



RECALLCODE AI — UNIFIED MASTER PRODUCT & ENGINEERING DOCUMENT

Version: 3.0 (Merged & Enhanced)

Date: December 4, 2025

Authors: Saheli + ChatGPT (Mentor) + Grok insights

Purpose: Complete blueprint for RecallCode: MVP → AI Coach → Full SaaS

This document contains:

- PRD
- SRS
- Architecture (HLD + LLD)
- System design diagrams (text-based)
- Backend models
- APIs (REST + GraphQL)
- Background jobs
- Data pipelines
- LeetCode sync
- AI coach behavior
- SRS engine
- Daily plan generator
- Frontend structure
- Deployment plan
- Roadmap (Phase 1–3)
- Security, DevOps, NFR

Ready to send straight into Cursor or Claude Code.

1. EXECUTIVE SUMMARY

1.1 WHAT IS RECALLCODE AI?

RecallCode AI is a lifelong coding brain for developers.

It does what LeetCode, NeetCode, Anki, and ChatGPT cannot:

- remembers everything YOU learned

- predicts exactly what you'll forget
- gives 5 perfect problems per day
- understands your weak DSA patterns
- coaches you using AI trained on your own history

It is essentially:

"Notion × SRS × AI × Your private mentor × LeetCode history → one system."

You become:

- Consistent
- Confident
- Interview-ready every day

No more "forgot everything after job"

No more "where do I start?"

1.2 MVP TIMELINE

4 Weeks → Fully working MVP

12 Weeks → AI-powered version (your v2.0)

6 Months → Full SaaS with monetization

1.3 WHY BUILD THIS?

Because:

- Forgetting DSA is universal
- People spend months relearning before every interview
- No existing tool solves memory decay
- Your idea is blue ocean — nothing identical exists
- It becomes an insane portfolio project

2. PRODUCT VISION

VISION

Create the world's first long-term memory engine for devs—a tool you use for 20 years, not 2 months.

CORE PHILOSOPHY

“Learning matters.
Remembering matters more.”

3. USER PERSONAS

PERSONA A — BUSY SOFTWARE ENGINEER

Has job
Doesn't remember DSA
Wants 10–20 mins/day structured learning

PERSONA B — JOB SEEKER / PLACEMENT PREP

Needs consistent recall
Wants to see progress
Wants AI feedback

PERSONA C — INTERVIEW CANDIDATE

Needs practice plans
Wants a weakness map
Needs mock interviews

4. USER JOURNEY

4.1 ONBOARDING

- Sign up
- Enter LeetCode username
- Sync all solved/submitted problems
- AI generates weakness DNA map
- Show “Your Coach’s Plan for Today”

4.2 DAILY FLOW

- Open app (5–10 mins/day)
- Review 3 SRS problems

- Practice 2 new AI-selected problems
- Rate recall (1–5)
- AI coach gives hints

4.3 WEEKLY FLOW

- Heatmap
- Weakness prediction
- "You will forget Trees in 7 days"
- Suggest practice

5. FEATURES

We merge both documents into a 3-phase roadmap.

5.1 PHASE 1 — MVP FEATURES (4 WEEKS)

These are the core building blocks you MUST deliver first.

- User Auth (JWT)
- Add Problem Manually
- SRS Engine (custom SM-17)
- Daily Review Queue
- Problem Notes (Markdown)
- Code Editor (Monaco)
- Basic AI Hint
- Dashboard (streak, due problems)
- Judge0 Basic Integration
- Offline-first PWA support (simple)

This version allows:

- Manual problem entry
- Daily habit
- Revising
- Remembering

This is where you START.

5.2 PHASE 2 — AI-POWERED SYSTEM (WEEKS 4-12)

(Your Grok v2.0 features integrated here.)

- LeetCode Full Sync
 - Fetch history
 - Parse code
 - Parse patterns
- Weakness DNA Mapping
 - Pattern success rates
 - Difficulty profile
 - Error clusters
- AI Coach (Context-Based)
 - Uses:
 - your code
 - your errors
 - your patterns
 - your weak concepts
- Daily AI Coach Plan
 - 3 SRS problems due today
 - 2 new problems from weak patterns
 - Predict what you'll forget next
- Smart Notes
 - AI summaries
 - Convert notes → structured knowledge
 - Pattern extraction
- Dashboard Pro
 - Mastery %
 - Pattern heatmap
 - Forgetting prediction

5.3 PHASE 3 — FULL SAAS (MONTHS 3-6)

- System Design SRS
- Mock Interview Simulator
- Real-time LeetCode Sync
- Community Leaderboard
- Mobile App (Expo)
- Subscription Model (\$99/year Pro)
- Notion export

6. FUNCTIONAL REQUIREMENTS (FULL SRS)

- F1. User Authentication
 - JWT
 - Refresh token
 - Password reset
- F2. Problem Management
 - Add manually
 - Edit
 - Delete
 - Notes
 - Code storage
- F3. LeetCode Sync (Phase 2)
 - Fetch submissions
 - Parse difficulty
 - Parse success/failure
 - Map patterns
 - Store code
- F4. SRS Review Engine
 - Rate 1–5
 - Compute next review
 - Log review
 - Display daily queue
- F5. AI Coach
 - Context-based explanation
 - Failure clustering
 - Pattern hints
 - Follow-up questions
- F6. Daily Plan Generator
 - Combine SRS + Weakness Map
 - 5-problem plan
 - Send as notification
- F7. Dashboard
 - Heatmaps
 - Mastery chart
 - Pattern scores

7. NON-FUNCTIONAL REQUIREMENTS

- API responsetime <200ms

- Sync <30 seconds
- 99% uptime
- Scalable worker pool
- Secure code execution
- 256-bit JWT
- Rate limiting
- Mass import safe

8. HIGH-LEVEL SYSTEM DESIGN (MERGED)

8.1 SYSTEM COMPONENTS

- Frontend: Next.js 15
- Backend: Django 5 + DRF + Strawberry GraphQL
- DB: PostgreSQL
- Cache/Broker: Redis
- Workers: Celery
- AI: Grok/Claude/Gemini
- Code Runner: Judge0 (Phase 1)
- Vector Search (Phase 2): pgvector

8.2 SYSTEM ARCHITECTURE SUMMARY

Frontend (Next.js)

↓ GraphQL/REST

Backend (Django)

↓

Postgres (Models)

↔

Redis (caching, broker)

↓ Celery

Workers (sync, SRS, AI tasks)

↔ LLM APIs

↔ LeetCode scraper

9. DETAILED BACKEND ARCHITECTURE

Apps:

- users
- problems
- leetcode
- coach
- srs
- ai

Each app has:

- models.py
- schema.py (GraphQL)
- views.py (REST optional)
- services.py
- tasks.py

10. DATABASE SCHEMA (MERGED)

10.1 USER MODEL

- User:
 - id
 - username
 - email
 - leetcode_username
 - weakness_scores (JSON)
 - created_at

10.2 LEETCODESUBMISSION

- user
- problem_title
- problem_slug
- difficulty
- code
- language
- runtime

- memory
- is_solved
- patterns (JSON list)

10.3 SRSREVIEW

- submission
- repetitions
- ease_factor
- interval_days
- next_review
- rating

10.4 DAILYPLAN

- user
- date
- problems (JSON)

11. API SPECIFICATIONS (MERGED)

11.1 GRAPHQL QUERIES

- me
- dailyPlan(date)
- weaknessMap
- submissions(filter)

11.2 MUTATIONS

- syncLeetCode(username)
- rateReview(submissionId, rating)
- chatWithCoach(query)
- addProblem(input)
- updateProblem(id, input)

12. SRS ENGINE (MERGED VERSION)

The merged algorithm includes:

- SM-2 base
- SM-17 improvement
- Coding-specific runtime/memory penalty
- Pattern-based difficulty adjustment

Already fully coded in your v2 version (and included above).

13. BACKGROUND JOBS

Celery Tasks:

- leetcode_sync_task
- calculate_weaknesses
- generate_daily_plans
- update_srs
- ai_hint_task
- pattern_extraction_job

14. SEQUENCE DIAGRAMS

14.1 LEETCODE SYNC

- User → Backend: syncLeetCode
- Backend → Celery: enqueue sync
- Celery → LeetCode API: fetch data
- Celery → DB: store submissions
- Celery → WeaknessCalc: compute scores
- Backend → User: sync result

14.2 DAILY PLAN

- Celery Beat → generate_daily_plans
- Plan → Redis cache
- Frontend → Backend: dailyPlan

- Backend → Redis: fetch plan
- Return plan

15. FRONTEND ARCHITECTURE

src/app/

- auth/
- dashboard/
- coach/
- problems/
- api/graphql.ts

components/

- CoachPlanCard
- MonacoCodeEditor
- PatternHeatmap
- WeaknessCard
- ProblemCard

16. DEPLOYMENT

- Frontend: Vercel
- Backend: Railway/Render
- DB: Railway Postgres
- Redis: Upstash
- Workers: Railway services
- CI/CD: GitHub Actions

17. SECURITY

- JWT
- CORS restricted
- Rate limiting
- Input sanitization

- Judge0 container isolation
- AI query abuse protection

18. MONITORING

- Sentry
- PostHog
- Redis slowlog
- Django request logging
- Worker logs

19. ENGINEERING BEST PRACTICES

- Fat services, thin views
- SQL indexing
- Redis caching for heavy queries
- Feature Flags for new features
- Tests for SRS + sync flows

20. ROADMAP (MERGED FINAL)

- Phase 1 (Weeks 1–4) – MVP
 - CRUD
 - SRS engine
 - Daily review
 - AI hints
 - Notes
 - Dashboard
- Phase 2 (Weeks 4–12) – AI Brain
 - LeetCode sync
 - Weakness detection
 - AI coach
 - Pattern extraction
 - Smart plans
- Phase 3 (3–6 months) – Full SaaS
 - System Design module

Mock interviews

Mobile app

Billing

Community