database needs to track what module user is at, and whether it’s a turn to take FL or MST

**database**

**collection: User**

{

userId: <unique id>,

shouldTake: [<FL/MST>, <FL/MST>],

moduleNumberToTake: <default: 1>,

score: {

fixed: <num>,

mst: <num>

},

answers:{

fixed: {

<questionId>: <selectedAnswer>,

...

}

mst: {

<questionId>: <selectedAnswer>,

..

}

}

}

**collection**: **Question**

{

id: <unique id>,

difficultyParameter: <between 0 and 1>,

type: <MCQ> or <short answer>,

moduleNumber: <Null>/<number>

content: “”,

options: [{optionId: “X&\*@#”,imgBlob: “”,},...]

correctAnswer: <optionId>/<correctShortAnswer>

}

**Check what user’s current state is**

GET /user/:userId

request: none

response: <**User** document>

**Get the fixed length question set**

GET /fixed

request: {}

response:

{

questions: [{

type: <MCQ> or <short answer>,

questionContent: “”,

questionOptions: [{optionId: “X&\*@#”,imgBlob: “”,},...]

}, …]

}

**Submit fixed length results**

POST /fixed

things it does:

gets user response set

updates fixed test score of User document

update answers of User document

pop shouldTake of User document

request:

{

userId: “”,

response:[

{

qid: “M19283”,

answer: <option number>/<short answer>

},

…

]

}

response: <none>

**Get question set of a specific module**

GET /mst/:moduleId

request: {}

response:

{

questions: [

{

type: <MCQ> or <short answer>

questionContent: “”,

questionOptions: [<option1>,<option2>,<option3>,<option4>]

},

…

]

}

**Submit a testlet response**

POST /mst

things it does:

gets user response

update the user collection with new responses

create 2D item parameter matrix and response vector by iterating through the mst answers of user

calls eapEST() for theta

then call nextModule()

check if this is nondefault stage to determine ‘finished’ of response

if finished=true, pop shouldTake of User doc

update User document (moduleNumberToTake, update score)

return the next module’s number

**eapEST:**

response vector,

1. 2D subset item bank - 1 column for each parameter (4 columns), 1 row for each item

2. response vector where response[i]=1 means item i (row i in item bank) in the item bank is correctly answered

**nextModule:**

1. 2D complete item bank - 1 column for each parameter (4 columns), 1 row for each item
2. modules binary matrix where modules[i][j]=1 means item i (row i in item bank) in the item bank belongs to module j
3. out vector, stores list of integers n where n is the item (row in item bank) that has already been answered

request:

{

userId: “”,

results:[

{

qid: “M19283”,

answer: <option number>/<short answer>

},

…

]

}

response:

{

finished: <true>/<false>,

nextModule: <module number>

}