SOFTWARE SYSTEMS DEVELOPMENT

Project Title - JSON Validator

TEAM - 23

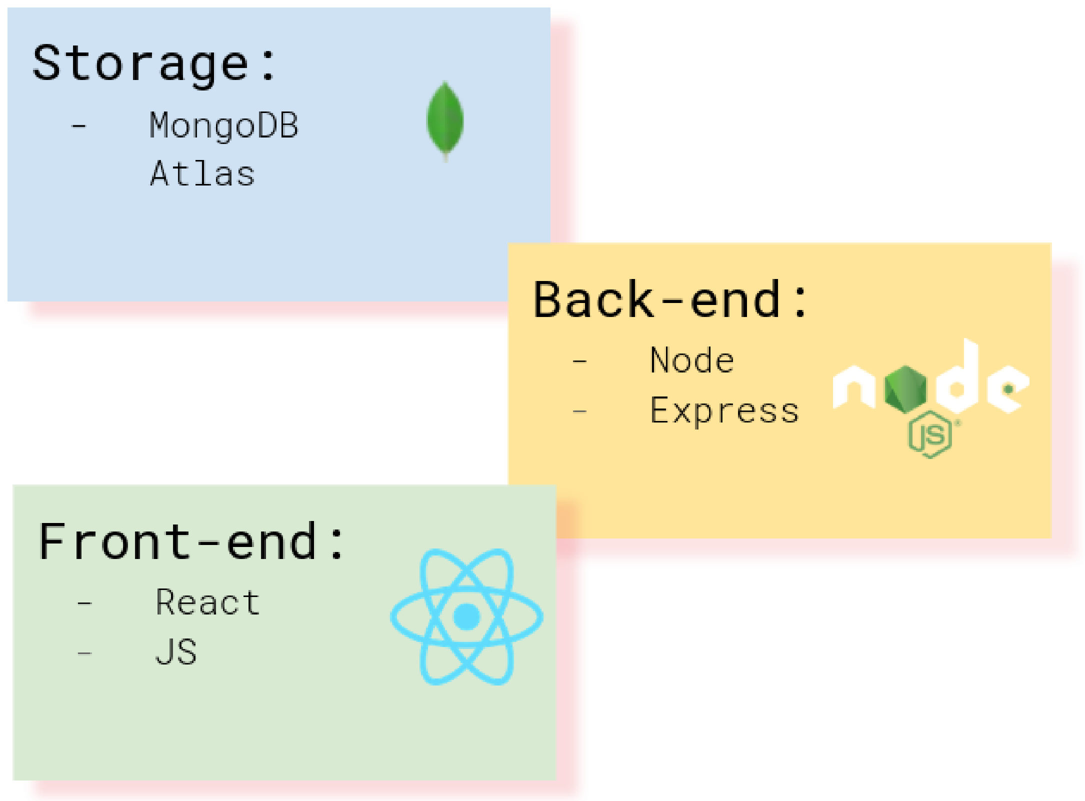
Description

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 VRSeql 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 is 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



Login Management

- An admin is predefined for an account for the organization.
We create a user by first entering the email ID along with the password. The JS code at the front end ensures that the email ID is of a valid format, then after pressing the submit button, it is sent to the MongoDB user collection and we store the new user.

- At the login page, we have the option of selecting if we are logging into the site as a user or as an admin and then enter the requisite credentials.

- At the login screen, we can see the dashboard which shows us the list of projects. The user has the option to create a new project. Upon clicking the new project button, we are taken to a new screen where we can enter the name of the new project, click create and then are redirected back to the homepage.

- The user can see the list of projects, with each having an edit or view button. The edit button is available as long as we have JSONS to validate. Click on the edit button takes us to the step in which we have to input the JSON in the text field.

- The panel on the left side contains the list of fields in the validator JSON. Hovering over them gives extra information. This is to act as a prompt for the user that is inputting the JSON data.

- After inputting JSON, we can then press the validate button to check if the input JSON is valid or not based on the validator JSONs that the admin has entered. If the JSON is valid, the upload button is enabled and we can then press the button to upload the JSON to MongoDB collection. We then press the next button to navigate to the next page.

After uploading the 3 JSONS: Scene, Asset and Action, we 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 syntax is valid or not. Then, we can save the base64 encoded code snippets into the mongoDB collection when the save button is pressed.
After validating and saving all the code snippets, we can then click on the next button to go to the next page and then validate the custom JSON.

MongoDB Collections

StepModel:
Its purpose is to keep track of the current step upto which the project is validated and the last step which aws

validated.

SessionModel:
Used to store the session of the logged in user at what time he/she logged in and logged out.

ProjectModel:
stores the projects of all the users, its creation time, its status and last updated time.

UserJSONModel:
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.

ValidatorModel:
This stores all the valid JSON uploaded by the admin.

IntegrateModel:
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:
Collection project stores details of our project.It contains details like owner of the project,project id,creation time,status of the project,the line no until which our json is validated and is_finished which tells us whether validation is finished or not.

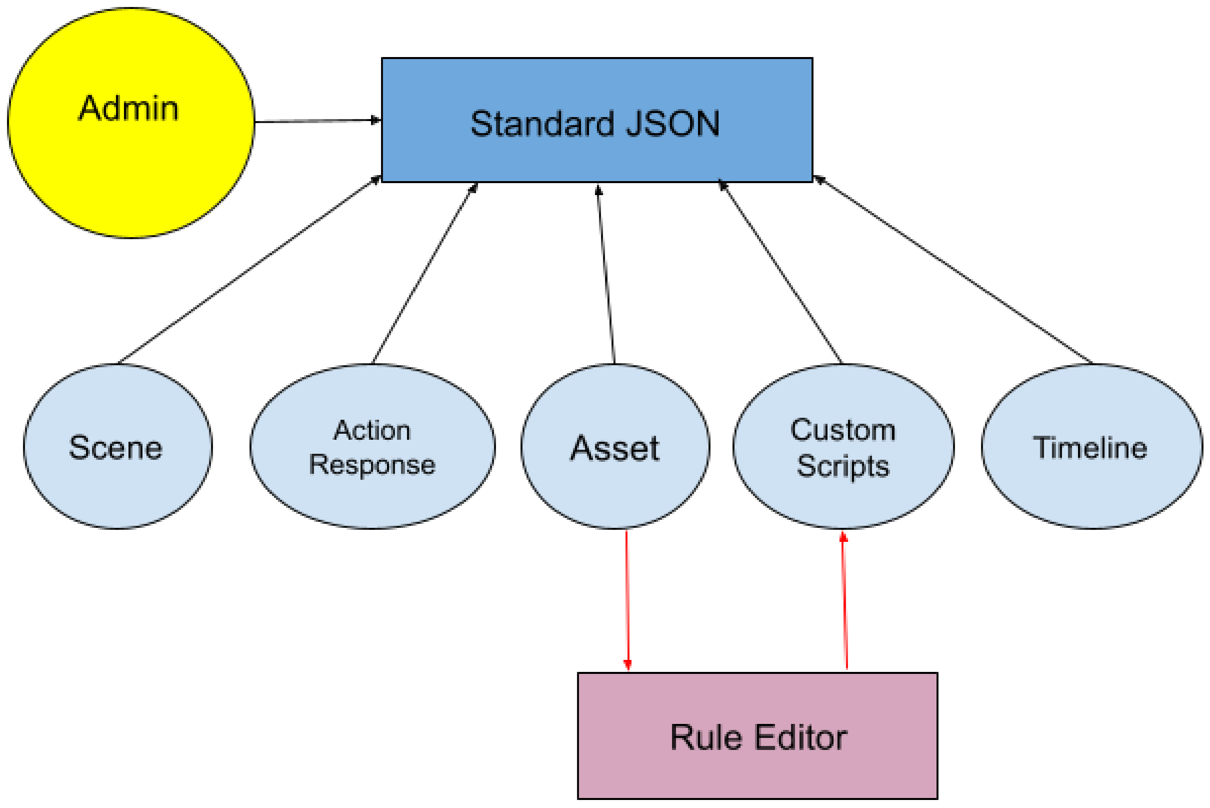
Grammar Parsing and Scene Validation

- 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 the values have the same datatype as the one mentioned inside validator JSON.

- In case a field is optional and the field is missing, no error is thrown. In case there is an excess field not mentioned in the validator JSON then an error is thrown.
If the data typ doesn't match in either the value or in the values of the nested JSON, then an error is thrown.
If the above scenarios do not occur, then an [OK] status is given against the field.

Block Diagram



Integration With Rule Editor Framework

- After uploading the 3 JSONS we 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 syntax is valid or not. Then, we can save the base64 encoded code snippets into the mongoDB collection when the save button is pressed.
- After validating and saving all the code snippets, we can then click on the next button to go to the next page and then validate the custom JSON.
- In our scenario,we are running the rule editor framework and our jSON validator on the same machine but on two different ports so the rule editor framework will work as a service provider for the JSON Validator Project.

Assumptions

- Admin will remain the same for complete organization. So initially admin is created in MongoDB. All further registrations will be allowed for users only.
- Admin will initially upload all the standard JSON files before any user starts a new project.
- Rule Editor Module will generate the base64 encoded strings and they get stored in MongoDB. In stage 4, those encoded strings will be fetched from MongoDB at stage 4 by JSON Validation Module.
- JSON Validation code is made such that it compares the datatype of the values of all the keys mentioned in the validator.

Links

- Link to Github Repository for JSON Validator: https://github.com/Advait-Shrivastava/23_JSON-Validator
- Link to Screen Recording for Project Demo: https://drive.google.com/file/d/1Dhd_Fwd5n-HrGd_Hd6V-LxYkgO520907/view
- Link to Github Repository for Rule Editor Framework: https://github.com/SrikantKonduri/20_CustomizedRuleEditorFramework