Persona Runner: AI-Powered UX Testing Platform

Automated UX Testing for Web Applications — Customer Deployment Guide

AWS Marketplace | Container Product | ECS Fargate | Amazon Bedrock

Overview

Persona Runner is a persona-driven UX testing and reporting platform that orchestrates AI browser agents to simulate real user personas on any live web interface. Built for accessibility-conscious development teams and QA engineers, it evaluates websites through the lens of an Elderly User (limited dexterity, large fonts preference), a Color-Blind User (contrast sensitivity), a University Student (speed-focused, multi-tab), and a Cognitive/Dyslexic User (plain language, clear navigation).

Each persona is configured with defined abilities, frustrations, interaction patterns, and success goals delivering qualitative and quantitative UX insights that traditional automated tests cannot provide.

The platform captures evidence at each interaction step — screenshots, raw interaction logs, and structured per-persona UX metric scores — then generates comparative accessibility reports delivered in a clean React dashboard.

Key Features and Benefits:

- AI-powered persona simulation using browser-use and Amazon Bedrock, executing realistic multi-step journeys on live production or staging websites with zero code changes required on the target site.
- Automated evidence capture: screenshots at each interaction step, raw interaction logs, and structured per-persona UX metric scores.
- Side-by-side comparative analysis across multiple UX test runs lets teams spot persona-specific regressions and cross-run accessibility gaps, with REST API and live SSE streaming for CI/CD integration.

- Rage click detection, navigation backtrack analysis, KLM time estimation, and automated accessibility scanning.

What You Provide

- An existing **AWS account** with permissions to deploy CloudFormation stacks, create IAM roles, and ECS services.
- A **VPC** with at least two public subnets and two private subnets.
- A **security group** for ECS Fargate tasks with appropriate inbound and outbound rules.
- A pre-created **Secrets Manager secret** containing your database credentials.
- **Amazon Bedrock model access** enabled for Anthropic Claude Sonnet 4 in your account and region.

Prerequisites

AWS Account Requirements

- Permissions to deploy CloudFormation stacks, create IAM roles, ECS services, ALB, and RDS instances.
- A **VPC** with DNS resolution enabled.
- At least **two public subnets** in different Availability Zones for the Application Load Balancer and ECS tasks.
- At least **two private subnets** in different Availability Zones for the RDS PostgreSQL instance.
- **Amazon Bedrock model access** enabled for Anthropic Claude Sonnet 4 in your AWS account and region.
- A pre-created **Secrets Manager secret** containing runtime credentials — see **Step 2** below.

Security Group for ECS

Create (or identify) a security group for the ECS Fargate tasks. Ensure it allows:

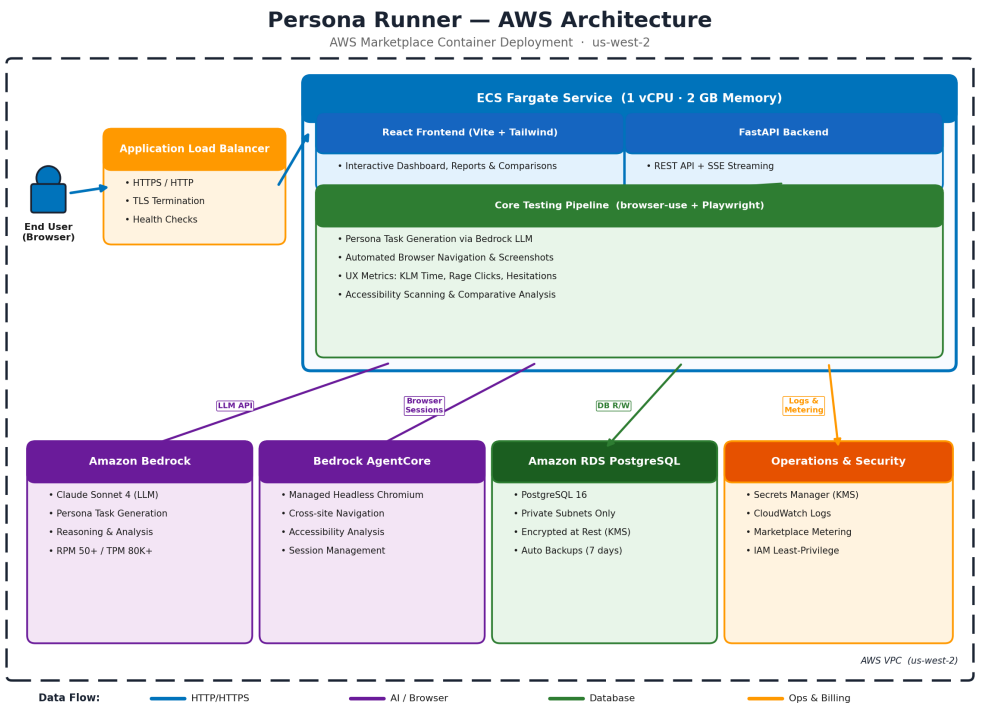
- **Inbound:** TCP port 8000 from your ALB or VPC CIDR.
- **Outbound:** HTTPS (443) for AWS API calls (Bedrock, Secrets Manager, ECR, Marketplace Metering).

The CloudFormation template automatically creates a separate RDS security group that permits inbound PostgreSQL (TCP 5432) only from this ECS security group. Contact your network administrator if you need assistance configuring security group rules.

Local Tools

- **AWS CLI v2** with credentials configured, or access to the AWS Management Console.

Architecture

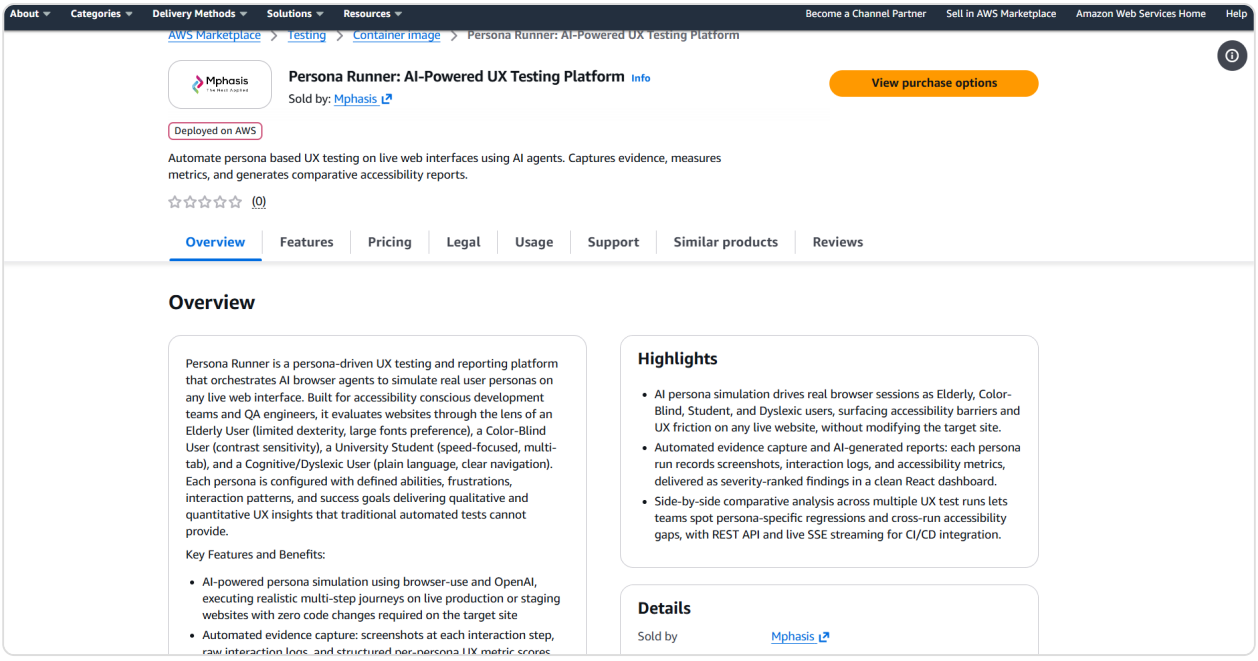


Layer	Technology	Description
Frontend	React 18, Vite, Tailwind CSS, Recharts	SPA for project management, persona selection, live execution, comparison dashboards
Backend API	FastAPI, Uvicorn	REST API + Server-Sent Events for real-time log streaming
Core Pipeline	browser-use, Playwright	AI-driven browser automation with persona-specific behaviors
AI / LLM	Amazon Bedrock (Claude Sonnet 4)	Task generation, intelligent navigation, accessibility analysis
Browser	Bedrock AgentCore BrowserClient	Managed headless Chromium browser service
Database	PostgreSQL 16 (Amazon RDS)	Projects, persona runs, metrics, and metering dimensions

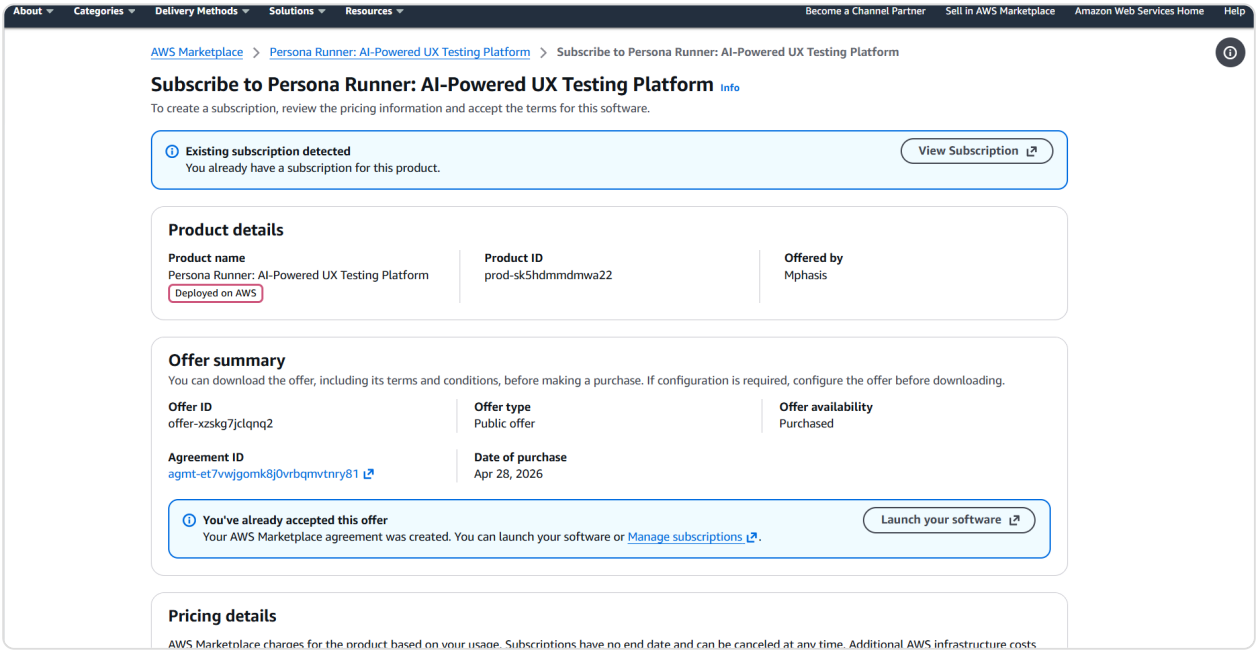
Layer	Technology	Description
Metering	AWS Marketplace Metering API	Usage-based billing — 1 unit per completed test run

Step 1 — Subscribe on AWS Marketplace

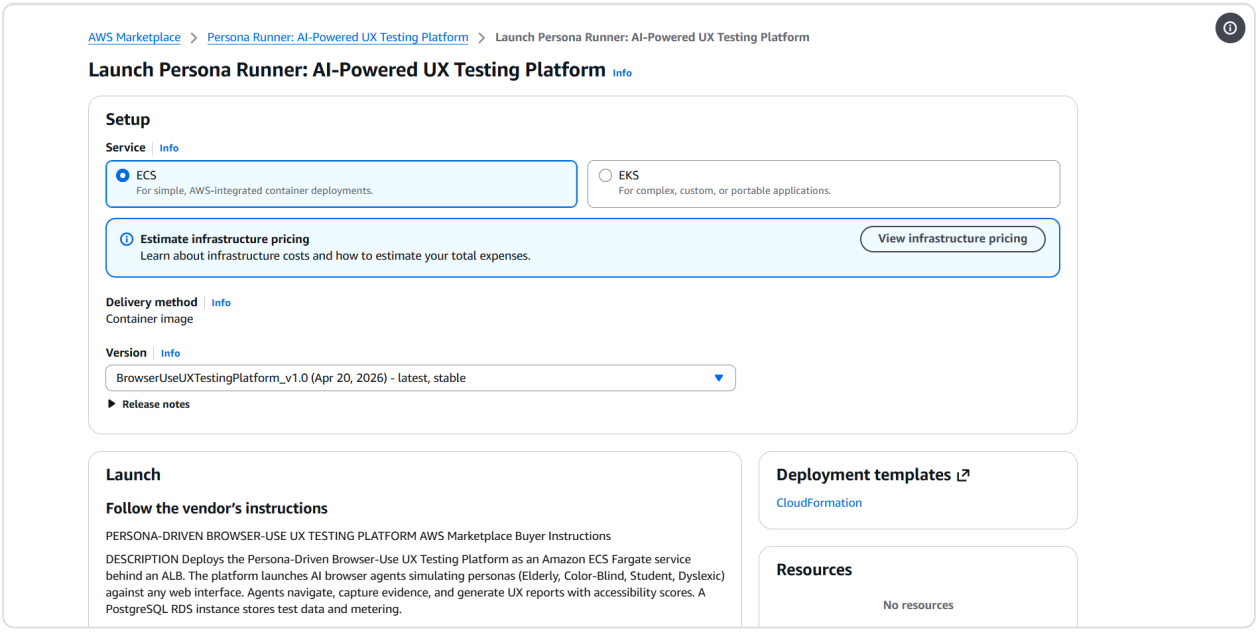
1 **Find the product** on AWS Marketplace by searching for "Persona Runner" and click *Continue to Subscribe*.



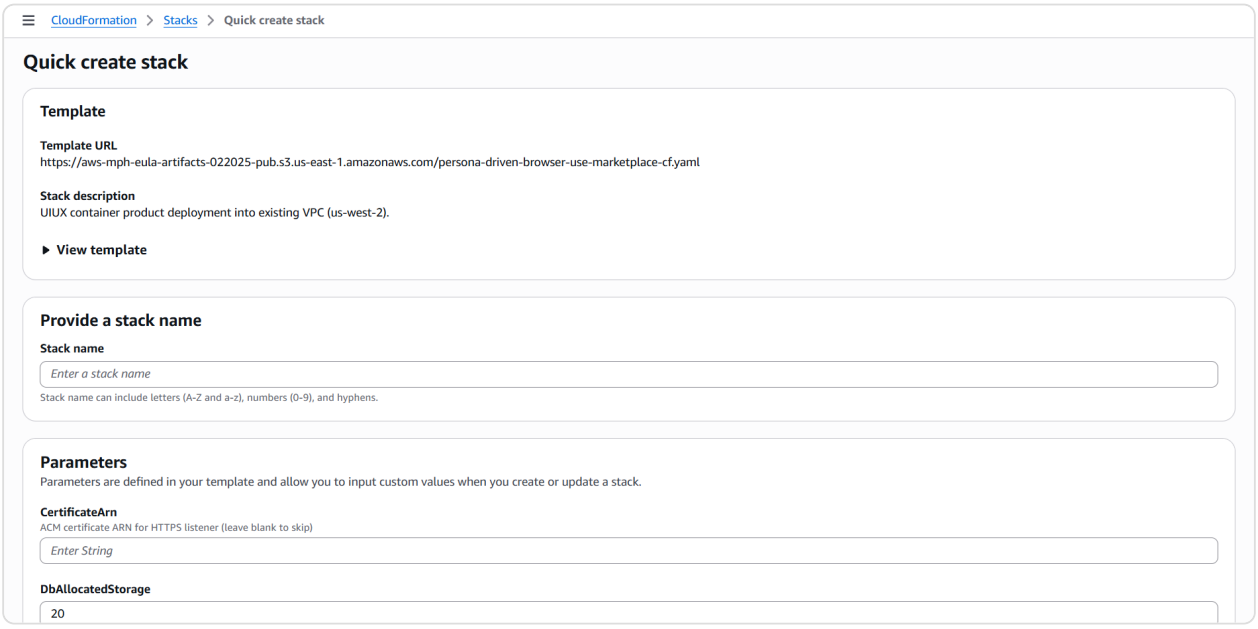
2 **Accept** the End User License Agreement and click *Continue to Configuration*.


About
Categories
Delivery Methods
Solutions
Resources
Become a Channel Partner
Sell in AWS Marketplace
Amazon Web Services Home
Help
AWS Marketplace > Persona Runner: AI-Powered UX Testing Platform > Subscribe to Persona Runner: AI-Powered UX Testing Platform
Subscribe to Persona Runner: AI-Powered UX Testing Platform Info
To create a subscription, review the pricing information and accept the terms for this software.
Existing subscription detected
You already have a subscription for this product.
View Subscription
Product details
Product name
Persona Runner: AI-Powered UX Testing Platform
Deployed on AWS
Product ID
prod-sk5hdmmdmwa22
Offered by
Mphasis
Offer summary
You can download the offer, including its terms and conditions, before making a purchase. If configuration is required, configure the offer before downloading.
Offer ID
offer-xzskg7jclqnq2
Offer type
Public offer
Offer availability
Purchased
Agreement ID
agmt-et7vwjgomk8j0vrbqmvtnry81
Date of purchase
Apr 28, 2026
You've already accepted this offer
Your AWS Marketplace agreement was created. You can launch your software or Manage subscriptions.
Launch your software
Pricing details
AWS Marketplace charges for the product based on your usage. Subscriptions have no end date and can be canceled at any time. Additional AWS infrastructure costs

3 **Note the Container Image URI** shown on the configuration page. Select the **ECS** service option and the latest version (**BrowserUseUXTestingPlatform_v1.0**). You will use the CloudFormation deployment template link provided on this page.



4 Click the **CloudFormation** link under *Deployment templates* to open the CloudFormation console with the template pre-loaded, or deploy manually using the template URL.



Step 2 — Prepare Your Secrets Manager Secret

The ECS container reads your database credentials at runtime from an existing Secrets Manager secret. You must create this secret **before** deploying the CloudFormation stack and provide its ARN as the `RuntimeSecretArn` parameter.

> 🔒 **Important:** The secret must use **exactly** these JSON key names — the application will fail to connect to the database if any key is missing or misspelled.

Required Secret JSON Structure

The ECS container reads the following database credentials from Secrets Manager at startup:

```
{
  "DB_HOST":     "your-db.xxxx.us-west-2.rds.amazonaws.com",
  "DB_USER":     "uiux_user",
  "DB_PASSWORD": "your_database_password",
  "DB_NAME":     "uiux",
  "DB_PORT":     "5432"
}
```

> 🔒 **Security Note:** These credentials are stored encrypted in AWS Secrets Manager using your default KMS key. The ECS service injects them securely at container startup — they never appear in environment variables, logs, or task definitions.

Create the Secret (AWS CLI)

Run the command below, replacing placeholders with your actual RDS endpoint and password:

```
aws secretsmanager create-secret \
    --name "ux-test/app/runtime" \
    --secret-string '{
      "DB_HOST":     "your-db-instance.c9akciq32.us-west-2.rds.amazonaws.com",
      "DB_USER":     "uiux_user",
      "DB_PASSWORD": "YourSecurePassword123!",
      "DB_NAME":     "uiux",
      "DB_PORT":     "5432"
    }' \
    --region us-west-2
```

Copy the returned `ARN` from the output — you will enter it as the `RuntimeSecretArn` CloudFormation parameter in Step 3.

Step 3 — Deploy the CloudFormation Stack

Option A: AWS Console (Recommended)

1. Open the **CloudFormation** console and click *Create stack* → *With new resources*, or use the CloudFormation link from the AWS Marketplace launch page.

2. The template URL is pre-loaded: `https://aws-mph-eula-artifacts-022025-pub.s3.us-east-1.amazonaws.com/persona-driven-browser-use-marketplace-cf.yaml`

3. Enter a unique **Stack name** (e.g., `persona-runner-prod`).

4. Fill in all parameters (see the **Parameters Reference** section below).

5. On the *Capabilities* page, check **I acknowledge that AWS CloudFormation might create IAM resources with custom names**.

6. Click **Submit** and wait for status `CREATE_COMPLETE` (~5–10 minutes).

CloudFormation Parameters Reference

General

Parameter	Default	Required	Description
`ProjectName`	`container-product`	No	Prefix for all created AWS resources.
`EnvironmentName`	`prod`	No	Environment suffix (e.g., prod, staging).

Networking

Parameter	Required	Description
`VpcId`	Yes	VPC ID with DNS resolution enabled. Must contain both public and private subnets.
`PublicSubnetA`	Yes	First public subnet for the ALB and ECS tasks.
`PublicSubnetB`	Yes	Second public subnet (different AZ) for the ALB and ECS tasks.
`PrivateSubnetA`	Yes	First private subnet for the RDS PostgreSQL instance.
`PrivateSubnetB`	Yes	Second private subnet (different AZ) for RDS.
`ExistingSecurityGroupId`	Yes	Security group for ECS Fargate tasks. Must allow inbound TCP 8000.

Database — RDS (Created by the Stack)

Parameter	Default	Required	Description
`DbName`	`uiux`	No	PostgreSQL database name.

Parameter	Default	Required	Description
`DbUser`	`uiux_user`	No	PostgreSQL master username.
`DbPassword`	—	Yes	PostgreSQL master password (NoEcho — masked in console).
`DbAllocatedStorage`	`20`	No	Initial RDS storage in GB (gp3).
`DbMaxAllocatedStorage`	`1000`	No	Maximum auto-scaling storage in GB.

Credentials & Secrets

Parameter	Required	Description
`RuntimeSecretArn`	Yes	ARN of a pre-existing Secrets Manager secret containing database credentials (`DB_HOST` , `DB_USER` , `DB_PASSWORD` , `DB_NAME` , `DB_PORT`). See Step 2.

IAM Roles

Parameter	Required	Description
`ExecutionRoleArn`	Yes	ECS Task Execution Role ARN. Requires `AmazonECSTaskExecutionRolePolicy` and `secretsmanager:GetSecretValue` .
`TaskRoleArn`	Yes	ECS Task Role ARN. Requires Bedrock `InvokeModel` , Marketplace `MeterUsage` , and AgentCore permissions.

Compute & Optional

Parameter	Default	Required	Description
`DesiredCount`	`1`	No	Number of ECS Fargate tasks to run.
`CertificateArn`	*(blank)*	No	ACM certificate ARN for HTTPS on the ALB. Leave blank for HTTP only.

Step 4 — Verify the Deployment

1. **Verify the stack deployed successfully** — open the CloudFormation console, select your stack, and confirm the status shows `CREATE_COMPLETE`.

2. **Update Secrets Manager** — copy the `RdsEndpoint` value from the CloudFormation stack **Outputs** tab and update your Secrets Manager secret with the actual RDS endpoint:

```
aws secretsmanager update-secret \
    --secret-id "ux-test/app/runtime" \
    --secret-string '{
      "DB_HOST":     "<RdsEndpoint from Outputs>",
      "DB_USER":     "uiux_user",
      "DB_PASSWORD": "<your DB password>",
      "DB_NAME":     "uiux",
      "DB_PORT":     "5432"
    }'
```

Then force a new ECS deployment to pick up the updated secret:

```
aws ecs update-service \
    --cluster <ECSClusterName from Outputs> \
    --service <ECSServiceName from Outputs> \
    --force-new-deployment
```

3. **Check the ECS service** — open the ECS console, find the cluster, and confirm the service has `runningCount: 1` with a `HEALTHY` task.

4. **Access the application** — find the `ALBDNSName` in the stack Outputs. Open it in your browser.

5 **Run your first test** — in the Persona Runner web interface:

1. Create a project with a name and target website URL.
2. Select a persona (e.g., "Elderly User").
3. Enter a task description (e.g., "Find the pricing page and request a demo").
4. Click **Run Test** and watch the live execution logs stream in real time.

Built-In Personas

Persona Runner includes four pre-configured user personas, each simulating a distinct user profile with realistic interaction patterns:

Persona	Age Group	Key Characteristics
Elderly User	70+	Limited digital literacy, requires large text and high contrast, slower interactions with difficulty on small targets, prefers linear step-by-step navigation, hesitates before actions
Color-Blind User	Any	Red-green color deficiency, struggles with color-only information, relies on patterns/labels/icons instead of color, frustrated by color-only charts and red/green error indicators
University Student	18–25	Digital native with high literacy, speed-focused with multi-tab research, uses search and filters, persists through multi-step tasks, frustrated by slow-loading sites
Cognitive/Dyslexic User	Adult	Difficulty with reading speed and attention, struggles with serif fonts and dense paragraphs, prefers structured menus with headings, needs simple clear instructions

Each persona generates realistic interaction patterns — **rage clicks**, **hesitation pauses**, **slow scrolling**, **search-preference biases** — that surface real-world UX issues traditional testing misses.

Security & Encryption

Persona Runner is designed with security as a top priority, adhering to the AWS Well-Architected framework and least-privilege access rules.

Data Storage & Encryption

> **Encryption Configurations**
>
> - **AWS Secrets Manager:** Database credentials are stored encrypted in Secrets Manager using default or custom AWS KMS keys. The ECS container reads credentials via the native ECS secrets injection mechanism; plaintext credentials are never written to environment variables in the task definition, files, or logs.
> - **RDS Database storage:** Database storage is encrypted at rest natively by AWS RDS using KMS keys (`StorageEncrypted: true`). Performance Insights is enabled for monitoring.
> - **RDS Network Security:** The RDS instance is deployed in private subnets with `PubliclyAccessible: false` . A dedicated RDS security group permits inbound PostgreSQL (TCP 5432) only from the ECS security group, with all egress denied.
> - **Credentials Rotation:** Credentials in Secrets Manager can be rotated manually or via automatic rotation. The ECS service picks up updated secrets on the next task launch or forced redeployment.

Security Best Practices Applied

- **Secrets Manager** — Database credentials are stored encrypted and injected at runtime via ECS native secrets integration. They never appear in the task definition, CloudFormation exports, or CloudWatch logs.
- **Least-privilege IAM** — Each role (ECS task, ECS execution) has only the minimum permissions it needs.
- **Non-root container** — The application runs as `appuser` (UID 10001) inside the container with no elevated privileges.
- **No public database** — RDS is deployed in private subnets only. A dedicated security group denies all egress from RDS.
- **No embedded credentials** — The Docker image contains no secrets. All sensitive values are injected at runtime.
- **RDS protection** — `DeletionProtection: true` and `DeletionPolicy: Snapshot` prevent accidental data loss. Automated backups are retained for 7 days.

- **Health checks** — ECS performs readiness checks every 30 seconds via a dedicated healthcheck script.

Environment Variable Security Guide

The table below shows every environment variable used by the application, its sensitivity level, and how it is stored and accessed.

Variable	Sensitivity	Storage Method	Notes
`DB_HOST`	**HIGH**	Secrets Manager → ECS secrets	RDS endpoint. Injected at startup.
`DB_USER`	LOW	Secrets Manager → ECS secrets	Database username.
`DB_PASSWORD`	**HIGH**	Secrets Manager → ECS secrets	Database password. Never in task def.
`DB_NAME`	LOW	Secrets Manager → ECS secrets	Database name (default: uiux).
`DB_PORT`	NONE	Secrets Manager → ECS secrets	Database port (default: 5432).
`APP_PORT`	NONE	CloudFormation → ECS env	Application port (8000).
`HOST`	NONE	CloudFormation → ECS env	Server bind host (0.0.0.0).
`PORT`	NONE	CloudFormation → ECS env	Server bind port (8000).
`FETCH_SECRETS_AT_RUNTIME`	NONE	CloudFormation → ECS env	Enable Secrets Manager fetch at boot.
`METERING_ENABLED`	NONE	CloudFormation → ECS env	Enable Marketplace metering (true).

IAM Permissions Summary

The CloudFormation template references two pre-existing IAM roles. No AWS-managed policies with broad access should be attached beyond the minimum required.

Role	Purpose	Key Permissions
ECS Execution Role (`ExecutionRoleArn`)	ECS agent: pull image, write CloudWatch logs, read secrets	`AmazonECSTaskExecutionRolePolicy` + `secretsmanager:GetSecretValue` on your runtime secret ARN
ECS Task Role (`TaskRoleArn`)	Runtime identity of the application container	`bedrock:InvokeModel` (Claude Sonnet 4); `bedrock-agentcore:*` (BrowserClient); `aws-marketplace:MeterUsage`; `secretsmanager:GetSecretValue`

Service Quotas

Ensure your AWS account regional service quotas are configured appropriately before deploying the stack:

AWS Service	Quota Name	Required Limit	Action if exceeded
Amazon Bedrock	Inference Profile RPM for Claude Sonnet 4	Min 50 RPM	Request increase via Service Quotas
Amazon Bedrock	Inference Profile TPM for Claude Sonnet 4	Min 80,000 TPM	Request increase via Service Quotas
Bedrock AgentCore	Concurrent BrowserClient sessions	Min 5 concurrent	Request increase if running parallel tests
Amazon ECS	Fargate tasks per region	Min 2 concurrent	Standard is 10+; increase if needed
AWS Secrets Manager	GetSecretValue API req/sec	Default (10,000/sec) sufficient	No action needed

Resource Pricing Estimates

Deploying this product launches additional paid AWS infrastructure within your AWS account. Estimated monthly costs under normal operations are detailed below:

AWS Resource	Pricing Type	Est. Base Cost	Usage Dimension Notes
Amazon RDS PostgreSQL	Instance hours (db.t3.micro)	~$15 / month	Single-AZ. Scale up for production. Multi-AZ doubles cost.
ECS Fargate Task	vCPU + Memory (1 vCPU, 2 GB)	~$30 / month	Charged per second of running task time.
Application Load Balancer	Hourly + LCU-hours	~$18 / month	Fixed hourly fee + data processing.
AWS NAT Gateway	Hourly + data processed	~$32 / month	Private subnet outbound. $0.045/GB data.

AWS Resource	Pricing Type	Est. Base Cost	Usage Dimension Notes
AWS Secrets Manager	Secret count + API calls	~$0.40 / month	$0.40/secret + $0.05 per 10K calls.
Amazon Bedrock	Claude Sonnet 4 API usage	Pay-as-you-go	~$0.003/1K input, $0.015/1K output tokens.
Bedrock AgentCore	BrowserClient session time	Pay-as-you-go	Charged per minute of browser session.
CloudWatch Logs	Ingestion + storage	~$2 / month	$0.50/GB ingested + $0.03/GB stored.

Usage Billing & Metering

Persona Runner is an AWS Marketplace **Usage-Based** product. Billing is calculated based on the number of successful pipeline runs (test executions) triggered.

> ℹ️ **Note:** Metering only occurs if `METERING_ENABLED=true` (the default). If disabled, the platform will function normally but no marketplace usage will be recorded (useful for internal testing).

How Metering Works:

1. Each completed test run records **1 unit** to the `dimension_table` in the Amazon RDS PostgreSQL database under the dimension name `"Inference"`.
2. A **background thread** runs every 3600 seconds (1 hour), aggregates unbilled dimensions, and calls `aws-marketplace:MeterUsage` to report consumption.
3. After successful reporting, records are marked `billing_done = TRUE`.
4. Metering is **non-blocking** — failures are logged but never break a test run.

Troubleshooting

ECS task keeps restarting / fails to start

- Check **CloudWatch logs** for the ECS task (log group matches your `ProjectName-EnvironmentName-backend-api` pattern).
- Confirm the ECS **execution role** has `secretsmanager:GetSecretValue` on your `RuntimeSecretArn`. An `AccessDeniedException` in logs means the role lacks this permission.
- Confirm the ECS **subnets have NAT Gateway routes** (ECR image pulls require outbound HTTPS 443).
- Verify **Amazon Bedrock model access** is enabled for Claude Sonnet 4 in your account/region.
- Confirm the Secrets Manager secret contains all required keys (`DB_HOST` , `DB_USER` , `DB_PASSWORD` , `DB_NAME` , `DB_PORT`).

Application returns "database connection failed"

- Verify the **RDS endpoint** in your Secrets Manager secret matches the actual RDS endpoint from the CloudFormation stack Outputs (`RdsEndpoint`).
- Confirm the ECS security group is the one specified in `ExistingSecurityGroupId` — the template automatically creates an RDS security group that permits TCP 5432 from this group.
- Confirm the RDS instance is in **Available** state (check the RDS console).
- The application runs database migrations automatically (`schema.sql`) at startup. Check logs for migration errors.

Browser agent fails with "AgentCore connection error"

- Confirm **Bedrock AgentCore** access is enabled in your AWS account and region.
- Verify the ECS **Task Role** has permissions for `bedrock-agentcore:*` .
- Ensure ECS subnets have **outbound internet access** (NAT Gateway) for AgentCore API calls.

ALB health checks failing

- The health check path is `/` with a 30-second interval. Allow the **start period of 60 seconds** for the application to initialize and run migrations.
- Confirm the ECS security group allows **inbound TCP 8000** from the ALB.
- Check CloudWatch logs for application startup errors.

Bedrock inference error ("ValidationException" or throttling)

- Ensure **Amazon Bedrock model access** is enabled for Anthropic Claude Sonnet 4. Open the Bedrock console → *Model access* and enable the model.
- For throttling errors, the pipeline retries automatically. For persistent issues, request a Bedrock service quota increase.

No metering data appearing

- Confirm `METERING_ENABLED=true` in the ECS task environment.
- Verify the ECS **Task Role** has `aws-marketplace:MeterUsage` permission.
- The metering thread sends data hourly — wait at least 1 hour after a test run.
- Check CloudWatch logs for metering-related errors.

Secret read fails at runtime

- The Secrets Manager secret must contain all eight keys exactly as named. Missing or differently named keys will cause a runtime error.
- Verify with:
  `aws secretsmanager get-secret-value --secret-id <YOUR-ARN> --query SecretString --output text`

Support

For issues with this AWS Marketplace product, use the **Support** link on the Marketplace product page. When contacting support, provide:

- Your **CloudFormation stack name** and **AWS region**.
- The relevant **CloudWatch log streams** (ECS task logs).
- The **error message** and any relevant screenshots from the web UI.
- Your **ECS task ID** (found in the ECS console under your cluster's tasks).

> ✅ Before contacting support, check the **Troubleshooting** section above — most common issues are covered there.
