ECE 49595 - Verification & Validation

Fall 2025

Purpose of Assignment

- This assignment will build on your test planning and RVTM workshops to develop a complete verification and validation plan. Verification and validation are essential because they ensure your system is built correctly (verification) and built for the right purpose (validation). These practices mirror some of the ways that industry teams ensure product quality, traceability, and compliance with user requirements.

***NOTE**: Each **bolded** section below must be included in your submitted document, to organize and communicate your work.*

Objectives

Obj 1: The profile should show up on the phone within seconds of recognition via glasses. (latency of 1 second).

Obj 2: 0 unauthorized identification allowed, all users opted out of an event should not be processed or displayed to other users.

Obj 3: <10% recognition error for profiles that are displayed on the app, i.e. less than 5% out of all recognized profiles that are displayed are incorrect matches (not the same person).

Obj 4: 100% correctness of displayed fields for profiles that are opted in (even if the match is incorrect, the information connected to whoever the system matches to should be displayed).

Obj 5: ≥95% uptime and no more than 1 failure per 3 hours.

Prioritization

Product Capability	Priority	Brief Justification
Real-Time Recognition Pipeline (FR-1, NFR-1)	High	Directly tied to Objective 1 (≤1 sec latency) and Objective 3 (accuracy). “Algorithm Performance Issues” is a High technical risk in the risk register. Core to MVP and user experience.
Privacy & Consent Enforcement (FR-3, FR-6, NFR-2, NFR-3)	High	Supports Objective 2 (0 unauthorized identification). Privacy violations are rated Medium-High risk (Privacy Compliance Violation, User Consent Mismanagement). Essential for FERPA-aligned operation.
Recognition Accuracy & Correct Matching (FR-1)	High	Critical for Objective 3 (<10% incorrect matches). Risk register lists Algorithm Performance and API Dependency Failure as High risk items. Incorrect matches break trust and make the tool unusable.
Backend Reliability & System Stability (NFR-5, FR-5)	High	Supports Objective 5 (≥95% uptime, ≤1 failure per 3 hours). API Dependency Failure is a High stakeholder risk, and backend failure prevents all recognition. Critical during live events.
Event-Based Privacy Filters (FR-6)	High	Ensures cross-event isolation and prevents unauthorized recognition. Part of Objective 2 enforcement. Misconfiguration poses Medium-High operational risk.
Supabase RLS + Audit Logging (FR-3)	High	Core to consent enforcement and privacy compliance. Helps validate Objective 2 and Objective 4. Without RLS, unauthorized access becomes likely. Supports several “Must Have” SDP requirements.
Profile Context Card Accuracy (FR-4)	Mediu m	Supports Objective 4 (100% correct displayed fields). “Integration Bugs Between Systems” identified as Medium risk. Impacts credibility but does not break MVP-level functionality.

Integration Between FastAPI, AWS Rekognition, Supabase, and Frontend (FR-5)	Medium	Identified as a Medium project management risk ("Delayed API Integration"). Impacts latency and reliability, but failures can be tested and mitigated with mocks; still essential but not as high-risk as privacy.
UI/UX Frontend Responsiveness (FR-9)	Medium	Helps support usability and Objective 1 responsiveness at the UI layer. Less critical than backend correctness and privacy but affects perceived performance. SDP labels this as "Should Have."
Similarity Score Engine (FR-7)	Low	Not tied to any of the five V&V objectives. Not required for privacy, accuracy, latency, or uptime. Lower risk and enhancement-based. SDP classifies as Must Have but low impact on core system correctness.
Conversation Starter Generation (FR-8)	Low	Does not impact any quantitative V&V objective. Depends on OpenAI API (High risk) but failure does not break core functionality. Enhancement feature only.
Event Analytics, Tracking, Dashboards (FR-10–12)	Low	Out of scope for core V&V objectives. Failure has no impact on real-time recognition or privacy. Classified as Should/Could Have in the SDP.

Full RVTM

Requirements				Test				
Project Capability	**Req. ID**	**Requirement (Shall Statement)**	**Supporting Context**	**Test Case ID**	**Test Description**	**Impacted Components**	**Additional Comments**	**Objective Alignment**
Privacy Enf	ID-1	The system shall restrict recognition to only those that have opted in to the event to ensure that we are respecting privacy concerns.	This is done to ensure that all privacy concerns are being met	TC-1	We can test our solution with faces that are not in the database to see if it recognizes anyone and we can do this with 10 faces.	SupaBase and Rekognition	Important for ethics	Obj 2: 0 unauthorized identifications
Profile Display	ID-2	The system shall display user's key info within 1 second of identification	We need our solution to be as seamless as possible so low latency is key when they are networking	TC-2	We can measure 100 different faces to ensure that at least 95% of those profiles show up in less than 1 second of being recognized	Frontend UI	Better user experience	Obj 1: 1-second latency
Similarity Ranking	ID-3	The system shall accurately rank recognized users by shared attributes (major, organization, interests) to highlight most relevant matches first	Allows prioritization of meaningful encounters, supporting the project's goal of intentional networking	TC-3	Input test datasets with overlapping interests; confirm ranked order aligns with defined similarity algorithm	FastAPI Backend, Supabase Query Layer, iOS UI	Enhances personalization and efficiency	Not tied to primary objectives (Low Priority)
Consent Management and Data Privacy	ID-4	The system shall allow users to control what profile fields are visible and shall encrypt all stored facial and profile data at rest and in transit.	Upholds ethical standards and user autonomy; meets FERPA standards	TC-4	Verify user toggles correctly hide data fields; run penetration tests confirming AES-256 encryption and HTTPS support.	Supabase DB, AWS Rekognition, FastAPI API layer	Strengthens trust, transparency, and regulatory compliance	Obj 2 (privacy), Obj 4 (correct fields)

Recognition Accuracy	ID-5	The system shall correctly match ≥90% of identified profiles (≤10% incorrect matches).	Ensures reliability and usefulness during real interactions.	TC-5	Test with 100 enrolled faces; at least 90 must match the correct profile.	Rekognition Matching, FastAPI Pipeline	High-risk area affecting credibility.	Obj 3: <10% incorrect matches
Backend Identity Linking	ID-6	The system shall map each Rekognition Face ID to exactly one Supabase user profile.	Prevents mismatches that degrade accuracy.	TC-6	Index 30 faces; confirm each Face ID maps uniquely to correct Supabase record.	Supabase Profile Table, Rekognition IndexFaces	Supports core recognition pipeline integrity.	Obj 3: accuracy
Profile Metadata Display	ID-7	The system shall always display accurate profile fields for the matched user (0 display errors)	Accuracy of displayed information affects trust.	TC-7	For 50 users, verify UI displays correct fields (name, major, role).	React UI, Supabase Query Layer	Important for credibility.	Obj 4: 100% correct displayed fields
Consent-Based Field Visibility	ID-8	The system shall hide any profile field the user sets as private.	Ensures respect for user privacy preferences.	TC-8	Toggle settings; verify UI hides private fields and audit logs capture access.	Supabase RLS, UI Layer	Required for privacy-by-design.	Obj 2 & Obj 4
API Reliability & Uptime	ID-9	The system shall maintain ≥95% uptime during use, with ≤1 system failure every 3 hours.	Ensures stability for live events.	TC-9	Run 3-hour continuous load test; measure uptime and log API failures.	FastAPI Backend, Supabase, AWS Rekognition	Supports reliability goals for events.	Obj 5: ≥95% uptime
Resilience Under Load	ID-10	The system shall handle up to 20 concurrent users without performance degradation.	Required for event readiness and operational robustness.	TC-10	Perform load test with 20 simultaneous recognition requests; ensure zero crashes.	Backend, Rekognition, Supabase Connection Pool	Validates system scaling capabilities.	Obj 5: reliability
Event Boundary Control	ID-11	The system shall prohibit cross-event matching by isolating face collections.	Enforces event-scoped consent.	TC-11	Attempt match across two events; system must return 0 cross-event matches.	Rekognition Event Collections, Supabase	Critical for compliance	Obj 2: privacy protection

Test Approach

Unit Testing

- Focus: helpers, API logic, authentication, similarity algorithm, data validation.
- Tools: pytest, pytest-asyncio, Jest (for UI components).
- Automated: Yes (CI-gated).

Integration Testing

- Tests backend interactions with Supabase, Rekognition, and event-specific filtering.
- Uses mocked Rekognition for reliability.
- Tools: pytest + HTTPX test client.

System Testing

- End-to-end scenario testing: glasses -> backend -> ui.
- Tools: Playwright (frontend), staging environment, mock video streams.

Manual Tests

- Live smart-glasses tests.
- Usability walk-throughs.
- Stress and load tests.

Automated Tests

- All unit and integration tests.
- Most correctness and functional tests.
- Regression suite triggered via Github Actions.

Non-Functional Requirement Testing

- Performance: Latency measurements; recognition pipeline speed tests.
- Privacy: Opt-out + field visibility enforcement.
- Reliability/Uptime: Long-duration backend simulation.
- Usability: Real networking session simulation.

Regression Testing Approach

- Regression suit runs on every PR through GitHub Actions.
- After major UI changes, automated Playwright tests are executed.
- After backend schema changes, run full integration suite.

Validation Plan

5.1 User Acceptance Testing (UAT)
Approach
- Small-scale pilot at Purdue networking sessions.
- Recruit 10-15 student volunteers.
- Users will wear the glasses, scan others, and confirm:
    - Profiles appear quickly
    - Field information is correct
    - System respects privacy settings

Success Metrics
- >90% positive feedback on speed and usefulness
- <1 reported case of incorrect privacy exposure
- >85% of users state the experience improves networking

5.2 Usability & Performance Evaluations
Approach
- Controlled lab test with simulated event scenarios.
- Measure smartphone UI responsiveness, clarity, and user flow.
- Evaluate performance under:
    - Different lighting conditions.
    - Multiple simultaneous recognitions.
    - Weak Wi-Fi.

Success Metrics
- Profile display <1 sec (95% of trials).
- Recognition error rate <10%.
- <5% UI navigation errors.
- Users rate UI clarity >4/5.

5.3 Pilot/Prototype Testing Procedures
- Run a full-length 90-minute mock networking event.
- Test real-time constraints (uptime, failures per hour).
- Gather feedback through post-event surveys.

Success Metrics
- >95% uptime during pilot.
- <1 failure every 3 hours (scaled).
- >80% user satisfaction score.

Tools, Environments & Test Data Sets

To ensure the system behaves correctly and reliably, we will use a combination of automated testing tools, manual testing workflows, simulated environments, and realistic test data.q

Testing Tools (with Configuration)

- **pytest + pytest-asyncio**
  Used for backend unit and integration tests (API logic, database access, external service adapters).
  Configuration: We will use a `pytest.ini` file to define test paths (`tests/`), async support, and markers (e.g., `unit`, `integration`, `slow`). We will enable `pytest-cov` and configure the suite to fail if overall backend code coverage drops below **80%**. Tests will load environment variables from a `.env.test` file so they always run against a dedicated test database and test service endpoints, never production.
- **HTTPX / requests test clients**
  Used within pytest to exercise REST and WebSocket endpoints.
  Configuration: The base API URL will be injected via an environment variable, allowing the same tests to run against local, CI, or staging environments. Shared pytest fixtures will configure common headers (e.g., auth tokens, content type) so tests are consistent and easier to maintain.
- **Jest + React Testing Library**
  Used to test frontend components, hooks, and user interface behavior.
  Configuration: A `jest.config` file will define the `jsdom` test environment, test file patterns, setup files (for global mocks and providers), and path aliases. We will enable coverage collection and require at least **80%** line/branch coverage for the frontend in CI. Separate `npm`/`yarn` scripts (e.g., `test` for local watch mode and `test:ci` for single-run, no-watch) will align behavior with local vs CI usage.
- **Postman**
  Used for manual endpoint testing during development (sanity checks on new endpoints, debugging payloads, and headers).
- **GitHub Actions (CI/CD)**
  Used as the continuous integration backbone to enforce testing and quality checks on each push and merge to main branch. CD pipeline will only run after a successful merge to main deploying our app to ngrok.

Simulation & Emulation Environments

- **Smart glasses / camera simulation:** We will use either a MentraOS/SDK simulator or a mocked camera/video stream to emulate real-time input without requiring physical hardware, allowing repeatable tests of the recognition pipeline.
- **Staging cloud environment:** A separate staging backend and database will be deployed to mirror production configuration but with test credentials and data. This will be the main target for end-to-end and Playwright tests.
- **Local development environment:** Virtual environments will be used to ensure everyone is working with the same dependencies to run the frontend and backend locally when testing. A local database like sqlite with the same schema as production, along with

mock services for external APIs in testing.

Test Data Sets

- **Synthetic / anonymized face images:** Stored in a dedicated test collection, representing multiple users and conditions while avoiding the use of real attendee data.
- **Mocked user profiles:** A separate test database containing realistic but anonymized profiles (names, majors/departments, organizations, interests, etc.) to exercise profile lookup, ranking, and similarity logic.
- **Generated encounter logs:** Pre-populated encounter records (user IDs, timestamps, event IDs) to validate analytics, history views, and any gamification or statistics features.
- **Consent/privacy variants:** Test users with different consent flags and visibility settings (opted in, opted out, event-restricted) to ensure the system enforces privacy and retention rules correctly.

All test data will live in a `testdata/` folder in the same directory as the corresponding tests/ folder (for seed files).