**Requirement Specification Tool**

**Team: 12**

**Vaibhav Saxena - 2022202026**

**Piyush Rana - 2022202012**

**Ardhendu Banerjee - 2022201005**

**Priyank Mahour - 2022201047**

**Description:**

As a software developer, while developing applications a critical part is to develop proper requirement specifications. Taking the example of VR systems, it is very important to have proper specification definition checking tools to aid the development. This is a segment where we lack well defined tools to do the job. Here we attempt to develop a tool to help do the following few tasks:

* Specifying VR Scene layout Properties
* Specifying Object Properties that are used in VR scene and validate them against predefined rules. This makes the validated properties specification a well defined data to be used further in the development process.
* As an output a JSON file which captures the validated requirements from multiple stages of a requirement validation.

**Requirement:**

To fulfill the required expectations at the very basic we need a web app to facilitate the processing of the given requirement specifications. Accompanied with a proper database setup to store the different user base and the save the progress of their respective progress over the projects for future references. And over this web app, we would need a validator system working in the background to provide the validation services to the users. The validator system should be able to identify any validator grammar provided and it should be able to validate any given data.

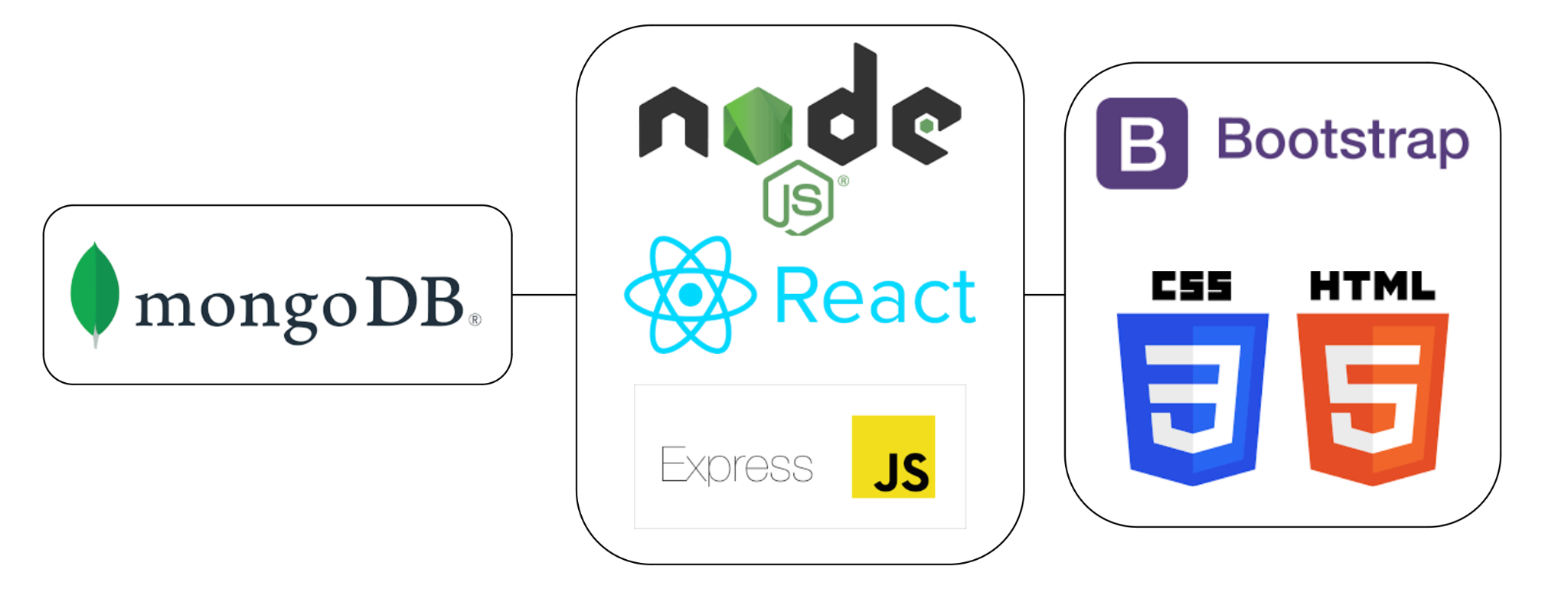
**System Design:**

* **Technology Stack:**

Frontend: To develop the web app we have made use of NodeJs runtime environment which provides us the development space to build our application. Over that we have made use of ReactJS to develop components which has been used over the application, and the React components consist of JSX which includes HTML/CSS and JavaScript.

WebServer: Express Router has been used to route from page to page inside the application. Mongoose Library has been used to connect with underlying database.

Database: As per our requirement we need a noSQL database to store the specification rules. And the specification rules can be in any shape, hence as per the requirement MongoDB has been used to fulfill the database module.



* **Login Management:**

To gain access to this tool, one should register by signing up on the registration page using

* + - Email – Requires validation (all email validation rules)
    - Password – Requires validation (String, alpha-numeric, non-nullable)

The Login is role based, namely “admin” and “user”. An “admin” role account gets the privilege of upload specification rules in the format of JSON. A “user” role account gets to use the application to get specifications validated. The password is hashed and stored into the corresponding DB entry.

* **Project Management Screen:**

A project management system which allows user to track and deal with progress on different projects

* + - After Login the user will land at Project Management.
    - It allows the user to see and edit existing projects.
    - Or new projects can be created from the same page.

**Grammar Parsing and Scene Validation Steps:**

* Admin can upload the grammar of his choice , which will be saved in the database as String.
* Only Admin user has Access to upload Validator-Grammar
* Users can upload any specification in the form of JSON and select a Grammar of choice from Drop-Down showing all the Validator-Grammar in the database, uploaded by Admins.

**Grammar Parsing:**

* When a user submits data to be validated against a selected JSON grammar( specification rules ), the grammar is fetched from the database in string form and is parsed to JSON object.
* This JSON object is then scanned to collect all the valid keys in the grammar. These keys refer to the valid specification properties.
* The parsed JSON object is read to get a list of mandatory specification elements based on their requirement tag (“req”).

**Specification Validation:**

* When the user provides the specifications data in the JSON format, initially a parsable check is done on the data. Invalid syntax errors are alerted to the user.
* Moving forward with valid JSON format data, all given specifications should be valid, i.e all specification tags should be defined in the specification rules. Errors are generated for the invalid keys.
* Next steps of the validation include the check to verify all “mandatory” marked specification details are must to be included in the data. Based on the empty or missing “mandatory” marked tags the errors are generated to the user.
* The last stage of the validation includes the data-type validation which checks the “typeof” for every specification described in the rule and validates it against the input data that is taken from the user.
* Any progress on each stage is saved against the current project and can be retrieved later from the database.

**Result Generation:**

* At the final stage we have an option to get specifications data from all the stages displayed to us and we are provided with a download option to get the data from all the stages consolidated in a single file in JSON format.

**Conclusion:**

* While developing the application to provide the functionality of validating a data in JSON format against a JSON grammar or a JSON rule set, we identified that this functionality can be useful in not only VR specifications validation, but in many other systems where JSON format data sets are used to transfer or feed data onto the systems. Many systems which are based on semi-structured and unstructured data, wherein data is feed into the system in JSON format, where the proper specification of the data is very critical to the system. In those system this tool can be integrated to pre-suffice the user with validation errors in the data.