

Reflectus alternatyvė

Pokalbio išrašas (mokytojos pusės darbų planas)

Sugeneruota: 2026-01-02

Šaltiniai: Reflectus_alternatyve_santrauka.pdf, PROJEKTO_INFORMACIJA.md

Vartotojas

gerai dabar is sito failo ir visos kitos info. padaryk labai nuoseklu darbu planas su kur jei eikia code exmples ir kad dabar padarom viska ka apterm ypac is sito failo. viska kas yra sitam project folder Reflectus alternatyve. pradekim is mokytojos puses . tai savrbiausia dalis. netiks is mokytojo. nu tai kas aptarem tam faile. su kodo pavyzdziais kur reikia

Vartotojas

cia daugiau info kuria turi remtis. dirbk toliau

Asistentas

Žemiau — labai nuoseklus darbų planas Reflectus alternatyvė projektui, pradedant nuo mokytojos pusės (tai bus “šerdis”), ir užtikrinant, kad viskas atitiktų specifikaciją: planavimas+galiojimas, analitika, universalūs atsakymai „Nenoriu atsakyti“/„Nežinau“, eksportai, privatumas ir saugumas.

0) Tikslas ir “mokytojos srautas” (ką turim turėti)

Mokytoja turi galėti:

- 1) Kurti refleksijas kaip įvykius (kalendoriuje), su galiojimo langu ir priminimais.
- 2) Priskirti refleksiją klasei/grupei (prisijungimas per kodą/QR).
- 3) Matyti realiu laiku: kas užpildė / kas ne, progresą %, likusį laiką, ir greitai valdyti (perkelti/redaguoti/trinti su patvirtinimais).
- 4) Turėti peržiūrą (review) ir komentarus (jau yra bazė).
- 5) Matyti analitiką: individas/grupė, momentas/laikas, filtrai ir statusai („neužpildė“, „nenoriu atsakyti“, „vėluota“).
- 6) Eksportuoti CSV/PDF (su N rodymu, privatumo žymomis).
- 7) Privatumo režimai (vardinis / pseudo-anon / anon agregatas su min N) + audit log + apsaugos (rate limit, bruteforce, CSRF jei cookies).

Kritinis UX reikalavimas (kiekvienam klausimui):

- Visada turi būti 2 papildomi pasirinkimai: „Nenoriu atsakyti“ (užskaitomas statusas) ir „Nežinau“ (paleidžia 1–3 nukreipiamus klausimus ir po to priverčia grįžti atsakyti į originalų, nebent pasirenka „Nenoriu atsakyti“).
-

1) Greitas esamos būklės “check” (kad planas atitiktų realų kodą)

Pagal projekto aprašą dabar turite:

- frontend/ (React+Vite) su mokytojo puslapiais (/teacher, /teacher/review, /teacher/classes, /teacher/tasks/new).
- backend/ (Node+Express) su auth/groups/questionnaires endpoint'ais.
- Kol kas veikia mock logika (localStorage), o backend integracija planuojama.

Ubuntu komandos (2 terminalai):

Terminalas A (backend):

```
cd /KELIAS/IKI/reflect/backend
npm install
npm run dev
```

Terminalas B (frontend):

```
cd /KELIAS/IKI/reflect/frontend
npm install
npm run dev
```

Rezultatas: matysi UI ir galėsi testuoti mokytojos srautus.

2) Duomenų modelis (pamatai mokytojos pusei)

Kad kalendorius+galiojimas+analitika veiktų, reikia aiškiai atskirti:

2.1 Esmė: “Refleksijos įvykis” (Scheduled Questionnaire)

Mokytoja sukuria įvykį klasei:

- startAt / endAt (galiojimo langas)
- template/questions versija (kad pakeitimai po atsakymų nesulaužytų statistikos)
- statusai: active/expired/cancelled

2.2 Atsakymas (Response) turi turėti statusą

Kiekvienas klausimas turi leisti:

- answered (normalus)
- skip (Nenoriu atsakyti)
- unknown (Nežinau → nukreipiamieji → privalomas grįžimas atsakyti)

2.3 Minimalūs klausimų tipai MVP

MVP tipai: Taip/Ne, Šviesoforas, Skalė, Emocijos, Atviras tekstas.

3) Backend darbų planas (mokytojos pusė pirmiausia)

3.1 DB įjungimas (kad neliktų localStorage)

Tikslas: viskas (įvykiai, atsakymai, komentarai, audit log) turi būti DB.

Rekomendacija MVP: MongoDB + Mongoose (greičiausia startui), arba Postgres (jei norėsi griežtų ataskaitų).

Modelių minimalus rinkinys:

- User (teacher/student)
- Group/Class
- ScheduledQuestionnaire (įvykis kalendoriuje)
- Question (įvykio klausimų snapshot'as)
- Response (studento pateikimas)
- AuditLog (kas ką kūrė/keitė/peržiūrėjo/eksportavo)

3.2 API (mokytojos branduolys)

Sukurk/užbaik šiuos endpoint'us (REST):

A) Kalendorius + kontrolė

- POST /api/schedules – sukurti refleksijos įvykį (mokytoja)
- GET /api/schedules?from=...&to=...&groupId=... – mokytojos kalendoriaus vaizdas
- PATCH /api/schedules/:id – perkelti/redaguoti (su įspėjimais, jei jau yra atsakymai)
- DELETE /api/schedules/:id – trinti su “ar tikrai prarasti duomenis?”

B) Užpildymo progresas

- GET /api/schedules/:id/progress – kas užpildė / kas ne, %.

C) Review + komentarai

- GET /api/responses?status=submitted&groupId=...
- POST /api/responses/:id/comment – mokytojo komentaras + statuso keitimas (reviewed/commented).

D) Analitika

- GET /api/analytics/schedule/:id/summary – grupės momentinė statistika
- GET /api/analytics/question/:questionId/trend?from..to.. – laiko dinamika
- filtrai: group, time, status (exclude “nenoriu atsakyti” kai reikia)

E) Eksportai

- GET /api/export/schedule/:id.csv
- GET /api/export/schedule/:id.pdf (server-side PDF generator).

3.3 “Nenoriu atsakyti” + “Nežinau” realizacija (backend taisyklės)

Backend'e laikyk vieną taisyklę:

- jei answer status = unknown → jis negali likti galutinis, turi virsti answered arba skip po nukreipiamųjų (frontend privers, bet backend turi validuoti).

Pavyzdinis atsakymo objektas (DB):

```
{
  questionId: "q1",
  status: "answered" | "skip" | "unknown",
  value: "...",
  unknownFlow: {
    steps: [
      { stepId: "u1", value: "..." },
      { stepId: "u2", value: "..." }
    ],
    resolvedTo: "answered" | "skip"
  }
}
```

3.4 Priminimai + galiojimo pabaiga (scheduler/queue)

Reikia: planavimas, galiojimo langas, priminimai, paskutinis priminimas prieš pabaigą, po pabaigos pažymėti “neužpildyta” (ar “vėluota” jei leidžiama).

MVP sprendimas:

- Naudoti job scheduler (pvz. node-cron) pradžiai.
- Vėliau: queue (BullMQ/Redis) — stabiliau VPS’e.

4) Frontend darbų planas (mokytojos UI pirma)

4.1 Mokytojos “Valdymo skydelis = kalendorius”

Sukurk 3 vaizdus:

- Savaitė (default)
- Mėnuo
- Diena

Kiekvienas įvykis rodo:

- klasė/grupė
- likęs laikas
- užpildymo progresas %
- “N dar neatsakė” alertai (kai liko 10 min).

4.2 Greiti veiksmai be gilių meniu

Ant įvykio:

- redaguoti/perkelti
- atšaukti
- trinti (su patvirtinimu ir įspėjimu, jei yra atsakymai)

4.3 Teacher Review (peržiūra)

Papildyti:

- filtrai: klasė, laikotarpis, įvykis, “neužpildė”, “vėluota”, “nenoriu atsakyti”
 - mygtukas “Eksportuoti PDF/CSV” prie įvykio.
-

5) Student pusė (tik tiek, kiek būtina mokytojai)

5.1 “Nežinau” srautas (UX)

Kai mokinys spaudžia “Nežinau”:

- atsidaro mini vedlys su 1–3 klausimais
- rodo progresą “1/3”
- pabaigoje 1 aiškus veiksmas: “Grįžti ir atsakyti”

5.2 “Nenoriu atsakyti”

- be papildomų žingsnių
 - statistikoje atskira kategorija.
-

6) Analitika (mokytojos pridėtinė vertė)

6.1 Pjūviai (MVP)

- Individidas momentu / laike
- Grupė momentu / laike
- filtrai: klasė, tema, laikotarpis
- statusai: neatsakė, nenoriu, vėluota

6.2 Grafikai (MVP)

Pagal klausimo tipą: Taip/Ne – stulpeliai; Šviesoforas – stacked; Skalė – histograma; Tekstas – TOP temos.

6.3 AI tekstų sužymėjimas į JSON

Atviriems atsakymams: topic, emotion+intensity, suggestion_type, summary, action_hint, keywords.

Svarbu MVP:

- AI kvietimas tik tekstiniams klausimams
 - saugoti AI rezultatą atskirai
 - perduoti AI tik klausimą+atsakymą (minimalus duomenų perdavimas).
-

7) Eksportai (mokytojos “oficialus dokumentas”)

7.1 CSV

- žali atsakymai + statusai + laikas + grupė
- agregatai (suminės lentelės)

7.2 PDF (A4, su privatumo žymomis ir N)

PDF šablonai:

- 1 puslapio santrauka
 - pilna ataskaita
 - individuali (jei leidžiama)
-

8) Saugumas ir privatumas (neatidėlioti)

Įdiegti nuo pradžių:

- RBAC (teacher/student)
 - audit log (peržiūrėjo/eksportavo/keitė)
 - rate limit + bruteforce apsauga
 - CSRF/Origin apsauga (jei cookie auth)
 - HTTPS (prod)
 - privatumo režimai + min N anon agregatui (pvz $N \geq 5$)
-

9) Konkretūs kodo pavyzdžiai (minimalūs)

9.1 Express RBAC middleware (backend)

```
// middleware/requireRole.js
module.exports = (...roles) => (req, res, next) => {
  if (!req.user) return res.status(401).json({ error: "Unauthenticated" });
  if (!roles.includes(req.user.role)) return res.status(403).json({ error: "Forbidden" });
  next();
};
```

Naudojimas route'ė:

```
const requireRole = require("../middleware/requireRole");

router.post("/schedules", auth, requireRole("teacher"), createSchedule);
```

9.2 Universalūs pasirinkimai klausimui (frontend render logika)

```
function UniversalChoices({ onSkip, onUnknown }) {
  return (
    <div className="mt-3 flex gap-2">
      <button className="px-3 py-2 rounded bg-gray-200" onClick={onSkip}>
```

```
        Nenoriu atsakyti
      </button>
      <button className="px-3 py-2 rounded bg-gray-200" onClick={onUnknown}>
        Nežinau
      </button>
    </div>
  );
}
```

9.3 “Nežinau” vedlio užbaigimo taisyklė (frontend)

```
setUnknownFlow({ steps, resolved: true });
navigateBackToOriginalQuestion();
```

10) Testavimo scenarijai (mokytojos pusės prioritetas)

Minimalūs “acceptance testai”:

- 1) Mokytoja sukuria įvykį (galiojimas 20 min) → mokiniai mato aktyvų → po 20 min dingsta/expire.
 - 2) Mokytoja mato progresą: kas užpildė / kas ne.
 - 3) “Nežinau” → vedlys → privalomas atsakymas į originalų (arba “Nenoriu atsakyti”).
 - 4) Review: mokytoja palieka komentarą → mokinys mato istorijoje.
 - 5) Eksportas PDF: rodo N, privatumo režimą, TOP temas (jei tekstas).
-

Kas dabar yra “pirmas realus darbas” (kad pajudėtume šiandien)

1) Išjungti mock mokytojos pusėje ir prijungti realų backend bent vienam srautui:

- mokytoja sukuria schedule
- mokinys pateikia response
- mokytoja mato progress + review sąrašą

2) Tada implementuoti universalų „Nenoriu atsakyti“ + „Nežinau“ per visus klausimus.

Jei įkelsi (ar įkopijuosi) dabartinius backend/src/routes/* ir controllers/* failus (arba bent jų sąrašą + turinį), aš šitą planą paversiu į tikslų TODO pagal failus: ką kur tiksliai įrašyti ir kokius konkrečius commit’us daryti, kad mokytojos pusė būtų pilnai MVP-ready pagal specifikaciją.

Vartotojas

sudek visa sita pokalbi i pdf fial