- as soon as you begin work,report it to your team PM
- when you are about to shut your system,report it to tejas or team-lead
- you work on dev/<your-name>
- until MVP is done, your branch will keep merging with main but once MVP is done your branch will merge to develop and only if it works there then it will be merged to main
- you write tests
- file names are in kebab case eg. “new-file.ts”
- incase you have exams,sick or any kind of leave inform tejas
- incase you’ve worked over-time you get that much eqvivalent time as credits which you can take some time off as long as we are in normal mode and all of them are not taking day off on same day
- you will get feedback form sometimes, it will either be on project or your work-life or just asking for feedback
- you get emails giving you a birds eye view on what is happening, this is not related to your day to day work

Roles

Here are all the roles

Exclusive teammate

- Core teammate

- PM (Project Manager)

  As a Project Manager, you are responsible for the success of all the assigned projects. At times, you will be required to hop into meetings with clients.

  Do's

  - you can assign tasks to other teammates under your supervision
  - you are supposed to do all the merging in develop and then merging to main
  - you are supposed to test the project before merging if it is proper
  - you can call teammates to meetings to make them clear on what is going on
  - you can message any teammate of that project to ask about status of the project
- Don't's
  - no poaching clients [applies all roles as well]
  - prices clients pay will not be set by you [applies all roles as well]
  - unless I tell you to, no sending proposals
- If you are the team-lead and we get 5 stars you get awarded with 4k as bonus for that month. This will increase as we grow bigger as a company. if we get a lead by your referral and he got converted then 10% of what client pays as bonus but no 4k thing that time

there will also be instances in some situations where you need to take my place for a short period of time. bascially the flow will go like how it went with reliant inspection part

Non-exclusive teammate

- contract teammate
- project based teammate

reserve teammate

experimentation teammate

Modes

- NORMAL MODE
- CRITICAL MODE
- LAZY MODE

  work normally, much work wont be there here. you might also get a day off

- CODE IS HACKED MODE

Develop Process

All merges happen **directly** to the `main` branch until MVP is done

- after MVP is done
  - **Develop Branch**:
    - Code is merged into the `develop` branch first.
    - If the code works well on `develop`, it can be merged into `main`.
    - If there are issues, the code should be merged into the `staging` branch for further testing.
  - **Merging Responsibilities**:
    - Only the **Founder** and **Captain** are authorized to perform merges into `main`, `develop`, or `staging`.

Task Completion

- After completing each task, **immediately inform** either the **Founder** or **Project Manager**.
- If you are the **Project Manager**, notify the **Founder** after completing your tasks.

File Naming Convention

- Use **kebab case** for file names (e.g., `my-component.tsx`).
- This applies primarily to web development; for mobile, further conventions can be discussed.

Coding Style

Our coding style is influenced by the following developers and repositories:

- **Sadman**: Sadmann7 on GitHub
- **Antonio**: Code with Antonio
- **Web Prodigies**: Web Prodigies YouTube
- **ShadCN**: We also plan to adopt aspects of shadcn-ui/taxonomy and shadcn-ui/ui in the future for component architecture.

For simplicity, we will currently stick with **Sadman's style**, but a more detailed coding style guide will be created through a dummy project in the future.

- some brief about coding styles

    - **File Naming**: Always use **kebab case** for file names (e.g., `new-feature.tsx`).
    - **Component Naming**: Components should be written in **PascalCase** for clarity (e.g., `FeatureComponent`).
    - **API Calls and Server-Side Code**:
        - Use **`try-catch`** blocks for all API requests and server actions to ensure proper error handling and user-friendly feedback.
        - **Graceful Failures**: Even if an API call fails, ensure that the user is informed gracefully (using error states and toast notifications where appropriate).
    - **Loading and Error States**: Every async operation should include **loading** and **error handling** states in the UI. Provide clear indicators when an action is in progress, and gracefully handle errors.
    - **Write Tests(for bigger clients with over 50k budget and more than or equal to 1 month timeline)**:
        - **Unit Tests**: Always aim to write unit tests, especially for critical logic.
        - **Integration Tests**: Where possible, write integration tests to ensure seamless interaction between different modules.
        - **Prioritize Shipments**: If testing isn't feasible within a tight deadline, ship features without tests, but ensure testing is done post-release.

    - **Experimental Features**: Any features or code tagged as `(try)` are considered experimental. These should be isolated and well-documented, so the team knows they aren't guaranteed to ship to production
    - Whenever you write some code and you want to write some todo write like

// todo:some todo
or
// todo:(some name) something

-

Git workflow

you always work on your branch ie dev/<name>. alternatively you can also fork the repo but when merging it must be dev/<your name>

I dont want all those feat:<something> fix:<something> etc. but keep commits descriptive, explain everything that you've done in that commit

only captain and founder can merge

steps to writing the code

step 1: browse in internet or chatgpt(try to use Cursor pro ask from the company to provide account)

step 2: copy if solution exists, if doesnt exists write on your own

step 3: focus on functionality