Phronex

From Greek φρόνησις ('Phronesis')

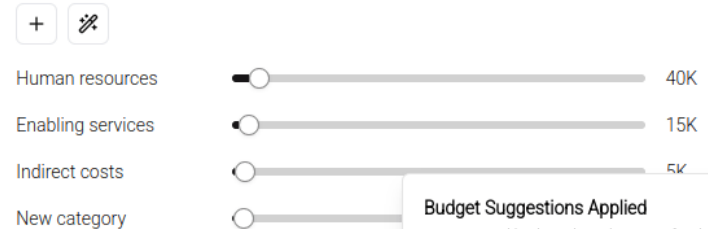*Practical wisdom in decision-making, planning, and budgeting*

Key features

- Project and portfolio management.
- **AI-powered** budget planning.
- Expense tracking and categorisation.
- Resource planning.
- Clean, responsive and accessible UI.
- Cloud ready.

Phronex highlights

Projects overview - with coloring adjust to budget adjustment.

User experience

The UI allows *free text search*, *date filtering* and *sorting* of projects, resources, and expenses across all of its screens.

AI features

Based in the declared scope and time duration, the application can propose an initial budget thanks to AI*.

Budget Distribution

Human resources	40K
Enabling services	15K
Indirect costs	5K
New category	

Budget Suggestions Applied
AI-generated budgets have been set for the project categories.
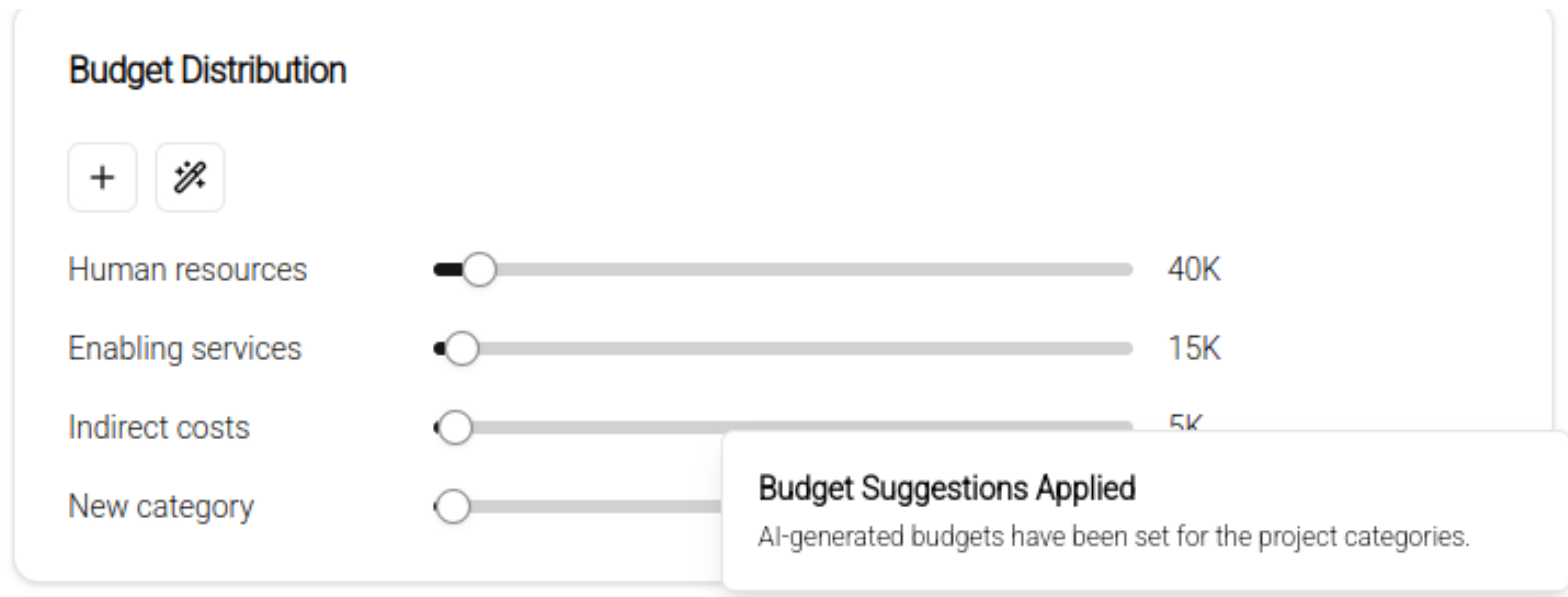
*Note this feature is in beta

Consistency

Categories

Filter...

Name ↑↓
Enabling services
Human resources
Indirect costs

Phronex allows a **consistent portfolio management** thanks to using a single set of **reusable cost categories** across the different projects.

Project Resources

Filter...

Person ↑↓
Aragorn Arathorn

Roles

Filter...

Description ↑↓
Business Analyst
Designer
Developer
Project Manager

Additionally it allows to manage a **single workforce*** that can be distributed across projects with generic but configurable roles.

* Limited features on v1.0

Accessibility

The UX/UI considers a **diverse set of users** with different needs. All reusable components adhere to **WAI-ARIA design principles** thanks to **ShadCN/UI** reusable components.

It also has a **high-contrast dark** mode to protect user vision.
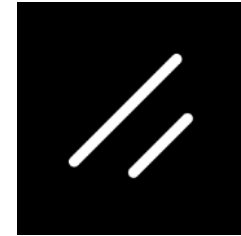
ShadCN/UI
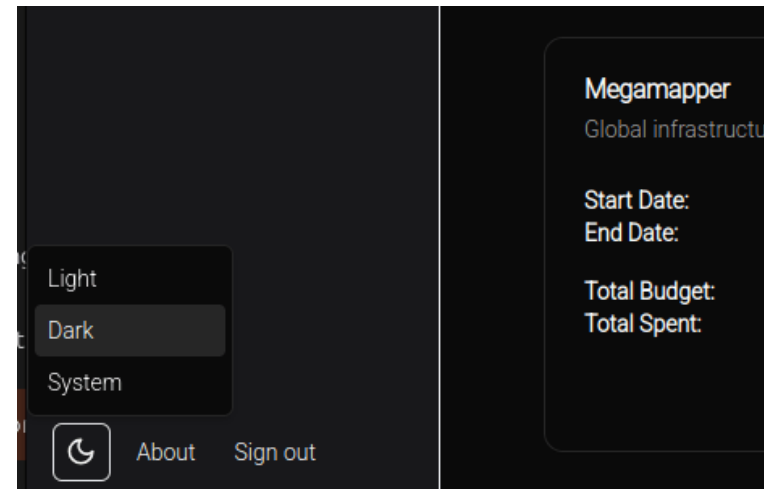
Phronex Architecture
Technical features

Technology stack

The application has considered a varied but consistent stack, based on full-stack javascript / typescript.

Next.js 15

React 19

TailwindCSS

ShadCN UI

PostgreSQL

pnpm

Docker

AWS

Solution architecture - TEST

The product is deployed in AWS (using EC2).

It is running fully dockerized.

It also interacts with OpenAI as an external service via API keys.
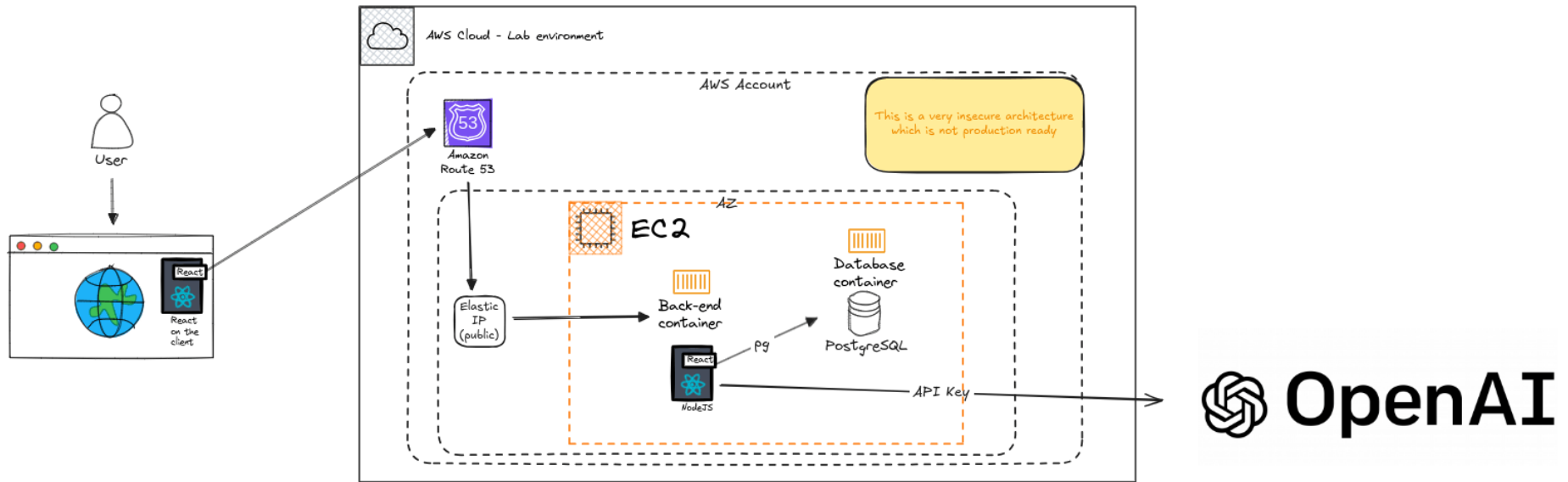
Solution architecture - PROD

Ideally this application should serverless. A solution based on OpenNext (for pushing server actions into AWS ambda has been looked into).
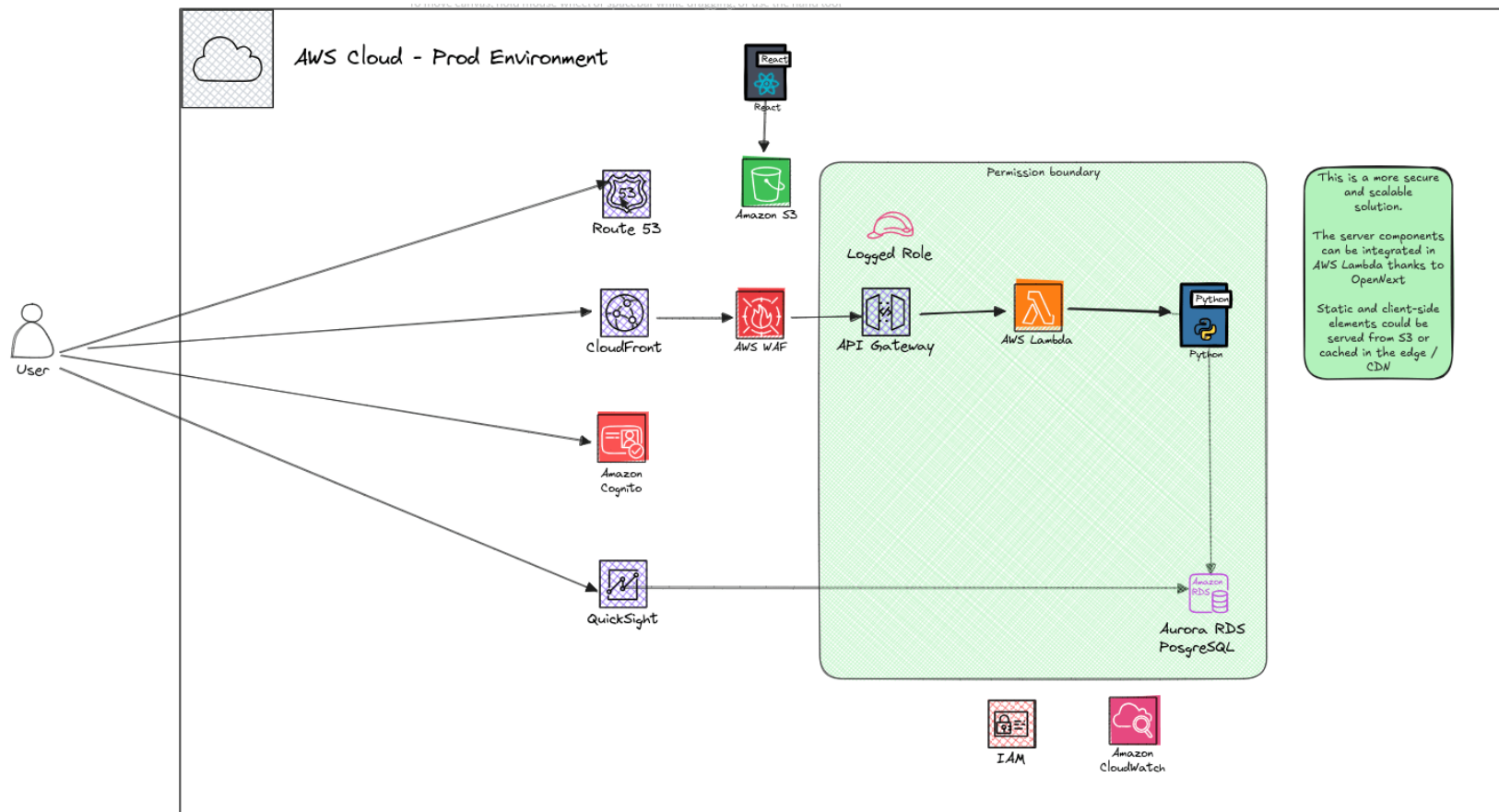
Navigation

Taking advantage of NextJS 15 as a well structured and opinionanted React framework, navigation has been designed to reuse layout elements across the screens.

Data model

Relational data model

Users

id
user_email
encpassword

Users != "Person" (human resources)

Project

project_id
project_name
project_scope
project_creation_date
project_start_date
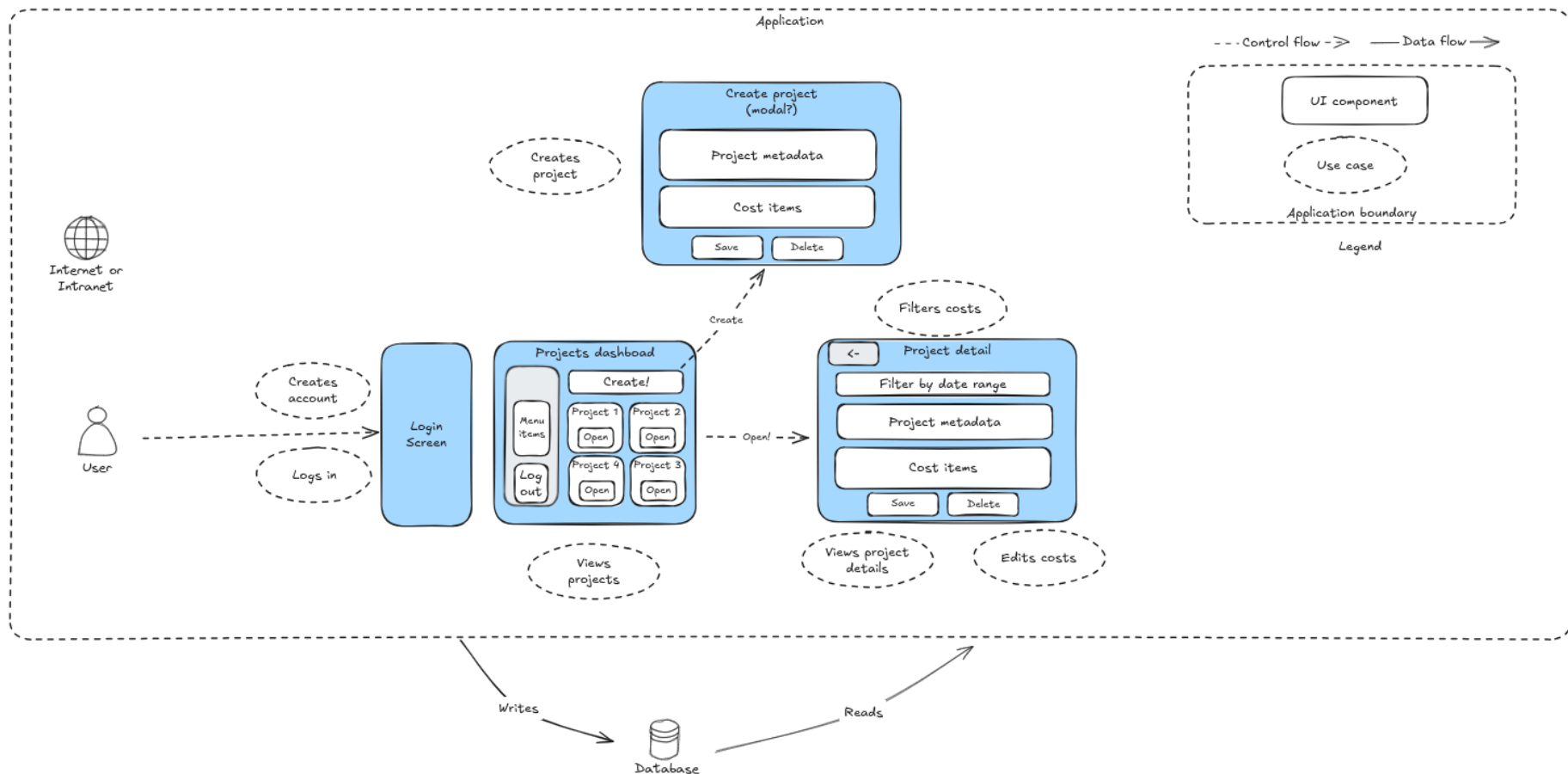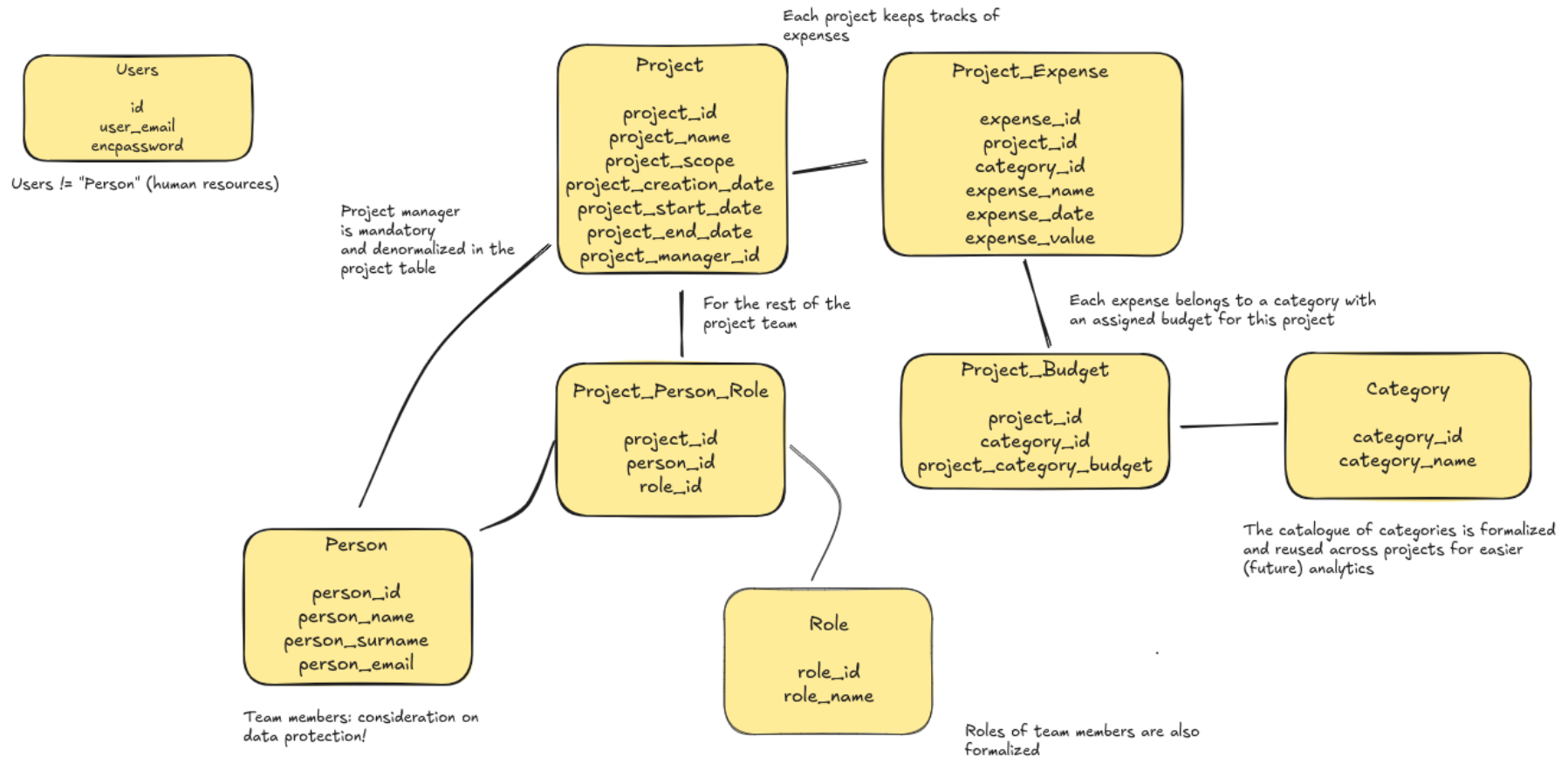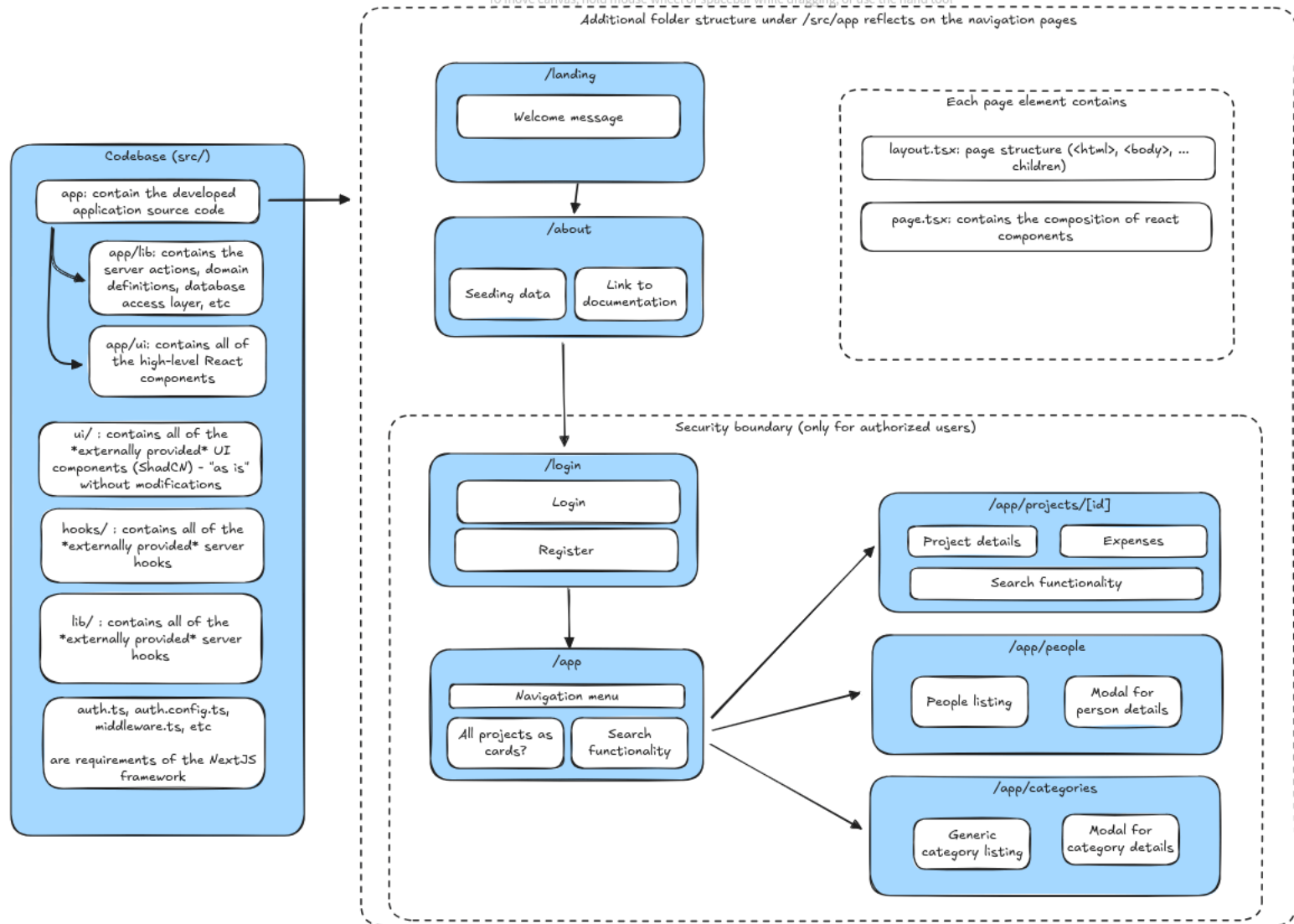project_end_date
project_manager_id

Each project keeps tracks of expenses

Project_Expense

expense_id
project_id
category_id
expense_name
expense_date
expense_value

Project manager
is mandatory
and denormalized in the
project table

For the rest of the
project team

Each expense belongs to a category with
an assigned budget for this project

Project_Person_Role

project_id
person_id
role_id

Project_Budget

project_id
category_id
project_category_budget

Category

category_id
category_name

The catalogue of categories is formalized
and reused across projects for easier
(future) analytics

Person

person_id
person_name
person_surname
person_email

Role

role_id
role_name

Team members: consideration on
data protection!

Roles of team members are also
formalized

Software design

Demo!

[Click here]

Thanks!

Questions?