Project Title: CityAssist — Citizen-Centric Urban Assistant

Project Summary

CityAssist is a citizen-focused Smart City platform that helps daily users (residents, commuters, small businesses) interact with city services, receive personalized alerts, and get actionable recommendations. Unlike the previous operations/command-center project focused on administrators, CityAssist centers on everyday utility: travel suggestions, service outage notifications, localized health & air-quality tips, personalized energy-saving suggestions, and simple two-way requests (report potholes, waste pickup requests, request streetlight repairs).

The system must be both **consumer-friendly** and **enterprise-grade** so back-office teams can use the same data and models for operations.

Core User Value (daily users)

- Real-time travel advisory and alternate route suggestions based on traffic and events
- Personalized air-quality alerts and health recommendations (mask/advisory) by profile (age, health)
- Utility outage notifications (water/power) with ETA and service updates
- Quick reporting for civic problems (image + location) with tracking and push updates
- Localized offers and services (community services, vaccination drives, local vendor alerts)

High-Level Architecture

- Mobile-first Web UI (PWA) + lightweight mobile app wrapper
- Java Spring Boot backend microservices for user management, reports, and integrations
- Python ML services for personalization, routing, and anomaly detection
- Real-time messaging via WebSocket or push (Firebase/FCM) for user notifications
- Data store: PostgreSQL (primary), Redis (cache), object storage (S3) for images
- DevOps: Docker, Kubernetes, CI/CD, monitoring
- Analytics: Power BI for city managers and aggregated insights

---

Domain-wise Detailed Tasks

1) Frontend (Citizen UX) — Deliverable: PWA + responsive portal

Objective: Build a mobile-first Progressive Web App that daily users will open for quick tasks.

Key features to implement:

- Landing / Onboarding: quick profile creation (age, medical flags, commute patterns) and consent for notifications
- Home feed: personalized alerts (AQI, traffic, utility) with action buttons (View route, Request refund, Report issue)
- Map & Commuter assistant: live traffic heatmap, alternate route suggestion, estimated travel time, public transit overlay
- Report an Issue: photo upload, geotagging, category selection, submit form — show ticket ID + timeline
- Service Status: subscribe to utilities (water/power/internet) for zone-level outage alerts & ETA
- Notifications center: read/unread, snooze, settings
- Local Services: directory of nearby hospitals, pharmacies, shelters, community centers with click-to-call and directions
- Accessibility, offline support (cache important pages), and small bundle size (PWA guidelines)

Integration points:

- Backend auth + profile APIs
- AI services for route suggestions and personalized tips
- Push notifications (FCM/Browser Push)

Deliverables:

- Full PWA with manifest & service worker
- Mock API integration using /data mocks and clear placeholders for real endpoints
- User flows (onboarding, report, receive update) documented

---

2) Java Backend (Citizen APIs) — Deliverable: User-centric microservices

Objective: Provide secure, scalable APIs consumed by PWA and city systems.

Microservices suggested:

- **UserService:** profile, preferences, authentication (JWT), subscription management
- **ReportService:** accept reports (image, location, category), generate ticket, manage lifecycle, attach timeline events
- **NotificationService:** queue/send push notifications and web notifications; manage subscriptions
- **RoutingService API gateway:** orchestrates calls to traffic & map services and personalization engine
- **IntegrationService:** connectors to utility providers, city 3rd-party systems

Key responsibilities:

- Clear, versioned REST APIs (OpenAPI/Swagger)
- Data validation & idempotency for report submissions
- Secure image uploads (S3 pre-signed URLs)
- Audit trails for user actions and ticket updates
- Role-based endpoints for power-users (operators) to update ticket statuses

Deliverables:

- API spec docs + Postman collection
- Database schema and migration scripts
- Docker images for each microservice

---

3) Python & Data Science (Personalization & Predictions) — Deliverable: ML APIs + models

Objective: Build models and services that directly improve daily user experience.

Focus areas & tasks:

- **Personalized Alerting:** build model that learns user sensitivity & preferences (e.g., elderly flagged for stronger AQI alerts)
- **Real-time Route Recommender:** lightweight model (or rules + ML) that recommends alternate routes considering congestion, events, and user preferences (avoid highways, prefer bus lanes)
- **Outage ETA Estimator:** predict time-to-restore using historical outage patterns, weather, and grid sensor data
- **Image triage for reports:** small CNN to auto-classify report images (pothole vs. garbage vs. tree fall) to prioritize routing
- **Explainability:** return short human-friendly reasons with each prediction ("High congestion due to accident at X")

Integration: expose FastAPI endpoints for predictions and model metadata. Cache frequent responses in Redis.

Deliverables:

- Model artifacts, notebooks, and inference API
- Sample explainability outputs and testing harness

---

4) DevOps— Deliverable: Production-grade CI/CD and infra

Objective: Make CityAssist deployable, observable, and resilient.

Major workstreams:

- **Infrastructure as Code:** Terraform modules for VPC, EKS/ECS cluster, S3 buckets, RDS, IAM roles
- **CI/CD Pipelines:** GitHub Actions or Jenkins pipelines per repo: build/test/dockerize/push/helm deploy
- **Kubernetes:** Helm charts for each microservice, k8s manifests, HPA, liveness/readiness
- **Secrets & Config:** use Vault or Kubernetes Secrets, ensure rotation best-practices
- **Observability:** Prometheus metrics, Grafana dashboards, ELK/OpenSearch for logs, distributed tracing with Jaeger
- **SRE:** Implement alerting (PagerDuty/SMS), automated rollback strategy, runbooks
- **Cost & Security:** set up cost monitoring, scanning images (Trivy), and IaC security checks

Deliverables:

- Terraform repo and state guidance
- CI/CD pipelines and example runs
- Monitoring dashboards and runbook docs

---

5) Power BI & Insights — Deliverable: Citizen Insights Dashboards

Objective: Provide digestible executive dashboards and public-facing summary pages.

Key dashboards:

- **Citizen Health Dashboard:** aggregated AQI alerts, demographics impacted, top risk zones
- **Service KPI Dashboard:** tickets created vs resolved, average restore ETA, SLA compliance
- **Mobility Dashboard:** peak congestion corridors, alternate route success, commuter delay estimates

Integration: use sanitized aggregated tables from DB or periodic API snapshots. Provide CSV export and data model documentation.

Deliverables: .pbix report files, measure definitions (DAX), scheduled refresh setup

---

6) Testing & QA — Deliverable: Test suites & reliability reports

Objective: Validate UX, API reliability, performance, and security for citizen workflows.

Tasks:

- **Functional tests:** onboarding, report submission, ticket tracking, notification receipt
- **End-to-end tests:** simulate a user reporting an issue and receiving updates
- **Load test:** ramp-up concurrent report submissions and push notifications
- **Security tests:** ensure image upload sanitization, auth checks, token expiry
- **Accessibility:** WCAG checks, keyboard navigation, screen reader tests

Deliverables: automated test scripts, performance reports, accessibility audit

---

Integration & Handoff

- Catalog of API endpoints and contracts for each microservice
- Shared test data sets and mock servers for frontend/backoffice testing
- Runbooks for DevOps and incident management

Deliverables (company-ready handout)

- A single comprehensive document (this doc expanded) for teams
- API specs, Postman collection
- Terraform + Helm + CI/CD pipelines
- PWA codebase scaffold (frontend) with mock data
- Model artifacts + serving endpoints
- Power BI reports
- Test cases and results