Team Roles and Responsibilities Breakdown

---

1. Team Structure & Responsibilities

Frontend (2 members)

- **Responsibilities**:
  - **Design**:
    - Create wireframes and user interface designs using Figma or Adobe XD.
    - Ensure responsive design (desktop and mobile).
  - **Code**:
    - Implement the UI using **React.js**.
    - Build reusable components and integrate with backend APIs.
    - Set up routing using React Router.
    - Handle state management using Context API or Redux.
  - **Communication**:
    - Work closely with backend developers for API integration and testing.

Backend Implementation (1 member - C#)

- **Responsibilities**:
  - Develop core business logic using **ASP.NET Core**.
  - Implement **MVC structure**:
    - **Models**: Define database schemas and objects.
    - **Controllers**: Handle API requests and responses.
    - **Views**: Optional if building server-rendered pages (mostly JSON responses for APIs).
  - Write endpoints for:
    - User Authentication (login, registration).
    - Product Management (CRUD operations).
    - Order Processing (create orders, fetch order history).
  - Unit Testing:
    - Ensure endpoints work correctly and integrate seamlessly with databases.

Microservices Development (2 members)

- **Responsibilities**:
  - **Member 1**:
    - Build core microservices:

            - **User Service**: Handles authentication and user profiles.
            - **Product Service**: Manages product catalog.
            - **Order Service**: Processes user orders.
        - Integrate RabbitMQ for asynchronous communication.
        - Use REST APIs for inter-service communication.
        - Write unit tests for individual services.
    - **Member 2**:
        - Work on deployment and infrastructure setup:
            - Create **Dockerfiles** for each service.
            - Use **Kubernetes (K8s)** for container orchestration.
            - Set up monitoring using **Prometheus** and **Grafana**.
        - Build the **CI/CD pipeline**:
            - Automate builds, tests, and deployments using GitHub Actions or Jenkins.

LLM Specialist (1 member)

- **Responsibilities**:
    - Research and train the **Large Language Model (LLM)**:
        - Choose a framework like **Hugging Face Transformers** or **OpenAI API**.
        - Fine-tune the model on domain-specific data (if applicable).
    - Package the model as a microservice:
        - Use **FastAPI** or Flask for the service.
        - Dockerize the service and ensure it integrates with other microservices.
    - Add endpoints for:
        - Querying the LLM (e.g., `/query`).
        - Retrieving model performance metrics.
    - Ensure scalability:
        - Optimize the model for production (e.g., using TensorRT or ONNX).

---

2. Workflow

- Use **Agile Methodology**:
    - Conduct daily stand-ups to track progress.
    - Divide tasks into sprints (2-week cycles).
- Centralize documentation using tools like Notion or Confluence.
- Use GitHub or GitLab for version control:
    - Create separate branches for frontend, backend, and each microservice.
    - Implement code reviews for quality assurance.

---

3. Collaboration and Integration

- **API Contracts**:
  - Define API structures and contracts upfront using **Swagger** or **Postman**.
- **Regular Syncs**:
  - Hold weekly integration sessions to test communication between frontend, backend, and microservices.
- **Data Flow Testing**:
  - Ensure that the frontend can consume data from microservices via the backend APIs.
  - Verify that LLM responses are correctly displayed on the frontend.

---

4. Deliverables

Week 1-2:

- Set up the environment (tools, frameworks, and repository).
- Finalize the architecture and API contracts.
- LLM: Begin data collection and model training.

Week 3-4:

- Frontend: Complete initial design and core pages (login, home, product listing).
- Backend: Build authentication and basic product endpoints.
- Microservices: Develop user and product services; set up RabbitMQ.

Week 5-6:

- Frontend: Integrate APIs and finalize UI functionality.
- Backend: Complete order service and inter-service communication.
- CI/CD: Deploy initial services to Kubernetes.
- LLM: Package the trained model into a microservice.

Week 7-8:

- Perform end-to-end testing.
- Optimize performance (queries, LLM inference, caching).
- Deploy the application to production.

This plan ensures clear roles and collaboration while keeping the project on track. If you'd like, I can provide more granular details for specific parts of the workflow.