Git Repository & Problem Solving and Algorithmic Thinking:

https://github.com/vinsensiuschristo/VhiWEB

Workload Management

To manage the team's workload effectively, I would start by holding a quick meeting with everyone to discuss a few important things.

1. Identify the top priorities
My team and I would determine which features or modules are absolutely essential for the MVP. We'd categorize tasks by urgency: what needs to be done immediately, what's important, and what's non-priority.

2. Assign tasks based on skill and capacity
Everyone on the team usually has something they're really good at. By assigning tasks that align with people's skills and how much they can handle, work gets done faster and more efficiently.

3. Break down large tasks & monitor regularly
Big tasks should be split into smaller, more manageable ones. Then, quick daily check-ins help track progress and address any blockers. This also keeps our time estimates more realistic.

Supporting the Team

If I notice a teammate is starting to feel overwhelmed, the first thing I would do is offer help directly. I'd ask something specific like, "Which part can I help with?" and if I can, I'll help debug or work alongside them.

I believe in collaboration, like pair programming or reviewing code together. Sometimes, two people working together can solve a problem faster than one person alone.

I'd also remind the team not to overwork themselves. If someone looks tired, I'd encourage a short break. I make a habit of celebrating even the smallest progress.

Most importantly, I try to foster an open environment where people feel safe to speak up when they're struggling — without fear of being judged. I believe what matters most is progress, not perfection. As long as we're moving forward, that's already a big step.

Communicating with Stakeholders

If I see a risk that we might miss the deadline, here's what I would do:

1. Be honest early
I would notify stakeholders as soon as I realize we may be delayed. It's better than informing them at the last minute.

2. Explain the real situation clearly
I'd outline what's done, what's still in progress, and what issues the team is facing.

3. Offer realistic solutions
Maybe ask for a few extra days, or suggest releasing a version that focuses only on the most critical features.

4. Emphasize the importance of quality
I'd explain that the goal isn't just to ship on time, but to deliver something stable and useful. Releasing a broken product just to hit a date isn't worth the trade-off.