

## Optimal CAT 2.2 Manual

Optimal CAT is an easy-to-use **production-grade** Computer Adaptive Testing (CAT) engine. It's both for psychometricians to run simulations and for IT engineers to deploy to production.

It builds upon the shadow-test approach. It leverages state-of-the-art Mathematical Programming solvers (FICO Xpress and/or Gurobi) to deliver **optimal** tests to each student (hence the name Optimal CAT). It guarantees that test blueprints and constraints are always met, and measurement accuracy is maximized.

The engine is built as a microservice (stateless REST API interface) and can be easily integrated with many test delivery systems.

Optimal CAT Core Team:

Jie Li, PhD

Wim J. van der Linden, PhD

Richard J. Patz, PhD

Support: [optimal-cat@outlook.com](mailto:optimal-cat@outlook.com)

## Contents

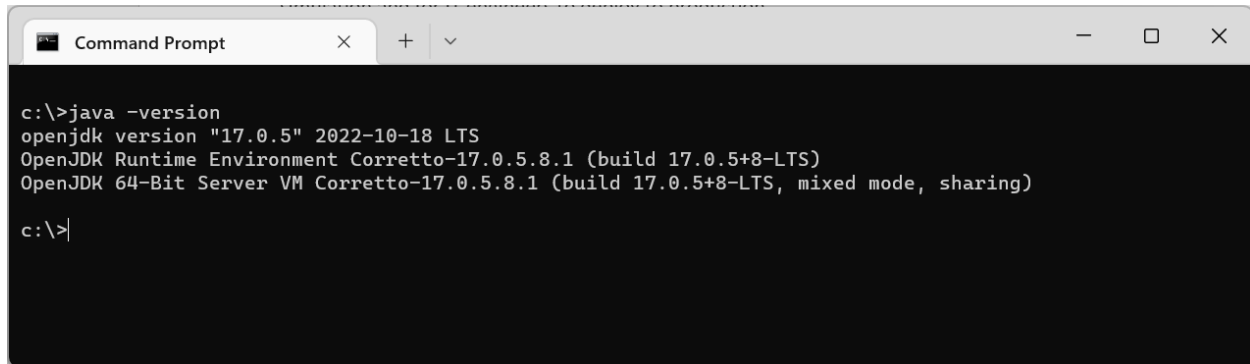
Optimal CAT 2.2 Manual .....	1
Optimal CAT License .....	3
Prerequisites .....	4
Getting Started Guide (Windows version, 30 minutes) .....	5
1. Test configuration.....	5
2. Start CAT engine .....	9
3. Run simulation (for Psychometricians).....	11
4. Run load test (for IT engineers) .....	15
Appendix I. Optimal CAT engine REST API .....	18
Appendix II. Optimal CAT engine IRT models and parameterization.....	23
1PL model.....	23
2PL model.....	23
3PL model.....	23
GPCM model .....	24
GRM model .....	24
Appendix III. Optimal CAT software libraries and licenses .....	25

## Optimal CAT License

	Use	Limitation and Support
<b>Optimal-CAT Free Version</b>  available on <a href="http://www.optimal-cat.com">www.optimal-cat.com</a>	Personal use and research use	Users need to install solver separately (FICO Xpress or Gurobi) before using Optimal-CAT  Current Optimal-CAT version expires: 10/31/2025 (download a new version to renew)  Max load 10 tests  Limited technical support
<b>Optimal-CAT Licensed Version</b>  shared with commercial users through private links	Commercial use	Solver and license are embedded in Optimal-CAT as OEM components (optimized for performance and deployment)  No max load or time limits  Support both Windows and Linux platform  Free training, consultation and full technical support  Please contact us for trial and pricing (solver license included)

## Prerequisites

To use the Optimal CAT engine, you need to have **Java 17 or above** installed on your computer. If you are not sure which Java to use, you could try Amazon Corretto (<https://aws.amazon.com/corretto>). Once you have installed Java, make sure you check the version.



```
c:\>java -version
openjdk version "17.0.5" 2022-10-18 LTS
OpenJDK Runtime Environment Corretto-17.0.5.8.1 (build 17.0.5+8-LTS)
OpenJDK 64-Bit Server VM Corretto-17.0.5.8.1 (build 17.0.5+8-LTS, mixed mode, sharing)

c:\>|
```

The Optimal CAT engine also uses **Microsoft Excel** (or Apache OpenOffice) as the editing tool for CAT test configurations.

**\* Optimal-CAT Free Version Users:** need to install solver software (Fico Xpress or Gurobi) before using Optimal-CAT. If you currently do not have a valid solver license, you could try FICO Xpress Community License or Gurobi Academics License (if you are an academic user).

FICO Xpress Community License (needs to install **version 9.5** or above)

<https://www.fico.com/en/fico-xpress-community-license>

Gurobi (needs to install **version 12.0.1**)

<https://www.gurobi.com/>

## Getting Started Guide (Windows version, 30 minutes)

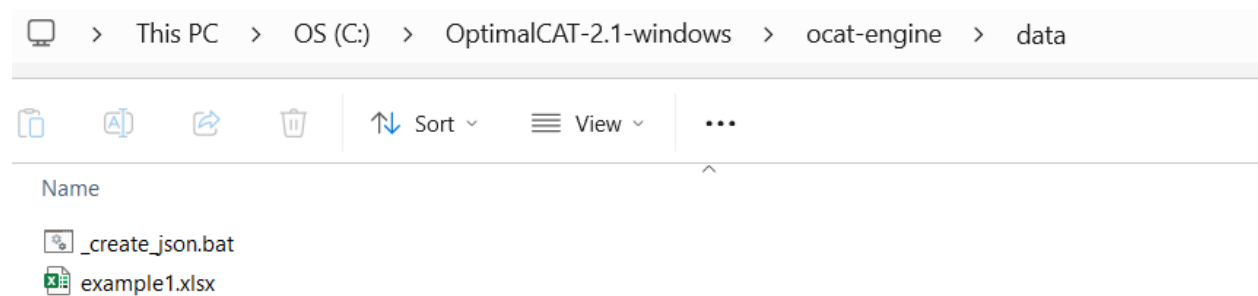
Download and unzip Optimal-CAT from [www.optimal-cat.com](http://www.optimal-cat.com). In the unzipped folder, you should see the following sub-folders.

- API\_examples
- ocat-engine
- ocat-loadtest
- ocat-simulator

- API\_examples: API examples in Java, Python and R <sup>1</sup>
- ocat-engine: Optimal CAT engine
- ocat-loadtest: Optimal CAT load test tool for IT engineers
- ocat-simulator: Optimal CAT simulator for psychometricians

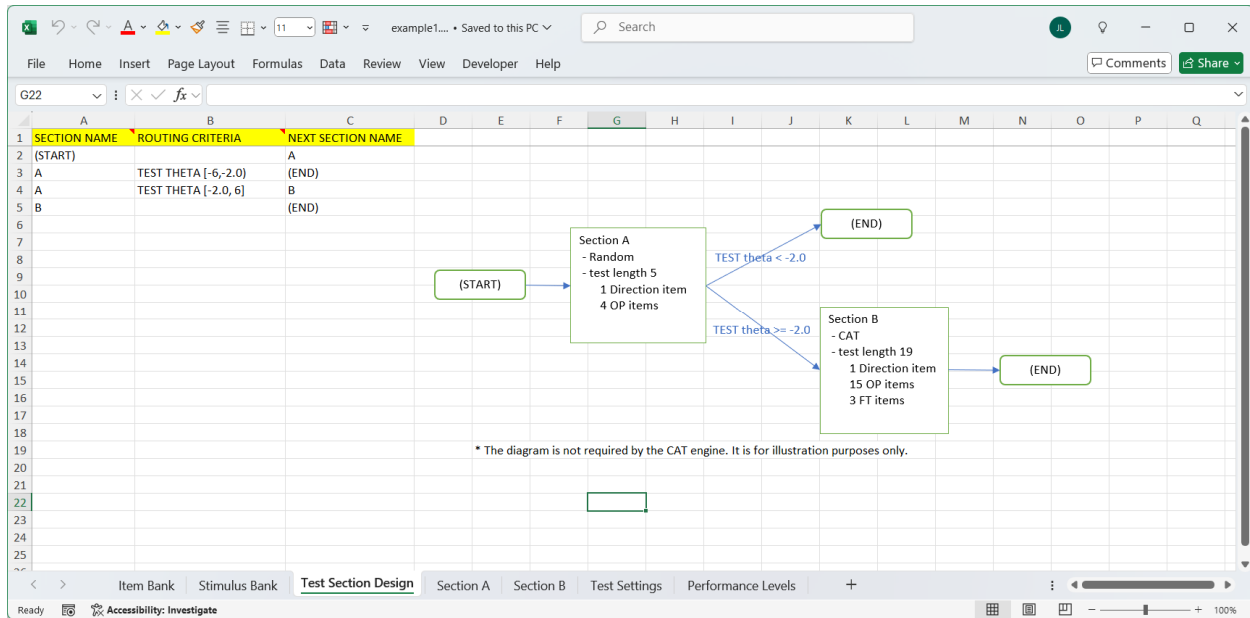
### 1. Test configuration

First, go to the “ocat-engine” folder, and then the “data” folder. There is one test configuration Excel file (example1.xlsx) in the folder. You could have multiple test configuration Excel files, each for one test.



---

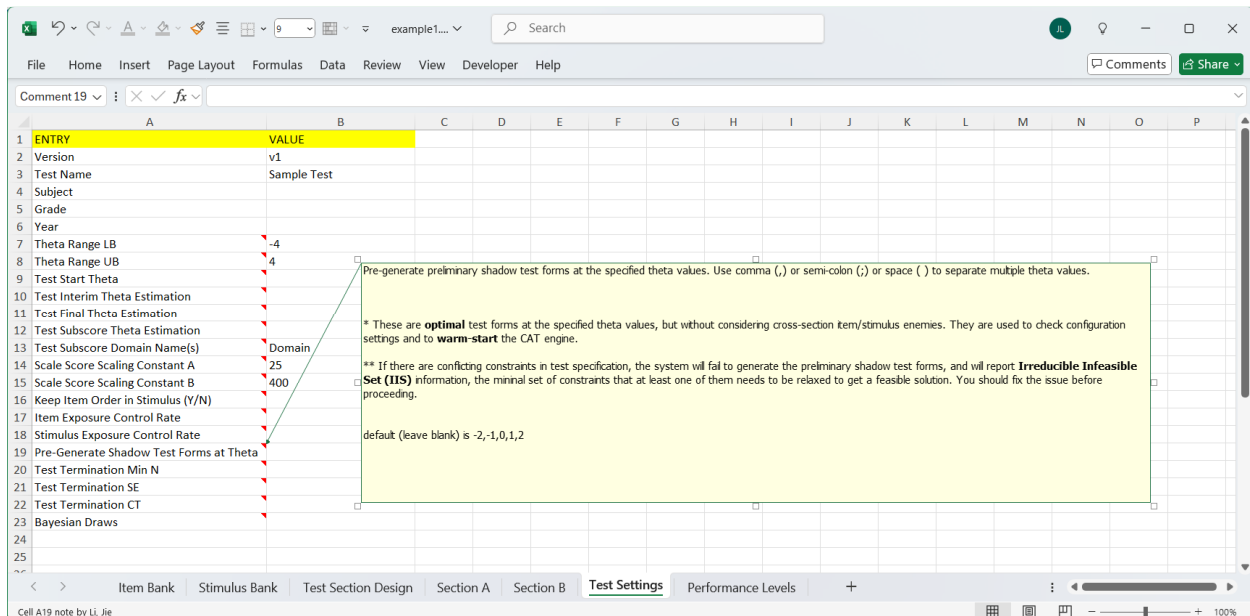
<sup>1</sup> No programming is needed for psychometricians to set up CAT tests and run simulations. These API examples are provided to assist IT engineers to integrate the Optimal-CAT engine into your testing delivery systems. You may also use these examples to build your own simulation client.



A quick overview of the sheets in test configuration files:

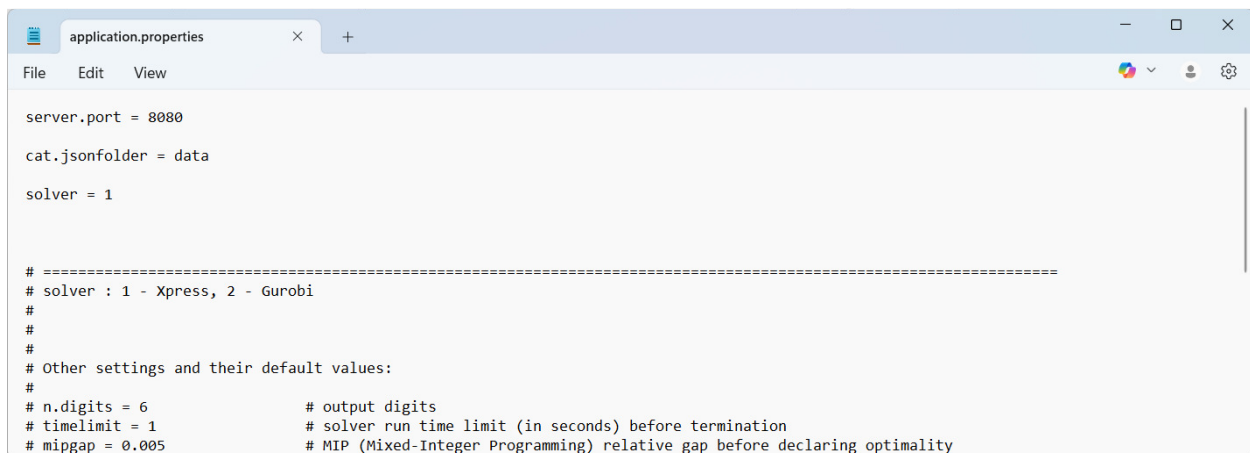
- **Item Bank** sheet contains item info. Yellow highlighted columns are required by the system. You could add any other columns you need, and you could use them later in Constraint definition.
  - The engine supports the following IRT models: 1PL, 2PL, 3PL, GPCM and GRM.
- **Stimulus Bank** sheet contains stimulus info.
- **Test Section Design** sheet contains test section routing info. You can have many test sections and define routing criteria from one section to another. This allows building many test configurations like Multi-Stage Testing (MST). There are two default dummy sections - (START) and (END) - to designate the beginning and the end of a test.
- **Section A, Section B ... (or simply A, B...)** for each section defined in Test Section Design sheet, there should be a corresponding sheet for the section, in which you define things like test format for the section. The engine supports the following test formats:
  - CAT - item/unit level CAT using Maximum Fisher Information
  - CAT\_BM - item/unit level CAT using b-param matching
  - Random - linear test, item order is random
  - Linear - linear test, item order specified by user
  - LOFT - linear on-the-fly test, this is a special case of CAT with no test form refresh
- **Test Settings** sheet contains test level parameters, e.g., theta range, scale score constants, etc.
- **Performance Levels** sheet contains cut scores and performance levels definition.

The **cell comments** (a pop-up message when moving over the cell with a red triangle) provide useful tips about the fields.




Once you have test specifications defined in Excel configuration files, the next step is to run **\_create\_json.bat**.


**\* Optimal-CAT Free Version Users:** Before you run **\_create\_json.bat**, you need to go to “ocat-engine” folder, make sure the “solver” setting in the ***application.properties*** file (1 – Xpress, 2 – Gurobi) matches the solver installed on your computer.





After running ***\_create\_json.bat***, you should see a log file (log.txt) and a JSON file (example1.json) for each test. Check the screen output and the log file (log.txt) carefully to make sure no error is reported. Otherwise, you should correct errors and then re-run.

Name

 \_create\_json.bat

 example1.json

 example1.xlsx

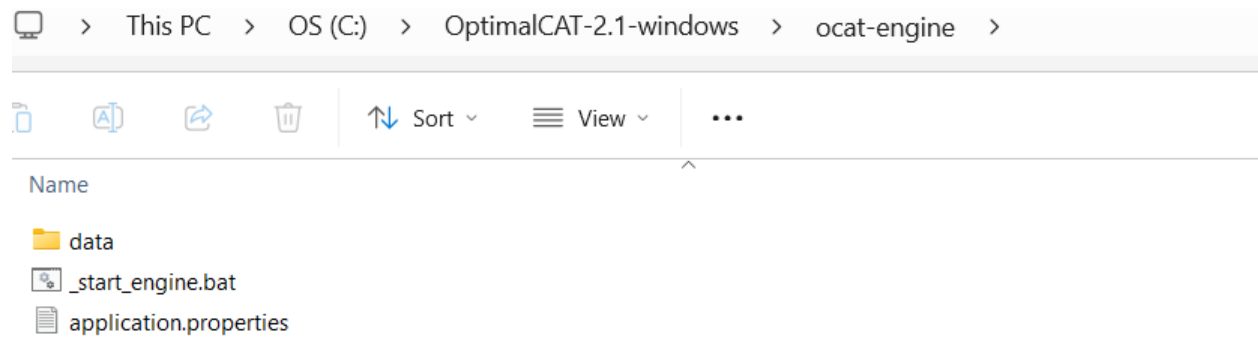
 log.txt

**Notice:** the CAT engine reads only JSON files, not test configuration Excel files.

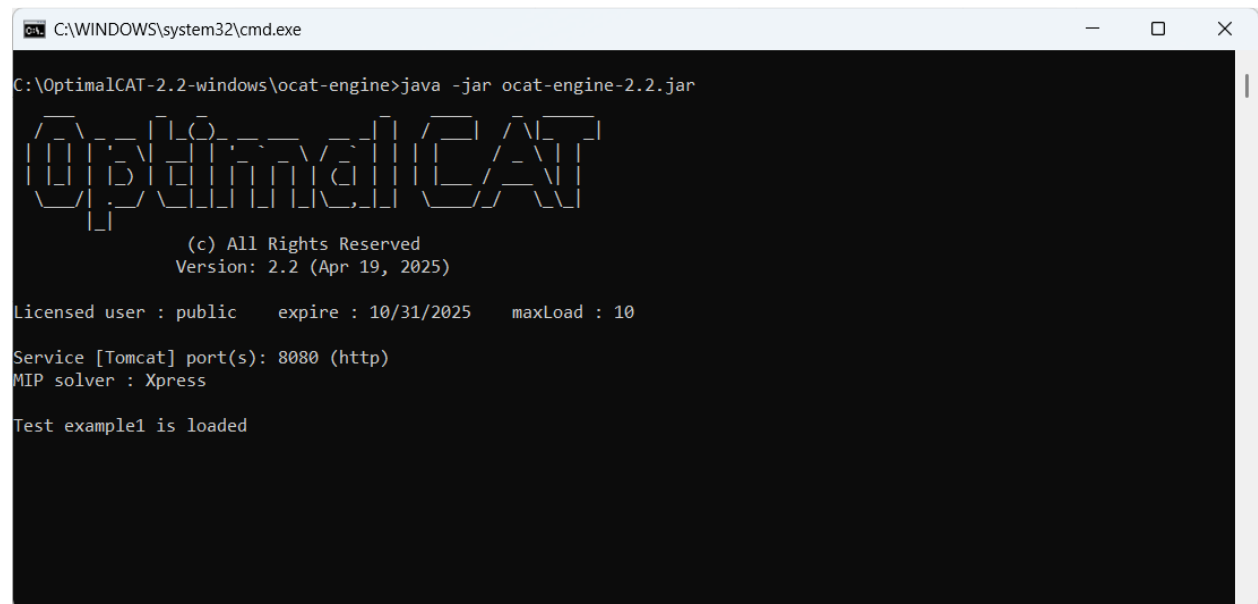


## 2. Start CAT engine

To start the CAT engine, go to “ocat-engine” folder and run **\_start\_engine.bat**.



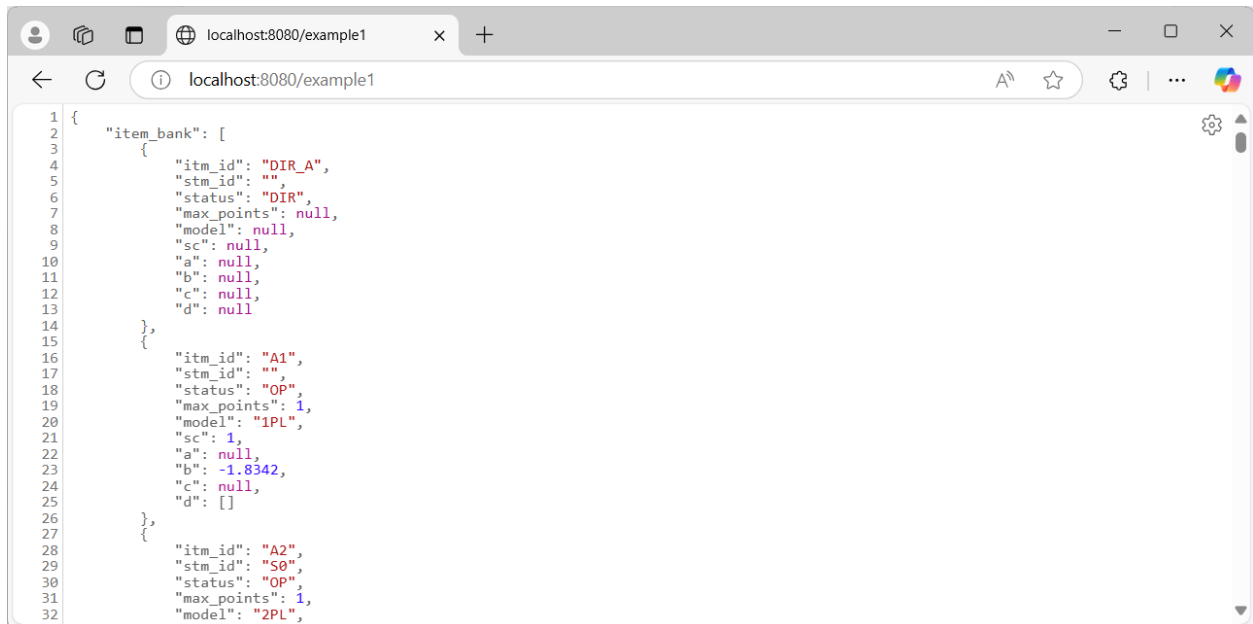
You should then see a command window like below. It shows that the engine is started and listening on port 8080, MIP solver is Xpress or Gurobi (depending on the settings in **application.properties** file) and test example1 is loaded. Leave the command window there. It's now ready to process incoming API calls.



The CAT engine has the following public REST API endpoints (see Appendix for details).

- / - HTTP GET request. It returns basic info of the engine and the loaded tests.
- /{testID} - HTTP GET request. It returns the test JSON file.
- /IEC/{testID} - HTTP GET request. It returns item/stimulus exposure counts.
- /cat - HTTP POST request. It returns the next items to administer and final scores.
- /rescore - HTTP POST request. It is for rescoring.

For GET request endpoints, you could use a browser to access them, for example,



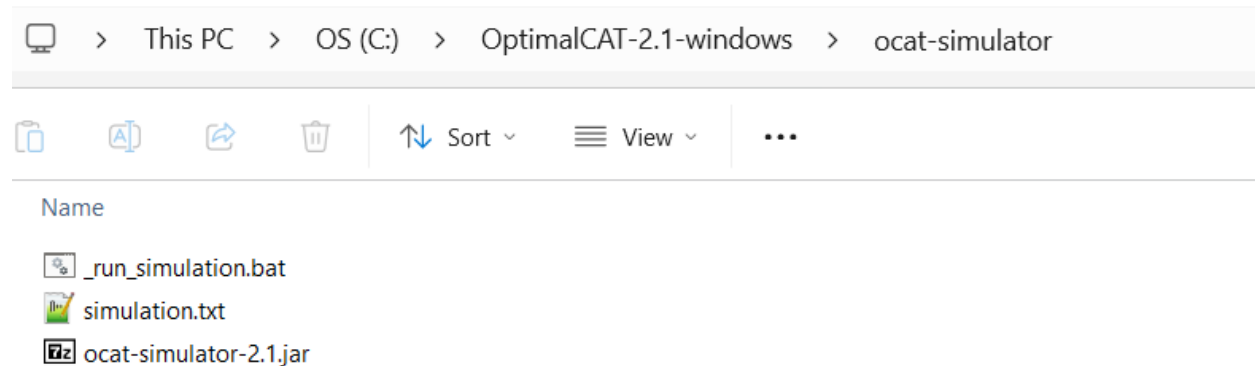
A screenshot of a web browser window. The address bar shows 'localhost:8080/example1'. The page content displays a JSON response, which is a list of items under the key 'item\_bank'. The JSON is formatted with syntax highlighting and line numbers on the left side of the editor. The response contains three items: 'DIR\_A', 'A1', and 'A2'. Each item has various attributes like 'itm\_id', 'stm\_id', 'status', 'max\_points', 'model', 'sc', 'a', 'b', 'c', and 'd'.

```
1 {
2   "item_bank": [
3     {
4       "itm_id": "DIR_A",
5       "stm_id": "",
6       "status": "DIR",
7       "max_points": null,
8       "model": null,
9       "sc": null,
10      "a": null,
11      "b": null,
12      "c": null,
13      "d": null
14    },
15    {
16      "itm_id": "A1",
17      "stm_id": "",
18      "status": "OP",
19      "max_points": 1,
20      "model": "1PL",
21      "sc": 1,
22      "a": null,
23      "b": -1.8342,
24      "c": null,
25      "d": []
26    },
27    {
28      "itm_id": "A2",
29      "stm_id": "S0",
30      "status": "OP",
31      "max_points": 1,
32      "model": "2PL",
```

For POST request endpoints (/cat and /rescore), they are for IT engineers to integrate with test delivery systems for test administration and scoring.

### 3. Run simulation (for Psychometricians)

Now the CAT engine is up-running, you could start simulation. Go to “ocat-simulator” folder



The *simulation.txt* file defines the parameters to run the simulation.

```
1 # Comments start with #
2
3
4 testID = example1
5
6
7 # Generate response files (optional)
8 # response.1 = student1.csv 100 FIXED(-2,-1,0,1,2)           # 100 students each at the fixed points -2, ..., 2, save to student1.csv
9 # response.2 = student2.csv 100 NORMAL(0,1)                 # 100 students from Normal(0,1) distribution, save to student2.csv
10 # response.3 = student3.csv 100 UNIFORM(-3,3)                # 100 students from Uniform(-3,3) distribution, save to student3.csv
11 #
12 # Simulator runs in multiple threads: thread.1, thread.2, ... (must start from thread.1)
13 # thread.1 = student1.csv
14 # thread.2 = 100 FIXED(-2,-1,0,1,2)
15 # thread.3 = 100 NORMAL(0,1)
16 # thread.4 = 100 UNIFORM(-3,3)
17
18
19 thread.1 = 100 FIXED(-2,-1.5,-1,-0.5,0,0.5,1,1.5,2)
20 thread.2 = 100 FIXED(-2,-1.5,-1,-0.5,0,0.5,1,1.5,2)
21 thread.3 = 100 FIXED(-2,-1.5,-1,-0.5,0,0.5,1,1.5,2)
22 thread.4 = 100 FIXED(-2,-1.5,-1,-0.5,0,0.5,1,1.5,2)
23 thread.5 = 100 FIXED(-2,-1.5,-1,-0.5,0,0.5,1,1.5,2)
24
25
26
27 # engine.url = localhost:8080           # simulation (fast mode) only runs on localhost
28 # random.seed = 12345                   # random.seed = 0 is no random seed
29
30
31
32
```

The Notepad++ window displays the 'simulation.txt' file. The status bar at the bottom indicates the file is a 'Normal text file' with a length of 1,146 characters, 32 lines, and is encoded in UTF-8.

Run **\_run\_simulation.bat** to start the simulation. Once it's completed, you should see the output below.

```
C:\WINDOWS\system32\cmd.exe

C:\OptimalCAT-2.1-windows\ocat-simulator>java -jar ocat-simulator-2.1.jar simulation.txt

engine.url = http://localhost:8080
testID = example1

thread.2 : 02/15/25 14:37:18 --- sent to server : 100 FIXED(-2,-1.5,-1,-0.5,0,0.5,1,1.5,2)
thread.4 : 02/15/25 14:37:18 --- sent to server : 100 FIXED(-2,-1.5,-1,-0.5,0,0.5,1,1.5,2)
thread.3 : 02/15/25 14:37:18 --- sent to server : 100 FIXED(-2,-1.5,-1,-0.5,0,0.5,1,1.5,2)
thread.5 : 02/15/25 14:37:18 --- sent to server : 100 FIXED(-2,-1.5,-1,-0.5,0,0.5,1,1.5,2)
thread.1 : 02/15/25 14:37:18 --- sent to server : 100 FIXED(-2,-1.5,-1,-0.5,0,0.5,1,1.5,2)

Run Statistics: 1.0 minutes, 4500 students

Saving files ...
Saving files ... done

C:\OptimalCAT-2.1-windows\ocat-simulator>pause
Press any key to continue . . .
```

The simulation is run on the server side (fast mode, to save the API traffic between client and server). While the simulation is running, you could see its progress in the Server window.

```
C:\WINDOWS\system32\cmd.exe

C:\OptimalCAT-2.1-windows\ocat-engine>java -jar ocat-engine-2.1.jar

OptimalCAT
(c) All Rights Reserved
Version: 2.1 (Feb 15, 2025)

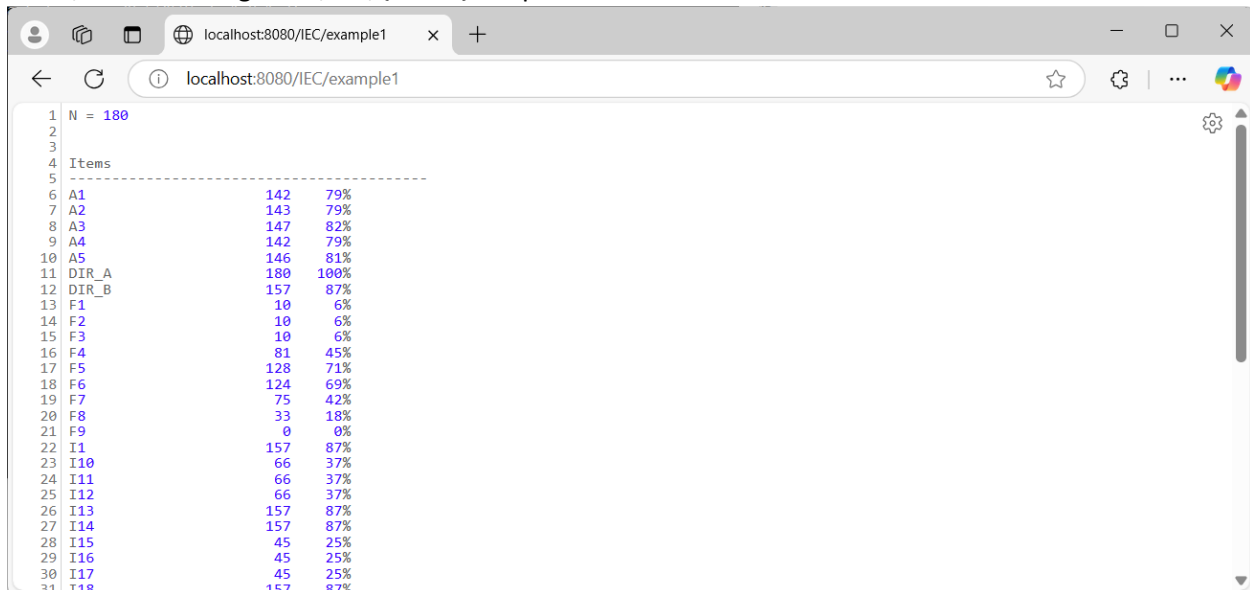
Licensed user : public    expire : 10/31/2025    maxLoad : 10

Service [Tomcat] port(s): 8080 (http)

Test example1 is loaded






[example1] thread.5 : 02/15/25 14:37:18 --- started
[example1] thread.2 : 02/15/25 14:37:18 --- started
[example1] thread.4 : 02/15/25 14:37:18 --- started
[example1] thread.3 : 02/15/25 14:37:18 --- started
[example1] thread.1 : 02/15/25 14:37:18 --- started
[example1] thread.1 : 02/15/25 14:37:24 --- 100/900      62 ms/test
[example1] thread.3 : 02/15/25 14:37:24 --- 100/900      64 ms/test
[example1] thread.2 : 02/15/25 14:37:25 --- 100/900      65 ms/test
[example1] thread.5 : 02/15/25 14:37:25 --- 100/900      65 ms/test
[example1] thread.4 : 02/15/25 14:37:25 --- 100/900      66 ms/test
[example1] thread.1 : 02/15/25 14:37:30 --- 200/900      56 ms/test
```

While the simulation is running, you could also check the item/stimulus exposure counts and turned-off items/stimuli through the /IEC/{testID} endpoint.



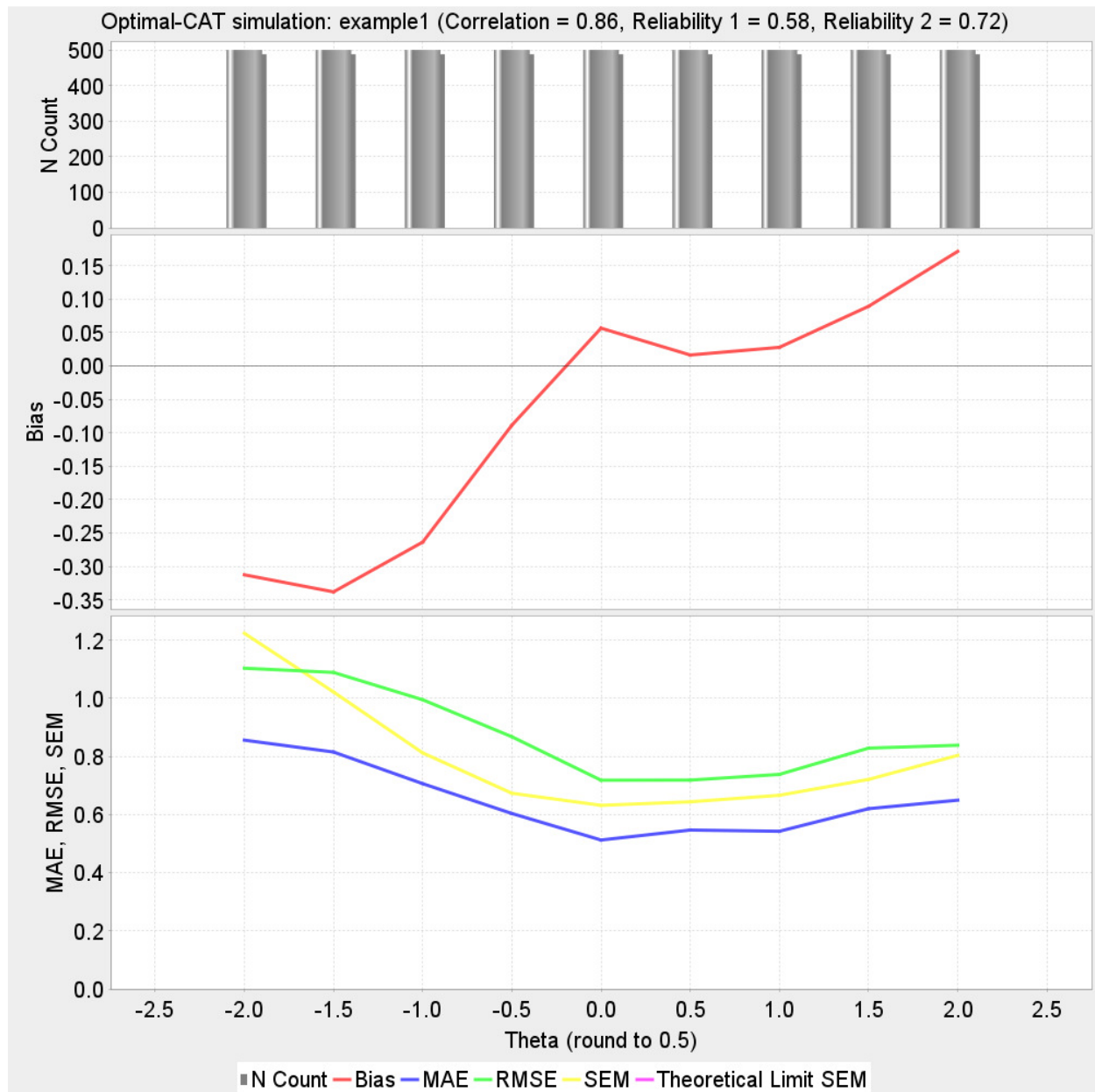
Item	Count	Percentage
N	180	
Items		
A1	142	79%
A2	143	79%
A3	147	82%
A4	142	79%
A5	146	81%
DIR_A	180	100%
DIR_B	157	87%
F1	10	6%
F2	10	6%
F3	10	6%
F4	81	45%
F5	128	71%
F6	124	69%
F7	75	42%
F8	33	18%
F9	0	0%
I1	157	87%
I10	66	37%
I11	66	37%
I12	66	37%
I13	157	87%
I14	157	87%
I15	45	25%
I16	45	25%
I17	45	25%
T18	157	87%

After the simulation run is completed, you should see results in a newly created folder named as {testid}\_{date}\_{time}. There are five files in the folder:

-  cat.csv
-  exposure.csv
-  score.csv
-  statistics.csv
-  statistics.png

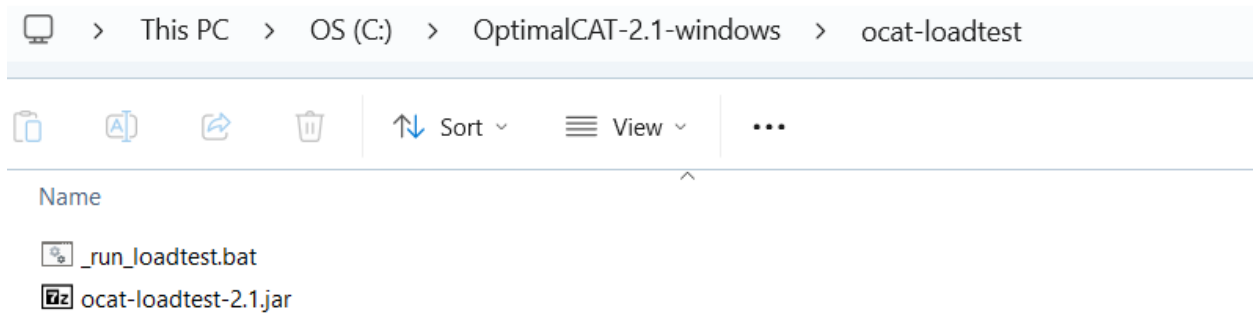
- cat.csv – this file contains detailed test session information, item by item.
- exposure.csv – this file contains exposure counts for both items and stimuli.
- score.csv – this file contains final scores and sub-scores.
- statistics.csv – this file contains the following measurement statistics
  - N Count
  - Bias
  - MAE - Mean Absolute Error
  - RMSE - Root Mean Square Error
  - SEM (or Bayesian\_SD) - Standard Error of Measurement (or Standard Deviation in Bayesian CAT)
  - Theoretical Limit SEM – it is calculated by aggregating all the pre-generated shadow test forms. **It is shown only when the CAT engine detects that all students see all the test sections defined in the configuration Excel file;** otherwise, the Theoretical Limit SEM is not shown (you must manually aggregate only the test sections that the student routes through to find out).

- Correlation of true theta ( $\theta_j$ ) and estimated theta ( $\hat{\theta}_j$ )
- Reliability 1 - defined as  $\rho^2 = \frac{Var(\theta_j)}{Var(\hat{\theta}_j)}$
- Reliability 2 - defined as  $\widehat{\rho^2} = 1 - \frac{E(CSEM_{\hat{\theta}_j}^2)}{Var(\hat{\theta}_j)}$ , where  $CSEM_{\hat{\theta}_j}$  is the Conditional Standard Error of Measurement (CSEM) at the estimated ability  $\hat{\theta}_j$
- Statistics.png – chart using the numbers in statistics.csv

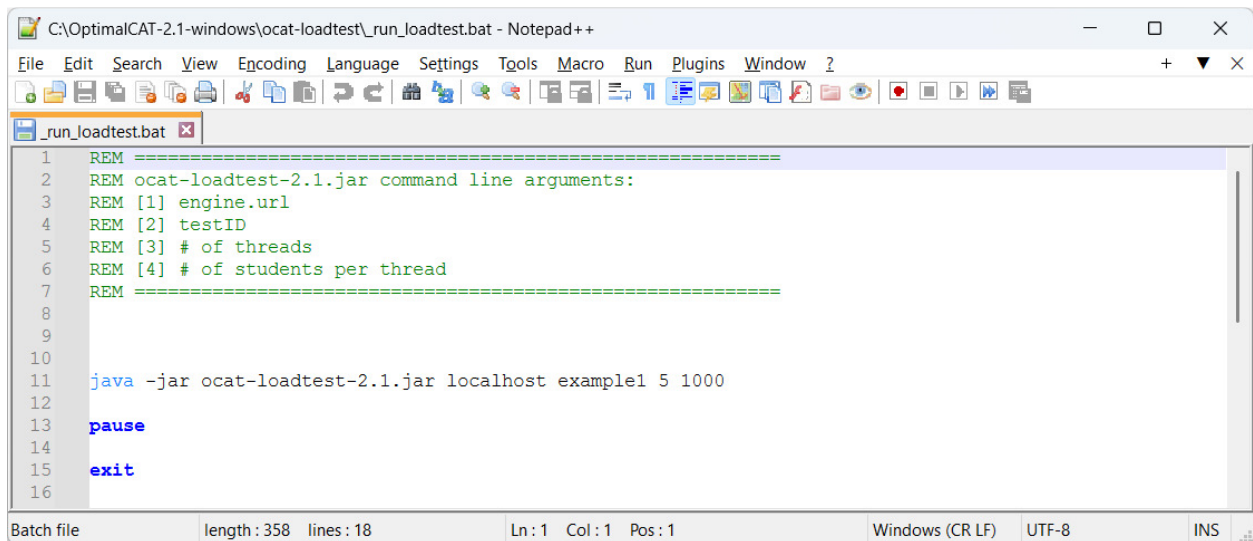


#### 4. Run load test (for IT engineers)

To get an idea of the actual response time of engine API calls in real operations, a load test tool is provided in “ocat-loadtest” folder.



You could change the command line arguments to fit your load test needs.



```
1 REM =====
2 REM ocat-loadtest-2.1.jar command line arguments:
3 REM [1] engine.url
4 REM [2] testID
5 REM [3] # of threads
6 REM [4] # of students per thread
7 REM =====
8
9
10
11 java -jar ocat-loadtest-2.1.jar localhost example1 5 1000
12
13 pause
14
15 exit
16
```

The Notepad++ window shows the batch file content. It includes comments for command line arguments and a Java command to run the load test. The status bar at the bottom indicates the file is a Batch file, 358 characters long, 18 lines, and uses Windows (CR LF) encoding with UTF-8 character set.

Run **\_run\_loadtest.bat** to start a load test. You should see a window like below.

```
C:\WINDOWS\system32\cmd.exe

C:\OptimalCAT-2.1-windows\ocat-loadtest>REM =====
C:\OptimalCAT-2.1-windows\ocat-loadtest>REM ocat-loadtest-2.1.jar command line arguments:
C:\OptimalCAT-2.1-windows\ocat-loadtest>REM [1] engine.url
C:\OptimalCAT-2.1-windows\ocat-loadtest>REM [2] testID
C:\OptimalCAT-2.1-windows\ocat-loadtest>REM [3] # of threads
C:\OptimalCAT-2.1-windows\ocat-loadtest>REM [4] # of students per thread
C:\OptimalCAT-2.1-windows\ocat-loadtest>REM =====
C:\OptimalCAT-2.1-windows\ocat-loadtest>java -jar ocat-loadtest-2.1.jar localhost example1 5 1000

engine.url = http://localhost:8080
testID = example1
threads = 5
students/thread = 1000

[example1] thread.1 : 02/15/25 15:07:59 --- started
[example1] thread.2 : 02/15/25 15:07:59 --- started
[example1] thread.5 : 02/15/25 15:07:59 --- started
[example1] thread.3 : 02/15/25 15:07:59 --- started
[example1] thread.4 : 02/15/25 15:07:59 --- started
[example1] thread.4 : 02/15/25 15:08:30 --- 100/1000 students, 1606 API calls, ~19.4 ms/call (client side)
[example1] thread.3 : 02/15/25 15:08:30 --- 100/1000 students, 1606 API calls, ~19.6 ms/call (client side)
[example1] thread.2 : 02/15/25 15:08:30 --- 100/1000 students, 1625 API calls, ~19.5 ms/call (client side)
[example1] thread.1 : 02/15/25 15:08:31 --- 100/1000 students, 1663 API calls, ~19.6 ms/call (client side)
[example1] thread.5 : 02/15/25 15:08:34 --- 100/1000 students, 1796 API calls, ~19.5 ms/call (client side)
[example1] thread.3 : 02/15/25 15:09:02 --- 200/1000 students, 3250 API calls, ~19.6 ms/call (client side)
[example1] thread.4 : 02/15/25 15:09:04 --- 200/1000 students, 3326 API calls, ~19.9 ms/call (client side)
[example1] thread.1 : 02/15/25 15:09:05 --- 200/1000 students, 3364 API calls, ~19.6 ms/call (client side)
[example1] thread.2 : 02/15/25 15:09:05 --- 200/1000 students, 3402 API calls, ~19.6 ms/call (client side)
[example1] thread.5 : 02/15/25 15:09:08 --- 200/1000 students, 3516 API calls, ~19.6 ms/call (client side)
```

At the end of the load test run, it reports the engine response time (average milliseconds per API call) from both server side and client side. Based on our experience, most of the real CAT tests have response time below 100 milliseconds. The larger the item bank and the more complicated the test specifications, the longer the response time.

```
C:\WINDOWS\system32\cmd.exe

[example1] thread.5 : 02/15/25 15:12:54 --- 900/1000 students, 15252 API calls, ~19.1 ms/call (client side)
[example1] thread.2 : 02/15/25 15:13:18 --- 1000/1000 students, 16516 API calls, ~19.2 ms/call (client side)
[example1] thread.1 : 02/15/25 15:13:23 --- 1000/1000 students, 16801 API calls, ~19.1 ms/call (client side)
[example1] thread.4 : 02/15/25 15:13:24 --- 1000/1000 students, 16801 API calls, ~19.2 ms/call (client side)
[example1] thread.5 : 02/15/25 15:13:25 --- 1000/1000 students, 16877 API calls, ~19.2 ms/call (client side)
[example1] thread.3 : 02/15/25 15:13:27 --- 1000/1000 students, 16991 API calls, ~19.0 ms/call (client side)

Run Statistics: 5.5 minutes, 5000 students, 83986 API calls, ~3.9 ms/call (server side), ~19.5 ms/call (client side)

C:\OptimalCAT-2.1-windows\ocat-loadtest>pause
Press any key to continue . . .
```



**Notice:** The load test results heavily depend on the hardware and software it runs on. We recommend using **Linux** servers for operation. Microsoft Windows Home/Professional edition is not the best OS to host web services. The CPU utilization varies a lot from time to time due to various background processes in Windows. Also, running on Linux is a lot faster.

A typical load test scenario: IT engineers set up Optimal CAT engine on a Linux server, and then set up 10 or 20 computers as clients each running 10 or 20 threads to stress test the CAT engine and see how many servers they would need for operational tests.

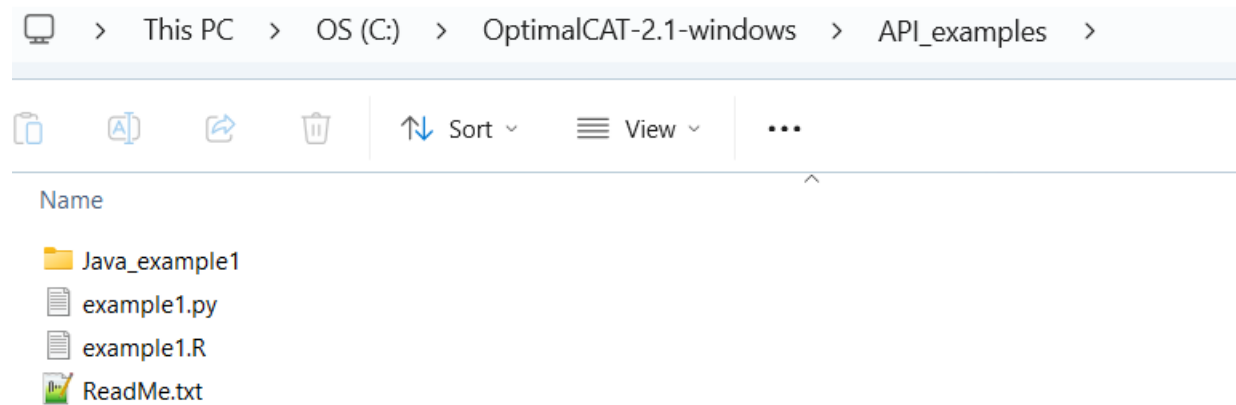
For most assessment programs, we believe one or two powerful servers should be enough. For very large testing programs with high concurrency requirements, we suggest creating a server pool and putting a load balancer in front, e.g., using [Load Balancer - Amazon Elastic Load Balancer \(ELB\) - AWS](#).

## Appendix I. Optimal CAT engine REST API

The CAT engine has the following public REST API endpoints:

- / - HTTP GET request. It returns basic info of the engine and the loaded tests.
- /{testID} - HTTP GET request. It returns the test JSON file.
- /IEC/{testID} - HTTP GET request. It returns item/stimulus exposure counts.
- /cat - HTTP POST request. It returns the next items to administer and final scores.
- /rescore - HTTP POST request. It is for rescoring.

There are examples (written in Java, Python and R) in “API\_examples” folder for your reference.



For example, this is the Python API example (example1.py).

```
1 import requests, json, random
2
3
4 # init HTTP body
5 body = {'testID': 'example1',
6         'administeredItems': [],
7         'sessionData': None}
8
9
10
11 # first call to /cat endpoint, display return
12 d = requests.post(url='http://localhost:8080/cat', json=body).json()
13 # print(json.dumps(d, indent=4))
14
15
16
17 # a complete test - repeated call to /cat until testCompleted=True
18 call_counter = 1
19 while (d['status'] != 'OK' and d['testCompleted'] == False):
20     for adminitem in d['nextItemsToAdminister']:
21         adminitem['itemScore'] = random.randrange(2) # assign a random 0 or 1 score
22         body['administeredItems'].append(adminitem) # append the administered item
23         print(call_counter, adminitem)
24
25     body['sessionData'] = d['sessionData'] # always post back sessionData
26
27     d = requests.post(url='http://localhost:8080/cat', json=body).json()
28     call_counter += 1
29
30
31 # score
32 print(d['score']['overall'])
33
34
35
36
37
```

IPython 8.7.0 -- An enhanced Interactive Python.

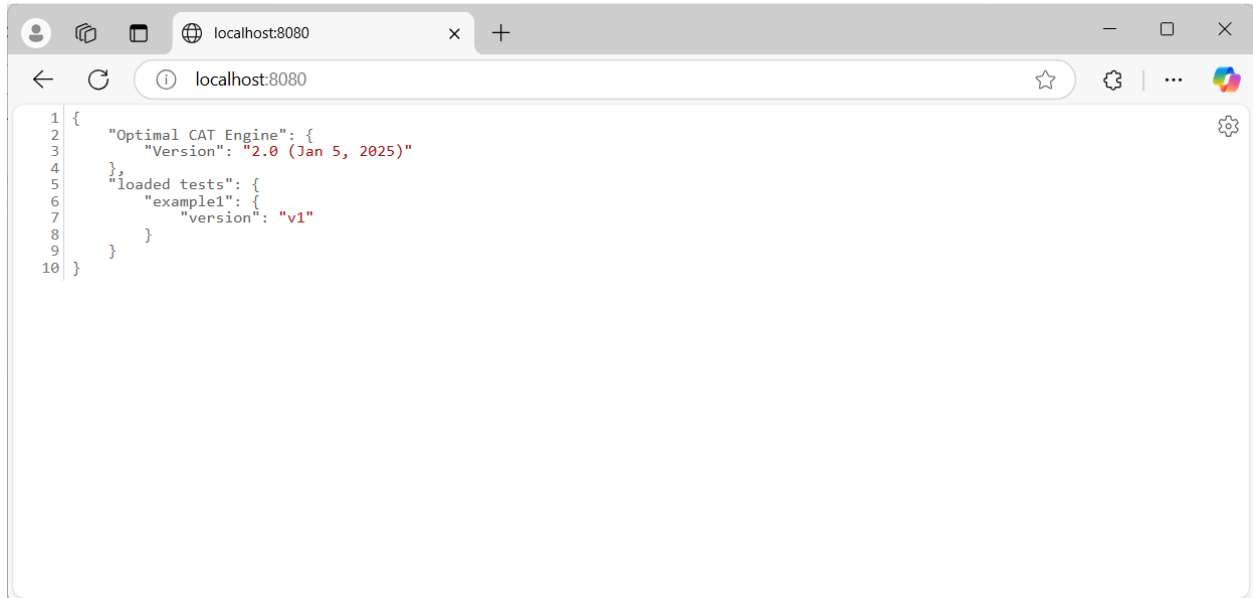
```
In [1]: runfile('C:/OptimalCAT-2.1-windows/API_examples/example1.py', wdir='C:/OptimalCAT-2.1-windows/API_examples')
1 {'slot': 1, 'sectionID': 'A', 'itemID': 'DIR_A', 'stimulusID': '', 'itemScore': 1}
1 {'slot': 2, 'sectionID': 'A', 'itemID': 'A1', 'stimulusID': '', 'itemScore': 0}
1 {'slot': 3, 'sectionID': 'A', 'itemID': 'A4', 'stimulusID': '', 'itemScore': 1}
1 {'slot': 4, 'sectionID': 'A', 'itemID': 'A2', 'stimulusID': 'S0', 'itemScore': 0}
1 {'slot': 5, 'sectionID': 'A', 'itemID': 'A3', 'stimulusID': 'S0', 'itemScore': 1}
2 {'slot': 6, 'sectionID': 'B', 'itemID': 'DIR_B', 'stimulusID': '', 'itemScore': 0}
3 {'slot': 7, 'sectionID': 'B', 'itemID': 'I20', 'stimulusID': '', 'itemScore': 0}
4 {'slot': 8, 'sectionID': 'B', 'itemID': 'I14', 'stimulusID': '', 'itemScore': 1}
5 {'slot': 9, 'sectionID': 'B', 'itemID': 'F8', 'stimulusID': '', 'itemScore': 1}
6 {'slot': 10, 'sectionID': 'B', 'itemID': 'F5', 'stimulusID': '', 'itemScore': 1}
7 {'slot': 11, 'sectionID': 'B', 'itemID': 'I12', 'stimulusID': 'S3', 'itemScore': 1}
8 {'slot': 12, 'sectionID': 'B', 'itemID': 'I11', 'stimulusID': 'S3', 'itemScore': 0}
9 {'slot': 13, 'sectionID': 'B', 'itemID': 'I10', 'stimulusID': 'S3', 'itemScore': 1}
10 {'slot': 14, 'sectionID': 'B', 'itemID': 'I5', 'stimulusID': '', 'itemScore': 1}
11 {'slot': 15, 'sectionID': 'B', 'itemID': 'F6', 'stimulusID': '', 'itemScore': 0}
12 {'slot': 16, 'sectionID': 'B', 'itemID': 'I23', 'stimulusID': 'S5', 'itemScore': 0}
13 {'slot': 17, 'sectionID': 'B', 'itemID': 'I21', 'stimulusID': 'S5', 'itemScore': 1}
14 {'slot': 18, 'sectionID': 'B', 'itemID': 'I22', 'stimulusID': 'S5', 'itemScore': 0}
15 {'slot': 19, 'sectionID': 'B', 'itemID': 'I13', 'stimulusID': '', 'itemScore': 0}
16 {'slot': 20, 'sectionID': 'B', 'itemID': 'I1', 'stimulusID': '', 'itemScore': 1}
17 {'slot': 21, 'sectionID': 'B', 'itemID': 'I19', 'stimulusID': '', 'itemScore': 0}
18 {'slot': 22, 'sectionID': 'B', 'itemID': 'I18', 'stimulusID': '', 'itemScore': 1}
19 {'slot': 23, 'sectionID': 'B', 'itemID': 'I19', 'stimulusID': '', 'itemScore': 1}
20 {'slot': 24, 'sectionID': 'B', 'itemID': 'I25', 'stimulusID': '', 'itemScore': 0}
{'domainValue': 'Overall', 'itemCount': 19, 'rawScore': 10, 'maxPoints': 22, 'theta': -1.381414, 'se': 0.616354, 'scaleScore': 365, 'scaleScore_se': 15, 'performanceLevel': 'Nearly Met'}
```

In [2]:

**GET** /

Parameters: (none)

Returns: basic info of Optimal CAT engine and loaded test

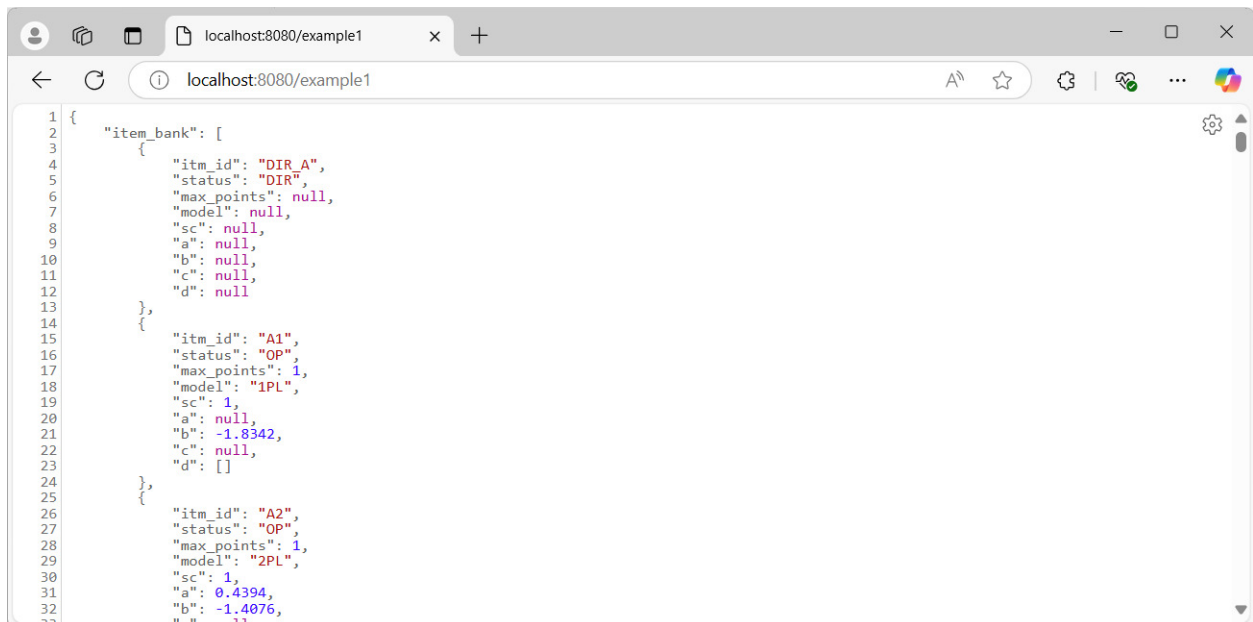


```
1 {
2   "Optimal CAT Engine": {
3     "Version": "2.0 (Jan 5, 2025)"
4   },
5   "loaded tests": {
6     "example1": {
7       "version": "v1"
8     }
9   }
10 }
```

**GET** /{testID}

Parameters: testID String

Returns: test JSON file, including item bank, test sections, test settings, etc.

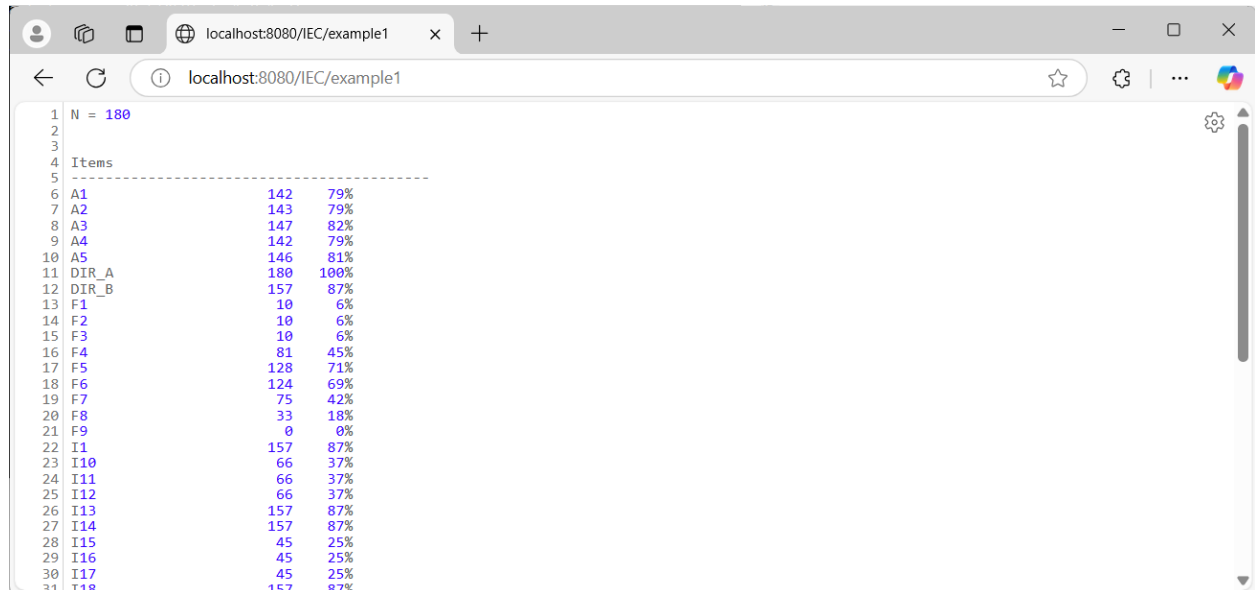


```
1 {
2   "item_bank": [
3     {
4       "itm_id": "DIR_A",
5       "status": "DIR",
6       "max_points": null,
7       "model": null,
8       "sc": null,
9       "a": null,
10      "b": null,
11      "c": null,
12      "d": null
13    },
14    {
15      "itm_id": "A1",
16      "status": "OP",
17      "max_points": 1,
18      "model": "IPL",
19      "sc": 1,
20      "a": null,
21      "b": -1.8342,
22      "c": null,
23      "d": []
24    },
25    {
26      "itm_id": "A2",
27      "status": "OP",
28      "max_points": 1,
29      "model": "2PL",
30      "sc": 1,
31      "a": 0.4394,
32      "b": -1.4076,
33      "c": null,
34      "d": []
35    }
36  ]
37 }
```

## GET IEC/{testID}

Parameters: testID String

Returns: item and stimulus exposure counts



```
1 N = 180
2
3
4 Items
5 -----
6 A1 142 79%
7 A2 143 79%
8 A3 147 82%
9 A4 142 79%
10 A5 146 81%
11 DIR_A 180 100%
12 DIR_B 157 87%
13 F1 10 6%
14 F2 10 6%
15 F3 10 6%
16 F4 81 45%
17 F5 128 71%
18 F6 124 69%
19 F7 75 42%
20 F8 33 18%
21 F9 0 0%
22 I1 157 87%
23 I10 66 37%
24 I11 66 37%
25 I12 66 37%
26 I13 157 87%
27 I14 157 87%
28 I15 45 25%
29 I16 45 25%
30 I17 45 25%
31 I18 157 87%
```

## POST /CAT - for test administration (what is the next item?)

Header: Content-Type = application/json

Parameters: (none)

Body: JSON format of the following fields:

- **testID** String valid testID
- **administeredItems** list of administered item information, as follows
  - slot Integer the slot in the test
  - sectionID String the section in the test
  - itemID String the item ID
  - stimulusID String the item's stimulus ID
  - itemScore Integer item score. If the item cannot be scored in real time, use **-1**, and the item will be ignored in theta calculation. If the item is a FT or a non-scorable item, itemScore is ignored by the engine.

- **sessionData**

The is the sesstionData returned by /cat call. The CAT engine requires the client to post back sessinData. See below Returns for details of the sessionData.

- **testStartTheta (optional)** for overriding the default test start theta
  - distribution: String NORMAL, UNIFORM, FIXED
  - param1 Double
  - param2 Double

for NORMAL distribution, param1 is mean and param2 is standard deviation  
 for UNIFORM distribution, param1 is min value and param2 is max value  
 for FIXED distribution, param1 is the fixed value, param2 is ignored

- **avoidItems (optional)** list of String avoid item IDs. If the item ID is not in the item pool, it will be ignored.

Returns: JSON format of the following fields:

- **testID** String testID
- **status** String the status of the request. “OK” if successful, otherwise see the detailed error message
- **testCompleted** Boolean whether the test has completed
- **nextItemsToAdminster** a list of the next items to administer
  - slot Integer the slot in the test
  - sectionID String the section in the test
  - itemID String the item ID
  - stimulusID String the item’s stimulus ID
- **score** same structure as “score” in /RESCORE (see below)  
score is only filled when testCompleted = True
- **sessionData** the session data that the CAT engine requires the client to always post back
  - shadowTest the current shadow test. A list of the following fields
    - slot Integer the slot in the test
    - sectionID String the section in the test
    - itemID String the item ID
    - stimulusID String the item’s stimulus ID
  - lastSlotSeen int the last slot seen
  - test\_theta Double interim test theta
  - test\_se Double interim test theta S.E.
  - session\_theta Double interim section theta

- session\_se Double interim session theta S.E.
- bayesian\_theta
  - mean Double mean of theta draws
  - sd Double S.D. of theta draws
  - vector Array of Double theta draws

## POST/RESCORE - for test rescoring

Header: Content-Type = application/json

Parameters: (none)

Body: JSON format

the same structure as in /CAT request body (see above), but sessionData could/should be set to **null**, because this endpoint is for rescoring.

Returns: JSON format of the following fields:

- **testID** String testID
- **status** String the status of the request. “OK” if successful, otherwise see the detailed error message
- **score**
  - overall overall score details with the following fields
    - domainValue String “Overall”
    - itemCount Integer number of items
    - rawScore Integer raw score
    - maxPoints Integer max possible points
    - theta Double overall theta
    - se Double overall theta S.E.
    - scaleScore Integer overall scale score
    - scaleScore\_se Integer overall scale score S.E.
    - performanceLevel String performance level
  - subscore list of subcores with the following fields
    - domainName String the subscore domain name
    - domainScore same structure as overall score
  - bayesian\_theta
    - mean Double mean of theta draws
    - sd Double S.D. of theta draws
    - vector Array of Double theta draws
  - scoredItems list of String the list of item IDs used in scoring
  - skippedItems list of String the list of item IDs skipped in scoring

## Appendix II. Optimal CAT engine IRT models and parameterization

The Optimal CAT engine supports the following IRT models: 1PL, 2PL, 3PL, GPCM and GRM.

The IRT model parameters are defined in the “Item bank” tab sheet in test configuration Excel file.

ITM_ID	STM_ID	ENEMIES	STATUS	MAX POINTS	MODEL	SCALING CONSTANT	A	B	C	D1	D2	D3	D4
DIR_A			DIR										
A1			OP	1	1PL	1		-1.8342					
A2			OP	1	2PL	1	0.4394	-1.4076					
A3			OP	1	3PL	1	0.9085	-1.4063	0.2452				
A4			OP	4	GPCM	1	1.25	-0.8342		0.6032	0.239	-0.2841	-0.5581
A5			OP	4	GRM	1	0.4394	-0.5750		0.7756	0.3673	-0.464	-0.6789

The following notations are used in the formulas below:

- m – MAX POINTS
- sc – SCALING CONSTANT
- a – A parameter (discrimination parameter)
- b – B parameter (difficulty parameter)
- c – C parameter (guessing parameter)
- d [ $d_1, d_2, \dots$ ] – D parameter vector (threshold parameters for GPCM and GRM)
- $\theta$  – student ability
- x – student score

### 1PL model

The item response function (the probability of a correct response to a dichotomous item) is

$$p(\theta, x = 1) = \frac{1}{1 + e^{-sc*(\theta - b)}}$$

### 2PL model

The item response function (the probability of a correct response to a dichotomous item) is

$$p(\theta, x = 1) = \frac{1}{1 + e^{-sc*a*(\theta - b)}}$$

### 3PL model

The item response function (the probability of a correct response to a dichotomous item) is

$$p(\theta, x = 1) = c + \frac{1 - c}{1 + e^{-sc*a*(\theta - b)}}$$

### GPCM model

For the Generalized Partial Credit Model (GPCM, Muraki 1992), there are  $m+1$  possible score points (0, 1, 2, ...,  $m$ ). The probability of getting a score point of  $x$  is

$$p(\theta, x = 0) = \frac{1}{1 + \sum_{r=1}^m \exp [\sum_{j=1}^r sc * a * (\theta - b + d_j)]}$$
$$p(\theta, x = 1, 2, \dots, m) = \frac{\exp [\sum_{j=1}^x sc * a * (\theta - b + d_j)]}{1 + \sum_{r=1}^m \exp [\sum_{j=1}^r sc * a * (\theta - b + d_j)]}$$

### GRM model

For the Graded Response Model (GRM, Samejima's 1969), there are  $m+1$  possible score points (0, 1, 2, ...,  $m$ ). The cumulative probability of getting a score of  $x$  or above is

$$p\_cum(\theta, x = 0) = 1$$
$$p\_cum(\theta, x = 1, 2, \dots, m) = \frac{1}{1 + e^{-sc*a*(\theta-b+d_x)}}$$

The probability of getting a score point of  $x$  is then

$$p(\theta, x = 0, 1, \dots, m - 1) = p\_cum(\theta, x) - p\_cum(\theta, x + 1)$$
$$p(\theta, x = m) = p\_cum(\theta, m)$$



### Appendix III. Optimal CAT software libraries and licenses

Optimal-CAT software includes the following software/libraries:

- FICO Xpress, which is proprietary software, needs a **valid license for commercial use**
- Gurobi, which is proprietary software, needs a **valid license for commercial use**
- Springframework library, which is licensed under Apache License 2.0.
- Fasterxml jackson library, which is licensed under Apache License 2.0.
- Apache Commons Math library, which is licensed under Apache License 2.0.
- License3j library, which is licensed under Apache License 2.0.
- jfreechart library, which is licensed under GNU Lesser General Public License.
- Pandas library, which is licensed under BSD 3-Clause License.
- Pandasql library, which is licensed under MIT License.
- Openpyxl library, which is licensed under MIT License.