Contents

1 Advanced Programming Capstone Description

The system enables multidisciplinary student teams from Computer Science/Software Engineering and Engineering to collaborate on real-world projects carried out at government facilities. It provides a unified platform to manage programs, facilities, services, equipment, projects, participants, and outcomes, ensuring that projects are well-organized, properly resourced, and aligned with Uganda's NDPIII, Digital Transformation Roadmap (2023–2028), and 4IR Strategy.

Each project team should consist of **five students**, ensuring diversity of skills and balanced workloads. For effective accountability and evaluation, teams are required to use project management tools such as *ClickUp* or *GitHub Projects* to track weekly contributions. This makes it possible to monitor individual responsibilities, visualize task ownership, and provide transparent evidence of participation when grading or assessing progress.

1.1 Core Entities and Their Fields

1.1.1 Program

Represents the collaboration umbrella under which projects run.

Table 1.1: *Program Fields*

Field	Description
ProgramId	Unique identifier for the program
Name	Program name
Description	Overview of the program's purpose
NationalAlignment	Link to NDPIII, Roadmap, or 4IR goals
FocusAreas	Domains such as IoT, automation, renewable energy
Phases	Cross-Skilling, Collaboration, Technical Skills, Prototyping, Commercialization

Relationships: One Program has many Projects.

1.1.2 Facility

Represents a government place where projects are executed.

Table 1.2: *Facility Fields*

Field	Description
FacilityId	Unique identifier for the facility
Name	Facility name
Location	Geographic location
Description	Overview of the facility's function
PartnerOrganization	Partner such as UniPod, UIRI, Lwera
FacilityType	Lab, Workshop, Testing Center
Capabilities	CNC, PCB fabrication, materials testing

Relationships: Facility → many Services, many Equipment, many Projects.

1.1.3 Service

Represents the types of work a facility can perform.

Table 1.3: *Service Fields*

Field	Description
ServiceId	Unique identifier for the service
FacilityId	Identifier of the facility offering this service
Name	Service name
Description	What the service does
Category	Machining, Testing, Training
SkillType	Hardware, Software, Integration

Relationships: A Facility has many Services.

1.1.4 Equipment

Represents machinery/tools available at a facility.

Table 1.4: *Equipment Fields*

Field	Description
EquipmentId	Unique identifier for the equipment
FacilityId	Identifier of the facility owning the equipment
Name	Equipment name
Capabilities	Functions it can perform
Description	Overview of equipment purpose
InventoryCode	Tracking code
UsageDomain	Electronics, Mechanical, IoT
SupportPhase	Training, Prototyping, Testing, Commercialization

Relationships: A Facility has many Equipment.

1.1.5 Project

Represents work carried out by student teams.

Table 1.5: *Project Fields*

Field	Description
ProjectId	Unique identifier for the project
ProgramId	Identifier of the program
FacilityId	Identifier of the facility hosting the project
Title	Project title
NatureOfProject	Research, prototype, or applied work
Description	Project overview
InnovationFocus	IoT devices, smart home, renewable energy
PrototypeStage	Concept, Prototype, MVP, Market Launch
TestingRequirements	Compliance and performance checks
CommercializationPlan	Path to market adoption

Relationships: Each Project belongs to one Program and one Facility; has many Participants and Outcomes.

1.1.6 Participant

Represents people contributing to projects.

Table 1.6: *Participant Fields*

Field	Description
ParticipantId	Unique identifier for the participant
FullName	Participant name
Email	Contact email
Affiliation	CS, SE, Engineering, or Other
Specialization	Software, hardware, business
CrossSkillTrained	Indicates cross-skill training (true/false)
Institution	SCIT, CEDAT, UniPod, UIRI, Lwera

Relationships: Many-to-many with Projects through ProjectParticipant.

1.1.7 ProjectParticipant

Join table linking participants to projects.

Table 1.7: *ProjectParticipant Fields*

Field	Description
ProjectId	Identifier of the project
ParticipantId	Identifier of the participant
RoleOnProject	Student, Lecturer, Contributor
SkillRole	Developer, Engineer, Designer, Business Lead

Relationships: Many ProjectParticipants belong to one Project; many to one Participant.

1.1.8 Outcome

Represents deliverables from a project.

Table 1.8: *Outcome Fields*

Field	Description
OutcomeId	Unique identifier for the outcome
ProjectId	Identifier of the related project
Title	Outcome title
Description	Description of the outcome
ArtifactLink	Link to the deliverable artifact
OutcomeType	CAD, PCB, Prototype, Report, Business Plan
QualityCertification	Compliance or test results (e.g., UIRI certification)
CommercializationStatus	Demoed, Market Linked, Launched

Relationships: Each Project has many Outcomes.

1.2 System UseCases

1.2.1 Register Facilities

The system must allow administrators to register and manage government facilities, such as laboratories, workshops, maker spaces, and testing centers.

Key Use Cases:

- List all registered facilities.
- View details of a given facility.
- Create a new facility record with name, location, description, partner organization, type, and capabilities.
- Edit the details of an existing facility.
- Delete a facility record (with safeguards if linked to projects).
- Search and filter facilities by type, partner, or capability.

> **Note**
>
> Facilities act as anchors in the ecosystem: every project, service, and equipment record is linked to one facility.

1.2.2 Register Services

Facilities provide services such as CNC machining, welding, PCB fabrication, or materials testing. The system must manage these services and associate them with the appropriate facility.

Key Use Cases:

- List all services across facilities.

- List services offered by a given facility.
- Create a new service under a facility (with category and skill type).
- Edit a service record.
- Delete a service record.
- Search for services by category (e.g., machining, testing).

> **Tip**
>
> Well-categorized services help students and supervisors rapidly match project requirements to available capabilities.

1.2.3 Register Equipment

The system records machines and tools at facilities, documenting their capabilities and usage domains.

Key Use Cases:

- List all equipment.
- List all equipment at a specific facility.
- Create a new equipment record with metadata (name, description, capabilities, inventory code, usage domain, support phase).
- Edit equipment details.
- Delete equipment (with constraints if tied to active projects).
- Search equipment by capability or usage domain.

1.2.4 Create and Manage Projects

Projects represent student and lecturer collaboration, and must be created, tracked, and managed within the system.

Key Use Cases:

- List all projects in the system.
- List projects under a given facility.
- List projects under a given program.
- View project details.
- Create a project (assign to a facility and program).
- Edit project details (title, description, innovation focus, prototype stage, commercialization plan).
- Delete a project.
- Assign participants to a project.

> **Note**
>
> Each project is bound to exactly one facility and one program, simplifying approval workflows and scheduling.

1.2.5 Manage Participants

Participants include students, lecturers, and collaborators. Their involvement must be tracked and linked to projects.

Key Use Cases:

- List all participants.
- View participant profiles.
- Create a participant record (with affiliation, specialization, institution, cross-skilling status).
- Edit participant details.
- Delete a participant.
- Assign or remove a participant from a project.
- List all projects a participant is engaged in.

1.2.6 Capture Project Outcomes

Project outcomes include prototypes, CAD files, PCB designs, reports, and business plans. These must be attached directly to projects with metadata.

Key Use Cases:

- List outcomes for a given project.
- Upload a new outcome (attach artifacts, specify type, certification status, commercialization status).
- Edit outcome details.
- Delete an outcome.
- View/download linked artifacts.

> **Tip**
>
> Storing outcomes directly under projects ensures auditability and simplifies grading and commercialization tracking.

1.2.7 Organize Work Under a Program

Programs provide the overarching structure, aligning student work with national development strategies.

Key Use Cases:

- List all programs.
- View program details (name, description, alignment, phases, focus areas).
- Create a new program.
- Edit program details.
- Delete a program.
- List all projects under a program.

1.3 Development Timeline and Deliverables

This project spans two months. The first month delivers the full web application and automated tests. The second month delivers a cross-platform mobile application (iOS and Android) with strict feature parity to the web app.

1.3.1 Month 1 (Weeks 1–4): Web Application

Weeks 1–2: Implement Core Use Cases (Web)

- **Facilities**: list, view, create, edit, delete; search/filter.
- **Services**: list (global/by facility), view, create, edit, delete.
- **Equipment**: list (global/by facility), view, create, edit, delete; search by capability/domain.
- **Projects**: list (global/by facility/by program), view, create, edit, delete; assign/remove participants.
- **Participants**: list, view, create, edit, delete; list projects per participant.
- **Outcomes**: list by project, view/download, create (upload/link), edit, delete.

> **Note**
>
> By the end of Week 2 the web app provides complete CRUD coverage for all entities and the linking flows (e.g., Project ↔ Facility, Project ↔ Participants, Project ↔ Outcomes).

Weeks 3–4: Unit and Integration Tests (Web)

- **Unit tests**: entity creation rules, required fields, referential integrity (e.g., Project must belong to exactly one Facility and one Program).
- **Integration tests**: end-to-end flows (create Project → assign Participants → add Outcomes), deletion constraints (e.g., prevent deleting Facility with dependent records until handled).
- **Test data builders**: fixtures for Facilities, Services, Equipment, Projects, Participants, Outcomes.
- **CI runs**: all tests on every push/PR with coverage report.

> **Tip**
>
> The Month 1 API surface is *frozen* at the end of Week 4 to guarantee a stable contract for the mobile app in Month 2.

1.3.2 Month 2 (Weeks 5–8): Mobile Application (iOS and Android)

The second month delivers a cross-platform mobile app that supports the *same features* as the web app: the same entities, the same CRUD operations, and the same linking flows. The mobile app consumes the Month 1 API without introducing new domain features.

Week 5: Foundations & Core Navigation

- Mobile project scaffolding (iOS + Android), app shell, theming, typography, icon set.
- Authentication wiring (same identity and roles as web).
- Read-only **lists & details**: Programs, Facilities, Services (by facility), Equipment (by facility), Projects (global/by facility/by program), Participants (global), Outcomes (by project).
- Shared API models and serializers aligned 1:1 with web DTOs.

Week 6: CRUD for Primary Entities

- **Projects**: create, edit, delete; assign/remove Participants.
- **Participants**: create, edit, delete; view projects for a participant.
- **Outcomes**: create (file/photo upload or link), edit, delete; view/download artifacts.
- Validation messages and error handling aligned with web rules.

Week 7: CRUD for Catalog Entities & Parity Gaps

- **Facilities**: create, edit, delete; search/filter.
- **Services**: create, edit, delete (scoped to facility).
- **Equipment**: create, edit, delete; search by capability/usage domain.
- Close any parity gaps (edge-case filters, paging, empty states).

Week 8: Hardening, Accessibility, Packaging

- Performance passes on large lists (paging/infinite scroll).
- Accessibility checks (labels, contrast, dynamic type).
- End-to-end sanity tests on device (happy paths for all use cases).
- App packaging for internal distribution (iOS TestFlight / Android internal track).

> **Note**
>
> The mobile app must not introduce new domain features. It must achieve *strict feature parity* with the web app: identical entities, CRUD operations, and linking flows.

1.3.3 Feature Parity Matrix (Web & Mobile)

Table 1.9: *Parity of Supported Operations Across Platforms*

Entity	List	View	Create	Edit	Delete	Linking
Program	Web/Mobile	Web/Mobile	Web/Mobile	Web/Mobile	Web/Mobile	Projects by Program
Facility	Web/Mobile	Web/Mobile	Web/Mobile	Web/Mobile	Web/Mobile	Services/Equipment/Projects
Service	Web/Mobile	Web/Mobile	Web/Mobile	Web/Mobile	Web/Mobile	By Facility
Equipment	Web/Mobile	Web/Mobile	Web/Mobile	Web/Mobile	Web/Mobile	By Facility
Project	Web/Mobile	Web/Mobile	Web/Mobile	Web/Mobile	Web/Mobile	Facility, Program, Participants, Outc
Participant	Web/Mobile	Web/Mobile	Web/Mobile	Web/Mobile	Web/Mobile	Projects by Participant
Outcome	Web/Mobile	Web/Mobile	Web/Mobile	Web/Mobile	Web/Mobile	By Project

> **Tip**
>
> A shared API contract (models, endpoints, validation) guarantees that the mobile client remains a thin layer. Any changes to behavior must be implemented server-side first and reflected in both clients.