

IIT Hyderabad

INTERNSHIP PROJECT REPORT

PROJECT TITLE

VReqST

**A Requirement Specification Tool for building Virtual
Reality Systems - System Design Document**

Submitted By

Snehashish Rout & Nisha Prakash

Under the guidance of

Dr Raghu Reddy, Saianirudh Karri and Vivek Pareek



Problem Statement-

Software practitioners use a variety of Requirement engineering approaches to produce a well-defined product. These methods impact the software product's ultimate traits and target a particular audience segment. Virtual Reality (VR) products are no different from such an influence. With the notable rise in product offerings across various fields, 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. Thus, we need to develop a tool (particularly a web application) called VReqST - a requirement specification tool for VR system development.

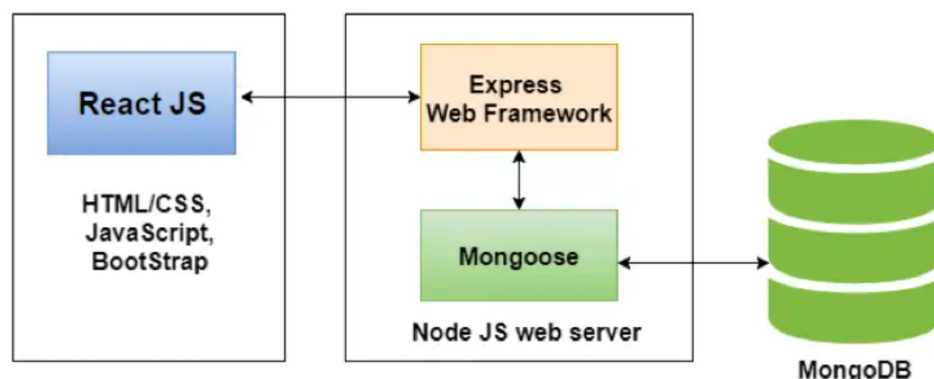
The tool serves to specify VR scene Layout properties and specifying object properties that are used in VR scene as well as Action-Responses allowed between Objects in the given Scene. Also Custom Scripts to document custom tasks-responses, defining algorithms etc would also be included.

Technology Stack Used-

This tool is a Web based tool with a MERN technology stack.

ReactJS is used as a front-end framework for building UI.

Express Web Framework running on NodeJS web server with MongoDB used as database provider. Axios is also used as a side tool for handling API calls.



Installation-

MongoDB Atlas

We used [MongoDB Atlas](#) as the database so that it can be accessible by other devices.

- After registration with Atlas we created a new project so a new Cluster will be created.
- Once the Cluster is created we have to create a Database and collections that are required.
- Now connect with the cluster by setting up your Username and Password and then choose a connection method.

- In Network Access add your current IP Address or allow access from anywhere.
- Now Install MongoDB in your system.
- To check if it's installed properly, run the following command-
mongo --version.
- Install mongoose in the server directory of your project using command-
Npm i mongoose
- Now we have to connect the backend to the database.
- In the environment variable store the database url with username and password and require the dotenv file in your main app.js file.
- Require the mongoose file and use mongoose.connect(Database Name).Also use a promise to log if the connection is successful. In case of error log the error.
- Use nodemon to start the server.

ReactJS

- Create your react app using the following command-
npx create-react-app app-name
- Go into the app directory using cd app-name
- Use npm start to start the react server.

Work Flow-



Registration-

To gain access to this tool, the VR practitioner should register by accessing a registration page with following details:

- FirstName (String, non-numeric, non-nullable)

- LastName (String, non-numeric, non-nullable)
- Email(all email validation rules)
- Organisation(String, non-numeric, non-nullable)
- Password(String, alpha-numeric, non-nullable)
- Cpassword(String, alpha-numeric, non-nullable)
- Token (Will be generated automatically if registration is successful)

Registration data is stored in a Mongo Collection in a collection called **USER**.

The password is encrypted using hashing and salting with bcrypt. We are doing 12 rounds of hashing.

Login and Session Management-

Cookie will be created everytime a user logs in destroy every time the user logs out. It also destroy after a fixed amount of time.

For authentication we are using a middleware in which we are using the username that we get from the token to authenticate the user.

Once the user will be authenticated they will be able to see their projects and information.

User Interface-

- Entry Point:
Sign In & Sign Up

VReqSL

Get Started

Sign In

Email :

Password :

Sign In

[Forgot password?](#)

Don't have an account? [Sign Up](#)

We are more than just a company

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

VReqSL

Create Account

First Name

Last Name

Organisation

Email

Password

Confirm Password

Sign Up

Already have an account?

Sign In

We are more than just a company


Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

- The project table will consist of all the project that the user will insert. Each row will have an ID which we will use to edit and delete. This field will be hidden. Also, it will have the userID of the user for authentication. The collection will have the following details in the DataBase
 - ID
 - Project
 - Owner
 - Stage
 - Username(We will get the user name from cookies using token and will use it for authentication)

VReqSL

- Profile
- Support
- LogOut

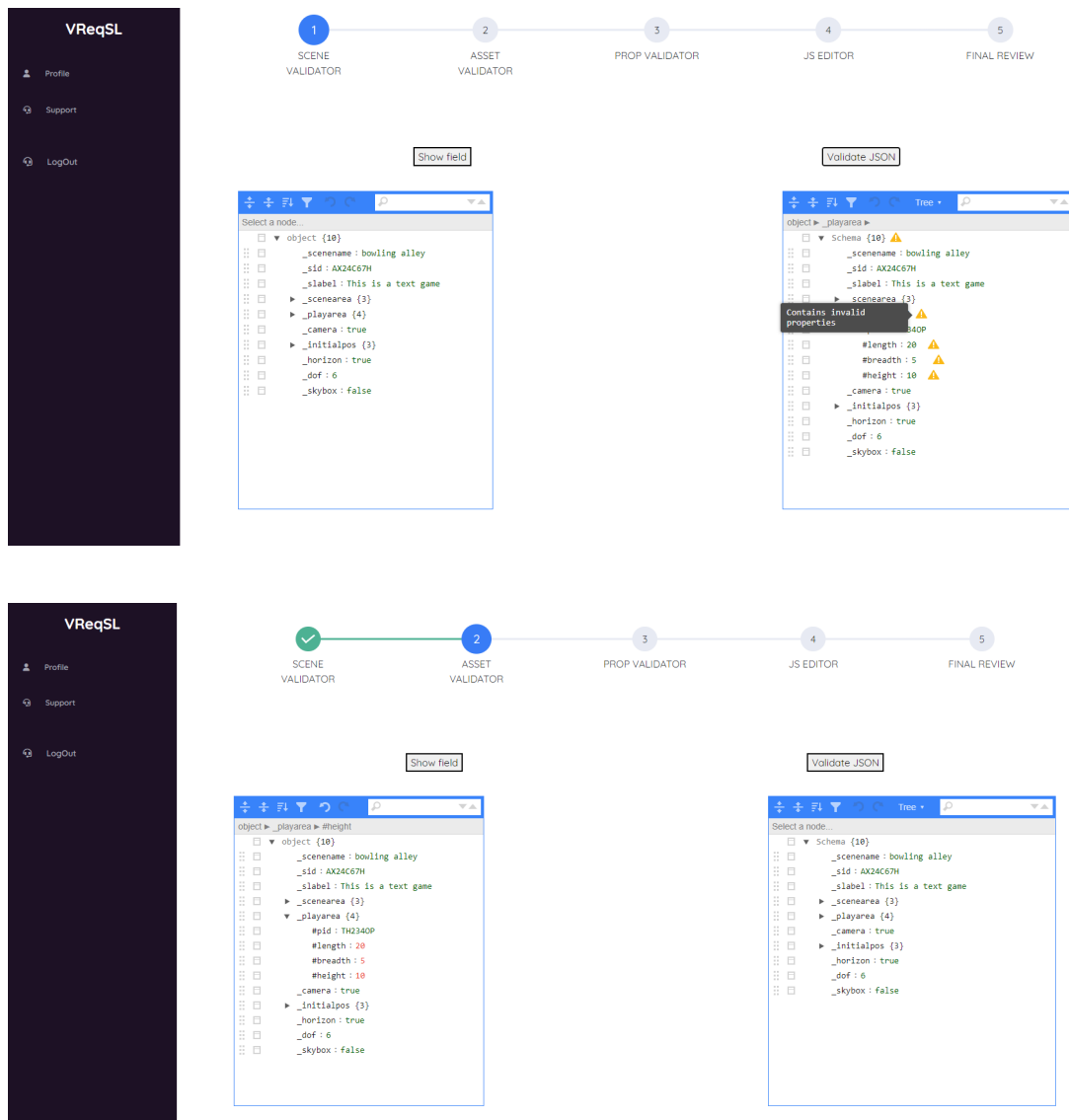
Dashboard

Support 

Home

Project Name	Owner	Stage	Action	
VReqST	Sai Annirudh	Last	View Edit Save	X
VREye	Bhaskar	Completed	View Edit Save	X
Umetrix	Krupal	Started	View Edit Save	X
VR Game	Kandar	Submitted	View Edit Save	X
Surf-Tribe	Nisha	Last	View Edit Save	X

- JSON Editor:**
 This is the Editor Window where the user edits the sample JSON data in the editor field which gets compared and validated by the editor with respect to the standard Schema or grammar file stored in background. All the errors get thrown in the editor. The JSON fields are editable and the user can change the user JSON field accordingly to remove the errors. The user is able to move forward only when all the errors have been dealt with.
- Progress Bar:** It shows the progress or the stage at which the user is currently in. It allows user to move forward only when the previous stage has been completed and is error free.



Specification Management-

Requirement Specification has 5 steps

1. Scene definition – A bare minimum Scene definition and all underlying properties of a Scene Class are made available in a JSON [file](#) format. The user will be able to fill in the values in this JSON file. This JSON file will be validated against a predefined JSON scene validator and allows the user to move to the next step.
2. Asset definition – A bare minimum Asset definition and all underlying properties of an Asset class are made available in a JSON file format. The user will be able to fill in the values in the JSON file. This JSON [file](#) will be validated against a predefined JSON asset validator and allows the user to move to the next step.

The moment a user logs in, we provide him a unique ID that is JSON WebToken(JWT) and then we use the token for session management. Cookie will be created every time a user logs in destroy every time the user logs out. It also destroy after a fixed amount of time.

For authentication we are using a middleware in which we are using the username that we get from the token to authenticate the user.

Once the user will be authenticated they will be able to see their projects and information.

User Interface-

Entry Point:

Sign In & Sign Up

- The project table will consist of all the project that the user will insert. Each row will have an ID which we will use to edit and delete. This field will be hidden. Also, it will have the userID of the user for authentication.

The collection will have the following details in the DataBase

- ID
- Project
- Owner
- Stage
- Username(We will get the user name from cookies using token and will use it for authentication)

·

JSON Editor:

This is the Editor Window where the user edits the sample JSON data in the editor field which gets compared and validated by the editor with respect to the standard Schema or grammar file stored in background. All the errors get thrown in the editor. The JSON fields are editable and the user can change the user JSON field accordingly to remove the errors. The user is able to move forward only when all the errors have been dealt with.

Progress Bar: It shows the progress or the stage at which the user is currently in. It allows user to move forward only when the previous stage has been completed and is error free.

Specification Management-

Requirement Specification has 5 steps :

1. Scene definition – A bare minimum Scene definition and all underlying properties of a Scene Class are made available in a JSON file format. The user will be able to fill in the values in this JSON file. This JSON file will be validated against a predefined JSON scene validator and allows the user to move to the next step.
2. Asset definition – A bare minimum Asset definition and all underlying properties of an Asset class are made available in a JSON file format. The user will be able to fill in the values in the JSON file. This JSON file will be validated against a predefined JSON asset validator and allows the user to move to the next step.
3. Action-Response Definitions – This step allows users to provide an action–response relationship between the assets described in step 2 in a JSON file format. This section provides clarity on possible interactions between assets with one-on-one and one-on-many in the scene.
4. Custom Script Logic – This section allows user to write custom actions, define custom logic using conditional statements like IF-THEN, IF-THEN/ELSE-THEN, FOR-EVERY, DO-UNTIL. These conditional statements are dragged onto the editor pane and are evaluated using an underlying conditional syntax grammar.
5. EventTimeline – This section is to define the sequence of events on a single or multiple or parallel timelines as per the need of a user in regard to defined scene. This is a JSON based event-timeline wizard. A bare minimum event timeline definition and all underlying properties of a time class are made available in a JSON file format.

Contributions (SNEHASHISH ROUT) :

I worked on Frontend all the way from designing and Implementing the UI to Integrating it with the Backend. Also worked on the JSON editor validator which takes in JSON schema and an editable data file and shows in errors in the code which are present in the data. The Login and LogOut pages have got fields to take in user data and based on user authentication it does session management. The Sidebar is shown across all pages to help User navigate across the website.

On the dashboard, a Table/List of all the projects of the user gets Displayed. The User can create a new Project and delete a previous one. The two Buttons View and Edit is present for each Project. The Edit button takes the user to a 5 stepped process of Validations. After the user takes note of and deals with each step, the app serves its purpose of helping the user create a format specifying the requirements for VR development.

I had to set Up the JSON editor such that when there is error in the validation, the app prohibits the user to move to the next step, until the whole error is sorted out.

Contributions (NISHA PRAKASH) :

I worked on Backend part of the web application. I created APIs using node, express and used Mongo DB Atlas as the data base. I did session management and authentication using cookies and tokens and also worked on integration of Registration and Login pages with front-end.

The functionality of the website is explained above.

GitHub Link:

https://github.com/VHIL-interns-hub/VReqSL_ReactApp_summer2022.git