

```
import LoadingSpinner from '@shared/components/loading-spinner';

import Modal from '@shared/components/modal';

import { Button } from '@shared/components/ui/Button';

import Input from '@shared/components/ui/Input';

import { useToast } from '@shared/components/ui/Toasts/use-toast';

import { debounce } from '@shared/utils/helpers';

import { trpc } from '@shared/utils/trpc/trpc';

import { ArrowLeft, Plus } from 'lucide-react';

import {

  Select,

  SelectContent,

  SelectItem,

  SelectTrigger,

  SelectValue,

} from '@shared/components/ui/select';

import { useMemo, useState } from 'react';

import { languageOptions } from '@shared/constants/explorer';

import { useAuthContext } from '@shared/components/ui/auth.provider';
```

```
interface Props {

  isOpen: boolean;

  onClose: () => void;

  onAddSuccessfully: () => void;

}
```

```
const AddAgentModal = ({ isOpen, onClose, onAddSuccessfully }: Props) => {
```

```
const { user } = useAuthContext();

const [step, setStep] = useState<'info' | 'requirement'>('info');

const [agentName, setAgentName] = useState("");

const [description, setDescription] = useState("");

const [agent, setAgent] = useState("");

const [tags, setTags] = useState("");

const [language, setLanguage] = useState('python');

const [useCases, setUseCases] = useState<

  {

    title: string;

    description: string;

  }[]

>([

  {

    title: "",

    description: "",

  },

]);

const [requirements, setRequirements] = useState<

  {

    package: string;

    installation: string;

  }[]

>([

  {

    package: "",
```

```
    installation: "",  
  },  
]);
```

```
const validateAgent = trpc.explorer.validateAgent.useMutation();
```

```
const debouncedCheckPrompt = useMemo(() => {  
  const debouncedFn = debounce((value: string) => {  
    validateAgent.mutateAsync(value);  
  }, 400);  
  return debouncedFn;  
}, []);
```

```
const toast = useToast();
```

```
const addAgent = trpc.explorer.addAgent.useMutation();
```

```
const nextStep = () => {  
  // Validate Agent  
  if (validateAgent.isPending) {  
    toast.toast({  
      title: 'Validating Agent',  
    });  
    return;  
  }  
}
```

```
if (agentName.trim().length < 2) {  
  
  toast.toast({  
  
    title: 'Name should be at least 2 characters long',  
  
    variant: 'destructive',  
  
  });  
  
  return;  
  
}
```

```
if (validateAgent.data && !validateAgent.data.valid) {  
  
  toast.toast({  
  
    title: 'Invalid Agent',  
  
    description: validateAgent.data.error,  
  
    variant: 'destructive',  
  
  });  
  
  return;  
  
}
```

```
setStep('requirement');  
  
};
```

```
const submit = () => {  
  
  // Validate use cases  
  
  for (const useCase of useCases) {  
  
    if (  
  
      useCase.title.trim().length === 0 ||  
  
      useCase.description.trim().length === 0
```

```

) {

  toast.toast({

    title: `Use case ${useCase.title.trim().length === 0 ? 'title' : 'description'} is required`,

    variant: 'destructive',

  });

  return;

}

}

// Validate requirements

for (const requirement of requirements) {

  if (

    requirement.package.trim().length === 0 ||

    requirement.installation.trim().length === 0

  ) {

    toast.toast({

      title: `Requirement ${requirement.package.trim().length === 0 ? 'package' : 'installer'} is
required`,

      variant: 'destructive',

    });

    return;

  }

}

const trimTags = tags

  .split(',')

```

```
.map((tag) => tag.trim())
```

```
.filter(Boolean)
```

```
.join(',');
```

```
// Add Agent
```

```
addAgent
```

```
.mutateAsync({
```

```
  name: agentName,
```

```
  agent,
```

```
  description,
```

```
  useCases,
```

```
  language,
```

```
  requirements,
```

```
  tags: trimTags,
```

```
})
```

```
.then(() => {
```

```
  toast.toast({
```

```
    title: 'Agent added successfully ',
```

```
  });
```

```
  onClose();
```

```
  onAddSuccessfully();
```

```
  // Reset form
```

```
  setAgentName("");
```

```
  setAgent("");
```

```
  setDescription("");
```

```
  setTags("");
```

```
    setUseCases([ { title: " ", description: " " }]);

    setRequirements([ { package: " ", installation: " " }]);

  });

};
```

```
if (!user) return null;
```

```
return (
```

```
  <Modal
```

```
    className="max-w-2xl overflow-y-auto"
```

```
    isOpen={isOpen}
```

```
    onClose={onClose}
```

```
    title="Add Agent"
```

```
  >
```

```
    {step === 'info' && (
```

```
      <div className="flex flex-col gap-2 overflow-y-auto h-[60vh] relative px-4">
```

```
        <div className="flex flex-col gap-1">
```

```
          <span>Name</span>
```

```
          <div className="relative">
```

```
            <Input
```

```
              value={agentName}
```

```
              onChange={setAgentName}
```

```
              placeholder="Enter name"
```

```
            />
```

```
          </div>
```

```
        </div>
```

```

<div className="flex flex-col gap-1">

  <span>Description</span>

  <textarea

    value={description}

    onChange={(e) => setDescription(e.target.value)}

    placeholder="Enter description"

    className="w-full h-20 p-2 border rounded-md bg-transparent outline-0 resize-none"

  />

</div>

<div className="flex flex-col gap-1">

  <span>Agent Code in Python</span>

  <div className="relative">

    <textarea

      value={agent}

      onChange={(v) => {

        setAgent(v.target.value);

        debouncedCheckPrompt(v.target.value);

      }}

      required

      placeholder="Enter agent code here..."

      className="w-full h-20 p-2 border rounded-md bg-transparent outline-0 resize-none"

    />

    {validateAgent.isPending ? (

      <div className="absolute right-2 top-2">

        <LoadingSpinner />

      </div>

```


) : (

<div className="absolute right-2.5 top-2.5">

{agent.length > 0 && validateAgent.data && (

<span

className={

validateAgent.data.valid

? 'text-green-500'

: 'text-red-500'

}

>

{validateAgent.data.valid ? " : "}

}}

</div>

}}

</div>

{agent.length > 0 &&

!validateAgent.isPending &&

validateAgent.data &&

!validateAgent.data.valid && (

{validateAgent.data.error}

}}

</div>

<div className="flex flex-col gap-1">

```
<span>Language</span>
```

```
<Select onChange={setLanguage} value={language}>
```

```
  <SelectTrigger className="w-1/2 cursor-pointer, capitalize">
```

```
    <SelectValue placeholder={language} />
```

```
  </SelectTrigger>
```

```
  <SelectContent className="capitalize">
```

```
    {languageOptions?.map((option) => (
```

```
      <SelectItem key={option} value={option}>
```

```
        {option}
```

```
      </SelectItem>
```

```
    )))
```

```
  </SelectContent>
```

```
</Select>
```

```
</div>
```

```
<div className="flex flex-col gap-1">
```

```
  <span>Tags</span>
```

```
  <Input
```

```
    value={tags}
```

```
    onChange={setTags}
```

```
    placeholder="Tools, Search, etc."
```

```
  />
```

```
</div>
```

```
<div className="flex justify-end mt-4">
```

```
  <Button
```

```
    disabled={addAgent.isPending}
```

```
    onClick={nextStep}
```

```

        className="w-32"

        >

        Next

    </Button>

</div>

</div>

)}

```

```

{step === 'requirement' && (
  <div className="flex flex-col gap-1 overflow-y-auto h-[60vh]">
    <div className="mt-2 flex flex-col gap-1">
      <span>Use Cases</span>
      <div className="flex flex-col gap-2">
        {useCases.map((useCase, i) => (
          <div key={i} className="flex gap-4 items-center">
            <span className="w-8">
              <span># {i + 1}</span>
            </span>
            <div className="w-full flex flex-col gap-1 py-2">
              <Input
                value={useCase.title}
                onChange={(v) => {
                  const newUseCases = [...useCases];
                  newUseCases[i].title = v;
                  setUseCases(newUseCases);
                }}
              </Input>
            </div>
          </div>
        ))}
      </div>
    </div>
  )}

```

```

placeholder="Title"

/>

<textarea

value={useCase.description}

onChange={(e) => {

  const newUseCases = [...useCases];

  newUseCases[i].description = e.target.value;

  setUseCases(newUseCases);

}}

placeholder="Description"

className="w-full h-20 p-2 border rounded-md bg-transparent outline-0 resize-none"

/>

</div>

<div className="w-4">

  {i > 0 && (

    <button

      onClick={() => {

        const newUseCases = [...useCases];

        newUseCases.splice(i, 1);

        setUseCases(newUseCases);

      }}

      className="text-red-500"

    >

    </button>

  )}

)}

```

</div>

</div>

)))

<div className="flex justify-center">

<button

onClick={() =>

setUseCases([...useCases, { title: "", description: "" }])

}

className="text-blue-500"

>

<Plus />

</button>

</div>

</div>

</div>

<div className="mt-2 flex flex-col gap-1">

Requirements

<div className="flex flex-col gap-2">

{requirements.map((requirement, i) => (

<div key={i} className="flex gap-4 items-center">

 {i + 1}

<div className="w-full flex flex-col md:flex-row gap-1 py-2">

<Input

```
value={requirement.package}

onChange={(value) => {

  const newRequirements = [...requirements];

  newRequirements[i].package = value;

  setRequirements(newRequirements);

}}

placeholder="Package"

/>
```

```
<Input

value={requirement.installation}

onChange={(value) => {

  const newRequirements = [...requirements];

  newRequirements[i].installation = value;

  setRequirements(newRequirements);

}}

placeholder="pip install package"

/>
```

```
</div>
```

```
<div className="w-4">
```

```
{i > 0 && (

  <button

    onClick={() => {

      const newRequirements = [...requirements];

      newRequirements.splice(i, 1);

      setRequirements(newRequirements);

    }}


```

```

        className="text-red-500 text-sm"

        >

        </button>

    })

</div>

</div>

)}}

<div className="flex justify-center">

    <button

        onClick={() =>

            setRequirements([

                ...requirements,

                { package: " ", installation: " " },

            ])

        }

        className="text-blue-500 text-sm"

        >

        <Plus />

    </button>

</div>

</div>

<div className="flex justify-between mt-4">

    <ArrowLeft

        onClick={() => setStep('info')}

        size={32}

```

```
    aria-label="Previous Button"
```

```
    className="text-primary cursor-pointer"
```

```
  />
```

```
  <Button
```

```
    disabled={addAgent.isPending}
```

```
    onClick={submit}
```

```
    className="w-32"
```

```
  >
```

```
    Submit
```

```
  </Button>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
  )}
```

```
</Modal>
```

```
);
```

```
};
```

```
export default AddAgentModal;
```