1.

1. Recruiting and Onboarding

- **Job Posting and Applicant Management**:
  Centralized management of applications across 600+ job portals.
- **Interview Scheduling and Contract Signing**:
  Integrated with Outlook for interviews; supports e-signatures.
- **Automated Onboarding Flows**:
  Template-based onboarding processes for quick new hire setup.

2. Employee Information and Organization Management

- **Digital Employee Files**:
  Manage personal information, contracts, salaries centrally.
- **Organizational Charts & Job Architecture**:
  Visualize organizational structures, define roles, salary bands.
- **Multi-Entity (Legal Entity) Management**:
  Suitable for companies operating across different countries/entities.

3. Payroll and Time Management

- **Payroll Management and Previews**:
  Syncs real-time employee data to ensure accurate payroll.
- **Time Tracking and Absence Management**:
  Employees log work hours and submit time-off requests via the platform.

4. Performance and Development

- **Goal Setting and Feedback Cycles**:
  Track employee goals and manage periodic performance reviews.
- **Training and Career Development**:
  Manage employee learning programs and career paths.

5. Employee Self-Service

- **Self-Service Portal**:
  Employees can update their own data, download payslips, request time off.
- **Employee Surveys and Whistleblowing**:
  Tools for employee feedback and anonymous compliance reporting.

6. Reporting and Analytics

- **Real-Time Analytics**:
  Custom dashboards and reporting tools to guide HR decisions.

Microservices Architecture	Each module (e.g., recruiting, payroll) functions as an independent service, improving scalability and maintainability.
Event-Driven Architecture (EDA)	Internally, services likely communicate through event messaging (e.g., employee created → trigger payroll setup).
Service-Oriented Architecture (SOA)	Modular, service-based communication with a well-defined API boundary between modules and external systems.

2.
Its using a combination of peer to peer architecture and server client architecture
The game world is split into regions and each player has an AOI around their own character.
For each character and object in the world there exists a master node which is responsible for the management of it. Usually the player with the ID closest to the object ID becomes the master node. Every other player who is in range of this object subscribes(pubsub architecture) to the master node and gets a replica that is stored locally. When the player makes changes to their local replica, that change is sent to the master node. The master node then transmits the change to all subscribed players. This architecture saves cost since for so many players having a client server architecture would be extremely expensive.

3.
a ticket machine is in one station, one station can have multiple ticket machines. Each station has

4.
a)
blackboard architecture(???)
b)
peer to peer for low latency. Maybe client server for user data and connection establishment
c)
event driven architecture(pub sub?) as we wait for the cat to move and dont want to be notified when nothing happens.