**SSB 1.0 — System Architecture Document**

**1. Executive Summary**

Tài liệu này mô tả **Thiết kế kiến trúc hệ thống** cho dự án **Smart School Bus Tracking System (SSB 1.0)**. Mục tiêu của tài liệu: cung cấp bản thiết kế cấp cao (high-level architecture), mô tả các module chính, luồng dữ liệu, lựa chọn công nghệ, và các quyết định kiến trúc cần thiết để triển khai hệ thống theo yêu cầu đã phân tích.

**2. Phạm vi (Scope)**

* Module chính: Bus Schedule & Tracking (trọng tâm), User Management, Notification, Student Management, Reporting.
* Hỗ trợ người dùng: Admin/Bus Manager, Driver (tài xế), Parent (phụ huynh), (tùy chọn) Student.
* Hệ thống cần theo dõi GPS real-time, gửi push notifications, lưu lịch sử hành trình, hỗ trợ check-in/check-out bằng QR/RFID.

**3. Stakeholders**

* **Bus Manager / School Admin**: lập lịch, quản lý tuyến, xem tổng quan.
* **Driver (Tài xế)**: gửi vị trí, cập nhật trạng thái, gửi alert.
* **Parent (Phụ huynh)**: xem vị trí xe, nhận thông báo, theo dõi lịch sử.
* **System Operator / DevOps**: deploy, giám sát, backup.
* **Vendor/Integrator (ABC)**: implement, tích hợp API bản đồ, tích hợp hệ thống trường học.

**4. Yêu cầu tóm tắt**

**4.1 Functional (chính)**

* Đăng ký/Đăng nhập/Phân quyền
* Theo dõi vị trí xe theo thời gian thực
* Gửi thông báo (khi đến gần, khi đón/xuống, khi chậm trễ)
* Quản lý tuyến, trạm, lịch trình, phân công tài xế
* Check-in/Check-out bằng QR/RFID
* Báo cáo lịch sử hành trình, trạng thái xe
* Cảnh báo khẩn cấp từ tài xế

**4.2 Non-functional**

* Hiệu suất: vị trí cập nhật < 2s phản hồi cho user
* Khả năng mở rộng: hỗ trợ 100+ tuyến, 1000+ concurrent users
* Bảo mật: TLS, JWT, RBAC
* Tính khả dụng: 24/7, phục hồi nhanh
* Khả năng tích hợp: Google Maps/OpenStreetMap, hệ thống quản lý trường

**5. Kiến trúc tổng quan (High-level Architecture)**

Hệ thống sử dụng kiến trúc **3-tier / microservices-lite**:

1. **Clients (Presentation)**
   * Mobile App (Flutter/React Native) — Driver, Parent
   * Web App (React) — Admin/Bus Manager, Reporting
   * Admin Console
2. **Backend Services (Application Layer)**
   * Auth & User Service
   * Bus Schedule Service
   * GPS Tracking Service (WebSocket / MQTT broker)
   * Notification Service (Push + SMS/Email)
   * Student Service (check-in/out)
   * Reporting Service
3. **Data Layer**
   * Relational DB (PostgreSQL)
   * Cache / Realtime store (Redis)
   * Time-series / blob storage for GPS traces (optional TSDB like InfluxDB or storing as files)

**Deployment**: containerized (Docker) orchestrated by Kubernetes; Cloud provider: AWS/Azure/GCP.

**6. Component Diagram (textual)**

[Parent App] ↔ [API Gateway] ↔ [Auth Service]

↳ [Bus Schedule Service]

↳ [GPS Tracking Service] ↔ [Redis / TSDB]

↳ [Notification Service] ↳ [FCM / APNs / SMS Gateway]

[Driver App] ↔ [WebSocket / MQTT Broker] ↔ [GPS Tracking Service]

[Admin Web] ↔ [API Gateway] ↔ [Reporting Service] ↔ [DB]

**Giải thích ngắn**: API Gateway/Load Balancer nhận request từ client, route đến các services. Tracking Service nhận WebSocket/TCP từ thiết bị driver, cập nhật Redis, push event tới Notification Service hoặc broadcast tới clients qua WebSocket/WS connections.

**7. Luồng chính (Flows)**

**7.1 Login / Auth**

1. Client gửi username/password hoặc social auth -> API Gateway
2. Auth Service xác thực -> trả JWT + refresh token
3. Client dùng JWT cho các call sau

**7.2 GPS Tracking (Real-time)**

1. Driver app gửi vị trí định kỳ (x s/giây) qua WebSocket hoặc MQTT
2. Tracking Service nhận -> validate -> ghi cache Redis và append vào hành trình ngày (PostgreSQL hoặc TSDB)
3. Tracking Service broadcast vị trí mới tới các client (parents subscribed) hoặc push notification nếu xe sắp đến

**7.3 Check-in / Check-out**

1. Học sinh/Phụ huynh quét QR/RFID ở trạm -> Driver/Device gọi API Student Service
2. Student Service update trạng thái, notify parent và log thời gian vào history

**8. Data Model (tổng quan bảng chính)**

* Users (id, role, name, contact, etc.)
* Buses (id, plate\_no, capacity, vehicle\_info)
* Drivers (id, user\_id, license\_info)
* Routes (id, name, list\_of\_stops)
* Stops (id, route\_id, lat, lng, sequence)
* Schedules (id, route\_id, bus\_id, driver\_id, departure\_time)
* Students (id, name, parent\_id, assigned\_route\_id, assigned\_stop)
* GPS\_Traces (id, bus\_id, timestamp, lat, lng, speed, heading)
* Checkins (id, student\_id, timestamp, stop\_id, action)

**9. Module Breakdown (chi tiết chức năng & APIs mẫu)**

**9.1 Auth & User Management**

**Chức năng**: đăng nhập, đăng ký, đổi mật khẩu, quên mật khẩu, quản lý profile, phân quyền RBAC.  
**API mẫu**:

* POST /auth/login {username,password} → {accessToken, refreshToken}
* POST /auth/refresh {refreshToken} → {accessToken}
* GET /users/me → User

**9.2 Bus Schedule Service**

**Chức năng**: quản lý tuyến/trạm, lịch trình, phân công tài xế/xe.  
**API mẫu**:

* GET /routes
* POST /schedules
* PUT /schedules/{id}

**9.3 GPS Tracking Service**

**Chức năng**: nhận vị trí, validate, lưu cache, tính ETA, broadcast  
**Conn**: WebSocket endpoint: ws://.../tracking?token=JWT  
**API mẫu**:

* POST /tracking/report {bus\_id, lat, lng, speed, ts}
* GET /tracking/bus/{bus\_id}/position → latest position

**9.4 Notification Service**

**Chức năng**: rule-based notifications (on-arrival, delay, checkin), push via FCM/APNs + SMS fallback  
**API mẫu**:

* POST /notify/send {target, message, type}

**9.5 Student Service**

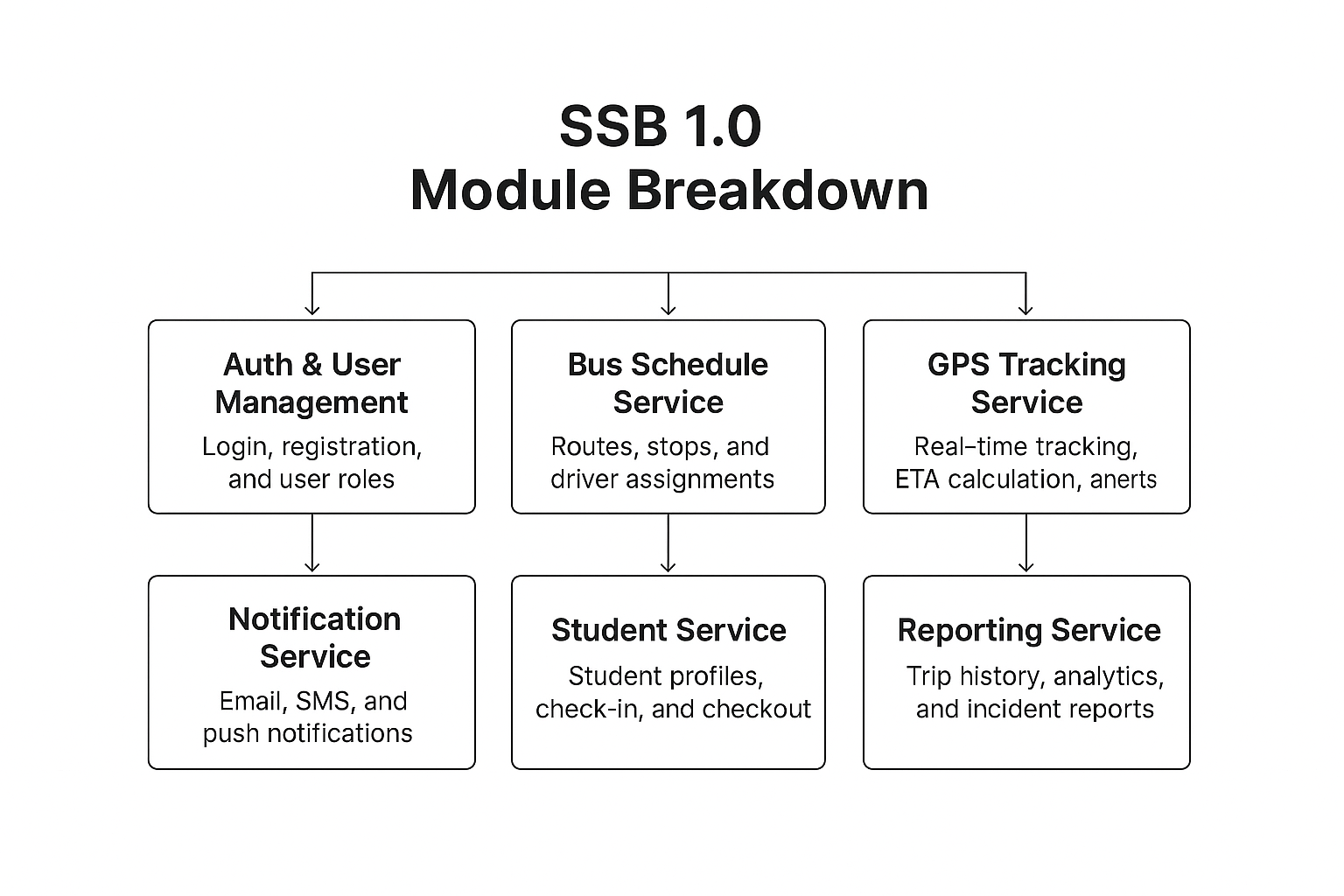
**Chức năng**: quản lý học sinh, check-in/out, mapping student ↔ route  
**API mẫu**:

* POST /students/{id}/checkin {stop\_id, ts}

**9.6 Reporting Service**

**Chức năng**: xuất báo cáo hành trình, tổng hợp trễ, thống kê sử dụng  
**API mẫu**:

* GET /reports/daily?date=YYYY-MM-DD



**10. Non-functional Considerations & Design Decisions**

* **Realtime store**: Redis for latest positions + pub/sub; long-term traces to PostgreSQL or TSDB.
* **Scale**: Tracking Service horizontally scalable; use partitioning for GPS traces by bus\_id/date.
* **Latency**: use WebSocket and local caching; server compute ETA asynchronously.
* **Resilience**: circuit breakers, retries, bulkhead for Notification Service.
* **Security**: HTTPS everywhere, JWT short-lived, refresh tokens stored securely, role-based access.
* **Privacy**: GPS data retention policy, anonymization if needed, parental consent logging.

**11. Deployment Topology**

* **K8s cluster**: services as Deployments + HPA
* **Ingress**: Load Balancer + API Gateway (e.g., Nginx/Envoy)
* **Databases**: managed PostgreSQL, Redis cluster
* **Messaging**: MQTT broker (EMQX/Mosquitto) or Redis pub/sub for internal events
* **CI/CD**: GitHub Actions / GitLab CI deploy to K8s

**12. Security & Privacy**

* TLS 1.2+ for all endpoints
* Access control: RBAC for admin actions
* Audit logs for critical actions (route changes, checkin overrides)
* Data retention: GPS raw traces stored *X* months (configurable), aggregated data indefinite
* Incident response plan for data leaks

**13. Monitoring & Observability**

* Metrics: Prometheus + Grafana
* Tracing: OpenTelemetry (Jaeger)
* Logs: centralized (ELK/EFK stack)
* Alerts: CPU/memory, queue length, error rate, delivery failures for notifications

**14. Operational Playbooks (tóm tắt)**

* **On GPS outage**: switch to store-and-forward on driver device; on reconnection, device replays buffered points
* **On high notification failure**: fallback to SMS for critical alerts
* **On DB outage**: read-only mode for reporting; emergency failover to replica

**15. Roadmap & Next Steps**

1. Proof-of-Concept: basic auth + tracking + map display + push notification (MVP)
2. Add check-in/check-out, QR/RFID integration
3. Reporting & analytics
4. Hardening, scaling, multi-school support

**16. Appendix**

**A. Sequence: Driver position → Parent receives update**

1. Driver app connects to ws://tracking
2. Driver app sends location
3. Tracking Service validates, saves to Redis, calculates ETA
4. Notification Service checks rules (if within threshold) → push message
5. Parent app receives push OR active WS message and updates map