

Les 9 Huiswerk

Bouw je eigen app met de 5 Cursor-technieken

Vak: AI-Assisted Development

Opleiding: NOVI Hogeschool Utrecht

Deadline: Voor de volgende les

Tech stack: Next.js 16, TypeScript, Tailwind CSS

Overzicht

In de les heb je 5 Cursor-technieken geleerd en toegepast op LinkVault. Nu is het tijd om te laten zien dat je deze technieken zelfstandig kunt toepassen. Je gaat een **eigen app** bouwen van scratch — geen LinkVault!

Kies je app

Kies een van de volgende projecten (of bedenk er zelf een):

Optie	Omschrijving
Recepten app	Recepten opslaan met ingrediënten, bereidingswijze en tags
Film tracker	Films bijhouden met rating, genre en watchlist
Notitie app	Notities met categorieën en zoekfunctie
Eigen idee	Overleg dit eerst met Tim voordat je begint!

Tech stack: Next.js 16, TypeScript, Tailwind CSS (net als in de les).

Checklist: Wat moet je opleveren?

Gebruik deze checklist om te controleren of je alles hebt gedaan. Elk punt is **verplicht** (tenzij anders aangegeven).

- 1 **Cursor Rules** — `.cursor/rules/` map aangemaakt met minimaal 2 rule bestanden (met YAML frontmatter!)
- 2 **Plan Mode** — Plan Mode gebruikt om je app te plannen (maak een screenshot van je plan)
- 3 **Agent Mode** — Agent Mode gebruikt om minimaal 2 features te bouwen
- 4 **Debug Mode** — Debug Mode gebruikt voor minstens 1 bug (maak een screenshot van je debug sessie)
- 5 **Tests** — Tests geschreven met de agent (minimaal 3 unit tests)
- 6 **Code Review** — "Find Issues" gedraaid en eventuele issues gefixt
- 7 **Semantic Commits** — Semantic commits gemaakt (feat:, fix:, docs:, etc.)
- 8 **GitHub** — Alles gepusht naar een GitHub repository

Bonus

Pull Request aangemaakt en door Cursor laten reviewen

Wat lever je in?

1. **GitHub repository link** — Zorg dat de repo public is (of nodig Tim uit als collaborator)

2. **Korte reflectie (200 woorden)** — Beantwoord deze vragen:

- Welke Cursor-technieken werkten het beste voor jou?
- Waar liep je vast?
- Wat doe je de volgende keer anders?

3. **Screenshots** van:

- Je Plan Mode plan
- Een Debug Mode sessie
- Je test resultaten (terminal output)

Tips

- Begin met Plan Mode! Laat Cursor eerst een plan maken voordat je gaat bouwen.
- Commit vroeg en vaak. Gebruik semantic commits vanaf het begin.
- Loop je vast? Gebruik Debug Mode — beschrijf de fout en laat Cursor helpen.
- Houd je prompt requests in de gaten. Je hebt 500 fast requests per maand op het Student Plan.

Appendix A: Keyboard Shortcuts

Alle belangrijke Cursor sneltoetsen op een rij:

Actie	Windows / Linux	macOS
Chat openen	Ctrl+L	Cmd+L
Plan Mode toggle	Shift+Tab	Shift+Tab
Inline Edit	Ctrl+K	Cmd+K
Composer	Ctrl+I	Cmd+I
Autocomplete accepteren	Tab	Tab
Terminal openen	Ctrl+`	Ctrl+`
Command Palette	Ctrl+Shift+P	Cmd+Shift+P

Appendix B: .cursor/rules Templates

Maak een `.cursor/rules/` map in je project root en voeg daar `.mdc` bestanden toe. Hieronder staan templates die je kunt kopiëren en aanpassen.

project.mdc (algemene projectregels)

```
# .cursor/rules/project.mdc
---
description: Algemene projectregels en conventies
alwaysApply: true
---

# Project Regels

## Tech Stack
- Next.js 16 met App Router
- TypeScript (strict mode)
- Tailwind CSS voor styling

## Conventies
- Gebruik functionele componenten met TS interfaces
- Bestandsnamen in kebab-case
- Componenten in PascalCase
- Gebruik async/await i.p.v. .then() chains

## Mappenstructuur
- src/app/ -- App Router pagina's en layouts
```

- src/components/ -- Herbruikbare componenten
- src/lib/ -- Utility functies en types
- src/types/ -- TypeScript type definities

components.mdc (componentregels)

```
# .cursor/rules/components.mdc
---
description: Regels voor React componenten
globs: "src/components/**/*.tsx"
---

# Component Regels
- Props interface boven de component
- Gebruik "use client" alleen wanneer nodig
- Gebruik Tailwind utility classes
- Alle images hebben een alt attribuut
- Interactieve elementen hebben aria-labels
```

styles.mdc (styling regels)

```
# .cursor/rules/styles.mdc
---
description: Regels voor styling en Tailwind gebruik
globs: "**/*.css,**/*.tsx"
---

# Styling Regels
- Gebruik de standaard Tailwind kleurenpalet
- Responsive design: mobile-first (sm:, md:, lg:)
- Geen component-specifieke CSS bestanden
- Gebruik globals.css alleen voor CSS vars en base
```

Appendix C: Prompt Templates

Goede prompts maken het verschil. Hieronder staan templates die je kunt gebruiken voor elke techniek.

1. Plan Mode prompt

```
Ik wil een [app naam] bouwen met Next.js 16,  
TypeScript en Tailwind CSS.
```

```
De app moet het volgende kunnen:
```

- [Feature 1]
- [Feature 2]
- [Feature 3]

```
Maak een gedetailleerd plan met:
```

1. Mappenstructuur
2. Benodigde componenten
3. Data model / types
4. Stappen om te bouwen (in volgorde)

2. Agent Mode feature prompt

```
Bouw de [feature naam] feature voor mijn app.
```

```
Wat het moet doen:
```

- [Beschrijving van de feature]
- [Verwacht gedrag]
- [Edge cases]

```
Gebruik bestaande types uit src/types/
```

```
en componenten uit src/components/.
```

```
Volg de regels in .cursor/rules/.
```

3. Debug Mode prompt

```
Ik krijg de volgende fout:
```

```
[Plak hier de foutmelding]
```

```
Wat ik verwacht: [verwacht gedrag]
```

```
Wat er gebeurt: [werkelijk gedrag]
```

```
Het relevante bestand is [bestandsnaam].
```

```
Kun je de bug vinden en fixen?
```

4. Test writing prompt

Schrijf unit tests voor [component/functie].

Test minimaal:

- Het happy path (normaal gebruik)
- Edge cases (lege input, lange strings)
- Error handling

Gebruik de bestaande test setup.

5. Semantic commits prompt

Maak semantic commit messages voor de huidige wijzigingen.

Gebruik dit formaat:

feat: voor nieuwe features
fix: voor bugfixes
docs: voor documentatie
refactor: voor code refactoring
test: voor tests
chore: voor overige taken

6. Code review prompt

Review de code in [bestandsnaam of map].

Let op:

- TypeScript best practices
- Performance issues
- Security problemen
- Toegankelijkheid
- Code duplicatie

Geef concrete suggesties voor verbeteringen.

Appendix D: Troubleshooting

Plan Mode verschijnt niet

Probleem	Oplossing
Plan Mode optie niet zichtbaar	Zorg dat je Cursor versie up-to-date is (Help > Check for Updates)
Toggle doet niets	Probeer Shift+Tab in het chat venster, niet in de editor
Plan wordt niet gegenereerd	Zorg dat je prompt duidelijk genoeg is — geef context

Agent Mode voert niets uit

Probleem	Oplossing
Agent maakt geen bestanden aan	Controleer of Agent Mode actief is (niet Ask Mode)
Agent stopt halverwege	Druk op "Continue" of geef een follow-up prompt
Agent maakt fouten	Wees specifieker in je prompt, verwijst naar bestanden

.cursor/rules worden niet opgepikt

Probleem	Oplossing
Rules hebben geen effect	Controleer dat de map .cursor/rules/ heet
YAML frontmatter fout	Controleer de --- blokken bovenaan elk .mdc bestand
Globs matchen niet	Check je glob patronen — src/**/*.* matcht alles onder src/

Tests draaien niet

Probleem	Oplossing
npm test geeft fout	Controleer of Vitest is geïnstalleerd: npm install -D vitest
Tests worden niet gevonden	Bestandsnaam moet eindigen op .test.ts of .test.tsx
Import errors in tests	Controleer je tsconfig.json paths en test configuratie

Git / GitHub problemen

Probleem	Oplossing
git push geeft permission denied	Controleer of je SSH key of token is ingesteld
Repo niet gevonden	Maak eerst een repo aan op github.com
Merge conflicts	Gebruik git status om conflicten te zien, los op in Cursor

Veel succes en plezier met bouwen! Kom je er niet uit, stel je vraag in het les-kanaal.