DAY 7 - LIVE DEPLOYMENT AND POST-LAUNCH PRACTICES

GOVERNOR INITIATIVE WEB AND METAWERSE

CREATED BY: NADIA

DAY 7 - LIVE DEPLOYMENT AND POST-LAUNCH PRACTICES

Table of Contents

1. Introduction

This document describes the procedures, best practices, and lessons learned during the live deployment and post-launch phase of our project. It outlines the steps taken on Day 7, details the deployment process, and discusses subsequent monitoring and maintenance practices.

Purpose:

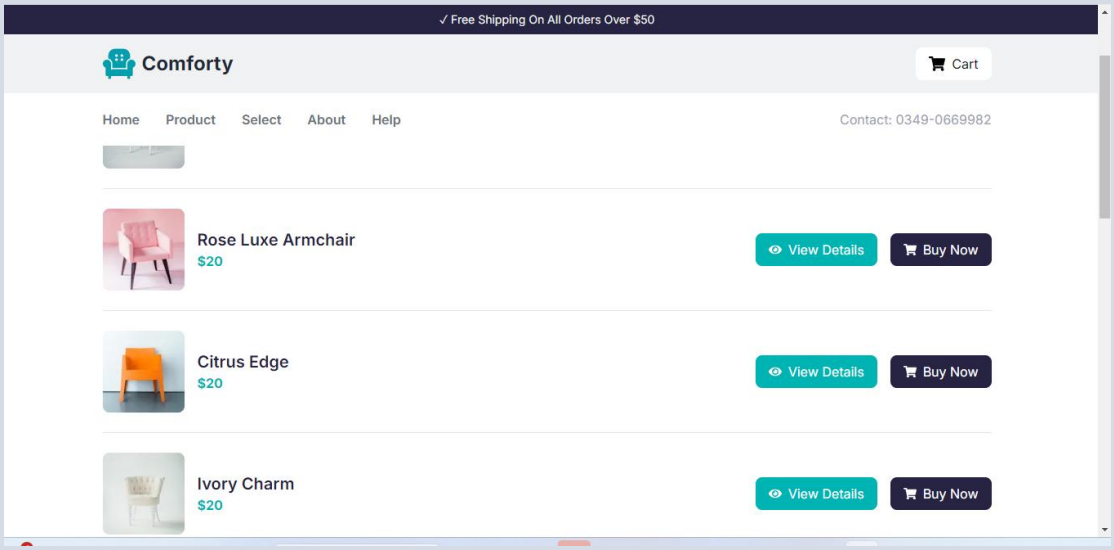
- To serve as a reference guide for similar deployments in the future.
- To provide a structured approach for live deployment and post-launch activities.
- To highlight challenges and recommended improvements.

2. Overview of Live Deployment

Live deployment refers to the process of transitioning a project from a staging environment to a live, production environment. This phase involves rigorous testing, careful coordination among teams, and a comprehensive rollback plan in case of unexpected issues.

Key Objectives:

- Ensure minimal downtime during transition.
- Validate that all systems function as expected.
- Establish monitoring protocols immediately post-launch.



DASHBOARD LAYOUT

3. Pre-Deployment Preparations

Before initiating the live deployment, the following preparations are essential:

3.1. Environment Setup

- **Staging vs. Production:** Ensure that both environments mirror each other as closely as possible.
- **Configuration Management:** Validate that all configuration files (e.g., database connections, API endpoints) are updated for production use.

3.2. Backup and Rollback Plans

- **Full Backup:** Perform a complete backup of current production data and code.
- **Rollback Procedure:** Prepare a documented rollback strategy if the deployment encounters critical issues.

3.3. Team Coordination

- **Deployment Team:** Confirm roles and responsibilities (e.g., lead developer, QA, operations).
- **Communication:** Establish a communication channel (e.g., Slack, Microsoft Teams) for real-time updates.

4. Live Deployment Process

(Approximately 2 pages)

The live deployment process is broken down into several key stages:

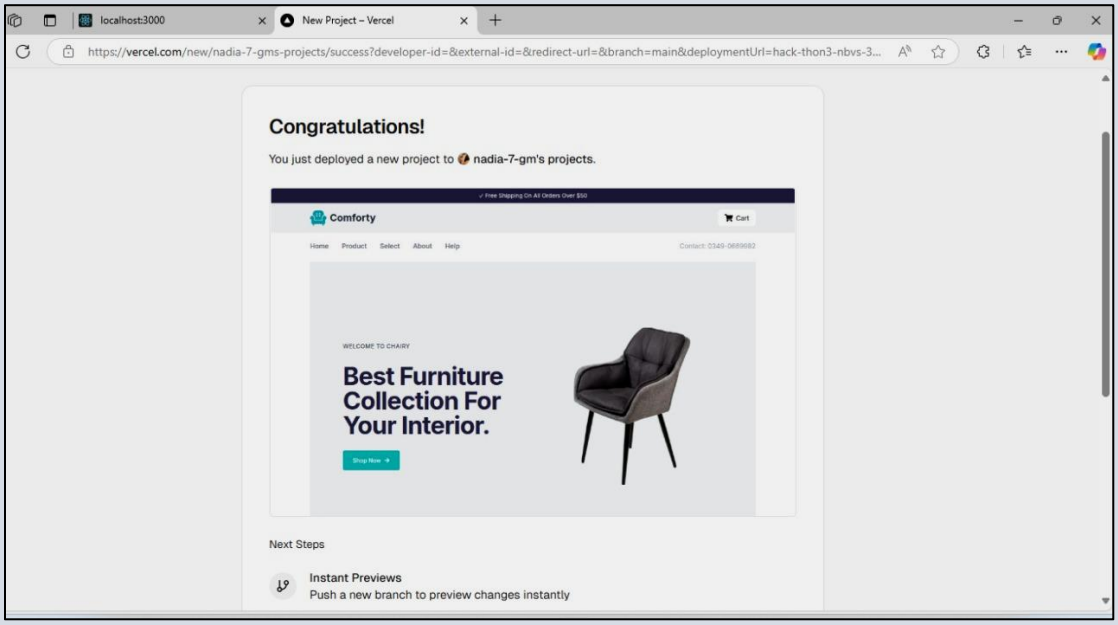
4.1. Initiation

- **Final Check:** Review the deployment checklist to ensure all prerequisites are met.
- **Deployment Window:** Schedule the deployment during a low-traffic period to minimize impact.

4.2. Execution

- **Step-by-Step Process:**

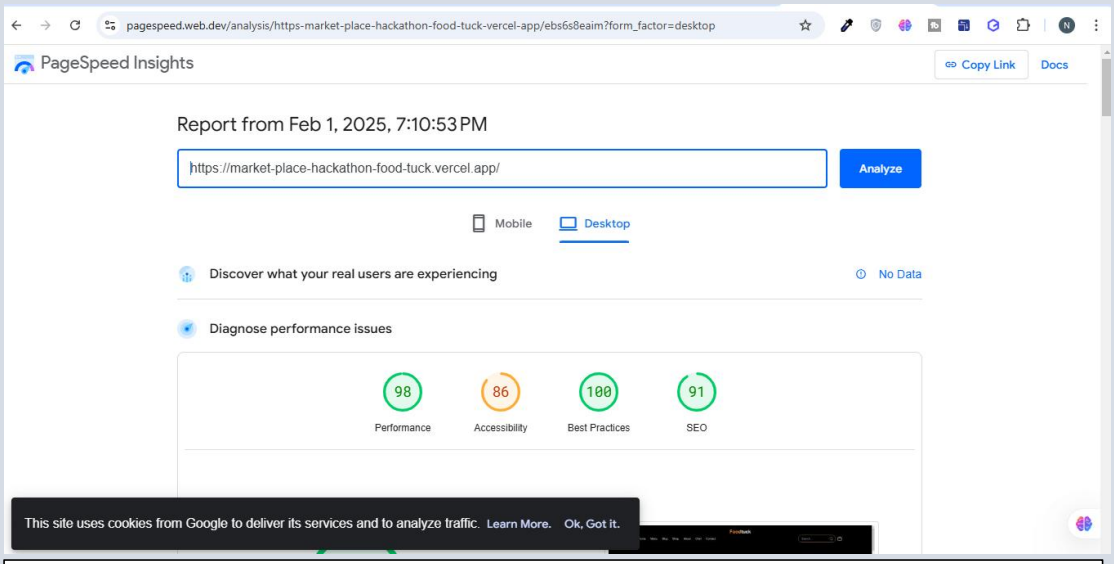
    1. Initiate downtime notification on the website (if required).
    2. Deploy updated code and configuration files using automated scripts or manual updates.
    3. Run database migrations if necessary.



DEPLOYEMENT PROCESS

4.3. Verification

- **Smoke Testing:** Conduct immediate tests to ensure the application is operational.



PAGE INSIGHTS

- **Monitoring Tools:** Activate monitoring systems (e.g., New Relic, Datadog) to track performance.

---

5. Testing and Validation

Post-deployment testing is critical to validate that the system performs as expected.

5.1. Functional Testing

- Verify that all core functionalities are working.
- Confirm user authentication, data retrieval, and transaction processing are error-free.

5.2. Performance Testing

- Monitor load times and responsiveness.
- Compare current performance metrics against previous benchmarks.

5.3. User Feedback

- Engage a subset of users to report any issues they face during early hours of the deployment.

---

6. Post-Launch Practices

(Approximately 1 page)

After the live deployment, a structured post-launch phase is critical.

6.1. Immediate Post-Launch Activities

- **Monitoring:** Continuous tracking of system performance and error logs.
- **Customer Support:** Establish a rapid response team to address any user issues.

6.2. Scheduled Reviews

- **Daily Check-ins:** Hold meetings for the first week post-launch to review system health.
- **User Engagement:** Collect user feedback and review analytics for potential improvements.

---

7. Monitoring and Maintenance

To ensure long-term stability:

7.1. Continuous Monitoring

- **Automated Alerts:** Configure alerts for critical metrics (e.g., server downtime, error rates).
- **Dashboard Reviews:** Regularly review monitoring dashboards to detect anomalies.

7.2. Maintenance Procedures

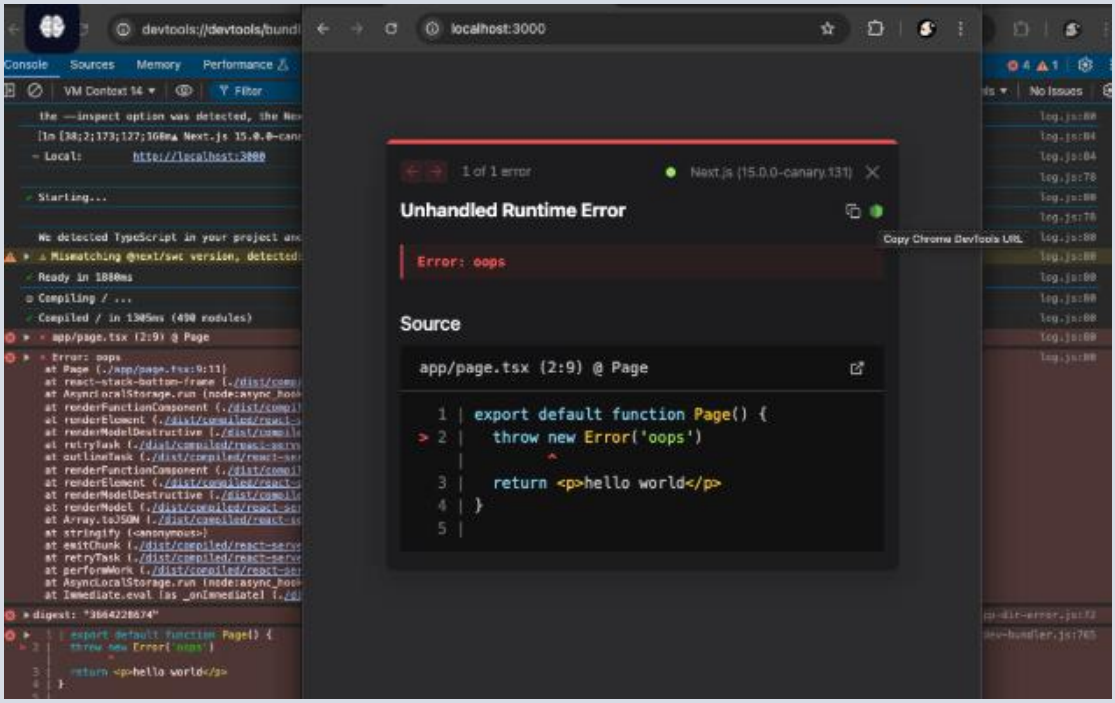
- **Regular Updates:** Schedule periodic updates and patches.
- **Documentation:** Keep the deployment and maintenance documentation updated for future reference.

---

8. Challenges and Lessons Learned

8.1. Challenges Encountered

- **Technical Hurdles:** List any unexpected issues (e.g., compatibility issues, server overload).
- **Communication Gaps:** Address any miscommunication or coordination problems.



ERRORS THAT TOOK TIME BUT MADE MN STRONG EOUGH TO SOLVE PROBLEMS

```
npm ERR! code EEXIST
npm ERR! syscall rename
npm ERR! path /Users/moeundara/.npm/_cacache/tmp/333f9c08
npm ERR! dest /Users/moeundara/.npm/_cacache/content-v2/sha512/be/37/47bd9b477cd
311cf874e459fa94bf5d8878a93c43028268a5407397f6b09b73550b9ed46737c1232fcd2f0f3806
818f41c45035f49b5f3474e86b8a786fe
npm ERR! errno EEXIST
npm ERR! Invalid response body while trying to fetch https://registry.npmjs.org/
globals: EACCES: permission denied, rename '/Users/moeundara/.npm/_cacache/tmp/3
33f9c08' -> '/Users/moeundara/.npm/_cacache/content-v2/sha512/be/37/47bd9b477cd3
11cf874e459fa94bf5d8878a93c43028268a5407397f6b09b73550b9ed46737c1232fcd2f0f38068
18f41c45035f49b5f3474e86b8a786fe'
npm ERR! File exists: /Users/moeundara/.npm/_cacache/content-v2/sha512/be/37/47b
d9b477cd311cf874e459fa94bf5d8878a93c43028268a5407397f6b09b73550b9ed46737c1232fcd
2f0f3806818f41c45035f49b5f3474e86b8a786fe
npm ERR! Remove the existing file and try again, or run npm
npm ERR! with --force to overwrite files recklessly.

npm ERR! A complete log of this run can be found in: /Users/moeundara/.npm/_logs
/2023-11-01T10_35_19_906Z-debug-0.log
```

NEXTJS INSTALLATION ERROR

8.2. Lessons Learned

- **Improved Planning:** Emphasize the need for detailed pre-deployment planning.
- **Better Automation:** Recommend enhancing automation to reduce human error during deployment.

Challenge	Solution
Git Push Rejection (Fetch First Error)	Use `git pull --rebase origin main` before pushing. If needed, use `git push --force` to overwrite remote changes.
Folder Structure Not Organizing Properly	Manually create folders before committing or use `mkdir FolderName` in PowerShell.
README.md Not Being Tracked	Ensure file is named correctly (not `README.md.txt`) and use `git add README.md` before committing.
Word File Naming Issues	Avoid special characters, long filenames, and check file save location.
OneDrive Sync Conflicts with Git	Pause OneDrive sync temporarily while working on the repo.
Push Force Needed to Replace Repo	Use `git push -u origin main --force` cautiously to prevent unwanted data loss.

9. Future Improvements

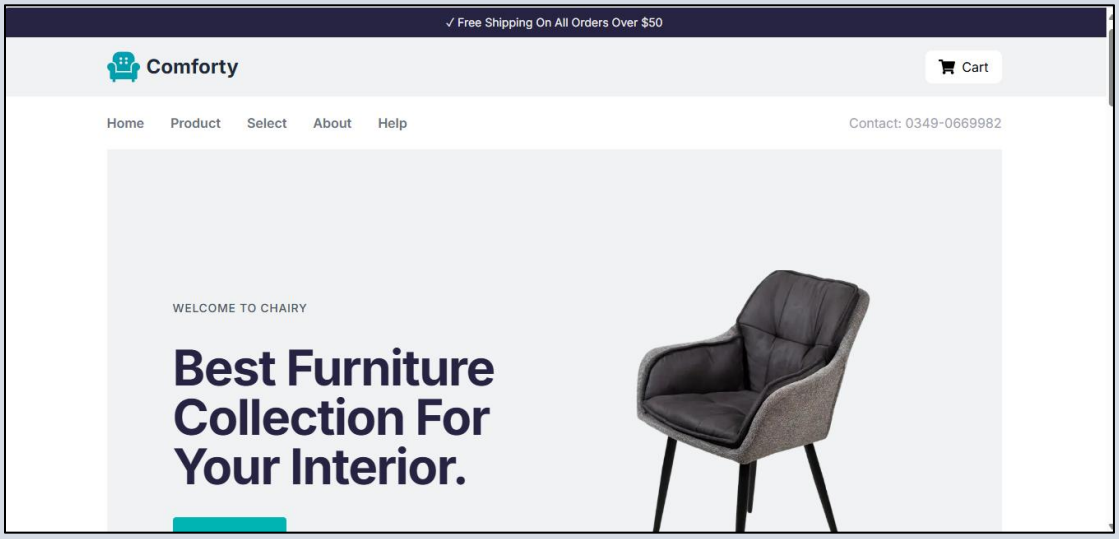
Outline steps for further enhancements:

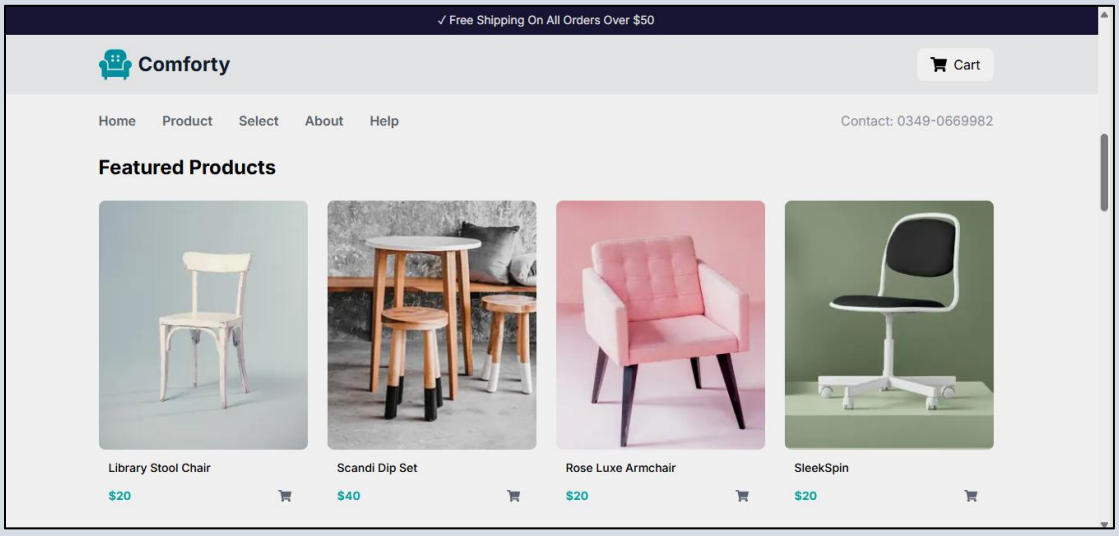
- **Process Optimization:** Automate additional parts of the deployment process.
- **Tool Upgrades:** Invest in advanced monitoring and analytics tools.
- **Training Sessions:** Regularly train the team on new deployment technologies and practices.

---

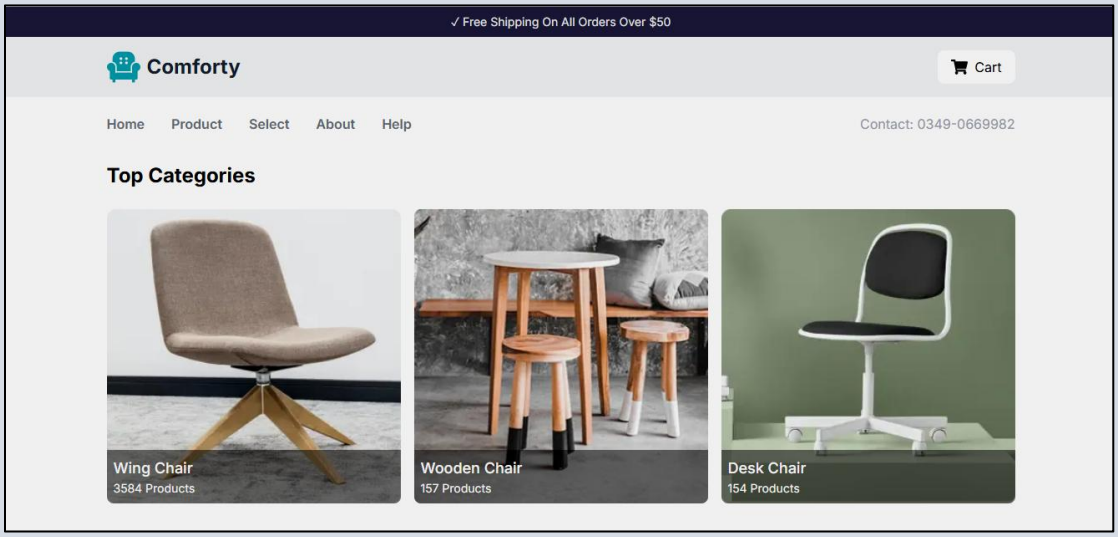
10. Conclusion

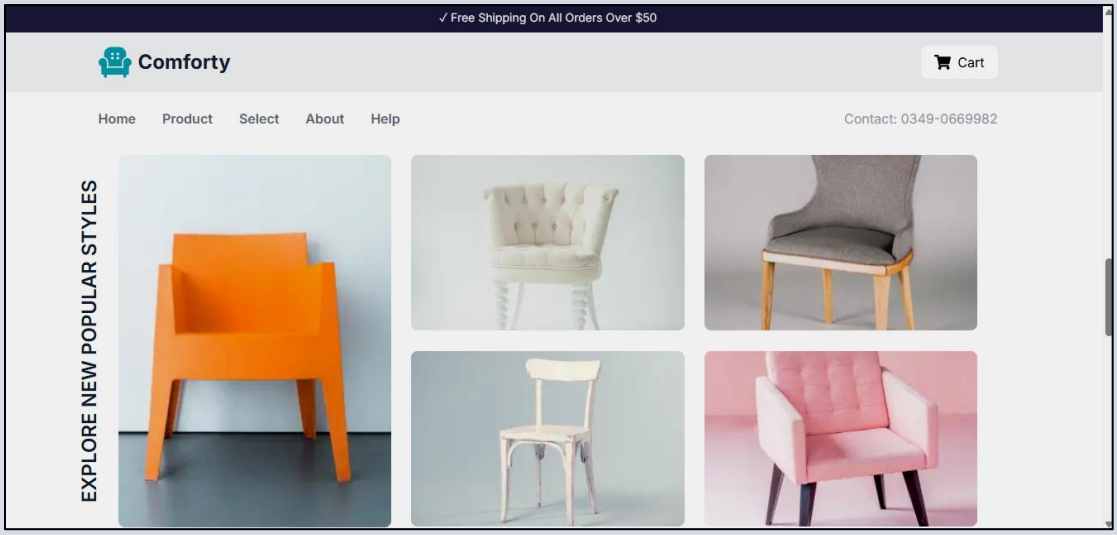
Summarize the deployment experience, highlighting the success factors and areas for improvement. Reiterate the importance of continuous monitoring and proactive maintenance in ensuring a seamless live environment.



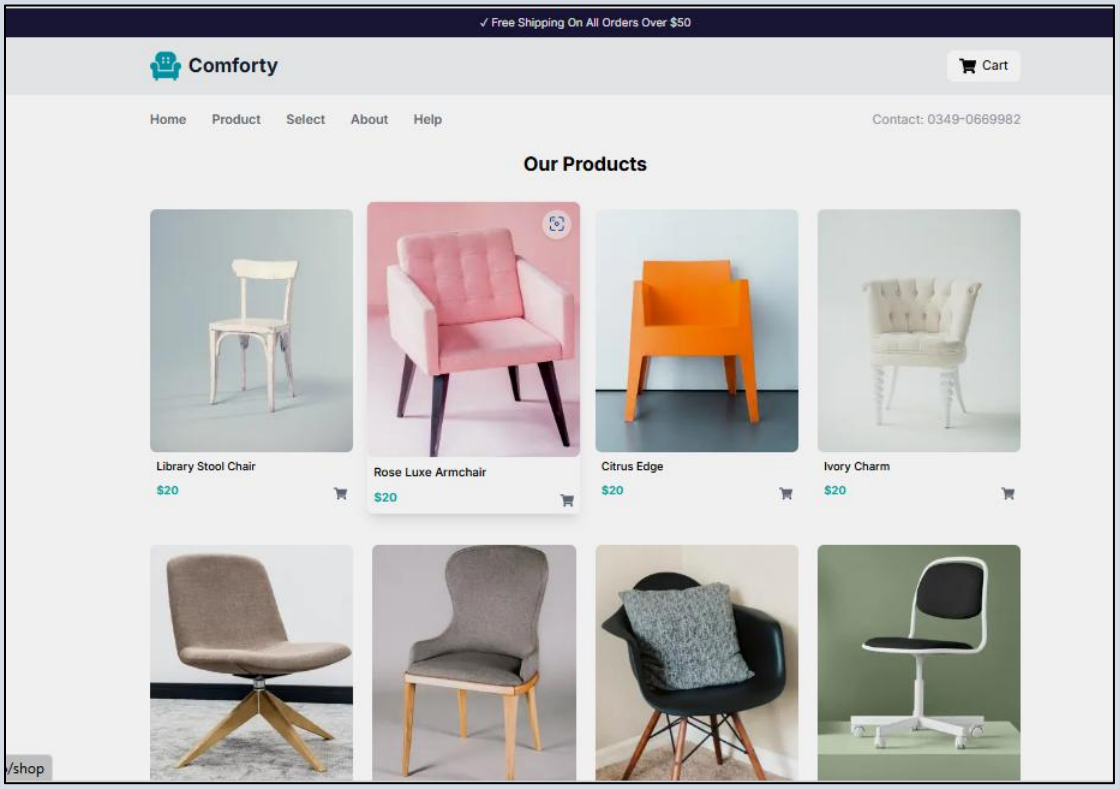


VARIOUS CATEGORIES

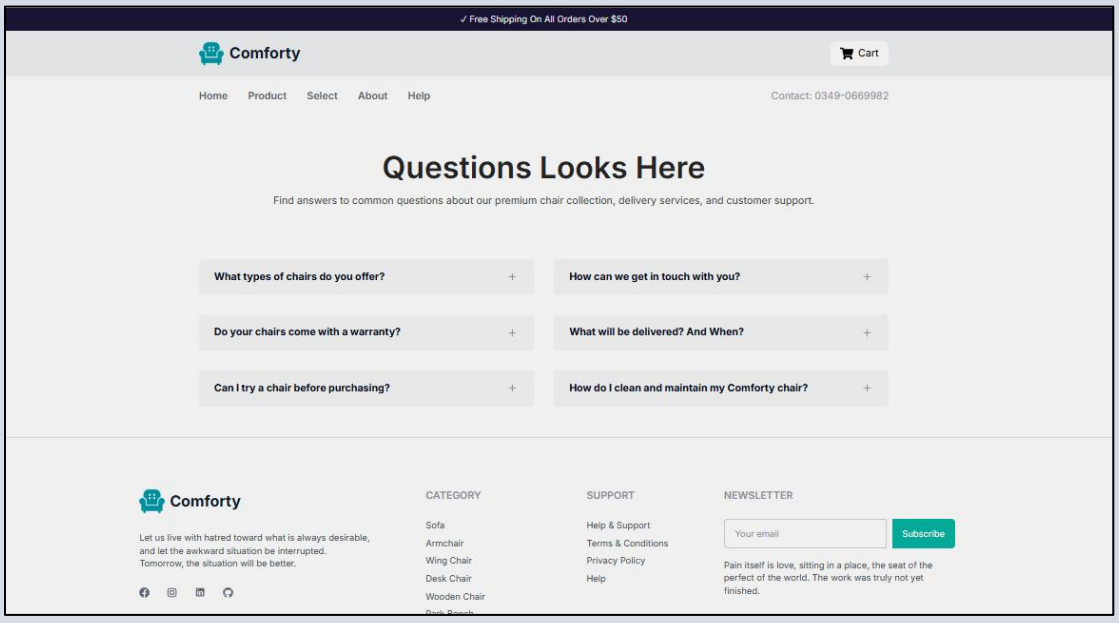




OUR PRODUCTS



QUERIES SOLUTION THROUGH FEEDBACK



11. Appendices

- Detailed deployment checklist.