AI-Assisted Coding: Challenges and Solutions

1. Security Risks

Issue: Secrets or API keys are ending up in frontend code, exposing vulnerabilities.

🧩 What to look out for:

- Keys in .js, .ts, .html, or other frontend-accessible files.
- AI inserting sensitive values directly into config or fetch calls.

🧭 PM Instructions:

- **Set a policy:** No secrets should ever exist in the frontend. Use environment variables or secret managers.
- **Integrate a vault:** Tools like HashiCorp Vault, Azure Key Vault, or AWS Secrets Manager should be used.
- **Code review step:** During pull requests, include a mandatory secret-scan step (automated or manual).
- **AI prompt hygiene:** Ask devs to instruct the AI: “Do not insert actual secrets. Use placeholders.”

✅ Questions to ask the team:

- Have all secrets been externalized to a secure vault or .env?
- Are devs using AI prompts that reinforce secret management best practices?
- Are there automated linters or secret scanners (e.g., GitGuardian)?

2. Code Hallucinations

Issue: AI makes major, unrelated changes to files when asked to do small edits.

🧩 What to look out for:

- Unrequested rewrites or reformatting.
- Subtle logic bugs introduced by hallucinated refactors.

🧭 PM Instructions:

- **Use Git wisely:** Instruct the team to use git diff before and after every AI interaction.
    - This may be overkill as most no-code have auto restore point but use these if needed.
- **Small scope tasks:** Encourage “micro-prompts” (e.g., “Refactor this function” instead of “Fix the page”).
- **Rollback safety:** Ensure local versioning (like Git stash or branch clones) is practiced.
- **Code review mindset:** Encourage team to never trust AI blindly — always validate with tests or logic review.

✅ **Questions to ask the team:**

- Are you reviewing diffs after AI modifications?
- Was this prompt specific enough to avoid scope creep?
- Is test coverage in place to detect AI-introduced regressions?

3. Lack of Structural Clarity

Issue: AI mixes frontend concerns with backend logic, violating separation of concerns.

🧩 **What to look out for:**

- Business logic leaking into frontend components.
- Backend code invoking UI elements, or vice versa.

🧭 PM Instructions:

- **Architecture mapping:** Make sure a clear folder and component structure is documented.
- **Prompt suggestions:** “Keep this logic server-side” or “Frontend only handles presentation.”
- **Review meetings:** Include a structure check in sprint demos or pull request reviews.

✅ Questions to ask the team:

- Is this logic properly scoped to the backend?
- Does the folder structure reflect frontend/backend separation?
- Are your prompts reinforcing architectural constraints?

4. Library Overload

Issue: AI introduces unnecessary third-party libraries, bloating the project and adding maintenance overhead.

🧩 **What to look out for:**

- New package.json dependencies with unclear purpose.
- Libraries used for trivial tasks (e.g., moment.js for formatting a date).

🧭 **PM Instructions:**

- **Define a baseline:** Maintain a list of approved or recommended libraries.
- **Review dependency diffs:** For every PR, check for added packages and justify each.
- **Prompt discipline:** Instruct devs to ask: “Use built-in methods unless absolutely needed.”

✅ **Questions to ask the team:**

- Did we already have a solution in the codebase?
- Is the new library well-maintained and widely used?
- Could this be done with native functionality?

5. Repetitive and Bloated Code

Issue: AI duplicates patterns instead of abstracting reusable components.

🧩 **What to look out for:**

- Repetition of the same HTML/TSX/JS blocks.
- Identical logic blocks not refactored into functions.

🧭 **PM Instructions:**

- **Component-first mindset:** Encourage modular design in all tasks.
- **Design systems:** Promote shared UI kits or logic utilities.
- **Refactoring reviews:** Include a checklist item for “redundancy reduction.”

✅ Questions to ask the team:

- Can this block be refactored into a shared component?
- Do you have reusable utility functions for this logic?
- Are components stored in a shared, organized structure?