Roadmap & Development History — “Gittes Glamping” (React + MUI)

Timeline Overview

Stage A — “Initial Build (React + MUI)”

- Project bootstrapped with Vite/React, MUI theme created (primary #33626C, background #CED3CD, accent #c5b496), grid-based layout.
- Implemented:
  - Navbar + routes (/, /ophold, /ophold/:id, /aktiviteter, /kontakt, /kontakt/sent, /mylist, /login, /admin/activities).
  - Core sections (Hero, CTA panels, cards, “Book” → Contact redirect).
  - Contact form with react-hook-form and MUI components (basic validation + toast notifications).
- Some early issues:
  - Duplicate navbar rendering.
  - Fetching reviews from lib/db/mcd/seed/seed.data.js adapter setup.

Stage B — Adjusted spacing, typography, shadows, border-radius, image aspect ratios, and color matching based on XD file.

- CTA “Book” correctly navigates to Contact page.
- Activities.jsx: hero image, card grid (canoe, hiking, yoga, wine tasting), chips/tags, fallback local dataset.

Stage C — “Double Navbar Fix”

- Cause: both a global MUI AppBar and page-specific Header were being rendered.
- Solution: unified into a single global Navbar component; removed duplicates from pages; routed via layout <Outlet />; corrected z-index and padding under header.

Stage D — “React + Node Integration”

Media College Denmark
/ Medieskolerne

Ionut Catalin Belu

- Goal: merge existing Node server index with React entry point.
    - Kept React bootstrap separate — ensuring backend runs without blocking the UI.
- Result: .env variables accessible globally, and project ready for API connection later.

Stage E — “Rebuilding the Stays (Ophold) Section”

- Routes: /ophold (list view) and /ophold/:id (detail view).
- List: cards with image, title, price, badges; future filtering/sorting prepared.
- Detail: small gallery, description, amenities, and “Book” button.
- Data: mock data + api hook layer for eventual backend link.
- UI tuning: spacing, font scale, icons matched to XD design.

Stage F — “Contact.jsx & Form Validation”

- MUI-based form with basic checks (name, email, subject, message) and toast.error messages.
- Issues addressed:
    - Inconsistent success/error feedback → added states and redirect to /kontakt/sent.

Stage G — “Attempt to Rebuild from Scratch (Failed Attempt)”

- Motivation: start clean to keep only necessary parts.
    - Tight deadline; incomplete data structure and routes.
    - Missing functionality at this point:
        - No full router setup or Backoffice guard.
        - No mock/API adapter or connected seed data.
        - No centralized theme or design tokens.

        - No shared components (Button, Section, Container) → inconsistent visuals.
    - Technical blocker: Babel “decorators” error in Forside.jsx:
        - Required @babel/plugin-proposal-decorators or @babel/plugin-syntax-decorators in config.
        - Config not properly loaded → build failed.

What Was Missing / Future Roadmap

1. UI/UX & Design System
    - Shared components: <Section>, <Container>, <CardGL>, <CTAButton>.
    - Implement hero image slider with autoplay & swipe gestures.
2. Routing & Access Control
    - Protected routes for Backoffice (auth guard).
    - Add 404 and ErrorBoundary pages with loading skeletons.
3. Form & Validation
    - Replace manual checks with react-hook-form + yup schemas.
    - Uniform pending/success/error states;
4. Code Quality & Testing
    - ESLint + Prettier setup.
    - Basic unit testing (Vitest/RTL).
    - Optional: Storybook for reusable components.

Technology Stack

Frontend

- React 18 + Vite

- MUI (Material UI): base components (Grid, Typography, Paper, etc.)
- Framer Motion: entrance animations and micro-interactions
- React Router v6: page navigation and dynamic routes (/ophold/:id).
- React Toastify: lightweight notification feedback (success / error).
- CSS Modules: full control of design per page with scoped class names.

Backend

- Node.js + Express: lightweight server.
- dotenv: environment management (.env.local).
- nodemon: hot reload during backend development.

Layout Replication

The most time-consuming part was translating the XD design into responsive code.
MUI components were wrapped with semantic tags and customized through sx props and CSS Modules. A recurring issue was the positioning and scaling of the large *Gitte Avatar* inside the teal capsule. At first, the image was cropped or hidden due to parent overflow.

Fonts Not Loading

Although @import of Google Fonts was included in CSS, MUI overrode font-family with its default *Roboto*.
Attempts with !important in global CSS failed.

Review Section Styling

Initial render displayed raw text in Arial. Still not solved.

Contact Page Redesign

The most difficult redesign came at the end, where the original MUI contact form was rebuilt from scratch with CSS Modules to match the provided mockups. Section animated with Framer Motion

Problem	Cause	Solution
“Support for experimental syntax ‘decorators’” error	Babel misread config in Forside.jsx	Removed unused decorator syntax and reinstalled dependencies
“Double navbar rendering”	Duplicate import of Navigation component	Cleaned routing and refactored App.jsx
CORS & env path issues	Incorrect .env.local path in dotenv.config	Added override: true and ensured relative path
Review section not picking font	MUI override	Added CssBaseline style overrides (didnt work)
Form validation not working	Missing .trim() checks	Added regex validation + Toastify feedback
Scroll overlap between hero and capsules	Negative margin conflict	Adjusted top spacing and added overflow: visible

Tailwind was initially planned for quick utility styling and responsive fine-tuning, but I couldn’t install and configure it properly within the Vite + MUI setup. The PostCSS integration caused build conflicts, and Tailwind classes weren’t being processed correctly.I eventually dropped it and relied entirely on MUI’s sx props and CSS Modules for styling control and responsiveness.

.Stay Detail Page (/ophold/:id)

Implemented routing to display single stay information from stub data.

Learning Outcomes

During the project I learned to:

- Structure a full-stack environment combining Node and React.
- Manage complex styling with CSS Modules while integrating MUI.
- Debug MUI’s internal style specificity using :global selectors.
- Use Framer Motion for lightweight transitions.
- Keep consistent design tokens (:root variables for colors / fonts).
- Translate design mockups into responsive web code.