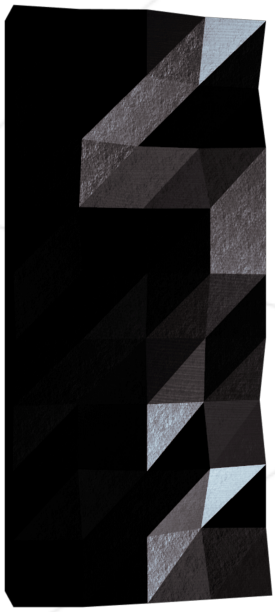

ft_transcendence

Surprise.

Summary:
This project involves undertaking tasks you have never done before.
Remember the beginning of your journey in computer science.
Look at you now; it's time to shine!

Version: 21.1

Contents

Chapter I

AI Instructions

Context

During your learning journey, AI can assist with many different tasks. Take the time to explore the various capabilities of AI tools and how they can support your work. However, always approach them with caution and critically assess the results. Whether it's code, documentation, ideas, or technical explanations, you can never be completely sure that your question was well-formed or that the generated content is accurate. Your peers are a valuable resource to help you avoid mistakes and blind spots.

Main message

- Use AI to reduce repetitive or tedious tasks.
- Develop prompting skills — both coding and non-coding — that will benefit your future career.
- Learn how AI systems work to better anticipate and avoid common risks, biases, and ethical issues.
- Continue building both technical and power skills by working with your peers.
- Only use AI-generated content that you fully understand and can take responsibility for.

Learner rules:

- You should take the time to explore AI tools and understand how they work, so you can use them ethically and reduce potential biases.
- You should reflect on your problem before prompting — this helps you write clearer, more detailed, and more relevant prompts using accurate vocabulary.
- You should develop the habit of systematically checking, reviewing, questioning, and testing anything generated by AI.
- You should always seek peer review — don't rely solely on your own validation.

Phase outcomes:

- Develop both general-purpose and domain-specific prompting skills.
- Boost your productivity with effective use of AI tools.
- Continue strengthening computational thinking, problem-solving, adaptability, and collaboration.

Comments and examples:

- You'll regularly encounter situations — exams, evaluations, and more — where you must demonstrate real understanding. Be prepared, keep building both your technical and interpersonal skills.
- Explaining your reasoning and debating with peers often reveals gaps in your understanding. Make peer learning a priority.
- AI tools often lack your specific context and tend to provide generic responses. Your peers, who share your environment, can offer more relevant and accurate insights.
- Where AI tends to generate the most likely answer, your peers can provide alternative perspectives and valuable nuance. Rely on them as a quality checkpoint.

✓ Good practice:

> I ask AI: "How do I test a sorting function?" It gives me a few ideas. I try them out and review the results with a peer. We refine the approach together.

✗ Bad practice:

> I ask AI to write a whole function, copy-paste it into my project. During peer-evaluation, I can't explain what it does or why. I lose credibility — and I fail my project.

✓ Good practice:

> I use AI to help design a parser. Then I walk through the logic with a peer. We catch two bugs and rewrite it together — better, cleaner, and fully understood.

✗ Bad practice:

> I let Copilot generate my code for a key part of my project. It compiles, but I can't explain how it handles pipes. During the evaluation, I fail to justify and I fail my project.

Chapter II

Preamble

First of all, **congratulations on reaching this milestone!** You are now entering the final project of your Common Core, and yes, it will not be easy.

`Transcendence` is a **group project (4-5 people)**, which is intended to boost your creativity, self-confidence, adaptability to new technologies, and **teamwork skills**.
You'll create a real-world web application **as a team** that can move in many directions, depending on the modules you choose and the choices you make. Make sure to think things through together as a team before you start.
The project is divided into two parts:

- The **mandatory part**, which is the fixed core of the project to which **every team member must contribute**.
- A set of **modules**, which you can choose and which count toward the final grade.

`This is a long group project. Poor early choices and lack of team coordination will cost a lot of time. Your project and team management will strongly impact your results. All team members must actively participate and contribute to both the mandatory part and the modules.`

II.1 Team Organization and Project Management

As this is a group project, proper team organization is crucial for success. You must establish clear roles and responsibilities from the start.

II.1.1 Required Team Roles

Your team **must assign** the following roles (one person can have multiple roles if the team has 4 members):

- **Product Owner (PO)**: Defines the product vision, prioritizes features, and ensures the project meets user needs.
    - Maintains the product backlog.

- Makes decisions on features and priorities.
- Validates completed work.
- Communicates with stakeholders (evaluators, peers).

- **Project Manager (PM) / Scrum Master**: Facilitates team coordination and removes obstacles.
  - Organizes team meetings and planning sessions.
  - Tracks progress and deadlines.
  - Ensures team communication.
  - Manages risks and blockers.
- **Technical Lead / Architect**: Oversees technical decisions and architecture.
  - Defines technical architecture.
  - Makes technology stack decisions.
  - Ensures code quality and best practices.
  - Reviews critical code changes.
- **Developers** (all team members): Implement features and modules.
  - Write code for assigned features.
  - Participate in code reviews.
  - Test their implementations.
  - Document their work.

Team Size:

- **4 people:** Some members will have multiple roles (e.g., PM + Developer, PO + Developer).
- **5 people:** Roles can be more specialized, with dedicated PO, PM, Tech Lead, and 2 Developers.

All roles must be clearly documented in your README.md.

II.1.2 Recommended Project Management Practices

While not mandatory, we strongly recommend implementing some basic project management practices to help your team succeed:

- **Regular communication**: Meet regularly (weekly or bi-weekly) to sync on progress and blockers.
- **Task organization**: Use simple tools like GitHub Issues, Trello, or even a shared document to track who does what.

- **Work breakdown**: Divide the project into smaller, manageable tasks.
- **Code reviews**: Try to have at least one other team member review important code changes.
- **Documentation**: Keep notes of important decisions and how things work.
- **Communication channel**: Use Discord, Slack, or similar for quick team communication.

These practices are recommendations to help you organize your work. Find what works best for your team! The important thing is that everyone contributes and the work is well-coordinated.

During evaluation, the team will be asked to explain:

- How roles were distributed.
- How work was organized and divided.
- How you communicated and coordinated as a team.
- How each member contributed to the project.

All team members must be able to explain the project and their contributions.

Chapter III

Mandatory part

The project content is up to you. Yes, you have to bring your own ideas and decide together what application to build as a team.

It's a step forward from what you did previously in the Common Core. You need to think about the project as a whole, not just the features you're going to implement.

Of course, you won't be completely free—there are constraints—but the idea is yours.

III.1 What are we doing?

First, you will have to create a comprehensive `README.md` file. The detailed requirements for the README are specified in the README Requirements section at the end of this document.

Project Examples: Your project can take many forms. Here are some valid examples:

- A multiplayer Pong game with tournament system
- A collaborative platform with real-time features
- A social network with user interactions
- An online game (Chess, Tic-Tac-Toe, etc.) with matchmaking
- A project management application
- Any other creative web application that meets the requirements

The key is to create something engaging that demonstrates your technical skills and creativity.

III.2 General requirements

Building an entire project is complicated, and many things can go wrong. To help you, we will provide a list of general requirements that you must follow. If you don't follow them, your project will be rejected.

The requirements are the following:

- The project must be a web application, and requires a frontend, backend, and a database.
- Git must be used with clear and meaningful commit messages. The repository must show:
  - Commits from all team members.
  - Clear commit messages describing the changes.
  - Proper work distribution across the team.
- Deployment must use a **containerization solution** (Docker, Podman, or equivalent) and run with a single command.
- Your website must be compatible with the latest stable version of **Google Chrome**.
- No warnings or errors should appear in the browser console.
- The project must include accessible **Privacy Policy** and **Terms of Service** pages with relevant content.

Privacy Policy and Terms of Service: These pages will be verified during evaluation. They must:

- Be easily accessible from the application (e.g., footer links).
- Contain relevant and appropriate content for your project.
- Not be placeholder or empty pages.

Missing or inadequate Privacy Policy/Terms of Service pages will result in project rejection.

Multi-user Support (Mandatory): Your website **must support multiple users simultaneously**. This is a core requirement of the project. Users should be able to interact with the application at the same time without conflicts or performance issues. This includes:

- Multiple users can be logged in and active at the same time.
- Concurrent actions by different users are handled properly.
- Real-time updates are reflected across all connected users when applicable.
- No data corruption or race conditions occur with simultaneous user actions.

III.3 Technical requirements

This section, like the previous one, is mandatory. You will then be able to choose the modules you want to use in the next chapter.

- A frontend that is clear, responsive, and accessible across all devices.
- Use a **CSS framework or styling solution** of your choice (e.g., Tailwind CSS, Bootstrap, Material-UI, Styled Components, etc.).
- Store credentials (API keys, environment variables, etc.) in a local `.env` file that is ignored by Git, and provide an `.env.example` file.
- The database must have a clear schema and well-defined relations.
- Your application must have a basic **user management system**. Users must be able to sign up and log in securely:
  - At minimum: **email** and **password** authentication with proper security (hashed passwords, salted, etc.).
  - Additional authentication methods (OAuth, 2FA, etc.) can be implemented via modules.
- All forms and user inputs must be properly validated in both the frontend and backend.
- Any connection to the backend, from a browser, from a script, from an external API, etc., must use **HTTPS**. Connections inside the backend itself (e.g., web server and database, software inside your container(s)) can be without encryption.

What is a Framework? For this project, a **framework** is defined as a comprehensive tool that provides:

- A structured architecture and conventions for organizing code.
- Built-in features for common tasks (routing, state management, etc.).
- A complete ecosystem of tools and libraries.

Examples:

- **Frontend frameworks:** React, Vue, Angular, Svelte, Next.js (these are frameworks).
- **Backend frameworks:** Express, Fastify, NestJS, Django, Flask, Ruby on Rails.
- **Not frameworks:** jQuery (library), Lodash (utility library), Axios (HTTP client).

Note: React is considered a framework in this context due to its ecosystem and architectural patterns, even though it is technically a library.

Chapter IV

Modules

You will need to earn **14 points** in total to complete your project. Each major module is worth **2 points**, and each minor module is worth **1 point**.

The following categories are available. You may choose multiple modules from any category:

- **Web**
- **Accessibility and Internationalization**
- **User Management**
- **Artificial Intelligence**
- **Cybersecurity**
- **Gaming and user experience**
- **Devops**
- **Data and Analytics**
- **Blockchain**
- **Modules of choice**

We strongly recommend choosing modules only after your ideas are clear and you have a good understanding of what you want to build.

Additionally, aiming for more than **14 points** in total may be a good idea, especially if some modules aren't validated during the evaluation.

Important - Module Dependencies and Evaluation:

- Some modules require other modules to be implemented first (marked with info notes).
- **Gaming modules** (AI Opponent, Tournament, Game customization, Spectator mode, Multiplayer 3+, Add another game) **require that at least one game be implemented first.**
- The Game Statistics module requires that a game be implemented.
- Advanced chat features require the basic chat functionality from the "User interaction" module.
- SSR is incompatible with the ICP blockchain backend.
- Plan your modules carefully to ensure they work together coherently!
- **During evaluation:** You will be asked to demonstrate each claimed module. Only fully functional and properly implemented modules will be counted toward your final score. Non-functional or incomplete modules = 0 points.

IV.1 Web

- **Major**: Use a framework for both the frontend and backend.
  - Use a frontend framework (React, Vue, Angular, Svelte, etc.).
  - Use a backend framework (Express, NestJS, Django, Flask, Ruby on Rails, etc.).
  - Full-stack frameworks (Next.js, Nuxt.js, SvelteKit) count as both if you use both their frontend and backend capabilities.
- **Minor**: Use a frontend framework (React, Vue, Angular, Svelte, etc.).
- **Minor**: Use a backend framework (Express, Fastify, NestJS, Django, etc.).
- **Major**: Implement real-time features using WebSockets or similar technology.
  - Real-time updates across clients.
  - Handle connection/disconnection gracefully.
  - Efficient message broadcasting.
- **Major**: Allow users to interact with other users. The minimum requirements are:
  - A **basic chat** system (send/receive messages between users).
  - A **profile** system (view user information).
  - A **friends** system (add/remove friends, see friends list).
- **Major**: A public API to interact with the database with a secured API key, rate limiting, documentation, and at least 5 endpoints:
  - `GET /api/{something}`
  - `POST /api/{something}`
  - `PUT /api/{something}`
  - `DELETE /api/{something}`
- **Minor**: Use an ORM for the database.
- **Minor**: A complete notification system for all creation, update, and deletion actions.
- **Minor**: Real-time collaborative features (shared workspaces, live editing, collaborative drawing, etc.).
- **Minor**: Server-Side Rendering (SSR) for improved performance and SEO.
- **Minor**: Progressive Web App (PWA) with offline support and installability.
- **Minor**: Custom-made design system with reusable components, including a proper color palette, typography, and icons (minimum: 10 reusable components).

- **Minor**: Implement advanced search functionality with filters, sorting, and pagination.
- **Minor**: File upload and management system.
  - Support multiple file types (images, documents, etc.).
  - Client-side and server-side validation (type, size, format).
  - Secure file storage with proper access control.
  - File preview functionality where applicable.
  - Progress indicators for uploads.
  - Ability to delete uploaded files.

IV.2 Accessibility and Internationalization

- **Major**: Complete accessibility compliance (WCAG 2.1 AA) with screen reader support, keyboard navigation, and assistive technologies.
- **Minor**: Support for multiple languages (at least 3 languages).
  - Implement i18n (internationalization) system.
  - At least 3 complete language translations.
  - Language switcher in the UI.
  - All user-facing text must be translatable.
- **Minor**: Right-to-left (RTL) language support.
  - Support for at least one RTL language (Arabic, Hebrew, etc.).
  - Complete layout mirroring (not just text direction).
  - RTL-specific UI adjustments where needed.
  - Seamless switching between LTR and RTL.
- **Minor**: Support for additional browsers.
  - Full compatibility with at least 2 additional browsers (Firefox, Safari, Edge, etc.).
  - Test and fix all features in each browser.
  - Document any browser-specific limitations.
  - Consistent UI/UX across all supported browsers.

IV.3 User Management

- **Major**: Standard user management and authentication.
  - Users can update their profile information.
  - Users can upload an avatar (with a default avatar if none provided).
  - Users can add other users as friends and see their online status.
  - Users have a profile page displaying their information.
- **Minor**: Game statistics and match history (requires a game module).
  - Track user game statistics (wins, losses, ranking, level, etc.).
  - Display match history (1v1 games, dates, results, opponents).
  - Show achievements and progression.
  - Leaderboard integration.

This module requires you to have implemented at least one game (see "Gaming and user experience" section). You cannot claim this module without a functional game.

- **Minor**: Implement remote authentication with OAuth 2.0 (Google, GitHub, 42, etc.).
- **Major**: Advanced permissions system:
  - View, edit, and delete users (CRUD).
  - Roles management (admin, user, guest, moderator, etc.).
  - Different views and actions based on user role.
- **Major**: An organization system:
  - Create, edit, and delete organizations.
  - Add users to organizations.
  - Remove users from organizations.
  - View organizations and allow users to perform specific actions within an organization (minimum: create, read, update).
- **Minor**: Implement a complete 2FA (Two-Factor Authentication) system for the users.
- **Minor**: User activity analytics and insights dashboard.

IV.4 Artificial Intelligence

- **Major**: Introduce an AI Opponent for games.
  - The AI must be challenging and able to win occasionally.
  - The AI should simulate human-like behavior (not perfect play).
  - If you implement game customization options, the AI must be able to use them.
  - You must be able to explain your AI implementation during evaluation.

This module requires you to have implemented at least one game (see "Gaming and user experience" section). The AI must be able to play your game competently.

- **Major**: Implement a complete RAG (Retrieval-Augmented Generation) system.
  - Interact with a large dataset of information.
  - Users can ask questions and get relevant answers.
  - Implement proper context retrieval and response generation.
- **Major**: Implement a complete LLM system interface.
  - Generate text and/or images based on user input.
  - Handle streaming responses properly.
  - Implement error handling and rate limiting.
- **Major**: Recommendation system using machine learning.
  - Personalized recommendations based on user behavior.
  - Collaborative filtering or content-based filtering.
  - Continuously improve recommendations over time.
- **Minor**: Content moderation AI (auto moderation, auto deletion, auto warning, etc.)
- **Minor**: Voice/speech integration for accessibility or interaction.
- **Minor**: Sentiment analysis for user-generated content.
- **Minor**: Image recognition and tagging system.

IV.5 Cybersecurity

- **Major**: Implement WAF/ModSecurity (hardened) + HashiCorp Vault for secrets:
  - Configure strict ModSecurity/WAF.
  - Manage secrets in Vault (API keys, credentials, environment variables), encrypted and isolated.

IV.6 Gaming and user experience

- **Major**: Implement a complete web-based game where users can play against each other.
  - The game can be real-time multiplayer (e.g., Pong, Chess, Tic-Tac-Toe, Card games, etc.).
  - Players must be able to play live matches.
  - The game must have clear rules and win/loss conditions.
  - The game can be 2D or 3D.
- **Major**: Remote players — Enable two players on separate computers to play the same game in real-time.
  - Handle network latency and disconnections gracefully.
  - Provide a smooth user experience for remote gameplay.
  - Implement reconnection logic.
- **Major**: Multiplayer game (more than two players).
  - Support for three or more players simultaneously.
  - Fair gameplay mechanics for all participants.
  - Proper synchronization across all clients.

This module requires you to have implemented at least one game (see "Gaming and user experience" section). You're extending your game to support three or more players.

- **Major**: Add another game with user history and matchmaking.
  - Implement a second distinct game.
  - Track user history and statistics for this game.
  - Implement a matchmaking system.
  - Maintain performance and responsiveness.

This module requires you to have already implemented a first game (see "Implement a complete web-based game" module above). You cannot claim this module without having a functional first game.

- **Major**: Implement advanced 3D graphics using a library like Three.js or Babylon.js.
  - Create an immersive 3D environment.
  - Implement advanced rendering techniques.
  - Ensure smooth performance and user interaction.
- **Minor**: Advanced chat features (enhances the basic chat from "User interaction" module).
  - Ability to block users from messaging you.
  - Invite users to play games directly from chat.
  - Game/tournament notifications in chat.
  - Access to user profiles from chat interface.
  - Chat history persistence.
  - Typing indicators and read receipts.

This module enhances the basic chat system from the "Allow users to interact" module. You cannot claim this module without having implemented the basic chat first.

- **Minor**: Implement a tournament system.
  - Clear matchup order and bracket system.
  - Track who plays against whom.
  - Matchmaking system for tournament participants.
  - Tournament registration and management.

This module requires you to have implemented at least one game (see "Gaming and user experience" section). You cannot have tournaments without a game to play.

- **Minor**: Game customization options.
  - Power-ups, attacks, or special abilities.

- Different maps or themes.
- Customizable game settings.
- Default options must be available.

This module requires you to have implemented at least one game (see "Gaming and user experience" section). You're adding customization to an existing game.

- **Minor**: A gamification system to reward users for their actions.
  - Implement at least 3 of the following: achievements, badges, leaderboards, XP/level system, daily challenges, rewards
  - System must be persistent (stored in database)
  - Visual feedback for users (notifications, progress bars, etc.)
  - Clear rules and progression mechanics

While this is a Minor module (1 point), implementing a complete gamification system can be substantial. Focus on quality over quantity-three well-implemented features are better than six poorly done ones.

- **Minor**: Implement spectator mode for games.
  - Allow users to watch ongoing games.
  - Real-time updates for spectators.
  - Optional: spectator chat.

This module requires you to have implemented at least one game (see "Gaming and user experience" section). Spectators need a game to watch.

IV.7 Devops

- **Major**: Infrastructure for log management using ELK (Elasticsearch, Logstash, Kibana).
  - Elasticsearch to store and index logs.
  - Logstash to collect and transform logs.
  - Kibana for visualization and dashboards.

  - Implement log retention and archiving policies.
  - Secure access to all components.
- **Major**: Monitoring system with Prometheus and Grafana.
  - Set up Prometheus to collect metrics.
  - Configure exporters and integrations.
  - Create custom Grafana dashboards.
  - Set up alerting rules.
  - Secure access to Grafana.
- **Major**: Backend as microservices.
  - Design loosely-coupled services with clear interfaces.
  - Use REST APIs or message queues for communication.
  - Each service should have a single responsibility.
- **Minor**: Health check and status page system with automated backups and disaster recovery procedures.

IV.8 Data and Analytics

- **Major**: Advanced analytics dashboard with data visualization.
  - Interactive charts and graphs (line, bar, pie, etc.).
  - Real-time data updates.
  - Export functionality (PDF, CSV, etc.).
  - Customizable date ranges and filters.
- **Minor**: Data export and import functionality.
  - Export data in multiple formats (JSON, CSV, XML, etc.).
  - Import data with validation.
  - Bulk operations support.
- **Minor**: GDPR compliance features.
  - Allow users to request their data.
  - Data deletion with confirmation.
  - Export user data in a readable format.
  - Confirmation emails for data operations.

IV.9 Blockchain

- **Major**: Store tournament scores on the Blockchain.
  - Use Avalanche and Solidity smart contracts on a test blockchain.
  - Implement smart contracts to record, manage, and retrieve tournament scores.
  - Ensure data integrity and immutability.
- **Minor**: Use ICP (Internet Computer Protocol) for a backend that runs on a blockchain (incompatible with SSR).

IV.10 Modules of choice

- **Major**: Implement a custom module that is not listed above.
  - The module must be substantial and demonstrate technical complexity.
  - You must provide proper justification in your README.md explaining:
    - Why you chose this module.
    - What technical challenges it addresses.
    - How it adds value to your project.
    - Why it deserves Major module status (2 points).
  - Taking shortcuts or implementing trivial features will result in rejection.
  - Be creative and think outside the box.
  - The module should be relevant to your project context.
- **Minor**: Same as the major module but smaller in scope and less complex.
  - Must still demonstrate technical skill and creativity.
  - Should add meaningful value to your project.
  - Requires justification in README.md (similar to Major, but for 1 point).

Chapter V

Project Ideas and Examples

To help you get started and inspire your creativity, this chapter provides concrete project ideas and examples of how to reach the required 14 points. Remember, these are just suggestions — feel free to be creative and come up with your own unique ideas!

V.1 Example: Building a Pong Game

If you choose to create a Pong game (like the original project), here's how you can reach 14 points:

- **Gaming and user experience**: Web-based game (2pts) + Remote players (2pts) + Tournament system (1pt) + Game customization (1pt) = 6 points
- **User Management**: Standard user management (2pts) + OAuth (1pt) = 3 points
- **Web**: Use frameworks (frontend + backend = 2pts) + ORM (1pt) = 3 points
- **Artificial Intelligence**: AI Opponent (2pts) = 2 points

Total: 14 points

This is just one example. You can mix and match modules from different categories to create your own unique project. The key is to ensure that your modules work together coherently and add value to your application.

V.2 Gaming Projects

These projects focus on interactive gameplay and user competition:

- **Multiplayer Pong**: Classic Pong with tournaments, remote play, AI opponents, and power-ups.
    - *Suggested modules*: Web-based game, Remote players, Tournament system, AI Opponent, Game customization
    - *Point potential*: 14+ points

- **Online Chess Platform**: Real-time chess with matchmaking, ELO rating, game analysis, and spectator mode.
  - *Suggested modules*: Web-based game, Remote players, AI Opponent, Spectator mode, Game statistics
  - *Point potential*: 15+ points
- **Card Game Arena**: Multiplayer card games (Poker, Uno, etc.) with tournaments and leaderboards.
  - *Suggested modules*: Web-based game, Multiplayer 3+, Tournament system, Gamification
  - *Point potential*: 14+ points
- **Battle Royale Mini-Game**: Simple browser-based battle royale game with multiple players.
  - *Suggested modules*: Web-based game, Multiplayer 3+, Real-time features, Game customization
  - *Point potential*: 14+ points
- **Trivia/Quiz Platform**: Real-time multiplayer quiz game with categories and tournaments.
  - *Suggested modules*: Web-based game, Multiplayer 3+, Tournament system, Gamification, Analytics dashboard
  - *Point potential*: 15+ points

V.3 Social and Collaborative Projects

These projects emphasize user interaction and community building:

- **Social Network**: User profiles, posts, comments, likes, friends, real-time chat, and notifications.
  - *Suggested modules*: User interaction, Real-time features, Notification system, Advanced chat, File upload
  - *Point potential*: 14+ points
- **Collaborative Workspace**: Real-time document editing, project management, team chat, and file sharing.
  - *Suggested modules*: Real-time collaborative features, User interaction, Organization system, File upload, Advanced permissions
  - *Point potential*: 15+ points
- **Forum Platform**: Discussion boards with categories, threads, moderation tools, and user reputation systems.

  - *Suggested modules*: User interaction, Advanced permissions, Gamification, Content moderation AI, Advanced search
  - *Point potential*: 14+ points

- **Event Management Platform**: Create and manage events, RSVP system, calendar integration, and notifications.
  - *Suggested modules*: User interaction, Notification system, Organization system, Public API, Advanced search
  - *Point potential*: 14+ points

- **Learning Management System**: Courses, assignments, quizzes, progress tracking, and student-teacher interaction.
  - *Suggested modules*: User interaction, Organization system, Advanced permissions, File upload, Analytics dashboard
  - *Point potential*: 15+ points

V.4 Creative and Media Projects

These projects focus on content creation and sharing:

- **Music Streaming Platform**: Upload and stream music, playlists, recommendations, and social features.
  - *Suggested modules*: File upload, User interaction, Recommendation system, Advanced search, Analytics dashboard
  - *Point potential*: 15+ points

- **Video Sharing Platform**: Upload and watch videos, comments, likes, subscriptions, and recommendations.
  - *Suggested modules*: File upload, User interaction, Recommendation system, Content moderation AI, Advanced search
  - *Point potential*: 16+ points

- **Art Gallery**: Share artwork in galleries, with comments, likes, and artist profiles.
  - *Suggested modules*: File upload, User interaction, Image recognition, Advanced search, Custom design system
  - *Point potential*: 14+ points

- **Blogging Platform**: Create and publish blogs, with comments, tags, categories, and reader engagement.
  - *Suggested modules*: User interaction, SSR, Advanced search, Sentiment analysis, Multiple languages
  - *Point potential*: 14+ points

- **Recipe Sharing Platform**: Share recipes, ratings, comments, meal planning, and shopping lists.
  - *Suggested modules*: User interaction, File upload, Advanced search, Recommendation system, PWA
  - *Point potential*: 14+ points

V.5 Productivity and Tools Projects

These projects help users organize and manage their work:

- **Task Management System**: Projects, tasks, assignments, deadlines, team collaboration, and progress tracking.
  - *Suggested modules*: Organization system, User interaction, Real-time collaborative features, Notification system, Analytics dashboard
  - *Point potential*: 15+ points
- **Code Collaboration Platform**: Share code snippets, collaborative coding, version control, and discussions.
  - *Suggested modules*: User interaction, Real-time collaborative features, Public API, Advanced search, Custom design system
  - *Point potential*: 14+ points
- **Booking System**: Reserve resources (rooms, equipment, appointments), calendar, and notifications.
  - *Suggested modules*: User interaction, Organization system, Notification system, Public API, Advanced search
  - *Point potential*: 14+ points
- **Marketplace Platform**: Buy and sell items, with user ratings, messaging, payment integration, and search functionality.
  - *Suggested modules*: User interaction, File upload, Advanced search, Recommendation system, Public API
  - *Point potential*: 14+ points
- **Fitness Tracker**: Log workouts, track progress, challenges, leaderboards, and social features.
  - *Suggested modules*: User interaction, Gamification, Analytics dashboard, PWA, Data export/import
  - *Point potential*: 14+ points

V.6 Specialized Projects

These projects target specific niches or industries:

- **Real-time Trading Simulator**: Stock and cryptocurrency trading simulation with real-time data and portfolios.
  - *Suggested modules*: Real-time features, User interaction, Analytics dashboard, Public API, Advanced 3D graphics
  - *Point potential*: 15+ points
- **Language Learning Platform**: Lessons, exercises, progress tracking, and peer practice.
  - *Suggested modules*: User interaction, Gamification, Multiple languages, Voice integration, Analytics dashboard
  - *Point potential*: 15+ points
- **Pet Adoption Platform**: Browse pets, adoption process, user profiles, and messaging.
  - *Suggested modules*: User interaction, File upload, Advanced search, Organization system, Notification system
  - *Point potential*: 14+ points
- **Travel Planning Platform**: Plan trips, share itineraries, recommendations, and social features.
  - *Suggested modules*: User interaction, Real-time collaborative features, Recommendation system, Multiple languages, Advanced search
  - *Point potential*: 15+ points
- **Crowdfunding Platform**: Create campaigns, donations, updates, and community engagement.
  - *Suggested modules*: User interaction, File upload, Public API, Analytics dashboard, Notification system
  - *Point potential*: 14+ points

These are just ideas to inspire you. The key is to choose a project that:

- Interests your team and motivates everyone to work on it.
- Allows you to implement the required modules (14 points minimum).
- Demonstrates technical complexity and creativity.
- Can be realistically completed within the project timeline.
- Has coherent module combinations that work well together.

Discuss with your team, review the available modules, and choose wisely!

Chapter VI

Readme Requirements

A `README.md` file must be provided at the root of your Git repository. Its purpose is to allow anyone unfamiliar with the project (peers, staff, recruiters, etc.) to quickly understand what the project is about, how to run it, and where to find more information on the topic.
The `README.md` must include at least:

- The very first line must be italicized and read: *This project has been created as part of the 42 curriculum by <login1>[, <login2>[, <login3>[...]]].*
- A "**Description**" section that clearly presents the project, including its goal and a brief overview.
- An "**Instructions**" section containing any relevant information about compilation, installation, and/or execution.
- A "**Resources**" section listing classic references related to the topic (documentation, articles, tutorials, etc.), as well as a description of how AI was used — specifying for which tasks and which parts of the project.

➠ **Additional sections may be required depending on the project** (e.g., usage examples, feature list, technical choices, etc.).

Any required additions will be explicitly listed below.

- The "Description" section should also contain a clear name for the project and its key features.
- The "Instructions" section should mention all the needed prerequisites (software, tools, versions, configuration like .env setup, etc.), and step-by-step instructions to run the project.

Additional sections required for this activity:

- **Team Information**:
  For each team member mentioned at the top of the README.md, you must provide:
  - Assigned role(s): PO, PM, Tech Lead, Developers, etc.
  - Brief description of their responsibilities.
- **Project Management**:
  - How the team organized the work (task distribution, meetings, etc.).
  - Tools used for project management (GitHub Issues, Trello, etc.).
  - Communication channels used (Discord, Slack, etc.).
- **Technical Stack**:
  - Frontend technologies and frameworks used.
  - Backend technologies and frameworks used.
  - Database system and why it was chosen.
  - Any other significant technologies or libraries.
  - Justification for major technical choices.
- **Database Schema**:
  - Visual representation or description of the database structure.
  - Tables/collections and their relationships.
  - Key fields and data types.
- **Features List**:
  - Complete list of implemented features.
  - Which team member(s) worked on each feature.
  - Brief description of each feature's functionality.
- **Modules**:
  - List of all chosen modules (Major and Minor).
  - Point calculation (Major = 2pts, Minor = 1pt).
  - **Justification for each module choice**, especially for custom "Modules of choice".
  - How each module was implemented.
  - Which team member(s) worked on each module.
- **Individual Contributions**:

- Detailed breakdown of what each team member contributed.
- Specific features, modules, or components implemented by each person.
- Any challenges faced and how they were overcome.

Any other useful or relevant information is welcome (usage documentation, known limitations, license, credits, etc.).

The README.md is a critical part of your project evaluation. It should be:

- Clear and well-organized.
- Complete with all required sections.
- Professional and easy to read.
- Honest about contributions and challenges.

A poor or incomplete README can negatively impact your evaluation.

Your README must be written in English.

Chapter VII

Bonus part

The bonus part will be considered only if all required modules have been implemented corresponding to the minimum of 14 mandatory points.
Each additional module implemented beyond the required 14 points may be considered as a bonus.
For each extra module:

- It must be fully functional
- It must meet the module requirements description
- It must add real value to the project
- It must include a proper justification in the README

Each validated extra module will be taken into account during the review as follows:

- Major modules: 2 points each
- Minor modules: 1 point each

You can have a maximum of 5 points (e.g., 5 minor modules, or 2 major modules + 1 minor module).

Chapter VIII

Submission and peer-evaluation

Submit your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the evaluation. Double-check file names.

We highly recommend that you discuss your ideas with your team and peers before starting to work on the project.

During the evaluation, a brief **modification of the project** may occasionally be requested. This could involve a minor behaviour change, a few lines of code to write or rewrite, or an easy-to-add feature.

While this step may **not be applicable to every project**, you must be prepared for it if it is mentioned in the evaluation guidelines.

This step is meant to verify your actual understanding of a specific part of the project. The modification can be performed in any development environment you choose (e.g., your usual setup), and it should be feasible within a few minutes — unless a specific time frame is defined as part of the evaluation.
You can, for example, be asked to make a small update to a function or script, modify a display, or adjust a data structure to store new information, etc.

The details (scope, target, etc.) will be specified in the **evaluation guidelines** and may vary from one evaluation to another for the same project.