Project Proposal: Members Record Management System (GotoGro-MRM) for Goto Grocery Inc.

Members:

Enzo Peperkamp - 102895415

After thorough analysis and collaboration among team members, the chosen solution for GotoGro's system is the cloud-hosted Model-View-Controller (MVC) architecture, specifically as outlined by Alex. This decision was reached through a comprehensive KoST (Knowledge, Skill, Technology) analysis, validating its alignment with industry needs and business requirements. Opting for a serverless architecture made implementation easier, with less maintenance overhead, fitting the organization's small-scale context, which may lack resources for maintaining its own server infrastructure. Among the alternatives evaluated, including traditional monolithic applications and various server-based solutions, this serverless architecture emerged as the most suitable option. The decision reflects a well-tested response to a common industry problem, allowing for easy extensibility for potential future upgrades, and solidifying it as the optimal choice for GotoGro.

Nelchael Kenshi Turija - 103057559

I wholeheartedly support the team's use of the solution as it perfectly meets GotoGro's demands. Our team's minimal resources are in line with the serverless nature of this method, and the modular design provides scalability and maintenance will be kept at a minimum. The project's value proposition is further strengthened by the attention to user experience through a React JS application and the implementation of predictive algorithms. This decision displays a carefully thought out plan that best meets Goto Grocery's objectives and overcomes obstacles that may be presented.

Julian Codespoti - 102997816:

Having closely reviewed the analyses and recommendations put forth by my team members, I am in full agreement with our collective decision to adopt the cloud-hosted Model-View-Controller (MVC) architecture for GotoGro's system. Drawing from my experience with software development and system design, the MVC approach is both efficient and effective, especially when hosted in the cloud. The separation of concerns, inherent in the MVC structure, facilitates a streamlined development process, ensuring that GotoGro's objectives are met without incurring unnecessary complexities. Additionally, leveraging cloud capabilities not only offers reliability and scalability but also ensures cost-effectiveness. This aligns with our goal of ensuring that GotoGro achieves its operational objectives without being burdened by excessive infrastructure costs. I appreciate the rigorous process and collaboration that led to this conclusion, and I am confident that this choice will prove to be the cornerstone of GotoGro's success.

Alex Kyriacou - 103059830

I support the solution outlined below. While the final architecture closely represents the one I proposed individually, it still accurately represents the conclusions we came to as part of our group discussions. The architecture that is seen below is one that is best suited to the GotoGro brief while also considering both the team's strengths and weaknesses (as described in the KoST analysis). We also ensured to follow architectural best practices as described within the lecture content. Examples of our discussions of the best solution can be seen within the alternatives section.

Marella Morad - 103076428

After extensive discussions with team members, we have concluded that a cloud-hosted Model-View-Controller (MVC) architecture (outlined by Alexander) is the optimal solution for Goto Gro's system. This choice aligns closely with my proposed solution, as both emphasize the separation of concerns into distinct components (i.e., Model, View, and Controller). The decision to opt for a cloud-hosted MVC architecture was driven by its dynamic scalability, enabling efficient performance during peak user loads—unlike the multi-layer architecture. Moreover, cloud hosting ensures inherent security measures and automated backups, directly aligning with the project's objectives of flexibility, security, and modernization.

Solution Direction

Description of chosen solution direction

After careful consideration and assessment between all group members' proposed solution. The team has decided to go with the solution proposed by Alex. Namely, we have decided to develop a cloud hosted Model-View-Controller (MVC) architecture. This architecture is a natural choice given the brief. GotoGro is a small organisation that is likely unable to maintain its own server infrastructure. Therefore maintaining a simple serverless architecture allows for easy extensibility if future upgrades are required. This solution is a well tested solution to this common industry problem and consists of the following architecture:

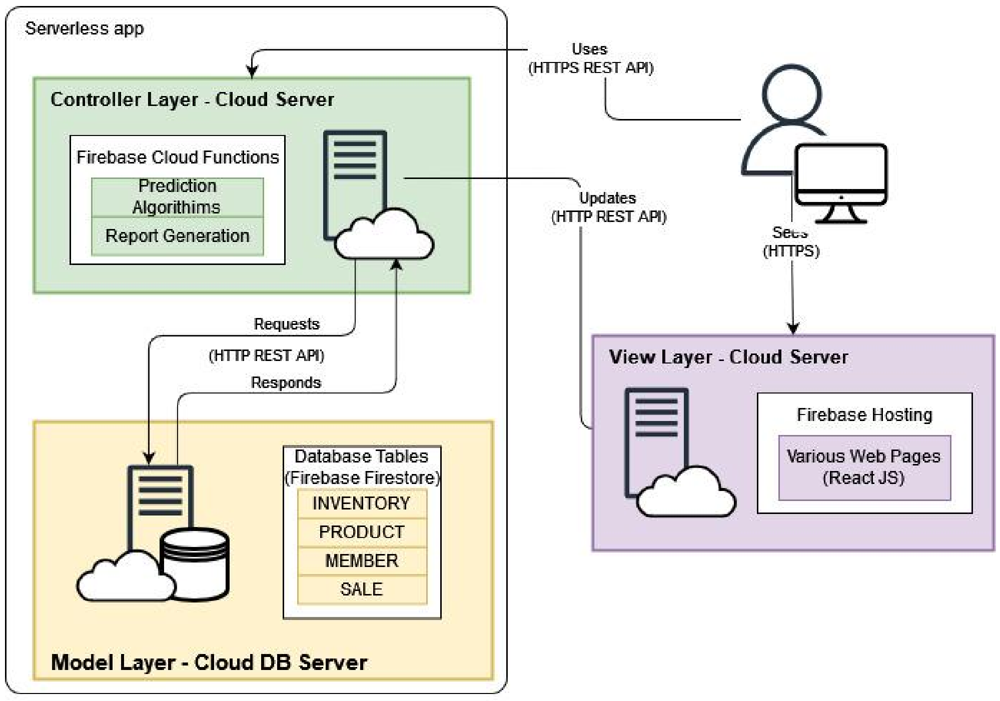

Figure 1: Architecture Of Agreed Solution Direction

Model Layer

This consists of a cloud hosted relational database service that will store all the database entities required for the system. This layer will interact with the controller layer in order to update records as well as retrieve relevant data from the database for The UI, report generation, algorithm prediction etc. Given this is using a serverless infrastructure, this will expose an endpoint that can be queried by the controller.

View Layer

This layer is what the end-user interacts with through their browser and represents the GUI. This consists of a cloud hosted React JS application that will display the relevant data, reports and UI to the user. This will query the controller to retrieve and update any data that is required.

Controller Layer

This consists of a server that serves as the intermediary between the View Layer and the Model Layer. The Controller serves to handle all requests from the View Layer and translate them into database queries and software processes before returning the final result. This includes generation of inventory reports, execution of a prediction algorithm or retrieval of a certain member's sales. The Controller Layer will be created by a number of cloud hosted serverless functions that will act as a robust REST API that can be queried by the View Layer to perform required tasks.

Alternatives

Architecture	Description	Reason for Discarding	Potential Use Cases for GotoGro
Traditional Monolithic Application	Single-tiered software where GotoGro's UI, business logic, and data access are all in one codebase, likely hosted on a single server.	While suitable for initial GotoGro setup with a single store, it lacks scalability. Updates would affect the entire application.	If GotoGro remained a small local business with no plans to expand and only required basic features.
Frontend + Serverless Backend	GotoGro's web interface connects to serverless functions that execute backend tasks, potentially leveraging platforms like AWS Lambda.	While it allows for auto-scaling, the unpredictable latency might disrupt GotoGro's real-time inventory or billing operations.	For specific event-driven features, like sending notifications to customers about offers or processing one-off bulk orders.

SPA with Integrated Backend	GotoGro as a Single Page Application (SPA) with dynamic content loading from an integrated backend.	Scalability concerns arise as the SPA grows. SEO might be a challenge if not properly optimised.	If GotoGro decided to launch a special promotional website with limited-time offers and wanted smooth user interactions without full page reloads.
PWA + API Backend	Progressive Web App (PWA) version of GotoGro providing native-like capabilities, interacting with a backend via APIs.	Complexities related to service workers, cache management, and potential pitfalls in offline data syncing.	If GotoGro decided to create an offline-capable app for locations with spotty internet, like pop-up stalls or kiosks in remote areas.
Micro-frontends with Microservices Backend	GotoGro's frontend and backend split into smaller, independently deployable units. Each store or department could have its own micro-frontend.	High operational overhead and intricate deployment pipelines.	Ideal for a future where GotoGro has multiple distinct departments or franchises, each wanting autonomy in their tech stack and deployments.
Full-stack Desktop Application	A desktop application tailored for GotoGro staff, managing everything from inventory to billing, and syncs to a remote database.	Challenges in cross-platform compatibility, deployment, and updating across stores.	If GotoGro planned to provide a highly specialised in-house software tool for staff, focusing on intensive tasks like 3D product visualisation or complex inventory planning.

Chosen Solution Analysis

KoST Analysis

Knowledge (K):

- **Problem Domain:**
  - **Paper-Based Records Management:**
    - GotoGro's current manual management, akin to traditional retailers, is prone to errors. Transitioning to an electronic system will streamline tracking and analysis.

   - **Brick and Mortar Retail:**
      - Sales at GotoGro, like other physical stores, require an electronic system for transaction entries, as records are not created automatically.
   - **Understanding of Grocery Store Operations:**
      - Comprehensive grasp of industry standards, including member management, sales records management, inventory management, and member-based business models, similar to those in prominent grocery chains.
- **Solution Domain:**
   - **Electronic Records Management:**
      - Implementation akin to modern e-commerce platforms, for tracking orders and members, reducing manual effort seen in GotoGro's current process.
   - **Electronic Inventory Control:**
      - Integration with systems similar to ERP solutions that manage stock levels.
   - **Data Analysis and Processing:**
      - Usage of predictive algorithms and ETL practices, comparable to leading data analytics tools, for trend analysis and data cleaning.
   - **Software Engineering Expertise:**
      - Familiarity with .NET languages like C#, C++, JavaScript (including frameworks like Angular, React), PHP, HTML, CSS, Python, and various DB systems like MongoDB, SQL, NoSQL, and Firebase.
   - **User Experience Design:**
      - Applying principles used in successful web applications to ensure a user-friendly interface.
   - **Alignment with Business Needs:**
      - Tailoring the solution to meet GotoGro's specific needs, leveraging understanding of grocery operations and member management, mirroring best practices in the industry.

Skill (S):

- **Web Development:**
   - **E-commerce Systems:**
      - Experience in building basic web-based e-commerce systems with inventory management, similar to some small-scale online retailers, using .NET, JS, and other related technologies.
      - Willingness to mentor the team and fill experience gaps in this area.

- **Cloud-Based Infrastructure:**
    - While there's an experience gap in using cloud-based hosting, there is a commitment to learning and implementing basic cloud-based hosting for websites, databases, and servers.

- **Industry Experience:**
    - **Supermarket Retailer Insight:**
        - Experience working within large supermarket retailers like those found in major chains, providing valuable insights into inventory management, usability, and UI decisions of competitors.
- **Data Analysis:**
    - **Gap in Predictive and Trend Analysis:**
        - Acknowledgment of a need to enhance skills in predictive and trend analysis, required for inventory prediction algorithms, mirroring sophisticated data science applications in other industries.
- **User Interface Design:**
    - **Intuitive Design:**
        - Proficient in designing intuitive and user-friendly graphical user interfaces, applying principles found in successful digital platforms.
- **Database Management:**
    - **Designing and Managing Relational Databases:**
        - Skillful in managing relational databases, utilising popular DB systems like SQL, MongoDB, Firebase, in line with best practices in database design.
- **Programming Skill:**
    - **Wide Range of Languages:**
        - Proficiency in languages including .NET languages (C#, C++), JavaScript (including Angular, React), PHP, HTML, CSS, Python, ensuring the development of required functionality.
- **Project Leadership:**
    - **Agile Model Leadership:**
        - Experience in leading teams under the Agile model, reflecting a strong understanding of Agile principles, and a readiness to take ownership in team development.
- **Alignment with Business Needs:**
    - Leveraging the above skills to precisely meet GotoGro's specific needs in terms of web development, UI design, inventory prediction, and member record management.

Technology (T):

- **Existing Systems:**
    - Platforms like Woolworths and Coles' websites demonstrate existing technology for online groceries, but they are not tailored to Goto Grocery's specific needs, especially regarding sales recording and member management.
- **Customization Gap:**
    - While similar member and sales record management applications exist, the supermarket industry has generally lacked systems that can be customised to individual retailers' unique needs and operations.
- **Technological Accessibility and Innovation:**
    - The proposed application for Goto Grocery leverages existing and well-established technologies such as web development frameworks and database technologies like MongoDB, SQL, Firebase.
    - This approach ensures accessibility without the need to introduce unique or new technology, in line with current industry standards.
- **Historical Gap in Supermarket Industry:**
    - Historically, the supermarket industry's focus has been on in-store experience and product availability rather than detailed member management and predictive inventory control.
    - This can be likened to the gap in the car industry where sensors were absent, leading to a lack of assistance in tasks like backing a car.
    - Addressing Goto Grocery's problem through a tailored Members Record Management System (MRM) fills this gap by emphasising member preferences, inventory prediction, and seamless sales recording.
    - Just as sensors revolutionised safety in cars, a custom-built MRM can transform member satisfaction and operational efficiency in supermarkets, setting a precedent for industry evolution.
- **Alignment with Business Needs:**
    - By focusing on Goto Grocery's specific requirements and industry-wide gaps, the technology chosen for this project represents a blend of innovation, accessibility, and customization, designed to elevate Goto Grocery's operations and potentially reshape technological norms within the supermarket industry.