Syllabus for the Course «Big Data Engineering»

Course volume: 5 credit units (180 hours)
Semesters: 2–3
Assessment format:

- Semester 2: pass/fail assessment / defence of semester project results;
- Semester 3: exam, coursework.

Contact hours:

- Semester 2: lectures — 22 hours; seminars / digital labs — 23 hours;
- Semester 3: lectures — 16 hours; seminars / digital labs — 16 hours.

Independent study:

- Semester 2 — 27 hours;
- Semester 3 — 40 hours;
- exam preparation — 32 hours.

Delivery format: AI-augmented learning (learning with the use of intelligent assistants).

1. General characteristics of the course

The course *Big Data Engineering* belongs to the professional module of the Master's programme *AI-based IT Engineering* (specialisation 09.04.02 *Information Systems and Technologies*).

The course is delivered in the second and third semesters and serves as a core course in the *Big Data (Big Data Technologies)* track. Within the Master's programme structure, it occupies an intermediate position between the overview and methodological courses of the first semester and more specialised courses in the track related to stream data processing, lakehouse architectures and adjacent areas.

The course has a theoretical and applied engineering character. Its goal is not to isolate the mastery of individual technologies, but to develop students' systemic understanding of how big data processing systems are designed, implemented and maintained under real engineering constraints.

The course treats big data not only as large volumes of information, but as a specific class of engineering tasks, where data origin, ingestion mode, storage architecture, processing logic, quality requirements, observability and reproducibility form a unified project system.

A distinctive feature of the course is the end-to-end use of large language models (LLMs) as a tool for engineering analysis, generation of architectural alternatives, project reflection and support of students' independent work.

2. Place of the course in the educational programme structure

The course serves as the foundational, system-forming course in the Big Data Technologies track and introduces the engineering logic of working with data at the level of architecture, pipeline and platform solutions.

It develops a basic understanding of:

- the role of data engineering in digital systems;
- data origin and its impact on architecture;
- the lifecycle of a data project;
- principles of data storage, batch and distributed processing;
- requirements for quality, reliability, observability and maintainability of a data pipeline.

The course is closely linked to previous courses:

- algorithms and data structures;
- algorithmisation and programming languages;
- data structures and databases;
- fundamentals of artificial intelligence and large linguistic models: prompt and context engineering;
- infrastructure and architecture of information systems.

The course provides a methodological and engineering foundation for subsequent courses in the track and related tracks:

- technologies for continuous stream data processing;
- lakehouse data management technologies;
- containerisation and Kubernetes;
- architecture of high-load information systems;
- neural networks, machine learning and deep learning;
- digital engineering projects and courseworks.

Thus, the course ensures the transition from general software and architectural thinking to engineering thinking within the data domain.

3. Learning objectives

The objective of the course is to develop students' systemic understanding of big data engineering as a field of designing and maintaining data processing platforms and pipelines.

Special attention is given to:

- understanding data engineering as a distinct engineering layer of digital systems;
- analysing data origin and its influence on architectural decisions;

- mastering principles of data pipeline and data platform design;
- developing the ability to choose architectural solutions depending on data type, ingestion mode and environmental constraints;
- building skills in engineering decomposition and design of data solutions;
- fostering a culture of using LLMs in engineering analysis and project activities.

4. Course tasks

The main tasks of the course are:

- to develop a systemic view of the place and role of data engineering in the architecture of digital systems;
- to study types of data origin and strategies for data acquisition;
- to master principles of designing data platform and data pipeline architecture;
- to learn data storage models and basic approaches to big data processing;
- to master ETL/ELT principles, batch and distributed processing;
- to study methods for ensuring data quality;
- to analyse principles of orchestration, reproducibility, reliability and observability of data pipelines;
- to build skills in project decomposition of engineering data solutions;
- to prepare for completing a coursework on the Big Data track topic;
- to foster a culture of conscious use of LLMs in engineering design and analysis.

5. Expected learning outcomes

Upon completion of the course, students should:

Know:

- the place of data engineering in the structure of digital systems;
- main types of data origin and scenarios for data acquisition;
- architectural structure of data processing platforms;
- principles of big data storage and data format selection;
- fundamentals of batch and distributed data processing;
- principles of ETL/ELT and data pipeline design;
- methods for data quality control;
- fundamentals of orchestration, reliability, reproducibility and observability of data pipelines;
- limitations and risks of using LLMs in data engineering.

Be able to:

- analyse an engineering task from the perspective of data and its origin;
- identify architectural constraints of a data project;
- design a high-level architecture of a data platform and data pipeline;
- select data storage and processing methods depending on task characteristics;

- formulate requirements for data and pipeline quality;
- justify architectural decisions with sound arguments;
- use LLMs for engineering analysis, generating alternatives and project reflection.

Have skills in:

- engineering analysis of data systems;
- project decomposition of data solutions;
- comparative analysis of architectural options;
- culture of engineering argumentation;
- conscious and critical use of intelligent assistants.

6. Methodological concept of the course

6.1. Big data engineering as an architectural layer of a digital system
A modern digital system includes not only application logic and an infrastructure platform, but also a dedicated data workflow. This workflow covers data origin, acquisition methods, storage architecture, processing, result publication, quality control and data lifecycle management.

Therefore, big data engineering is treated in the course not as a set of isolated technologies, but as a full-fledged architectural layer of a digital system.

6.2. Methodological principle of the course
The methodological logic of the course is expressed by the principle:
«Data origin and lifecycle shape the architecture of data processing».

Architectural decisions in data engineering are always made under constraints set by:

- type of data origin;
- ingestion mode;
- structure and variability of sources;
- quality and result delivery time requirements;
- execution environment constraints;
- maintenance and evolution requirements.

Thus, designing a pipeline and data platform is about managing trade-offs between result requirements and the real properties of data and environment.

6.3. Engineering thinking through analysis of data origin and architectural alternatives
Within the course, data solutions are treated not as standard templates, but as a space of alternatives. Students analyse differences between:

- transactional, event-based, API and web sources;
- regular loading and changing source contours;
- centralised and distributed processing;
- simple pipeline and more mature data platform;
- batch logic and architectures requiring further transition to stream processing.

The goal of the course is to build the ability to select a solution depending on the real characteristics of the task, not on technology popularity.

6.4. Data pipeline and lifecycle of an engineering project
The data pipeline is treated in the course as part of the lifecycle of an engineering project. The choice of pipeline architecture affects:

- result quality and reproducibility;
- solution resilience to failures and source changes;
- complexity of maintenance;
- readiness for scaling;
- ability to integrate with analytical and AI components later.

That is why the course is structured as a two-semester programme: in the second semester, students master basic engineering logic and design a solution, and in the third — they develop it to a more mature architectural level in the form of a coursework.

7. Use of LLMs in the educational process

Within the course, large language models are used as a tool to support engineering analysis and project activities.

LLMs are applied in the following modes:

- LLM as a reference. The model is used to explain data engineering terms, architectural concepts, processing patterns, source types and basic technological mechanisms.
- LLM as an intelligent assistant. The LLM helps the student:
    - to analyse the formulation of a data task;
    - to identify architectural constraints;
    - to generate pipeline and architectural solution variants;
    - to formulate choice arguments.
- LLM as an analytical opponent. In seminars and project work, the LLM is used to:
    - identify weak points in an architectural solution;
    - generate alternative options;
    - pose clarifying questions on quality, reliability, scalability and observability.
- LLM as a feedback tool. During independent work, the model can be used to:
    - perform initial analysis of project notes;
    - detect incomplete argumentation;
    - verify the logic of project decomposition;
    - prepare questions for subsequent discussion with the instructor.

The final assessment is formed by the instructor.

Critical stance: domain-specific risks of LLMs in data engineering

Using LLMs in data engineering requires caution, as the model may confidently produce outwardly convincing but engineering-incomplete or incorrect solutions.

Typical LLM errors in the context of this course include:

- recommending pipeline architecture without analysing data origin and source constraints;
- mixing different data processing modes without considering time, quality and maintenance cost requirements;
- reproducing popular templates instead of analysing the specific task;
- ignoring source data instability and quality issues;
- substituting engineering choice with listing «pros and cons» without an explicit conclusion;
- uncritical use of outdated or overly general technological concepts.

Mandatory elements for verifying LLM results:

- check whether the model accounts for the type of data origin and ingestion mode;
- verify whether the proposed solution is consistent with task constraints;
- check whether the model distinguishes between engineering and general descriptive arguments;
- verify whether an explicit choice of architectural variant is given, not just a list of possible approaches;
- document in independent work which LLM suggestions were accepted, which were rejected and why.

Key principle of the course:
The LLM operates with the form of engineering reasoning. The student ensures the engineering content and responsibility for the correctness of the solution.

8. Educational technologies

The course applies the AI-augmented learning model, combining traditional learning forms with the use of intelligent assistants.

The educational process includes:

- lecture sessions;
- seminar sessions;
- digital laboratory work;
- practicum on data origin;
- project-based learning;
- completion of an end-to-end project in semester 2;
- completion of a coursework in semester 3;
- use of LLMs as a tool for analysis and engineering reflection.

9. Organisation of students' independent work

Independent work is aimed at developing skills in engineering analysis and design of data processing systems.

Main forms of independent work:

- analysis of architecture and data origin;
- preparation of project notes and comparative tables;
- development of pipeline diagrams;
- formulation of data quality rules;
- analysis of architectural constraints;
- preparation of materials for the end-to-end project;
- completion of coursework;
- work with LLM assistants with mandatory engineering verification of results.

A distinctive feature of the course is that independent work follows two tracks:

- regular thematic assignments linked to current course sections;
- stages of the end-to-end project / coursework, ensuring gradual formation of a holistic engineering solution.

10. Forms of assessment

The course uses the following forms of assessment:

Ongoing assessment:

- discussion of engineering case studies;
- analysis of data origin and architectural constraints;
- completion and discussion of digital laboratory work;
- evaluation of students' analytical and project materials;
- quick checks of knowledge and skills on key topics;
- assessment of stages of the end-to-end project and coursework.

Intermediate assessment:

- Semester 2: pass/fail assessment / defence of semester project results, including:
    - analysis of the subject formulation and data origin;
    - presentation of pipeline architecture;
    - justification of adopted engineering solutions;
    - description of quality rules, reliability and project development logic.

Final assessment:

- Semester 3: exam and coursework.
    - The exam tests:
        - understanding of the engineering logic of the data project lifecycle;
        - ability to analyse architectural situations;
        - skill in distinguishing alternatives and engineering trade-offs;
        - ability to justify solution choice with sound arguments.
    - Coursework involves presenting the project solution as a holistic engineering system.

The LLM can be used for:

- generating case study variants;
- initial analysis of written work;
- preparing project materials;
- posing questions for discussion.

Final evaluation is carried out by the instructor.

Calendar-thematic plan of the course *Big Data Engineering*

Semester 2 (spring)

- 2 credit units;
- theory — 22 hours, seminars/digital labs — 23 hours, independent work — 27 hours.

Week	Content	Lect. (h)	Sem./Lab. (h)	IW (h)
1	**Lecture.** Introduction to the course as part of the engineering Master's programme: the place of Data Engineering in digital systems, differences from Data Science, analytics and applied development. Discussion of the lifecycle of an engineering data task: from problem statement and requirements to solution operation. **Seminar/discussion.** Analysis of contrasting cases: transactional data, logs, telemetry, media data. Students discuss where a Big Data	2	2	2

Week	Content	Lect. (h)	Sem./Lab. (h)	IW (h)
	approach is truly needed and where traditional storage and processing tools suffice. **Quick knowledge check.** Assesses understanding of the data engineer's role, distinguishing Big Data tasks from regular storage/analytics tasks, ability to justify why large data volume does not guarantee the need for Big Data infrastructure. **Indep. work — regular HW.** Brief case analysis answering whether Big Data infrastructure is required and why. **Indep. work — project.** The end-to-end project is not yet selected; the student outlines 2–3 possible subject areas of interest.			
2	**Lecture.** Typical data platform architecture: sources, ingest layer, storage, processing, result publication, monitoring, quality control. Emphasis on architecture as a way to reconcile requirements and constraints, not just a set of technologies. **Seminar/digital lab.** Analysis of architectural schemes for analytical dashboard, log analytics, IoT and transactional analytics. Students identify mandatory and optional blocks and explain what determines their necessity. **Quick knowledge check.** Tests ability to identify data platform components and relate architectural blocks to engineering tasks. **Indep. work — regular HW.** Build a high-level architectural diagram for a given case with brief explanation. **Indep. work — project.** Form a preliminary list of possible data sources for areas of interest.	2	2	2
3	**Lecture.** Introduction of data origin as a key factor of architecture. Discussion of transactional sources, event streams, APIs and regular data dumps; analysis of how data origin affects processing mode, update frequency and quality requirements. **Seminar/practicum on data origin, part 1.** Mini-case on transactional data and mini-case on event data: analysis of structure, possible consistency issues,	2	2	2

Week	Content	Lect. (h)	Sem./Lab. (h)	IW (h)
	transformation and publication methods. **Quick knowledge check.** Tests ability to distinguish transactional and event data and understanding of their engineering constraints. **Indep. work — regular HW.** Describe data acquisition method for a given case: format, frequency, quality risks, possible information loss points. **Indep. work — project.** Form a preliminary list of possible data sources for the future project.			
4	**Lecture.** Continuation of data origin topic for more complex scenarios: web scraping, regular collection from external sources, adversarial sources, changing source scope. Discussion of technical, organisational and legal constraints of such data acquisition methods. **Seminar/practicum on data origin, part 2.** Analysis of API source and web source: selection of data acquisition strategy, assessment of solution stability, analysis of access restrictions and structural instability. **Quick knowledge check.** Tests ability to choose data acquisition strategy and understanding of rate limit risks, structural instability and data incompleteness. **Indep. work — regular HW.** Comparative analysis of several data acquisition strategies and their impact on pipeline architecture. **Indep. work — project.** Describe possible data sources for the future project and preliminary conclusion on their architectural implications.	2	2	2
5	**Lecture.** After the practicum, introduction of the lifecycle of an engineering data project: problem statement, requirements, source analysis, architectural design, pipeline development, quality control, operation and evolution. Separate introduction of architectural forks concept. **Seminar.** Presentation of end-to-end project options differing by subject area and data/technology constraints. Students compare options by engineering essence, not just by	1	1	2

Week	Content	Lect. (h)	Sem./Lab. (h)	IW (h)
	name. **Quick knowledge check.** Tests understanding of project lifecycle and ability to identify specific option constraints. **Indep. work — regular HW.** Select a project option and provide brief description of task, result and constraints. **Indep. work — project.** Fix selected project, formulate goal and scope, describe data origin.			
6	**Lecture.** Decomposition of an engineering project into stages and components: transition from general idea to sequence of steps for collection, preparation, storage, processing, quality control and result delivery. **Seminar/lab.** Development of draft project implementation plan, discussion of stage dependencies, critical points and architectural forks. **Quick knowledge check.** Tests ability to decompose a project and identify stage dependencies. **Indep. work — regular HW.** Prepare a structural pipeline diagram with explanation for each stage. **Indep. work — project.** First mandatory assignment: document «Project Implementation Plan» with lifecycle stages.	1	2	2
7	**Lecture.** Principles of data storage in Big Data systems: local, centralised and distributed storage; file, table and object approaches. Discussion of how storage method affects processing and maintenance. **Seminar/digital lab.** Comparison of identical data in different formats and structures, discussion of common mistakes in format and storage model selection for a task. **Quick knowledge check.** Tests understanding of differences between storage models and ability to select format for a specific scenario. **Indep. work — regular HW.** Comparative note on several formats and storage models for different pipeline layers. **Indep. work — project.** Select storage model for	2	2	2

Week	Content	Lect. (h)	Sem./Lab. (h)	IW (h)
	the project and justify data formats and structures.			
8	**Lecture.** Batch processing as the basic mode of data pipeline operation. Discussion of ETL and ELT, processing step, launch window, idempotency, re-run and result reproducibility. **Seminar/digital lab.** Construction of educational batch processing scenario: reading, cleaning, transformation, aggregation and result publication. **Quick knowledge check.** Tests ability to distinguish ETL from ELT, understanding of idempotency and batch processing logic. **Indep. work — regular HW.** Design an ETL/ELT scheme for an educational case, indicating quality control and error handling points. **Indep. work — project.** Design a batch processing step for own project.	2	2	2
9	**Lecture.** Idea of distributed data processing: why computations are moved to a cluster, differences between heavy and light operations, why join, shuffle and aggregations become architecturally significant. **Seminar/digital lab.** Analysis of distributed data processing scenario and identification of most resource-intensive operations and why. **Quick knowledge check.** Tests understanding of reasons for distributed computations and ability to identify heavy processing operations. **Indep. work — regular HW.** Analyse a given scenario, identify bottlenecks and their causes. **Indep. work — project.** Describe project steps requiring distributed processing and justify them.	2	2	2
10	**Lecture.** Data quality, cleaning and validation. Discussion of missing values, duplicates, incorrect types, outliers, inconsistent values and business rule violations; explanation of differences between technical validation, substantive checking and cleaning. **Seminar/lab.**	2	2	2

Week	Content	Lect. (h)	Sem./Lab. (h)	IW (h)
	Development of data quality rules for an educational dataset and discussion of which defects are automatically corrected and which should trigger an incident. **Quick knowledge check.** Tests ability to classify quality defects and choose reaction to them. **Indep. work — regular HW.** Formulate a set of quality rules for a given dataset. **Indep. work — project.** Develop own set of data quality checks for the project.			
11	**Lecture.** Pipeline orchestration and processing execution management: step dependencies, schedule, re-runs, status control, logging. Orchestration as means of reproducibility and manageability, not just launch schedule. **Seminar/digital lab.** Design of DAG for educational pipeline with analysis of success, failure and re-execution conditions. **Quick knowledge check.** Tests understanding of orchestration role and ability to identify step dependencies. **Indep. work — regular HW.** Prepare DAG diagram and describe conditions for successful/unsuccessful stage completion. **Indep. work — project.** Design orchestration scheme for own project.	2	2	2
12	**Lecture.** Reliability, maintenance and reproducibility of data pipeline: logging, version control, traceability, failure diagnostics, re-execution and change management. **Seminar/lab/discussion.** Analysis of typical incidents: source schema change, incomplete loading, partial processing, result quality violation. **Quick knowledge check.** Tests understanding of result reproducibility and ability to suggest reliability improvement measures. **Indep. work — regular HW.** Prepare diagnostics and recovery plan for a given failure scenario. **Indep. work — project.** Extend project with reliability section: logging, versions, control points and re-run.	2	2	2

Week	Content	Lect. (h)	Sem./Lab. (h)	IW (h)
13	**Lecture.** Basic performance and processing optimisation issues: redundant steps, heavy joins, repeated data reading, ill-advised intermediate layers. Emphasis on the trade-off between speed and maintainability of the solution. **Seminar/lab.** Comparison of a «naive» and an «improved» pipeline variant, discussion of how acceleration is achieved and what the cost of architecture complexity is. **Quick knowledge check.** Tests ability to detect bottlenecks and explain optimisation trade-offs. **Indep. work — regular HW.** Prepare suggestions for improving a given pipeline with assessment of their feasibility. **Indep. work — project.** Primary audit of own project: likely bottlenecks and basic optimisation measures.	1	1	1
14	**Lecture.** Overview topics: streaming processing, event-driven approach, data lake and lakehouse. The material is conceptual and serves as a link to subsequent disciplines in the track, not as an in-depth study of these technologies. **Seminar/discussion.** Comparison of batch, near real-time, and lake/lakehouse logic for one case; discussion of which requirements actually lead to architecture complexity. **Quick knowledge check.** Tests understanding of differences between batch and streaming, and conceptual distinction between dashboard, lake and lakehouse logic. **Indep. work — regular HW.** Comparative analysis of several architectural approaches by criteria of complexity, cost and applicability. **Indep. work — project.** Clarification of the project's architectural positioning and directions for its possible development.	1	1	1
15	**Lecture.** No special lecture; the week is devoted to integrating the project into a holistic engineering system. **Seminar/pre-defence.** An	0	0	1

Week	Content	Lect. (h)	Sem./Lab. (h)	IW (h)
	engineering seminar-pre-defence is held, where students present the project architecture, implementation stages, constraints, quality rules and decisions made. **Quick knowledge check.** Tests the ability to present the project holistically and argue the logic of decisions. **Indep. work — regular HW.** Revision of the report and presentation based on pre-defence feedback. **Indep. work — project.** Fixing weak points of the project, clarifying constraints, quality rules and pipeline stages.			
16 (session)	**Final assessment.** During the session, the final defence of semester 2 results is held. Not only the presence of a technical solution is assessed, but also understanding of data origin, architectural logic of the pipeline, quality rules, reliability and project development prospects. **Indep. work.** Finalise the explanatory note, architectural diagram, data source description, implementation plan, pipeline diagram and quality control rules.	0	0	2

Semester 3 (autumn)

- 3 credit units;
- theory — 16 hours, seminars/digital labs — 16 hours, independent work — 40 hours;
- additionally: coursework; exam preparation — 32 hours; exam — 4 hours.

Week	Content	Lect. (h)	Sem./ Lab. (h)	IW (h)
1	**Lecture.** At the start of the semester, the project is rethought not as a learning pipeline but as a data platform architecture. Functional contour, data	2	2	3

Week	Content	Lect. (h)	Sem./ Lab. (h)	IW (h)
	contour, execution contour, quality, observability and operation are considered as a unified engineering system. **Seminar/project audit.** Students analyse semester 2 results and identify which architectural layers and requirements are missing for transition to coursework. **Quick knowledge check.** Tests understanding of the differences between a learning pipeline and a mature platform architecture, and ability to identify missing project elements. **Indep. work — regular HW.** Analytical note on what improvements the project needs to become the basis for coursework. **Indep. work — coursework.** Update the architectural diagram and formulate expanded project goals.			
2	**Lecture.** Study of scalability of data processing systems: growth in data volume, number of sources, update frequency and user load. Scalability is considered as an architectural property, not as a late add-on. **Seminar/scenario analysis.** Students consider how their solution will change with load growth and which architectural adaptation measures are possible. **Quick knowledge check.** Tests understanding of scalability and ability to link growth patterns with architectural changes. **Indep. work — regular HW.** Describe several load growth scenarios and possible adaptation measures. **Indep. work — coursework.** Prepare the «Scalability and growth limits» section.	2	2	3
3	**Lecture.** High-load data processing systems: delays, queue overflows, load imbalance, storage and computation bottlenecks. Focus on engineering thinking under load growth and instability. **Seminar/project review.** Students analyse overload scenarios and compare several ways to improve solution resilience. **Quick knowledge check.** Tests ability to identify causes of	2	2	3

Week	Content	Lect. (h)	Sem./ Lab. (h)	IW (h)
	degradation under load and propose several alternative solutions. **Indep. work — regular HW.** Brief analysis of the project's resilience to load growth and list of likely failure zones. **Indep. work — coursework.** Describe high-load risks and architectural forks of the project.			
4	**Lecture.** Topic of the week: observability of data systems — monitoring, metrics, logging, data flow tracing and deviation diagnostics. Discussion of why a system is not engineering mature without observability. **Seminar/design of observability.** Students select metrics, logs, signals and diagnostic points for their project. **Quick knowledge check.** Tests distinction between logging and monitoring, ability to select critical observability parameters. **Indep. work — regular HW.** Prepare a list of metrics and observable parameters for the pipeline. **Indep. work — coursework.** Add the observability and diagnostics section to the work.	2	2	3
5	**Lecture.** Metadata, data cataloguing and documentation: origin, schema, version, quality, lineage and the role of metadata in platform maintenance. **Seminar/documentation work.** Students describe the project's metadata: sources, schemas, formats, transformations, dependencies between entities and publications. **Quick knowledge check.** Tests understanding of metadata role and ability to distinguish technical and substantive metadata. **Indep. work — regular HW.** Compile a data passport or passport of one pipeline layer. **Indep. work — coursework.** Formulate the metadata and documentation structure of the project.	2	2	3
6	**Lecture.** Data governance in engineering terms: data usage rules, quality responsibility, access, versioning and change management. Emphasis on	2	2	3

Week	Content	Lect. (h)	Sem./ Lab. (h)	IW (h)
	governance impact on practical architecture resilience. **Seminar/case study.** Analysis of situations where lack of access rules, responsibility and change control destroys solution resilience. **Quick knowledge check.** Tests understanding of governance role and ability to select the minimum necessary set of data management rules. **Indep. work — regular HW.** Describe a basic set of data management rules for own project. **Indep. work — coursework.** Add a block on change management and data responsibility.			
7	**Lecture.** Integration of Data Engineering and ML contours: data preparation for models, feature pipeline, difference between research and industrial contours. This is not a separate MLOps course, but a conceptual link to the AI direction of the programme. **Seminar/scenario analysis.** Students discuss whether their pipeline can be used as a data provider for AI/ML components and what enhancements are needed. **Quick knowledge check.** Tests understanding of differences between analytical and ML contours and ability to identify additional data requirements. **Indep. work — regular HW.** Analyse the possibility of using the project as a data provider for an ML task. **Indep. work — coursework.** Formulate a possible project extension towards AI/ML or justify the infeasibility of such an extension.	2	2	3
8	**Lecture.** Summary of architectural deepening: discussion of engineering project maturity criteria, requirement completeness, architectural consistency, reproducibility and maintainability. **Seminar/project seminar.** Mid-term defence of coursework concepts with presentation of architecture, constraints, observability, risks and further work plan. **Quick knowledge check.** Tests ability to present project architecture as a holistic system and argue responses to comments. **Indep.**	2	2	4

Week	Content	Lect. (h)	Sem./ Lab. (h)	IW (h)
	work — regular HW. Revise the project based on seminar feedback. **Indep. work — coursework.** Fix the coursework structure and update the completion plan.			
9	**Targeted consultation theory.** Short explanations on problematic topics of specific projects: architectural forks, performance, reliability, integration with external systems. **Project seminar / design review.** Discussion of technical solutions and their alternatives. **Quick knowledge check.** Tests maturity of engineering thinking: ability to argue choice and see risks. **Indep. work — regular HW.** Revise coursework sections based on consultation feedback. **Indep. work — coursework.** Systematic work on project implementation and documentation.	0	0	3
10	**Targeted consultation theory.** Continuation of support for individual project trajectories. **Project seminar / consultations.** Discussion of architecture, data quality, reliability and documentation. **Quick knowledge check.** Tests ability to defend architectural decisions and adjust them when new constraints emerge. **Indep. work — regular HW.** Prepare and revise next block of coursework materials. **Indep. work — coursework.** Continuation of systematic work on the coursework: implementation, analysis, adjustment.	0	0	3
11	**Targeted consultation theory.** Brief explanations on specific engineering issues of projects. **Project seminar.** Discussion of implementation, architectural changes and justification of solution variants. **Quick knowledge check.** Tests ability to relate implementation to project requirements and constraints. **Indep. work — regular HW.** Revise problematic sections of the coursework. **Indep.**	0	0	3

Week	Content	Lect. (h)	Sem./ Lab. (h)	IW (h)
	work — coursework. Continuation of systematic project work and preparation for pre-defence.			
12	**Targeted consultation theory.** Final consultation support before pre-defence. **Project seminar.** Integration of the project into a holistic system and verification of coursework completeness. **Quick knowledge check.** Tests readiness of the project for pre-defence and integrity of engineering logic in the solution. **Indep. work — regular HW.** Final revision of main coursework sections. **Indep. work — coursework.** Completion of main text body and materials.	0	0	3
13	**Lecture / установочное занятие к предзащите.** Introduction to pre-defence: discussion of requirements for the final engineering presentation, logic of demonstrating architecture, constraints and proof of solution validity. **Seminar / pre-defence of courseworks.** Identification of weak points in logic, evidence, completeness of architectural description and engineering argumentation. **Quick knowledge check.** Tests ability to logically defend an engineering solution and perceive comments as tasks for revision. **Indep. work — regular HW.** Revision of coursework and preparation of final materials package. **Indep. work — coursework.** Addressing feedback and finalising text, diagrams, tables and results.	0	0	3
14 (session)	**Coursework defence.** During the session period, the student presents the project as an engineering system: data origin, requirements, architecture, reliability, observability and development prospects. **Indep. work.** Finalise presentation, appendices and final materials package.	0	0	3

Week	Content	Lect. (h)	Sem./ Lab. (h)	IW (h)
15 (session)	**Exam preparation.** This week is dedicated to systematising all course material: data origin, architectures, ETL/ELT, distributed processing, quality, orchestration, reliability, observability, metadata, governance. **Indep. work.** Review theory, analyse engineering cases, prepare answers on key topics and use own project as an integrative example of applying studied approaches.	0	0	8
16 (session)	**Exam.** Tests not only knowledge of concepts and technologies, but primarily the ability to think in terms of data project lifecycle logic, distinguish architectural situations, explain engineering trade-offs and argue evaluation of solutions. **Final session load.** Separately accounted: 32 hours of independent exam preparation and 4 hours for exam delivery/defence.	0	0	