Software Engineering Department
ORT Braude College

Capstone Project Phase B – 61999

Shelter Construction Management

24-1-D-43

Project Supervisor: Assoc. Prof. Zeev Barzily.

Project Members:
Almog Khaikin, Almog.Khaikin@e.braude.ac.il.
Nir Betesh, Nir.Betesh@e.braude.ac.il.

Table of Contents

1. Introduction

The Shelter Construction Management system has now been fully implemented, achieving its goal of transitioning the company's workflow from manual, paper-based methods to a streamlined, digital solution. This system has been designed to meet the specific needs of the shelter construction company, simplifying complex processes and reducing the likelihood of errors.

One of the key enhancements made during the second phase of the project was the introduction of a feature that allows for the distinction between different apartments within the same building in a contract. This improvement was implemented in response to feedback from our project supervisor, ensuring that each apartment is treated as a separate entity within the contract, thus providing more accurate and detailed management of construction projects.

Throughout the development process, the team faced several challenges. Notably, the college's restrictions on accessing certain websites, such as the Drizzle database, hindered our ability to work on the project within the college environment. Additionally, as we were learning a new programming language and framework (TypeScript, Svelte, and SvelteKit), it took time to acclimate and fully leverage their capabilities. This learning curve led us to refactor and optimize our code, resulting in a more efficient and robust system.

Despite these challenges, our progress has been substantial. The project is large in scope, and while there were occasional setbacks due to communication issues and unexpected problems, we have maintained a steady pace of development. Through this experience, we have learned the importance of sprints and Agile methodologies, which have been critical in managing our workload and adapting to changes throughout the project lifecycle.

Overall, the Shelter Construction Management system is now a comprehensive solution that significantly improves the company's ability to manage contracts, inventory, and construction projects. It represents a significant step forward in the company's operational efficiency, providing a model that could be adapted for use in similar industries.

2. Solution Description

2.1. System Architecture

The Shelter Construction Management system was designed with a distributed architecture comprising three main components: the Frontend, the Backend, and the Database. This architecture allows for flexibility and efficient management of complex processes.

- **Frontend**: The frontend of the system was developed using Svelte and SvelteKit, providing an intuitive and user-friendly interface tailored to non-technical users. This interface is used by all roles within the system, including the CEO, secretaries, installers, and inventory managers.
- **Backend**: The central server, built using SvelteKit, handles all data requests, including database management and communication with the frontend. The backend is structured to support multiple concurrent users while maintaining data consistency and security.
- **Database**: Drizzle was chosen as the database to store all system data, including information on clients, contracts, inventory, and employees. The database is cloud-hosted, offering secure and fast access to data from any location.

2.2. Software architecture diagram

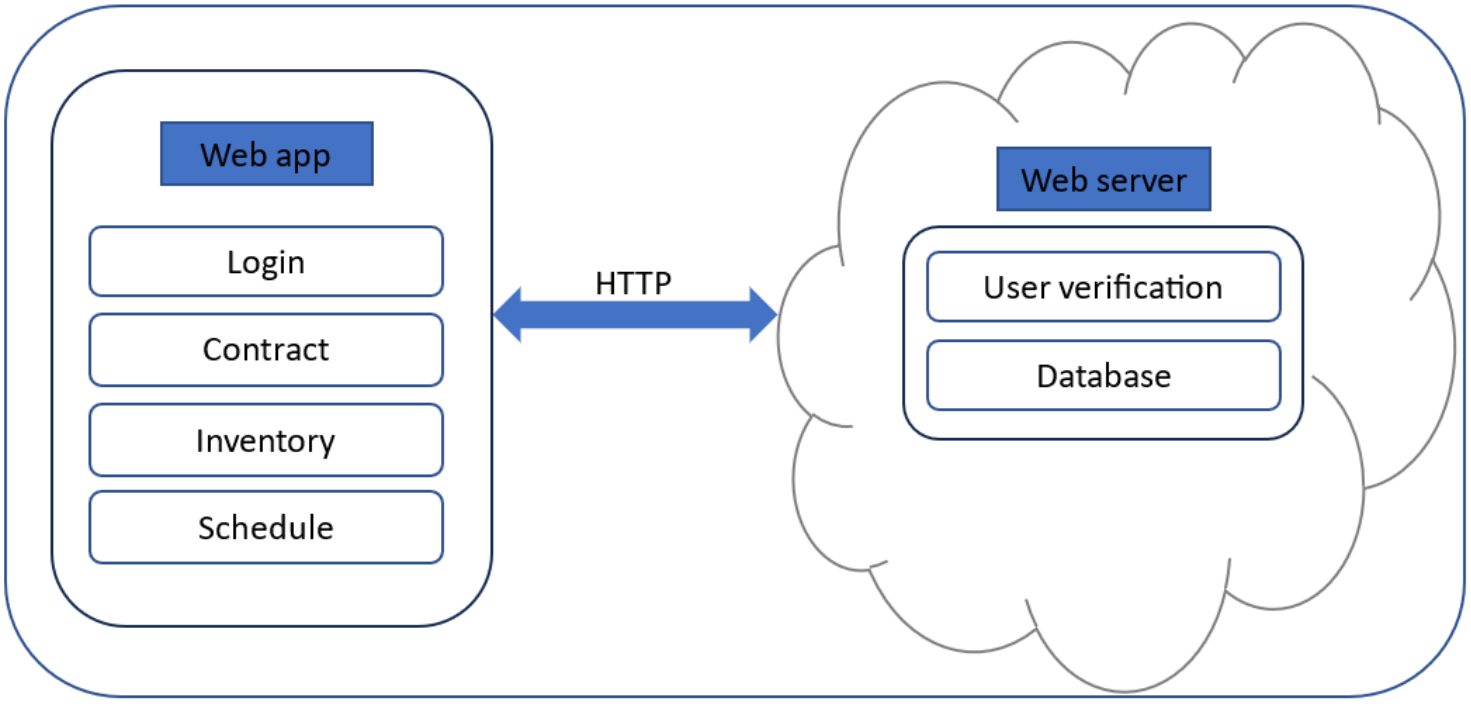


Figure 1: Architecture overview

2.3. Key Features

In Phase 2, key feature was added and enhanced:

Separate Apartment Management in Contracts: Each apartment within the same building is now treated as a separate entity within the contract, allowing for better management of complex projects involving multiple apartments.

2.4. Technological Considerations

The selection of technologies was driven by considerations of efficiency, flexibility, and ease of use. Svelte and SvelteKit were chosen for their ability to build intuitive interfaces with minimal code, while TypeScript was used to ensure code reliability and reduce errors. These choices were made to create a robust, scalable system that meets the specific needs of the shelter construction company.

2.5. Flowchart and Process

The system operates through a well-defined process, with each step connected to the next. Below is an example of how the system manages the creation and tracking of a contract:

2.5.1. Creating a New Contract: A secretary creates a new contract in the system, adds details about the apartments within the building, and enters the client's information.

2.5.2. Inventory Management: The system checks the inventory needed to fulfill the contract. If there are shortages, the system alerts the inventory manager to place an order.

2.5.3. Approval and Payment: Once the client approves the contract and makes the payment, the system updates the contract status from "Pending" to "In Progress."

2.5.4. Installer Scheduling: The secretary builds a schedule for the installers, who can view and update their progress through the system.

2.5.5. Apartment Completion: The team leader mark apartments as “Complete” at the moment the team has done their work and the constructor approved the work.

2.5.6. Contract Completion: After the work is completed and the client is satisfied, the secretary marks the contract as "Completed" and archives all relevant information.

3. Development Process Description

The development process of the Shelter Construction Management system followed a structured approach, adhering to best practices in software engineering. Here is an overview of the key stages and methodologies used throughout the project.

3.1. Initial Planning and Requirements Gathering

- **Stakeholder Meetings:** The project began with a series of meetings with the primary stakeholders, including the CEO of the shelter construction company. These meetings were crucial in gathering detailed requirements and understanding the specific needs of the company.
- **Documentation:** Based on these meetings, we documented the functional and non-functional requirements, focusing on creating a system that is easy to use, efficient, and scalable.

3.2. System Design

- **Architecture Design:** We designed the system architecture to be modular, allowing for easy maintenance and future scalability. The architecture included a clear separation between the frontend, backend, and database layers.
- **User Interface Design:** Given the non-technical nature of many users, the UI was designed with simplicity and usability in mind. We used Svelte and SvelteKit to create an intuitive and responsive interface.

3.3. Technology Stack Selection

- **Frontend:** Svelte and SvelteKit were chosen for their simplicity and performance, allowing us to build a fast and responsive web application.
- **Backend:** The backend was developed using SvelteKit, which provided a seamless integration with the frontend and allowed for efficient data management.
- **Database:** Drizzle was selected as the database for its lightweight nature and ease of integration with the system, ensuring secure and reliable data storage.

3.4. Agile Development and Sprints

- **Sprint Planning:** We adopted an Agile methodology, breaking the development into sprints. Each sprint focused on a specific set of features or components, allowing us to iterate quickly and respond to changes in requirements.
- **Daily Standups:** Regular standup meetings were held to discuss progress, identify any blockers, and adjust the sprint plan as needed.
- **Sprint Reviews:** At the end of each sprint, we conducted reviews to demonstrate the completed work to stakeholders and gather feedback for future sprints.

3.5. Challenges and Solutions

- **Learning Curve:** One of the significant challenges was learning a new language and framework (Svelte and SvelteKit). To address this, the team dedicated time to learning and experimenting with the technology before fully integrating it into the project.
- **Technical Constraints:** We encountered issues related to college-imposed restrictions on accessing certain websites, such as the Drizzle database. To overcome this, we adjusted our development environment and made use of local resources where possible.
- **Code Refactoring:** As we became more familiar with the technologies, we identified opportunities for refactoring and optimizing the code. This led to a more efficient and maintainable system.

3.6. Testing and Quality Assurance

- **Unit Testing:** We implemented comprehensive unit tests to ensure that individual components functioned correctly.
- **Integration Testing:** Integration tests were conducted to verify that different components of the system worked together as expected.
- **User Acceptance Testing (UAT):** Towards the end of the development, UAT was performed with a small group of users to ensure the system met their needs and was user-friendly.

3.7. Final Deployment and Documentation

- **Deployment:** The final version of the system was deployed on the company's servers, with all necessary configurations and security measures in place.

- **Documentation:** We prepared detailed documentation for both users and developers. The user guide provided clear instructions for operating the system, while the developer documentation included technical details necessary for future maintenance and updates.

3.8. Retrospective and Lessons Learned

- **Retrospective Meetings:** After the completion of the project, we held retrospective meetings to discuss what went well and what could be improved. The importance of clear communication and the benefits of Agile methodology were key takeaways.
- **Continuous Improvement:** We identified areas for potential improvements and created a backlog for future development phases, ensuring that the system could evolve with the company's needs.

3.9. Team Collaboration

The Shelter Construction Management system was developed by a team of two, with each team member responsible for different parts of the project. This division of labor allowed the team to focus on specific aspects of the system, ensuring that each component was developed with attention to detail and expertise.

4. Results and Conclusions

4.1. Project Achievements

The Shelter Construction Management system has been successfully developed, achieving all the primary goals set at the beginning of the project. The system now effectively digitizes the shelter construction company's workflow, which was previously managed manually through paper-based methods. The key functionalities include:

- **Contract Management:** The system allows for the creation and management of construction contracts, with the added ability to distinguish between different apartments within the same building. This enhancement provides a more detailed and accurate way of handling complex construction projects.
- **Inventory Tracking:** The system includes a feature for tracking inventory levels, ensuring that materials are available when needed, and automatically

alerting the inventory manager when stock levels are low. Additionally, the system's inventory tracking feature ensures that materials are closely monitored by keeping a record of the number of items that go out, which helps in preventing theft and unauthorized use of resources. By knowing exactly who was responsible for handling inventory, the quantities involved, and which employees were engaged in a project when issues arise, the company can quickly address any discrepancies and improve overall accountability.

- **Installer Scheduling:** The system allows secretaries to build and manage schedules for installers, streamlining the assignment of tasks and tracking progress.
- **User Role Management:** The system includes role-based access control, allowing the CEO to manage employee roles and permissions within the system.
- **History Tracking:** One of the critical features of the system is its ability to track the history of all contracts and inventory transactions. This functionality allows the company to maintain a detailed record of who worked on specific contracts and when, providing a valuable audit trail.

4.2. Comparison with Initial Goals

The final system meets and, in some cases, exceeds the objectives outlined at the start of the project. The transition from manual to digital processes has been successfully implemented, with the system providing a simplified, intuitive interface tailored to the specific needs of the company's employees. The system’s performance under load and its ease of use have been tested and confirmed to meet the company’s requirements.

4.3. User Feedback and Satisfaction

Feedback from the project supervisor led to the implementation of the apartment distinction feature within contracts, which significantly improved the system’s ability to handle complex projects. Overall, user feedback has been positive, particularly regarding the system’s usability and its ability to reduce errors and manual effort.

4.4. Challenges Overcome

Throughout the project, the team faced several challenges, including:

- **Technical Learning Curve:** Adapting to the new technologies (Svelte, SvelteKit, and Drizzle) required significant learning. This was managed by dedicating time to learning and experimenting with the technology before full integration.
- **Environmental Constraints:** College-imposed restrictions on accessing certain websites, like the Drizzle database, posed a challenge. The team overcame this by using local resources and adjusting the development environment accordingly.
- Refactoring for Optimization: As the team gained more familiarity with the technologies, opportunities for code refactoring were identified and implemented, resulting in a more efficient and maintainable system.

4.5. Overall Impact

The Shelter Construction Management system is expected to have a significant impact on the company's operations. By moving to a digital platform, the company can now manage contracts, inventory, and scheduling more efficiently, with reduced errors and improved oversight. The system's design also allows for quick training of employees, making it easier to integrate into the company's workflow.

4.6. Future Work

Looking forward, there are several potential areas for further development:

- **Advanced Reporting:** Implementing more detailed reporting features could provide the company with insights into project performance and resource utilization.
- **Mobile Interface:** Developing a mobile-friendly interface could enhance the system's accessibility for field workers.
- **Integration with External Systems:** Future versions of the system could include integration with other software, such as accounting or CRM systems, to provide a more comprehensive management tool.
- **Automated Email Sending:** Implement an automated email system that sends messages to customers and other recipients using a consistent predefined pattern, ensuring uniform communication.

5. User Guide:

5.1. Introduction

The Shelter Construction Management System is designed to streamline and automate the management of shelter construction projects. This guide will help users navigate the system's features, including contract management, inventory tracking, installer scheduling, and user role management. The system is tailored for various user roles, ensuring that each user has access to the tools they need to perform their duties efficiently.

5.2. Installation and Access

System Access - Access the system by navigating to the designated URL through a web browser: https://shelter-construction-manager.vercel.app

- Users need to log in using their assigned credentials (username and password).

Login Procedure -

1. Open your web browser and navigate to the system's URL.
2. Enter your username and password on the login screen.
3. Click "**Log in**" to access the system.

5.3. Using the System

5.3.1. Contract Management

Creating a New Contract:

1. Click "**Create New Contract.**"
2. Fill in the contract details, including client information and contract terms.
3. For multi-apartment buildings, add each apartment as a separate entry within the contract.
4. Choose due date for each floor.
5. Click "Create Contract" to store the contract in the system.

Editing Contract Status:

1. Click "**Update contract status**".
2. Search the contract you wish to update in the sorted list by ID.
3. Update to the next step in the process by clicking the "**Mark as In Progress\Complete** " button.

Viewing Contracts:

1. Navigate to the "**View contracts**" from the main menu.

2. By fill the fields(one or more): ID, Signing Date, Due Date, Status, and Type you can filter the contract list and find easily the desired contract.

5.3.2. Inventory Tracking:

Checking Inventory Levels:

1. Navigate to the "**View and update items**" from the main menu.
2. View the current stock levels for all materials.
3. If any item is out of stocks, an alert will be displayed near item name.

Responding to shortage:

1. Navigate to the "**View and handle shortages**" from the main menu.
2. View the current stock shortages for all Contracts.
3. For Each shortage will be a "**Mark as ordered**" or "**Mark as complete**" button, in case the item is ordered or arrived respectively.

5.3.3. Team management:

Creating new team:

1. Navigate to the "**View teams**" from the main menu.
2. Click on Hyper button "**Create a new team**".
3. Input team name, and team members names.
4. Click "**Create team**" to save the new team.

Update apartment in contract:

1. Navigate to the "**View teams**" from the main menu.
2. Click on the name of the desired team.
3. Search in Team Schedule section the desire schedule.
4. Click on Hyper button "**View apartments**".
5. Click on "**Mark as complete**" button.

5.3.4. Update team

Add installer to team:

1. Navigate to the "**View teams**" from the main menu.
2. Click on the name of the desired team.
3. Click "**Add installer**" to add new installer.
4. Click on "**Update team**" button to save changes.

Delete installer from team:

1. Navigate to the "**View teams**" from the main menu.
2. Click on the name of the desired team.
3. Press the red X to delete team member.
4. Click "Confirm" to delete.

Delete team:

1. Navigate to the "**View teams**" from the main menu.
2. Click on the name of the desired team.
3. Click "**Delete**" to delete team.
4. Click "**Confirm**" confirm delete.

5.3.5. Schedule management:

Create schedule:

1. Navigate to the "**Update daily schedule**" from the main menu.
2. Choose team from select box and click "**Get Schedule**".
3. Assign tasks to team, specifying the contract, apartments, date, etc.
4. Click "**Add New Schedule**" to finalize the schedule.

Edit schedule:

1. Navigate to the "**Update daily schedule**" from the main menu.
2. Choose team from select box and click "**Get Schedule**".
3. Search the desire schedule in the list of the schedules.
4. Click "**Edit**" on the wanted schedule.
5. Click "**Save changes**" to update.

Delete schedule:

1. Navigate to the "**Update daily schedule**" from the main menu.
2. Choose team from select box and click "**Get Schedule**".
3. Search the schedule in the list of the schedules.
4. Click "**Delete**" on the desired schedule to update.
5. Click "**Confirm**" confirm delete.

5.3.6. User Management

Adding a New User:

1. Navigate to the "**View and edit employee data**" from the main menu.
2. Click "**Add new employee**".
3. Enter the user's details and assign a role (e.g., Secretary, Installer, CEO).
4. Click "**Save**" to create the user account.

Editing User Roles:

1. Navigate to the "**View and edit employee data**" from the main menu.
2. Search for the user you wish to edit and click "**Edit**" near the user.
3. Modify their role or access permissions.
4. Click "**Save**" to update the user's settings.

5.4. User Role Permissions

CEO: Has access to **all** system features, including viewing and editing employees, contracts, schedules, and inventory.
Secretary: Can create and update contracts, manage schedules, and update contract statuses.
Team Lead: Focuses on viewing and updating the daily schedule and teams.
Inventory Manager: Responsible for viewing and updating inventory, including handling shortages.

5.5. Troubleshooting Common Issues

Login Problems:
Issue: Unable to log in.
Solution: Ensure your username and password are correct. If you've forgotten your password, please contact with the CEO to reset the password.

System Performance Issues:
Issue: The system is slow or unresponsive.
Solution: Check your internet connection. If the problem persists, contact the system administrator.

Error Messages:
Issue: Encountering an unexpected error message.
Solution: Take note of the error code and contact support for assistance.

5.6. Summary

The Shelter Construction Management System provides an efficient, user-friendly platform for managing construction projects, inventory, and scheduling. Should you require further assistance, please contact our support team for help

6. Maintenance Guide

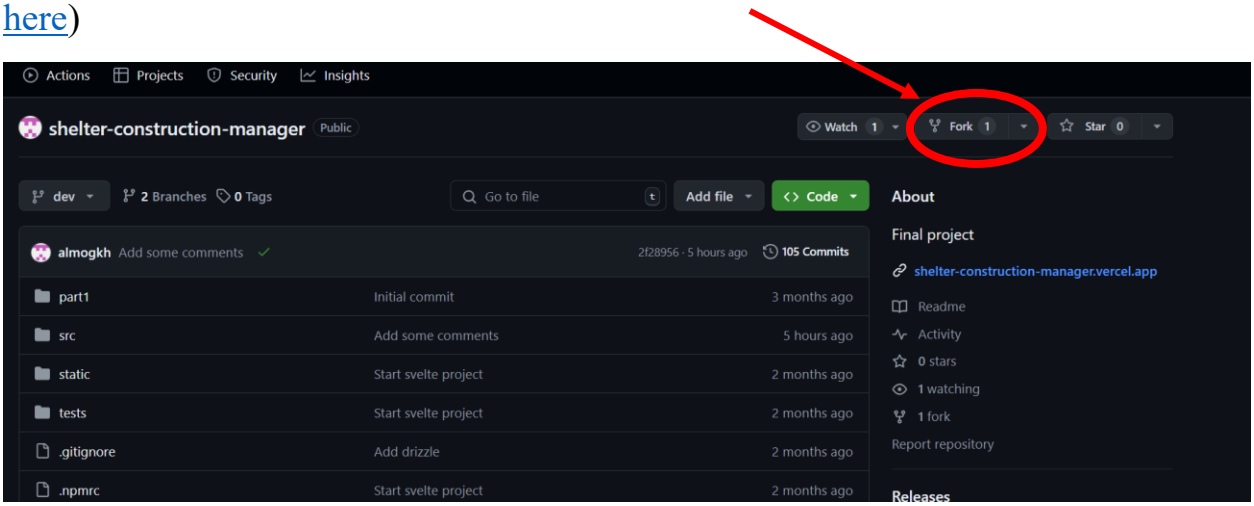
6.1. Operating Environment and Installation Instructions

The Shelter Construction Management System is designed to run primarily in a Node.js environment and is deployed via Vercel. No special hardware or software is required beyond what is available in the project's GitHub repository.

Installation Steps:

Fork the GitHub Repository:

Start by forking the repository to your own GitHub account. (Link for repository here)



Vercel Deployment: Point Vercel at your forked repository, and Vercel will automatically deploy the system.

Database Setup:

Set up a PostgreSQL database either within Vercel or using an external service.

Add an environment variable named `POSTGRES_URL` to your Vercel project, setting its value to the PostgreSQL database URL.

Local Development:

Clone the GitHub Repository: Clone the repository to your local machine using `**git clone**.`

Install Dependencies: Run `**npm i**` to install all necessary dependencies from the package.json file.

Environment Setup: Create a `**.env**` file in the root directory with the following variables:

- `**POSTGRES_URL**`: Set this to your local development database URL.
- `**NODE_DB**`: Assign any non-empty value to this variable to indicate that the app is running in a local development environment.

Database Schema: Run `**npx drizzle-kit push**` to apply the database schema to your PostgreSQL database.

Run the App: Use `**npm run dev**` to start the local development server.

Updating the System

Hot-Reload in Development: During local development, any code changes will automatically hot-reload, so there's no need to restart the server manually.

Deploying Updates to Vercel: After making changes to the code, push your changes to GitHub. Vercel will automatically detect the new changes, build the updated version of the app, and deploy it.

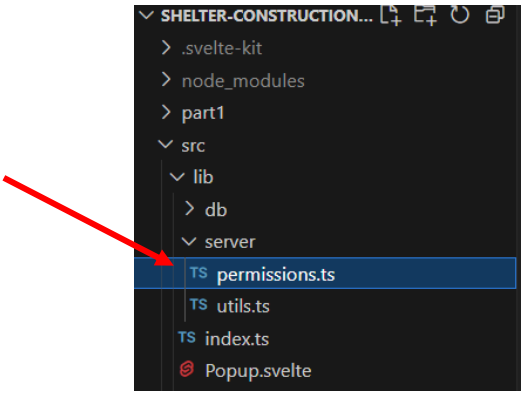
6.2. Code Modifications

The system is built using the following key technologies:

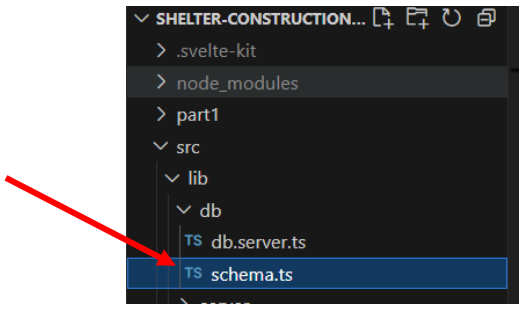
- **Frameworks:** Svelte and SvelteKit are used as the primary frameworks for building the frontend and handling backend logic.
- **ORM:** DrizzleORM is used for managing database operations.
- **Styling:** TailwindCSS is used for styling the user interface.

Key Files:

Permissions: Modify the `**permissions.ts**` file to manage access to new routes and buttons on the homepage. This file controls which user roles can access specific parts of the system.



Database Schema: The `**schema.ts**` file defines the database schema. Any changes to the database structure should be made here, followed by running `**npx drizzle-kit push**` to apply the new schema.



Help Text: To add help text on any page, include a `<Help>` component with the desired text, e.g., `<Help>Help text</Help>`

6.3. Troubleshooting and Common Issues

Common Issues:

Hot-Reload Problems: If hot-reloading in the local development environment fails, try clearing the cache or restarting the development server.

Deployment Issues: Ensure that all necessary environment variables are correctly set in Vercel. Missing or incorrect variables can cause deployment failures.

Error Logs: Access error logs through Vercel's dashboard under the “Logs” section for each deployment. These logs provide detailed information about errors that occur during deployment and runtime.