Maintenance Manual

Table of Contents

---

1. System Overview

1.1 System Description

The Moral Decisions project supports “AI + Human Exploration of Daily Moral Decisions” via two interactive websites:

- **Moral Profile Website:** Scenario-based moral mini-games (inspired by online forums) to help users explore common moral dilemmas.
- **Opinion Survey Website:** Collects real moral choices and generates personalized feedback reports based on decision patterns.

The goal is to leverage AI–human collaboration to build a data-driven platform for everyday, nuanced moral reasoning, creating a scalable foundation for ethical AI research.

Tech stack and deployment highlights (from the LandingSite repo and linked resources):

Deployment: Azure Web App (Docker supported) Frontend: Next.js (with API Routes; index page may directly access DB to simplify backend) Backend: NestJS (separate from frontend; interacts via REST API) API: Two key endpoints Deliver survey questions by studyId Receive survey responses by prolificId (documentation also mentions “profilicId”; use actual implementation in code) Version Control: Git (GitHub) Dev environment: Unified via server + Docker Testing: Postman for API testing Audience and impact:

Users: Global participants Beneficiaries: Researchers in AI ethics, moral psychology, and social computing Client: Ziyu Chen (Computational Media Lab) Reference repository: https://github.com/24-S1-2-C-Moral-Decisions/LandingSite

1.2 Key Components

- **Frontend:** Next.js
    - Provides UI/UX for Moral Profile and Opinion Survey
    - Uses Next.js API Routes for some data reads (especially index), reducing backend overhead
    - May include SSR/ISR and build cache (.next)
- **Backend:** NestJS
    - Decoupled from frontend; exposes REST APIs
    - Communicates with frontend via API calls

- **Database:** MangoDB
- **API:**
    - GET /api/survey?studyId=: returns survey by studyId
    - POST /api/response: accepts responses with prolificId/profilicId and answer payload
    - Actual paths/params must follow code in the repos
- **Deployment:**
    - Azure Web App (Linux), container-friendly
    - Environment variable management (PORT, DATABASE_URL, etc.)
- **Tooling:**
    - Git (GitHub)
    - Docker (local and/or cloud images)
    - Postman (API testing)

1.3 Components Requiring Maintenance

- Azure Web App: availability, scaling, logs, monitoring
- Next.js app: builds, dependencies, env vars, ISR/cache
- NestJS service: API uptime, dependencies, configuration
- Database: backups, indexing, health
- Container images: base image updates, security patches, rebuilds
- Logging and monitoring: retention, rotation, alert thresholds
- Secrets and credentials: env vars, key rotation
- Postman collections: keep in sync with API contracts

---

2. Regular Maintenance Tasks

2.1 Maintenance Schedule Overview

Frequency	Tasks	Estimated Time
Daily	Health checks, error log review, resource monitoring	15–25 min
Weekly	DB backup, log management, dependency checks and minor updates	45–60 min
Monthly	System health review, DB optimization, security review	80–105 min
Quarterly	DR drills, security assessment/pen test, Node LTS/base image upgrades	8-20 hrs

2.2 Daily Maintenance Tasks

Task 2.2.1: Check System Health

Frequency: Daily **Estimated Time:** 5-10 minutes **Priority:** High

Steps:

1. Visit homepage and key pages (survey page, results page), expect HTTP 200.
2. Probe core APIs:
    - GET /api/survey?studyId=
    - POST /api/response with minimal valid payload
3. Check Azure Web App runtime status and error trend in the last hour

Expected Results:

- Pages and APIs return 2xx.
- No new fatal errors (e.g., spike in 5xx).

What to do if issues found:

- Grab logs and triage (see Logs and Common Issues).
- Consider restart/rollback during off-peak; create a GitHub issue to track.

Task 2.2.2: Review Error Logs

Frequency: Daily **Estimated Time:** 10-15 minutes **Priority:** High

Steps:

1. Access log files at `Azure App Service: /home/LogFiles`
2. Scan error/warn entries: MangoDB connection errors, timeouts, memory errors.
3. Summarize recurring and new exceptions; add to issue list.
4. For known issues, confirm mitigation/fix status.

Expected Results:

- No critical errors
- Routine warnings do not affect functionality.

What to do if issues found:

- Correlate with commits/config changes, rollback or fix.
- Escalate priority and notify owners.

2.3 Weekly Maintenance Tasks

Task 2.3.1: Database Backup

Frequency: Weekly (Every Monday at Monday 10:00 am) **Estimated Time:** 45–120 minutes **Priority:** Critical

Steps:

1. Confirm DB engine and connection string (DATABASE_URL)

2. Create backup (choose per actual DB)
3. Upload to backup storage (recommend Azure Blob Storage) with date/version tags.

Backup Location:

```
Azure Blob Storage: <STORAGE_ACCOUNT>/<CONTAINER>/backups/moral-decisions/
```

Backup Commands:

```
az storage blob upload \
  --account-name <STORAGE_ACCOUNT> \
  --container-name <CONTAINER> \
  --file "$FILE" \
  --name "backups/moral-decisions/$FILE"
```

Verification:

```
az storage blob list \
  --account-name <STORAGE_ACCOUNT> \
  --container-name <CONTAINER> \
  --prefix "backups/moral-decisions/" \
  --output table

# Sample restore validation (Postgres)
pg_restore -l "$FILE" >/dev/null
```

Task 2.3.2: Log File Management

Frequency: Weekly **Estimated Time:** 10 minutes **Priority:** Medium

Steps:

1. Check log file sizes at `[log-directory]`
2. Archive logs older than [7 days]
3. Delete logs older than [3 months]
4. Verify log rotation is working

Log Locations:

- Application logs: `[path]`
- Error logs: `[path]`
- Access logs: `[path]`

Commands:

```
du -sh /home/LogFiles/*

# Archive > 7 days
find /home/LogFiles -type f -mtime +7 -name "*.txt" -print \
  -exec tar -czf "/home/LogFiles/archive_$(date +%F).tgz" {} +

# Delete > 90 days
find /home/LogFiles -type f -mtime +90 -delete
```

Task 2.3.3: Dependency Updates Check

Frequency: Weekly **Estimated Time:** 15 minutes **Priority:** Medium

Steps:

1. Check for security updates
2. Review update changelog
3. Test updates in development environment
4. Schedule update deployment if needed

Commands:

```
npm ci
npm outdated
npm audit
# Selective upgrades
npm install <pkg>@<version>
npm run build && npm test
```

2.4 Monthly Maintenance Tasks

Task 2.4.1: Full System Health Review

Frequency: Monthly (First Monday of month) **Estimated Time:** 30-45 minutes **Priority:** High

Steps:

1. Review p95/p99 latency, error rate, throughput, and capacity (CPU/mem/connections).
2. Review deployment changes and open risks.
3. Evaluate cost and scaling plan (Azure SKU).
4. Update runbook and known issues.

Task 2.4.2: Database Optimization

Frequency: Monthly **Estimated Time:** 20-30 minutes **Priority:** Medium

Steps:

1. Analyze database performance
2. Rebuild indexes if needed
3. Clean up orphaned data
4. Verify database integrity

Commands:

```
-- PostgreSQL
VACUUM (VERBOSE, ANALYZE);
REINDEX DATABASE moral_decisions;
-- Use EXPLAIN ANALYZE to optimize key queries

# MongoDB (in maintenance window)
db.runCommand({ compact: "<collection>" })
```

Task 2.4.3: Security Review

Frequency: Monthly **Estimated Time:** 30 minutes **Priority:** High

Steps:

1. Review access logs for suspicious activity
2. Check security patches available
3. Review user access permissions
4. Update firewall rules if needed

---

3. Common Issues and Solutions

3.1 Application Not Starting

Symptoms:

- Deployment “succeeds” but container/process exits immediately
- Initial page/API

Possible Causes:

1. Missing required env vars (DATABASE_URL, PORT, API keys, etc.)
2. Port binding issue (not listening on $PORT or missing WEBSITES_PORT in container)
3. Node version/build mismatch, missing dependencies, build failure

Diagnosis Steps:

```
az webapp log tail -n <WEBAPP_NAME> -g <RESOURCE_GROUP>
az webapp config appsettings list -n <WEBAPP_NAME> -g <RESOURCE_GROUP>
```

```
echo $PORT
```

Solutions:

Solution 1: [Solution Name]

```
# Ensure the app listens on process.env.PORT
# Next.js defaults to 3000; on Azure use PORT env var
# For custom containers, set WEBSITES_PORT=<PORT> if needed
```

Solution 2: [Solution Name]

```
npm ci
npm run build
npm run start
# For monorepos, ensure correct working directory and output paths
```

Prevention:

- Add startup and health checks in CI
- Validate env var completeness pre-deploy
- Pin Node version (.nvmrc or engines) and lockfiles

3.2 Performance Issues

Symptoms:

- Slow response times
- High CPU usage
- High memory usage

Possible Causes:

1. Missing DB indexes or N+1 queries
2. SSR/render hot paths with cache misses
3. Undersized instances or network jitter

Diagnosis Steps:

```
# System resources
top / htop

# Next.js performance
npm run build

# Database
```

```
-- Postgres: EXPLAIN (ANALYZE, BUFFERS) <query>;
-- MongoDB: db.collection.find({...}).explain("executionStats")
```

Solutions:

Solution 1: Clear Cache

```
rm -rf .next && npm run build
# If ISR/manual invalidation is configured, refresh appropriately
```

Solution 2: Restart Application

```
az webapp restart -n <WEBAPP_NAME> -g <RESOURCE_GROUP>
```

Solution 3: Scale Resources

- Upgrade Azure plan or increase instance count
- Add DB compute/IO or connection pooling **Prevention:**
- Regular monitoring
- Scheduled restarts
- Resource optimization

3.3 Database Connection Problems

Symptoms:

- "Cannot connect to database" errors
- Timeout errors
- Authentication failures

Possible Causes:

1. Database service not running
2. Wrong credentials
3. Network connectivity issues
4. Connection pool exhausted

Diagnosis Steps:

```
# PostgreSQL
pg_isready -h <HOST> -p <PORT> -d <DB> -U <USER>

# MongoDB
mongosh "<DATABASE_URL>" --eval 'db.runCommand({ping:1})'
```

```
# App logs
az webapp log tail -n <WEBAPP_NAME> -g <RESOURCE_GROUP>
```

Solutions:

Solution 1: Restart Database Service

```
Use platform-specific controls (Azure Database, VM, self-managed)
```

Solution 2: Verify Credentials

```
Update DATABASE_URL from secret store/env vars
Enforce least privilege; avoid superuser in apps
```

Solution 3: Check Network

```
nc -vz <HOST> <PORT>
# Confirm Azure firewall/privatelink rules
```

Prevention:

- Monitor database health
- Regular backups
- Connection pool configuration

3.4 Common Error Messages

Error: "listen EADDRINUSE: address already in use"

Meaning: Port in use **Cause:** Not using $PORT or duplicate process **Solution:** Listen on process.env.PORT; free port or restart instance

Error: "ECONNREFUSED/ETIMEDOUT connecting to DB"

Meaning: DB unreachable/timeout **Cause:** Firewall/allowlist, wrong host/port/credentials **Solution:** Verify DATABASE_URL, allow network, retry and check pooling

Error: "Cannot find module '...'" or "MODULE_NOT_FOUND"

Meaning: Missing dependency or incomplete build artifact **Cause:** Install failure, lock mismatch, wrong build path **Solution:** npm ci && npm run build; fix working dir and build outputs

---

4. Contact Information

4.1 Team Contact Details

Primary Contact:

- Name: Shutong Li
- Email: u7768183@anu.edu.au
- Phone: NA
- Availability: Weekdays 09:00–18:00 (local time)

Secondary Contact:

- Name: Shutong Li
- Email: u6825537@anu.edu.au
- Phone: NA
- Availability: Weekdays 09:00–18:00 (local time)

4.2 Stakeholder Contact Information

Client/Stakeholder:

- Name: Ziyu Chen
- Organization: ANU CSS
- Email: Ziyu.Chen@anu.edu.au
- Phone: NA

4.3 Repository and Documentation Links

Main Repository:

- GitHub: https://github.com/24-S1-2-C-Moral-Decisions/LandingSite
- Branch: main

LandingSite Documentation:

- Location: `LandingSite/25S2/`
- Meeting Minutes: `LandingSite/25S2/Minutes/`
- Deliverables: `LandingSite/25S2/Delivery/`

Other Documentation:

- System Architecture: [link]
- API Documentation: [link]
- Database Documentation: [link]
- Deployment Guide: [link]

4.4 Emergency Contacts

For Critical Issues (System Down, Security Breach):

1. Contact: Lingziluo Xiong, Zihao Li

2. If unavailable: YU MA, Fei Li
3. Escalation: Ziyu Chen (Client, ANU CSS)

Support Hours:

- Regular support: Weekdays 09:00–18:00
- Emergency support: 24x7 (per on-call roster and emergency contact availability)

Appendix: Maintenance Checklists

Daily Checklist

- [ ] Check system health
- [ ] Review error logs
- [ ] Monitor system resources
- [ ] Verify backups completed (if scheduled)

Weekly Checklist

- [ ] Perform database backup
- [ ] Manage log files
- [ ] Check for dependency updates
- [ ] Review security logs
- [ ] Test backup restoration (monthly sample)

Monthly Checklist

- [ ] Full system health review
- [ ] Database optimization
- [ ] Security review
- [ ] Update documentation if needed
- [ ] Review and update this manual

Quality Checklist

- [x] Practical and actionable guidance
- [x] Clear troubleshooting steps
- [x] Useful for future maintainers
- [x] All contact information accurate
- [x] All sections complete
- [x] Reviewed by team