# Microbiome-Informatics-Lab-IIIT-Delhi-Website-2025

We are developing the official website for the Microbiome Informatics Lab at IIIT Delhi as a digital extension of the FMT Hub. The platform is being built using Next.js 15, React 19, Node.js, Tailwind CSS, and TypeScript, focusing on modular UI, SEO, and responsiveness. We're implementing secure APIs, dynamic routing, and scalable backend architecture for seamless performance. Deployment is handled on Oracle Linux with Apache, using SSL-secured HTTPS. This site will serve as a centralized digital gateway for lab research, data resources, and collaborations.

Lab Lead by Professor Dr. Tarini Shankar Ghosh WebDev Team Members

- Siddhant Bali
- Rishabh Kumar
- Siddharth Bhaskar This Document Serves as a Project Tracking, Planning and Guide for Developers of Now and Future.

## Introduction

We are currently building the official website for the Microbiome Informatics Lab at IIIT Delhi, as a natural extension of our previous work on the FMT Hub. This project aims to establish a unified and professional digital presence for the lab, integrating research insights, personnel, publications, and data tools into a modern, performant web platform. The development stack includes Next.js 15, React 19, Node.js, Tailwind CSS, and TypeScript, with a focus on modular UI design, SEO optimization, and cross-device responsiveness. We're engineering dynamic routing, secure API endpoints, and scalable backend architecture to ensure smooth frontend-backend communication and future-proof extensibility. The deployment is being handled on Oracle Linux servers with Apache, using SSL-secured custom HTTPS for encrypted connections. In this phase, we are focusing on:

- Creating reusable component libraries for consistent lab branding.
- Building user-centric interfaces that highlight lab research, team profiles, and live updates.
- Integrating the FMT Hub data layer for seamless navigation between resources.
- Ensuring performance and accessibility compliance through Lighthouse audits and testing.

This project will serve as a centralized hub for collaborators, researchers, and external stakeholders, significantly elevating the visibility and digital maturity of the Microbiome Informatics Lab.

This is a Next.js project bootstrapped with `create-next-app`.

## Getting Started

First, run the development server:

```
npm run dev
# or
yarn dev
# or
pnpm dev
```

```
# or
bun dev
```

Open http://localhost:3000 with your browser to see the result.

You can start editing the page by modifying `app/page.tsx`. The page auto-updates as you edit the file.

This project uses `next/font` to automatically optimize and load Geist, a new font family for Vercel.

## Learn More

To learn more about Next.js, take a look at the following resources:

- Next.js Documentation - learn about Next.js features and API.
- Learn Next.js - an interactive Next.js tutorial.

You can check out the Next.js GitHub repository - your feedback and contributions are welcome!

## Deploy on Vercel

The easiest way to deploy your Next.js app is to use the Vercel Platform from the creators of Next.js.

Check out our Next.js deployment documentation for more details.

---

To deploy your statically exported **Next.js site (with `output: 'export'`)** to **GitHub Pages**, follow this clean and battle-tested setup:

---

## ✅ Step-by-Step GitHub Pages Deployment for `next export`

### 1. **Pre-requisites**

- Your project must be **using `output: 'export'`** in `next.config.js`
- Your Git repo must be initialized and pushed to GitHub

---

### 2. **Install `gh-pages`**

```
npm install --save-dev gh-pages
```

---

### 3. **Update `next.config.js`**

You **must set the base path and asset prefix** if your repo is not hosted at the root domain:

```
// next.config.js
const repoName = 'your-repo-name'; // CHANGE THIS to your GitHub repo name
```

```
/** @type {import('next').NextConfig} */
const nextConfig = {
  output: 'export',
  basePath: `/${repoName}`,
  assetPrefix: `/${repoName}/`
};

module.exports = nextConfig;
```

---

## 4. Update `package.json`

**Add `homepage` key:**

```
"homepage": "https://yourusername.github.io/your-repo-name",
```

**Add scripts:**

```
"scripts": {
  "dev": "next dev",
  "build": "next build && next export",
  "start": "npx serve out",
  "predeploy": "npm run build",
  "deploy": "gh-pages -d out"
}
```

---

## 5. Build and Deploy

```
npm run deploy
```

This will:

- Build and export the static site to `/out`
- Push the contents of `/out` to the `gh-pages` branch of your repo

---

## 6. Enable GitHub Pages in Repo Settings

- Go to your repo on GitHub → **Settings** → **Pages**
- Source: select `gh-pages` branch, then `/ (root)`
- Wait a few seconds, and your site will be live at:

```
https://yourusername.github.io/your-repo-name/
```

---

## 🧠 Tip: Automate it

If you want to run everything in one go:

```
npm run deploy
```

That’s it! Let me know if you want this to also work with a **custom domain** or need a `CNAME` file setup.

---

## not-found.tsx

```
"dev": "next dev",
```

"build": "next build && next export", "start": "next start", "lint": "next lint", "export": "next export", "meow": "npm i && next dev", "meowmeow": "npm run build && npx serve@latest out"

for handling Error: "next start" does not work with "output: export" configuration. Use "npx serve@latest out" instead.