**1. Deploying the Backend (FastAPI) on AWS**

**Goal: Host the FastAPI backend to handle ticket validation requests.**

**How to do it?**

* **Choose AWS Service: Deploy on EC2 (VM-based) or AWS App Runner (managed service).**
* **Setup Deployment:**
  + **Create an EC2 instance (Ubuntu).**
  + **Install dependencies (Python, FastAPI, Uvicorn).**
  + **Run the API as a service (gunicorn or systemd).**
  + **Set up a reverse proxy with NGINX for security.**
* **Make it Seamless:**
  + **Automate deployment using GitHub Actions/GitLab CI/CD.**
  + **Use Terraform to define infrastructure so setup is repeatable.**
  + **Attach a Load Balancer if needed for scalability.**

**2. Hosting the Frontend**

**Goal: Make the website publicly accessible.**

**How to do it?**

* **Static Frontend (React/Vue/Angular)?**
  + **Host on AWS S3 + CloudFront (cheapest & scalable).**
* **Dynamic Frontend?**
  + **Deploy on EC2 or AWS Amplify.**
* **Make it Seamless:**
  + **Use GitHub Actions to deploy updates automatically.**
  + **Configure CloudFront caching to reduce latency.**

**3. Enhancing Ticket Validation with GenAI**

**Goal: Use AI to verify if a ticket is fake, tampered with, or bought from an unofficial site.**

**How to do it?**

* **Model Selection: Use Amazon Bedrock (LLMs) or AWS SageMaker for AI validation.**
* **Implementation:**
  + **Train the model to check QR authenticity, ticket metadata, and tampering patterns.**
  + **Integrate this AI validation step in the backend before marking a ticket as valid/invalid.**
* **Make it Seamless:**
  + **Fine-tune AI with real-world ticket data for better accuracy.**
  + **Cache AI responses to avoid repeated processing.**

**4. Automating Deployments (CI/CD)**

**Goal: Ensure automatic deployments without manual effort.**

**How to do it?**

* **Pipeline Setup:**
  + **Use GitHub Actions/GitLab CI/CD to deploy frontend and backend on each code update.**
  + **Terraform manages AWS infrastructure, ensuring consistent deployment.**
* **Make it Seamless:**
  + **Automate rollback in case of failure.**
  + **Use versioning for easy debugging.**

**5. Logging, Monitoring & Security**

**Goal: Ensure stability, detect fraud, and maintain system health.**

**How to do it?**

* **AWS CloudWatch: Monitor API logs.**
* **AWS GuardDuty: Detect suspicious activity.**
* **AWS WAF: Protect against fake ticket injection attacks.**
* **Make it Seamless:**
  + **Set up alerts for API failures.**
  + **Automate security patching.**

**Final Outcome:**

**✅ Users can upload tickets → AI validates authenticity → Backend processes it → Frontend shows results.  
✅ Deployed securely on AWS with automation & monitoring.**