HarvestHub: Technical Approach and Lessons Learned

Introduction

"Nothing beats a firsthand experience" by Hans Rosling

HarvestHub, a unique community allotment management system, was designed to streamline the allocation and management of garden plots for community members and the public. This project employs a comprehensive full-stack web development approach guided by modern web technologies and best practices. Our journey in developing HarvestHub was a valuable learning experience, shedding light on the intricacies of application development. This document delves into our technical approach, the lessons we learned, and the potential areas for further improvement.

Technical Approach

Frontend Development: HarvestHub utilises `HTML`, `CSS`, and `JavaScript (JS)` tech stacks as a basic structure. `HTML` is the fundamental structure, `CSS` is used for styling, and `JS` adds interactivity, enabling dynamic content updates. *Responsive Design:* The `Bootstrap` Framework is used here. It is a framework that provides responsive UI using the `Bootstrap` grid system and a pleasing user interface. The application will look equally great on all devices because of its pre-designed components, such as navigation bars and buttons. It also maintains consistency across the application. *Server-Side Development:* `Node.js`, combined with the `Express.js` framework, allowed the creation of middleware to handle `HTTP` requests and responses efficiently because `Express.js` also includes `body-parser` libraries. Routes were defined for different functionalities. Each route is linked to a specific controller function responsible for processing the request. *Business Logic:* In HarvestHub, business logic is handled by controller functions. They fetch data from the database, process it, and render the appropriate view. For example, a controller function confirms an assignment, updates the database accordingly, and deletes the corresponding waiting record once the assignment is confirmed. *Server-Side Rendering:* `EJS` was adopted for server-side rendering of dynamic web pages, so the main content of each page is rendered dynamically based on user choices and actions. Example: The `user.ejs` template displays content for the user dashboard. First, `Express.js` routes fetch necessary data from the database and render the `EJS` template with this data. *Modular Page Design:* Modularity with `EJS` templates promoted a modular page design. This modular approach permits easier maintenance and provides a consistent look and feel across the application, making reusable components. Example: `header.ejs` and `footer.ejs` were created and included on the main page to maintain uniformity. *Database Utilisation:* `PostgreSQL` was chosen here. The database schema was designed to facilitate critical functionalities such as user and plot management and request tracking. Example: Controller functions execute `SQL` queries to

interact with the database, fetching and updating data as needed. *Database Setup:* `SQL` scripts were developed to create the necessary tables as designed, hoping to assist in setting up the database. *Environment Configuration:* The `dotenv` library handles environment variables securely because sensitive information, such as database credentials, remains protected and is not hard coded into the application code. This comprehensive technical approach ensures the success and reliability of HarvestHub, giving you the confidence that it meets the project goals.

Lessons Learned

Using Frameworks and Libraries: Frameworks and libraries speed up development and make the project easier to scale and maintain. They help avoid reinventing the wheel and focus on the project's unique features. It's essential to research and choose the right ones. *Modular Design:* Reusing code in controller functions and templates makes maintenance easier. This approach helps find errors quickly, allows easy updates, and ensures consistency throughout the application. *Managing Database Sequences:* During testing, we encountered duplicate key issues, highlighting the need for proper database sequence management. We fixed this by resetting the sequences to match the highest values in the tables. *Continuous Learning and Adaptation:* The project highlights the importance of continuous learning and adapting to feedback. Iterative development and regular feedback led to a more refined and user-friendly application. This commitment to continuous learning and improvement is what sets HarvestHub apart.

Conclusion

HarvestHub was developed with a clear focus on meeting the project objectives, serving as a successful exercise in modern web application development. It incorporates efficient frontend frameworks, powerful backend technologies, and robust database management practices. While HarvestHub has successfully met our initial project goals, there are still several technical improvements that we could work on. For instance, it features an automated system that identifies when any plot has completed the agreed period (e.g., after 12 months). In terms of security, user authentication is robust, employing best practices in password hashing and session management.

The lessons gleaned from this project serve as a roadmap for future enhancements. The development of HarvestHub has underscored the importance of continuous learning and adaptation in the application development process. Regularly gathering and integrating feedback is pivotal in refining the system from conception to implementation. HarvestHub is a testament to the value of ongoing learning and adaptation to add value to its users as a reliable and maintainable application.