15. Team Agreements & Guidelines

- Small, focused teams are best. Ideally 2-3 developers per squad, with mixed skills.
- Share lessons from production issues—what went wrong, how we fixed it, and what we'll do differently next time.
- Encourage regular knowledge-sharing moments (code review sessions, mini demos, informal chats).
- Everyone should keep learning. Whether that's backend, cloud, or soft skills—small steps forward make the whole team better.
- Tools / system
  - Use the same core stack across teams unless there's a strong reason not to.
  - .NET and Azure are the base. (unless we decide otherwise)
  - Document exceptions and rationale when tools diverge.
- Naming
  - Use clear, consistent terms across services (e.g. "club," "member," "account").
  - Match code names, API routes, and database columns where possible.
  - Avoid inventing new names for old concepts.
- First design, then build
  - Before coding, think through architecture, naming, data structures, and edge cases.
  - Small diagrams or lightweight write-ups are enough—but design comes first.
  - **AI is also coding**
- Process + QA
  - What matters is progress, planning, and transparency.
  - Pull requests go through review by at least one peer.
  - Use automated tests and CI/CD pipelines to catch issues early.
  - QA should test all the new features, but devs and pms should test them as well.
- Separation of concerns
  - Keep **API and UI in separate projects**, ideally with their own deploy pipelines.
  - Interfaces should talk via versioned APIs, not by sharing code or DB directly.
- Git strategy
  - Use short-lived feature branches off `main` or `develop`.
  - Merge regularly to avoid conflicts.
  - Use clear commit messages; squash if needed to keep history clean.
- AI usage
  - Use AI tools (like Copilot, Cursor, etc.) for brainstorming, summaries, code suggestions—but always review and adapt.
  - Don't paste in sensitive data.
  - Don't rely on AI to make architectural or security decisions—double-check with humans.