



Senior BBI Self-Study & Proposal Roadmap

Choosing your stack, thinking like a founder, and the semester plan

BBI 6416 · Senior BBI · Semester I, 2026–2027 · September 2026 – January 2027

1. The situation you are in

This semester you choose a project, get it approved, and write and defend Chapters 1–3. Next semester you build it. The system you propose in January is the system you must deliver in June 2027, in the technology you name in Chapter 3 — so the stack decision is made now, not in February. Be honest with yourself about the table below.

What the BBI curriculum gave you	Where	What it did NOT give you
Programming fundamentals in C, then object-oriented programming in C#	Introduction to Computer Programming (C); Application Programming (C#)	A web framework, an API, or how a real multi-user product is structured (MVC, authentication, deployment)
HTML/CSS page design; web programming and e-commerce concepts	Web Design & Publishing; Internet-Based Programming & e-Commerce (this semester)	A modern front end (React, component-based UI), responsive design for phones, payment and SMS integrations
Databases: Access, SQL, MongoDB	DBMS I–III (MongoDB this semester)	Connecting an application to a database through code (ORMs, migrations), data modelling for many customers
Python basics	Introduction to Python (this semester)	A Python web framework (Django/Flask) or data/report automation
Data structures, networks, AI & software agents, multimedia	Junior and senior courses	Testing your own code, version control with Git, hosting a live application
Business: entrepreneurship, marketing, accounting, BIS, organisational behaviour, IT project management, research methods	Business and management courses	— (you are well covered here: use them for the product, market, business case and methodology)

The conclusion: you already understand business better than any other ICT programme. What you lack is a modern building stack. This project is your chance to fix that — and to leave university with something you can sell or show an employer.

2. Why plain PHP is not enough

Of the 54 BBI proposals submitted in January 2026, 50 used PHP and 42 used MySQL; 48 used plain HTML/CSS as the front end. PHP is not a bad language — much of the web runs on it —

but plain PHP pages with a MySQL table behind each form is what every other graduate in the room can also show. It does not teach you how modern products are built, and it will not make your CV or your project stand out.

What employers and startups in the region actually ask for

- A component-based front end (React / Next.js, or Flutter for mobile) and a clean API behind it.
- A framework, not hand-written pages: Express/NestJS, Django, ASP.NET Core or Laravel — routing, authentication, migrations, validation.
- Git and GitHub, a deployed application with a real URL, and code someone else can read.
- Integrations that business software needs: SMS, e-mail, PDF invoices, mobile money workflows, dashboards and reports.

Your final-year project is the single biggest item in your portfolio. Build it in the stack you want to be hired for.

3. Choose one stack — and commit to it

Explore in September, decide by Week 2 (it goes in your concept note), then stay on it. Changing stacks in February is the most common way projects die.

Track	Stack	Builds on	Best for	Learning curve	Where it leads
A — JavaScript full-stack	React or Next.js + Node.js (Express/NestJS) + PostgreSQL or MongoDB; Tailwind CSS	Internet-Based Programming; DBMS III (MongoDB); HTML/CSS	SaaS platforms, marketplaces, booking and ordering, dashboards	Medium-high: new ecosystem, but one language front and back	The most in-demand web skill set; startups, remote work
B — Python web	Django (or Flask) + PostgreSQL/MySQL + HTML/Tailwind or a React front end	Introduction to Python; DBMS	Business platforms with heavy reporting, admin panels, data and automation	Medium: Python is readable; Django gives you admin, auth and ORM for free	Web development, data analysis, automation, AI integration
C — C# / .NET	ASP.NET Core (MVC or Web API) + SQL Server + Razor or React; Blazor	Application Programming (C#); SQL	Enterprise-style systems for banks, telecoms, NGOs, government	Medium: you know C#; the web part is new	Corporate and NGO IT departments, enterprise software
D — Mobile	Flutter (Dart) + Firebase/Supabase, or Flutter + your own API (Track A/B)	C/C# fundamentals	Customer-facing apps: ordering, loyalty, field sales, bookings	High: new language, mobile lifecycle, plus a back end	Mobile development — high demand, fewer local developers
E — PHP, done properly (minimum)	Laravel + MySQL + Blade/Livewire or Vue; Tailwind	Web Design; server-side PHP if you took it	Web systems that must run on cheap shared hosting	Low-medium if you know PHP	Freelance web work; acceptable only with the framework

Recommendation

If you are unsure, take **Track A (JavaScript full-stack)** or **Track B (Python/Django)**. Both reuse what you are learning this semester, both are what employers ask for, and both can power a product

that many businesses use. Take **Track C** if you are strong in C# and want a corporate career; **Track D** only if you are ready for the most work. **Track E** is the minimum: plain PHP pages will be sent back at the concept-note stage.

4. Think like a founder

You have taken Entrepreneurship, Marketing and Accounting. Your proposal should read like the start of a business, not a class exercise. The case organisation is your **first customer (pilot)**; the product is built for **all organisations like it**.

Question	Where it appears in your proposal
Which customer segment has this problem, and how many of them are there?	Concept note §6; Chapter 1 — Significance (1.4)
What do they use today, and what does it cost them (time, money, errors, lost customers)?	Chapter 1 — Problem Statement (1.2); Chapter 2 — Current System (2.2)
What do competing products do, and why is yours better or cheaper for this market?	Chapter 2 — Reviews and comparison table (2.3–2.8)
How will it make money? (subscription, licence, commission, freemium)	Concept note §6; Chapter 1 — Significance; Appendix B — Budget
What will the pilot prove, and how will you measure it?	Chapter 1 — Objectives (1.3); Chapter 3 — Testing and changeover (3.5–3.6)
What is version 2?	Chapter 1 — Scope (1.5): excluded now, planned later

Product design choices that make a project sellable:

- **More than one business can use it** — each shop, school or clinic has its own account and data (multi-tenant).
- **Roles** — owner, manager, staff, customer — each seeing only what they need.
- **Outputs that owners pay for** — daily/weekly reports, alerts, receipts and invoices, reminders, dashboards.
- **Works on a phone** — most of your users will never open a desktop browser.
- **Fits Somaliland** — mobile money workflows, SMS/WhatsApp, Somali and English, low bandwidth.

The strongest projects can go further than the defense: into startup competitions such as the annual Somaliland Tech Expo, or into a real business with your pilot customer.

5. The semester roadmap — September 2026 to January 2027

Plan on 6–8 hours per week of self-study alongside the proposal. Each row lists the learning milestone and the proposal milestone that must be true by the end of that month.

Month	Learning milestone (self-study)	Proposal milestone (course deliverable)
Sept 2026	Explore two tracks for a week each (one tutorial, one tiny page). Choose one. Install the tools; create a GitHub account; learn basic Git.	Concept note submitted (Task 1) and approved.

Month	Learning milestone (self-study)	Proposal milestone (course deliverable)
Oct 2026	Language and framework basics: routes, templates/components, forms, a first page that saves to a database.	Chapter 1 written and reviewed (Task 2). Fieldwork at the case begins: observation and interviews.
Nov 2026	Authentication (login, roles), database from code (ORM/models, migrations), CRUD for one real entity from your case.	Chapter 2 written (Tasks 3–4): 8–12 sources read, 5–7 systems reviewed, comparison table.
Dec 2026	A small practice app deployed online with a real URL (free tier). One integration you will need: PDF, charts, SMS or e-mail.	Chapter 3 written (Tasks 5–6): DFD/ERD, architecture diagram, wireframes, test plan.
Jan 2027	Pause learning; rehearse the defense. Show your practice app if asked about your stack.	Proposal Book compiled (Task 7) and defended.
Feb–June 2027	Next semester (BBI 6426): build, test and pilot the product.	Final project defense — first week of June 2027.

5.1 Free learning resources that are good enough

Need	Use
Git and GitHub	GitHub Docs “Hello World” and GitHub Skills; freeCodeCamp Git course.
JavaScript, React, Node.js	MDN Web Docs (JavaScript); react.dev official tutorial; Next.js Learn course; The Odin Project (Full Stack JavaScript path); freeCodeCamp.
Python and Django/Flask	Official Python Tutorial; CS50P (Harvard, free); Django official tutorial (parts 1–8); Django Girls tutorial; Flask Mega-Tutorial.
C# and ASP.NET Core	Microsoft Learn — “Create web apps and services with ASP.NET Core”; .NET official tutorials.
Flutter	Flutter official codelabs; Flutter documentation “Get started”.
Laravel	Laravel Bootcamp (official); Laracasts “Laravel From Scratch” (free series).
Databases	PostgreSQL tutorial (postgresqtutorial.com); MongoDB University (free courses); your framework’s ORM docs.
UI and wireframes	Figma (free education plan); draw.io / diagrams.net for DFD, ERD and architecture diagrams; Tailwind CSS docs.
Deployment	Vercel, Render, Railway, PythonAnywhere free tiers; Let’s Encrypt for HTTPS.
Business side	Lean Canvas / Business Model Canvas templates; Y Combinator Startup School (free videos).

6. What last year’s BBI cohort did — and why it matters to you

Fifty-four Senior BBI proposals were submitted in January 2026: 49 web applications, 3 mobile and 2 desktop. Back ends: PHP 50, Python 2, C#/.NET 1, Java 1. Databases: MySQL 42, SQL Server 7, MongoDB 4, PostgreSQL 1.

The same domains appeared again and again:

- Student registration, information, attendance and performance systems — seven proposals.
- Shop, store, supermarket, stationery and electronics stock systems — seven.
- Fee and college-payment systems — four; library — three; real estate and rental — three; hotel booking — two.
- Pharmacy, hospital registration, patient registration and blood bank records — four.

Lesson: examiners have seen these many times. You may work in a familiar domain only if the product, the customer and the business model are clearly different — not a new case name on the same record-keeping forms. Previous students’ proposals are not distributed; do not ask seniors for theirs.

7. Ideas to avoid (they get sent back)

Red flags

- **Record book only.** Forms that add, edit, delete and list records, with no business result, report or decision support.
- **Plain PHP pages** with no framework and nothing new learned.
- **One-organisation thinking** where the product could obviously serve many — design it for the market, pilot it at the case.
- **Integrations you cannot get.** Live Zaad/e-Dahab or bank APIs need corporate contracts; design around them (payment reference entry, receipts, reconciliation).
- **AI-first projects without data.** “AI sales predictor” with no sales history is a proposal that cannot be built. Add AI as a feature only if you have the data.
- **Whole-ERP scope.** Three to five modules is a project; fifteen is a company.
- **Copies.** A classmate’s idea, a previous student’s project, or a template downloaded from the internet.

8. Rules of engagement

1. You need a real case you can reach. A permission letter from the organisation goes in Appendix D of the Proposal Book.
2. Personal data collected at the case (customer names, phone numbers, balances) is minimised, kept private, used only for the project, and deleted or anonymised after the viva.
3. Interviews and questionnaires are voluntary; tell participants what the study is for.
4. Keep your work in a private Git repository from the day you start learning; supervisors will ask to see commit history as evidence of your own work.
5. Every figure, statistic and system you cite must be real and retrievable. Fabricated sources fail the course.

Effective: 24 September 2026.