# Designing and Developing Shukria Meat: A Comprehensive Retail Web Platform

*by*

Zunaid Radin Uddoula
22301778

An internship report submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
October 2025

# Declaration

It is hereby declared that

1. The internship report submitted is my/our own original work while completing a degree at BRAC University.
2. The report does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The report does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. I have acknowledged all main sources of help

**Student's Full Name & Signature:**

**Zunaid Radin Uddoula**
22301778

# Approval

The thesis/project titled "Designing and Developing Shukria Meat: A Comprehensive Retail Web Platform" submitted by

1. Zunaid Radin Uddoula (22301778S)

Of Summer, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on October 9, 2025.

**Examining Committee:**

Supervisor:
(Member)

Dr. Md. Golam Rabiul Alam

Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam

Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi

Associate Professor
Department of Computer Science and Engineering
Brac University

# Abstract

This report documents a six-month Software Developer Internship at Skarbol.Tech (February 2–July 31), highlighting the design and development of "Shukria Meat," a modern web platform for the meat retail domain. The platform enables customers to browse categorized products, discover outlet locations, read frequently asked questions, and subscribe to newsletters. The system was implemented using Next.js 14 (App Router) with TypeScript for the frontend, Drizzle ORM with MySQL for the data layer, CSS Modules for modular styling, and a custom authentication context for session management. In parallel, I contributed to additional initiatives including an AI-powered virtual assistant prototype, a portable water filtration concept, and several small applications (Weather App, IP Tracker, Login, To–Do, JavaScript games). The internship provided hands-on exposure to agile practices, full-stack development, API testing, collaborative version control, and industry-standard security practices. The experience meaningfully strengthened my technical competence and professional skills.

**Keywords:** Internship, React, Next.js, TypeScript, Drizzle ORM, MySQL, CSS Modules, Authentication, API testing, Agile.

# Table of Contents

# Chapter 1

# Introduction

## 1.1 About Internship

The importance of internships is to facilitate the transition between theoretical learning in the academics to the real-life requirements of the working environment. When in the context of the Skarbol.Tech placement, I found myself exposed to a collaborative, engineering-focused culture where I was supposed to design, build, test, and refine software applications in a way that was within the industry standards. The flagship web application that I worked on was the Shukaria Meat, a modern retail application of meat products display and management, and I also worked on an AI-assistant prototype and a number of integrated training projects, including a Weather App, an IP Tracker, a login application, and a to-do application, and lightweight JavaScript games.
In these projects, I worked with front-end (React, Next.js, TypeScript, CSS Modules), and back-end/data issues (route handlers, Drizzle ORM with MySQL). I used Postman to test the API and was working in a framework based on Agile principles where short feedback loops, peer-reviews, and constant incremental improvement were favored.

## 1.2 Report Scope

This report records what I did, what I got to learn and why it was important. It particularly focuses on the Shukria Meat platform, as the main deliverable of my internship, and then it summarizes supporting work (AI assistant, water filtration concept, and skill building mini projects). I explain the tools, practices, and decisions that informed the software; reflect on the professional skills that I acquired; and provide a list of suggestions on how the platform could be expanded in the future. The rest of the document has a typical academic format: organization overview, training and development, specific activities and responsibilities, knowledge acquired, and a reflection as a conclusion.

## 1.3 Research Objectives

The internship provided a possibility to investigate practical research questions that were in accordance with my roles. The major goals were:

- Develop a retail web platform using modern and stylish full-stack tooling. Read about the way Next.js 14 (App router) with TypeScript can provide fast, sustainably performant user experience to a catalog-like site (products, categories, outlets, faqs, newsletter).
- Database design and type safe access to data. Applying a modern ORM system Drizzle ORM to MySQL to formalize schema, constraints and queries to analyse the effect type safety has on runtime errors. Assess safe user-friendly authentication.
- Evaluate secure, user-centric authentication. Implement and assess session handling and authorization patterns appropriate for public browsing plus admin-only actions.
- Practice Agile delivery. Note how sprints, backlogs and frequent review of the codebase provided an excellent workflow that propelled the momentum and allowed requirement changes without sacrificing quality.
- Survey adjacent work. Learnings on the AI-assistant interface (processing variable model output) and the ways AI model must be adopted in current web and ERP systems.

## 1.4 Internship Aim

I wanted to transform theoretical knowledge into practical contributions in a professional setting. I also wanted to enhance my front-end engineering, back-end integration, database management, security best practice, and collaborative development process and produce a full platform demonstrating capability in Shukria Meat. It was simple: I was going to transform theory into a workable practice. This involved writing readable code, writing modeling data, writing defensively validated inputs, and being part of the team processes; planning, code review, testing, and documentation. The capstone objective was to leave behind a deployable, coherent basis of Shukria Meat that would be able to sustain the addition of new features in the future, including cart/checkout and administration dashboard.

## 1.5 Methodology

The approach is a mix of practical project work with formal study of documentation, code reviews and team workshops that are also required team meetings and work with dev team. The delivery was based on agile planning and sprint execution and continuous testing. The strategy focused on evidence-based decision making, peer feedback and refinement.

### 1.5.1 Primary Data

The main source of this report was the direct result of the work that I created and the experiences that influenced this date. The codebase has been subjected to numerous commits, branches, and pull requests, which provided a history of the decisions, experimenting, and refinement made on the Shukria Meat platform. The

features were taken through a repeatable cycle, scoped work on the sprint board, a series of commits associated with the work, and a peer-reviewed pull request with design intent, trade-offs, and testing notes being taken seriously because there was a need to track of improvements. I maintained Postman collection and run histories of the major routes together with the source code. These did not just demonstrate successful results but also intentionally followed error paths and boundary conditions to use it later. Testing on screens and devices of different sizes created a screen and notes on the responsiveness and were observed on the details of the interaction (keyboard navigation and focus management).
This primary data was completed by team interactions. notes of the features, sprint reviews and ad-hoc pairing sessions taken recorded the rationale behind scope or implementation changes. In the case of performance or accessibility, I took snapshots of the next work and made brief checks of the Web Vitals to capture the impact of optimizing images or splitting the code or changing the layout. All this was arranged in a lightweight trail, task links in commit messages, the descriptions of pull requests citing acceptance criteria, and transient thoughts in our internal notes, so that the conclusions in this report could be linked to tangible objects instead of memory.

### 1.5.2 Secondary Data

Secondary information was made out of reputable sources and conventional practices that guided my decisions when implementing. To ensure the use of the core technologies, Next.js 14, TypeScript, Drizzle ORM, MySQL, and related tooling, I used official documentation to ensure proper usage patterns, suggested APIs, and current configuration recommendations.
When I was required to study about Architectural flows, I used well established engineering tutorials on such issues as server-side rendering techniques, route-handler architecture, schema design, and type-safe access to data, reference to these sources to defend decisions in design notes and in pull requests. Checklists and standards used in the industry were used to influence non-functional aspects: accessibility guidelines, performance guidelines (image formats, caching behavior, layout stability), and basic security standards that had to be observed during the development. Where more than one viable approach was available: such as deciding between ISR and pure dynamic rendering, or between other validation libraries, I compared the implications with the project goals and requirements in our internal documentation and chose the one best suited to maintainability and end-user experience. Through this, secondary sources can supplement the primary evidence as it provides a framework on which the work can be based on a generally accepted practice and leave in the history of such contributions.

## 1.6 Expectations and Outcomes

My initial assumption of the internship was that it would advocate me out of the classroom practice into the rhythm of the software delivery: collecting the requirements, translating them into small, cutable sections, discussing the code with other interns, and testing features against the business requirements. I have three specific objectives, namely: to become proficient with a production grade stack comprising of React, Next.js 14 App Router and TypeScript, Drizzle ORM on top of MySQL,

and modular CSS; to own at least one end user experience, design to test, and to internalize a rigorous workflow in Git, documentation and Agile practices. These objectives were tested in Shukria Meat platform. Through multiple sprints I would capture requirements in components and route handlers, model entities and constraints in Drizzle, and add authentication in such a way that public browsing and actions restricted to the administration were really distinct. During the process which I refactored interfaces to make them readable and tightened schema or api validation to make failures self-evident and recoverable I used Postman to explore edge cases and verify API behavior and the presence of errors. The results are in line with expectations and in some aspects they were even higher than the expectations. At the end of the project, I had a solid base that hosts the fundamental journeys, namely homepage, categorized product browsing, outlets information, FAQs, newsletter subscription and sign-in/sign-up, with a code that is modular, typed and readable by other individuals. I also got to learn to reason trade-offs (where server rendering is more important than perceived speed and vice versa, where client interactivity is more important than contentment-with-SEO), and I had to learn to transform reviewing feedback into real-life solutions instead of one-off solutions. Most significantly, perhaps, I have learned to complete work to the end: not just writing code, but also the tests to verify that the code is correct, documentation to clarify intent, and minor performance and accessibility touches that give a feature a finishing touch. The overall outcome is a deployable and extendable base on Shukria Meat that can be expanded on in the future to include cart/checkout, payments, and an admin dashboard.

## 1.7 Key Learnings

The article's main insights regarding the internship experience include: This encounter changed my perception about professional software. I came to know that quality is not something that is added later but it is a by-product of many such small, considered decisions: naming objects accurately, making components small, using types to spot errors early, writing easy-to-test interfaces. The experience I had working on an Agile methodology taught me that incremental delivery is a better choice than a massive rewrite, to maintain a clean and visible codebase and to use git pull requests as technical storytelling within performance that describes purpose as much as it presents code. Whenever requirements varied, as they always do, I observed how a typed codebase and well-defined limits allowed the system to flex without failure. Even more banal practices, such as recording assumptions and leaving breadcrumbs in commit messages, were rewarded when the team reviewed decisions weeks later. Lessons about humans were also very vital. Sessions and reviews revealed to me that feedback is a multiplier in case it is received early and responded with evidence, i.e. screenshots, test results, and brief diffs, instead of defensiveness. I learned to communicate indecisiveness and suggested experiments and identify risks before turning into blockers. The process of debugging became less of a trial and error experiment and more of a reproducible process: reproduce, isolate, guess, check and only after that the fix can be made permanent. Lastly, the development of a culturally visible retail environment helped me understand more deeply how performance, accessibility, security are all linked: quick pages imply trust, easy interaction expands the market, and cautious treatment of credentials

and inputs will ensure the safety of the users and the business. These are the habits rather than any particular feature that I bring along with me through the internship.

## 1.8 About Internship

The importance of internships is to facilitate the transition between theoretical learning in the academics to the real-life requirements of the working environment. When in the context of the Skarbol.Tech placement, I found myself exposed to a collaborative, engineering-focused culture where I was supposed to design, build, test, and refine software applications in a way that was within the industry standards. The flagship web application that I worked on was the Shukaria Meat, a modern retail application of meat products display and management, and I also worked on an AI-assistant prototype and a number of integrated training projects, including a Weather App, an IP Tracker, a login application, and a to-do application, and lightweight JavaScript games.
In these projects, I worked with front-end (React, Next.js, TypeScript, CSS Modules), and back-end/data issues (route handlers, Drizzle ORM with MySQL). I used Postman to test the API and was working in a framework based on Agile principles where short feedback loops, peer-reviews, and constant incremental improvement were favored.

## 1.9 Report Scope

This report records what I did, what I got to learn and why it was important. It particularly focuses on the Shukria Meat platform, as the main deliverable of my internship, and then it summarizes supporting work (AI assistant, water filtration concept, and skill building mini projects). I explain the tools, practices, and decisions that informed the software; reflect on the professional skills that I acquired; and provide a list of suggestions on how the platform could be expanded in the future. The rest of the document has a typical academic format: organization overview, training and development, specific activities and responsibilities, knowledge acquired, and a reflection as a conclusion.

## 1.10 Research Objectives

The internship provided a possibility to investigate practical research questions that were in accordance with my roles. The major goals were:

[leftmargin=*,itemsep=2pt]

- Develop a retail web platform using modern and stylish full-stack tooling. Read about the way Next.js 14 (App router) with TypeScript can provide fast, sustainably performant user experience to a catalog-like site (products, categories, outlets, faqs, newsletter).

- Database design and type safe access to data. Applying a modern ORM system Drizzle ORM to MySQL to formalize schema, constraints and queries

to analyse the effect type safety has on runtime errors. Assess safe user-friendly authentication.

- Evaluate secure, user-centric authentication. Implement and assess session handling and authorization patterns appropriate for public browsing plus admin-only actions.
- Practice Agile delivery. Note how sprints, backlogs and frequent review of the codebase provided an excellent workflow that propelled the momentum and allowed requirement changes without sacrificing quality.
- Survey adjacent work. Learnings on the AI-assistant interface (processing variable model output) and the ways AI model must be adopted in current web and ERP systems.

## 1.11 Internship Aim

I wanted to transform theoretical knowledge into practical contributions in a professional setting. I also wanted to enhance my front-end engineering, back-end integration, database management, security best practice, and collaborative development process and produce a full platform demonstrating capability in Shukria Meat. It was simple: I was going to transform theory into a workable practice. This involved writing readable code, writing modeling data, writing defensively validated inputs, and being part of the team processes; planning, code review, testing, and documentation. The capstone objective was to leave behind a deployable, coherent basis of Shukria Meat that would be able to sustain the addition of new features in the future, including cart/checkout and administration dashboard.

## 1.12 Methodology

The approach is a mix of practical project work with formal study of documentation, code reviews and team workshops that are also required team meetings and work with dev team. The delivery was based on agile planning and sprint execution and continuous testing. The strategy focused on evidence-based decision making, peer feedback and refinement.

### 1.12.1 Primary Data

The main source of this report was the direct result of the work that I created and the experiences that influenced this date. The codebase has been subjected to numerous commits, branches, and pull requests, which provided a history of the decisions, experimenting, and refinement made on the Shukria Meat platform. The features were taken through a repeatable cycle, scoped work on the sprint board, a series of commits associated with the work, and a peer-reviewed pull request with design intent, trade-offs, and testing notes being taken seriously because there was a need to track of improvements. I maintained Postman collection and run histories of the major routes together with the source code. These did not just demonstrate successful results but also intentionally followed error paths and boundary conditions

to use it later. Testing on screens and devices of different sizes created a screen and notes on the responsiveness and were observed on the details of the interaction (keyboard navigation and focus management).
This primary data was completed by team interactions. notes of the features, sprint reviews and ad-hoc pairing sessions taken recorded the rationale behind scope or implementation changes. In the case of performance or accessibility, I took snapshots of the next work and made brief checks of the Web Vitals to capture the impact of optimizing images or splitting the code or changing the layout. All this was arranged in a lightweight trail, task links in commit messages, the descriptions of pull requests citing acceptance criteria, and transient thoughts in our internal notes, so that the conclusions in this report could be linked to tangible objects instead of memory.

### 1.12.2 Secondary Data

Secondary information was made out of reputable sources and conventional practices that guided my decisions when implementing. To ensure the use of the core technologies, Next.js 14, TypeScript, Drizzle ORM, MySQL, and related tooling, I used official documentation to ensure proper usage patterns, suggested APIs, and current configuration recommendations.
When I was required to study about Architectural flows, I used well established engineering tutorials on such issues as server-side rendering techniques, route-handler architecture, schema design, and type-safe access to data, reference to these sources to defend decisions in design notes and in pull requests. Checklists and standards used in the industry were used to influence non-functional aspects: accessibility guidelines, performance guidelines (image formats, caching behavior, layout stability), and basic security standards that had to be observed during the development. Where more than one viable approach was available: such as deciding between ISR and pure dynamic rendering, or between other validation libraries, I compared the implications with the project goals and requirements in our internal documentation and chose the one best suited to maintainability and end-user experience. Through this, secondary sources can supplement the primary evidence as it provides a framework on which the work can be based on a generally accepted practice and leave in the history of such contributions.

## 1.13 Expectations and Outcomes

My initial assumption of the internship was that it would advocate me out of the classroom practice into the rhythm of the software delivery: collecting the requirements, translating them into small, cutable sections, discussing the code with other interns, and testing features against the business requirements. I have three specific objectives, namely: to become proficient with a production grade stack comprising of React, Next.js 14 App Router and TypeScript, Drizzle ORM on top of MySQL, and modular CSS; to own at least one end user experience, design to test, and to internalize a rigorous workflow in Git, documentation and Agile practices. These objectives were tested in Shukria Meat platform. Through multiple sprints I would capture requirements in components and route handlers, model entities and constraints in Drizzle, and add authentication in such a way that public browsing and actions restricted to the administration were really distinct. During the process

which I refactored interfaces to make them readable and tightened schema or api validation to make failures self-evident and recoverable I used Postman to explore edge cases and verify API behavior and the presence of errors. The results are in line with expectations and in some aspects they were even higher than the expectations. At the end of the project, I had a solid base that hosts the fundamental journeys, namely homepage, categorized product browsing, outlets information, FAQs, newsletter subscription and sign-in/sign-up, with a code that is modular, typed and readable by other individuals. I also got to learn to reason trade-offs (where server rendering is more important than perceived speed and vice versa, where client interactivity is more important than contentment-with-SEO), and I had to learn to transform reviewing feedback into real-life solutions instead of one-off solutions. Most significantly, perhaps, I have learned to complete work to the end: not just writing code, but also the tests to verify that the code is correct, documentation to clarify intent, and minor performance and accessibility touches that give a feature a finishing touch. The overall outcome is a deployable and extendable base on Shukria Meat that can be expanded on in the future to include cart/checkout, payments, and an admin dashboard.

## 1.14 Key Learnings

The article's main insights regarding the internship experience include: This encounter changed my perception about professional software. I came to know that quality is not something that is added later but it is a by-product of many such small, considered decisions: naming objects accurately, making components small, using types to spot errors early, writing easy-to-test interfaces. The experience I had working on an Agile methodology taught me that incremental delivery is a better choice than a massive rewrite, to maintain a clean and visible codebase and to use git pull requests as technical storytelling within performance that describes purpose as much as it presents code. Whenever requirements varied, as they always do, I observed how a typed codebase and well-defined limits allowed the system to flex without failure. Even more banal practices, such as recording assumptions and leaving breadcrumbs in commit messages, were rewarded when the team reviewed decisions weeks later. Lessons about humans were also very vital. Sessions and reviews revealed to me that feedback is a multiplier in case it is received early and responded with evidence, i.e. screenshots, test results, and brief diffs, instead of defensiveness. I learned to communicate indecisiveness and suggested experiments and identify risks before turning into blockers. The process of debugging became less of a trial and error experiment and more of a reproducible process: reproduce, isolate, guess, check and only after that the fix can be made permanent. Lastly, the development of a culturally visible retail environment helped me understand more deeply how performance, accessibility, security are all linked: quick pages imply trust, easy interaction expands the market, and cautious treatment of credentials and inputs will ensure the safety of the users and the business. These are the habits rather than any particular feature that I bring along with me through the internship.

# Chapter 2

# Company Profile

## 2.1 Company Snapshot

Skarbol.Tech is a software company which transforms ideas of customers into trusted web applications. The team develops the product, codes it, tests it and subsequently enhances it according to the feedback. They want not to complete a project but to be able to maintain clean code and documentation to ensure that the software could be easily updated in the future. I was treated like a full-time member of the team: as an intern, I had to work on the tasks, open the pull request, and get the feedback. This was to assist me in realizing how professional software is constructed on a daily basis.

## 2.2 Vision and Mission

The vision of Skarbol.Tech is to build software that people trust and teams enjoy maintaining. The mission is to ship useful features without sacrificing speed, security, or accessibility. In practice this means the team prefers tools that are stable and well-documented. They also try to improve the user experience a little with every release—pages should load quickly, screens should be readable on mobile and desktop, and forms should guide users clearly if something goes wrong.

## 2.3 Operating Model

The organization of work is in short sprints in which the progress is seen. A feature begins as a mini task that has a definite end product. The developer will code it locally in a different branch that he will test locally after which he opens a pull request. The changes are reviewed by the teammates and only then the code is merged with some improvements proposed. This method maintains small size of changes and reduces the chances of bugs. When there is a change in priorities, the team is able to change very fast since every piece of work is independent and well defined. Products and Services section covers all the products and services the company offers. The primary services provided by Skarbol.Tech are the creation of business-specific web platforms. These are platforms and dashboards on which users can log in and view information, as well as take actions such as browsing items or filling forms. The company also links together the existing systems, such as

linking data between a web site and customer database, such that the tools used by the client collaborate effectively. They embed AI capabilities within straightforward, secure interfaces in other projects in order to allow users to benefit by model outputs without being confused. Shukria Meat in this case is a retail/catalog retail platform. It allows visitors to browse meat products by category, locate stores outlets, visit frequently asked questions and subscribe to a newsletter. It has been designed to be simple on purpose to ensure that the business can easily add ordering, payments, and admin pages without the need to build everything again.

## 2.4 Products and Services

Skarbol.Tech mainly builds custom web platforms for businesses. These are sites and dashboards where users can log in, see information, and perform actions like browsing items or submitting forms. The company also connects existing systems—for example, syncing data between a website and a customer database—so that the client's tools work together smoothly. In some projects, they place AI features inside simple, safe interfaces so users can get value from model outputs without confusion. In this context, Shukria Meat is a retail/catalog platform. It lets visitors explore meat products by category, find store outlets, read answers in an FAQ, and sign up for a newsletter. The design is intentionally simple so the business can later add ordering, payments, and admin pages without rebuilding everything.

## 2.5 Software Development Process

The delivery process is straightforward. To begin with, the team composes a brief scope outlining the problem, the key features, and the risks to monitor. Then they take a few initial choices concerning the way pages should be rendered, the flow of data and the safety of information held privately. Then the development occurs in small steps, namely: one builds a feature, adds a screenshots to the pull request, and runs tests. A round of performance (fast load, optimized images), accessibility (keyboard navigation, useful labels), and basic security (input validation, safe storage of keys) checks are also performed by the team before release. Any release that is made is marked and the notes are stored to ensure that the next sprint begins with an accurate plan.

## 2.6 Technology Stack and Tools

Standard stack is concerned with typed and modular code. On the frontend, the team employs React and Next.js 14 (App Router) and TypeScript. This assists in clean routing and debugging errors in development. CSS Modules or Tailwind are used to style and provide consistent and responsive layouts. Drizzle ORM with MySQL are used on the data side to define table and query in a type-safe manner. New Node.js route handlers and other server logic are powered by node.js. To work on a daily basis, the team resorts to the version control and code reviews via Git/GitHub, the APIs testing with Postman, and documentation, planning and quick communication with the help of Notion, Trello and Slack. Shukria Meat is

precisely using such a stack, which simplified the process of obtaining feedback on a review and reusing already proven patterns.

## 2.7 Project Portfolio

Retail or catalog stores such as Shukria Meat where the primary task is to classify products, categories and store data in such a way that the user has access to what they want within the shortest time possible. Others are assistant-type interfaces that encapsulate AI capabilities within it and are flawless such that results are useful and safe. Not only these a Cattle firm management project is also deployed named Cattlesync a couple of days ago which was a great milestone. Smaller concept projects also take the form of testing the ideas like basic software with device-level solutions like a portable water filtration monitor to ensure that the team learns quickly and makes decisions on what to build next. Although these projects are dissimilar, they are guided by the same habits: maintaining boundaries between system components, documenting decisions to allow others to follow, and pursuing small changes that are simple to test and may be reversed in case of necessity.

## 2.8 Corporate Culture and Values

The culture is associated with clarity, respect, and consistent improvement. Clarity should imply naming things well, commenting on them in brief where necessary and including a few lines of explanation on the decisions in pull requests. Respect is evident in the way reviews are conducted: the feedback is relevant and gentle and everyone knows that early identification of the problem saves time to the entire team. It is recommended that the intern should ask questions whenever they get stuck and to make notes of assumptions and propose small experiments. Learning is a life long process, no one is supposed to learn everything on the first day.

## 2.9 Team Collaboration Model

Collaboration and communication within the team: The team members will be involved in collaborative work alongside a facilitator to address any areas identified as needing enhancement.¡—human—¿section: Team Collaboration and Communication: The team members will be engaged in a collaborative work with facilitator to discuss any areas that were found lacking in. The majority of the cooperation is asynchronous and documented. Features and each pull request has a short summary, screenshots and testing notes so that the reviewer can have an overview of the change. Slack is the home of quick questions, demos and planning are reviewed weekly and longer write-ups, diagrams and decision logs are saved in Notion. This arrangement ensured that I could easily join Shukria Meat, get a feel of what the team required and deliver code that matched that of the rest of the project.

# Chapter 3

# Training and Development

## 3.1 Overview

The onboarding of my training in Skarbol.Tech was combined with practical work on the project. The initial weeks I spent on getting to know the team code conventions, branching strategy, review process and simple security guidelines regarding how to handle environment variables and credentials. In a very short period, this changed to reading and small tasks to making actual changes on Shukria Meat and sponsoring side projects. The concept was straightforward: study the workflow, use it on a small task, receive feedback and repeat on a little bit more complex tasks. This cycle assisted me in transforming theory into daily practices.

## 3.2 Work Environment

The working setting was cooperative and peaceful. Tasks were scoped out, deadlines were realistic and discussions remained result-oriented. In case I got stuck I would be able to ask on Slack or request a brief call, and someone would redirect me. Reviewing was done in a direct yet respectful way; the feedback on the review was clear, no one telling what to do, but why, making it actionable. This culture made it safe to ask questions early, to show work-in-progress, and to learn by improving the same piece of code across a few iterations.

## 3.3 Learning and Support

There were three channels of learning, which are documentation, pairing and review. To document, I used official instructions on Next.js, TypeScript, Drizzle ORM, and MySQL and made some small internal notes about the organization of the folders, name choice, and error management. To pair, a colleague would provide more context - why a route is designed this way, how a schema is developed, what trade-offs are between SSR and client rendering and I would apply the following slice with that context in mind. In the case of reviews, I got certain comments which are related to code lines and to the acceptance criteria of the user story. This closed the loop: read, try, and refine based on evidence rather than guesswork.

## 3.4 Tools and Platforms Used

On the frontend I learned to be comfortable developing using React, Next.js 14 (App Router) and TypeScript: writing components, sending typed props, deciding between server and client component, and connecting forms with validation. On the data side I modeled data in both Drizzle ORM and MySQL as well as formulated type-safe queries, dealt with common constraints including uniques, foreign keys, and simple enums. I had to practise route handlers in read/write operations, tested inputs at the limit and provided clear error messages which the UI could use. Personally, I have used Postman to test endpoints, Git/GitHub to use feature branches and pull requests, and Notion/Trello/Slack to organize the specifications, tasks, and to make a quick communication. I also did some work in Ubuntu, which enhanced my command line fluency as well as scripting and reading logs became more intuitive.

## 3.5 Mentorship and Guidance

My mentor would want me to give the problem statement, the options that I have taken and the rationale behind the course of action I took when I had proposed a change. This allowed me to get used to writing short design notes before I touched code and also to leave proof, screenshots, test results, performance checks, etc, when opening a pull request. I was also taught to consider the non-functional requirements at an initial stage: the basics of accessibility (labels, keyboards navigation), the hints of good performance (image optimization and layout stability), and the hints of rather easy security (input validation, safe config handling). Gradually I did not need so many hints as I began to pose the same questions to myself.

## 3.6 Personal Growth and Development

The main growth was in consistency. I got to know how to create tasks in small steps that can be revised frequently; have branches that stay on point; complete work end to end, i.e. code, tests where reasonable, notes to be reviewed, and follow-up refinements after feedback. Communication was also enhanced: my pull requests were more understandable, my commit messages were brief as they have a story, and my stand-up update was not about activity but risks and decisions. Technically, I have transitioned to be able to implement trade-offs but reason about trade-offs, particularly on the topics of rendering strategies, schema shape, and validation points. As of the end of the internship I was able to comfortably and without a lot of oversight bring a finite feature on Shukria Meat to a stable, mergeable change.

# Chapter 4

# Activities and Responsibilities

## 4.1 Web Development Projects

I took a few specialized projects to reinforce basic knowledge: a Weather App that used real-time API, an IP Tracker that used an IP address as input to locate the geolocation, a 2D car racing game to introduce basic form processing, and a To–Do application to use full CRUD. I also studied interactivity and animation utilizing little JavaScript games. Such practices strengthened the trends of managing states, dealing with errors, and responsive designs.

## 4.2 AI-based Projects

I worked on both building web features and adding UI support for the assistant. It displayed model outputs in a secure and reliable manner. I programmed the front end to handle a wide range of responses. These were messages from the model that were long, short, or broken. I added guards for rendering and fallbacks for fields that don't have a response. I made status indicators for actions that are waiting, retrying, or refusing. Users understood these and they didn't cause any problems. I made it so that you could navigate with the keyboard and labeled every control clearly. Focus stayed in control even after things were changed and updated. These details made it easier to use and less likely to make mistakes. I also helped test verification by using Postman request collections. I looked at proxy endpoints that send messages to the model. I tried both success and failure cases, like timeouts. I also checked how it deals with invalid or broken API inputs. Before/after screenshots were also taken, which I gave to the reviewers, to determine the consistency of the layout. The largest lesson was that the succeeding products of AI are only successful when the product environment is serene and predictable. Clarity in copying, constant loading, and truthful error messages reduced user anxiety and support tickets were significantly making it easier to users to make their query. These trends subsequently assisted me on Shukria Meat regarding planning FAQ conduct, newsletter confirmation flows, as well as defensive UI when network hiccups happen. Though this was a separate assistant project, the same habits I developed, which were communicating uncertainty and designing to be variable, were applied directly to customer-facing retail interfaces.

## 4.3 Meetings and Collaborations

Team coordination followed a simple rhythm that kept work moving without heavy process. Our meetings were brief and effective, we had sprint plannings to discuss priorities, review meetings to demonstrate increments and get feedback. The vast majority of the work was made asynchronous in GitHub: each change came as a pull request, and it had a brief description, screenshots, and a comment on the testing. The response of the reviewers was in the form of line commenting and structure level suggestions regarding the structure, naming, and performance. Slack was used to fill in fast questions and handoff. In the case of Shukria Meat, this cadence enabled me to gain traction within a short period of time and produce useful slices, such as homepage sections, category views, and authentication flows, without being out of context. I also got to know that I should write design Notes prior to the coding, that I should suggest alternatives whenever trade-offs were observed (such as server vs. client rendering) and that I needed to justify my decisions in a plain language to allow reviewers to critically assess them within seconds. In cases where feedback was received, I would make specific commits instead of big rewrites, which were easy to follow up.This balance of async and short, focused syncs limited rework and raised the quality bar for each merge.

## 4.4 Challenges and Solutions

There were three types of repeating challenges in the projects. The former was related to the uncertainty of the data: third-party APIs would occasionally send half-completed fields or modified response shapes. I have addressed this by adding narrow parsing functions, occasionally through hooks and components when developing in React and visible error boundaries that allow the UI errors to be corrected gracefully. The second was rendering choice in Next.js 14 making decisions about using SSR or server components to optimize SEO and initial speed or using client components to optimize rich interactivity. This was an attempt at being an experiment: find, compare Lighthouse traces, and write the justification in the pull request. The third one was authentication edge cases: expired tokens, race between redirects, and inconsistent post-refresh states. In this case, ambiguity was eliminated by explicit guards, idempotent handlers and centralized management of context. In addition to technical problems, I encountered a typical delivery problem of scope change. Beyond technical issues, I met the common delivery challenge of changing scope. The fix was to keep features small, to record assumptions in the user story, and to use feature flags when uncertainty was high. Onboarding to unfamiliar code was another friction point; the remedy was naming consistency, short module headers that explain intent, and a bias for predictable folder structures. Each solved problem left behind a pattern—validation utility, wrapper component, or checklist—that reduced the chance of repeat failures and sped up future work on Shukria Meat.

## 4.5 Personal Projects

Alongside team work, I set aside time to build two small Ubuntu desktop widgets—one for system uptime and one for weather—so I could practice real-world

scripting without relying on a full web framework. The uptime widget reads system information from Linux's proc filesystem and presents a clean, human-readable timer (days, hours, minutes). I wrote it as a simple script, added basic logging, and wrapped it with a lightweight launcher so it can auto-start on login. This forced me to get comfortable with the command line, file permissions, environment variables, and the small details that make a script feel reliable (for example, handling unexpected values and writing helpful error messages instead of failing silently). The weather widget pulls current conditions from a public API at a fixed interval. To avoid hitting rate limits or showing empty data, it keeps a small local cache and falls back to the "last known good" response if the network is slow or unavailable. I stored the API key outside the code in an environment file, added timeouts and retries to the request, and formatted the output so it stays readable on both a dark and light desktop theme. A scheduled task (cron/systemd timer) refreshes the widget every few minutes, and a minimal log file records successes and failures for quick troubleshooting. These micro-projects made me faster and more confident in Linux. I practiced process monitoring, scheduling, and basic observability; I wrote short readme notes so future me (or someone else) can install and tweak the widgets quickly; and I reused the same habits on Shukria Meat—validate inputs at the edge, fail gracefully with a clear message, and keep configuration out of the source code. Even though they are small, these tools helped turn good practices into everyday habits.

## 4.6 Technical Skills Acquired

By the middle of my internship, I had turned random bits of knowledge into daily routines. I learned how to use TypeScript and Next.js 14 on the front end. I wrote client and server components with careful state management. Forms included validation and made sure that the DOM was always open. CSS Modules helped me keep styles separate and avoid name conflicts. I used Drizzle ORM with MySQL to model data on the backend. I made sure that constraints were followed and wrote queries that were safe for types and easy to review. I wrote route handlers that had clear contracts and correct error codes. Postman made it easy to test both successful and unsuccessful paths. I made version control better by using small branches and focused commits. My pull requests told a short story with checks and pictures. I got better at using the terminal and scripts by working in Ubuntu. Every day, I set up environments and checked logs with ease. During this time, I also learned some important non-functional basics. These were semantic HTML and performance budgets for each feature. I was interested in how to use variables safely and how to navigate with the keyboard. Not only did each feature work in perfect situations, but they also felt complete. These habits got me ready to work harder on Shukria Meat.

## 4.7 Code Implementation Snapshots

The following screenshots highlight key parts of the Shukria Meat codebase used during the internship.

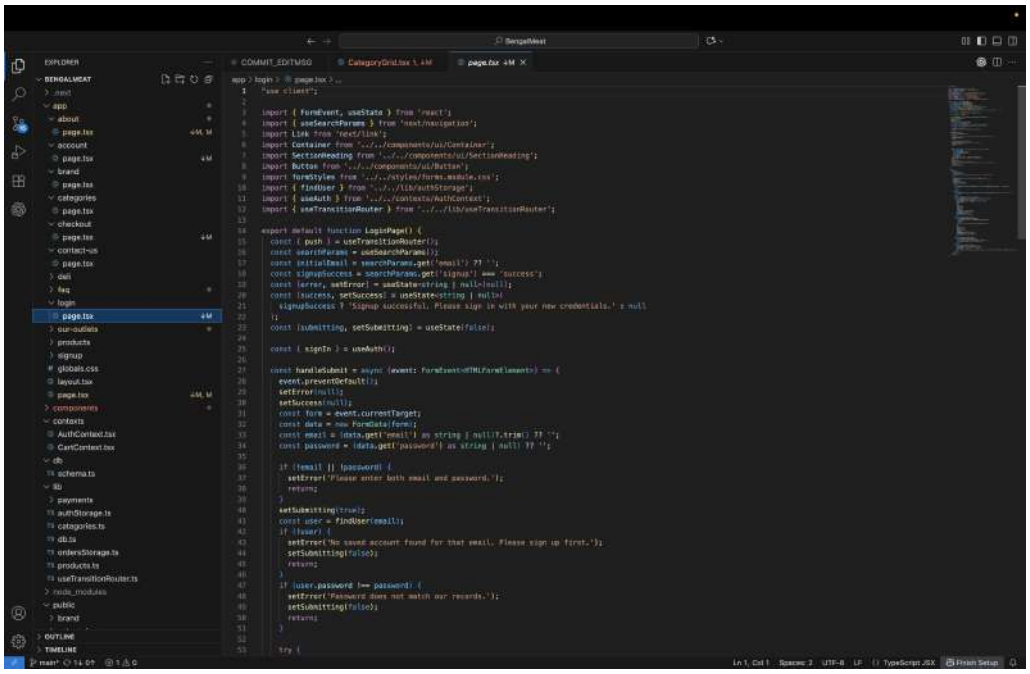

Figure 4.1: Login page logic

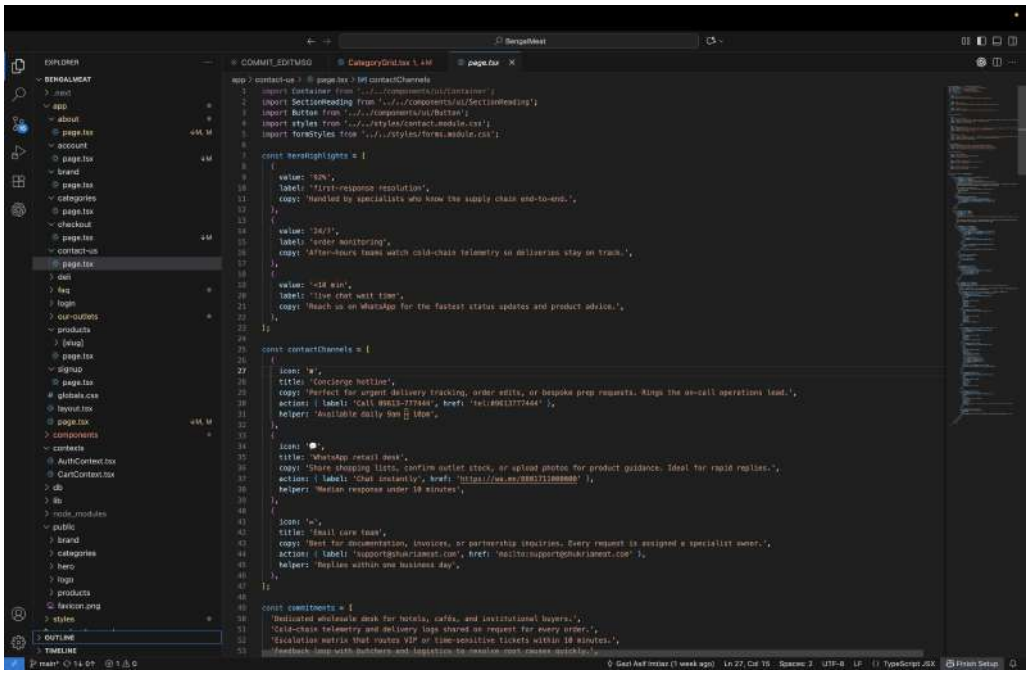

Figure 4.2: Contact page configuration

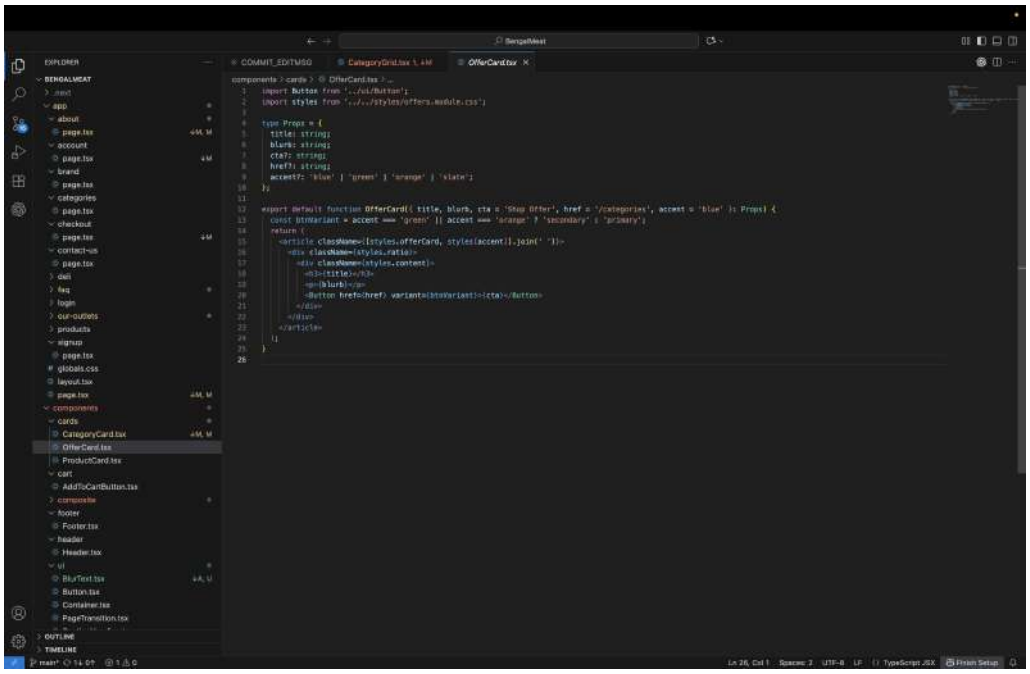

Figure 4.3: Reusable UI

## 4.8 Contributions to the Shukria Meat Project

My personal task was to convert several modules of user ends to merged code. In the case of Product Management I defined the types, mapped fields to database columns through Drizzle and constructed list/detail views whose loading and error states were predictable. Search and filtering were also made to be self-evident: category chips, search of names, and clear no results indicators. In the case of Category Management, I developed a light schema, and a navigational surface, which allows users to pivot in and out of product groups; on the administration side (scaffolded later), forms validate required fields and bust slug collisions. I added authentication to a custom context, safe redirects, and guarded actions, and carefully considered loss of the session on a refresh and error messages that did not leak internals and which assisted in achieving a higher security of the webview. The FAQ and newsletter were developed as dynamic content: entries and subscriptions will be maintained in route handlers and will be validated at the boundary. I wrote little utilities along the way e-mail checks, schema guards, response mappers, that help to localize complexity and make it easier to review. I confirmed endpoints in postman, confirmed responsive behavior on typical breakpoints, and corrected layout shifts that impacted the perceived performance. At the end of every module, there was a brief note on decisions (rendering mode, data boundaries, known limitations) to ensure that future contributors do not have to re-establish work-related decisions in the same area of work.

## 4.9 Planning & Work Flow

We carried out the project in 2 weeks sprints transforming high level requirements into visible, testable increments. Sprint 1 defined the skeleton that consisted of

repository structure, routing conventions, base layout and development of homepage or landing page. Sprint 2 added catalog features, which included product list, category navigation page, and route handlers with Drizzle integrations, and the first steps of authentication. Sprint 3 was more content-oriented and interactive: FAQ behavior, newsletter subscription, outlets information, and polish on blank screens. Sprint 4 was stabilization: accessibility, performance tuning (image formats, code splitting, layout stability), documentation. The process of work was the same: write the story with acceptance notes, write a brief design plan, make it happen in a feature branch, make a pull request with screenshots and Postman evidence, and work on review feedback. Status and design decisions and fast clarifications went to Trello and Notion respectively. This flow maintained the level of risk below and made any progress that was being made to be easily inspected and created a continuous flow of improvement that the user would instantly feel even before the greater features such as a checkout are added later.

## 4.10 Task Breakdown and Responsibility Allocation

Sharing of responsibilities was done to maintain clarity of ownership and maintain cooperation at boundaries. The composition and rendering of components, as well as their accessibility, were frontend-driven (I was among contributors). We settled on a design system, including the spacing, typography, and card patterns, such that pages were harmonious (when features were being shipped concurrently). The Drizzle ORM schema was constructed on MySQL by Backend/data contributors, route handlers were defined that responded to both read and write operations, and rules to ensure consistency in the data were written in code. The QA functionality arranged exploratory testing, put together Postman collections, and defects to the closure with clear reproduction procedures. Role to role handshakes occurred at the beginning of both stories: we defined payload shapes and error codes prior to creating UIs, which reduced software reuse. Cross-cutting changes (authentication redirects, shared utilities) were also encouraged through pairing, and we recorded our decisions in Notion so that they would not exist in chat threads. The small scale of tasks and explicitness of the interfaces brought by the team allowed it to provide visible functionality each sprint without disrupting previous work. It is this division of labor, along with rigorous reviews that have led to the easy extending of the codebase of Shukria Meat to cart, payments and administration in the next stage. section Tools and Platforms Used section provides information on the tools and platforms employed. There were Next.js 14, TypeScript, React Context API, CSS Modules, Drizzle ORM, MySQL, Node.js, Postman, Git and GitHub. The collaboration tools included Slack (communication), Notion, or Trello (planning, documentation).

## 4.11 Security and Authentication Handling

Security was considered first-class. Environment variables were used to store sensitive credentials; API routes authored input and authorised; secure tokens were used

on session management; and error messages shown to users did not leak internal context. Patterns of defensive coding minimized the risk of typical attacks. .

## 4.12 Version Control Using Git & GitHub

The work process was based on feature-branch work and pull requests containing descriptive commit messages. Readability, performance and testability were maintained through code reviews. Difficulties in merging were solved by just replaying the minimal changes and keeping the history clean.

## 4.13 Communication and Team Collaboration

PR discussions were asynchronous and daily updated with Slack, weekly with review meetings, and decisions recorded. Such cadence minimized the reworking and enhanced predictability of deliveries.

## 4.14 Ubuntu and System Tools

Sharing my experience on Ubuntu enhanced my competence and comprehension of developer processes between operating systems and command lines. I set up Node.js, Git and database clients, wrote automated procedures to accelerate the development of the environment and also tested the application in browsers.

## 4.15 Ubuntu Projects

The weather and uptime widgets supported viable scripting and system API. These were small, concrete projects that were useful in refining ideas that are transferred into software used in production: clear logging, stateful updates, and ergonomic presentation of the UI.

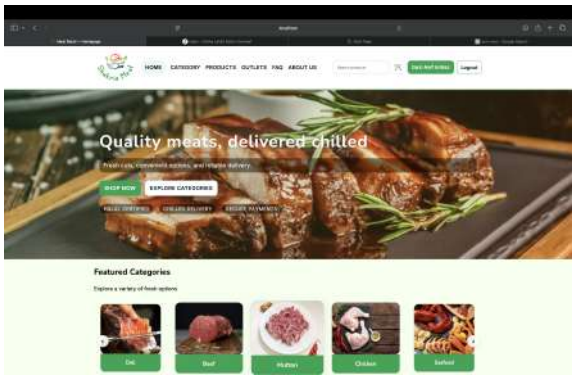


Figure 5.1:Shukria Meat homepage

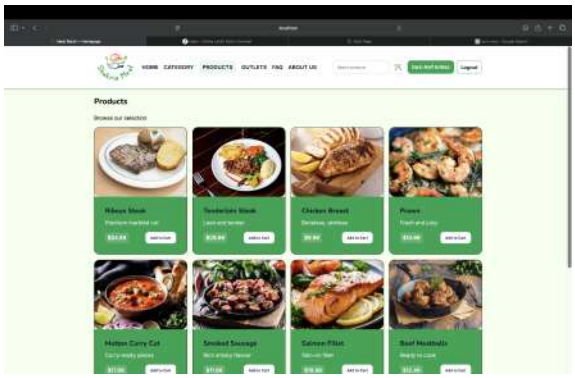


Figure 5.2: Product listing grid

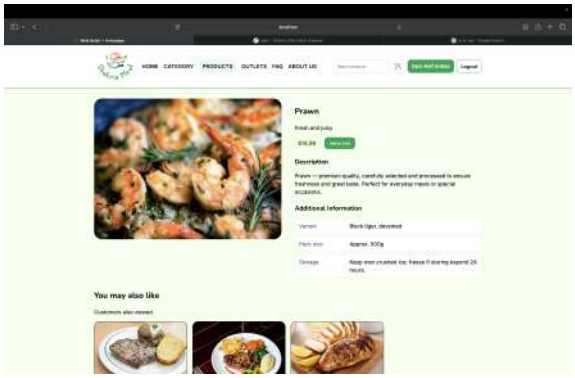


Figure 5.3: Product detail page

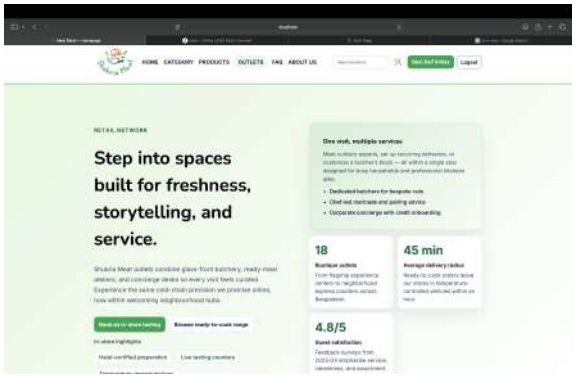


Figure 5.4:Outlets page

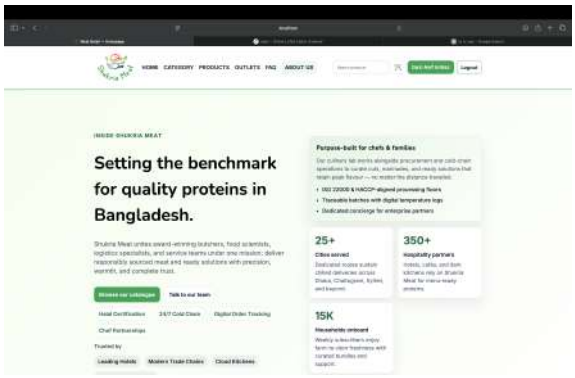


Figure 5.5:About Us


Figure 5.6:2D racing game

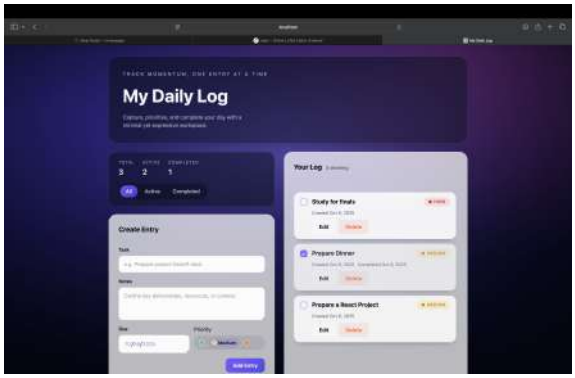


Figure 5.7:To-Do app

# Chapter 5

# Acquired Knowledge

## 5.1 Problem Solving

My problem solving process was made more systematic during the internship. I got to know that the first thing is to reproduce the issue in a consistent manner, and then isolate the tiniest of the surfaces on which it appears. It was then that I was able to derive a sound hypothesis and test one change at a time until it was proven or disproved. Such a process: reproduce, isolate, hypothesize, verify, prevented me in following symptoms. On Shukria Meat, it assisted with the problematic cases, including the expiring sessions when switching between pages and the category pages that were displayed differently with server and client paths. I relied on logs, network traces, and postman requests to find out the actual pair of requests and responses instead of making assumptions about the UI. In the case where a fix was productive, I recorded the root cause of the error and left a small utility or guard in the code to make the same error more difficult to introduce again in the future. This eventually made my estimates closer to the truth since I had some idea of the way I would go about the investigation beforehand.

## 5.2 Technical Skills

I was technically transitioned to Next.js 14 (App Router) and TypeScript. I got used to making decisions on which element to serve out of the server to be optimized by SEO and time-to-first-byte and when client interaction was worth a priority. My designs were informed by strong typing: I created my props and data models at the beginning of the process and ensured that any mistakes were known during the development process rather than post-release. My data model On the data side I modeled data in Drizzle ORM over MySQL, constrained it, and wrote readable queries predictably that would be readable in a short time by the reviewer. I put route handlers in between every read and write, checked inputs on the border, and presented regular shapes to the UI. I also tested the success and failure paths using Postman and I also used Lighthouse to measure basic performance to identify image and layout changes. My day to day activity in Ubuntu has enhanced my command line, getting an environment ready to work on codebases, logging, and writing small scripts. Branch discipline Division of labor around version control, clear commit messages, and short pull requests helped streamline the collaboration process and minimized friction during merges.

## 5.3 Soft Skills

Communication was the most change that was precious as soft-skill. I learned to make pull requests in the form of short stories, saying what is wrong, how to fix it and why it works- screenshots, test notes or metrics. This rendered the reviews quicker and less subjective. I also developed the ability to seek help early on with a particular question and a brief overview of what I have already attempted. Time management was also enhanced because I was able to divide work into smaller bites which gave me noticeable advancement. I felt more comfortable when I gave a review by presenting a demo, discussing trade-offs and read between the lines when making decisions, to give background to future contributors. Providing feedback also became simpler: I learned to propose alternatives with a rationale, not only to highlight problems, which allowed to have constructive and result-oriented discussions.

## 5.4 Industry Understanding

Developing a live retail-based environment helped me understand the ways product, engineering, and operations interact. Agile ceremonies do not simply exist on a process level, but generate a regular planning, feedback, and release rhythm. I have observed how user experience, performance, and accessibility have a direct impact on trust, which is that fast pages decrease bounce, clarity of copy decreases support requests, and keyboard friendly control increases the audience. I also learned why there is separation between the environments (development, staging, production) and the secrets have non-source control. Tagged releases, small, made the rollback very easy and allowed the team to learn successfully with no high risk. Lastly, I observed the role of documentation and naming in making a codebase a shared asset, and not a set of individual shortcuts.

## 5.5 Lessons Learned

During this internship I also understood that effective software is the result of doing a lot of little things very well again and again. The greatest change was the ability to design towards change. I have attempted to maintain the separation and simplicity of parts of the system to ensure the addition of a new feature or correcting a bug would not break another part. The use of clear names, typed data (with TypeScript and Drizzle), and short files facilitated the further reading of the code. I also got to know that I should take measurements prior to making any changes. A fast Lighthouse test or a minimal timing test informed me of the location of the real issue. This saved me time and prevented my guessing. I made the pull request small and to the point when I opened a pull request, included screenshots and a brief note. It could be grasped by reviewers in a short time and it could be amalgaminated with ease. The other lesson was to ensure that you do validate inputs at edges. Forms and API routes would verify data early and give useful errors to correct the issue which saved time. That brought a lot of problems to testing rather than after its release. I developed the practice of completing each task to the end that involves coding the code, testing success and error paths, adding a simple test or Postman run, and leaving a brief comment as to why. I have understand that being available

and deliver is a job aspect and not a polish. The app is reliable because of the use of keyboard-friendly controls, clear labels, and stable layouts. Lastly, communication was important: requesting assistance early, telling about blockers, and writing PR descriptions helped the work easier. These guidelines were applicable to all things I made on the homepage, product and FAQs, newsletter, and smaller apps, and this is what I will apply to other features of Shukria Meat.

## 5.6 Industry Insights

The current trend in web practice converts typed full-stack workflows in which the frontend and the backend have common models and contracts. TypeScript and an ORM such as Drizzle completely eliminate the entire category of runtime errors, and frameworks such as Next.js (App Router) promote server-first patterns, which trade off performance with interactivity. Meanwhile, teams are relying on edge-aware deployments and CDNs in order to maintain perceived speed. The features of AI are proliferating but without a clear UX, guardrails, and audit trails they cannot add value. There is an increase in security expectations: least-privilege access, scrutinizing secret treatment and evident input validation are no longer additional, but standard. Experience of the developer is important as well, as good tooling and small releases make the developer learn faster and enhance the quality of the product.

## 5.7 Future Skill Development

The following objectives are practical and stratified. On quality, I would like more automated testing, including component tests and limited end-to-end flows, which will ensure that regressions are spotted earlier. I will also acquire further knowledge on the CI/CD pipelines and containerization on delivery to make builds reproducible and deployments a regular procedure. On data, I will examine indexing and query plans to make the performance better without complexity. In the case of Shukria Meat in particular, I will plan and execute the following set of features-cart, check-out, payments, admin dashboards without changing the accessibility, performance, and security standards. Lastly, I intend to write more straightforward internal documentation and provide reviews to make other people ship better codes in less time; educating me on what I have learned will ensure that these habits become long-lasting.

# Chapter 6

# Conclusion

## 6.1 Summary

This internship provided me with the entire picture of the way the current software is developed: you receive a problem that has an impact on a real user and break it down into small, testable pieces of work, then take those pieces through the design, implementation, reviewing, and verifying process until the product feels reliable. The Shukria Meat web platform is my main contribution and displays this direction. I assisted in coming up with the design where visitors can view products by category, know the outlets, read answers in a frequently asked questions list, and subscribe so that their journeys on a site are not mixed with administrative tasks, and vice versa. Next.js 14 (App Router) and TypeScript were used on the front-end, data modeling was done with Drizzle ORM on MySQL, and the styling was made consistent with CSS Modules. I was taught to choose, feature by feature, whether to render a screen on a server as fast as possible and SEO-friendly or on a client as an interactive environment and demonstrate the decisions with screenshots, Postman traces, and fast performance tests. The same skill was applied to the less critical projects, including an interface of an AI assistant and a portable water filtration concept, where I needed to design to be uncertain and make mistakes readable and maintain a calm UI. The way people worked was just as important as writing code. We did short sprints, small pull requests, and respectful reviews. Clear acceptance notes kept things moving along and kept risks low. I always finished my work, not just writing code and then leaving. I clearly explained the purpose, checked the inputs, and wrote down the limits. I left tests and checklists behind to make future updates easier. Shukria Meat was stable and able to grow by the end. The business could grow to include shopping carts, payment systems, and admin dashboards. We didn't have to break anything that was already there. I also changed the way I think about adding features. I now explain why something works, not just how it works. The change in my mind was the most important thing that happened to me. I have better skills, clearer habits, and faster starts now.

## 6.2 Recommendations for Future Strategic Actions

The next stage for Shukria Meat should focus on turning a clear browsing experience into a complete purchasing flow while protecting performance, accessibility, and security. First, add a lightweight order pipeline: cart, checkout, order confirmation, and transactional emails. Choose a PCI-compliant payment provider and design the flow so sensitive card data never touches application servers; keep error paths honest and helpful (e.g., retry, different method, or save cart for later). Pair this with an admin dashboard that includes role-based access (owner, staff, viewer), bulk product tools, and safe content management for FAQs and outlets. Second, instrument the platform with observability: structured logs, simple metrics (request counts, latency, errors), and a dashboard for core Web Vitals. Small alerts—error spikes, newsletter failures—let the team react before users complain. Third, invest in quality gates: a handful of high-value component tests and two or three end-to-end paths (login, product browse, newsletter) wired into CI. This need not be heavy; the goal is to catch the most expensive regressions early. On the performance side, set budgets (image sizes, script weight, LCP targets) and enforce them in reviews. For accessibility, keep a running checklist: semantic headings, focus order, visible focus states, alt text, and ARIA only when necessary. On the data layer, document the schema and add migration practices so changes are reversible. For security, maintain a clear boundary: validate inputs at route handlers, centralize auth checks, rotate credentials, and keep environment variables out of source control. Finally, operationalize collaboration: standardize the PR template (problem, approach, screenshots, tests), keep decision logs in Notion, and tag releases with plain-language change notes. Together these actions raise the ceiling of what the team can ship without raising the risk. They also build confidence with stakeholders: the software remains fast, trustworthy, and easy to extend as the retail needs grow.

## 6.3 Future Goals

My objectives are realistic and connected with the following action of Shukria Meat. To start with, I would enhance testing. I will also include some component tests (forms, lists) and two or three end-to-end tests of the most significant flows (sign in, browse products, subscribe to newsletter). This will ensure that changes are safer and faster reviews. Secondly, I will enhance our construction and issuance process. I would like to find out more about CI/CD so that the project could assemble each time and launch without hysteria. In addition to it, I will include basic monitoring, like error logs and a simple metrics dashboard, to check the problems in time. Third, I would like to develop in terms of data. My practice will include reading query plans, selecting indexes, and pre- and post-change measures. I also intend to keep performance and accessibility in mind by keeping small budgets that includes image sizes, script weight and using a brief checklist on each new page. In features, I hope to assist in designing and developing cart, checkout and payments, and finally, a light administrative section that consists of products, categories and content. Short design notes will be written before coding, trade-offs in PRs explained, and changes

kept to a minimum. Lastly, I would like to mentor the new contributors when its time to contribute, answer questions, pair-up on hard sections, or write a small guide since the mentoring will solidify my own knowledge and will also help the team work faster.

## 6.4 Reflections on the Internship Experience

The piece was large and vague at the beginning. Gradually I learnt to take small steps in it, which included defining a slice of work, writing the code, providing evidence, in the form of screenshots or a Postman run, receiving feedback, and shipping. That plain beat inspired confidence. The culture of the team was conducive to premature questions and presenting work in progress: it was safe, and courtesy in every form was expected, along with the written notes. I saw that a project moves along more quickly when people share information and there aren't many changes. I also learned that users care about information. Exceptional mistakes help people understand by reducing confusion. More people will be able to use the site if it has clear labels and keyboard navigation. Small performance wins, like low-res images and fewer layout changes, make pages run smoothly. Taking care of basic security issues like checking inputs, making sure configurations are safe, and not having secrets in code can help you avoid problems in the future. I worked on the home page, product pages, outlets, frequently asked questions, and mini apps, so these lessons became habits. The biggest change in me is how I think about quality. I don't want to just make it work anymore. I try to make it understandable, safe, and changeable. That way of thinking affects how I name things, how I organize files, and how I write a pull request. I'm also thankful for the guidance and the chance to ship real features. I'm taking home skills that I can show off, a body of code that I helped write, and the strength to move on to the next stage.

# References


[1] *Git documentation*, https://git-scm.com/doc, Retrieved January 30, 2025, n.d.

[2] *Github documentation*, https://docs.github.com/en, Retrieved January 30, 2025, n.d.

[3] *Laravel documentation*, https://laravel.com/docs, Retrieved January 30, 2025, n.d.

[4] *Mongodb documentation*, https://docs.mongodb.com/, Retrieved January 30, 2025, n.d.

[5] *Node.js documentation*, https://nodejs.org/en/docs/, Retrieved January 30, 2025, n.d.

[6] *Php documentation*, https://www.php.net/docs.php, Retrieved January 30, 2025, n.d.

[7] *Postman documentation*, https://learning.postman.com/docs/, Retrieved January 30, 2025, n.d.

[8] *React documentation*, https://reactjs.org/docs/getting-started.html, Retrieved January 30, 2025, n.d.

[9] Skarbol.Tech. "Skarbol.tech." Accessed: 2025-01-30. [Online]. Available: https://skarbol.tech/.

[10] *Tailwind css documentation*, https://tailwindcss.com/docs, Retrieved January 30, 2025, n.d.