coverage (if-else)

all decision cover = all test case will be covered

Decision Test coverage

age = 18

if ($age \geq 0$) {

 if ($age \geq 18 \text{ } \&\& \text{ } age \leq 100$) {

 print("voter") $\rightarrow S_1$

 else if ($age \geq 100$) { $\rightarrow D_2$

 print("Not allowed to give vote") $\rightarrow S_2$

 else { $\rightarrow D_3$

 print("not voter") $\rightarrow S_3$

 }

 else { $\rightarrow D_4$

 print("age cannot be negative") $\rightarrow S_4$

}

input : 120

else if will be execute (কিন্তু মাত্র 1/4 execute করে) করা হবে ১৫% নয়। $\therefore 25\%$ test case covered.

100% statement coverage cannot guarantee 100% decision coverage

ex: method overload, invisible if else

~~Statement~~ Statement coverage rules for find out minimum test case:

For nested condition:

$$\text{Statement coverage} = \text{no of else} + 1$$

For un-nested condition:

no of test case \nwarrow Statement coverage = 2 (when else are there)

u u = 1 (when no else are there)

Branch coverage rules for find out minimum test case

For nested condition:

$$\text{Branch coverage} = \text{count of if} + 1$$

For un-nested condition: \nwarrow if/else of

DC = 2 always

N.B: Statement for priority function to be missed

MIS, D.C for priority function SC must be present

class-5

Experienced based testing technique → test charter (ISTQB)

1. Error guessing (2) Exploratory

data flow/integration

(3) Checklist-based

→ মাঝি test কর্তৃত সেটিং
shortest way → checklist
ওয়াল্ট ফিল্ড

checklist based testing

Testcase vs checklist

~~test~~ planning

describe the objective, resource and process for a test project.

it ~~outlines~~ outlines [how and when] we will achieve our test goals

• common activity of SDLC & TSLC

Test strategy: test planning ও কোর্ট নির্দিষ্ট চৰ্যাবে
organization define কৈ কৈ

Test planning: context of testing, assumption and constraints, test approach (test planning ও পি, কো
ন্ট্ৰান্ট ও টেস্ট স্ট্ৰাটেজি কৈ কৈ)

Test effort estimation

1. Metrics based technique:

i) Estimation based on ratios: from previous job experience (people need)

ii) extrapolation : iteration 1, iteration 2, ... iteration n.

iteration n. নতুন গুরুত্বপূর্ণ সময় কাটার জন্য এবং প্রক্রিয়া সময়ের অন্তরে কাটার জন্য, যদি task complete হয়ে থাকে

বিনামূলে estimation আগে করে ফল পাবে,

Based on the task complexity, you can get idea how much effort you can give for the release.

2. Expert based techniques

i) Wideband Delphi: less effective for QA but more effective for developers.

ii) Three-point estimation: optimistic = best likely = moderate , pessimistic = worst

$$E = (a + 4m + b)/6$$

↓ worst ↓ ↓ best
 medium

$$\text{measurement error, } SD = (b-a)/6$$

Acceptance criteria

- i) scenario oriented
- ii) rule oriented (most used)
 - ↓ take long time
 - ↓ bullet point based

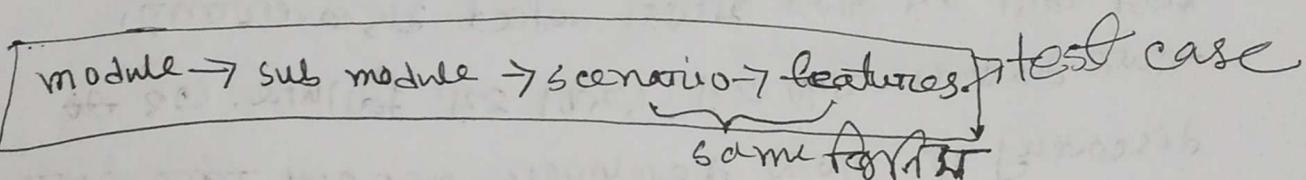
acceptance criteria কিভাবে অনুরূপ test case ফর্মেট হবে?

- test condition

Recipient should have valid account

- test case

- i) a valid user sends money to a valid account (pass test)
- ii) " " " " " " " " an invalid " " (expected result: fail send ₹200 ग)



class - 6

test scenario: user login

- ii title: user can do login using valid email and p/w

test scenario: user login

- ii title: user can't login using invalid email/pw

test scenario: send money

- ii title: user can send money to a valid recipient
 - ii " " " " " " " " an invalid " "
 - ii " " " " " " " " more than his balance

test title : user can't send money negative amount
" " " " full amount excluding service fee

test case 2011 रुपये, test case id १८ २०११ रुपये

Static testing : scenario, title, expected result. जटिलता
dynamic testing or कठिन फॉर्म

Bug report

defect : defect may arise, यहां software dynamically
test करा इस प्रकार वाहां defect discover होता,
यहां discover करा इस तरीका से failure. तो यह
discovery करना developer accept करते अगले
प्रीति bug होता,

Bug release : operation environment ~~fail~~ failure -
Bug QA टेस्ट विकास प्रोड,

Bug leakage : ultimate failure. यहां end user को
मार्ग नहीं बदला failure होता,

Prioritization : business और financial
लकड़ा उत्तरी, (from user side)

Severity : feature और फीचर तो महत्वी important
feature (system side)

Bug Triage: नियन्त्रित करने के लिए bug की ओर से evaluate करकि bug का कठि important, severity categorized करकि based on priority and severity अब इसका आधार base पर उत्तमी issue शूलों noted down करकि.

Test completion: development done, testing done, system release.

automation & test automation are different.

Class - 7 : API architecture

API testing : gray box testing

Frontend 품질 테스팅을 위한 모듈화된 테스트 구조 / CI/CD feature 툴
execute 모듈 테스트를 통해 network calling 테스트를 수행
execute DB 품질 API 및 logic 테스트를 수행 execute 테스트

Frontend ৰ আপনা যাই পদ্ধতি, যেটা backend-ৰ logic API
এবং মাঝিটো কৈখন হয়।

labeling: Integration testing

API: এক ক্ষেত্রে interface এর UI থাইন, পিলে backend
G URL call ~~পাই~~. (terminal also PI)

Postman → API testing

Jmeter → API calling → server → performance count

Browser দেখতে load হবে, তাহলে backend API call
করে থাকে, \rightarrow network calling

Browser এর মাধ্যমে only get API এর কথা আসে

What API do?

\Rightarrow কোনো application দেখি internal module কোথায়
থাকে, different system দেখি কোথায় bridge কোথায়
থাকে, কোনো data flow হচ্ছে পারে।

1. REST (ঠিকঁ): API REST API দ্বাৰা পৰিচয় কৰা হৈছে
2. SOAP হৈছে অন্যটো

REST API

- http \rightarrow Protocol
- domain.roadtocareer.net \rightarrow base URL (domain/
subdomain/ip)
- user \rightarrow controller
- Search? phone-number=0168660690 \rightarrow end point
 $\underbrace{\text{phone}}_{\text{action}} \downarrow \underbrace{\text{parameter}}$

controller: CRUD activity

Query param: action কে পতে (?) থাকে, multiple
value থাকতে পারে, filtering নাও হৈব। use কৰা হয়।

Path param: action কে পতে (1) থাকে, সূচী

value থাকে।

theoretically full url

endpoint, controller / action / parameter (Practically) →
for internal API

Thirdparty URL → external endpoint

Gateway: ক্ষেত্রে কাউন্টুন facility দিয়ে রয়ে, তা কাউন্টুন data
access এবং প্রাপ্ত রয়ে নই বিনিয়োগে controller করা হয়,
(This is a separate component)

Why use API gateway?

gateway নই মান্যমে API রয়ে distribute করা system এতে
হয়ে controller এতে পাই, নতুন তাকে common
behavior রয়ে, gateway এতে controller এর মধ্য customize করা,
controller এর under তে API রয়ে এর activity ~~to~~ controller করে,

Activities of API gateway

1. Protocol translation: HTTP → SOAP (automatically convert)

Software system architecture

1. Monolithic: without API

2. Microservice: DB, UI, Backend এবং অন্যান্য অন্যান্য
যাই, যেগুলো নথি (যেগুলো fix change করা হয়ে
(e.g. MySQL → PostgreSQL)

Different types of API

1. Public: ex: sandbox API (demo API) - No need any permission. ex: google map
2. Partner: need subscription for premium feature ex: chat GPT.
3. Internal: system to backend for every API
4. Composite: 1st API to call another backend API
অন্তরের API কল করে আরেক অন্তরের API দিয়ে আবশ্যিক ধূমগতি পাইবে।
Any response দিবলো সাবলো।

HTTPS://api.bb1.com/fimoney/transfer?fromAC=123
&toAC=001&~~toBank~~RoutingID=12345&amount=2000

1. check auth by API

2. account validation API

3. routing API

4. to account validation by API

5. enough balance by API

উপরোক্ত API

call করা হবে।

যদিও কল

হবে।

5. Private: no one can access. limited to server/router.

API HTTP method

1. Get: only retrieve data (status code 200)
2. Post: DB ~~entry~~ (information entry to db)

→ Higher bandwidth

API's body আওয়াবের জন্য সামগ্রী payload

3. PUT: update data in db. (entire obj কে update কর)

4. Patch: Particular properties কে update কর,
lower bandwidth.

API authorization

1. API keys (stateless): user এর জন্য session থাল্লু না,
পরু কে জো info decrypt কর + পার্টাই করা হবে user
valid কিনা?

statefull: cookie দ্বয় কর্তৃ info থালে, session এর
কে under এ রাখিয়ে রাখি, info কুকি হওয়ার possibility
মাত্র।

Bearer token / JWT

Basic auth: api key কে base64 ফর্মে token কে convert
করা হবে token এর অভ্যন্তরে থাকে, secured.

Basic auth: user name, P/W কে token কে convert
করে কর্তৃ decrypt কর Plain text কে কোথায় রাখবে?

OAuth: Google দ্বাৰা through to sign up

AWS: Amazon দ্বাৰা in in

class-8 | Postman

যদি আপনী API মিলে `dataflow-test` এবং `integration testing`.

Variable in Postman

- i) collection ii) environment iii) global

Environment: যদি কোনো environment create করা হয় তবে select অবশ্যই ভাবে, তাহলে আমরা একেন collection নি করে না করি। যে অবস্থার env variable কে access করতে পারব।

collection variable: bearer token টুন বাবুর নিয়ে আসো (dynamic করো)

Script :

```
const jsonResponse = pm.response.json();
pm.collectionVariables.set("token", jsonResponse.token)
```

Use/ create

Script :

```
const jsonResponse = pm.response.json();
```

```
const customerId = jsonResponse.user.id
```

```
pm.collectionVariables.set("cusID", customerId)
```

view documentation

assertion: যদি response properly আসে কিনা, expectation meet
করুন ফল তার এর assertion point add করতে হবে, (test
script ফাইলে ২২৫), → positive test & negative test case

login → json value check

```
const jsonResponse = pm.response.json
```

```
pm.collectionVariables.set('token', jsonResponse.token)
```

```
pm.test("verify user login is successful", function() {  
    pm.expect(jsonResponse.message).contains("Login successful")  
})
```

in case কিন্তু মাঝে input field বাজে, তাহলে কোন
minimum কী হলেও test case কিম্বা সার্ভের, (query
param) ex: server error, (-1)

class-9

environment variable: এক collection different different
environment run করতে চাহি, like different different
servers (localhost, dev, staging, production, ...)

যদি প্রথম, প্রথম মডুলো release করে, staging করে,
প্রথম প্রথম, staging এ আবার test করতে থব, এমতন

Collection variable এখানে base url change করা হবে।
যদি রয়ে রয়ে separate environment যিন্তে কিছু config-
uration আছে (যেমন) env spec specific. যেমন local
এবং staging দেখা database same না, তবে মাত্র
local এবং staging নয় তবে separate collection
হবলেব।

collection রয়েছে তাই স্থায়ী, আইনি env change
করা।

Newman (for report generate)

• We need

1) Node JS (check version: node -v)
then

- i) Create a folder
- ii) Open folder with VS code
- iii) terminal → new terminal (~~node npm init -y~~)

node pkg manager initialization

npm init -y → will create package.json

npm i newnam → dependencies

জন্য under newnam

বর্জন show করব

Run through newman

1. console mode

collection share (share via API) → copy link

npx newman run Paste link

new (del and file can have more things need config MACS
Administrator privilege (root))

2. write report

in report.js

6

i) collection: collection with chance via api token from API.

Creates a new folder: Reports

in terminal: npm i newman-reporter-htmlextra
install success & dependencies in under Global
extra file will be created

ii) in terminal: node .\report.js (for a single program
application token API 22) npm Go through (npm run
test)

package.json

"test": "node .\report.js" → in terminal: npm test

GitHub

(Forbidden) First log in with secret key. It shows which github & push can
be done. Then hide token API becaz it's private key. Other
local IP address shown

Create .env file (dotenv means server da variable nahi)

Create .env file

Keep API key in variable inside .env

report.js const Newman = require('newman');
require('dotenv').config();

accessKey = \${process.env.accessKey};

npm test

another way: link থেকে মুদ্রণ করে static
জ্বর-ফাইল থেকে run করতে চাইলে

Postman collection → export → collection v2.1
→ exporting in my local machine → then in ~~report.js~~

collection requires (.collection/filename)
NPM test
exported

Github push

[class - 10] → Jmeter

Jmeter is a performance testing tool.

Performance test: একটি system continuous load
যদি কোন behavior দেখে অব্যু হলে feature multiple
user করে ফিল্ড নথ্য বিনে এবং use করলে
যাতে, তারে system ক্রিয়া করত ক্ষেত্রে নাকি
crash হয়ে , behavior observe করা হবে

capacity testing : acceptable level → কোন
কষ্ট হাজ কেন কাটী অবলম্বন কর্ত্ত্ব করে নিতে

হচ্ছ capacity, ex. একটি নিম্ন ২০ কেজি মানবান
বিয়ে নিম্ন হল প্রাপ্ত মেডে আমের প্রাপ্ত যেতে আছে,

उष्णजे त्रियों capacity इलो २० लीटर. यसी आदा १८ लीटर याहाला
२५ लीटर त्रिया त्रियां आहे, वाढवावा. In capacity we have to
sure that system always behave नव्हावा, in
acceptable range.

Jmeter
stress
test +
पर्याप्त
time
योग्यता,
useful
प्र०/
प्र०

3. Stress testing: ප්‍රමාද දෙපාල විසින් load test pass කළුතු, එයෙන් තිබු
 2000 ග්‍රෑන්ටර් නියු ලද ලද pass කළුතු, යෙන උග්‍රෝ මේ අනු
 ප්‍රමාද යාග්‍රහ ලෙස පරිඛු යි(2) fail නො ඇතු යාරේ, මිනි
 point C fail යාරේ, මිනි මාර්ගෝ Point D නි නො capacity point,
 capacity බැංකු තොනෙන් system fail නො ඇතු යාරු.

4. volume testing: DB to huge transaction data load for
that is, simulated data insert.

5. Spike testing: Specific time(s) before analysis useful
যাতে প্রতি সময়ে এবং অন্য সিস্টেমের কাছে
কোন পরিবর্তন হবে।

spike load $\xrightarrow{\text{pos}} \sigma$ stress $\xrightarrow{\text{pos}}$ capacity, volume.

performance test on jmeter

→ Non functional testing

① Jmeter
run
কার্যক্রম
CMD
থেক্স

bottleneck: stress test point

↓ lower bottleneck: (or point তখন ফেল দেওয়া শুরু করা হবে)
↑ upper n : n : n → system normal crash

② in Jmeter, load → response (load করা হবে এবং response
script পরে response, thread group → listener → view result
GUI mode
CMD
Load test
প্রোফিল
ex

10 hrs → 100000 people read paper
1 hrs → 10000 people read paper
3600 sec → 10000 (প্রতি sec 250 user load)
1 sec → 2.78 (প্রতি user load)
1 min / 60 sec → 166.67 (প্রতি sec 2.78)
throughput

iteration	min	sec	people/user
1	1	60	166.67 (2.78 * 60)
2	5	300	833.33
3	10	600	166.67
4	30	1800	5000
5	60	3600	10000

নথি iteration
একটি শুধু সহ
আবেদন করা নয়।
যাই 5 min break
বিনামূলে server বি
restart করা হবে।

Load test এর ফল দেখানে/ fail করার stress test
করতে না।

ফেল আগে time করে রাখা হবে, তাও করতে
না ফেল করতে।

load test for 1 ক্ষেত্রে application / server / db রয়ে পেরফোর্মেন্স কাছে হবে। তাহলে API load test কাছে হবে।

throughput
per sec
server
avg call
request
accept
বাস্তু

Report generate in Jmeter

jmeter -n -t BookAPI.jmx (test file name) -l BookAPI.jtl
-e -o reports → reports name বা মডেল ফলোর নাম করে
generate করবো।

log

class-11

API chaining with Jmeter

signup → login → Purchase product.

In response data: message: user found

আমরা এখন automatically করবো, আমা system কে ধূঁধাত হচ্ছে
কে কোন user found? কোটু কি check কো। তাই, এই
assertion. we must have to add assertion in any test
case (API testing, performance testing, web automation
etc.)

পৃষ্ঠা Postman কে আমরা base url collection variable করেছি।
আমা Jmeter কে user defined variables করে আপনার (thread
group → add → config element → user defined variables)

Create new user on jmeter

create new user → add → config element → random variable

(Interview)
Timer: কোটু ফিল্ট কোটু time range কো করে কোটু

সব কোটু request randomly send কো

Uniform random timer: প্রতিটি request কে সময়ে
আন্তরিকে request আবেক্ষণ করা মুশু নাগাবে।

Gaussian random timer: স্টেট dynamic.

total delay random এর value নিয়ে তাৰ মধ্য
medium থাকবে। সুজেন্টি randomized বী হৈছে

(2) median থাকবে (2) median এৰ ক্ষেত্ৰ depend
কোৱে gaussian random timer.

Deviation = 100, constant = 300

$$(300-100), (300+100)$$
$$= 200 \quad 400$$

200-400 range এৰ
মাঝি randomly
automate কৰে কৰিব
ৰ

Distributed load testing.

multiple pc থেকে
load কৰতে পাই, তাহে server pc থাকতো

হ'ল slave pc থাকতো হৰা পৰিকল্পনা
Some LAN/WLAN এৰ under নি থাকতো হ'ব কো

ম'ছ'ন্দের firewall disable কৰা থাকতো হ'ব / allow

কো থাকতো হ'ব, then
configure (slide)

Volume testing: DB এর সাথে যোগাযোগ স্থাপন করে এবং তার মধ্যে ডেটা ম্যানিপুলেট করে দেখতে পাবি কতক কিম্বা লোড করে পাবো (especially for monolithic software). microservice software গুলিকে API base হয়ে গেড়ে।

use of multiple thread group

withdraw, payment, deposit. একটি thread এর মতৃ ক্ষেত্রে বেশি threads এর মতৃ ক্ষেত্রে তা দ্বারা যথেষ্ট করা হবে।

Scenario

20 people → withdraw, payment, send money (loop এর মধ্যে একসাথে parallel হবে না, ৩মি আলাদা thread group নিতে হবে। withdraw, payment, send money)

class-13

my SQL database

Database Testing

Schema: Database design

Triggers: DB এর কোর্সর মাধ্যমে ফিল্টার, query execute করা এবং O/P এর পর্যাপ্ত করা।

Data integrity: কোর্সর table কিম্বা কোর্সর এর মধ্যে কোর্সর কিম্বা

Consistency: সামনে data ক্ষেত্রে আলাদা করা (যেখানে কোর্সর কিম্বা)

Relational db: এটা table কে আরেক ক্ষেত্রে আরেক table কে আপনাকে আপনাকে।