

D Glassick Portfolio

All of the projects below were developed in React using TypeScript. All of these projects do use education jargon which consists of a lot of acronyms. Don't worry if you don't know what it is the general concepts of web design should still be there.

For the sake of brevity, and both giving a thorough look at some of my work, I'm going to include my most recent project at work, and the most recent project I made on my own experimenting. in some of the screen shots below you can see a Side Nav Menu and also a the main grid of reports. I developed both of those components as well.

Project # 1 - Calculator Pane

The Calculator pane is a tab in one of our side menus that allows users to fill out forms that will filter data. There are 7 different types of calculators and types were made to ensure the form modified the correct data.

Types file for the Calculator Pane.

```
export type CalculatorType =
  'calculators_summary' |
  'calculators_sgp_mstep' |
  'calculators_sgp_star' |
  'calculators_cgp_map' |
  'calculators_ag_mstep' |
  'calculators_slo_tieredGrowth' |
  'calculators_slo_effectSize';

export type SummaryIds =
  'sgp_mstep' |
  'sgp_star' |
  'cgp_map' |
  'ag_mstep' |
  'slo_tieredGrowth' |
  'slo_effectSize';

export type SummaryInputs = {
  [key in SummaryIds]: {
    active: boolean;
  };
}
```

```

        weight: number;
    };
};

export type SgpMstepInputs = {
    availableYears: string[];
    selectedYears: string[];
    threeYearWeight1: number;
    threeYearWeight2: number;
    threeYearWeight3: number;
    twoYearWeight1: number;
    twoYearWeight2: number;
    cutScore1: number;
    cutScore2: number;
    cutScore3: number;
    minStudents: number;
    includeTests: string[];
    includeSubjects: string[];
    excludeStudents: boolean;
};

export type SgpStarInputs = {
    availableYears: string[];
    selectedYears: string[];
    threeYearWeight1: number;
    threeYearWeight2: number;
    threeYearWeight3: number;
    twoYearWeight1: number;
    twoYearWeight2: number;
    cutScore1: number;
    cutScore2: number;
    cutScore3: number;
    minStudents: number;
    includeSubjects: string[];
    excludeStudents: boolean;
};

export type CgpMapInputs = {
    availableYears: string[];
    selectedYears: string[];
    threeYearWeight1: number;
    threeYearWeight2: number;
    threeYearWeight3: number;
    twoYearWeight1: number;
    twoYearWeight2: number;
    cutScore1: number;
    cutScore2: number;
    cutScore3: number;
    minStudents: number;
    includeSubjects: string[];
    excludeStudents: boolean;
};

export type AgMstepInputs = {

```

```

availableYears: string[];
selectedYears: string[];
model: {
  label: string;
  closeToProficiency: number;
  likelyImproved: number;
  certainlyDeclined: number;
};
threeYearWeight1: number;
threeYearWeight2: number;
threeYearWeight3: number;
twoYearWeight1: number;
twoYearWeight2: number;
cutScore1: number;
cutScore2: number;
cutScore3: number;
minStudents: number;
includeTests: string[];
excludeStudents: boolean;
};

export type SloTieredGrowthInputs = {
  cutScore1: number;
  cutScore2: number;
  cutScore3: number;
  quintile1: {
    active: boolean;
    override: number;
  };
  quintile2: {
    active: boolean;
    override: number;
  };
  quintile3: {
    active: boolean;
    override: number;
  };
  quintile4: {
    active: boolean;
    override: number;
  };
  quintile5: {
    active: boolean;
    override: number;
  };
  minStudents: number;
  excludeStudents: boolean;
};

export type SloEffectSizeInputs = {
  model: {
    label: string;
    cutScore1: number;
    cutScore2: number;

```

```

        cutScore3: number;
    };
    minStudents: number;
    excludeStudents: boolean;
};

export type CalculatorInputs = SummaryInputs | SgpMstepInputs | SgpStarInputs | CgpMapInputs | AgMstepInputs | SloTieredGrowthInputs | SloEffectSizeInputs;

export type Calculator = {
    id: string;
    type: CalculatorType;
    label: string;
    inputs: CalculatorInputs;
    sortOrder: number;
    sharedByUsername?: string;
    sharedUTC?: string;
    lastUpdatedUTC?: string;
};

export type CalculatorState = Calculator[];

export type CalculatorAction =
    | { type: 'ADD_CALCULATOR'; newCalculator: Calculator }
    | { type: 'EDIT_CALCULATOR'; editCalculator: Calculator }
    | { type: 'REMOVE_CALCULATOR'; currentCalculatorId: string };

export type CalculatorDispatch = (action: CalculatorAction) => void;

export type CalculatorContextProps = { state: CalculatorState; dispatch: CalculatorDispatch };
export type CalculatorProviderProps = { children: React.ReactNode };

export type CalculatorFormType = 'add' | 'edit';

export type CalculatorModalType = CalculatorFormType | 'share' | 'delete';

export type ModalState = {
    [key in CalculatorModalType]: boolean;
};

```

eidex Current Account Eidex Account Contact Profile Student level

Academic > MAP School Year: 2022 Term: Fall Growth Term: Fall - Fall

SUBJECT: ELA Math Grade 4

mstep_scale_score Incomplete visualization

mstep_sgp Incomplete visualization

Bar: None Group: None

Calculator

Calculator: Summary + Add Calculator

Allegan Summary Summary Show more

Apply

eidex Current Account Eidex Account Contact Profile Student level

Academic > MAP School Year: 2022 Term: Fall Growth Term: Fall - Fall

SUBJECT: ELA Math Grade 4

mstep_scale_score Incomplete visualization

mstep_sgp Incomplete visualization

Bar: None Group: None

Calculator

Calculator: Summary + Add Calculator

Allegan Summary Summary Show less

SGP MSTEP & PSAT-8: 50%
 SGP STAR: 0%
 CGP MAP: 50%
 AG MSTEP: 0%
 SLO Tiered Growth: 0%
 SLO Effect Size: 0%

Last Updated: February 23rd, 2023
 Shared By: Danny, 5

Apply

Calculator: **SGP MSTEP** ▾

+ Add Calculator

Allegan SGP MSTEP

SGP MSTEP

School Years: 2020, 2021, 2022


Included Tests: MSTEP, MIACCESS

Included Subjects: ELA, ERW, Math

Min Student Count: 20

Flagged Students Excluded

 Edit

 Share

 Remove

Three-year Weights

2022: 50%

2021: 30%

2020: 20%



Two-year Weights

Second Year: 60%

First Year: 40%

Cut Scores

Highly Effective (4): 60

Effective (3): 40

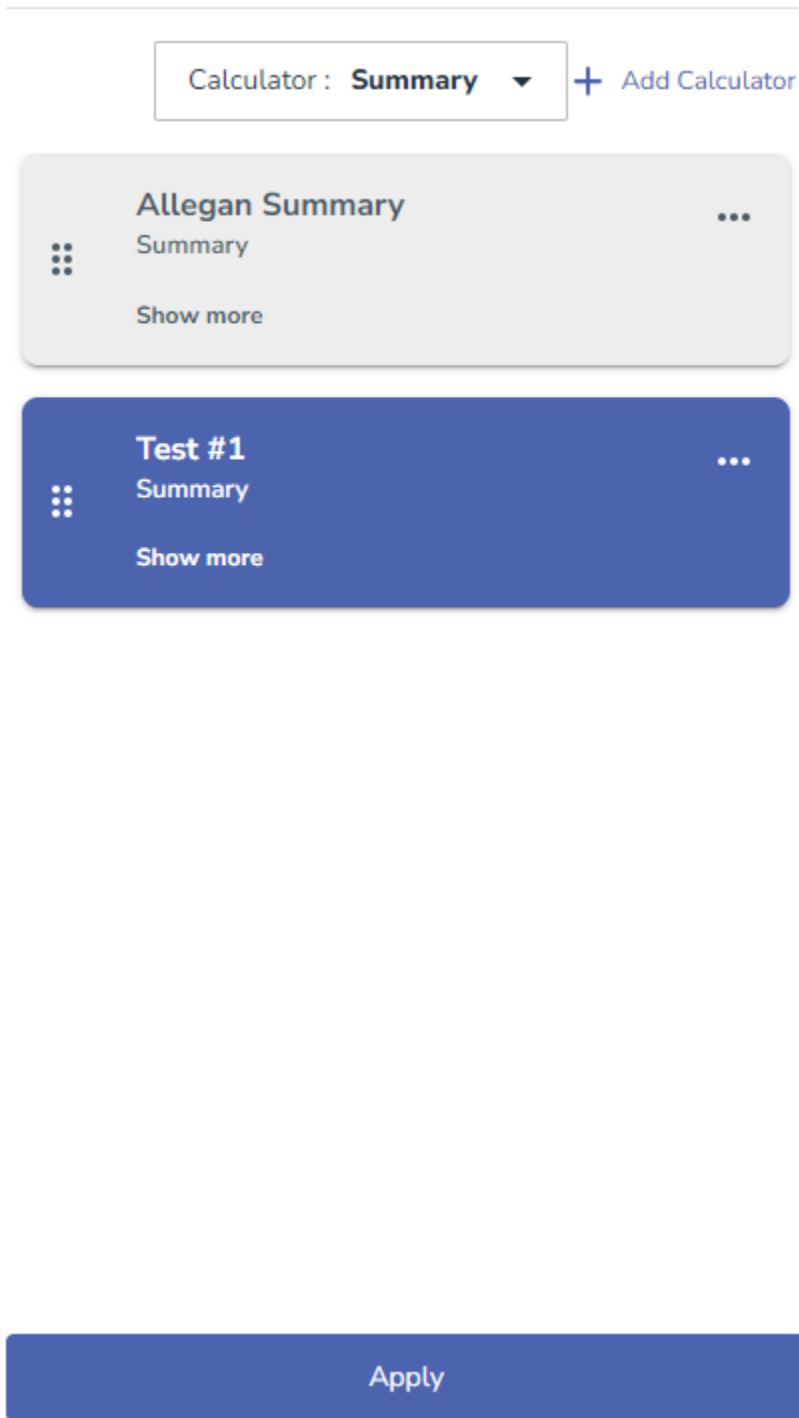
Not Effective (2): 20

Last Updated: February 23rd, 2023

Shared By: Danny. S

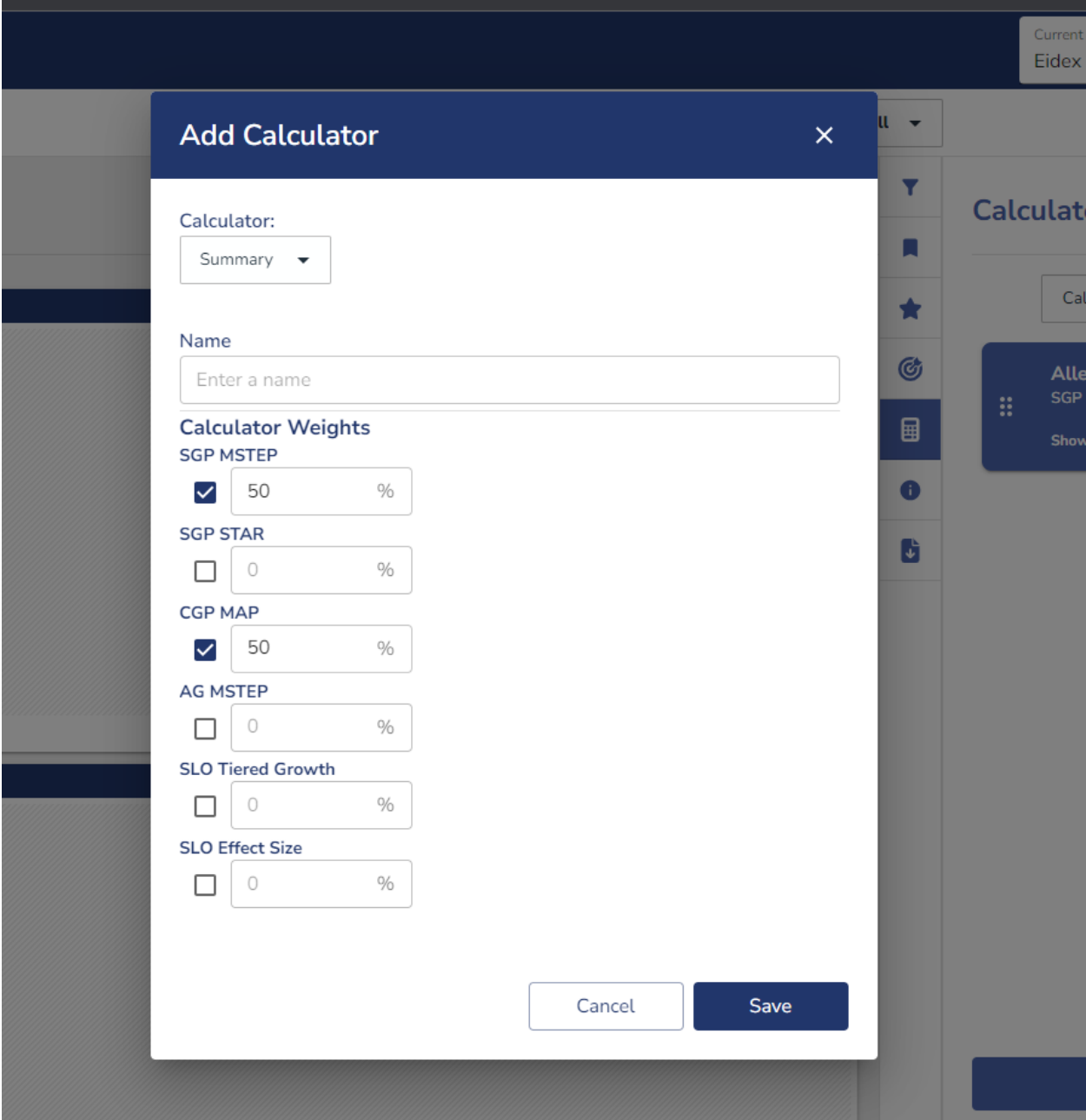
Show less

Apply



The panel itself consisted of functions that were used to save, edit, remove, and add new calculator cards. The calculator cards displayed different sections based on which assessment was selected. The cards are expandable and selectable. When selected the color will adjust and the apply button will be enabled allowing the data from the

calculator to adjust the visualization. Each card has a small popover menu that allows for that card to be edited, shared with another user, or deleted.



Add Calculator ✕

Calculator:
SGP MSTEP ▾

School Years
 2016 2017 2018 2019 2020 2021
 2022 2023

Include Tests
 MSTEP Mi-Access PSAT-8 (MDE)

Include Subjects
 ELA ERW Math Science Social Studies

Min Student Count
20

Exclude Flagged Students:

Three-year Weights
2023 Weight % % %

Two-year Weights
Second Year Weight % %

Cut Scores

Cancel Save

The forms are placed in a modal and since there are multiple forms there is a dropdown that enables the user to change forms to whichever they need. Forms consist of number inputs, text fields, toggles, and checkboxes that are validated using `react-hook-forms`.

The components themselves typically consist of a container that is styled using `styled-components` and then CSS classes inside of that component style all of it's children.

This is the code for the main parent container of the Calculator Panel.

```
import Button from '@mui/material/Button';
import DialogActions from '@mui/material/DialogActions';
import DialogContent from '@mui/material/DialogContent';
import FormControl from '@mui/material/FormControl';
import FormLabel from '@mui/material/FormLabel';
import { useState } from 'react';
import styled from 'styled-components';

import ModalDialog from '../../components/ModalDialog';
import BasicSelect from '../../components/Select/BasicSelect';

import { useCalculatorContext } from './CalculatorContext';
import CalculatorInputForm from './CalculatorInputForm';
import { calculatorOptions, NEW_AG_MSTEP_CALCULATOR_ITEM, NEW_CGP_MAP_CALCULATOR_ITEM, NEW_SGP_MSTEP_CALCULATOR_ITEM, NEW_SGP_STAR_CALCULATOR_ITEM, NEW_SLO_EFFECT_SIZE_CALCULATOR_ITEM, NEW_SLO_TIERED_GROWTH_CALCULATOR_ITEM, NEW_SUMMARY_CALCULATOR_ITEM } from './constants';

import type { ModalDialogProps } from '../../components/ModalDialog';
import type { CalculatorFormType, CalculatorType, Calculator } from './types';

type CalculatorModalProps = Omit<ModalDialogProps, 'title'> & {
  /** calculator type; determines which form is shown */
  calculatorId: CalculatorType;
  /** calculator data object */
  calculator: Calculator;
  /** Type to determine form actions */
  formType: CalculatorFormType;
};

const StyledCalculatorModal = styled<ModalDialog>`
  .modal__contents {
    display: flex;
    flex-direction: column;
    margin-top: 1rem;
    gap: .2rem;
    overflow: auto;
  };

  .modal__actions > .MuiButton-root {
    width: 8rem;
    height: 2.5rem;
    font-size: 1rem;
  };
`;
```

```

    .modal_select {
      margin: 1.5rem;
      margin-bottom: 0px;
    }
  `;

const StyledFormControl = styled(FormControl)`

  margin-bottom: .3rem;

  .MuiFormLabel-root {
    color: ${(props) => props.theme.palette.eidxBlue.main};
    font-size: 1rem;
    font-weight: 600;

    .MuiButtonBase-root {
      margin-left: 1rem;
      padding-inline: 0.2rem 0.5rem;
      background-color: ${(props) => props.theme.palette.fadedBlue.light};
      border-radius: 0.5rem;
      .MuiButton-startIcon {
        margin: 0.2rem;
      }
    }
  }

  .MuiFormHelperText-root {
    margin-inline: 0;
  }
  `;

const StyledBasicSelect = styled(BasicSelect)`
  .dropdown__menu > .MuiPaper-root {
    max-height: 200px;
  }

  .dropdown__button {
    padding: 10px 10px 10px 16px;
    min-width: 125px;
    height: 40px;
    border-radius: 0.125rem;
    border: 1px solid ${(props) => props.theme.palette.lightGray.dark}
  }

  .dropdown__menuItem {
    height: 40px;
  }
  `;

/** A modal that is a container for the Calculator form */
const CalculatorModal = (props: CalculatorModalProps) => {
  const { calculatorId, calculator, formType, open, onClose } = props;

```

```

const { dispatch: calculatorDispatch } = useCalculatorContext();

const newCalculators: { [key in CalculatorType]: Calculator } = {
  calculators_summary: NEW_SUMMARY_CALCULATOR_ITEM,
  calculators_sgp_mstep: NEW_SGP_MSTEP_CALCULATOR_ITEM,
  calculators_sgp_star: NEW_SGP_STAR_CALCULATOR_ITEM,
  calculators_ag_mstep: NEW_AG_MSTEP_CALCULATOR_ITEM,
  calculators_cgp_map: NEW_CGP_MAP_CALCULATOR_ITEM,
  calculators_slo_tieredGrowth: NEW_SLO_TIERED_GROWTH_CALCULATOR_ITEM,
  calculators_slo_effectSize: NEW_SLO_EFFECT_SIZE_CALCULATOR_ITEM,
};

const [selectedType, setSelectedType] = useState<CalculatorType>(calculatorId);
const initCalculator = newCalculators[selectedType];

console.log({ selectedType, calculator, initCalculator });

const handleSubmit = (data: Calculator) => {
  if (formType === 'add') {
    calculatorDispatch({ type: 'ADD_CALCULATOR', newCalculator: data });
    onClose();
  }
  if (formType === 'edit') {
    calculatorDispatch({ type: 'EDIT_CALCULATOR', editCalculator: data });
    onClose();
  }
};

return (
  <StyledCalculatorModal title={`${formType === 'add' ? 'Add' : 'Edit'} Calculator`}
    maxWidth='sm' open={open} onClose={onClose}>
    <StyledFormControl className='modal_select'>
      <FormLabel>Calculator: </FormLabel>
      <StyledBasicSelect
        options={calculatorOptions}
        value={selectedType}
        ButtonProps={{ variant: 'outlined' }}
        onSelect={(id: string) => setSelectedType(id as CalculatorType)}
      />
    </StyledFormControl>
    <DialogContent className='modal__contents'>
      <CalculatorInputForm calculator={formType === 'add' ? initCalculator : calculator}
        onSubmit={handleSubmit} />
    </DialogContent>
    <DialogActions className='modal__actions'>
      <Button variant='outlined' onClick={onClose}>Cancel</Button>
      <Button variant='contained' type='submit' form={selectedType}>Save</Button
    >
    </DialogActions>
  </StyledCalculatorModal>
);
};

export default CalculatorModal;

```

This is the file that determines which form to be displayed inside of the modal.

```
import styled from 'styled-components';

import AgMstepForm from './CalculatorForms/AgMstepForm';
import CgpMapForm from './CalculatorForms/CgpMapForm';
import SgpMstepForm from './CalculatorForms/SgpMstepForm';
import SgpStarForm from './CalculatorForms/SgpStarForm';
import SloEffectSizeForm from './CalculatorForms/SloEffectSizeForm';
import SloTieredGrowthForm from './CalculatorForms/SloTieredGrowthForm';
import SummaryForm from './CalculatorForms/SummaryForm';

import type { Calculator } from './types';

export type CalculatorInputFormProps = {
  /** calculator data with distinct inputs that are determined by the type */
  calculator: Calculator;
  /** callback function for when the form data is submitted */
  onSubmit: (data: Calculator) => void;
};

const StyledCalculatorInputForm = styled.div`
  .form {
    height: 100%;
    display: flex;
    flex-direction: column;
    overflow: scroll;
  }

  .form__section {
    display: flex;
    flex-direction: column;
  }

  .form__header {
    width: 100%;
    font-size: 1.1rem;
    font-weight: 700;
    white-space: nowrap;
    text-overflow: ellipsis;
    color: ${({props}) => props.theme.palette.eidexBlue.main};
  }

  .form__subheader {
    width: 100%;
    font-size: .9rem;
    font-weight: 700;
    overflow: hidden;
    white-space: nowrap;
    text-overflow: ellipsis;
  }
`
```

```

    color: ${({props}) => props.theme.palette.eidexBlue.main};
  }

  .form__inputGroup {
    display: flex;
    gap: 5px;
  }

  .form__checkbox {
    height: 32px
  }

  .form__input--twoYear {
    width: 50%;
  }

  .form__helperText--error {
    color: #ff0000;
  }

  .form__helperText--warning {
    color: #8a6d3b;
  }

  .form__control {
    margin-bottom: .3rem;

    .MuiFormLabel-root {
      color: ${({props}) => props.theme.palette.eidexBlue.main};
      font-size: 1rem;
      font-weight: 600;

      .MuiButtonBase-root {
        margin-left: 1rem;
        padding-inline: 0.2rem 0.5rem;
        background-color: ${({props}) => props.theme.palette.fadedBlue.light};
        border-radius: 0.5rem;
        .MuiButton-startIcon {
          margin: 0.2rem;
        }
      }
    }
  }

  .MuiFormHelperText-root {
    margin-inline: 0;
  }
}

.form__control--switch { // TODO: See if this is actually needed or if it .form__control is sufficient
  display: flex;
  flex-direction: row;
  align-items: center;

```

```

    .MuiFormLabel-root {
      margin-top: .1rem;
    }
  }

  .form__divider {
    margin-top: .75rem;
    margin-bottom: .75rem;
  }

  .form__standardErrors {
    display: flex;
    align-items: center;
    gap: 5px;
  }

  .form__year {
    gap: 8px;
  }

  .input-checkbox {
    height: 32px
  }

  .input__numberInput {
    width: 150px;
  }
};

/** A container that determines which Calculator form is shown depending on the type */
const CalculatorInputForm = (props: CalculatorInputFormProps) => {
  const { type } = props.calculator;

  return (
    <StyledCalculatorInputForm>
      {
        type === 'calculators_summary' ? <SummaryForm {...props} />
        : type === 'calculators_sgp_mstep' ? <SgpMstepForm {...props} />
        : type === 'calculators_sgp_star' ? <SgpStarForm {...props} />
        : type === 'calculators_cgp_map' ? <CgpMapForm {...props} />
        : type === 'calculators_ag_mstep' ? <AgMstepForm {...props} />
        : type === 'calculators_slo_tieredGrowth' ? <SloTieredGrowthForm {...prop
s} />
        : type === 'calculators_slo_effectSize' ? <SloEffectSizeForm {...props} />
        : null
      }
    </StyledCalculatorInputForm>
  );
};

export default CalculatorInputForm;

```

This is the code for one of the forms. This is one of the more complex forms.

```
import { Checkbox, Divider, FormControl, FormGroup, FormHelperText, FormLabel, InputAdornment, Switch, TextField, Typography } from '@mui/material';
import { useEffect } from 'react';
import { useForm, useWatch } from 'react-hook-form';

import mathUtils from '../../utils/mathUtils';

import type { CalculatorInputFormProps } from '../CalculatorInputForm';
import type { Calculator, SgpMstepInputs } from '../types';
import type { FieldErrors } from 'react-hook-form';

const SgpMstepForm = (props: CalculatorInputFormProps) => {
  const { calculator, onSubmit } = props;

  const {
    register,
    handleSubmit,
    formState: { errors },
    control,
    getValues,
    trigger,
  } = useForm<Calculator>({ defaultValues: calculator });
  const [inputs] = useWatch({ control, name: ['inputs'] }) as [SgpMstepInputs];
  const inputErrors = errors.inputs as FieldErrors<SgpMstepInputs> || {};

  const VALIDATION_RULES = {
    label: {
      required: 'Please provide a name.',
      maxLength: {
        value: 50,
        message: 'You cannot exceed more than 50 characters',
      },
    },
    excludeStudents: {
      setValueAs: (v: boolean) => typeof v === 'boolean',
    },
    sgpMstepInputsYears: {
      validate: {
        selectOneYear: (v: string[]) => !v.length ? 'Must have at least one year selected' : undefined,
      },
    },
    sgpMstepInputsTests: {
      validate: {
        selectOneTest: (v: string[]) => !v.length ? 'Must include at least one test' : undefined,
      },
    },
    sgpMstepInputsSubjects: {
      validate: {
```



```

        selectOneSubject: (v: string[]) => !v.length ? 'Must include at least one
subject' : undefined,
    },
},
sgpMstepInputsThreeYearWeights: {
    valueAsNumber: true,
    validate: {
        weightMustBeNumber: () => (Number.isNaN(getValues('inputs.threeYearWeight
3')))
        || Number.isNaN(getValues('inputs.threeYearWeight2')) || Number.isNaN(getV
alues('inputs.threeYearWeight1'))
        ? 'Input value must be a number' : undefined,
        between0And100: () => ((getValues('inputs.threeYearWeight3') < 0 || getVal
ues('inputs.threeYearWeight3') > 100)
        || (getValues('inputs.threeYearWeight2') < 0 || getValues('inputs.threeYea
rWeight2') > 100)
        || (getValues('inputs.threeYearWeight1') < 0 || getValues('inputs.threeYea
rWeight1') > 100))
        ? 'Number must be between 0 and 100' : undefined,
        sumEquals100: () => (getValues('inputs.threeYearWeight3') + getValues('inp
uts.threeYearWeight2')
        + getValues('inputs.threeYearWeight1')) !== 100
        ? 'Sum of weights must be 100%' : undefined,
    },
},
sgpMstepInputsTwoYearWeights: {
    valueAsNumber: true,
    validate: {
        weightMustBeNumber: () => (Number.isNaN(getValues('inputs.twoYearWeight
2')) || Number.isNaN(getValues('inputs.twoYearWeight1')))
        ? 'Input value must be a number' : undefined,
        between0And100: () => ((getValues('inputs.twoYearWeight2') < 0 || getValu
es('inputs.twoYearWeight2') > 100)
        || (getValues('inputs.twoYearWeight1') < 0 || getValues('inputs.twoYea
rWeight1') > 100))
        ? 'Number must be between 0 and 100' : undefined,
        sumEquals100: () => (getValues('inputs.twoYearWeight2') + getValues('input
s.twoYearWeight1')) !== 100 ? 'Sum of weights must be 100%' : undefined,
    },
},
sgpMstepInputsCutScore: {
    valueAsNumber: true,
    validate: {
        weightMustBeNumber: () => (Number.isNaN(getValues('inputs.cutScore3'))
        || Number.isNaN(getValues('inputs.cutScore2')) || Number.isNaN(getValues
('inputs.cutScore1')))
        ? 'Input value must be a number' : undefined,
        between0And100: () => ((getValues('inputs.cutScore3') < 0 || getValues('in
puts.cutScore3') > 100)
        || (getValues('inputs.cutScore2') < 0 || getValues('inputs.cutScore2')
> 100)
        || (getValues('inputs.cutScore1') < 0 || getValues('inputs.cutScore1')
> 100))
        ? 'Number must be between 0 and 100' : undefined,
    },
},

```

```

        isInOrder: () => !(mathUtils.isInRange(getValues('inputs.cutScore1'), 0, g
etValues('inputs.cutScore2'))
        && mathUtils.isInRange(getValues('inputs.cutScore2'), getValues('inputs.cu
tScore1'), getValues('inputs.cutScore3'))
        && mathUtils.isInRange(getValues('inputs.cutScore3'), getValues('inputs.cu
tScore2'), 100)) ? 'Cutscores must be in ascending order' : undefined,
    },
  },
  sgpMstepInputsMinStudents: {
    valueAsNumber: true,
    validate: {
      minStudentsMustBeNumber: (v: number) => Number.isNaN(v) ? 'Input value mus
t be a number' : undefined,
      minStudentsPositive: (v: number) => v < 0 ? 'Number must be positive' : un
defined,
    },
  },
};

// triggers error on threeYearWeights
useEffect(() => {
  trigger('inputs.threeYearWeight3');
}, [inputs.threeYearWeight3, inputs.threeYearWeight2, inputs.threeYearWeight1]);

// triggers error on twoYearWeights
useEffect(() => {
  trigger('inputs.twoYearWeight2');
}, [inputs.twoYearWeight2, inputs.twoYearWeight1]);

// triggers error on cutScores
useEffect(() => {
  trigger('inputs.cutScore3');
}, [inputs.cutScore3, inputs.cutScore2, inputs.cutScore1]);

return (
  <form className='form' id={calculator.type} noValidate onSubmit={handleSubmit(onSu
bmit)}>
    <FormControl className='form__control'>
      <FormLabel>Name</FormLabel>
      <TextField
        variant='outlined'
        size='small'
        {...register('label', { ...VALIDATION_RULES.label })}
        error={!!errors.label}
        helperText={errors.label?.message}
        placeholder='Enter a name'
        inputProps={{ maxLength: 50 }}
      />
    </FormControl>
    <Divider />
    <FormControl className='form__control'>
      <FormLabel component='legend'>School Years</FormLabel>
      <FormGroup row sx={{ gap: '8px' }}>
        {

```

```

        inputs.availableYears.map((year) => (
            <Typography key={year} display='flex' alignItems='center'>
                <Checkbox checked={inputs.selectedYears.includes(year)} di
sabled={inputs.selectedYears.length === 3 && !inputs.selectedYears.includes(year)} {...reg
ister('inputs.selectedYears', { ...VALIDATION_RULES.sgpMstepInputsYears })} value={year} /
>
                    {year}
                </Typography>
            ))
        )
    </FormGroup>
    <FormHelperText className='form__helperText--error'>{inputErrors.selectedY
ears?.message?.toString()}</FormHelperText>
</FormControl>
<FormControl className='form__control'>
    <FormLabel component='legend'>Include Tests</FormLabel>
    <FormGroup row>
        <Typography display='flex' alignItems='center' marginRight='8px'>
            <Checkbox checked={inputs.includeTests.includes('MSTEP')} {...regi
ster('inputs.includeTests', { ...VALIDATION_RULES.sgpMstepInputsTests })} value='MSTEP' />
                MSTEP
            </Typography>
            <Typography display='flex' alignItems='center' marginRight='8px'>
                <Checkbox checked={inputs.includeTests.includes('MIACCESS')} {...r
egister('inputs.includeTests', { ...VALIDATION_RULES.sgpMstepInputsTests })} value='MIACCE
SS' />
                    Mi-Access
                </Typography>
                <Typography display='flex' alignItems='center' marginRight='8px'>
                    <Checkbox checked={inputs.includeTests.includes('PSAT-MDE')} {...r
egister('inputs.includeTests', { ...VALIDATION_RULES.sgpMstepInputsTests })} value='PSAT-M
DE' />
                        PSAT-8 (MDE)
                    </Typography>
                </FormGroup>
                <FormHelperText className='form__helperText--error'>{inputErrors.includeTe
sts?.message?.toString()}</FormHelperText>
            </FormControl>
            <FormControl className='form__control'>
                <FormLabel component='legend'>Include Subjects</FormLabel>
                <FormGroup row>
                    <Typography display='flex' alignItems='center' marginRight='8px'>
                        <Checkbox checked={inputs.includeSubjects.includes('ELA')} {...reg
ister('inputs.includeSubjects', { ...VALIDATION_RULES.sgpMstepInputsSubjects })} value='EL
A' />
                            ELA
                        </Typography>
                        <Typography display='flex' alignItems='center' marginRight='8px'>
                            <Checkbox checked={inputs.includeSubjects.includes('ERW')} {...reg
ister('inputs.includeSubjects', { ...VALIDATION_RULES.sgpMstepInputsSubjects })} value='ER
W' />
                                ERW
                            </Typography>
                            <Typography display='flex' alignItems='center' marginRight='8px'>

```

```

        <Checkbox checked={inputs.includeSubjects.includes('Math')} {...re
gister('inputs.includeSubjects', { ...VALIDATION_RULES.sgpMstepInputsSubjects })} value='M
ath' />
        Math
        </Typography>
        <Typography display='flex' alignItems='center' marginRight='8px'>
        <Checkbox checked={inputs.includeSubjects.includes('Science')}
        {...register('inputs.includeSubjects', { ...VALIDATION_RULES.sgpMstepInputsSubjects })} v
alue='Science' />
        Science
        </Typography>
        <Typography display='flex' alignItems='center' marginRight='8px'>
        <Checkbox checked={inputs.includeSubjects.includes('Social Studie
s')} {...register('inputs.includeSubjects', { ...VALIDATION_RULES.sgpMstepInputsSubjects
})} value='ELA' />
        Social Studies
        </Typography>
    </FormGroup>
    <FormHelperText className='form__helperText --error'>{inputErrors.includeSu
bjects?.message?.toString()}</FormHelperText>
</FormControl>
<FormControl className='form__control'>
    <FormLabel component='legend'>Min Student Count</FormLabel>
    <TextField
        type='number'
        size='small'
        margin='dense'
        error={!inputErrors.minStudents?.message}
        helperText={inputErrors.minStudents?.message}
        sx={{ width: '150px' }}
        {...register('inputs.minStudents', { ...VALIDATION_RULES.sgpMstepInput
sMinStudents })}
    />
</FormControl>
<FormControl className='form__control form__control--switch'>
    <FormLabel>Exclude Flagged Students: </FormLabel>
    <Switch
        checked={inputs.excludeStudents}
        {...register('inputs.excludeStudents', { ...VALIDATION_RULES.excludeSt
udents })}
    />
</FormControl>
<Divider className='form_divider' />
{
    inputs.selectedYears.length === 3 && (
        <div className='form__section'>
            <Typography className='form__header'>Three-year Weights</Typograph
y>
            <div className='form__inputGroup'>
                <TextField
                    type='number'
                    label={` ${inputs.selectedYears[0]} Weight`}
                    size='small'

```

```

        margin='dense'
        error={!!inputErrors.threeYearWeight3?.message}
        {...register('inputs.threeYearWeight3', { ...VALIDATION_RULES.sgpMstepInputsThreeYearWeights })}
        InputProps={
          {
            endAdornment: <InputAdornment position='end'%>/InputAdornment>,
          }
        }
      />
    <TextField
      type='number'
      label={`${inputs.selectedYears[1]} Weight`}
      size='small'
      margin='dense'
      error={!!inputErrors.threeYearWeight3?.message}
      {...register('inputs.threeYearWeight2', { ...VALIDATION_RULES.sgpMstepInputsThreeYearWeights })}
      InputProps={
        {
          endAdornment: <InputAdornment position='end'%>/InputAdornment>,
        }
      }
    />
    <TextField
      type='number'
      label={`${inputs.selectedYears[2]} Weight`}
      size='small'
      margin='dense'
      error={!!inputErrors.threeYearWeight3?.message}
      {...register('inputs.threeYearWeight1', { ...VALIDATION_RULES.sgpMstepInputsThreeYearWeights })}
      InputProps={
        {
          endAdornment: <InputAdornment position='end'%>/InputAdornment>,
        }
      }
    />
  </div>
  <FormHelperText className='form__helperText--error'>{inputErrors.threeYearWeight3?.message?.toString()}</FormHelperText>
  <Divider className='form_divider' />
</div>
)
}
<div className='form__section'>
  <Typography className='form__header'>Two-year Weights</Typography>
  <div className='form__inputGroup'>
    <TextField
      className='form__input--twoYear'
      type='number'

```

```

        label={`${inputs.selectedYears.length === 3 ? 'Second Year' : inputs.selectedYears[0]} Weight`}
        size='small'
        margin='dense'
        error={!inputErrors.twoYearWeight2?.message}
        {...register('inputs.twoYearWeight2', { ...VALIDATION_RULES.sgpMstepInputsTwoYearWeights })}
        InputProps={
          {
            endAdornment: <InputAdornment position='end'>%</InputAdornment>,
          }
        }
      />
    <TextField
      className='form__input--twoYear'
      type='number'
      label={`${inputs.selectedYears.length === 3 ? 'First Year' : inputs.selectedYears[1]} Weight`}
      size='small'
      margin='dense'
      error={!inputErrors.twoYearWeight2?.message}
      {...register('inputs.twoYearWeight1', { ...VALIDATION_RULES.sgpMstepInputsTwoYearWeights })}
      InputProps={
        {
          endAdornment: <InputAdornment position='end'>%</InputAdornment>,
        }
      }
    />
  </div>
  <FormHelperText className='form__helperText--error'>{inputErrors.twoYearWeight2?.message?.toString()}</FormHelperText>
</div>
<Divider className='form_divider' />
<div className='form__section'>
  <Typography className='form__header'>Cut Scores</Typography>
  <div className='form__inputGroup'>
    <TextField
      type='number'
      label='Highly Effective (4)'
      size='small'
      margin='dense'
      error={!inputErrors.cutScore3?.message}
      {...register('inputs.cutScore3', { ...VALIDATION_RULES.sgpMstepInputsCutScore })}
    />
    <TextField
      type='number'
      label='Effective (3)'
      size='small'
      margin='dense'
      error={!inputErrors.cutScore3?.message}

```

```

        {...register('inputs.cutScore2', { ...VALIDATION_RULES.sgpMstepInp
utsCutScore })}
      />
      <TextField
        type='number'
        label='Not Effective (2)'
        size='small'
        margin='dense'
        error={!inputErrors.cutScore3?.message}
        {...register('inputs.cutScore1', { ...VALIDATION_RULES.sgpMstepInp
utsCutScore })}
      />
    </div>
    <FormHelperText className='form__helperText--error'>{inputErrors.cutScore
3?.message?.toString()}</FormHelperText>
  </div>
</form>
  );
};

export default SgpMstepForm;

```

Similar to the forms there are cards for each one, and a component that determines which card is used depending on the type.

```

import { Divider } from '@mui/material';

import type { SgpMstepInputs } from '../types';

type SgpMstepCardProps = {
  inputs: SgpMstepInputs;
};

/** A selectable card that displays information about the calculator */
const SgpMstepCard = (props: SgpMstepCardProps) => {
  const { inputs } = props;

  return (
    <div className='card__contentBody'>
      <div className='card__rowInfo'>
        <h4>School Years:</h4>
        <div className='card__typography'>{inputs.selectedYears.reverse().join(',
')}}</div>
      </div>
      <div className='card__rowInfo'>
        <h4>Included Tests:</h4>
        <div className='card__typography'>{inputs.includeTests.join(', ')}</div>
      </div>
      <div className='card__rowInfo'>
        <h4>Included Subjects:</h4>

```

```

        <div className='card__typography'>{inputs.includeSubjects.join(', ')}</div
    >
    </div>
    <div className='card__rowInfo'>
        <h4>Min Student Count:</h4>
        <div className='card__typography'>{inputs.minStudents}</div>
    </div>
    <div className='card__rowInfo'>
        <h4>{inputs.excludeStudents ? 'Flagged Students Excluded' : 'Flagged Stud
nts Included'}</h4>
    </div>
    <Divider className='card__divider' />
    {
        inputs.selectedYears.length === 3 && (
            <>
                <div className='card__subGroup'>
                    <h3>Three-year Weights</h3>
                    <div className='card__subGroup__content'>
                        /* eslint-disable-next-line react/jsx-one-expression-per-
line */
                        <h4>`${inputs.selectedYears[2]}: `</h4>
                        <div className='card__typography'>`${inputs.threeYearWeig
ht3}%`</div>
                    </div>
                    <div className='card__subGroup__content'>
                        /* eslint-disable-next-line react/jsx-one-expression-per-
line */
                        <h4>`${inputs.selectedYears[1]}: `</h4>
                        <div className='card__typography'>`${inputs.threeYearWeig
ht2}%`</div>
                    </div>
                    <div className='card__subGroup__content'>
                        /* eslint-disable-next-line react/jsx-one-expression-per-
line */
                        <h4>`${inputs.selectedYears[0]}: `</h4>
                        <div className='card__typography'>`${inputs.threeYearWeig
ht1}%`</div>
                    </div>
                </div>
                <Divider className='card__divider' />
            </>
        )
    }
    <div className='card__subGroup'>
        <h3>Two-year Weights</h3>
        <div className='card__subGroup__content'>
            /* eslint-disable-next-line react/jsx-one-expression-per-line */
            <h4>{inputs.selectedYears.length === 3 ? 'Second Year: ' : `${inputs.s
electedYears[1]}: `}</h4>
            <div className='card__typography'>`${inputs.twoYearWeight2}%`</div>
        </div>
        <div className='card__subGroup__content'>
            /* eslint-disable-next-line react/jsx-one-expression-per-line */
            <h4>{inputs.selectedYears.length === 3 ? 'First Year: ' : `${inputs.se

```



```

lectedYears[0]]: `}</h4>
      <div className='card__typography'>`${inputs.twoYearWeight1}%`</div>
    </div>
  </div>
  <Divider className='card__divider' />
  <div className='card__subGroup'>
    <h3>Cut Scores</h3>
    <div className='card__subGroup__content'>
      <h4>Highly Effective (4): </h4>
      <div className='card__typography'>{inputs.cutScore3}</div>
    </div>
    <div className='card__subGroup__content'>
      <h4>Effective (3): </h4>
      <div className='card__typography'>{inputs.cutScore2}</div>
    </div>
    <div className='card__subGroup__content'>
      <h4>Not Effective (2): </h4>
      <div className='card__typography'>{inputs.cutScore1}</div>
    </div>
  </div>
</div>
);
};

export default SgpMstepCard;

```

Project #2 - NotesGalore - <https://note-taking-app-pn8h.vercel.app/>

This was a simple project that I made exploring using an npm package that I was asked to incorporate in a project for a client that has me work on his website part-time. This project was built with NextJS and incorporates some of the updated routing that was recently implemented in NextJS. The editor comes from a package called TipTap. It's an incredibly versatile and customizable editor, while also maintaining a sleek look. Login uses a package called Clerk/NextJS which includes components that allow easily create login and logout components that use Google Authentication. This application used TailwindCSS.

Welcome to Notesgalore

[Sign-up](#)

[Sign-in](#)

Secured by  clerk

Sign in

to continue to notion-app



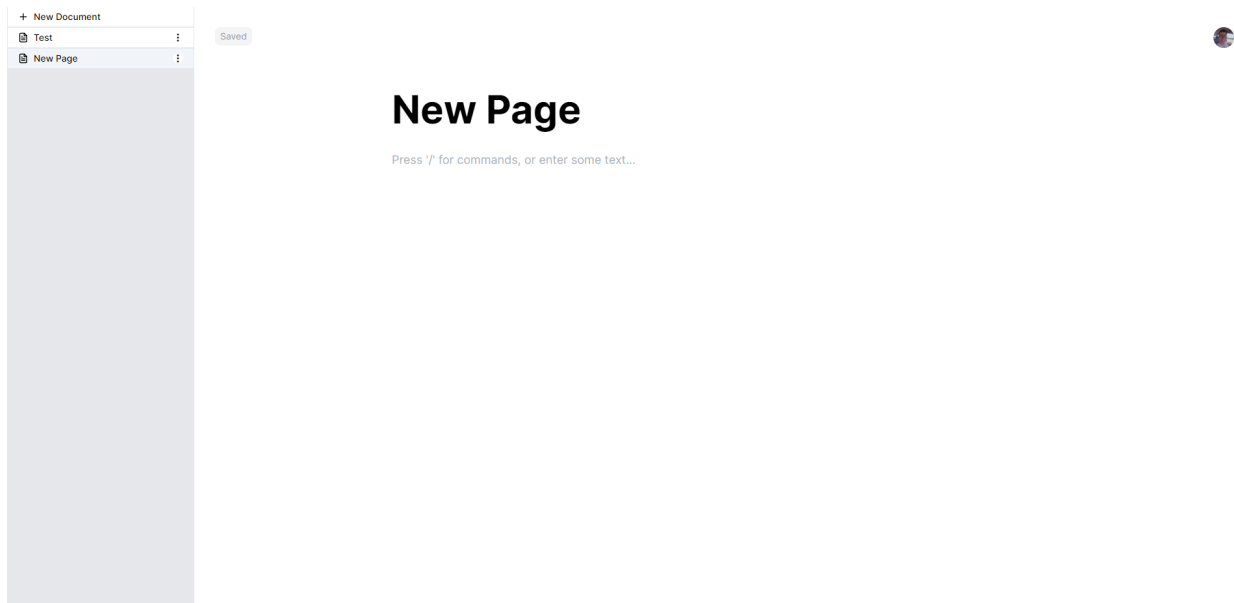
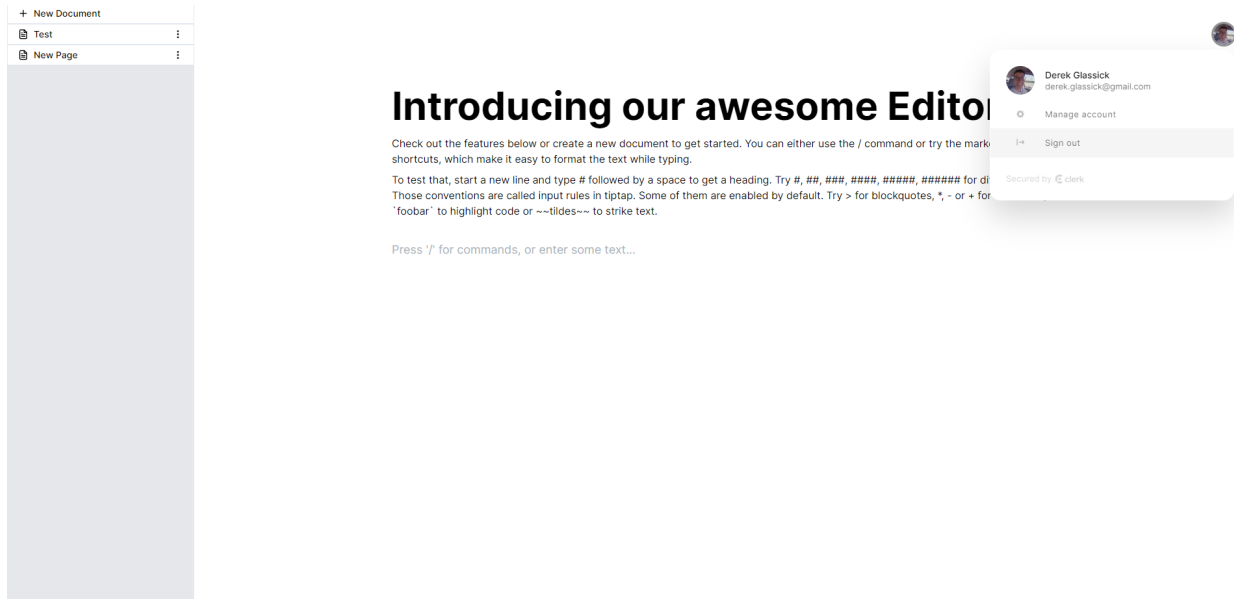
Continue with Google

or

Email address

CONTINUE

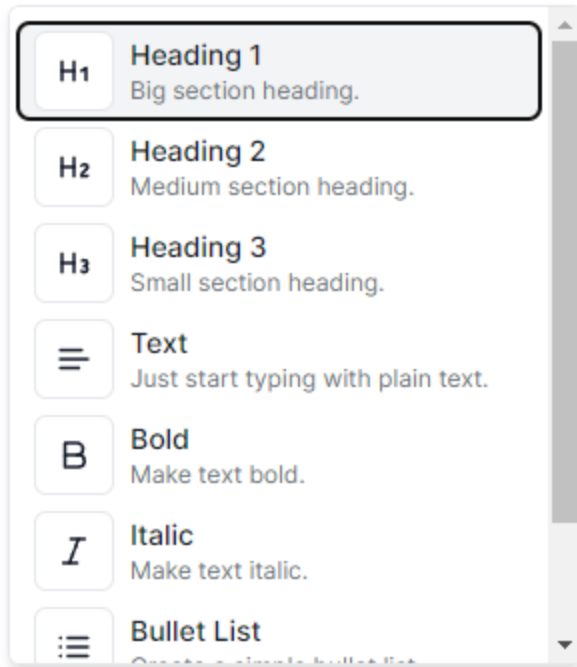
No account? [Sign up](#)



The application allows you to create several pages to take notes. Up in the top right corner you can manage your account or log out. Application works in a very similar way to Notion. After typing a `/` you can select which component you're wanting to type with and take notes. Notes are saved to an SQL data base on PlanetScale. Prisma is the ORM that I'm using to connect my frontend to my database.

... ..

/



Code for the editor is below. Includes debounced function that allows for automatic saving to the database after a set timeout has been reached.

```
"use client";

import { useEditor, EditorContent } from "@tiptap/react";
import { useState, useEffect, useTransition } from "react";
import { useRouter } from "next/navigation";
import { TipTapEditorExtensions } from "../extensions";
import { TipTapEditorProps } from "../props";
import { PatchDocType } from "@app/api/documents/[publicId]/route";
import { useDebouncedCallback } from "use-debounce";

export default function Editor({
  document,
```

```

    publicId,
  }: {
    document: PatchDocType;
    publicId: string;
  }) {
    const router = useRouter();
    const [isPending, startTransition] = useTransition();
    const [saveStatus, setSaveStatus] = useState<string>("Saved");
    const [hydrated, setHydrated] = useState<boolean>(false);
    const [content, setContent] = useState<PatchDocType["document"]>();

    async function patchRequest(publicId: string, title: string, document: any) {
      const response = await fetch(`/api/documents/${publicId}`, {
        method: "PATCH",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify({
          title: title,
          document: document,
        }),
      });

      if (!response.ok) {
        setSaveStatus("Waiting to Save.");
        throw new Error("Failed to update document");
      }

      setSaveStatus("Saved");

      startTransition(() => {
        // Force a cache invalidation.
        router.refresh();
      });
    }

    const debouncedUpdates = useDebounceCallback(async ({ editor }) => {
      const json = editor.getJSON();
      setContent(json);
      await patchRequest(publicId, document.title, json);
      // Simulate a delay in saving.
      setTimeout(() => {
        setSaveStatus("Saved");
      }, 500);
    }, 1000);

    const editor = useEditor({
      extensions: TipTapEditorExtensions,
      editorProps: TipTapEditorProps,
      onUpdate: (e) => {
        setSaveStatus("Saving...");
        debouncedUpdates(e);
      },
      content: content,

```

```


});

// Hydrate the editor with the content from the database.
useEffect(() => {
  if (editor && document && !hydrated) {
    editor.commands.setContent(document.document);
    setHydrated(true);
  }
}, [editor, document, hydrated]);

console.log(document)

return (
  <div
    onClick={() => {
      editor?.chain().focus().run();
    }}
    className="relative flex min-h-screen w-full cursor-text flex-col items-center
p-32"
  >
    <div className="w-full max-w-screen-lg">
      <div className="absolute left-8 top-8 rounded-lg bg-gray-100 px-2 py-1 tex
t-sm text-gray-400">
        {saveStatus}
      </div>
      <h1 className="mb-8 text-6xl font-bold">{document.title}</h1>
      <EditorContent editor={editor} />
    </div>
  </div>
);
}

```

This is the code for the menu that pops up when you type  .

```

import React, {
  useState,
  useEffect,
  useCallback,
  ReactNode,
  useRef,
  useLayoutEffect,
} from "react";
import { Editor, Range, Extension } from "@tiptap/core";
import Suggestion from "@tiptap/suggestion";
import { ReactRenderer } from "@tiptap/react";
import tippy from "tippy.js";
import {
  Bold,
  Heading1,
  Heading2,
  Heading3,

```

```

    Italic,
    List,
    ListOrdered,
    MessageSquarePlus,
    Text,
  } from "lucide-react";

interface CommandItemProps {
  title: string;
  description: string;
  icon: ReactNode;
}

interface Command {
  editor: Editor;
  range: Range;
}

const Command = Extension.create({
  name: "slash-command",
  addOptions() {
    return {
      suggestion: {
        char: "/",
        command: ({
          editor,
          range,
          props,
        }): {
          editor: Editor;
          range: Range;
          props: any;
        }) => {
          props.command({ editor, range });
        },
      },
    };
  },
  addProseMirrorPlugins() {
    return [
      Suggestion({
        editor: this.editor,
        ...this.options.suggestion,
      }),
    ];
  },
});

const getSuggestionItems = ({ query }: { query: string }) => {
  return [
    {
      title: "Heading 1",
      description: "Big section heading.",
      icon: <Heading1 size={18} />,
    },
  ];
};

```



```

command: ({ editor, range }: Command) => {
  editor
    .chain()
    .focus()
    .deleteRange(range)
    .setNode("heading", { level: 1 })
    .run();
},
},
{
  title: "Heading 2",
  description: "Medium section heading.",
  icon: <Heading2 size={18} />,
  command: ({ editor, range }: Command) => {
    editor
      .chain()
      .focus()
      .deleteRange(range)
      .setNode("heading", { level: 2 })
      .run();
  },
},
{
  title: "Heading 3",
  description: "Small section heading.",
  icon: <Heading3 size={18} />,
  command: ({ editor, range }: Command) => {
    editor
      .chain()
      .focus()
      .deleteRange(range)
      .setNode("heading", { level: 3 })
      .run();
  },
},
{
  title: "Text",
  description: "Just start typing with plain text.",
  icon: <Text size={18} />,
  command: ({ editor, range }: Command) => {
    editor
      .chain()
      .focus()
      .deleteRange(range)
      .toggleNode("paragraph", "paragraph")
      .run();
  },
},
{
  title: "Bold",
  description: "Make text bold.",
  icon: <Bold size={18} />,
  command: ({ editor, range }: Command) => {
    editor.chain().focus().deleteRange(range).setMark("bold").run();
  },
},
}

```

```

    },
  },
  {
    title: "Italic",
    description: "Make text italic.",
    icon: <Italic size={18} />,
    command: ({ editor, range }: Command) => {
      editor.chain().focus().deleteRange(range).setMark("italic").run();
    },
  },
  {
    title: "Bullet List",
    description: "Create a simple bullet list.",
    icon: <List size={18} />,
    command: ({ editor, range }: Command) => {
      editor.chain().focus().deleteRange(range).toggleBulletList().run();
    },
  },
  {
    title: "Numbered List",
    description: "Create a list with numbering.",
    icon: <ListOrdered size={18} />,
    command: ({ editor, range }: Command) => {
      editor.chain().focus().deleteRange(range).toggleOrderedList().run();
    },
  },
].filter((item) => {
  if (typeof query === "string" && query.length > 0) {
    return item.title.toLowerCase().includes(query.toLowerCase());
  }
  return true;
});
// .slice(0, 10);
};

export const updateScrollView = (container: HTMLElement, item: HTMLElement) => {
  const containerHeight = container.offsetHeight;
  const itemHeight = item ? item.offsetHeight : 0;

  const top = item.offsetTop;
  const bottom = top + itemHeight;

  if (top < container.scrollTop) {
    container.scrollTop -= container.scrollTop - top + 5;
  } else if (bottom > containerHeight + container.scrollTop) {
    container.scrollTop += bottom - containerHeight - container.scrollTop + 5;
  }
};

const CommandList = ({
  items,
  command,
}): {
  items: CommandItemProps[];

```

```

    command: any;
  }) => {
    const [selectedIndex, setSelectedIndex] = useState(0);
    const commandListContainer = useRef<HTMLDivElement>(null);
    const selectedButtonRef = useRef<HTMLButtonElement>(null);

    const selectItem = useCallback(
      (index: number) => {
        const item = items[index];
        if (item) {
          command(item);
        }
      },
      [command, items]
    );

    useEffect(() => {
      const navigationKeys = ["ArrowUp", "ArrowDown", "Enter"];
      const onKeyDown = (e: KeyboardEvent) => {
        if (navigationKeys.includes(e.key)) {
          e.preventDefault();
          if (e.key === "ArrowUp") {
            setSelectedIndex((selectedIndex + items.length - 1) % items.length);
            return true;
          }
          if (e.key === "ArrowDown") {
            setSelectedIndex((selectedIndex + 1) % items.length);
            return true;
          }
          if (e.key === "Enter") {
            selectItem(selectedIndex);
            return true;
          }
          return false;
        }
      };
      document.addEventListener("keydown", onKeyDown);
      return () => {
        document.removeEventListener("keydown", onKeyDown);
      };
    }, [items, selectedIndex, setSelectedIndex, selectItem]);

    useEffect(() => {
      setSelectedIndex(0);
    }, [items]);

    useEffect(() => {
      const container = commandListContainer.current;
      const item = selectedButtonRef.current;

      if (item && container) {
        container.scrollTop = item.offsetTop - container.offsetTop;

        item.focus();
      }
    });
  }
}

```

```

    }

    if (selectedIndex === 0 && items.length > 0) {
      setTimeout(() => {
        selectedButtonRef.current?.focus();
      }, 10);
    }
  }, [selectedIndex, items]);

  return items.length > 0 ? (
    <div
      ref={commandListContainer}
      className="z-50 h-auto max-h-[330px] w-72 overflow-y-auto scroll-smooth rounded-md border border-gray-200 bg-white px-1 py-2 shadow-md transition-all"
    >
      {items.map((item: CommandItemProps, index: number) => {
        const isSelected = index === selectedIndex;
        return (
          <button
            ref={isSelected ? selectedButtonRef : null}
            className={`flex w-full items-center space-x-2 rounded-md px-2 py-1 text-left text-sm text-gray-900 hover:bg-gray-100 ${isSelected ? "bg-gray-100 text-gray-900" : ""}
              `}
            key={index}
            onClick={() => selectItem(index)}
          >
            <div className="flex h-10 w-10 items-center justify-center rounded-md border border-gray-200 bg-white">
              {item.icon}
            </div>
            <div>
              <p className="font-medium">{item.title}</p>
              <p className="text-xs text-gray-500">{item.description}</p>
            </div>
          </button>
        );
      })}
    </div>
  ) : null;
};

const renderItem = () => {
  let component: ReactRenderer | null = null;
  let popup: any | null = null;

  return {
    onStart: (props: { editor: Editor; clientRect: DOMRect }) => {
      component = new ReactRenderer(CommandList, {
        props,
        editor: props.editor,
      });
    };
  };
};

// @ts-ignore

```

```

    popup = tippy("body", {
      getReferenceClientRect: props.clientRect,
      appendTo: () => document.body,
      content: component.element,
      showOnCreate: true,
      interactive: true,
      trigger: "manual",
      placement: "bottom-start",
    });
  },
  onUpdate: (props: { editor: Editor; clientRect: DOMRect }) => {
    component?.updateProps(props);

    popup &&
      popup[0].setProps({
        getReferenceClientRect: props.clientRect,
      });
  },
  onKeyDown: (props: { event: KeyboardEvent }) => {
    if (props.event.key === "Escape") {
      popup?.[0].hide();

      return true;
    }

    // @ts-ignore
    return component?.ref?.onKeyDown(props);
  },
  onExit: () => {
    popup?.[0].destroy();
    component?.destroy();
  },
};
};

const SlashCommand = Command.configure({
  suggestion: {
    items: getSuggestionItems,
    render: renderItem,
  },
});

export default SlashCommand;

```