SEIP v2 Curriculum Update

High-Level Overview for Internal Circulation

1. Purpose of this document

This document provides a high-level overview of the new version of the Software Engineering Intensive Program, SEIP v2. It summarizes the main curriculum structure, planning expectations, pedagogical updates, content changes, and major improvements introduced across the three trimesters.

The goal of SEIP v2 is to provide a more progressive, modern, and integrated learning path for students, while preserving the core engineering foundations of the program. The new version strengthens early onboarding, introduces artificial intelligence as a practical development tool, updates the Python and SQL learning paths, expands software design and backend foundations, and gives the final trimester a stronger infrastructure and DevOps orientation.

This document does not describe every individual project in detail. Instead, it groups projects by content area and learning purpose in order to provide a clear view of the overall curriculum direction, implementation model, and educational rationale.

---

2. Current status of this version

This first version of SEIP v2 is considered stable in terms of conceptual direction, learning objectives, content coverage, and pedagogical approach. The main learning progression, trimester-level outcomes, and overall educational vision are already defined.

However, some projects and content blocks are still being rewritten, updated, or refined. This is especially the case for the frontend block, including HTML, CSS, and JavaScript, which is expected to be reviewed and aligned with the new curriculum direction.

It is also important to note that the current curriculum organization and division into specific tracks or curriculum units may evolve in the future. These structural decisions are partially influenced by the capabilities and limitations of the current LMS. While the way content is grouped or presented may change over time, the underlying content, learning objectives, and overall approach are expected to remain stable.

This version should therefore be considered a stable reference for planning and communication purposes, while understanding that some project-level details, sequencing decisions, and curriculum-unit organization may continue to evolve before or during implementation.

---

3. Curriculum structure

SEIP v2 is organized into three trimesters. Each trimester combines a main technical track with complementary modules, integration projects, AI-related content, and portfolio or capstone work.

Trimester 1 — Foundations, C Programming, Tools, and AI Introduction

Trimester 1 introduces students to computational thinking, programming logic, the Holberton learning environment, development tools, Linux, Git and GitHub workflows, shell usage, containers, C programming, debugging, memory management, secure coding practices, data structures, algorithmic foundations, and introductory AI concepts.

Curriculum links:

- Holberton Tools: https://intranet.hbtn.io/curriculums/657
- Main Track: https://intranet.hbtn.io/curriculums/774
- AI Introduction: https://intranet.hbtn.io/curriculums/658
- Advanced Topics: https://intranet.hbtn.io/curriculums/660
- Integration Projects: https://intranet.hbtn.io/curriculums/659

Trimester 2 — Python, Algorithms, Software Design, Backend, Frontend, AI for Developers, and Capstone

Trimester 2 builds on the foundations from T1 and transitions students into Python, object-oriented programming, software design, UML, design patterns, databases, SQL, ORM, APIs, asynchronous programming, frontend foundations, algorithms, AI-assisted development workflows, real-time communication, agents, MCP, and a trimester capstone project.

Curriculum links:

- Main Track: https://intranet.hbtn.io/curriculums/740
- AI 4 Devs: https://intranet.hbtn.io/curriculums/743
- PLD Activities: https://intranet.hbtn.io/curriculums/751
- Algorithms: https://intranet.hbtn.io/curriculums/752
- Agents and Capstone: https://intranet.hbtn.io/curriculums/772

Trimester 3 — Infrastructure, DevOps, Cloud, Security, and Portfolio

Trimester 3 is fully oriented toward infrastructure, DevOps, production delivery, cloud deployment, CI/CD, observability, security, incident response, and professional portfolio development.

Curriculum links:

- Main Track: https://intranet.hbtn.io/curriculums/769
- Portfolio: https://intranet.hbtn.io/curriculums/427

---

4. High-level trimester planning

The following planning is intended as a reference model. Specific timing may be adapted depending on local calendars, holidays, cohort needs, staff availability, and operational constraints.

Trimester 1 planning

Holberton Tools starts in week 1 and is expected to span approximately three weeks, although this may be adapted depending on the onboarding strategy. This block serves as an introduction to the Holberton learning environment, platform, and essential tools that students will use throughout the program.

The Main Track also starts in week 1 and spans the full trimester. It includes the progressive introduction to computational thinking, Scratch, the transition to text-based programming, computer systems, Git and GitHub, shell, containers, C programming, debugging, memory, secure coding, data structures, sorting, Big O, and systems-level programming foundations.

AI Introduction is planned for weeks 3, 4, and 5. This module introduces students to responsible AI use, prompting basics, and large language models.

Advanced Topics are planned between weeks 6 and 10. This block includes projects such as recursion and file I/O that were removed from the Main Track because they are not considered strictly necessary for achieving the core curriculum outcomes. However, these topics remain valuable for strengthening a student's understanding of fundamental operating system concepts, including process execution, memory usage, and interaction with the filesystem. For this reason, they are offered as advanced or optional projects and do not impact the Main Track score.

Integration Projects are distributed throughout the trimester depending on project length and the desired evaluation moments. These projects are used to validate that students can integrate concepts into complete programs. In addition to major integration projects such as printf and Simple Shell, a new Simple Calculator project has been introduced early in the trimester to provide an earlier validation point and help students demonstrate their ability to combine fundamental programming concepts into a complete application.

Trimester 2 planning

The Main Track starts in week 1 and spans the full trimester. It focuses on Python, object-oriented programming, software design, SQL, databases, ORM, APIs, asynchronous programming, frontend foundations, and backend-oriented engineering foundations.

Algorithms also start in week 1 and span the full trimester. This track reinforces problem-solving, algorithmic reasoning, data manipulation, recursion, complexity awareness, and preparation for technical interviews.

AI 4 Devs starts in week 6 and spans six weeks, with one project per week. This module focuses on practical AI usage in developer workflows.

PLD Activities are scheduled at different moments during the trimester depending on the project. These are one-day, in-campus, group-based activities designed to promote collaboration, research, prototyping, and presentation skills.

Agents and Capstone starts in week 10 and spans four weeks. The first two weeks cover WebSockets, AI Agents, and MCP. The final two weeks are dedicated to the HBntory capstone project. During the capstone period, no other projects should be assigned.

Trimester 3 planning

The Main Track starts in week 1 and spans approximately two months. It is entirely focused on infrastructure, DevOps, cloud, CI/CD, security, observability, incident response, and production delivery.

Portfolio starts in week 1 and spans the full trimester. During the final month of the trimester, students should only work on portfolio-related projects.

---

5. Pedagogical tools and evaluation improvements

SEIP v2 introduces new tools and mechanisms to support a more flexible and scalable evaluation model.

AI Checker

The AI Checker allows projects to evaluate student input based on a prompt. This enables evaluation of responses, explanations, design decisions, reasoning, and other types of submissions that are difficult to validate with traditional automated checkers.

This functionality is being introduced gradually across the curriculum, primarily replacing manual reviews previously performed by staff members or peers in projects where qualitative evaluation is required.

This tool is especially useful for projects where students must justify decisions, analyze a scenario, compare alternatives, or explain their approach.

However, this tool is not intended to completely replace manual reviews by staff. Critical projects that require expert evaluation, personalized feedback, mentorship, or deeper assessment of student reasoning and implementation quality should continue to include manual review as part of the evaluation process.

Self-Validation Tasks

A new validation mechanism called Self-Validation has been introduced for projects where traditional automated checking is not always appropriate.

In a Self-Validation task, students are presented with a checklist of required actions, experiments, observations, or implementation steps and must explicitly mark each item as completed. This approach is primarily intended for laboratory-style projects where students are encouraged to explore, investigate, test different approaches, and learn through experimentation.

The goal is not to validate a single final answer, but rather to ensure that students complete the key activities that make up the learning experience. Self-Validation is especially useful in projects involving infrastructure, debugging, benchmarking, AI experimentation, security exploration, configuration exercises, and other hands-on activities where multiple valid outcomes may exist.

While Self-Validation provides greater flexibility for exploratory learning, it may be combined with quizzes, automated checks, AI Checker evaluations, or manual reviews when additional validation is required.

New quiz types

SEIP v2 introduces new quiz types that differ significantly from the standalone questionnaires previously available on the platform. Rather than existing as separate assessments, these quizzes are integrated directly into projects and can be used as task-level validations within the project workflow.

This approach allows projects to combine different validation methods in a single learning experience. For example, a project may include automated checker tasks, self-validation activities, and quiz-based validations together. As discussed in the self-validation section, this combination provides a more flexible and effective way to assess both practical implementation and conceptual understanding.

Supported quiz types include:

- Multiple Choice
- Fill in the Blanks
- True / False
- Matching

Because these quizzes are embedded within projects, they can be used to validate theoretical concepts, design decisions, reading comprehension, and foundational knowledge at the exact moment students need to demonstrate understanding. This makes them especially useful in introductory projects, AI-related projects, software design modules, and conceptual infrastructure topics.

---

6. Main content scope and updates by trimester

Trimester 1 updates

Overall content scope

Trimester 1 combines onboarding, tool usage, programming foundations, C programming, AI introduction, and integration projects.

At a high level, the trimester includes:

- Introduction to Holberton development tools and learning environment.
- Computational thinking and visual programming through Scratch.
- Transition from visual programming to text-based programming.

- Introduction to computer systems.
- Git, GitHub, and source control workflows.
- Shell usage, permissions, redirections, filters, variables, and expansions.
- Introductory container concepts.
- C programming foundations.
- Debugging, benchmarking, memory analysis, and secure input handling.
- Pointers, arrays, strings, dynamic memory, structures, function pointers, variadic functions, and command-line arguments.
- Linked lists, hash tables, binary trees, sorting algorithms, and Big O.
- Advanced C topics such as recursion and file I/O.
- Integration projects that validate complete-program thinking.

AI Introduction

A new AI Introduction module has been added to introduce students to artificial intelligence and its application in software development.

The module includes:

- Introduction to Responsible Use of AI and Digital Tools
- Prompting Basics
- LLMs

The objective is not to turn students into AI specialists at this stage, but to establish responsible usage habits, basic AI literacy, and practical understanding of how AI tools can support learning and development.

Main Track introductory sequence

A new introductory sequence has been added before students move fully into text-based programming. This provides a smoother entry point into computational thinking and programming logic.

This sequence includes computational thinking, Scratch-based projects, an integrative Scratch project, transition to text-based programming, and an introduction to computer systems.

Most of these projects are evaluated through quizzes and selected practical exercises with automated checkers. The objective is to reduce unnecessary early friction while still validating that students understand fundamental programming concepts.

Git, GitHub, and workflow foundations

The curriculum includes Git and GitHub workflow foundations, including Git basics and GitHub Flow.

This helps students develop better workflow habits earlier in the program, including version control, branching, commits, pull requests, and collaboration-oriented development practices.

Linux, shell, and containers

The Linux and shell content remains a core foundation of the first trimester. Students work through shell basics, permissions, input and output redirections, filters, init files, variables, and expansions.

The Linux block now also includes an introductory project on containers and Docker concepts. This gives students early exposure to containerization, which becomes increasingly relevant in later trimesters, especially in infrastructure, DevOps, deployment, and cloud-related projects.

C programming updates

The C track remains one of the central foundations of T1. It includes introductory C programming, control flow, functions, nested loops, pointers, arrays, strings, memory allocation, structures, function pointers, variadic functions, command-line arguments, linked lists, hash tables, binary trees, sorting algorithms, and Big O.

Several new C projects have been added to reinforce debugging, benchmarking, memory analysis, secure input handling, and systems-level reasoning.

New or updated projects include:

- Debugging Visualization with PythonTutor
- Green Tech Efficiency & Benchmarking Lab
- Secure Input & Memory Lab
- AI Memory Visualizer + Valgrind Tracer
- Dynamic Analysis & Reverse Logic with GDB
- Secure Data Handling Lab

These projects are designed to make low-level programming more observable and practical. They also introduce students to professional debugging and analysis tools earlier in the curriculum.

Advanced Topics

Advanced Topics currently include C-focused projects such as recursion and file I/O.

These topics were separated from the Main Track because they are not strictly required for achieving the central outcomes of the trimester. However, they remain valuable for students who can benefit from deeper exposure to operating-system-level concepts, filesystem interaction, memory usage, and execution flow.

The separation also provides flexibility: students can continue strengthening their foundations without making these topics mandatory for the core score of the Main Track.

Integration Projects

Integration projects are used to validate that students can combine multiple concepts into complete programs.

A new early integration project, Simple Calculator, has been added. Its purpose is to validate early whether students can develop the logic of a complete program.

A new version of the printf project has also been created. This version has fewer tasks and a more direct scope, while preserving the project's value as a key integration milestone in C programming.

The integration track also includes additional projects that reinforce applied C programming, code review practices, and larger systems-level development.

Trimester 2 updates

Overall content scope

Trimester 2 expands the student's technical foundation from low-level programming into Python, software design, databases, backend development, frontend foundations, algorithms, AI-assisted development, and capstone work.

At a high level, the trimester includes:

- Python environment setup and first programs.
- Python control flow, functions, modularity, data structures, exceptions, classes, inheritance, polymorphism, abstract classes, interfaces, file handling, serialization, and debugging.
- UML modeling and design patterns using Python.
- SQL, relational database operations, joins, relationships, database design, normalization, and ORM.
- RESTful APIs.
- Asynchronous Python.
- Frontend foundations with HTML, CSS, JavaScript, DOM manipulation, and server-side rendering.
- Algorithmic problem-solving and interview-style challenges.
- AI-assisted development workflows.
- In-campus PLD activities.
- Real-time communication, AI agents, MCP, and the HBntory capstone.

AI 4 Devs

A new AI 4 Devs module introduces practical AI usage in software development workflows.

Projects include:

- IDE Workflow Integration
- API Prototyper
- AI Code Review
- UI Mockup from Text
- Architecture Blueprint Assistant
- Fix the AI-Generated Code

This module focuses on using AI as a development assistant rather than as a replacement for engineering knowledge. Students are expected to evaluate, adapt, and validate AI-generated outputs.

Python Main Track restructuring

The Python projects have been reworked to better reflect the fact that students already have programming experience from the C projects in Trimester 1.

The updated Python sequence keeps many relevant tasks but uses a more efficient approach. The focus shifts from general programming mechanics to Python language features, idioms, modularity, object-oriented programming, debugging, and practical application.

The Python block includes foundational syntax, control flow, functions, data structures, exceptions, object-oriented programming, inheritance, interfaces, file handling, serialization, and debugging strategies.

Software design and architecture

New projects have been added to introduce software design and architecture concepts using Python.

These projects include UML modeling and design patterns with Python.

The objective is to help students move beyond syntax and implementation, and start reasoning about structure, maintainability, abstraction, and design decisions.

SQL, databases, ORM, and APIs

The SQL and relational database module has been fully rewritten.

The updated database sequence provides a clearer progression from basic database operations to relationships, schema design, and normalization. These new projects use SQLite as the underlying database engine because it is lightweight, easy to use, and sufficient for the intended learning objectives. This approach removes the installation and configuration overhead associated with more complex relational database management systems such as MySQL or PostgreSQL, allowing students to focus on database concepts and application development.

The broader backend path also includes object-relational mapping, RESTful APIs, asynchronous programming, and server-side rendering. This gives students a more complete view of how Python applications interact with databases, expose services, and support web application behavior.

Frontend foundations

The T2 Main Track currently includes a frontend block covering HTML, CSS, JavaScript, DOM manipulation, and related web rendering concepts.

This block is part of the current curriculum scope, but it is one of the areas expected to be rewritten, updated, or better aligned with the new SEIP v2 direction. The current implementation should be treated as

part of the active curriculum while being aware that future updates may refine its structure, scope, and sequencing.

Algorithms

The Algorithms track reinforces problem-solving and technical interview readiness through focused projects. It covers algorithmic reasoning, recursion, parsing, validation, matrix manipulation, greedy reasoning, graph-style thinking, and other common problem-solving patterns.

This track runs alongside the Main Track throughout the trimester and supports the development of structured reasoning, implementation discipline, and complexity awareness.

PLD Activities

PLD Activities introduce one-day, in-campus, group-based projects.

These projects are designed to promote collaboration, research, prototyping, and presentation skills. Students work in groups throughout the day, investigate a topic, build a solution or proof of concept, and deliver a final presentation.

Evaluation is performed manually by staff.

Current PLD activities include UML-oriented design work and exploration of alternative data models beyond the relational model.

Agents and Capstone

A temporary independent curriculum groups together WebSockets, AI Agents, MCP, and the HBntory capstone project.

This block includes:

- Real-time communication with WebSockets.
- AI Agents in Python.
- MCP Servers in Python.
- HBntory.

The first three projects are expected to become part of the Python Main Track in the future. The capstone project will remain an independent integration project.

HBntory is a capstone project focused on building a comprehensive application that integrates the concepts and skills learned throughout the second trimester.

During the capstone period, students should not receive additional projects. This is important to preserve focus and allow students to complete a meaningful integration experience.

Trimester 3 updates

Overall content scope

Trimester 3 has been redesigned to be fully oriented toward infrastructure, DevOps, cloud, security, deployment, observability, and production-readiness.

At a high level, the trimester includes:

- Networking fundamentals and topology design.
- Web stack architecture, scalability, load balancing, and redundancy.
- Infrastructure as code.
- Configuration management.
- Container image optimization and multi-service application composition.
- CI/CD pipelines.
- Secure delivery pipelines.
- Secrets and configuration management.
- SBOM and software composition analysis.
- IAM and cloud hardening.
- Cloud deployment with AWS fundamentals.
- Observability with OpenTelemetry, Prometheus, and Grafana.
- Incident response and blameless postmortems.
- A production delivery capstone.
- Portfolio development and presentation.

This trimester is designed to help students understand how modern applications are deployed, operated, secured, observed, and maintained in production environments.

DevOps and production delivery focus

The T3 Main Track gives the final trimester a clearer identity. Students move from application development toward infrastructure, cloud, automation, security, observability, and production operations.

The curriculum introduces networking primitives, topology and routing, scalable web stacks, load balancing, Terraform, Ansible, Docker image optimization, multi-service composition, CI/CD pipelines, hardened delivery pipelines, secrets management, software composition analysis, IAM, AWS deployment, observability, incident response, and a secure production delivery capstone.

This progression is intended to help students understand not only how to build applications, but also how to deploy, secure, monitor, and operate them responsibly.

Portfolio

The Portfolio curriculum starts in week 1 and spans the full trimester.

Portfolio work is structured as a staged process that includes team formation and idea development, project charter development, technical documentation, MVP development and execution, project closure, and landing page preparation.

During the final month, students should focus exclusively on portfolio work. This ensures they have enough time to consolidate their work, improve their public-facing projects, prepare technical narratives, and strengthen their professional presentation.

---

7. Key improvements in SEIP v2

SEIP v2 introduces several important improvements compared to the previous version of the program.

Smoother onboarding

The new introductory sequence gives students a more progressive entry into programming. Computational thinking, Scratch, and the transition to text-based programming help reduce early overload while still building strong foundations.

Earlier validation of student readiness

Projects such as Simple Calculator provide an early signal of whether students can combine programming logic, control flow, input handling, and complete-program thinking.

This allows earlier identification of students who may need additional support before reaching larger C integration projects.

Stronger debugging culture

New projects involving PythonTutor, Valgrind, GDB, memory visualization, and debugging strategies help students develop practical troubleshooting skills earlier and more explicitly.

The objective is to make debugging a core engineering habit rather than a reactive activity students only perform when something fails.

Better alignment with modern development workflows

The curriculum now includes GitHub Flow, Docker concepts, AI-assisted workflows, architecture blueprints, code review, agents, MCP, CI/CD, infrastructure as code, observability, and cloud deployment.

This better reflects the tools, practices, and workflows students are likely to encounter in modern software engineering environments.

More intentional AI integration

AI is introduced progressively:

- T1 focuses on responsible use, prompting basics, and AI literacy.
- T2 focuses on AI as a practical developer tool.
- Later T2 projects introduce agents, tools, orchestration, and MCP.

This progression helps students use AI critically and responsibly, while still developing their own technical judgment.

Clearer Python progression

The Python track has been adapted for students who already completed a C-based programming foundation. This avoids unnecessary repetition and allows Python projects to focus more directly on language features, abstractions, debugging, and design.

Stronger software design foundations

UML, design patterns, architecture, and database design are now more explicit parts of the learning path.

This helps students move from simply implementing tasks to reasoning about how software systems are structured, maintained, and extended.

Broader full-stack preparation

Trimester 2 combines Python, databases, APIs, asynchronous programming, frontend foundations, and server-side rendering. While some frontend content is still pending review or rewrite, the current structure gives students exposure to the main layers involved in building web applications.

Stronger final integration experience in T2

The new HBntory capstone replaces the current HBnB project as the final integrative experience of the second trimester.

This change preserves the main goals of HBnB, including architecture, backend, database design, authentication, authorization, APIs, frontend integration, and teamwork, while expanding the scope to include modern topics such as AI agents, MCP, external APIs, and service separation.

Production-oriented final trimester

The third trimester now has a clear infrastructure and DevOps identity. Students finish the program with stronger exposure to deployment, pipelines, cloud, security, monitoring, incident response, and production operations.

---

8. Implementation considerations

The proposed trimester planning should be used as a reference model, while allowing for local adaptations when necessary.

Important implementation considerations include:

- This first version is stable in terms of conceptual direction, content coverage, and curriculum approach, but some project-level content is still being rewritten or updated, including the frontend block covering HTML, CSS, and JavaScript.
- The project lists included in the curriculum units should be considered the current reference for implementation, even when this document only describes content areas at a grouped level.
- T1 onboarding may require adjustment depending on student background and local operational needs.
- AI Introduction should be positioned early enough to shape responsible student behavior when using AI tools.
- Integration projects should be scheduled carefully to avoid overloading students.
- Advanced Topics in T1 should be understood as valuable reinforcement, but not part of the Main Track score.
- T2 capstone time should be protected. No other projects should be assigned during the HBntory capstone period.
- PLD Activities require campus-level coordination, staff presence, group organization, and manual evaluation.
- T3 portfolio time should be protected, especially during the final month.
- Staff should become familiar with the new quiz types, Self-Validation tasks, and AI Checker before the cohort starts.
- Manual review should remain available for critical projects requiring deeper evaluation, mentorship, or qualitative feedback.
- The updated project sequence should be reviewed before launch to identify local calendar conflicts.

---

9. Expected student outcomes

By the end of SEIP v2, students are expected to have stronger foundations across the full software engineering lifecycle.

They should be able to:

- Understand and apply computational thinking.
- Use Linux, shell, editors, Git, GitHub, and core developer tools.
- Build programs in C and Python.
- Debug programs using professional tools and strategies.
- Understand memory, input handling, data structures, and secure coding fundamentals.
- Use AI tools responsibly and critically as part of development workflows.
- Apply basic software design and architecture concepts.
- Work with relational databases and SQL.

- Build backend-oriented applications using Python, APIs, ORM, and asynchronous programming.
- Understand frontend foundations with HTML, CSS, JavaScript, DOM manipulation, and rendering concepts.
- Solve algorithmic problems with structured reasoning.
- Understand real-time communication, AI agents, and MCP-based tool integration.
- Design, deploy, secure, observe, and operate modern infrastructure.
- Build and present portfolio-ready technical work.

Appendix A — HBntory and the replacement of HBnB

A.1 Context

HBntory is the new final integrative project for Trimester 2. It replaces the current HBnB project as the main integration experience for the trimester.

The current HBnB project is developed progressively throughout Trimester 2, in parallel with the Main Track, and is organized into four major parts:

- HBnB - UML
- HBnB - BL and API
- HBnB - Auth & DB
- HBnB - Simple Web Client

HBntory changes this implementation model. Instead of running throughout the trimester as several parallel sub-projects, it is planned as a concentrated two-week final capstone at the end of T2. During this period, students should not receive additional projects, so they can focus on integrating the concepts learned throughout the trimester into a coherent final system.

This change is important from an implementation perspective. HBnB required students to build one evolving application over multiple checkpoints during the trimester. HBntory keeps the integrative purpose but shifts the experience toward a protected final project window, where the expectation is to apply accumulated skills in a more complete, realistic, and modern system.

A.2 What HBntory is

HBntory is an inventory management platform for a fictional retail company with multiple branches.

The system includes two main user-facing areas:

1. A Backoffice used by authenticated internal users to manage users, branches, and stock.
2. A Client Web Interface where anonymous external users can ask natural-language questions about products and stock.

The final system integrates backend development, relational databases, authentication, authorization, external APIs, AI agents, MCP servers, and web interfaces into a single application.

At a high level, HBntory requires students to build and integrate:

- A Backoffice Service.
- A relational database.
- An external Product API.
- A Product MCP Server.
- An AI Query Service.
- A public Client Web Interface.

The project is designed for teams of two to three students and is expected to be completed over two full-time weeks.

A.3 Continuity with HBnB

HBntory preserves the main pedagogical purpose of HBnB: giving students a substantial project where they must design, build, integrate, document, and explain a medium-sized software system.

The domain changes from an AirBnB-like application to an inventory management system, but the core learning objectives remain strongly aligned.

Both projects require students to work with:

- Software architecture and technical documentation.
- UML or architecture diagrams.
- Layered or service-oriented design.
- Business logic implementation.
- Backend APIs.
- Relational data modeling.
- Persistent storage.
- Authentication.
- Authorization and role-based access control.
- Secure password handling.
- Frontend or client-side interaction.
- Integration between frontend and backend components.
- Testing, documentation, and final explanation of technical decisions.

In this sense, HBntory should not be understood as a removal of HBnB's learning objectives. It is a redesign of the final integration experience around a new domain, a more current architecture, and a broader set of technologies.

A.4 Main differences from HBnB

The most visible change is the project domain.

HBnB focuses on a lodging marketplace model, with users, places, reviews, amenities, authentication, database persistence, APIs, and a simple web client.

HBntory focuses on an inventory management context, with users, branches, stock, product identifiers, a Backoffice, a public natural-language query interface, and integration with an external product system.

The second major difference is the delivery format.

HBnB is divided into four parts that run across the trimester. HBntory is designed as a final two-week integrative project. This makes the project more concentrated and allows it to function as a true capstone for T2.

The third major difference is the architecture.

HBnB is mainly centered on a Flask-based backend, persistence layer, API, authentication, and frontend client. HBntory expands the architecture into a multi-component system with separate responsibilities:

- Backoffice Service for authenticated internal operations.
- AI Query Service for natural-language user questions.
- Product MCP Server as a bridge between AI agents and the external Product API.
- External Product API as the authoritative source of product data.
- Client Web Interface for anonymous product and stock queries.

The fourth major difference is the introduction of AI and MCP.

HBntory requires students to integrate one or more AI agents into a backend service and connect those agents to tools through MCP. Students must also ensure that AI-generated responses are grounded in actual product and stock data, and that the system does not invent unavailable information.

A.5 Concepts preserved from HBnB

HBntory keeps the most important software engineering concepts already present in HBnB.

Architecture and design

HBnB begins with technical documentation, UML diagrams, layered architecture, class diagrams, and sequence diagrams. HBntory keeps this emphasis by requiring architecture planning, service responsibility definition, data-flow reasoning, communication strategy decisions, and documented trade-offs.

Students must identify the services in the system, explain the responsibility of each service, describe how services communicate, define which data is stored locally, and clarify which data comes from the external Product API.

Backend and business logic

HBnB requires students to implement business logic entities and expose API endpoints. HBntory preserves this expectation through stock management, user management, branch management, product integration, and Backoffice functionality.

The business domain changes, but students still need to model rules, enforce constraints, and build the operational logic of a real system.

Database and persistence

HBnB introduces persistent storage with SQLAlchemy and relational database modeling. HBntory continues this direction by requiring a relational schema for users, branches, and stock, implemented with SQLAlchemy.

HBntory also introduces an important architectural constraint: product details must not be stored locally. The local database stores only internal system data, such as users, branches, stock quantities, and product identifiers. Product names, descriptions, prices, and metadata must come from the external Product API.

This reinforces a stronger understanding of data ownership, system boundaries, and integration with external services.

Authentication and authorization

HBnB requires authentication, secure password storage, and role-based access control. HBntory keeps these objectives through Backoffice authentication, secure password hashing, admin/common-user roles, and backend-enforced authorization rules.

In HBntory, common users are assigned to exactly one branch and can only operate on the stock of that branch. The admin user can manage users but cannot manage stock. These constraints require students to implement authorization rules in backend logic, not only in the user interface.

Frontend integration

HBnB includes a simple web client built with HTML, CSS, JavaScript, and API interaction. HBntory preserves frontend integration through both the Backoffice interface and the public Client Web Interface.

The Backoffice interface supports authenticated internal operations. The public client interface allows anonymous users to ask natural-language questions about products and stock.

Final integration and presentation

HBnB requires students to connect multiple parts of a larger application. HBntory makes this expectation explicit through integration testing, README documentation, architecture documentation, test evidence, and a final presentation.

Students must not only build the system but also explain architectural decisions, justify trade-offs, and demonstrate the final integrated application.

A.6 Concepts expanded in HBntory

HBntory expands the scope of the final project in several important ways.

External system integration

Product data must come from an external Product API. Students must avoid duplicating product details in the local Backoffice database and must reason about boundaries between internal and external data.

This introduces a more realistic integration scenario where the application depends on an external service as the authoritative source for part of its data.

Service separation

The AI Query Service is independent from the Backoffice. This requires students to think beyond a single monolithic backend and reason about service responsibilities, communication, and integration.

This separation also helps students understand how different backend services can have different responsibilities, users, interfaces, and data-access patterns.

MCP server implementation

Students must implement a Product MCP Server that exposes controlled tools for AI agents to list products and retrieve product details. This introduces a modern tool-integration pattern that is not present in HBnB.

The MCP server acts as a bridge between the AI system and the external Product API. It must expose clear tools, handle product lookup, manage errors, and provide information in a structure that the AI agent can use.

AI agent integration

Students must integrate one or more AI agents into the system. These agents must answer natural-language questions using actual product and stock information, rather than generating unsupported responses.

This introduces practical experience with AI agents as part of a software system, not only as standalone tools.

REST vs WebSocket trade-offs

Students must decide whether the Client Web Interface communicates with the AI Query Service using REST or WebSockets. Both options are valid, but students must justify their decision based on the system requirements and their implementation capacity.

A REST API may be simpler because each question can be treated independently. WebSockets may be justified if the team wants to support real-time behavior, streaming responses, or a chat-like experience.

The goal is not to force a single technical answer, but to make students reason about trade-offs and choose an approach that fits the project.

Grounded AI behavior

HBntory explicitly requires that the AI system avoid inventing product names, product details, stock quantities, or branch availability. If information is unavailable, the response must clearly say so.

This introduces an important practical lesson about reliability, tool use, and responsible AI behavior in software systems.

Students must design the AI Query Service so that responses are grounded in available data from the Product API and stock information, rather than relying only on the model's generated output.

Controlled data access for AI

HBntory requires students to decide how the AI system will access stock information.

Possible approaches include:

- Extending the MCP server with stock-query tools.
- Using a database MCP tool.
- Implementing a controlled internal API used by the AI service.

This decision introduces important architectural reasoning around safety, boundaries, and controlled access patterns.

A.7 Comparison summary

Area	HBnB	HBntory
Project role	Main integrative project across T2	Final T2 capstone
Format	Four parts distributed through the trimester	Concentrated two-week final project
Domain	Lodging marketplace	Inventory management system
Architecture focus	Layered backend, API, persistence, frontend	Multi-component system with separated services
Design work	UML, class diagrams, sequence diagrams	Architecture planning, service diagrams, technical decisions

Area	HBnB	HBntory
Backend	Flask API and business logic	Backoffice service and AI Query Service
Database	SQLAlchemy and relational persistence	SQLAlchemy relational database for users, branches, and stock
Authentication	JWT/authentication and role-based access	Backoffice authentication and role-based authorization
Frontend	Simple web client	Backoffice interface and public natural-language client
External systems	Limited external service integration	External Product API
AI integration	Not a central objective	AI agents as a required system component
MCP	Not included	Required Product MCP Server
Communication decisions	Mainly REST/API interaction	REST or WebSocket decision for client-to-AI-service communication
Final integration	Progressive integration through parts	Final integrated system and presentation

A.8 Why this replacement is pedagogically consistent

HBntory maintains the central role that HBnB played in the curriculum: it is still the project where students must connect architecture, backend development, database design, authentication, authorization, frontend interaction, documentation, teamwork, and final presentation.

The replacement is pedagogically consistent because it keeps the core software engineering objectives while aligning the final project with the new direction of SEIP v2:

- More explicit architectural decision-making.
- Stronger integration of AI into software development.
- Exposure to agent-based systems and tool use.
- Use of MCP as a modern integration mechanism.
- A clearer final capstone period with protected project time.
- A domain that naturally supports internal tools, external APIs, and public user-facing queries.

HBntory should therefore be presented as an evolution of HBnB, not as a reduction of scope. The project preserves the original integration objectives and extends them into a more modern, AI-aware, service-oriented application.

A.9 Implementation implications

The replacement of HBnB with HBntory has several practical implications.

First, the capstone period must be protected. Since HBntory concentrates the integration experience into two full-time weeks, students should not receive additional projects during that period.

Second, the preceding T2 content becomes more important as preparation for the final project. Python, SQL, ORM, APIs, frontend foundations, WebSockets, agents, and MCP all contribute directly to the final integration experience.

Third, the project requires careful expectation management. HBntory is not intended to be a polished commercial application. The priority is correctness, integration, architecture, role separation, grounded AI behavior, and the ability to explain technical decisions.

Fourth, evaluation should consider both the final system and the reasoning behind the implementation. Important aspects include architecture, database design, authentication, authorization, service boundaries, product API integration, MCP tool design, AI grounding, frontend usability, testing evidence, documentation, and final presentation.

---

10. Summary

SEIP v2 is a significant evolution of the Software Engineering Intensive Program. It keeps the core engineering foundations of the original program while introducing a clearer progression, stronger onboarding, modern development workflows, AI literacy, AI-assisted development, improved evaluation tools, software architecture foundations, full-stack exposure, and a production-oriented infrastructure and DevOps trimester.

The new structure is designed to better support students from their first exposure to computational thinking through to production delivery and portfolio preparation.

This version should be understood as stable in its educational direction and content coverage, while still allowing for continued refinement of specific projects, sequencing details, and curriculum-unit organization.

The replacement of HBnB with HBntory is one of the clearest examples of the SEIP v2 direction: the core integration goals are preserved, while the final project is updated to include service separation, external APIs, AI agents, MCP, and grounded natural-language interaction.

The curriculum links and project lists should be reviewed as the current implementation reference. Key integration, capstone, and portfolio periods should be protected, and the new evaluation tools should be introduced with enough preparation to ensure a smooth rollout.