Reflection Report Template

Student 1
University of Southern Denmark, SDU Software Engineering, Odense, Denmark
Email: student1@mmmi.sdu.dk

I. Burger 4.0 Production System

A. Use Cases

- Order Placement: Customers can place orders through the GUI, selecting from the menu options.
- Order Tracking: Customers can view the progress of their orders in real-time through the GUI.
- Ingredient Preparation: Stations prepare and process the ingredients required for burger, fries, and soda orders.
- Burger Assembly: A dedicated station assembles the burger orders according to the customer's specifications.
- Robotic Transport: Robots autonomously move ingredients and assembled items between stations.
- Packaging: A packaging station ensures that orders are appropriately packaged before delivery.
- Order Delivery: Robots, similar to MIR-bots, deliver completed orders to the pick-up station for customers.

B. Required Stations

1) Ordering GUI Station: Provides the user interface for customers to place orders and track their progress.
   - Programming Language: HTML, JavaScript
   - Libraries/Frameworks: React, Angular, Vue.js
2) Ingredient Preparation Stations: Multiple stations equipped with various technologies for preparing burger ingredients, fries, and soda.
   - Programming Language: C++
   - Libraries/Frameworks: OpenCV (for computer vision), PLC Programming (for control)
3) Burger Assembly Station: A specialized station for assembling burgers with precision.
   - Programming Language: Python
   - Libraries/Frameworks: Robotic Process Automation (RPA) software, such as UiPath
4) Robotic Transport System: Autonomous robots equipped with sensors and navigation systems for moving ingredients and assembled items between stations.
   - Programming Language: ROS (Robot Operating System)
   - Libraries/Frameworks: ROS Navigation, Gazebo (for simulation)
5) Packaging Station: A station responsible for packaging orders securely.
   - Programming Language: Java
   - Libraries/Frameworks: JavaFX (for HMI), RFID libraries
6) Order Pick-up Station: Customers collect their orders from this station, where robot deliveries are made.
   - Programming Language: C
   - Libraries/Frameworks: .NET, MIR Robot SDK

C. Relevant Technologies

- IoT Sensors: Utilized in ingredient preparation stations for monitoring and controlling cooking processes.
  - Programming Language: Python, C++
  - Libraries/Frameworks: Raspberry Pi (for IoT), DHT22 (for temperature and humidity sensing)
- Machine Vision Systems: Used in robotic transport systems for object recognition and navigation.
  - Programming Language: Python, C++
  - Libraries/Frameworks: OpenCV, PCL (Point Cloud Library)
- Artificial Intelligence (AI): AI algorithms can optimize the routing and coordination of robots within the production system.
  - Programming Language: Python, Java
  - Libraries/Frameworks: TensorFlow, PyTorch, scikit-learn
- IoT Communication Protocols: Utilized for real-time data exchange between stations and the central control system.
  - Protocols: MQTT, CoAP
- Event-Driven Architecture: Enables communication between stations and updates the GUI with order progress information.
  - Programming Language: Node.js (JavaScript)
  - Libraries/Frameworks: Apache Kafka, RabbitMQ
- Databases: Used for storing order data, inventory information, and system logs.
  - Database Management System (DBMS): PostgreSQL
  - Libraries/Frameworks: SQLAlchemy (Python ORM)

D. Assumptions

- The system operates in a controlled environment, simulating the production process without physical

machines.

- Sufficient computing resources and network connectivity are available to support the system's operations.
- Real-time data exchange and communication among stations are reliable and low-latency.

E. Challenges and Considerations

- Scalability: Consider the system's scalability to accommodate varying order volumes and production demands.
- Fault Tolerance: Implement redundancy and failover mechanisms to ensure uninterrupted production.
- Data Security: Safeguard customer data and system integrity against potential cyber threats.
- Maintenance and Upkeep: Plan for regular maintenance and updates of the simulated system.
- Integration: Ensure seamless integration of technologies and stations for efficient production.

II. Discussion

III. Reflection

Reflection

IV. Conclusion