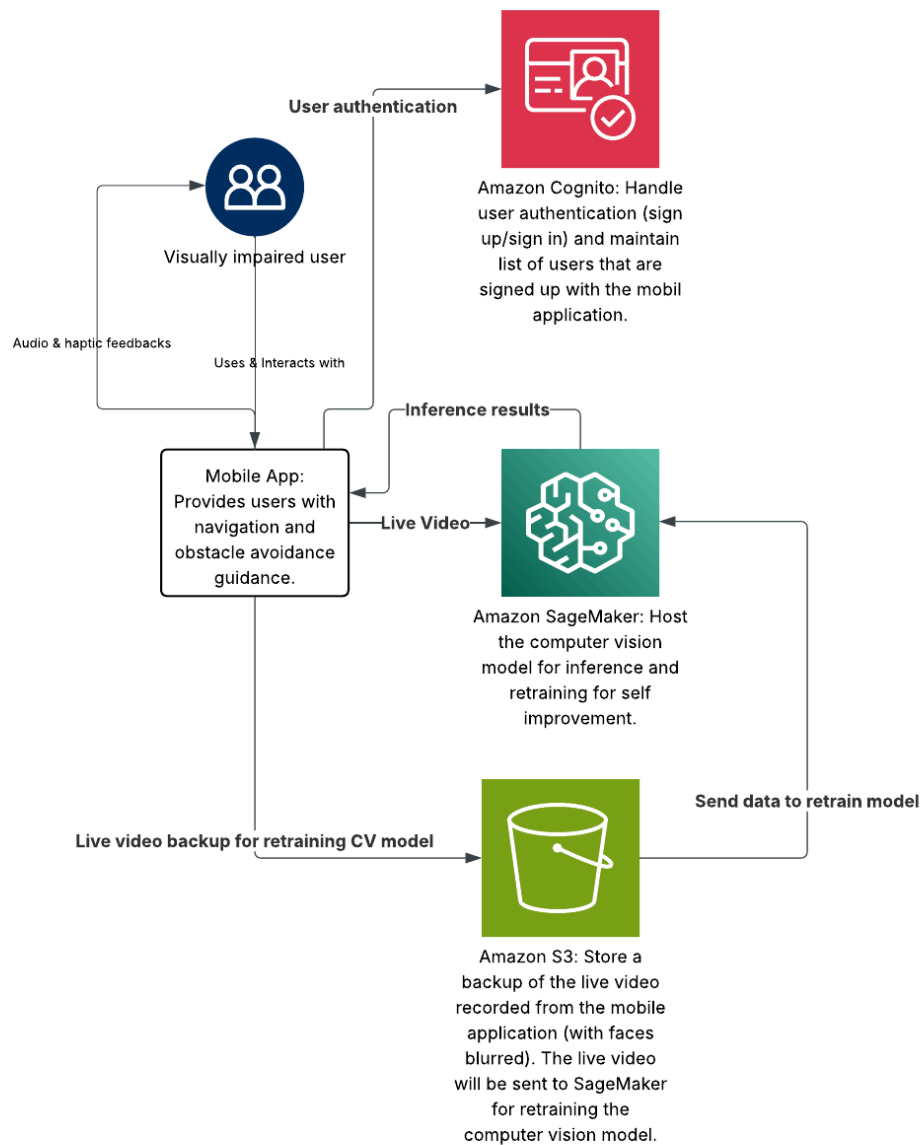


ECE 49595 - Design Document

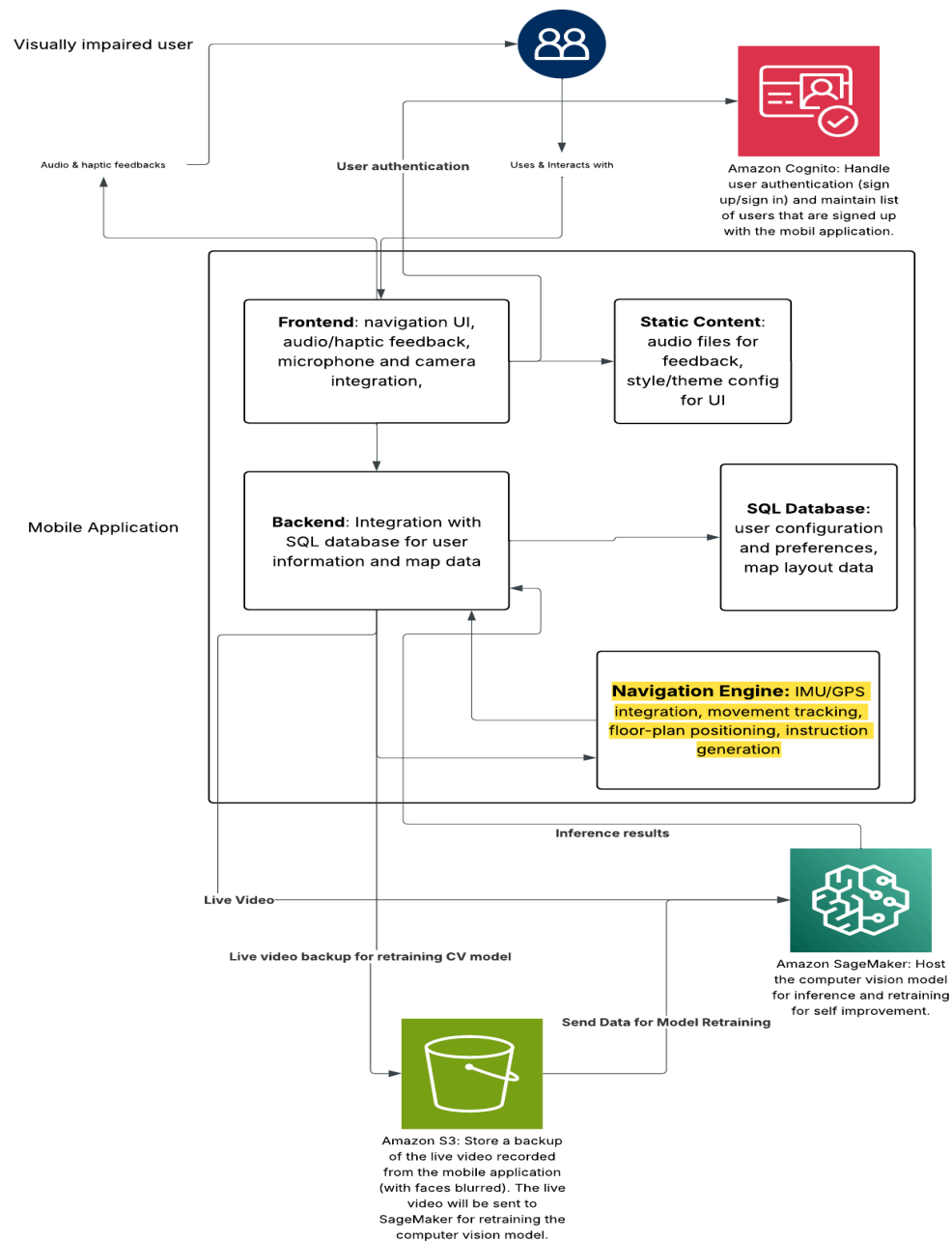
Fall 2025

Design Diagrams:

System Context Diagram



System Container Diagram



Technology Stack

Tech Stack Item	Usage
Amazon S3	Stores anonymized (face-blurred) live video backups and other large artifacts used for computer vision model retraining.
Amazon DynamoDB	Stores structured application data such as user configuration, preferences, and indoor map metadata.
Amazon SageMaker	Hosts the computer vision model for real-time inference and performs retraining to improve navigation accuracy over time.
Amazon API Gateway	Exposes secure REST APIs that allow the mobile application to communicate with backend services.
AWS Lambda	Executes backend logic for authentication checks, data processing, inference routing, and database interactions.
Amazon Cognito	Manages user authentication, including sign-up and sign-in, and securely maintains the user identity pool.
React Native	Provides the cross-platform mobile application framework for building the accessible navigation UI.
Expo	Allows for quick iteration and deployment of of application for better development experience
Python	Used for model training and all machine learning operations
Kotlin	Used for implementing backend logic
TypeScript	Used to implement frontend, ensuring type safety and maintainable code.

Figma:

<https://www.figma.com/design/uehOwEhUkeCgLXKh55bQQQ/StrideUI?node-id=0-1&t=uMWs3h6qDwrtXJny-1>

Design Trade: Computer Vision Detection Model

- **Design Alternatives:**

- **YOLOv11:** The newest YOLO model that has the new C3k2 architecture and C2PSA modules. It includes better feature extraction for blocked objects and higher parameter efficiency (around 22% fewer parameters than v8) while maintaining a really efficient deployment ecosystem.
- **YOLOv8-S:** This is a YOLO model that is reliable and is industry standard that is efficient, accurate, and effective. Even though it is slightly older, it is still a strong baseline that has extensive community support and proven performance on SageMaker endpoints.
- **EfficientDet-D3:** This design focuses on high parameter efficiency and small-object detection accuracy using a weighted bi-directional feature network, though it typically means higher latency costs compared to the YOLO family.

- **Evaluation Table**

Criterion (Weight %)	YOLOv11	YOLOv8-S	EfficientDet-D3
Inference Latency	10 (250)	9 (225)	7 (175)
Detection Accuracy (25%)	10 (250)	9 (225)	8 (200)
Robustness to Indoor Conditions (15%)	9 (135)	8 (120)	8 (120)
Deployment on SageMaker (25%)	10 (250)	10 (250)	7 (175)

Cost Efficiency (10%)	9 (90)	8 (80)	7 (70)
Total Weighted Score (100)	975	900	740

- **Summary:**

We chose **YOLOv11** as the best solution because it received the highest score (975) and offers the best safety and performance for our app. It is an improvement over YOLOv8 because it is more efficient, meaning it can detect obstacles faster than our 1-second limit and helps us stay within our AWS budget. While YOLOv8 is still a good option, YOLOv11 handles "messy" indoor environments better, such as crowded hallways or dim lighting. We did not choose EfficientDet-D3 because it is too slow and expensive to run, which would fail our requirement for real-time feedback.

Design Trade: UI/UX Subsystem

- **Design Alternatives:**

- **Design 1 (Audio-Prioritized Interface):** This design focuses strictly on voice-based interaction using text-to-speech (TTS) and speech recognition with minimal visuals to prioritize accessibility for users with little to no vision.
- **Design 2 (Hybrid Interface):** This option combines audio feedback with haptic feedback and clean visual icons, providing a multimodal experience that supports both visually impaired and partially sighted users.
- **Design 3 (Gesture-Based Minimal Interface):** This alternative utilizes simple physical gesture controls (like swipes and taps) to minimize button usage and screen complexity, though it requires users to memorize specific inputs.

- **Evaluation Table**

Design #	Accessibility	Ease of Navigation	Performance	Deployability	User Satisfaction
1	9	7	8	10	7
2	9	9	6	10	8
3	8	7	8	10	7

Design #	Accessibility	Ease of Navigation	Performance	Deployability	User Satisfaction	Weighted Total
1	2.25	1.4	2	1.5	1.05	8.2
2	2.25	1.8	1.5	1.5	1.20	8.25
3	2	1.4	2	1.5	1.05	7.95

- **Summary**

Design 2 (Hybrid Interface) was selected as the superior solution because it achieved the highest weighted score (8.25) by balancing full accessibility with redundancy, ensuring users receive feedback even in noisy environments or low-battery situations. While Design 1 offered excellent hands-free operation, it was rejected because reliance solely on audio poses safety risks in loud areas where cues might be missed.

Additionally, Design 3 was discarded because the risk of gesture misrecognition could mislead users and jeopardize safety, making Design 2's multimodal approach the most reliable and effective choice

Design Trade:

- **Design Alternatives**

- **Design A1 – AWS Cognito with IAM Integration.** This design uses AWS Cognito as a fully managed identity provider to handle user signup, login, and session management. Cognito issues secure tokens that integrate natively with AWS IAM, allowing authenticated users to access backend services such as API Gateway, Lambda, and S3 with minimal custom code.
- **Design A2 – Custom Authentication Service with DynamoDB.** This approach involves building a custom authentication backend using AWS Lambda and a REST API. User credentials are stored in DynamoDB using salted and hashed passwords, and custom JWT tokens are issued and validated for each session.
- **Design A3 – Firebase Authentication.** Firebase Authentication provides a managed identity solution with strong mobile SDK support for React Native. It supports email/password and social logins, issuing tokens that are verified by the backend before allowing access to AWS-hosted services.

- **Evaluation Table**

Criterion	Weight	A1: AWS Cognito + IAM	A2: Custom Auth + DynamoDB	A3: Firebase Auth
Security Strength	30%	10 - 300 (managed tokens, built-in MFA, AWS IAM integration, strong key rotation)	6 - 180 (we control it, but also we can break it)	9 - 270 (Google-managed identity, production-grade hardening)
Dev / Maintenance Effort	20%	8 - 160 (config heavy but code-light)	4 - 80 (we own/maintain everything forever)	9 - 180 (SDKs handle most features out of the box)
Latency / Runtime Experience	20%	9 - 180 (token reuse, low round-trip inside AWS region, no custom hops)	7 - 140 (depends on our own infra tuning, but can be decent)	8 - 160 (fast on-device token issue, but extra hop to AWS backend for validation)
Scalability / Reliability	15%	9 - 135 (Cognito scales by MAU; high availability is built-in)	6 - 90 (we'd have to scale the auth API and manage DDoS / load)	8 - 120 (Firebase Auth is designed to scale to tens of thousands+ MAU without extra work)
Cost	15%	9 - 135 (free tier up to	7 - 105 (infra cost is low	9 - 135 (Auth mostly

Efficiency		10K–50K MAU; charges per MAU after that, still low fractions of a cent per MAU)	at tiny scale, but labor/maintenance cost is high even if AWS bill is cheap)	free for email/password and social login up to ~50K MAU; SMS costs extra)
Total Weighted Score	100%	910	595	865

- **Summary**

AWS Cognito with IAM integration was selected as the final authentication and login solution because it achieved the highest weighted score (910) and best satisfies the system's security, reliability, and usability requirements. Cognito provides managed user pools, secure token issuance, and built-in protections such as encryption, MFA, and automatic key rotation, eliminating the risk and maintenance burden of implementing custom security logic. Its native integration with AWS services allows authentication to remain within the same cloud boundary, minimizing latency and ensuring login and signup meet the $\leq 1\text{--}2$ second requirements. While Firebase offers an excellent developer experience, it introduces cross-cloud complexity and weaker integration with AWS IAM. The custom authentication approach was rejected due to its high development effort and long-term security risks, making Cognito the most robust and maintainable choice for a safety-critical navigation application.

Design Trade: Backend Communication Architecture

- **Design Alternatives**

- **Design 1 – Serverless API Gateway with Cognito Authentication.** This design uses Amazon Cognito for user authentication and an API Gateway endpoint to securely receive video frames from the mobile app. After token validation, AWS Lambda retrieves map data from S3, invokes the SageMaker endpoint for inference, and returns combined results to the client in a request–response pattern.
- **Design 2 – Real-Time Streaming Pipeline (Kinesis Video Streams).** In this approach, the client authenticates with Cognito and streams video frames continuously to AWS Kinesis Video Streams. Backend Lambda functions process the stream, invoke SageMaker for inference, and send results back to the client through a separate communication channel such as WebSockets.
- **Design 3 – Direct SDK Invocation with Cognito.** This design gives the client temporary AWS credentials through Cognito and allows it to directly fetch map data from S3 and invoke the SageMaker endpoint using the AWS SDK. All coordination and logic for merging inference and navigation data is handled on the client side.

- **Evaluation Table**

Criterion (Weight %)	Serverless API Gateway	Real-Time Streaming Pipeline	Direct SDK Invocation
Latency (40%)	7	4	10
Security (25%)	9	9	4
Cost Efficiency (20%)	9	5	8
Scalability (10%)	9	10	7
Implementation Complexity (5%)	9	3	5
Total Weighted Score	8.2	6.0	7.55

- **Summary**

The **Serverless API Gateway architecture (Design 1)** was selected as the optimal backend communication solution because it achieved the highest weighted score (8.2) while maintaining a strong balance between latency, security, cost efficiency, and implementation simplicity. This design satisfies the strict timing requirements of FR-1 and FR-2 by keeping inference latency low without exposing sensitive AWS resources directly to the client. By validating users through Cognito at the API Gateway layer, the system protects user data such as GPS location and video frames while minimizing the attack surface. Although the real-time streaming pipeline offers scalability, its higher cost and complexity make it unsuitable for a small-scale, budget-constrained project. Direct SDK invocation was rejected due to security risks associated with granting the client access to core AWS services, making Design 1 the most secure, scalable, and practical choice for real-time navigation and collision detection .