ECE 49595 - Verification & Validation

Fall 2025

Purpose

This Verification and Validation (V&V) Plan defines the strategy used to ensure that *Stride* is both built correctly (verification) and built for the correct user needs (validation). The plan establishes measurable success criteria, testing approaches, traceability to functional and non-functional requirements, and validation methods involving real users.

Objectives

1. Verify Human Collision Detection Latency
    a. Objective: Verify that the system detects humans within 5 feet of the user and delivers feedback to the client device within 1 second for at least 97% of test instances.
    b. Linked Product Capability: FR-1

2. Verify Static Object Collision Detection Latency
    a. Objective: Verify that the system can detect objects within 10 feet of the user and deliver the feedback within 2 seconds for at least 97% of the test instances.
    b. Linked Product Capability: FR-2

3. Validate Indoor Navigation Success Rate
    a. Objective: Validate that the navigation system successfully guides a user from a starting point to the intended destination in at least 80% of trials.
    b. Linked Product Capability: FR-3

4. Verify Backend Scalability Under Normal Load
    a. Objective: Verify that the AWS backend maintains latency thresholds defined by FR-1 and FR-2 under normal operating load.
    b. Linked product capability: FR-1, FR-2

5. Verify Accessibility Interface Response
    a. Objective: Verify that the application correctly triggers audio voice prompts and haptic feedback patterns upon receiving navigation or obstacle data with a 90% success rate, while also supporting control via standard voice assistants.
    b. Linked Product Capability: FR-6, NFR-2

Prioritization

Product Capability	Priority (High, Medium, Low)	Brief Justification
Human Collision Detection (FR-1)	**High**	**Safety Critical:** Failure to detect a human within 1 second directly risks user injury. This aligns with Risk #2 (Severe Impact) regarding detection delays.
Object Collision Detection (FR-2)	**High**	**Safety Critical:** While objects are static, missing an obstacle within 10 feet poses a severe safety risk to the blind user, classified as a "Must Have" in the SDP.
Indoor Navigation Accuracy (FR-3)	**High**	**Core Functionality:** The primary purpose of "Stride" is navigation. If the user cannot reach their destination (Risk #3), the product fails its mission. This is a "Must Have" requirement.
Accessibility Interface (FR-6, NFR-2)	**High**	**User Blocker:** The target audience (visually impaired) relies entirely on audio/haptics. If this interface fails or isn't ADA compliant, the app is unusable.
Backend Scalability (FR-1/2 Support)	**Medium**	**Reliability:** Necessary to support the High priority safety features under load (Risk #4). While critical, it supports the primary functions rather than being a user-facing feature itself.

Battery & Performance (NFR-1)	**Medium**	**Usability:** A "Should Have" requirement. Excessive battery drain is a negative user experience but does not pose an immediate safety threat like collision failure.
User Authentication (FR-4, FR-5)	**Low**	**Operational:** Classified as "Could Have" in the SDP. While important for data security, the app can function locally or in a guest mode without critical failure.

Full RVTM

Product Capability	Req ID	Requirement Shall (Statement)	Supporting Context	Test Case ID	Test Description	Impacted Components	Additional Comments
Human Collision Detection	FR-1	The system shall detect humans within 5 ft and return a response within <= 1 second under normal load	Benchmark computation estimates ~400 ms;	TC-CD-H-01	Send 100 test videos with approaching humans at varying distances; verify that >= 97% responses <1s.	CV Model, Inference Endpoint, Mobile App	
Object Collision Detection	FR-2	The system shall detect static objects within 10 ft and return a response within <= 2 seconds.	Objects are stationary so there are looser latency requirements	TC-CD-02	Execute 100 object detection test runs; verify >= 97% responses <2 seconds	CV Model, Inference Endpoint, Mobile App	
Indoor Navigation and Guidance	FR-3	The system shall provide navigation (audio + haptics) and guide the user successfully >= 80% of attempts	Depends of integration of CV, and mapping	TC-NAV-03	Conduct 20 navigation trials inside BHEE; user must reach destination in >= 80% of attempts	Navigation logic, mapping, App UI, audio + haptics	
User Authentication	FR-4	The system shall authenticate a user within <= 1 second	Normal load 50 users	TC-AUTH-L-04	Perform 100 valid login attempts; measure response	Auth service, AWS cognito, backend, mobile	Must align with privacy

		for 95% of login attempts			time	app	requirements
User Signup	FR-5	The system shall create a new FaceID-backend user account within <= 1 seconds on onboarding.	Signup must occur before login	TC-AUTH-S-05	Perform 100 valid signup attempts; measure response time	Auth Service, AWS Cognito, backend, mobile app	Must align with privacy requirements
Accessible UI for Visually Impaired Users	FR-6	The system shall provide an ADA compliant UI operable through haptics, speech, and audio	Designed for blind users	TC-UI-06	Manually test, set destination, receive feedback, verify 90% success rate	Mobile App UI, Audio/Haptics Engine	Core accessibility requirement
UI for Sighted Users	FR-7	The system shall allow entering a destination and loading directions within <= 5 seconds	Assumes normal load and responsiveness	TC-UI-07	Test against set of destinations and verify that routing loads <= 5 seconds	Mobile App UI, Navigation Engine	
Batter and Performance	NFR-1	The application shall run with minimal battery drain and maintain	Expected long-term usage.	TC-PERF-08	Run app for 10 minutes; measure average battery % drop across multiple	Mobile App	

		optimal performance.			trials		
WCAG/ Accessibility Compliance	NFR-2	The system shall comply with WCAG 2.1 and support assistive technologies	Users must be able to interact fully via voice	TC-WCAG-0-9	Test all workflows using native voice assistant (Siri, Alexa, etc)	UI, Voice Interface	
Cross Platform Deployability	NFR-3	The application shall run on IOS and Android Devices	Ensure accessibility	TC-DEPLOY-10	Install and run on IOS and Android	Mobile App, React Native Layer	

Test Approach

Component	Strategy Description
Unit, Integration, and System Testing	**Unit Testing:** Individual React Native components and Python backend logic are tested using Jest and Pytest.
	Integration Testing: Verifies communication between the mobile application and AWS services (API Gateway, Lambda, Cognito, SageMaker). This includes testing the API Gateway endpoints, the handover of video frames to SageMaker, and the return of inference results to the app. The specific integration points include functionalities for user authentication through Amazon Cognito and database connections to retrieve map data.
	System Testing: Validates the full end-to-end workflow. This involves testing the complete loop from image capture to haptic/audio feedback. Key metrics such as latency will be verified through the system testing.
Manual vs Automated Testing	**Automated:** Latency and accuracy for collision detection will be automated by sending a dataset of about 100 videos from a client device to the backend. The response time will be measured without requiring a physical user. Login performance will also be tested via 100 automated attempts. We use 100 videos and attempts to verify the percentage of success.
	Manual: User interface and accessibility testing will be manual. This includes verifying that the app works with voice assistants, ensuring haptic feedback is perceptible, and conducting real-world navigation trials. All of these tests require physical users to ensure proper application functionality.
Non-Functional Requirement Verification	**Performance:** Battery life testing will be conducted by measuring the change in battery percentage over continuous 10-minute usage intervals.

	Accessibility: We will verify compliance with WCAG 2.1 standards and ensure compatibility with standard mobile voice assistants such as Siri and Google Assistant.
	Scalability: AWS CloudWatch metrics and alarms will be used to monitor system health and performance under simulated loads. This will ensure that the backend can handle concurrent requests without exceeding latency thresholds.
Regression Testing	Regression testing will be an iterative process and will be executed after CV model retraining and backend updates using previously validated datasets to ensure performance does not degrade.

Validation Plan

1. Validation Plan Intro

The validation plan is a critical component of our project's development process, as it demonstrates how the final system will perform for real users outside of a controlled testing environment. For Stride, we will follow an iterative testing approach, allowing us to identify issues early, refine functionality continuously, and ensure users are not negatively affected by preventable failures. This iterative approach helps us maintain alignment with our primary goal which is to build an application that genuinely supports and creates value for individuals with accessibility needs.

Below are the two validation methods we will use as part of our validation and verification phase.

2. UAT

We will conduct recurrent User Acceptance Testing throughout development. UAT will serve as a key indicator of success because it allows real users to validate whether the system meets their needs and expectations. Through repeated UAT cycles, we aim to:

- Ensure Stride is intuitive and useful for our target users
- Confirm that we are building features people will actually use
- Prevent misalignment between our implementation and real user needs

Regular feedback from users throughout development will allow us to refine our design, improve usability, and maintain focus on delivering meaningful value.

3. Pilot Testing

Pilot testing will take place once our app reaches the MVP stage and therefore ready for users to fully interact with. This testing format will simulate real-world use and help us evaluate whether the app performs effectively in the environments it is designed for.

We will invite a group of five users to test Stride in the BHEE building over the course of one week. During this pilot phase, we will collect feedback on:

- Whether the app helped users navigate the environment
- Whether additional assistance (such as a white cane) was required
- How the app performed under real environmental conditions (e.g., Wi-Fi reliability, sensor accuracy, building layout constraints)
- How do they feel about the battery efficiency after running Stride, will this affect their decision of using Stride vs other methods

Insights from the pilot will inform our final adjustments and guide decisions about next development steps.

4. Validation Success Metrics

The success metrics to evaluate the effectiveness of UAT and Pilot Testing are as follows.

UAT:

- **Task Completion Rate:** Percentage of users who successfully complete assigned tasks without assistance
- **User Satisfaction Score:** Average satisfaction rating (1–5) collected through UAT surveys
- **Usability Issues Identified:** Number and severity of issues reported during each UAT cycle
- **Accessibility Alignment:** Percentage of UAT testers who report that Stride meets their accessibility needs

Pilot Testing:

- **Navigation Accuracy:** Percentage of guidance steps delivered correctly
- **Reduction in External Assistance:** How often users needed additional tools (like white cane)
- **Performance Stability:** Crash rate, lag, or performance drops experienced during real use
- **User Confidence Score:** Self-reported confidence in navigating BHEE using Stride (1-5)
- **Performance/Efficiency Score:** Self-reported (battery) performance/efficiency score (1-5)
- **Issue Resolution Rate:** Percentage of pilot-reported issues fixed before final implementation

Tools, Environments & Test Data Sets

Testing Tools

- **Jest:** Will be used for Unit Testing the React Native mobile application components to ensure UI rendering and logic correctness.
- **Pytest:** Will be used for Unit and Integration Testing of the Python-based backend logic, including the object detection wrappers and pathfinding algorithms.
- **Postman:** Will be used to verify API Endpoints (Gateway to Lambda), ensuring that image data is correctly received and navigation instructions are returned.
- **AWS CloudWatch:** Will be configured to monitor System Performance, specifically tracking latency metrics to verify the <1 second response time requirement during load testing.

Simulation or Emulation Environments

- **Android Studio Emulator & iOS Simulator:** Used for daily development and initial integration testing to verify the "Cross Platform Deployability".
- **BHEE Building (Physical Environment):** The primary "System Testing" environment. Validation trials will occur here to verify the 80% navigation success rate in the actual target location.

Data Sets

- **Collision Validation Set:** A collected dataset of 100 video clips containing humans and static objects at measured distances (0-15ft). This will be used to automate the testing of FR-1 and FR-2 without requiring physical walkthroughs for every regression test.
- **BHEE Digital Map Data:** The ground 1 0 dataset linking room numbers to specific geospatial

coordinates, stored in the SQL/S3 hybrid database for navigation logic verification.

Version Control & Test Management

- **GitHub:** The central repository for all source code (React Native & Python).
- **GitHub Actions:** Will act as the CI/CD pipeline to automatically run Unit Tests (Jest/Pytest) on every pull request to prevent regressions before merging.