PRACTICAL NO : 08

Title : Design and develop custom Application (Mini Project) using Salesforce Cloud.(Pitching Management System)

Author: Sarthak Velapure

1. Introduction

The purpose of this project is to demonstrate how to deploy a web application using Netlify, a cloud platform that provides automated continuous deployment for static sites and serverless functions. This report outlines the steps taken to deploy an application, the challenges encountered, and the outcomes.

1.1 Project Overview

The project involves deploying a general knowledge website using Netlify's cloud platform. This deployment utilizes Netlify's CI/CD pipeline, automated build processes, and environment configuration to ensure smooth and efficient hosting of the application.

2. Objectives

- Deploy the application to Netlify from a Git repository (GitHub, GitLab, etc.).
- Implement automatic builds upon each code change.
- Ensure proper configuration of environment variables and build settings.
- Verify the application’s performance, security, and responsiveness on the Netlify platform.

3. Tools and Technologies

- Netlify: Cloud platform for hosting and deploying static sites.
- Git: Version control system.

- GitHub/GitLab/Bitbucket: Repository platform.
- HTML/CSS/JavaScript (or relevant stack used in the project): Front-end development.
- Node.js (if applicable): Backend or serverless functions for dynamic features.

4. Methodology

4.1 Project Setup

1. Codebase Preparation: The codebase was prepared using [technologies and frameworks] such as [React, Vue.js, or any other framework] to ensure compatibility with static hosting on Netlify.
2. Repository Setup: The application code was hosted on [GitHub/GitLab/Bitbucket]. A public/private repository was created for storing the project files. Git was used to manage version control, and the repository was linked to the Netlify platform.

4.2 Deployment Process

1. **Connecting the Repository to Netlify:**
    - The repository was connected to Netlify by linking the GitHub/GitLab/Bitbucket account.
    - Netlify automatically detected the branch used for deployment (e.g., main or master).
2. **Configuring Build Settings:**
    - Specified the build command (e.g., npm run build, gatsby build).
    - Set the publish directory (e.g., build/, public/).
    - Configured environment variables (if any) for production build (e.g., API keys, API URLs).

3. **Automated Deployment:**
    - Once the repository was connected, Netlify automatically built and deployed the project.
    - Any push or pull request to the connected branch triggered a new deployment.
4. **DNS & Custom Domain Configuration:**
    - Configured a custom domain for the project (e.g., www.example.com).
    - Set up DNS records to point to Netlify's servers for handling the domain.
    - Configured SSL/TLS certificates using Netlify's built-in automatic SSL support for secure browsing.

4.3 Performance & Testing

- Performance Testing: Verified application load times, performance, and optimization using Netlify Analytics and external tools like Google Lighthouse.
- Security Checks: Ensured HTTPS was enforced and reviewed Netlify's security features like edge protection and serverless security.

5. Challenges and Resolutions

5.1 Deployment Errors

- Build Failure: Encountered issues due to incorrect build commands or outdated packages. Resolved by ensuring proper configuration in package.json and updating dependencies.

5.2 Environment Variables Configuration

- Encountered issues with environment variables not being available during the build. Solved by properly configuring the Netlify Environment Variables section.

5.3 Domain and DNS Setup

- Faced slight delays during DNS propagation after setting up a custom domain. Resolved by using Netlify's domain management tools.

6. Results

- The web application was successfully deployed to the Netlify cloud platform.
- Automatic builds were triggered with each update to the Git repository.
- The application was accessible via both the default Netlify subdomain and the custom domain.
- Performance: The website performed well under test conditions, showing fast load times.
- Security: The application used SSL certificates for secure connections.

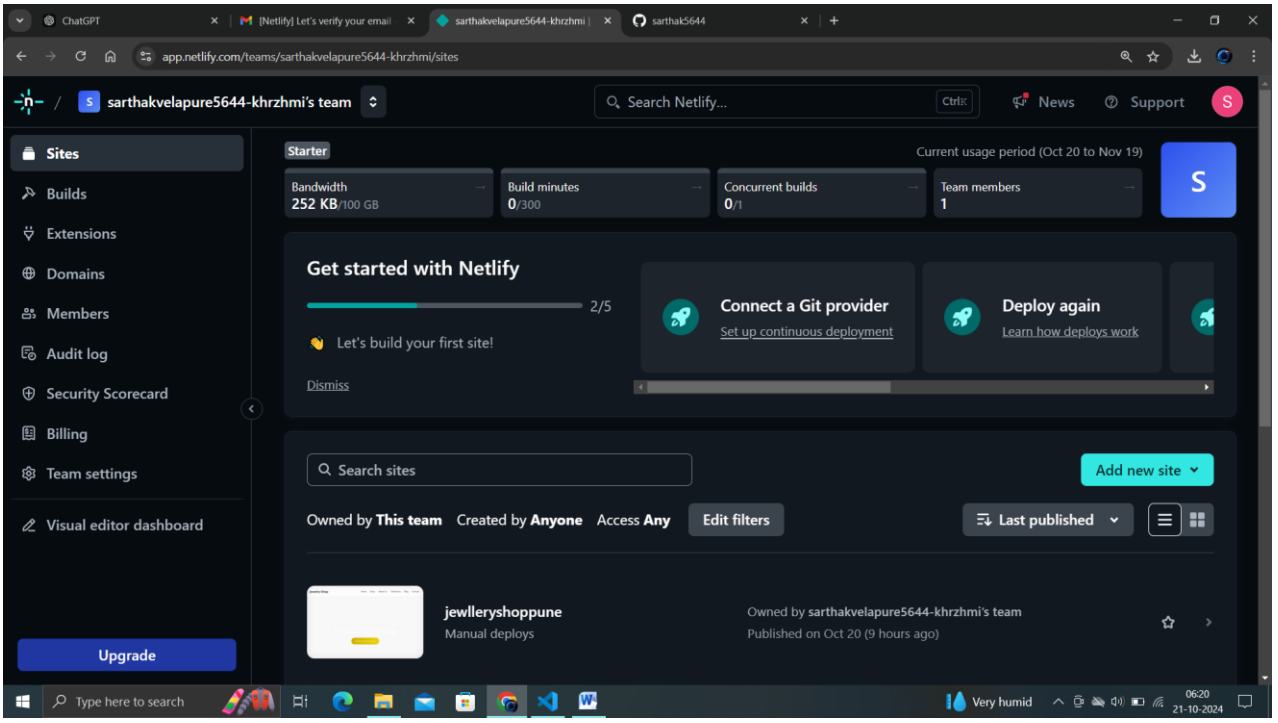
7. Conclusion

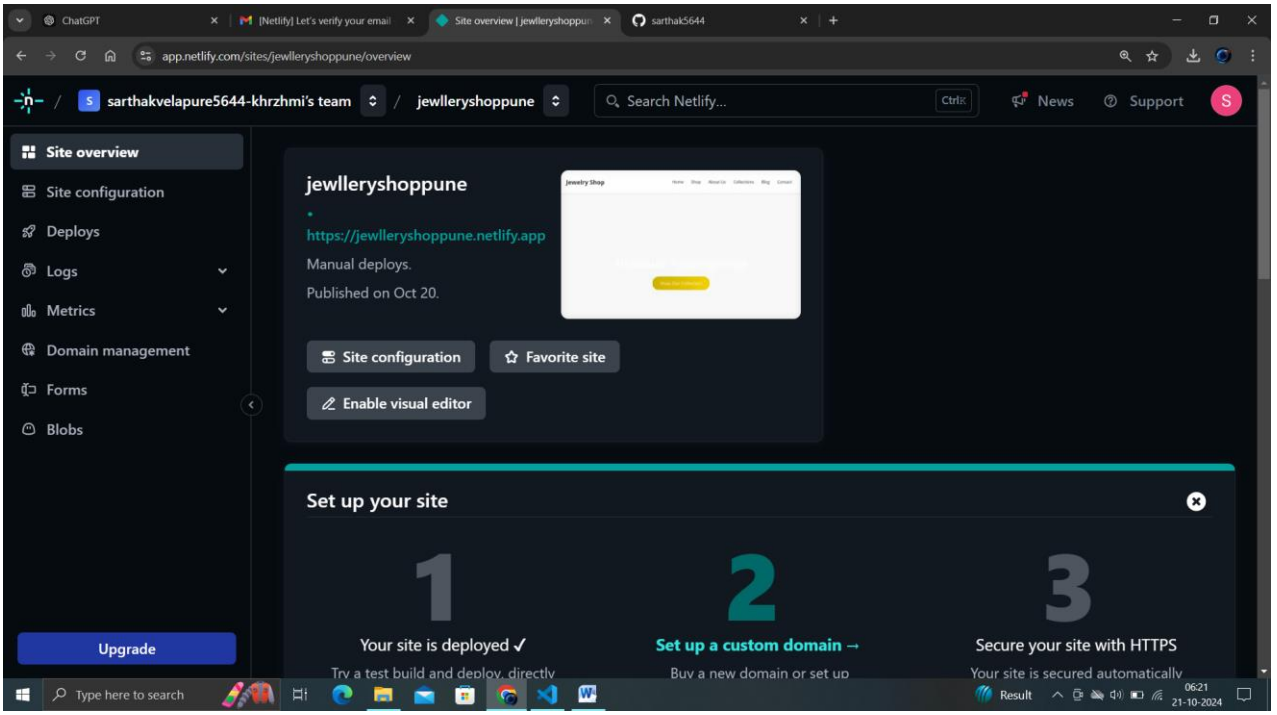
The deployment of the web application to Netlify was successful, with Netlify offering a simple yet powerful CI/CD pipeline that ensures continuous deployment of the project. With features like automated SSL, custom domain handling, and environment variables, Netlify proved to be a robust platform for deploying static websites and serverless applications.

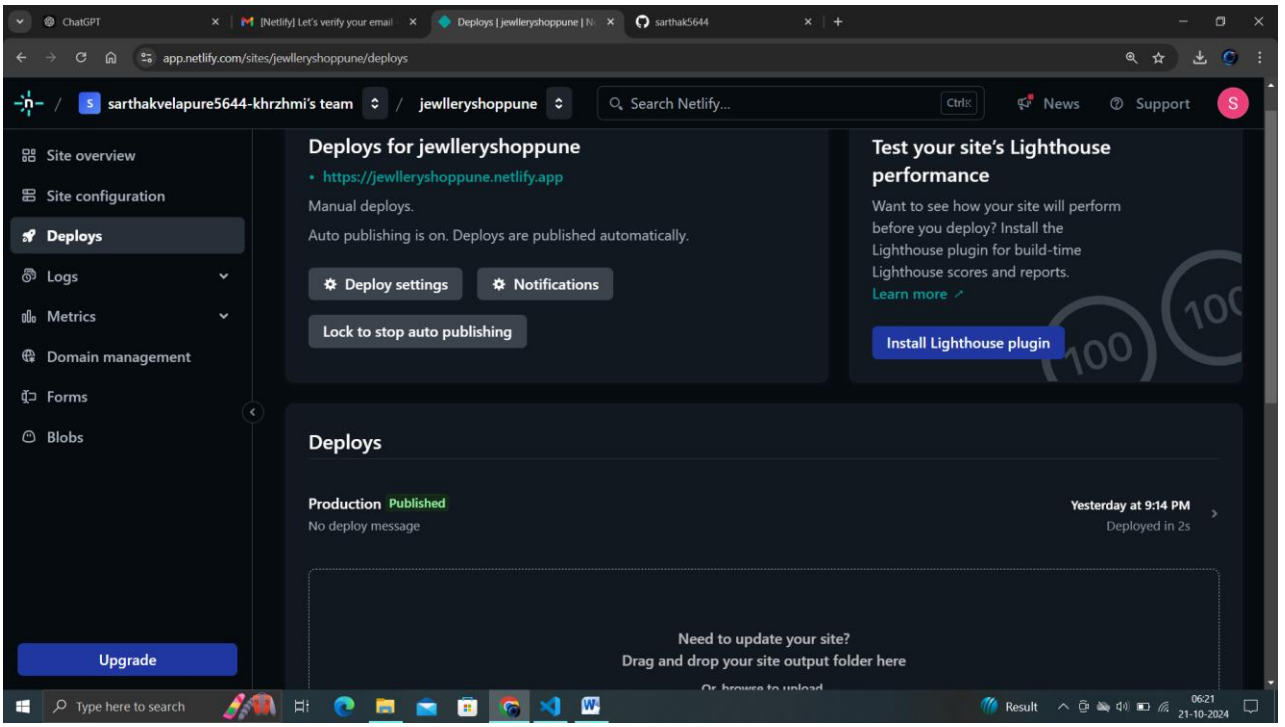
7. References

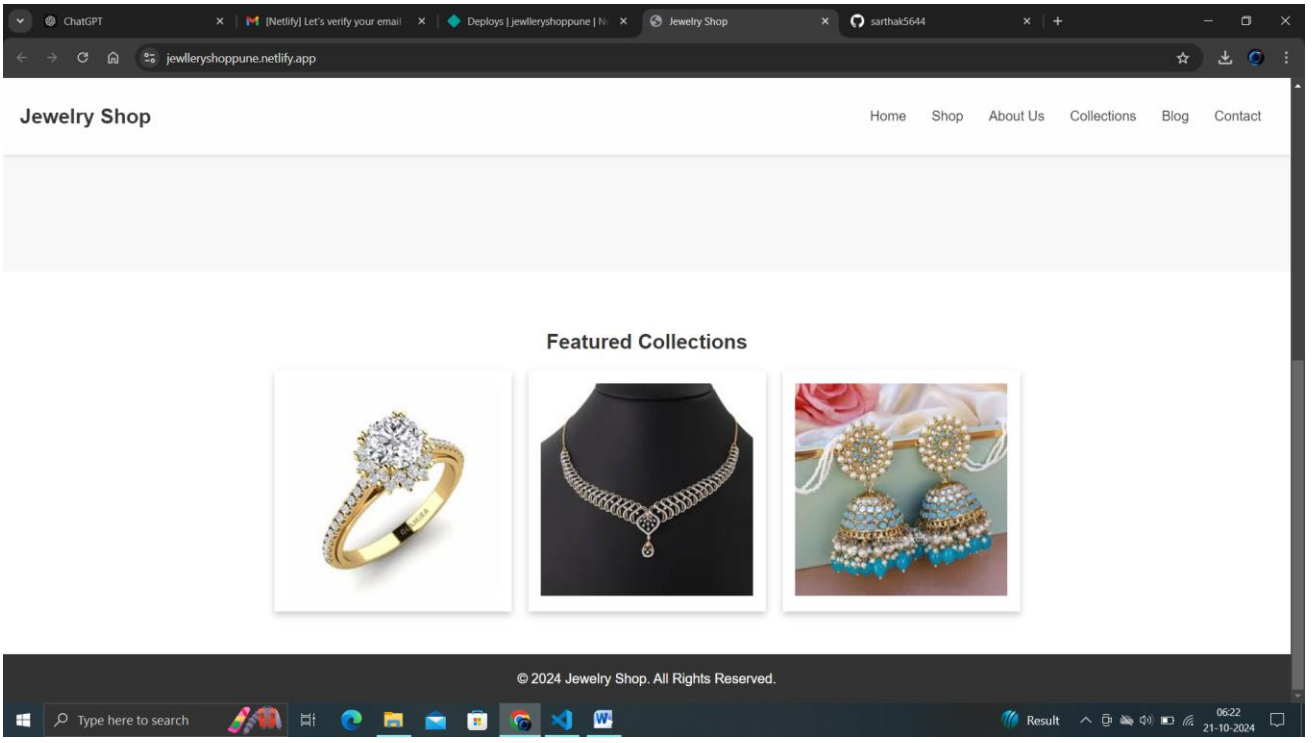
- **Netlify Documentation**
- **https://github.com/sarthak5644**

8. Screenshots:









Link: https://jewlleryshoppune.netlify.app