# **App Development Roadmap - Checkpoints**

This document contains the structured roadmap for building the MVP of the product-sharing app.  
 The project is divided into phases and smaller steps (checkpoints). Each step can be used as a reference point so progress can be tracked easily.

## **Phase 0: Setup & Foundations**

* **Step 0.1**: Define folder structure (backend, frontend, db, infra)
* **Step 0.2**: Initialize Git repo & environment (Python/Node, Docker, etc.)

## **Phase 1: Backend MVP (Local)**

* **Step 1.1**: Setup FastAPI project
* **Step 1.2**: Create basic health-check API (/ping)
* **Step 1.3**: Setup Postgres in Docker
* **Step 1.4**: Create User model & User APIs (signup/login with JWT)
* **Step 1.5**: Add basic CRUD for Products (title, description, price, image)
* **Step 1.6**: Add Trips API (traveler posts trip)
* **Step 1.7**: Add Requests API (user requests item, links to trip + product)

## **Phase 2: Conversations (Chat System)**

* **Step 2.1:** Design and create Message model (sender, receiver, request, content, timestamp).
* **Step 2.2:** Implement chat APIs:
  + POST /messages → send message.
  + GET /messages/{request\_id} → fetch conversation thread.
* **Step 2.3:** Add access control so only the requester and traveler of a given request can send/read messages.
* **Step 2.4:** Start with text-only messages, ordered by timestamp.
* **Step 2.5 (optional upgrade):** Later replace polling with **WebSockets** for real-time chat.

## **Phase 3: Payments (Simulated)**

* **Step 3.1**: Setup Stripe test mode
* **Step 3.2**: Create Escrow model in DB
* **Step 3.3**: Link Escrow to requests (hold, release, refund simulation)

## **Phase 4: Messaging**

* **Step 4.1**: Add chat model (messages between requester & traveler)
* **Step 4.2**: Implement WebSockets for real-time chat

## **Phase 5: Frontend**

* **Step 5.1**: Setup Next.js project
* **Step 5.2**: User login/signup screens
* **Step 5.3**: Browse products/trips UI
* **Step 5.4**: Request flow (product request → trip selection → escrow → delivery confirmed)
* **Step 5.5**: Chat UI

## **Phase 6: Orchestration & Testing**

* **Step 6.1**: Docker Compose for API + DB + MinIO
* **Step 6.2**: Integration testing (Postman/Newman or pytest)
* **Step 6.3**: Demo-ready build

## **Phase 7: Cloud Migration (Later)**

* Replace Postgres → RDS
* Replace MinIO → S3
* Replace JWT-only auth → Cognito
* Replace Docker Compose → ECS/Lambda + API Gateway

Token -

ghp\_ENujw54cgCe98UakfFRQgDvoSswsCj0Zhh20