PROJECT SHOWCASE / ENGINEERING & PRODUCT PORTFOLIO

Bills Reminder

Technical Case Studies

Prepared for professional review | November 4, 2026 | English edition

Evidence note: Based on prior project discussions, described architecture, selected development/debugging history and portfolio records available at preparation time. This is a portfolio case study, not a fresh source-code audit, external security review, market study or verified commercial due diligence. Planned capabilities and unverified outcomes are labeled accordingly.

Bills Reminder | Technical Case Study

Recurring payment organization | Stage: Backend/data model and reminder scheduling worked on; complete feature parity unverified

Technical summary

Bills Reminder is a personal payment-organization application intended to consolidate recurring bill schedules, payment status and proactive notifications. Development history includes MongoDB Atlas collections, payment records, notification logs and work on job locking for scheduled processing.

Problem and engineering context

Recurring bills vary in frequency, due date and status. Manual calendars and disconnected inbox reminders can lead to missed payments, poor visibility and duplicate alerts.

Technology stack

- Next.js-based web administration and MongoDB Atlas persistence discussed in project history.
- Bill, BillPayment, NotificationLog and BillOverride domain models.
- Scheduler/job-lock approach for processing upcoming bill notifications.
- Email delivery workflow and weekly digest concept.
- Docker/VPS deployment context alongside other user applications.

System architecture and component boundaries

- User-facing bill management for recurring schedules, status and history.
- Billing domain: recurring bill definition separated from individual payments/overrides.
- Scheduler: select due reminders, acquire a short-lived job lock, emit notifications and log outcomes.
- Persistence: maintain a per-user boundary and normalized date/time semantics.
- Notification subsystem: idempotency controls and retryable delivery recording.

Data model and domain contracts

- Bill: owner, merchant label, amount, cadence, due date and notification preferences.
- BillPayment: amount/date/payment status for a specific recurrence.
- BillOverride: exceptional due date or amount changes for one instance.
- NotificationLog: delivery intent, channel, outcome, retry count and deduplication key.
- Job lock: owner, lock expiration and TTL index to support safe concurrent schedules.

Primary application flows

- Create a recurring obligation and choose notification preferences.
- Compute the next due occurrence including exceptions.
- Scheduler evaluates alert windows, acquires lock and records notification outcome.
- Mark a bill as paid; show history without destroying the recurring bill definition.

Engineering challenges and tradeoffs

- A previously observed joblocks index lacked the intended TTL expireAfterSeconds setting; final correction was not confirmed.
- Multiple scheduler instances may issue duplicate emails without atomic lock and deduplication.
- Monthly recurrence edge cases: 29th-31st, leap years, DST and calendar changes.
- Email provider failures require retry/backoff and a defensible sending log.
- Sensitive financial reminders require careful permissions, retention and encryption policies.

Available evidence and limits

- Prior MongoDB shell checks and named collections for bills/payments/overrides/notifications.
- TTL locking investigation directly discussed; final deployed behavior was not verified.
- CSV import, complete dashboard and polished production notification system cannot be claimed without source review.

Recommended technical validation / next steps

- Test index shape with actual MongoDB metadata and TTL expiry.
- Unit test recurrence generation and exception behavior over multiple years.
- Integration test concurrent schedulers for duplicate delivery.
- Add preview/dry-run mode, observability and privacy controls.
- Validate value beyond built-in banking and calendar alerts.

Recruiter interview prompts

- What prevents duplicate notifications under concurrent scheduled workers?
- How are month-end recurrences and overrides calculated?

- What is the correct MongoDB TTL configuration and its limits?

Disclosure: This report intentionally distinguishes observed discussions from proposed engineering improvements. Tests, benchmarks, revenue, user adoption and security compliance are not asserted without supporting evidence.