SSD Group Deployment Report – Dizzy's Disease

**Prepared for Secure Software Development – Group Deployment Assignment**Generated: 2025-10-25 23:51 UTC

# Purpose

Document the deployment of the Dizzy's Disease application to a shared staging environment, and provide access instructions and evidence for peer security evaluation.

# Deployment Overview

|  |  |
| --- | --- |
| Staging URL | https://thankful-wave-09a5b000f.3.azurestaticapps.net/ |
| Hosting Platform | Azure Static Web Apps (Free tier, East US 2) |
| Backend | Azure App Service (https://dizzy-api-app.azurewebsites.net/) |
| Database | Azure PostgreSQL Flexible Server – dizzy-db-57a0 (database 'dizzy') |
| Build Source | Godot 4.4 HTML export located at capstone/build/web/ |
| CI/CD | .github/workflows/deploy-static-web-app.yml with action=upload |

# Rubric Alignment

The table below maps rubric criteria to delivered evidence.

|  |  |  |
| --- | --- | --- |
| Criterion | Evidence | Notes |
| Operation of Deployed System | Staging build live at https://thankful-wave-09a5b000f.3.azurestaticapps.net/. Registration/Login flows verified; offline mode available. | Manual testing recorded after deployment (browser + Playwright). |
| Access to Source Code | Classroom repo: https://github.com/Steve-at-Mohawk-College/capstone-project-Jinphinity Personal repo: https://github.com/Jinphinity/capstone | Both repositories include Godot project, FastAPI backend, and deployment workflow. |
| Access Instructions for Ethical Hackers | Step-by-step tester instructions provided later in this document (login, API health, offline mode). | Includes staging URL, API endpoint, and dev tools guidance. |
| Document Format | Deliverables include this DOCX plus supporting artifacts in docs/ssd/deliverables. | Structured with tables/lists as requested. |

# Source Code Repositories

|  |  |  |
| --- | --- | --- |
| Repository | URL | Notes |
| Classroom | https://github.com/Steve-at-Mohawk-College/capstone-project-Jinphinity | Primary SSD submission repository (GitHub Classroom). |
| Personal mirror | https://github.com/Jinphinity/capstone | Mirrors same codebase for individual development/testing. |

# Tester Instructions

Follow this checklist when evaluating the deployment:

1. Open https://thankful-wave-09a5b000f.3.azurestaticapps.net/ (Chrome/Edge recommended).
2. Register a new account or log in with test credentials; backend stores bcrypt hashes only (no email verification).
3. After login, exercise Continue/New Game, inventory, and market flows. Use 'Skip Login' to validate offline mode.
4. Open DevTools → Network and confirm requests hit https://dizzy-api-app.azurewebsites.net/ (e.g., /auth/login, /market/prices).
5. Check API health: https://dizzy-api-app.azurewebsites.net//health (expect JSON response with status='healthy').
6. Optional: redeploy locally using the command in Operational Notes to verify build reproducibility.

# Evidence Summary

|  |  |
| --- | --- |
| API Health | curl https://dizzy-api-app.azurewebsites.net//health -> {'status': 'healthy', ...} |
| Successful Actions Runs | https://github.com/Jinphinity/capstone/actions/runs/18809305400 https://github.com/Steve-at-Mohawk-College/capstone-project-Jinphinity/actions/runs/18809048337 |
| Manual Verification | Registration/login verified post-deploy; Playwright console captures stored in deliverables folder. |

# Operational Notes

**Manual redeploy command:**swa deploy ./capstone/build/web \  
 --app-name dizzy-disease-swa \  
 --resource-group DizzySWA-RG \  
 --subscription-id eedd13b0-fd70-45bc-99a0-7d1be4d779ec \  
 --env production  
Backend container image: dizzyacr57a0.azurecr.io/dizzy-api:latest  
Database connection: postgresql://dbadmin@dizzy-db-57a0/dizzy?sslmode=require

# Next Steps

Attach screenshots or penetration test findings to docs/ssd/deliverables/ as additional SSD artifacts are produced.