



# JSON Validator

Submitted by **TEAM 23**

<b>Abhay Kaushik</b>	2022201054
<b>Ayush Lakshkar</b>	2022201051
<b>Advait Shrivastava</b>	2022201069
<b>Prem Sai Kumar</b>	2022201036

Under the Supervision of  
**Mr. Sai Anirudh Karre**  
Research Scholar  
SERC Lab

## Description

- Software practitioners use a variety of Requirement engineering approaches to produce a well-defined product. These methods impact the software product's ultimate traits. VR has become an essential technology for the future. Nevertheless, very few tools and methods practiced for requirement engineering are not capable enough of addressing all aspects of VR product development. Here we attempt to develop a requirement specification tool for VR system development.

## Requirements

- The VReqst tool is aimed to reduce the developer creation time by taking input from the customer and validating it against the constraints mentioned by the admin of the organization, to ensure the the requirements of the customer are valid. There is an integrated rules editor framework that allows the user to write code faster by giving code snippets.

- To cater to the above requirements, the automated validation method are developed that validates the requirements at five stages along with the rule editor framework to code logic that stores the final results in the database for later viewing and downloads.

# Technology Stack

## Storage:

- MongoDB  
Atlas



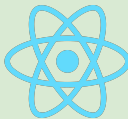
## Back-end:

- Node
- Express



## Front-end:

- React
- JS



## Login Management

- Created interface for login, registration and user authentication.
- Created frontend validation to validate field entries.
- Created all required endpoints in React.
- An admin is predefined for an account for the organization.
- User is created by entering the details.
- After login, the list of projects associated with the user will be displayed. User can also create new project.
- Dashboard at login shows the list of projects

## Workflow

- Each project will have an edit or view button. The edit button is available as long we have JSONS to validate.
- View enables user to view the edited and saved json till that point.
- After uploading the 3 JSONS: Scene, Asset and Action, the tool will be redirected to the portal for custom rule editor framework. Here the user can use the code snippets to write the code and check if the the syntax is valid or not.
- Rule editor generates the base 64 encoded strings and will be uplaoded to mongo db collection.
- After generating base64 encoded strings, tool redirects the user to validate other JSONs i.e Custom JSON and Scene Timeline. Base64 encoded strings will be used to fill Custom JSON.

## List of MongoDB Collections

**Step:** keep track of the current step upto which the project is validated and the last step which was validated.

**Session:** Store the session of the logged in user at what time he/she logged in and out.

**Project Model:** Stores projects of all users with creation time, status, last updated time.

**User JSON Model:** Stores the valid user JSONs that has been uploaded by the user. It also stores the project ID and step number of the step of the project number.

**Validator Model:** Stores all the valid JSON uploaded by the admin.

**Integrate Model:** Stores the list of BASE64 encoded strings sent from the separate Rule Editor Framework project used further in validating user input custom JSON in our project.

**PersonModel:** Stores the details of the person. It also stores the role of the person.

**LoginModel:** Stores the login details of the user. It also stores the role of the user.

**ProjectModel:** Stores details like owner, project id, creation time, status of the project.

## **Grammar Parsing and Scene Validation**

- Validated scene definition script for given input JSON.
- Checked for optional fields and typeof input.
- The validation code takes the input JSON and the validator JSON as input and then checks if the input JSON has a JSON that is valid according to the validator JSON.
- In case the input JSON has nested JSON objects or arrays, we send a recursive function to check if they values have the same datatype as the one mentioned inside validator JSON.

# MongoDB Collections Screenshots

```
_id: ObjectId('6385ea6a38f21f4800fe2cfe')
input_json_content: "{
  \"_scenename\": \"Bowling Alley\",
  \"_sid\": \"s45189\",
  \"_slabel\": \"\"\"

pid: \"2\"
step_no: \"0\"
proj_id: \"TEST-PROJECT\"
__v: 0
```

```
_id: ObjectId('6385eff87e366787574bdb75')
input_json_content: "{
  \"_scenename\": \"Bowling Alley\",
  \"_sid\": \"s45189\",
  \"_slabel\": \"\"\"

pid: \"2\"
step_no: \"0\"
proj_id: \"TEST-2\"
v: 0
```



```
_id: ObjectId('6369502e6f913e3e6f9bc7f2')
pid: "200"
firstname: "admin"
lastname: "admin"
email: "admin@admin.com"
organization: "iiith"
role: "1"
__v: 0
```

```
_id: ObjectId('6385e7f038f21f4800fe2cd1')
pid: "2"
firstname: "Advait"
lastname: "Shrivastava"
email: "advait_shrivastava@students.iiit.ac.in"
organization: "IIITH"
role: "0"
__v: 0
```



# MongoDB Collections Screenshots

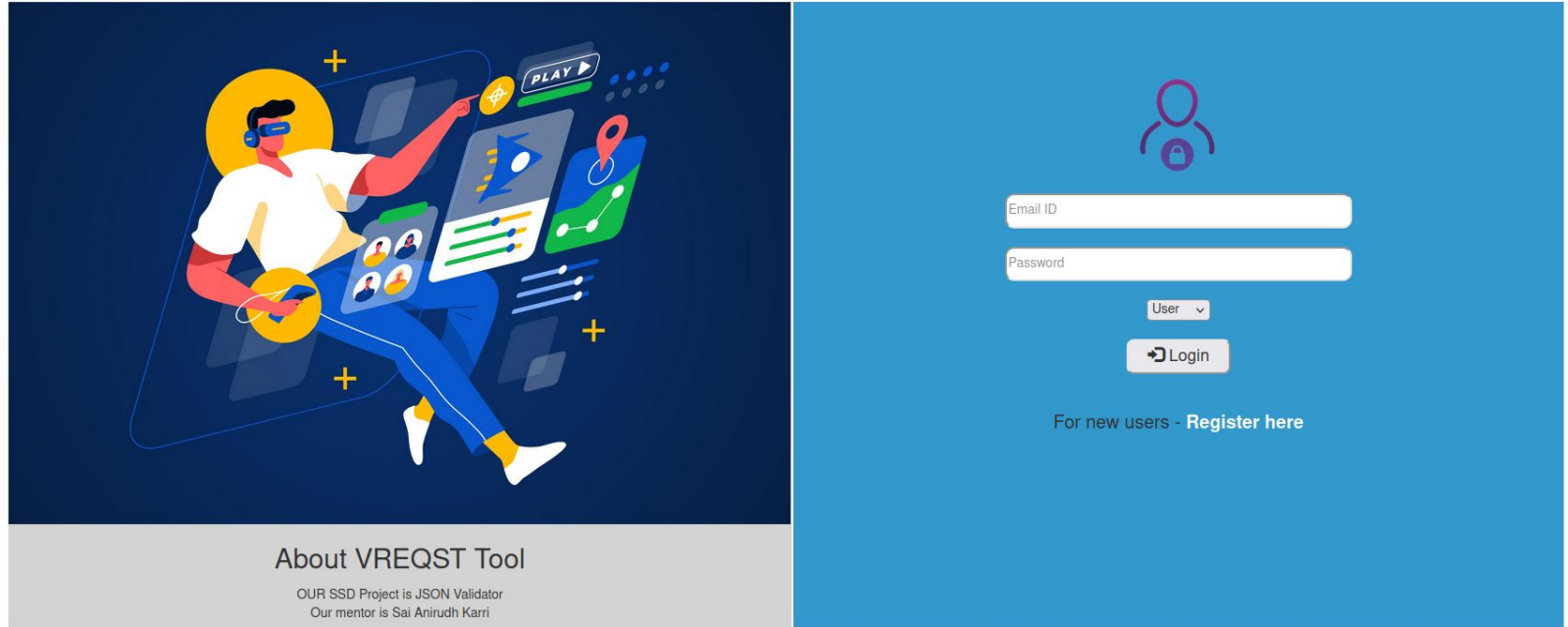
```
▶ _id: ObjectId('6385e85e38f21f4800fe2cda')
Projid: "TEST-PROJECT"
Ownerid: "2"
CreatedTime: "Tue Nov 29 2022 16:39:18 GMT+0530 (India Standard Time)"
Status: "Intermediate"
Step: "2"
IsFinished: "No"
LstUpd: "Tue Nov 29 2022 17:58:18 GMT+0530 (India Standard Time)"
__v: 0
```

```
_id: ObjectId('6385efb77e366787574bdb6e')
Projid: "TEST-2"
Ownerid: "2"
CreatedTime: "Tue Nov 29 2022 17:10:39 GMT+0530 (India Standard Time)"
Status: "Completed"
Step: "4"
IsFinished: "Yes"
LstUpd: "Tue Nov 29 2022 17:38:04 GMT+0530 (India Standard Time)"
__v: 0
```

```
▶ _id: ObjectId('6385e85e38f21f4800fe2cda')
Projid: "TEST-PROJECT"
Ownerid: "2"
CreatedTime: "Tue Nov 29 2022 16:39:18 GMT+0530 (India Standard Time)"
Status: "Intermediate"
Step: "2"
IsFinished: "No"
LstUpd: "Tue Nov 29 2022 17:58:18 GMT+0530 (India Standard Time)"
__v: 0
```

```
_id: ObjectId('6385efb77e366787574bdb6e')
Projid: "TEST-2"
Ownerid: "2"
CreatedTime: "Tue Nov 29 2022 17:10:39 GMT+0530 (India Standard Time)"
Status: "Completed"
Step: "4"
IsFinished: "Yes"
LstUpd: "Tue Nov 29 2022 17:38:04 GMT+0530 (India Standard Time)"
__v: 0
```

# WebApp Screenshots



Login Page

# WebApp Screenshots



Working with VReqSL

About Us

Welcome advait\_shrivastava@students.iiit.ac.in

Logout

Create Project

Stage - 0: Scene JSON

Stage - 1: Action Response

Stage - 2: Asset

Stage - 3: Custom JSON

Stage - 4: Scene Timeline

Sr.No.	Project Name	Owner ID	Stage	Action
1	TEST-PROJECT	2	3	<a href="#">Edit</a> <a href="#">View</a>
2	TEST-2	2	<input checked="" type="checkbox"/> Completed	<a href="#">View</a>
3	MOHIT	2	<input checked="" type="checkbox"/> Completed	<a href="#">View</a>
4	ABHAY	2	3	<a href="#">Edit</a> <a href="#">View</a>

Project Dashboard

# WebApp Screenshots



[Working with VReqSL](#)

[About Us](#)

Welcome advait\_shrivastava@students.iiit.ac.in

[Logout](#)

[+ Create Project](#)

[⌘ Manage Project](#)

▼ Name of Project :

[↗ Create](#)

Create Project

# WebApp Screenshots

[Home](#)

~ Project Name ~  
QWERTY

Scene JSON Properties

▶ \_scenename

▶ \_sid

▶ \_slabel

▶ \_scenearea

▶ \_playarea

▶ \_camera

▶ \_initialpos

▶ \_horizon

▶ \_dof

▶ \_skybox

▶ \_control

▶ \_gravity

▶ \_interaction

▶ \_nestedscene

▶ \_audio

▶ \_timeline

▶ \_Optxt1

▶ @context\_mock

(MANDATORY, NUMBER) - accepts two values 3 and 6. Throw errors for the rest of the values

>>> Scene JSON

Action Response

Asset

Custom JSON

Scene Timeline

Copy and Paste JSON here

~ No Error ~

Validate

Upload

Next

JSON Validation Page

# WebApp Screenshots



Working with VReqSL

About Us

Welcome advait\_shrivastava@students.iiit.ac.in

Logout

Home

~ Project Name ~  
QWERTY

Scene JSON Properties

\_scenename

\_sid

\_slabel

\_scenearea

\_playarea

\_camera

\_initialpos

\_horizon

\_dof

\_skybox

\_controllers

\_gravity

\_interaction

\_nestedscene

\_audio

\_timeline

\_Opttxt1

@context\_mock

Scene JSON

Action Response

Asset

Custom JSON

Scene Timeline

```
{
  "_scenename": "Bowling Alley",
  "_sid": "45189",
  "_slabel": "This is a interactive game",
  "_playarea": {
    "_pid": "001",
    "#length": "20",
    "#breadth": "10",
    "#height": "abcd"
  },
  "_scenearea": {
    "#length": "20",
    "#breadth": "10",
    "#height": "15"
  },
  "_camera": {
    "IsSceneObject": "true",
    "trackingorigin": "true"
  },
  "_initialpos": {
    "#x": "0",
    "#y": "5",
    "#z": "0"
  },
  "_horizon": "true",
  "_dof": "6",
}
```

[OK] \_scenename

[Error]: \_sid input is of string type

[OK] \_slabel

[OK] \_scenearea

[Error]: #height input is of string type

[OK] \_camera

[OK] \_initialpos

[OK] \_horizon

[OK] \_dof

[OK] \_skybox

[OK] \_controllers

[OK] \_gravity

[OK] \_interaction

[OK] \_nestedscene

[OK] \_audio

[OK] \_timeline

[OK] \_Opttxt1

[OK] @context\_mock

Validate

Upload

Next

JSON Validation Page

# WebApp Screenshots



Working with VReqSL

About Us

Welcome advait\_shrivastava@students.iiit.ac.in

Logout

Home

~ Project Name ~  
ABHAY

Custom JSON

- setvalues
- rules
- ruleorder

Scene JSON

Action Response

Asset

Custom JSON

Scene Timeline

```
{
  "setvalues":
  {
    "custom_variables": "$fall,$hits,$roll",
    "custom_functions": "#count",

    "_anypins": "_pinsetter1,_pinsetter2,_pinsetter3,_pinsetter4,_pinsetter5,_pinsetter6,_pinsetter7,_pinsetter8,_pinsetter9,_pinsetter10",
    "point": "null",
    "strike_point": "10 units",
    "ind_point": "1 units",
    "min_score": "0 units",
    "max_score": "null",
    "robotlist": [
      "_camera",
      "_gameball",

      "_pinsetter1", "_pinsetter2", "_pinsetter3",
      "_pinsetter4", "_pinsetter5", "_pinsetter6",
      "_pinsetter7", "_pinsetter8", "_pinsetter9"
    ]
  }
}
```

Base Encoded Strings

SWYjYT5iIzoKICAgIGh1bGxvCiEKRUlmI2E8YiM6C1AgICBxd2VydHkKIQ==

SWYjYT5iIzoKICAgIGh1bGxvCiEKRUlmI2E8YiM6C1AgICBxd2VydHkKIU6PGNvZGUgaGVyZT4h

SWYjYT5iIzoKICAgIGh1bGxvCiEKRUlmI2E8YiM6C1AgICBxd2VydHkKIU6PGNvZGUgaGVyZT4hCkZvc1M8Y29uZD4vPGNvbmQ+Lzxb25kP1M6PGNvZGUgaGVyZT4h

[OK] setvalues  
[OK] rules  
[OK] ruleorder

Validate

Upload

Next

Base64 Encoded Strings Page

# WebApp Screenshots



Working with VReqSL

About Us

Logout

[Home](#)  
  
~ Project Name ~  
TEST-2  
  
~ Current Stage ~  
☒ Completed  
  
[Scene JSON](#)  
  
[Action Response JSON](#)  
  
[Asset JSON](#)  
  
[Custom JSON](#)  
  
[Scene Timeline JSON](#)

Stage - 0: Scene JSON   Stage - 1: Action Response   Stage - 2: Asset   Stage - 3: Custom JSON   Stage - 4: Scene Timeline

### Scene JSON

```
{  "name": "Scene",  "length": "20",  "breadth": "10",  "height": "15",  "_scenearea": {    "length": "20",    "breadth": "10",    "height": "15"  },  "_camera": {    "IsSceneObject": "true",    "trackingorigin": "true"  },  "_initialpos": {    "x": "0",    "y": "5",    "z": "0"  },  "_horizon": "true",  "_dof": "6",  "_skybox": "1",  "_controllers": {    "type": "hand",    "raycast": "ok",    "raydistance": "five",    "raythickness": "bold",    "raycolor": "red",    "raytype": "curve"  },  "_gravity": {    "value": "10"  },  "_interaction": "true",  "_nestedscene": {    "value": "0",    "scenecount": "0",    "sid_order": "0"  }}
```

[Download](#)

View Project Page



# JS Code Snippet

```
const validate = (jsonObject, validator) => {  
  // get the list of keys of the sceneValidator object  
  let validator_keys = Object.keys(validator);  
  let flag = true; // assume valid entries in jsonObject  
  for (let k of validator_keys) {  
    if(jsonObject[k] = undefined) {  
      // the field is not there, so we check if it is mandatory  
      if(validator[k].req = "mandatory") {  
        // the field is mandatory, so we throw error  
        console.log("Error: Input JSON doesn't have field " + k);  
      }  
      // else the field is optional, so we continue  
    }  
    else {  
      // we check if the properties are valid  
      if(typeof jsonObject[k] = "object") {  
        // console.log("Object : " + jsonObject[k])  
        if(Array.isArray(jsonObject[k] = true)) {  
          // it is an Array Object  
          flag = check_type_array(jsonObject[k], validator[k].typeof)  
        }  
        else {  
          // it is a JSON Object  
          flag = check_type(jsonObject[k], validator[k].typeof)  
        }  
      }  
      if(flag = true) {  
        console.log("[OK] " + k)  
      }  
    }  
    else {  
      if(extract(jsonObject[k]) != validator[k].typeof) {  
        // property type is invalid  
        console.log("Error: " + k + " input is of " + typeof jsonObject[k])  
      }  
      else {  
        // property type is valid  
        console.log("[OK] " + k)  
      }  
    }  
  }  
}
```

```
[OK] _scenename  
Error: _sid input is of string type  
[OK] _slabel  
Error: Input JSON doesn't have field _scenearea  
Error: #pid input is of string type  
Error: trackingorigin input is of string type  
[OK] _initialpos  
[OK] _horizon  
[OK] _dof  
Error: _skybox input is of string type  
Error: raycast input is of string type  
Error: raydistance input is of string type  
Error: raythickness input is of string type  
[OK] _gravity  
[OK] _interaction  
Error: #value input is of string type  
Error: #scenecount input is of string type  
Error: #sid_order input is of string type  
[OK] _audio  
[OK] _timeline  
[OK] _Opttxt1  
[OK] @context_mock  
Error: _viewport is not part of validator JSON  
Error: _clippingplane is not part of validator JSON  
object
```

Thank You