Step 1: Hosting Platform Setup

Choose a hosting platform that supports Next.js applications, such as Vercel or Netlify, which are popular for their simplicity and integration with GitHub.

1. **Platform Selection and Setup:**
    - **Vercel**: Recommended for Next.js projects due to its seamless integration.
    - **Netlify**: Also a good choice with support for serverless functions.
2. **Repository Connection:**
    - Connect your GitHub repository directly through the Vercel or Netlify dashboard.
3. **Configuration:**
    - Set up build commands and environment variables through the platform's dashboard

```
// Example of a typical Vercel configuration file (vercel.json)
{
  "version": 2,
  "builds": [{ "src": "next.config.js", "use": "@vercel/next" }],
  "routes": [{ "src": "/(.*)", "dest": "/" }]
}
```

Step 2: Configure Environment Variables

Securely manage sensitive information such as API keys and database URLs.

1. **Local Environment Setup:**
    - Create a `.env.local` file in your Next.js project root.

```
// .env.local
NEXT_PUBLIC_API_URL=https://api.example.com
DATABASE_URL=your_database_connection_string
```

Deployment Environment Variables:

- Add these variables to your hosting platform (Vercel/Netlify) to ensure they are available in the staging environment.

Step 3: Deploy to Staging

Deploy your application to a staging environment to mimic production settings.

1. **Trigger Deployment:**
   - Use the hosting platform's dashboard to deploy the latest commit from the connected GitHub repository.

2. **Validate Deployment:**
   - Ensure the staging site is live and accessible without build errors.

Step 4: Staging Environment Testing

Comprehensive testing in the staging environment to ensure functionality and performance.

1. **Functional Testing with Cypress:**
    - Write end-to-end tests in TypeScript to verify critical workflows

```
// Example Cypress test in TypeScript
describe('Product Interaction', () => {
  it('allows users to add products to the cart', () => {
    cy.visit('/products');
    cy.findAllByText('Add to Cart').first().click();
    cy.get('[data-testid="cart-count"]').should('contain', '1');
  });
});
```

Performance Testing with Lighthouse:

- Automate performance testing using Lighthouse CI.

```
npx @lhci/cli autorun
```

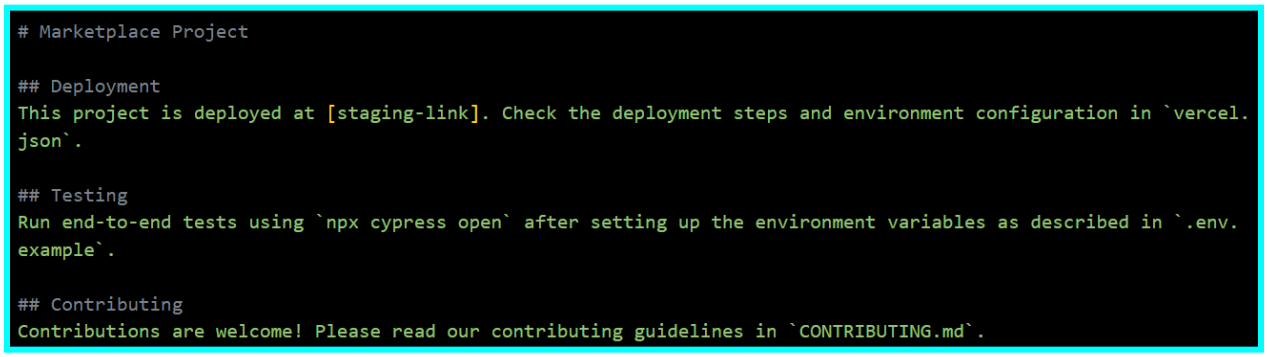
Security Testing:

- Ensure secure communication and data handling practices are in place.

Step 5: Documentation Updates

Create detailed documentation including all test results, configurations, and deployment steps.

1. **README.md Update:**
    - Summarize the project setup, deployment process, and how to run tests.



```
# Marketplace Project

## Deployment
This project is deployed at [staging-link]. Check the deployment steps and environment configuration in `vercel.
json`.

## Testing
Run end-to-end tests using `npx cypress open` after setting up the environment variables as described in `.env.
example`.

## Contributing
Contributions are welcome! Please read our contributing guidelines in `CONTRIBUTING.md`.
```

1. **Test Reports and Performance Metrics:**
    - Include CSV files or links to performance dashboards.

Expected Output:

- A live staging site that mirrors the production environment.
- Comprehensive test coverage with documented results.
- A well-documented repository that facilitates easy understanding and contributions.