App Internationalization (i18n) Implementation Plan — English ⇄ Chinese

This document specifies a production-ready plan to add internationalization (i18n) to the Scrum App. It enables instant language switching between English and Chinese across the React frontend and FastAPI backend. Use this after you complete the Scrum feature add-ons roadmap.

# 1. Goals & Principles

- Instant toggle between English and Chinese (zh-CN).

- Keep user-entered content as-is; only translate UI labels/messages.

- Return stable machine-readable codes from backend; map to localized strings on the client.

- Keep dictionaries versionable, testable, and easy to extend.

# 2. Frontend (React) — i18next + react-i18next

2.1 Install

Run in frontend folder:

npm i i18next react-i18next i18next-browser-languagedetector

2.2 File layout

frontend/src/i18n/  
 en/common.json  
 zh-CN/common.json  
 index.ts  
frontend/src/components/LanguageSwitcher.tsx

2.3 Example dictionaries

File: frontend/src/i18n/en/common.json

{  
 "app": {  
 "title": "Scrum App",  
 "language": "Language",  
 "english": "English",  
 "chinese": "Chinese"  
 },  
 "nav": {  
 "backlog": "Backlog",  
 "sprints": "Sprints",  
 "board": "Board",  
 "reports": "Reports",  
 "settings": "Settings"  
 },  
 "backlog": {  
 "addItem": "Add item",  
 "title": "Title",  
 "assignee": "Assignee",  
 "points": "Points"  
 },  
 "errors": {  
 "NETWORK\_ERROR": "Network error. Please try again.",  
 "UNAUTHORIZED": "You are not signed in.",  
 "UNKNOWN": "Something went wrong."  
 }  
}

File: frontend/src/i18n/zh-CN/common.json

{  
 "app": {  
 "title": "Scrum 应用",  
 "language": "语言",  
 "english": "英文",  
 "chinese": "中文"  
 },  
 "nav": {  
 "backlog": "产品待办",  
 "sprints": "迭代",  
 "board": "看板",  
 "reports": "报表",  
 "settings": "设置"  
 },  
 "backlog": {  
 "addItem": "新增事项",  
 "title": "标题",  
 "assignee": "负责人",  
 "points": "点数"  
 },  
 "errors": {  
 "NETWORK\_ERROR": "网络错误，请重试。",  
 "UNAUTHORIZED": "您尚未登录。",  
 "UNKNOWN": "发生未知错误。"  
 }  
}

2.4 Bootstrap (frontend/src/i18n/index.ts)

import i18n from "i18next";  
import { initReactI18next } from "react-i18next";  
import LanguageDetector from "i18next-browser-languagedetector";  
  
import en from "./en/common.json";  
import zhCN from "./zh-CN/common.json";  
  
i18n  
 .use(LanguageDetector)  
 .use(initReactI18next)  
 .init({  
 resources: { en: { translation: en }, "zh-CN": { translation: zhCN } },  
 fallbackLng: "en",  
 supportedLngs: ["en", "zh-CN"],  
 interpolation: { escapeValue: false },  
 detection: { order: ["querystring", "localStorage", "navigator"], lookupQuerystring: "lang", caches: ["localStorage"] }  
 });  
  
export default i18n;

Import once in your entry file (e.g., src/main.tsx or index.tsx):

import "./i18n";

2.5 Usage in components

import { useTranslation } from "react-i18next";  
  
export default function BacklogHeader() {  
 const { t } = useTranslation();  
 return (  
 <div className="flex items-center gap-3">  
 <h1 className="text-xl font-semibold">{t("nav.backlog")}</h1>  
 <button className="btn">{t("backlog.addItem")}</button>  
 </div>  
 );  
}

2.6 Language switcher (frontend/src/components/LanguageSwitcher.tsx)

import { useTranslation } from "react-i18next";  
  
export default function LanguageSwitcher() {  
 const { i18n, t } = useTranslation();  
 const setLang = (lng: "en" | "zh-CN") => i18n.changeLanguage(lng);  
 return (  
 <div className="inline-flex gap-2 items-center">  
 <span>{t("app.language")}:</span>  
 <button onClick={() => setLang("en")} className="btn-outline">{t("app.english")}</button>  
 <button onClick={() => setLang("zh-CN")} className="btn-outline">{t("app.chinese")}</button>  
 </div>  
 );  
}

2.7 Dates, numbers, plurals

Use Intl APIs or dayjs with locale packs; ensure formats respect current i18n.language. Example: new Intl.NumberFormat(i18n.language).format(12345.67)

# 3. Backend (FastAPI) — Locale Awareness

3.1 Middleware to detect Accept-Language and expose locale

from starlette.middleware.base import BaseHTTPMiddleware  
from starlette.requests import Request  
  
def parse\_accept\_language(header: str) -> str:  
 if not header:  
 return "en"  
 return header.split(",")[0].split(";")[0].strip() or "en"  
  
class LocaleMiddleware(BaseHTTPMiddleware):  
 async def dispatch(self, request: Request, call\_next):  
 lang = parse\_accept\_language(request.headers.get("accept-language", ""))  
 request.state.locale = lang if lang in ("en", "zh-CN") else "en"  
 response = await call\_next(request)  
 response.headers["Content-Language"] = request.state.locale  
 return response

Register middleware in app/main.py

from fastapi import FastAPI  
from .middleware.i18n import LocaleMiddleware  
  
app = FastAPI()  
app.add\_middleware(LocaleMiddleware)

3.2 Return stable error codes and let frontend translate

from fastapi import HTTPException  
  
def unauthorized():  
 raise HTTPException(status\_code=401, detail={"code": "UNAUTHORIZED"})

3.3 Optional: server-side message catalogs for emails/notifications

MESSAGES = {  
 "en": {"reset\_subject": "Reset your password"},  
 "zh-CN":{"reset\_subject": "重置您的密码"}  
}  
  
def t(request, key):  
 return MESSAGES.get(getattr(request.state, "locale", "en"), MESSAGES["en"]).get(key, key)

# 4. Data Model & Content Strategy

- Do not auto-translate user data; store as entered.

- For predefined statuses/labels, store stable keys (e.g., IN\_PROGRESS) and translate in UI.

- Include lang in report/export endpoints so headers/labels are localized.

# 5. Testing Checklist

- No lang set → browser language detected; fallback to English.

- ?lang=zh-CN overrides and persists via localStorage.

- All visible strings sourced from dictionaries; no hard-coded strings remain.

- API error detail.code maps to i18n error messages.

- Date/number formatting respects locale.

- Playwright e2e runs validate both locales after reload.

# 6. Incremental Rollout Plan

- Add i18n bootstrap and wrap the top nav + one page.

- Extract strings from Backlog, Board, Sprints, Reports.

- Map API error codes to i18n and replace hard-coded toasts/labels.

- Add LanguageSwitcher in header and Settings.

- Finalize dictionaries, verify formatting, write e2e tests.