ventionteams.com

React.js moduldar project architecture

vention

Built to scale

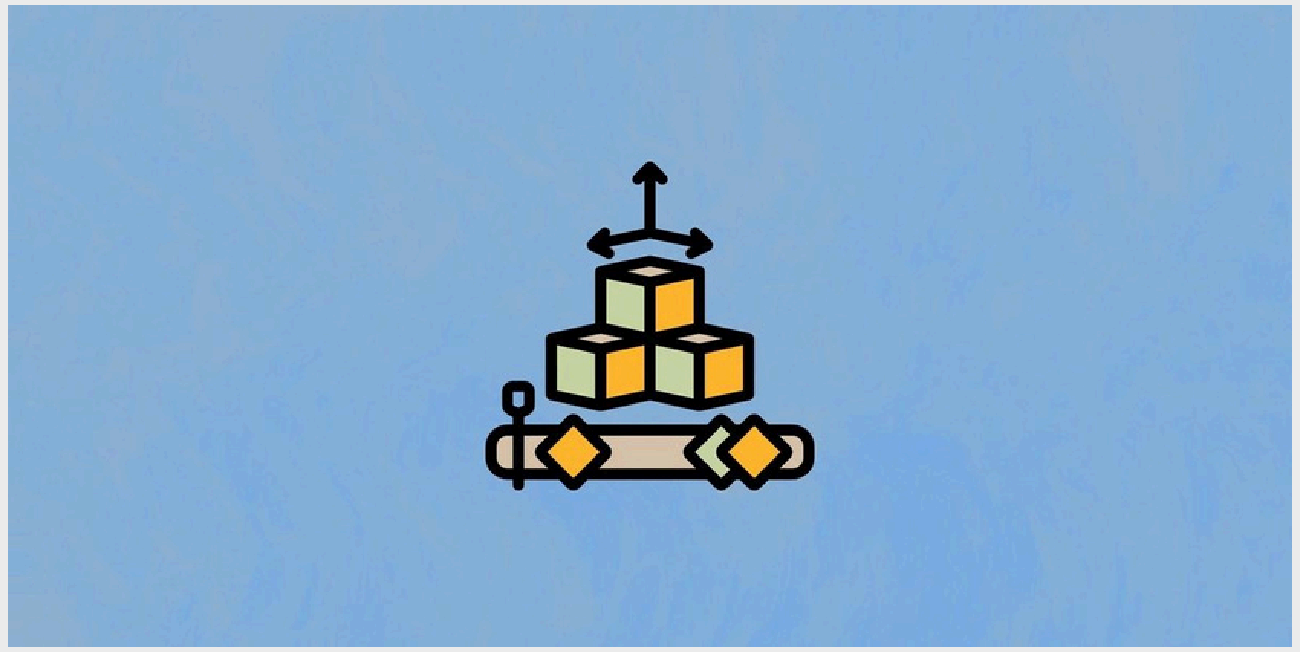
Feruza Makhmudova

Software Engineer (4+ years prof. experience)

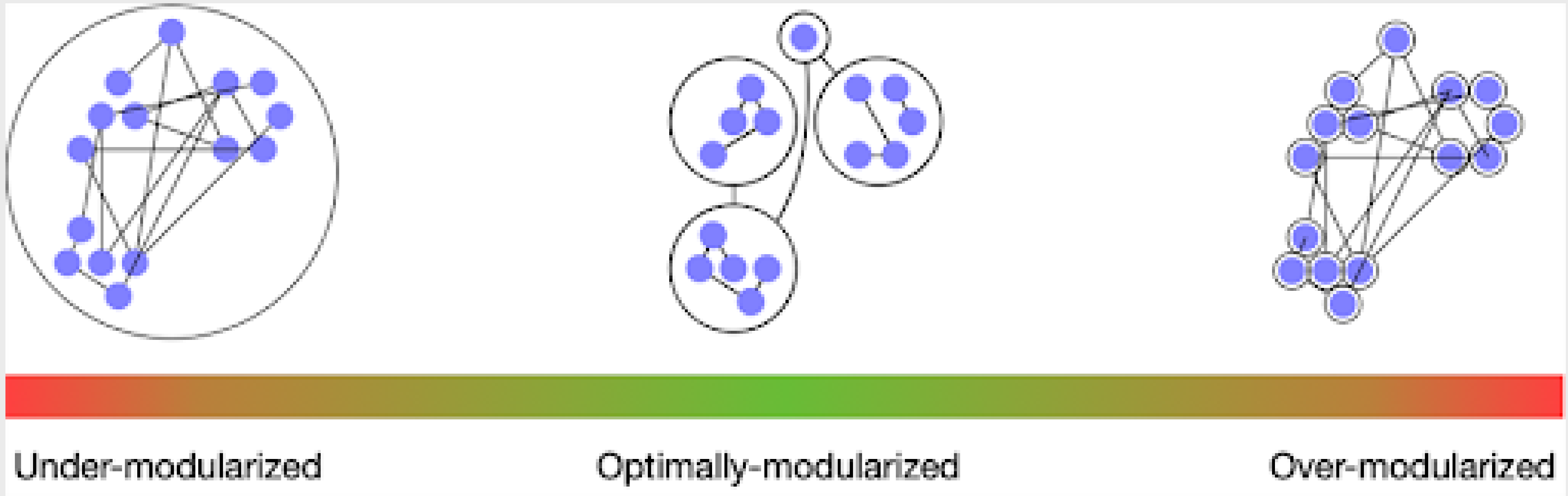
- Mcs in Computer science and software engineering
- **email:** feruza.maxmudova@ventionteams.com

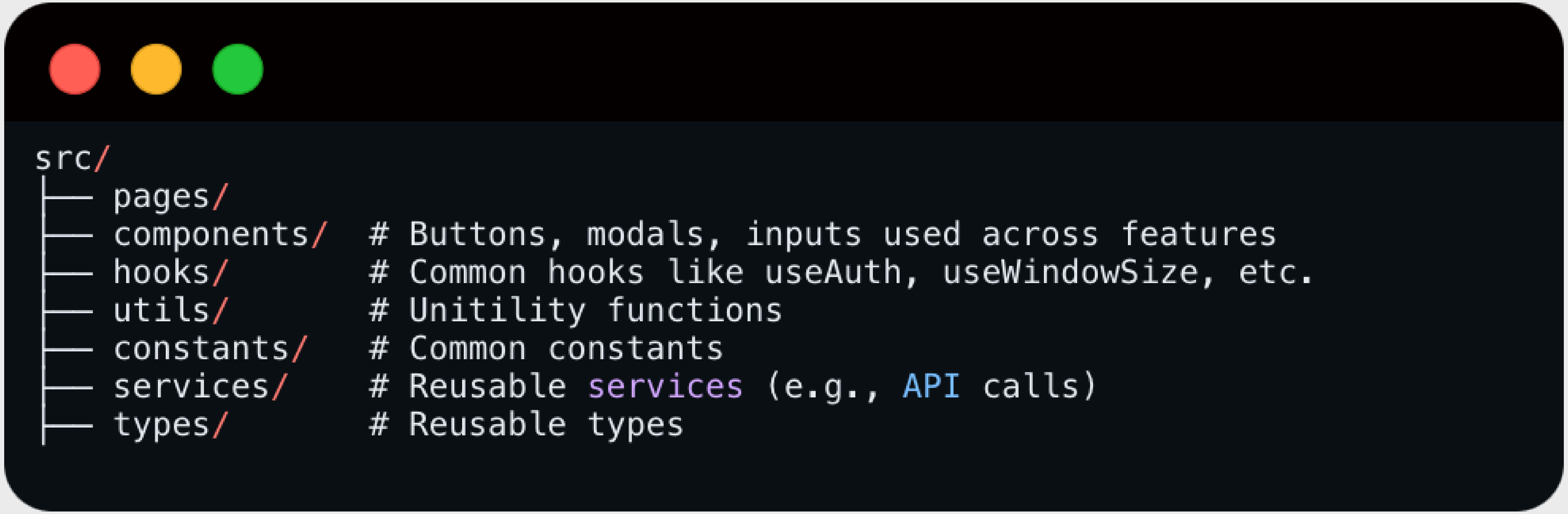
Agenda:

- What is modular architecture
- Why to use modular architecture
- Project structure
- Example Project

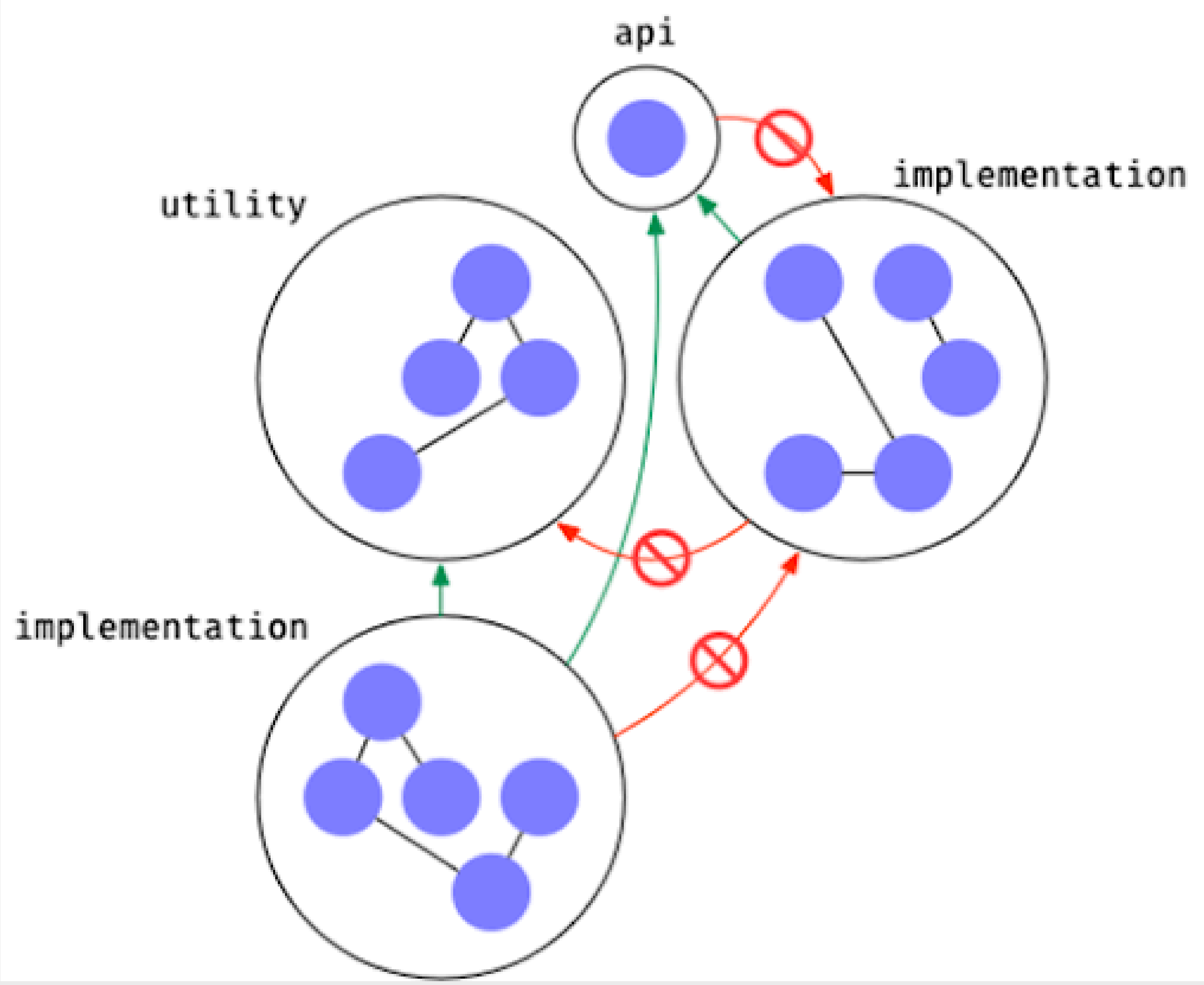


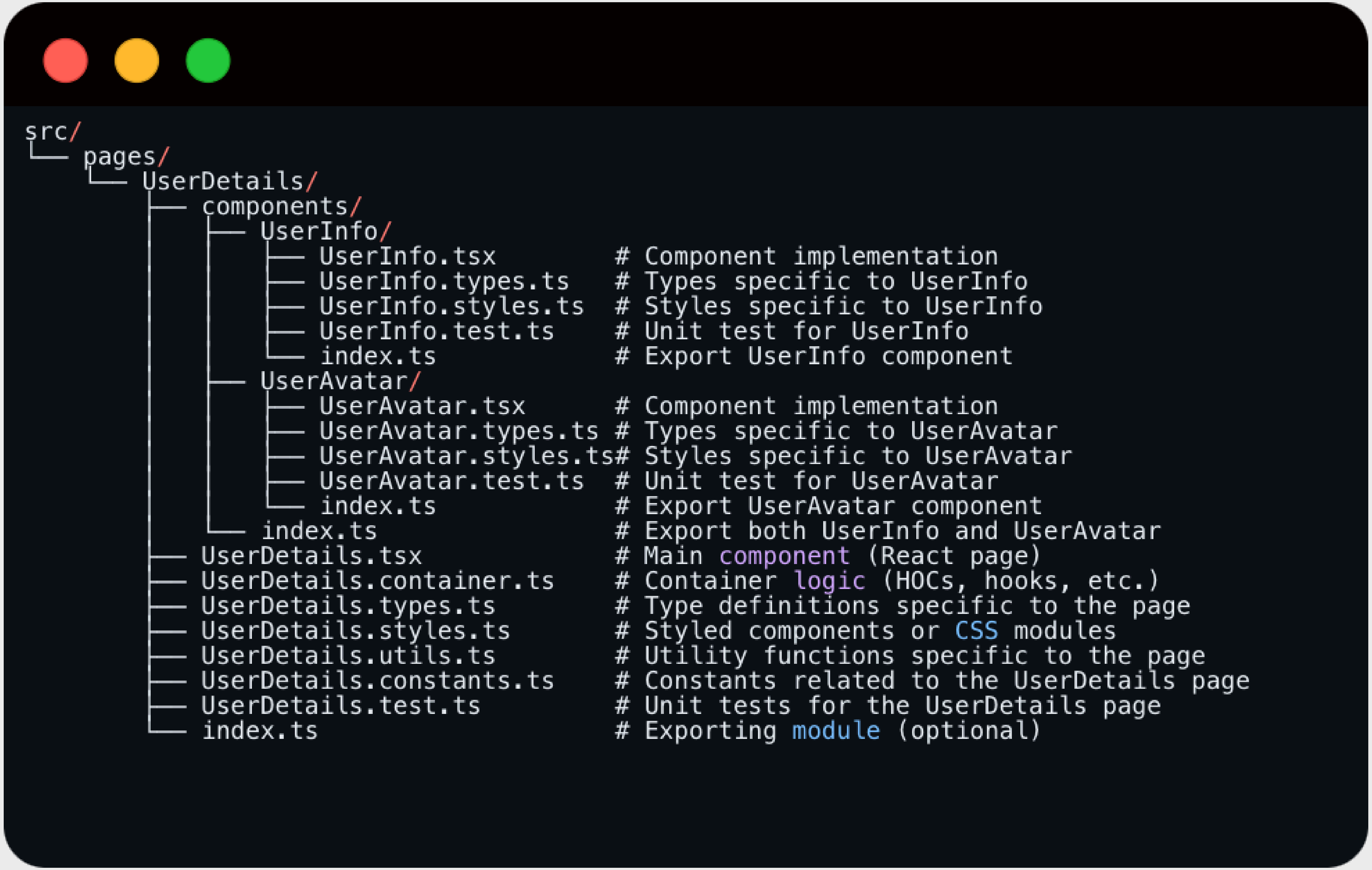
- Modular React project architecture - an approach where the codebase is structured into self-contained, reusable modules, each responsible for a specific feature or domain of the application.
- This design helps in scaling the project by promoting separation of concerns, maintainability, and reusability.


Under-modularized
Optimally-modularized
Over-modularized

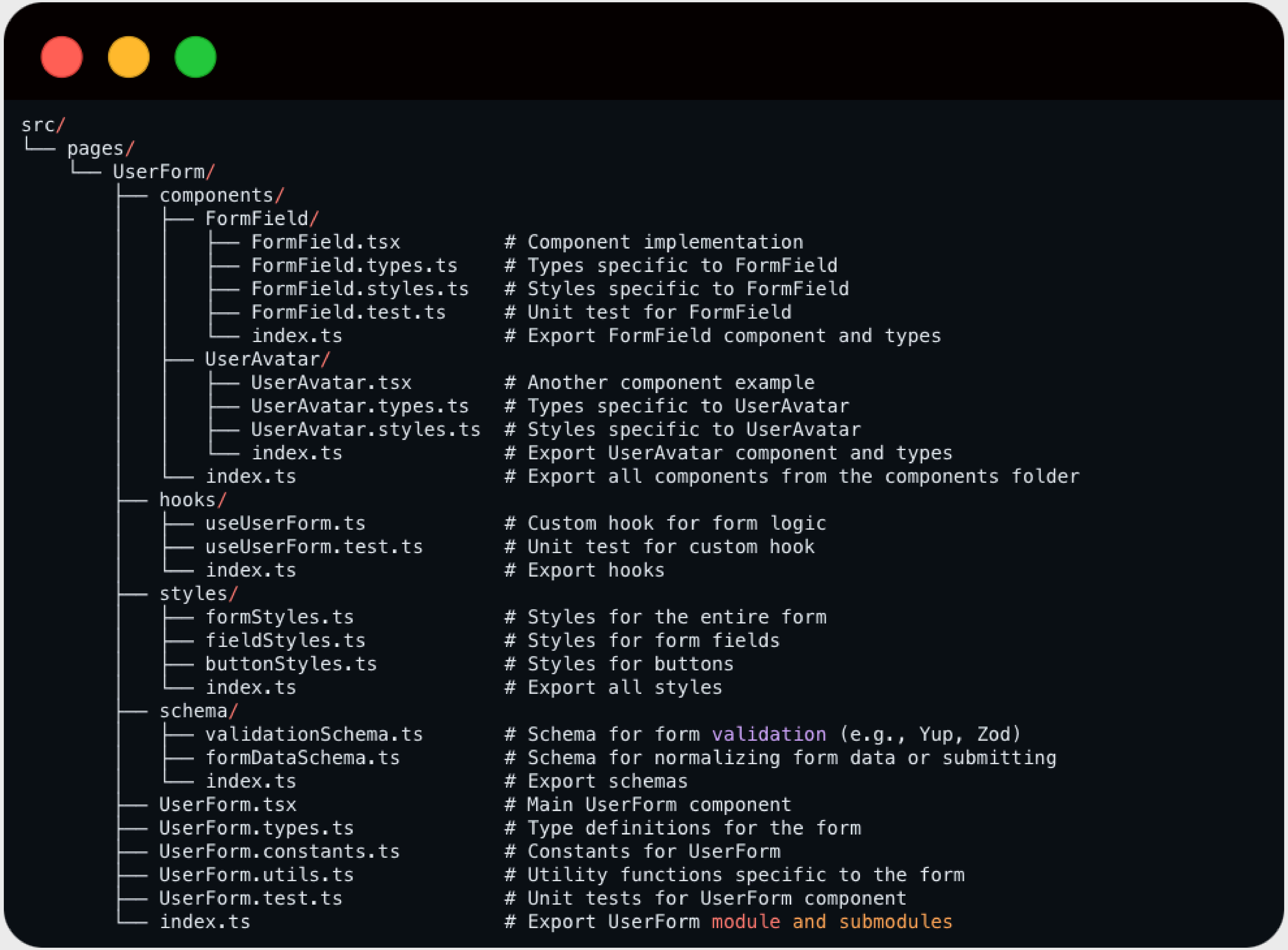


```
src/
├── pages/
├── components/  # Buttons, modals, inputs used across features
├── hooks/       # Common hooks like useAuth, useWindowSize, etc.
├── utils/       # Unitility functions
├── constants/   # Common constants
├── services/    # Reusable services (e.g., API calls)
├── types/       # Reusable types
```


api
utility
implementation
implementation



```
src/
└── pages/
    └── UserDetails/
        ├── components/
        │   ├── UserInfo/
        │   │   ├── UserInfo.tsx        # Component implementation
        │   │   ├── UserInfo.types.ts   # Types specific to UserInfo
        │   │   ├── UserInfo.styles.ts  # Styles specific to UserInfo
        │   │   ├── UserInfo.test.ts    # Unit test for UserInfo
        │   │   └── index.ts            # Export UserInfo component
        │   ├── UserAvatar/
        │   │   ├── UserAvatar.tsx      # Component implementation
        │   │   ├── UserAvatar.types.ts # Types specific to UserAvatar
        │   │   ├── UserAvatar.styles.ts# Styles specific to UserAvatar
        │   │   ├── UserAvatar.test.ts  # Unit test for UserAvatar
        │   │   └── index.ts            # Export UserAvatar component
        │   └── index.ts                # Export both UserInfo and UserAvatar
        ├── UserDetails.tsx             # Main component (React page)
        ├── UserDetails.container.ts    # Container logic (HOCs, hooks, etc.)
        ├── UserDetails.types.ts        # Type definitions specific to the page
        ├── UserDetails.styles.ts       # Styled components or CSS modules
        ├── UserDetails.utils.ts        # Utility functions specific to the page
        ├── UserDetails.constants.ts    # Constants related to the UserDetails page
        ├── UserDetails.test.ts         # Unit tests for the UserDetails page
        └── index.ts                    # Exporting module (optional)
```



```
src/
└── pages/
    └── UserForm/
        ├── components/
        │   ├── FormField/
        │   │   ├── FormField.tsx         # Component implementation
        │   │   ├── FormField.types.ts    # Types specific to FormField
        │   │   ├── FormField.styles.ts   # Styles specific to FormField
        │   │   ├── FormField.test.ts     # Unit test for FormField
        │   │   └── index.ts              # Export FormField component and types
        │   ├── UserAvatar/
        │   │   ├── UserAvatar.tsx        # Another component example
        │   │   ├── UserAvatar.types.ts   # Types specific to UserAvatar
        │   │   ├── UserAvatar.styles.ts  # Styles specific to UserAvatar
        │   │   └── index.ts              # Export UserAvatar component and types
        │   └── index.ts                  # Export all components from the components folder
        ├── hooks/
        │   ├── useUserForm.ts            # Custom hook for form logic
        │   ├── useUserForm.test.ts       # Unit test for custom hook
        │   └── index.ts                  # Export hooks
        ├── styles/
        │   ├── formStyles.ts             # Styles for the entire form
        │   ├── fieldStyles.ts            # Styles for form fields
        │   ├── buttonStyles.ts           # Styles for buttons
        │   └── index.ts                  # Export all styles
        ├── schema/
        │   ├── validationSchema.ts       # Schema for form validation (e.g., Yup, Zod)
        │   ├── formDataSchema.ts         # Schema for normalizing form data or submitting
        │   └── index.ts                  # Export schemas
        ├── UserForm.tsx                  # Main UserForm component
        ├── UserForm.types.ts             # Type definitions for the form
        ├── UserForm.constants.ts         # Constants for UserForm
        ├── UserForm.utils.ts             # Utility functions specific to the form
        ├── UserForm.test.ts              # Unit tests for UserForm component
        └── index.ts                      # Export UserForm module and submodules
```

Any question?

Thank you very much!