نظام إدارة مشاريع التخرج الجامعية

UPPMS — University Project Portfolio Management System

الفصل الخامس

التصميم عالي المستوى

وفق دليل كتابة أطروحة مشروع التخرج

كلية هندسة الذكاء الاصطناعي — الجامعة السورية الخاصة (SPU)

المخططات كصور منفصلة في مجلد diagrams/

المحتويات

5-1 التصميم عالي المستوى

System Block Diagram 5-1-1

Architecture Pattern 5-1-2

System Architecture Design 5-1-3 (الشرح النصي)

5-2 تصميم قاعدة البيانات

ER Diagram 5-2-1

Relational Schema 5-2-2

Constraints and Keys 5-2-3

External APIs Design 5-3

Security and Permissions Design 5-4

5-1 التصميم عالي المستوى

يهدف هذا القسم إلى توضيح كيفية تنفيذ نظام UPPMS بناءً على التحليل السابق، من خلال تحديد المكوّنات الرئيسية، نمط المعمارية المعتمد، والشرح النصي لآلية عمل النظام.

5-1-1 System Block Diagram

يوضح مخطط الكتل المكوّنات الأساسية للنظام وتفاعلها مع الخدمات الخارجية.

الشكل 5-1: System Block Diagram

ملف الصورة: `diagrams/01-system-block-diagram.png`

مكوّنات المخطط:

المكوّن	الوصف
Web Browser (React SPA)	واجهة المستخدم في المتصفح
Frontend Layer	صفحات ومكوّنات React مع MUI وإدارة التوجيه
API Layer	Laravel Controllers تستقبل طلبات REST
Business Layer	Services ومحرك الجدولة الجينية
Data Layer	Eloquent Models مع عزل الجامعات
MySQL	قاعدة البيانات الرئيسية
Google Gemini API	اقتراح أفكار المشاريع وتوليد المهام
GitHub OAuth + API	ربط الحساب ومزامنة المستودع والإصدارات

5-1-2 Architecture Pattern

يعتمد نظام UPPMS على نمط معماري مركّب يناسب تطبيقات الويب الحديثة، ويجمع بين:

1. **SPA + REST API**: الواجهة الأمامية تطبيق React منفصل، والخلفية واجهة Laravel REST تحت المسار `/api`.

2. **Layered Architecture**: الطبقات على الخادم هي Routes → Controllers → Services → Models → Database، حيث يوجد منطق الأعمال في طبقة Services.
3. **MVC داخل Laravel**: Models تمثّل البيانات وControllers تنسّق الطلب/الرد، بينما واجهة العرض الأساسية هي React وليست Blade.
4. **Multi-Tenant SaaS**: عدة جامعات على قاعدة بيانات مشتركة مع عزل منطقي عبر `university_id` و`TenantScope`.

الشكل 5-2: Architecture Layers

ملف الصورة: `diagrams/02-architecture-layers.png`

مبررات اختيار هذا النمط:

- فصل الواجهة عن المنطق يسهّل التطوير والصيانة.
- طبقة Services تسمح بإعادة استخدام القواعد بين المقترحات والمسارات والجدولة.
- REST مع Bearer Token مناسب لتطبيق React أحادي الصفحة.
- عزل الجامعات ضروري لأن النظام متعدد المستأجرين.

5-1-3 System Architecture Design (شرح نصي للمخطط)

نظام UPPMS منصة لإدارة مشاريع التخرج داخل الجامعات. يتفاعل المستخدم (طالب، مشرف، مدير جامعة، أو مدير منصة) مع واجهة React عبر المتصفح. ترسل الواجهة طلبات JSON إلى الـ API مع رمز Sanctum.

عند وصول الطلب، يمر عبر Middleware للتحقق من المصادقة، وارتباط المستخدم بجامعة، وحالة الحساب، والدور. ثم يصل إلى الـ Controller المناسب الذي يستدعي Service لتنفيذ منطق العمل (مثل إنشاء مقترح، توليد جدول، استيراد XML). يحفظ الـ Service النتائج عبر Eloquent Models في MySQL ضمن نطاق جامعة المستخدم.

الوحدات الأساسية في النظام:

- **Authentication & Users**: تسجيل، دخول، اعتماد الحسابات، الملف الشخصي.
- **Projects & Proposals**: مقترح ← موافقة مشرف ← مشروع، مع مهام وتعليقات ودعوات وإصدارات.
- **Academic Tracks**: مسارات ومراحل وتقدّم طلابي مرتبط بنتائج المناقشات.
- **Scheduling**: إعداد الغرف والتواريخ والتوافر، ثم توليد جدول بالخوارزمية الجينية (وضع فردي أو لجان)، ثم الاعتماد والإشعارات.

- **Committees**: إدارة لجان المناقشة وربطها بالجلسات.
- **XML Import**: استيراد المستخدمين المصرّح لهم ومطابقتهم عند التسجيل.
- **AI Services**: Gemini لاقتراح الأفكار وتوليد المهام.
- **Platform Admin**: إدارة الجامعات ومديريها عبر `super_admin`.

الخدمات الخارجية المستخدمة هي Gemini لتوليد المحتوى، وGitHub لربط الحساب ومستودع المشروع. الإشعارات داخلية في قاعدة البيانات وتظهر في واجهة المستخدم للمشرفين والطلاب بعد اعتماد الجدول.

5-2 تصميم قاعدة البيانات

5-2-1 ER Diagram

يوضح مخطط الكيانات-العلاقات الكيانات الأساسية وعلاقاتها في جوهر النظام (الجامعات، المستخدمون، المشاريع، المقترحات، المسارات، الجداول، الجلسات، اللجان، XML).

الشكل 5-3: ER Diagram Overview

ملف الصورة: `diagrams/03-erd-overview.png`

أهم العلاقات:

- الجامعة تضم المستخدمين والمشاريع والمسارات والجداول واللجان.
- المقترح المعتمد ينشئ مشروعاً (`project_proposals` → `projects`).
- المسار يحتوي مراحل هرمية؛ تقدّم الطالب يرتبط بمرحلة المسار.
- الجدول المعتمد يُنشئ جلسات مناقشة؛ الجلسة قد ترتبط بلجنة.
- XML يحدد المستخدمين المصرّح لهم قبل/عند التسجيل.

5-2-2 Relational Schema

الجداول الرئيسية في النظام:

الجدول	الغرض
universities	المستأجر (الجامعة)
roles / users	الأدوار والمستخدمون
projects / tasks / comments	المشاريع والتعاون
project_proposals	مقترحات المشاريع
tracks / track_stages / student_progress	المسارات والتقدّم
academic_stages_config / available_rooms / doctor_availabilities	إعداد الجدولة
approved_schedules / defense_sessions	الجداول والجلسات
committees / committee_members / committee_assignments	اللجان
xml_authorized_users / xml_import_logs	استيراد XML
notifications	الإشعارات
personal_access_tokens	رموز Sanctum

التفاصيل العمودية الكاملة للحقول والأنواع موجودة في الملف النصي المساند: `schemas/relational-schema.md`

ملخص حقول الجداول الأهم

users: id, name, email (unique), password, role_id, university_id, track_id, student_number, status (pending/active/rejected/graduated), github_token.

projects: id, university_id, owner_id, supervisor_id, proposal_id, title, description, status, github_repo_url.

project_proposals: id, university_id, student_id, supervisor_id, track_stage_id, title, description, status (pending/approved/rejected), resubmission_count.

tracks / **track_stages**: المسار ومراحله الهرمية (parent_id, stage_kind, is_decisive, academic_stage_id).

student_progress: student_id, track_id, track_stage_id, status (in_progress/passed/failed/incomplete).

approved_schedules / **defense_sessions**: الجدول المعتمد وجلسات المناقشة (غرفة، وقت، لجنة، حالة).

committees: لجان المناقشة مع أعضاء chair/member.

xml_authorized_users: university_number, email, user_type (student/supervisor), is_used, registered_user_id.

Constraints and Keys 5-2-3

Primary Keys: جميع الجداول تستخدم `id` كمفتاح أساسي.

Foreign Keys: ربط المستخدم بالدور والجامعة، المشروع بالمقترح والمالك، المراحل بالمسار، الجلسات بالجدول والمشروع واللجنة، وسجلات XML بالجامعة.

Unique Constraints:

- users.email
- (university_id, student_number) على users
- (university_id, name) على tracks و committees
- (student_id, track_id, track_stage_id) على student_progress
- (university_id, university_number, email, user_type) على xml_authorized_users

قواعد أعمال مهمة:

- لا أكثر من مقترح pending نشط لنفس الطالب.
- عزل المستأجر عبر TenantScope + university_id.
- مطابقة XML عند التسجيل تفعّل الحساب وتربط السجل.
- جدول معتمد واحد نشط لكل مرحلة أكاديمية.
- لا يوجد SoftDeletes؛ تُستخدم حالات مثل is_active / cancelled / voided.

التفاصيل الكاملة في: `schemas/constraints-and-keys.md`

5-3 External APIs Design

يستخدم النظام واجهات خارجية، ويعرّض أيضاً REST API داخلياً لواجهة React.

5-3-1 Google Gemini API

البند	التفاصيل
الاستخدام	اقتراح أفكار مشاريع + توليد مهام
Method / Path	POST …/models/{model}:generateContent
المصادقة	x-goog-api-key (GEMINI_API_KEY)
النموذج	gemini-2.5-flash (افتراضي)
خدمات UPPMS	AIIdeationService, AITaskService
مسارات النظام	POST /api/ai/suggest-projects — POST /api/projects/{project}/generate-tasks
الأخطاء	مفتاح غير صالح، حد المعدل، JSON غير صالح

5-3-2 GitHub OAuth + REST API

البند	التفاصيل
الاستخدام	ربط الحساب، مزامنة commits، دفع الإصدارات
OAuth	GET /api/auth/github/redirect و /callback (Socialite, scope: repo)
API	api.github.com عبر GithubService و/أو users.github_token
الأخطاء	رفض التفويض، توكن منتهٍ، مستودع غير موجود

5-3-3 Internal REST API (ملخص)

المجموعة	أمثلة	Methods
Auth	register, /login, /logout/	POST
Users / XML	*/users, /admin/xml-import/	GET/POST
Projects / Tasks	*/projects, /task/	CRUD
Proposals / Tracks	proposals, /tracks/	GET/POST/PUT
Schedules	schedules/generate, /approve/	POST/GET
Committees	committees, assign-committee/	GET/POST/PUT
AI / Platform	ai/*, /admin/universities/	POST/GET

المصادقة الداخلية: Bearer Token (Sanctum). أكواد شائعة: 200/201 نجاح، 401 غير مصادق، 403 ممنوع، 404 غير موجود، 422 تحقق فاشل، 429 حد معدل.

5-4 Security and Permissions Design

5-4-1 المصادقة

العنصر	التنفيذ
الآلية	Laravel Sanctum — Personal Access Tokens
التخزين في العميل	localStorage + Authorization: Bearer
تسجيل الخروج	حذف التوكن الحالي
ملاحظة	ليست JWT؛ التوكنات في personal_access_tokens

5-4-2 الأدوار

الدور	الاسم	الصلاحية العامة
Student	student	مقترحات، مشاريع، مهام، تقدّم، دعوات
Supervisor	supervisor	مراجعة مقترحات، إشراف، توافر، مناقشات
University Admin	admin	مستخدمون، XML، مسارات، جدولة، لجان
Super Admin	super_admin	إدارة الجامعات والمنصة

5-4-3 Middleware ومستويات الوصول

- `auth:sanctum` — يتطلب توكن صالح.
- `EnsureUserHasUniversity` — المستخدم مرتبط بجامعة (استثناء super_admin).
- `user.status / CheckUserStatus` — يسمح فقط بـ active أو graduated.
- `...:role` — تقييد المسارات حسب الدور.
- `TenantScope` — عزل بيانات الجامعة تلقائياً.

5-4-4 دورة التسجيل والاعتماد

1. مدير الجامعة يستورد XML للمستخدمين المصرّح لهم.
2. عند التسجيل: إن وُجد تطابق XML يصبح الحساب active، وإلا pending.
3. مدير الجامعة يوافق أو يرفض الحسابات المعلّقة.
4. حسابات pending/rejected لا تدخل الـ API المحمي.

5-4-5 عزل البيانات (Multi-Tenancy)

كل سجل تشغيلي تقريباً يحمل university_id. السمة BelongsToUniversity مع TenantScope تمنع رؤية بيانات جامعة أخرى. أما super_admin فيتجاوز النطاق لإدارة المنصة عبر الجامعات.