**Project Report**

**Vision statement**

The product we've developed is valuable to our client as it fulfills the specified requirements of the project. It has a range of user-friendly features, making it a versatile and efficient tool for users. One of the main features is the ability to compare prices from various brands and supermarkets, presenting users with the cheapest option. Additionally, users can easily search for specific items through the search function and apply filters to enhance their search, ensuring they find exactly what they're looking for.

Users can add items to their cart that they wish to purchase. They can review and edit their cart before confirming an order. The application allows users to register an account. The account holds details such as their delivery address, preferred delivery time and method, along with their name, email, and phone number. If the user is logged in while confirming their order, the details will be automatically used when checking out. Once the order is confirmed, the order is dispatched and arrives at the specified address at the designated time.

The application also provides detailed information about each item, including its name, description, image, and price. This detailed information allows users to easily compare products from different supermarkets, enabling them to identify the most cost-effective option for the item they desire. Overall, the product enhances the user's shopping experience by providing a convenient, transparent, and informative platform for comparing prices, selecting items, and placing orders online.

**Final Design Architecture**

We ended up having one database to store all the information instead of having multiple databases as it was not within our timeline and not within our current capabilities so we had to opt with a single database although ease of implementation has many security issues as SQL injection would allow the attacker to access private information.

**Git Organisation**

In our meticulously organised Git repository, we have adhered to a disciplined approach to software development, ensuring efficiency and collaboration. Our workflow revolves around the practice of branching out for each feature implementation, creating a dedicated branch where developers can focus on their respective tasks without impacting the stability of the main codebase. This strategy not only promotes isolation and parallel development but also facilitates a streamlined process for code reviews and quality assurance.

Moreover, our commitment to frequent and granular commits has been paramount in maintaining code health. We've emphasized the importance of breaking down changes into smaller, logically coherent units, allowing for better code traceability and easy identification of potential issues. This practice not only fosters a culture of transparency but also simplifies the debugging process and minimizes the risk of introducing complex and hard-to-diagnose problems.

At the core of our Git workflow is the principle of feature completeness. We diligently merge feature branches back into the main branch only when they are fully developed, tested, and validated. This ensures that the main branch remains stable and deployable at all times, safeguarding against regressions and providing a reliable foundation for ongoing development efforts. By maintaining this meticulous approach to branching and merging, we empower our team to work cohesively, embrace rapid iterations, and deliver high-quality software that meets both user expectations and project milestones.

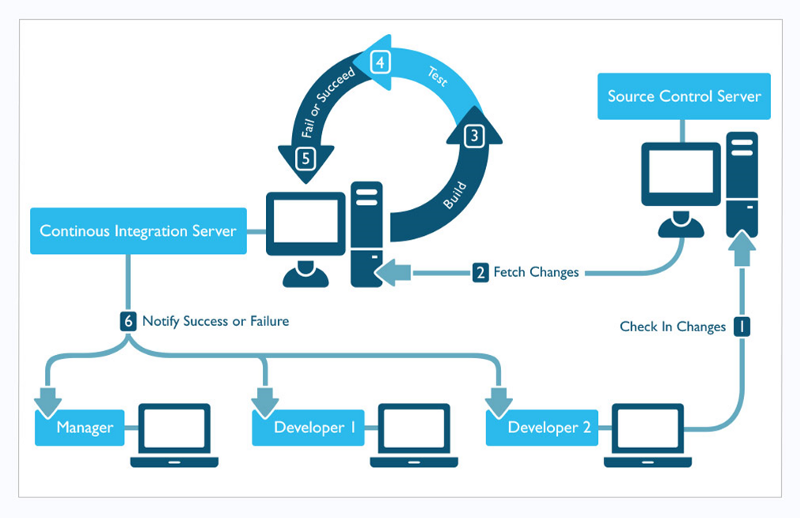
**Scrum Process**

Our project's success hinged on a well-structured communication framework, which prioritized regular meetings and a steadfast commitment to Scrum principles. Meeting at least once a week, our team gathered to foster collaboration, transparency, and collective progress. These meetings, meticulously organized and facilitated by our adept Scrum Master, Brian Tran, were pivotal in steering our development journey toward smooth waters, with remarkably few hiccups along the way.

Brian Tran, our guiding force, played a crucial role in orchestrating these meetings and ensuring they were productive. As the Scrum Master, Brian brought to bear a wealth of experience and expertise, serving as a facilitator, coach, and advocate for the Scrum framework. His leadership was instrumental in maintaining our adherence to Scrum principles, such as time-boxed sprints, self-organization, and empirical process control. Brian's efforts went beyond merely conducting meetings; he acted as a guardian of the Scrum process, shielding the team from external distractions and allowing us to focus on delivering value to our stakeholders.

Through our weekly meetings and Brian's guidance, we fostered an environment where ideas flowed freely, challenges were addressed promptly, and collective decisions were made with a shared understanding of project goals and priorities. These gatherings became a forum for sharing progress, discussing impediments, and refining our approach, ultimately steering our development efforts in the right direction.

**CI/CD Pipeline**



Note: Rough Diagram of the desired pipeline

In our CI/CD pipeline, we effectively harnessed the power of GitHub Actions to seamlessly orchestrate our continuous integration (CI) processes. GitHub Actions allowed us to automate various aspects of our development workflow, including building, testing, and validating our code changes. This automation ensured that each code commit was thoroughly examined and that any issues were detected early in the development cycle.

As for our continuous delivery (CD) efforts, we leveraged the capabilities of Amazon Web Services (AWS) to facilitate the deployment of our applications. Specifically, we utilized AWS Elastic Container Repository (ECR) to manage and store our Docker images securely. Once our Docker images were pushed to ECR, they were readily available for deployment to various environments.

Our CD pipeline took advantage of the seamless integration between AWS services, allowing us to deploy our containerized applications to Amazon Elastic Container Service (ECS), Elastic Kubernetes Service (EKS), or any other appropriate AWS compute service. This approach enabled us to efficiently manage and scale our applications in response to user demand while maintaining the reliability and security that AWS provides.

In summary, our CI/CD pipeline incorporated the best of both worlds, with GitHub Actions streamlining our CI processes, and AWS, including ECR and ECS/EKS, providing a robust and scalable platform for our CD needs. This combination of tools and services ensured that our development and deployment processes were efficient, reliable, and capable of meeting the demands of our dynamic software projects.