React 19 + Server Driven UIs

A Perfect Match

Ameer Sami

@ameersami.com | ameersami.com

Ameer Sami

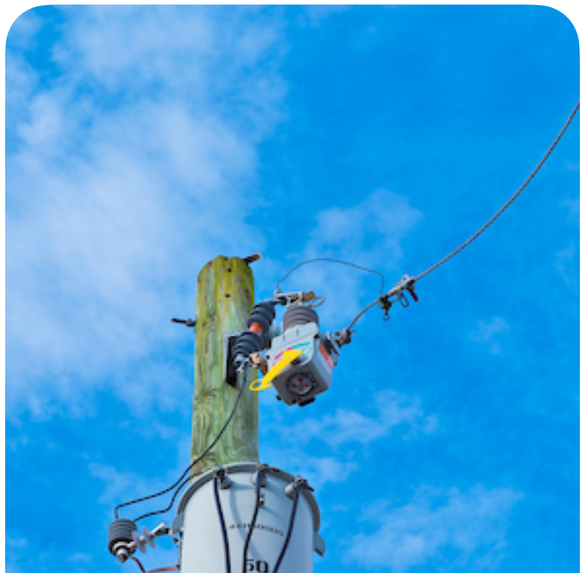
Senior Software Engineer - S&C Electric Company


netlify
it works on
my machine
RUN
BSD
facebook
NO DEPLOY FRIDAY


ROGUE

S&C ELECTRIC COMPANY

S&C ELECTRIC COMPANY



What is SDUI

How can React 19 help

SDUI + React 19 in practice

What is SDUI

How can React 19 help

SDUI + React 19 in practice

Server **D**riven **UI**

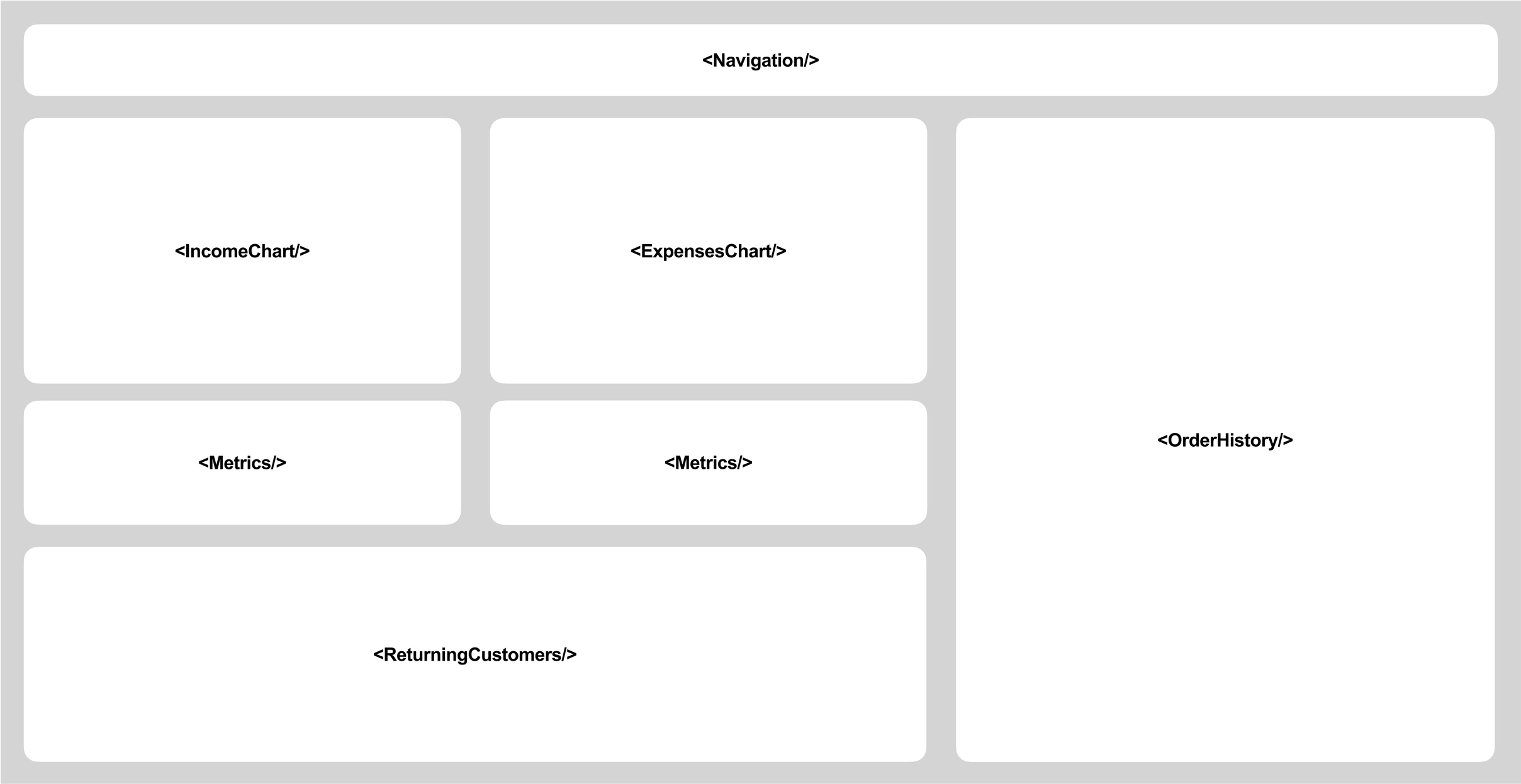
SDUI

WHAT IS SDUI

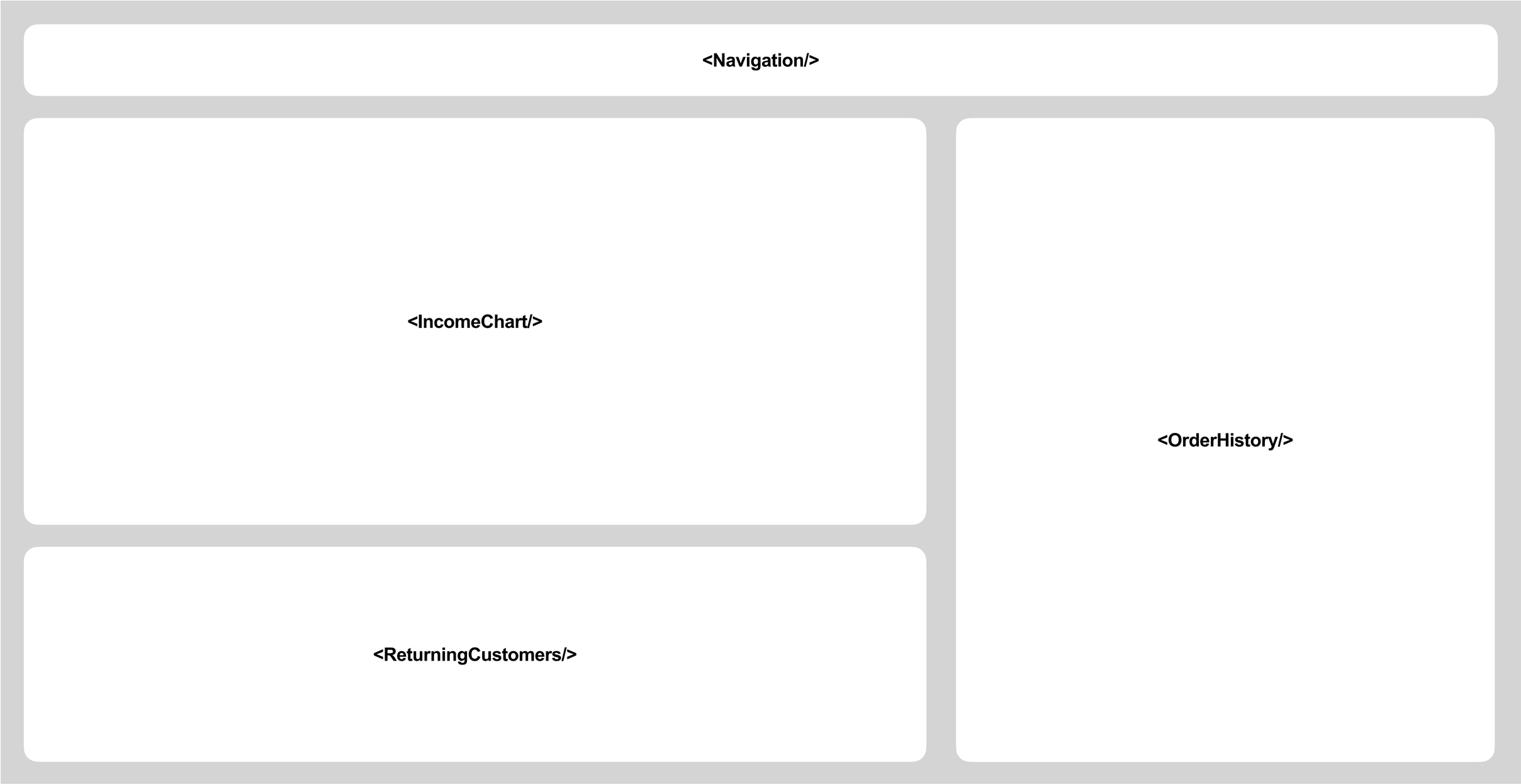
WHAT IS SDUI


<Navigation/>
<IncomeChart/>
<Metrics/>
<Metrics/>
<Metrics/>
<OrderHistory/>
<ReturningCustomers/>

WHAT IS SDUI


<Navigation/>
<IncomeChart/>
<ExpensesChart/>
<OrderHistory/>
<Metrics/>
<Metrics/>
<ReturningCustomers/>

WHAT IS SDUI


<Navigation/>
<IncomeChart/>
<OrderHistory/>
<ReturningCustomers/>

WHAT IS SDUI


<Navigation/>
<IncomeChart/>
<Metrics/>
<Metrics/>
<Metrics/>
<OrderHistory/>
<ReturningCustomers/>

WHAT IS SDUI

```
{
  "id": "total-sales-metric",
  "type": "metrics-card",
  "title": "Total Sales",
  "position": {
    "width": 3,
    "height": 2
  },
  "config": {
    "metrics": [
      {
        "label": "Total Sales",
        "value": 10892,
        "format": "number"
      }
    ]
  }
}
```

WHAT IS SDUI

```
{
  "id": "total-sales-metric",
  "type": "metrics-card",
  "title": "Total Sales",
  "position": {
    "width": 3,
    "height": 2
  },
  "config": {
    "metrics": [
      {
        "label": "Total Sales",
        "value": 10892,
        "format": "number"
      }
    ]
  }
}
```

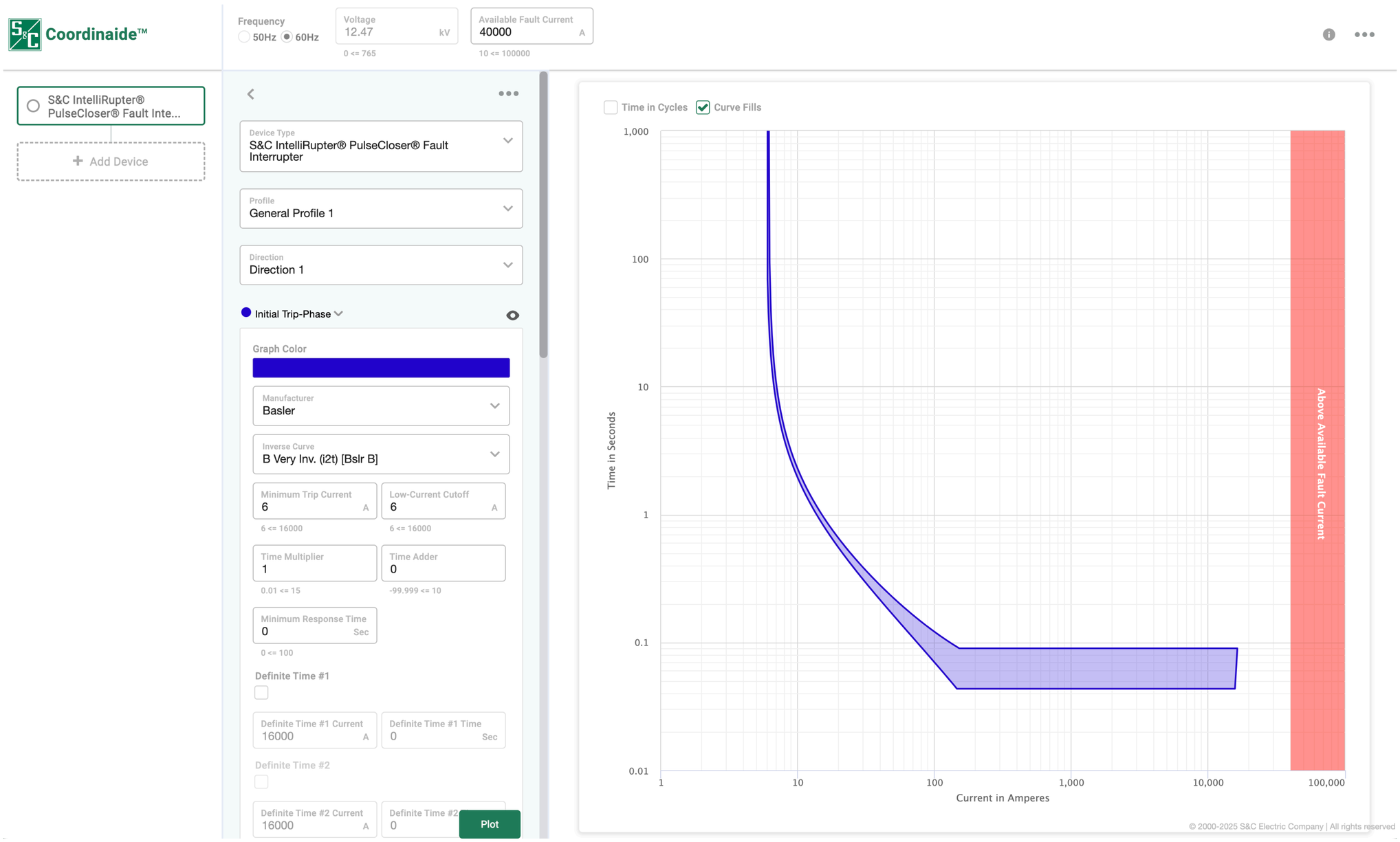
WHAT IS SDUI

<Metrics/>

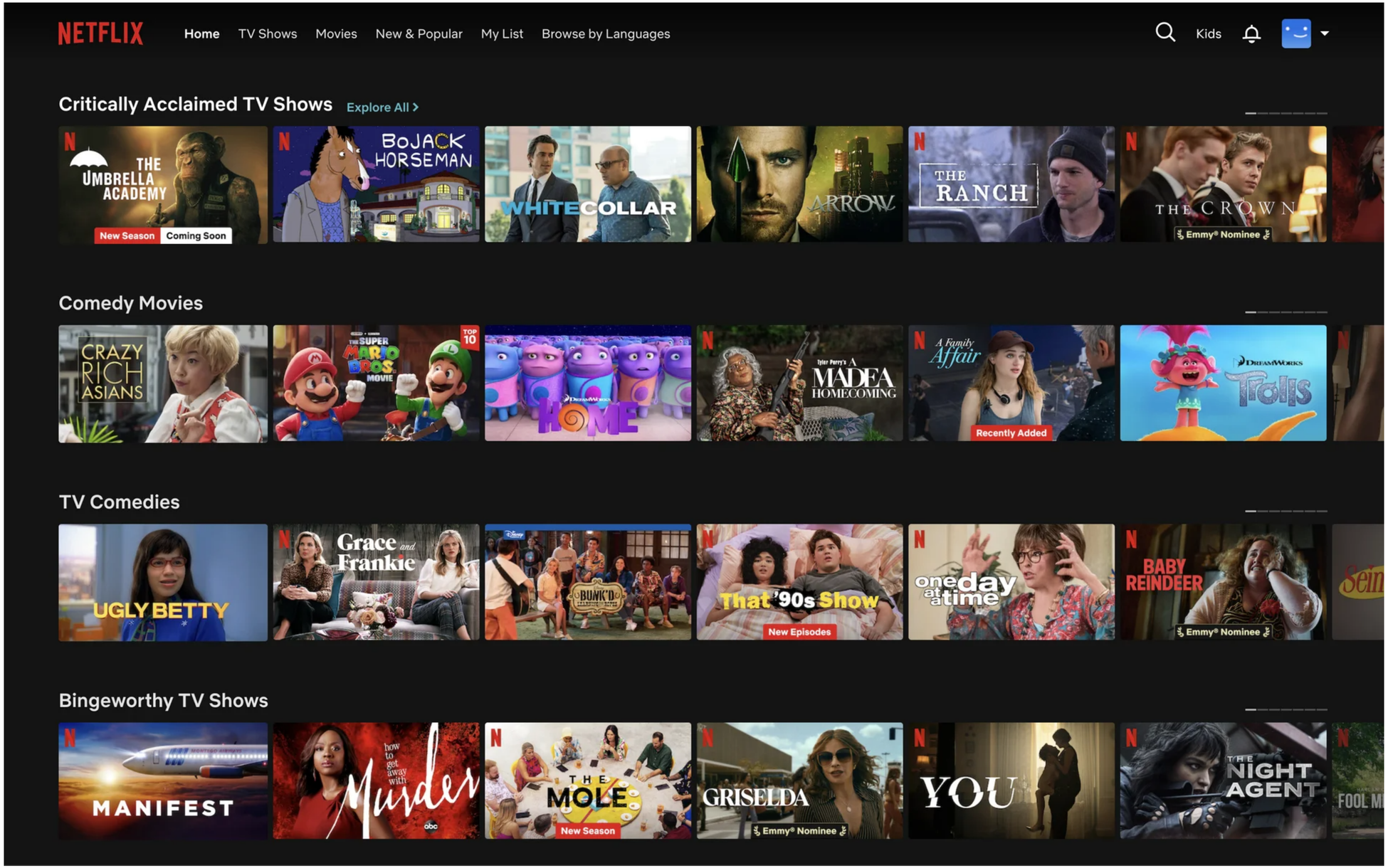
WHAT IS SDUI


<Navigation/>
<IncomeChart/>
<Metrics/>
<Metrics/>
<Metrics/>
<OrderHistory/>
<ReturningCustomers/>

WHAT IS SDUI - EXAMPLES



WHAT IS SDUI - EXAMPLES



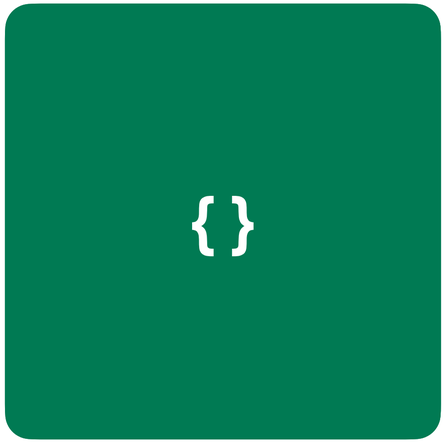
WHAT IS SDUI - EXAMPLES

S&C ELECTRIC COMPANY

SDUI TAKEAWAYS

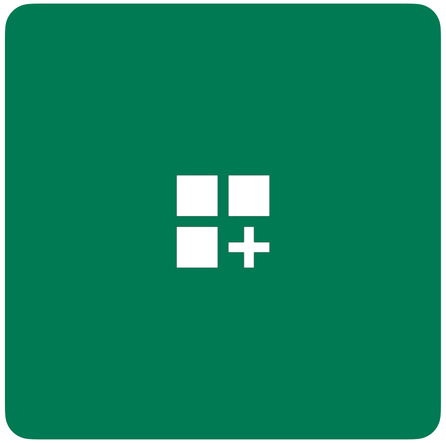
SDUI TAKEAWAYS

- SDUI allows for the configuration of an application’s UI through a dynamic schema



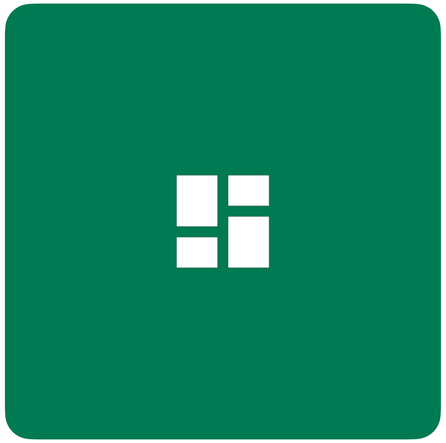
SDUI TAKEAWAYS

- SDUI allows for the configuration of an application’s UI through a dynamic schema
- Enables the creation of personalized/dynamic pages in an application without requiring additional UI work



SDUI TAKEAWAYS

- SDUI allows for the configuration of an application’s UI through a dynamic schema
- Enables the creation of personalized/dynamic pages in an application without requiring additional UI work
- Can be leveraged for entire applications, pages, or individual features



What is SDUI

How can React 19 help

SDUI + React 19 in practice

HOW CAN REACT 19 HELP

- Server Components
- Server Functions
- useActionState()
- useFormStatus()
- use()
- useOptimistic()

HOW CAN REACT 19 HELP

- Server Components
- Server Functions
- useActionState()
- useFormStatus()
- use()
- useOptimistic()

HOW CAN REACT 19 HELP

useActionState()

- Allows for the updating of state based on the result of an action/function
- Returns state, an action, and a pending state
- Allows for the execution of async code and auto updates pending state and state accordingly
- Can be utilized to retrieve new data for an SDUI

HOW CAN REACT 19 HELP

Server Components

- Render initial/entire state for a SDUI on the server
- Prevent an unnecessary network request, by parsing and generating the UI from a data source before sending to the client
- Cache initial state for an SDUI
- Improve first contentful paint (FCP) for SDUIs

HOW CAN REACT 19 HELP

- Utilized for retrieval of data for SDUI in both server and client components
- Can render and return server components in both server and client components
- Can cache responses for SDUI (cache server component responses for faster generation and rendering of SDUI)

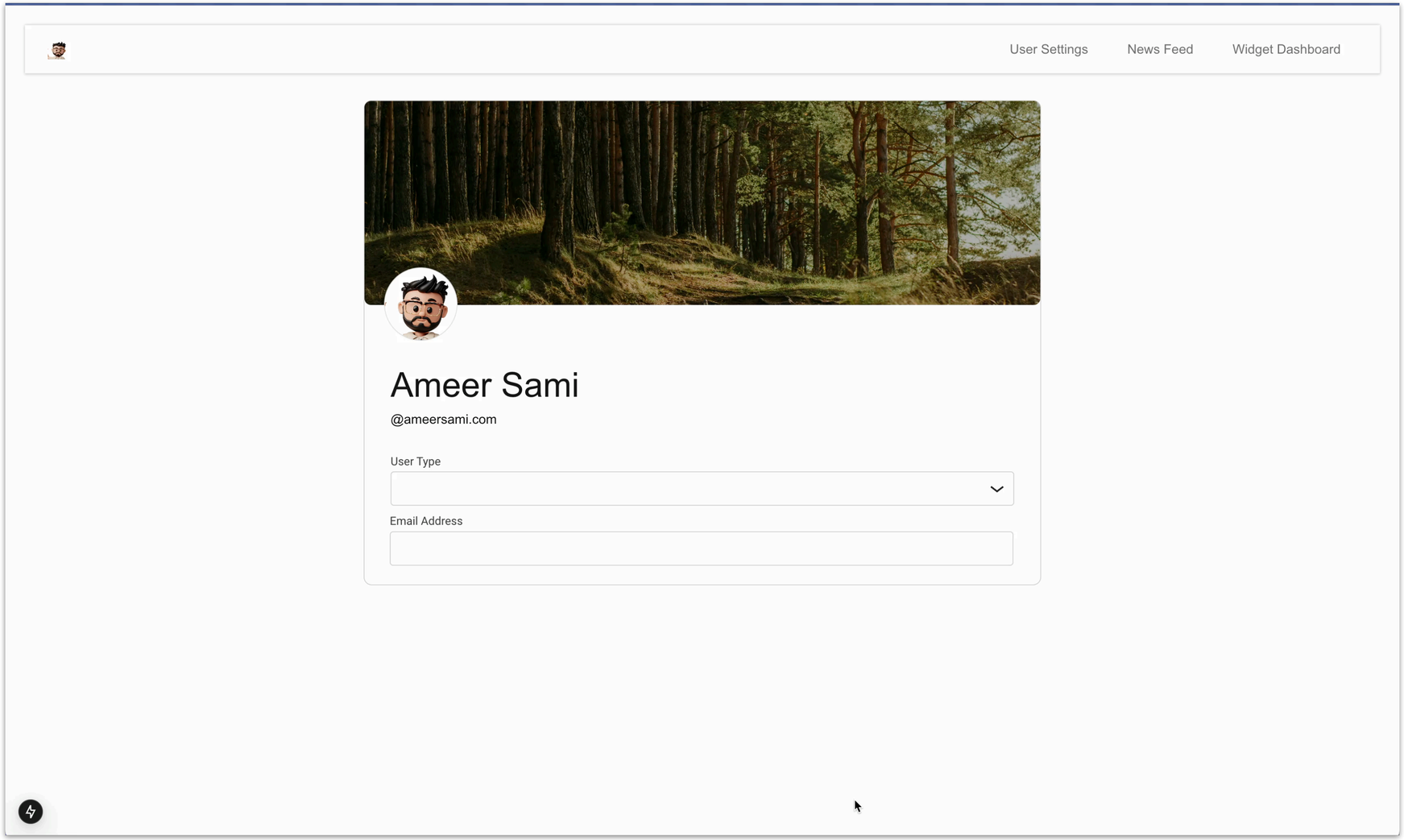
Server Functions

What is SDUI

How can React 19 help

SDUI + React 19 in practice

USER SETTINGS



USER SETTINGS - SDUI JSON

```json
{
  "type": "form",
  "title": "User Configuration",
  "components": [
    {
      "type": "select",
      "id": "userType",
      "label": "User Type",
      "required": true,
      "options": [
        {"label": "Individual", "value": "individual"},
        {"label": "Business", "value": "business"},
        {"label": "Developer", "value": "developer"}
      ],
      "defaultValue": "individual"
    },
    {
      "type": "textInput",
      "id": "email",
      "label": "Email Address",
      "required": true,
      "validations": {
        "email": true
      }
    },
    {
      "type": "conditional",
      "dependsOn": {
        "field": "userType",
        "value": "individual"
```

USER SETTINGS - SDUI JSON

- Define the component type

```
{
  "type": "select",
  "id": "userType",
  "label": "User Type",
  "required": true,
  "options": [
    {"label": "Individual", "value": "individual"},
    {"label": "Business", "value": "business"},
    {"label": "Developer", "value": "developer"}
  ],
  "defaultValue": "individual"
},
```

USER SETTINGS - SDUI JSON

- Define the component type
- Define props to configure the component

```
{
  "type": "select",
  "id": "userType",
  "label": "User Type",
  "required": true,
  "options": [
    {"label": "Individual", "value": "individual"},
    {"label": "Business", "value": "business"},
    {"label": "Developer", "value": "developer"}
  ],
  "defaultValue": "individual"
},
```

USER SETTINGS - SDUI JSON

- Define the component type
- Define props to configure the component
- Configure the default value

```
{
  "type": "select",
  "id": "userType",
  "label": "User Type",
  "required": true,
  "options": [
    {"label": "Individual", "value": "individual"},
    {"label": "Business", "value": "business"},
    {"label": "Developer", "value": "developer"}
  ],
  "defaultValue": "individual"
},
```

USER SETTINGS - ROOT

```
const UserSettingsPage = async () => {

  const initialComponentState = await getUserSettingsComponents([]);

  return (
    <div className={styles.container}>
      <UserSettingsForm>
        {initialComponentState.map(component => <UserSettingsComponentFactory {...component} />)}
      </UserSettingsForm>
    </div>
  );
}
```

USER SETTINGS - ROOT

- Invoking our server function to retrieve initial state

```
const UserSettingsPage = async () => {

  const initialComponentState = await getUserSettingsComponents([]);

  return (
    <div className={styles.container}>
      <UserSettingsForm>
        {initialComponentState.map(component => <UserSettingsComponentFactory {...component} />)}
      </UserSettingsForm>
    </div>
  );
}
```

USER SETTINGS - ROOT

- Invoking our server function to retrieve initial state
- Using a client component to render our form

```
const UserSettingsPage = async () => {

  const initialComponentState = await getUserSettingsComponents([]);

  return (
    <div className={styles.container}>
      <UserSettingsForm>
        {initialComponentState.map(component => <UserSettingsComponentFactory {...component} />)}
      </UserSettingsForm>
    </div>
  );
}
```

USER SETTINGS - ROOT

- Invoking our server function to retrieve initial state
- Using a client component to render our form
- Server rendering our initial form components

```
const UserSettingsPage = async () => {

  const initialComponentState = await getUserSettingsComponents([]);

  return (
    <div className={styles.container}>
      <UserSettingsForm>
        {initialComponentState.map(component => <UserSettingsComponentFactory {...component} />)}
      </UserSettingsForm>
    </div>
  );
}
```

USER SETTINGS - FORM COMPONENT USEACTIONSTATE()

```
const [state, submitAction, isPending] = useActionState<Array<FormComponent>, ChangeEvent<HTMLFormElement>>(
  async (previousState, newComponentData) => {

    const newData = new FormData(newComponentData.target);
    const convertedComponentData = Array.from(newData.entries()).map(([id, value]) => ({
      id,
      selectedValue: value
    })) as Array<FormComponent>;

    const newComps = await getUserSettingsComponents(convertedComponentData);

    return newComps;
  },
  []
);
```

USER SETTINGS - FORM COMPONENT USEACTIONSTATE()

- Defining state type and action data param type

```
const [state, submitAction, isPending] = useActionState<Array<FormComponent>, ChangeEvent<HTMLFormElement>>(
  async (previousState, newComponentData) => {

    const newData = new FormData(newComponentData.target);
    const convertedComponentData = Array.from(newData.entries()).map(([id, value]) => ({
      id,
      selectedValue: value
    })) as Array<FormComponent>;

    const newComps = await getUserSettingsComponents(convertedComponentData);

    return newComps;
  },
  []
);
```

USER SETTINGS - FORM COMPONENT USEACTIONSTATE()

- Defining state type and action data param type
- Invoking our server function to retrieve the next set of components

```
const [state, submitAction, isPending] = useActionState<Array<FormComponent>, ChangeEvent<HTMLFormElement>>(
  async (previousState, newComponentData) => {

    const newData = new FormData(newComponentData.target);
    const convertedComponentData = Array.from(newData.entries()).map(([id, value]) => ({
      id,
      selectedValue: value
    })) as Array<FormComponent>;

    const newComps = await getUserSettingsComponents(convertedComponentData);

    return newComps;
  },
  []
);
```

USER SETTINGS - FORM

```
<form
  className={styles.form}
  onSubmit={handleSubmit}
>
  {props.children}
  {state.map(component => <UserSettingsComponentFactory key={`fetched-component-${component.id}`} {...component} />
</form>
```

USER SETTINGS - FORM

- Render initial server components

```
<form
  className={styles.form}
  onSubmit={handleSubmit}
>
  {props.children}
  {state.map(component => <UserSettingsComponentFactory key={`fetched-component-${component.id}`} {...component} />
</form>
```

USER SETTINGS - FORM

- Render initial server components
- Client side render new components

```
<form
  className={styles.form}
  onSubmit={handleSubmit}
>
  {props.children}
  {state.map(component => <UserSettingsComponentFactory key={`fetched-component-${component.id}`} {...component} />
</form>
```

USER SETTINGS - FORM

- Render initial server components
- Client side render new components
- Pass a function to the form onSubmit

```
<form
  className={styles.form}
  onSubmit={handleSubmit}
>
  {props.children}
  {state.map(component => <UserSettingsComponentFactory key={`fetched-component-${component.id}`} {...component} />
</form>
```

USER SETTINGS - FORM SUBMISSION FUNCTION

```
const handleSubmit = (event: ChangeEvent<HTMLFormElement>) => {
  event.preventDefault();
  startTransition(() => submitAction(event));
}
```

USER SETTINGS - FORM SUBMISSION FUNCTION

- Preventing default page refresh with form submission

```
const handleSubmit = (event: ChangeEvent<HTMLFormElement>) => {
  event.preventDefault();
  startTransition(() => submitAction(event));
}
```

USER SETTINGS - FORM SUBMISSION FUNCTION

- Preventing default page refresh with form submission
- Invoking useActionState action (must be done within a startTransition)

```
const handleSubmit = (event: ChangeEvent<HTMLFormElement>) => {
  event.preventDefault();
  startTransition(() => submitAction(event));
}
```

What can be improved?

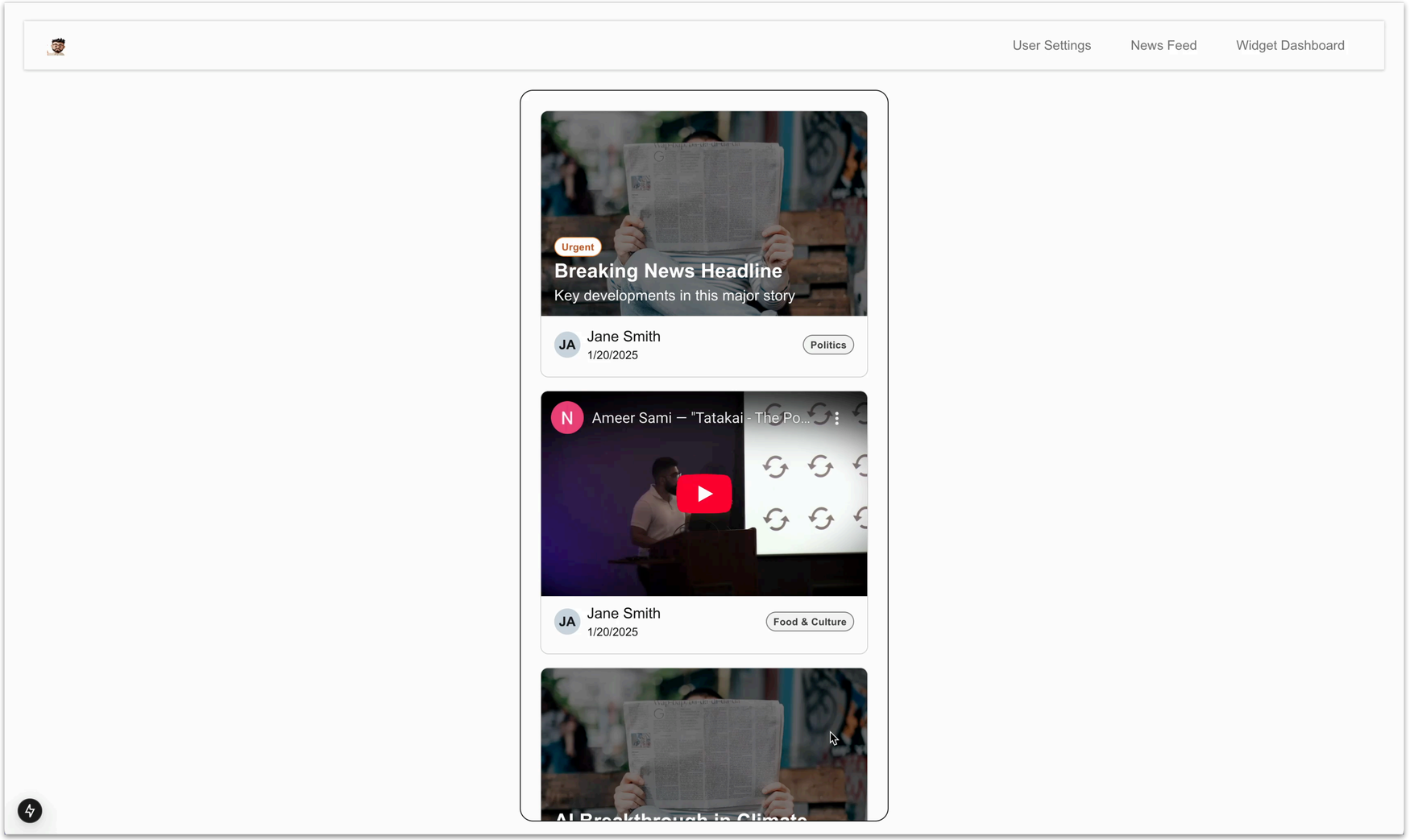
USER SETTINGS

```
<form
  className={styles.form}
  onSubmit={handleSubmit}
>
  {props.children}
  {state.map(component => <UserSettingsComponentFactory key={`fetched-component-${component.id}`} {...component} />
</form>
```

USER SETTINGS

```
<form
  className={styles.form}
  onSubmit={handleSubmit}
>
  {props.children}
  {state.map(component => <UserSettingsComponentFactory key={`fetched-component-${component.id}`} {...component} />
</form>
```

NEWS FEED


User Settings
News Feed
Widget Dashboard
Urgent
Breaking News Headline
Key developments in this major story
JA
Jane Smith
1/20/2025
Politics
N
Ameer Sami — "Tatakai - The Po...
JA
Jane Smith
1/20/2025
Food & Culture

NEWS FEED - ROOT

```
const NewsFeed = async () => {

  const Posts = await getPosts(0);

  return (
    <div className={styles.container}>
      <div className={styles.phoneContainer}>
        <NewsFeedLoadMore>
          {Posts}
        </NewsFeedLoadMore>
      </div>
    </div>
  );
};
```

NEWS FEED - ROOT

- Invoking our server function to retrieve initial state

```
const NewsFeed = async () => {

  const Posts = await getPosts(0);

  return (
    <div className={styles.container}>
      <div className={styles.phoneContainer}>
        <NewsFeedLoadMore>
          {Posts}
        </NewsFeedLoadMore>
      </div>
    </div>
  );
};
```

NEWS FEED - ROOT

- Invoking our server function to retrieve initial state
- Passing our initial posts as children

```
const NewsFeed = async () => {

  const Posts = await getPosts(0);

  return (
    <div className={styles.container}>
      <div className={styles.phoneContainer}>
        <NewsFeedLoadMore>
          {Posts}
        </NewsFeedLoadMore>
      </div>
    </div>
  );
};
```

NEWS FEED - CLIENT COMPONENT USEACTIONSTATE()

```
const [state, submitAction, isPending] = useActionState<Array<any>, number>(
  async (previousState, newOffset) => {

    const newComponents = await getPosts(newOffset);

    if (!newComponents) {
      return previousState;
    }

    return [
      ...previousState,
      newComponents
    ];
  },
  [],
);
```

NEWS FEED - CLIENT COMPONENT USEACTIONSTATE()

- Invoking server function to retrieve the next set of **server** components

```
const [state, submitAction, isPending] = useActionState<Array<any>, number>(
  async (previousState, newOffset) => {

    const newComponents = await getPosts(newOffset);

    if (!newComponents) {
      return previousState;
    }

    return [
      ...previousState,
      newComponents
    ];
  },
  [],
);
```

NEWS FEED - CLIENT COMPONENT USEACTIONSTATE()

- Invoking server function to retrieve the next set of **server** components
- Appending new components to the posts state

```
const [state, submitAction, isPending] = useActionState<Array<any>, number>(
  async (previousState, newOffset) => {

    const newComponents = await getPosts(newOffset);

    if (!newComponents) {
      return previousState;
    }

    return [
      ...previousState,
      newComponents
    ];
  },
  [],
);
```

NEWS FEED - CLIENT SCROLLING COMPONENT

```
return (
  <div
    ref={scrollContainerRef}
    className={styles.scrollContainer}
  >
    {children}
    {state}
  </div>
);
```

NEWS FEED VS USER SETTINGS

```
return (
  <div
    ref={scrollContainerRef}
    className={styles.scrollContainer}
  >
    {children}
    {state}
  </div>
);
```

```
<form
  className={styles.form}
  onSubmit={handleSubmit}
>
  {props.children}
  {state.map(component => <UserSettingsComponentFactory key={`fetched-component-${component.id}`} {...component} />
</form>
```

NEWS FEED VS USER SETTINGS

```
return (
  <div
    ref={scrollContainerRef}
    className={styles.scrollContainer}
  >
    {children}
    {state}
  </div>
);
```

```
<form
  className={styles.form}
  onSubmit={handleSubmit}
>
  {props.children}
  {state.map(component => <UserSettingsComponentFactory key={`fetched-component-${component.id}`} {...component} />
</form>
```

NEWS FEED - SERVER FUNCTION

```
'use server';

export default async (offset: number) => {
  const slicedPosts = posts.slice(offset, offset + 5);

  if (!slicedPosts.length) {
    return null;
  }

  return slicedPosts.map(post => (
    <NewsFeedFactory key={`news-feed-post-item-${post.id}-${post.author}`} post={post as NewsPost} />
  ));
};
```

NEWS FEED - SERVER FUNCTION

- Instead of returning JSON we return the server component(s)

```
'use server';

export default async (offset: number) => {
  const slicedPosts = posts.slice(offset, offset + 5);

  if (!slicedPosts.length) {
    return null;
  }

  return slicedPosts.map(post => (
    <NewsFeedFactory key={`news-feed-post-item-${post.id}-${post.author}`} post={post as NewsPost} />
  ));
};
```

NEWS FEED - SERVER FUNCTION

- Instead of returning JSON we return the server component(s)
- Leverage caching to improve the user experience

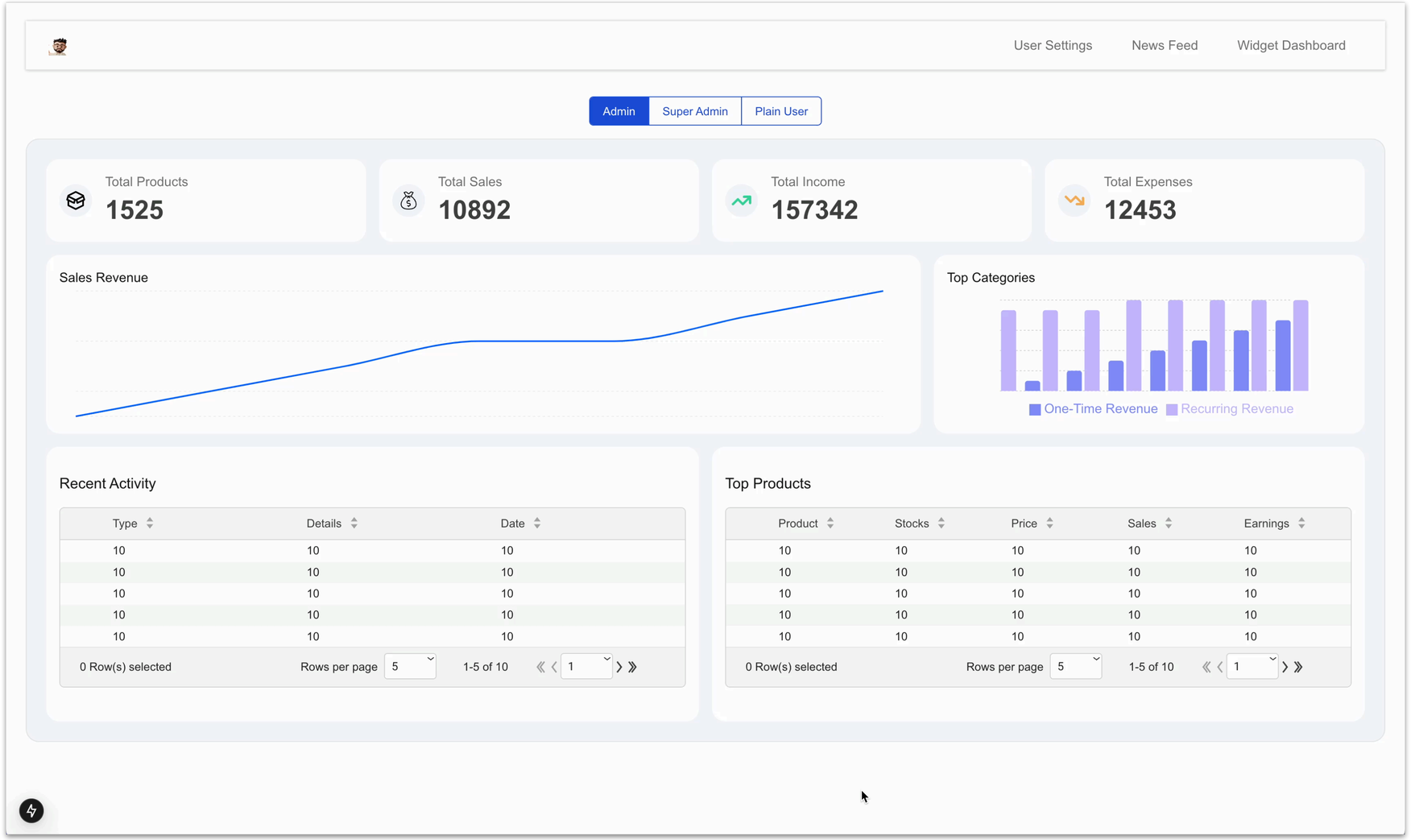
```
'use server';

export default async (offset: number) => {
  const slicedPosts = posts.slice(offset, offset + 5);

  if (!slicedPosts.length) {
    return null;
  }

  return slicedPosts.map(post => (
    <NewsFeedFactory key={`news-feed-post-item-${post.id}-${post.author}`} post={post as NewsPost} />
  ));
};
```

DASHBOARD


User Settings
News Feed
Widget Dashboard
Admin
Super Admin
Plain User
Total Products
1525
Total Sales
10892
Total Income
157342
Total Expenses
12453
Sales Revenue
Top Categories
One-Time Revenue
Recurring Revenue
Recent Activity
Type
Details
Date
10
0 Row(s) selected
Rows per page
5
1-5 of 10
1
Top Products
Product
Stocks
Price
Sales
Earnings
10
0 Row(s) selected
Rows per page
5
1-5 of 10
1

REACT 19 + SDUI TAKEAWAYS

useActionState()

Server Components

- Utilize server components to improve first contentful paint by rendering initial SDUI
- Utilize server functions to return server components and cache the responses on the server action level to improve performance
- Leverage useActionState to more easily manage pending states, application state, and invoke actions for dynamic SDUI driven forms

Server Functions

REACT 19 + SDUI TAKEAWAYS

- Server Components
- Server Functions
- useActionState()
- useFormStatus()
- use()
- useOptimistic()

Thank you!

@ameersami.com | ameersami.com