Dean Gibson

ST10326084

PROG7314 ICE TASK 1

Based on my POE PART 1 document.

Contents

ICE TASK 1 — RESTful API Design (StrideHabits / SummitAPI)

1) API Functionality

Purpose (context of the app)

SummitAPI powers the StrideHabits Android habit tracker. It stores Users, Habits, and per-day Check-ins, returns streak/history data to the app, and supports an offline-first sync model (the app queues local changes and syncs when online).

Core resources & endpoints (v1)

Auth & User

- POST /api/users/register → Create account (email/password).
  Req: { "name": "Sam", "email": "sam@ex.com", "password": "P@ssw0rd!" }
  Res 201: { "id": "guid", "token": "jwt", "name": "Sam" }
- POST /api/users/login → Exchange credentials for JWT.
  Req: { "email": "sam@ex.com", "password": "P@ssw0rd!" }
  Res 200: { "token": "jwt", "user": { "id": "guid", "name": "Sam" } }
- GET /api/users/me (Auth) → Current user profile.

(IAmTimCorey, 2022)

Habits

- GET /api/habits?since=2025-08-01T00:00:00Z&limit=100 → List habits (supports delta/pagination).
- GET /api/habits/{id} → Fetch one.
- POST /api/habits → Create.
  Req: { "name":"Read 20min", "frequencyPerWeek":4, "tag":"LEARNING", "imageUrl":null }
  Res 201: Habit DTO with id, timestamps.
- PUT /api/habits/{id} → Full update (idempotent).
- DELETE /api/habits/{id} → Remove habit (soft-delete recommended).

(IAmTimCorey, 2022)

Check-ins (date-scoped events)

- GET /api/checkins?habitId={id}&from=2025-08-01&to=2025-08-31 → History for charts/streaks.
- POST /api/checkins → Record completion (unique per habitId + date).
  Req: { "habitId":"guid", "date":"2025-08-28" }
  Res 201: { "id":"guid", "habitId":"guid", "date":"2025-08-28" }

- DELETE /api/checkins/{id} (or DELETE /api/checkins?habitId=...&date=...) → Undo.

(IAmTimCorey, 2022)

Sync (offline-first)

- POST /api/sync/outbox → Upsert a batch of local changes (idempotent by (habitId, date)).
  Req: { "checkins":[{ "habitId":"...", "date":"2025-08-28" }, ...] }
  Res 200: { "upserted": 5 }
- GET /api/sync/changes?since=2025-08-25T10:00:00Z → Return server deltas for Habits/Check-ins.

(IAmTimCorey, 2022)

Utilities

- GET /healthz → Health check.
- GET /api/export?format=csv → CSV/JSON export (user-scoped).

(IAmTimCorey, 2022)

Data formats (DTOs)

```
// Habit
{
  "id": "guid",
  "userId": "guid",
  "name": "Read 20min",
  "frequencyPerWeek": 4,
  "tag": "LEARNING",
  "imageUrl": null,
  "createdAt": "2025-08-27T12:30:00Z",
  "updatedAt": "2025-08-27T12:30:00Z"
}
```

```
// CheckIn
{
  "id": "guid",
  "habitId": "guid",
  "date": "2025-08-28",
  "createdAt": "2025-08-28T06:12:00Z"
}
```

2) Implementation Plan

Stack & tools

- **Language/Framework:** C# **ASP.NET Core (Web API)**
- **Data: SQLite** via **Entity Framework Core** (code-first). Optional **LiteDB** for lightweight logs/cache.
- **Auth: JWT** (Phase 2 adds Google SSO).
- **Docs/Testing:** Swagger (OpenAPI), Postman, xUnit.
- **Android client:** Kotlin (MVVM, **RoomDB**), Retrofit, WorkManager, AlarmManager.
  These match my POE design & research.

Approach

- **REST principles:** resource-oriented URIs, proper verbs (GET/POST/PUT/DELETE), status codes (201 for create), and idempotency(property of operations) for PUT and sync upserts.
- **Data model (MVP):** User, Habit, HabitCompletion, Configuration/Settings.
- **Sync strategy:** Android keeps an **outbox** (temp table) in Room; WorkManager pushes to POST /api/sync/outbox; server upserts by (habitId, date) (last-write-wins or additive counts if times-per-day is enabled). Client pulls deltas with GET /api/sync/changes?since=….
- **Security:** JWT bearer auth; HTTPS enforced; CORS only for the Android app origin/package signature.
- **Validation & mapping:** FluentValidation (or DataAnnotations); DTOs mapped from entities.
- **Observability:** Minimal request logging; structured logs to disk (and optional LiteDB).

Why these choices? (brief justification)

- **SQLite on Render** keeps costs low and fits MVP scale; EF Core speeds delivery.
- **JWT** is portable across mobile clients and suits stateless REST.
- **Offline-first** matches my Loop/Habitify market findings and POE scope; sync is simple & date-scoped.

o

3) Deployment & Hosting

Host: Render (free tier) with **persistent disk** for SQLite DB file; HTTPS; CI/CD from GitHub. **Supabase Storage** can be used later for images/BLOBs if needed (Siegfriedson, 2023).

Connection & configuration

- **Connection string:** Data Source=/data/app.db;Cache=Shared (mounted disk path on Render).
- **Env vars:** JWT__Issuer, JWT__Key, ASPNETCORE_URLS, ConnectionStrings__Default.
- **CORS:** Allow Android app package SHA/signature (or a controlled origin for local testing).
- **Base URL (example):** https://<service-name>.onrender.com (Swagger at /swagger).

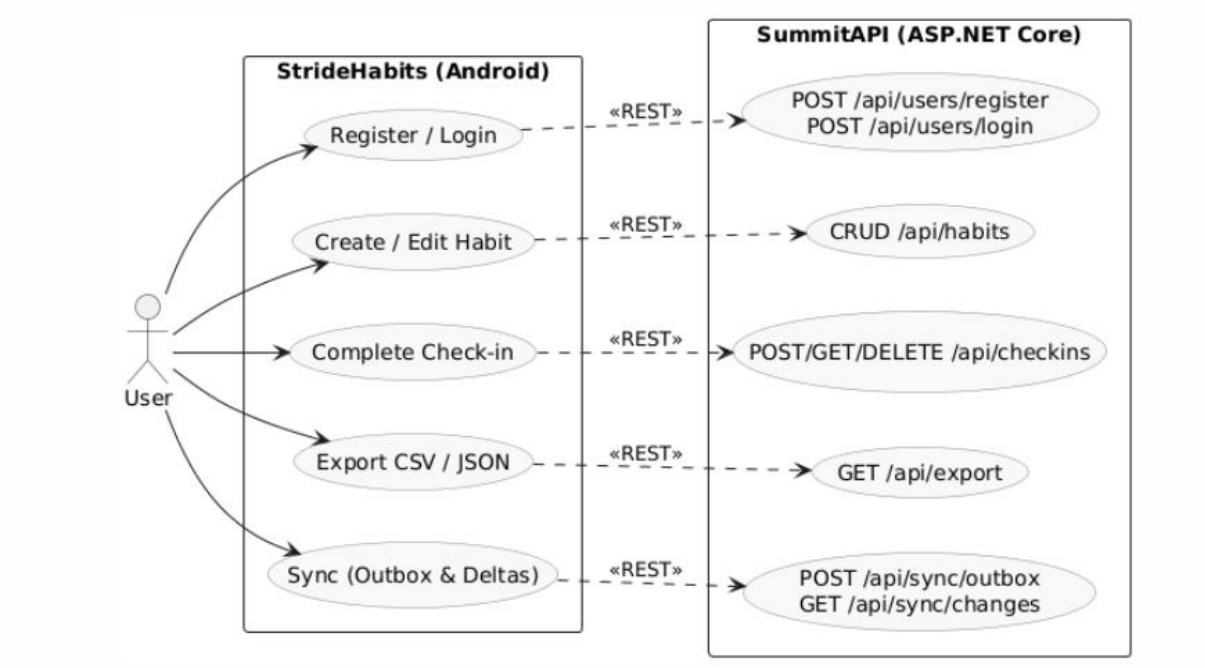
Deployment steps

1. Push to GitHub; enable Render "Web Service" build.
2. On first deploy, EF Core dotnet ef database update (migrations) or ensure automatic migration on startup.
3. Verify /healthz then smoke test(basic prelim tests) endpoints via Swagger/Postman.
   Matches my POE "Hosting and project creation" plan.

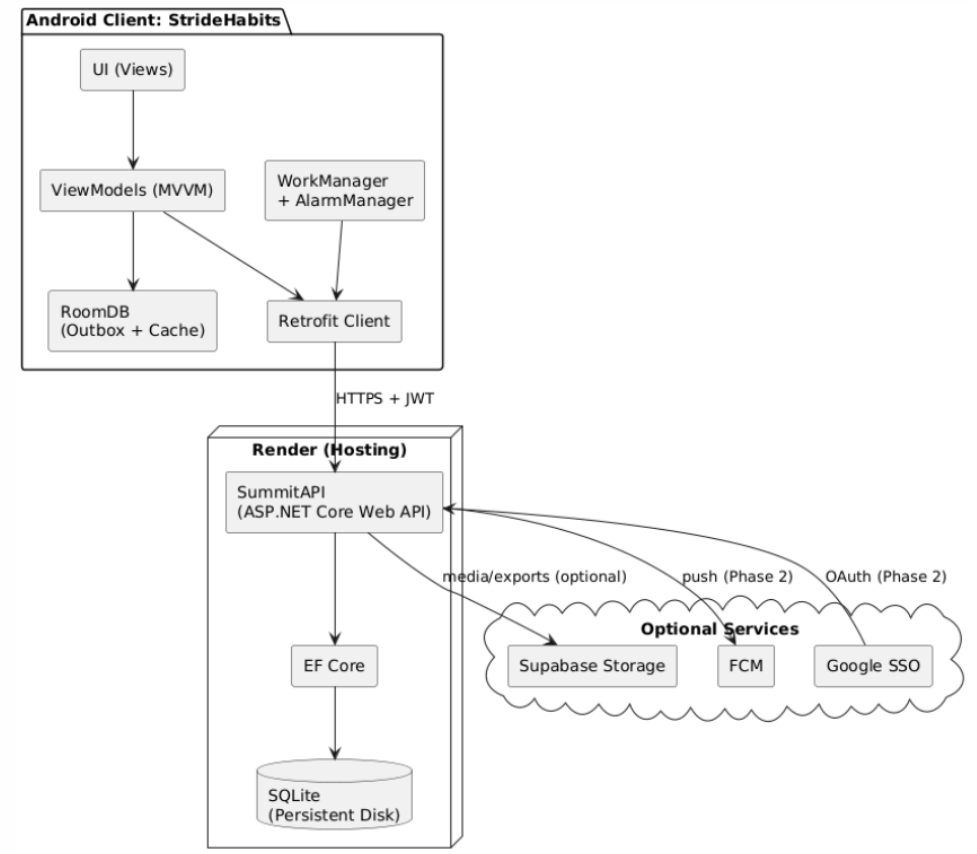
(Siegfriedson, 2023)

4) UML Diagrams

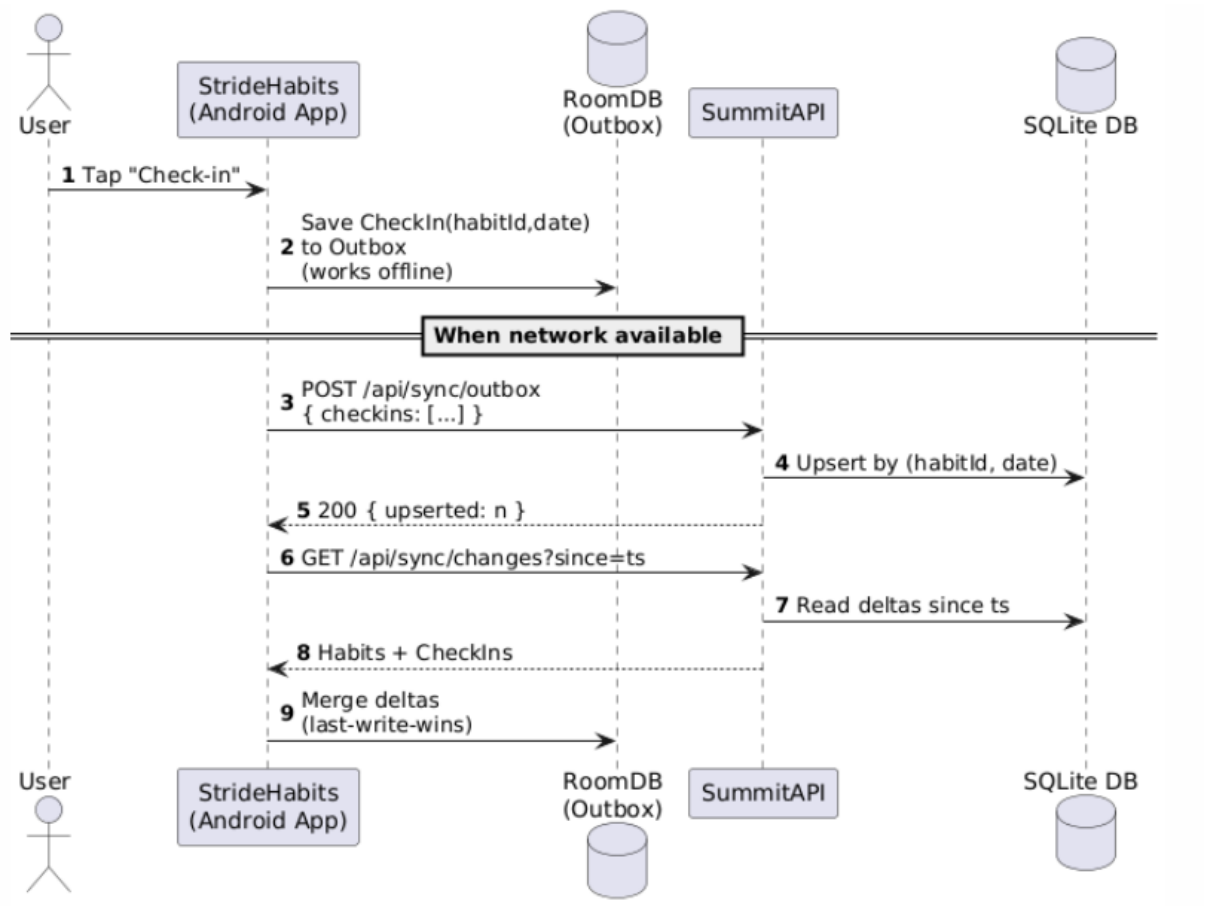
4.1 Use Case Diagram (user + API interactions)



2) Component / Architecture Diagram



3) Sequence Diagram (Offline Check-in Sync)



References

IAmTimCorey (2022). ASP.NET Core Web API Features You Need to Know In 10 Minutes or Less. [online] Youtube. Available at: https://www.youtube.com/watch?v=s1bk-68aB1U (Accessed: 22 August 2025).

Siegfriedson (2023). Thoughts on Render.com - Sort of Like a Tech Diary - Medium. [online] Medium. Available at: https://medium.com/sort-of-like-a-tech-diary/thoughts-on-render-com-214176365ecc (Accessed: 18 August 2025).

Simplilearn (2023). UML Diagram For Software Engineering | Unified Modelling Language Diagram | Simplilearn. [online] YouTube. Available at: https://www.youtube.com/watch?v=WqT_qGgoZbw (Accessed: 20 August 2025).

Tatis, M. (2021). What Is an API for Mobile Apps? [online] Koombea. Available at: https://www.koombea.com/blog/api-for-mobile-apps (Accessed: 20 August 2025)

(Siegfriedson, 2023)