TaskFlow – Project Blueprint
From MVP to Full Product

1. What is TaskFlow?

TaskFlow is a small, focused team/family task management app. It lets users create projects, add tasks under thos
The primary goals of this project are:
- To practice real-world full■stack development.
- To strengthen understanding of backend (Node, Postgres), mobile (React Native), DevOps (CircleCI, Fastlane), a
- To build a strong portfolio piece that looks good on CV, GitHub, and in interviews.

2. Overall Vision

In the end state, TaskFlow will be a cross-platform system with:
- A backend API (Node + Postgres) deployed on AWS.
- A React Native mobile app (Android first, then iOS).
- A Web client consuming the same API.
- Push notifications via Firebase (FCM) for assigned tasks and due-date reminders.
- Basic DevOps/CI/CD setup with GitHub, CircleCI, and Fastlane for mobile delivery.
- Clean, documented code and a professional README that tells the project story.

3. Phase 1 – MVP (Backend + React Native + Basic DevOps)

Goal: Get a complete, simple vertical slice working from database → API → mobile app, with proper version contro

3.1 Backend (Node + Postgres)

Tech:
- Node + Express + TypeScript
- Postgres (via Prisma or another ORM)
- JWT authentication

Core entities:
- User (id, name, email, password_hash, role, timestamps)
- Project (id, title, description, owner_id, timestamps)
- Task (id, title, description, due_date, status, priority, project_id, assignee_id, timestamps)
- DeviceToken (id, user_id, fcm_token, platform, created_at) – used later for push notifications

MVP API endpoints:
- Auth:
  - POST /auth/register – register a user
  - POST /auth/login – login and receive JWT + user info
  - GET /me – get current user profile from token

- Projects:
  - POST /projects – create project for current user
  - GET /projects – list all projects for current user

- Tasks:
  - POST /projects/:projectId/tasks – create task under a project
  - GET /projects/:projectId/tasks – list tasks for a project
  - PATCH /tasks/:id – update task (title, description, status, etc.)
  - DELETE /tasks/:id – delete task

Focus in this phase:
- Clean API design with proper validation and error handling.

- Secure auth with hashed passwords (bcrypt) and JWT.
- Simple but clear relational schema and foreign keys.

3.2 Mobile App (React Native – Android first)

Tech:
- React Native
- State management (e.g., simple context or a lightweight store)
- HTTP client (axios or fetch)

MVP screens:
- LoginScreen
  - Email + password
  - Calls /auth/login
  - Saves JWT token (AsyncStorage or MMKV) and user info
  - Navigates to main app on success

- ProjectsListScreen
  - Calls /projects with auth header
  - Displays list of projects
  - Floating button or simple UI to add a new project

- ProjectDetailScreen
  - Shows selected project details
  - Calls /projects/:id/tasks to load tasks
  - Displays tasks with title and status
  - Allows adding a new task
  - Allows updating task status (Todo / In Progress / Done)

- TaskFormScreen
  - Used for creating and editing tasks
  - Contains fields for title, description, status, and optionally due date

Focus in this phase:
- Stable end-to-end flow: login → view projects → view tasks → create/update tasks.
- Handling loading states, errors, and basic UX correctly.

3.3 Dev Setup from Day One (GitHub, CircleCI, Fastlane basics)

Even in phase 1, the project should be structured and professional:
- GitHub:
  - Separate repositories or a monorepo with /backend and /mobile.
  - Clear commit messages and branches (feature branches, main branch).

- CircleCI (initial):
  - Basic workflow that runs on push/PR.
  - For backend: install dependencies, run tests (or at least a build/lint step).
  - For mobile: install dependencies, ensure the app builds in debug mode (later can be expanded).

- Fastlane (initial for Android):
  - Setup fastlane in /mobile/android.
  - Define a lane that builds a release APK/AAB.
  - This will later connect to Play Store internal testing.

Outcome of Phase 1:
- A working MVP where a user can log in, manage projects and tasks from an Android app.
- Code is in GitHub with basic CI, and mobile build automation via Fastlane.

- You can already show this in interviews as a complete vertical slice.

4. Phase 2 – Web + AWS Deployment + Better Notifications

Once the MVP is stable and you are comfortable with the flow, move to expanding the system.

4.1 Web Client (React)

Tech:
- React (Vite or Next.js)
- Same API as the mobile app

Features:
- Login page (email/password) using /auth/login.
- Projects page listing projects and letting user create/edit projects.
- Project detail page showing tasks, with ability to add/edit tasks.

Goal:
- Demonstrate that the backend is reusable across platforms.
- Improve React skills for MERN-style interviews (even though DB is Postgres, concepts are same).

4.2 AWS Deployment (Backend + Postgres)

Backend:
- Containerize or use a simple Node deployment approach.
- Deploy to an AWS service (e.g., Elastic Beanstalk, ECS, EC2, or Lightsail).

Database:
- Use AWS RDS (Postgres) or another hosted Postgres solution.
- Migrate schema from local to cloud DB using migrations.

Other:
- Configure environment variables (DB URL, JWT secret, etc.) securely.
- Set up logs and health checks.

Goal:
- Learn how to deploy a real backend to the cloud.
- Understand environment configuration, production DBs, and error monitoring basics.

4.3 Push Notifications (Firebase FCM)

Mobile:
- Integrate @react-native-firebase/messaging (or equivalent).
- Obtain device FCM token and send it to backend via /device-tokens.

Backend:
- Integrate Firebase Admin SDK.
- Implement a NotificationService that can send push notifications to a user based on userId.
- Use this service:
  - When a task is created with an assigneeId, send a “New task assigned” push.
  - Later, create a cron/queue worker that checks for tasks nearing due_date and sends reminders.

Goal:
- Learn how to connect mobile clients, backend, and third-party service (FCM).
- Demonstrate knowledge of async work, workers, or cron jobs.

Outcome of Phase 2:

- Same backend now used by both React Native app and Web app.
- Backend deployed to AWS with a real Postgres DB.
- Mobile app supports basic push notifications for assigned tasks and reminders.
- Stronger story for DevOps, cloud, and cross-platform development.

5. Phase 3 – Polish, iOS, and Extras

Once the system is fully working, focus on refinement and extra features.

5.1 iOS App

- Reuse most of the existing React Native code.
- Configure iOS project, signing, and FCM for iOS.
- Add iOS lanes to Fastlane for building and distributing test builds.

5.2 UX and Features

Possible extras:
- Filters and sorting on tasks (by status, priority, due date).
- Simple activity log per task (who changed what and when).
- Basic analytics (e.g., number of completed tasks per week, per project).
- Role-based access (owners vs members) if you want to extend to multi-user teams.

5.3 Documentation and Presentation

- Clean README describing:
  - Project overview
  - Tech stack
  - Architecture diagram (if you want to add one later)
  - Local setup instructions
  - Deployed URLs (API, web app) if applicable

- Optional short case study / blog-style write-up:
  - Problem you solved
  - Design decisions
  - Challenges faced
  - What you learned

Outcome of Phase 3:
- A polished, multi-platform product (Android, Web, and later iOS) with a real backend on AWS.
- A great central project for your portfolio.
- Clear proof of your skills in:
  - Backend (Node, REST, auth, DB)
  - Frontend/mobile (React Native and React)
  - DevOps/CI/CD (GitHub, CircleCI, Fastlane)
  - Cloud deployment (AWS, Postgres)
  - Notifications and async processing (Firebase FCM, cron/queue).

6. Final Outcome and Why This Project Matters

By the end of this roadmap, TaskFlow will not just be “a todo app”. It will be:
- A complete demonstration of your ability to take an idea from MVP to production-style architecture.
- A strong talking point in interviews:
  - You can explain schema design, API design, authentication, and state management.
  - You can talk about CI/CD, cloud deployment, and mobile release pipelines.
  - You can showcase code quality through your GitHub repo and documentation.

Most importantly, it gives you:
- Deeper understanding of how real-world systems are built and shipped.
- Confidence that you can own a project end-to-end, not just work on isolated tickets.