## Principali informazioni sull'insegnamento

| | |
|---|---|
| Denominazione dell'insegnamento | **Ingegneria del Software** |
| Corso di studio | Informatica e Comunicazione Digitale |
| Anno Accademico | 2024/25 |
| Crediti formativi universitari (CFU) / European Credit Transfer and Accumulation System (ECTS) | 9 CFU |
| Settore Scientifico Disciplinare | INF/01 - Informatica |
| Lingua di erogazione | Italiano |
| Anno di corso | Secondo |
| Periodo di erogazione | 2° semestre, le date esatte sono riportate nel manifesto/regolamento |
| Obbligo di frequenza | La frequenza è fortemente raccomandata |
| Sito web del corso di studio | https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea/informatica-icd-taranto-270/laurea-triennale-in-informatica-e-comunicazione-digitale-sede-di-taranto-d.m.-270 |

## Docente/i



| | |
|---|---|
| Nome e cognome | Azzurra Ragone |
| Indirizzo mail | azzurra.ragone@uniba.it |
| Telefono | +39 080-5443289 (int. 3289) |
| Sede | Dipartimento di Informatica, Via Orabona 4, 70125, Bari. Stanza n.616, VI piano. |
| Sede virtuale | Piattaforma E-LEARNING - https://elearning.di.uniba.it/ |
| Sito web del docente | https://www.uniba.it/it/docenti/ragone-azzurra |
| Ricevimento (giorni, orari e modalità, es. su appuntamento) | (da confermare)<br>mercoledì 12.30 - 13-30 (previo appuntamento) |



## Syllabus

| | |
|---|---|
| **Obiettivi formativi** | L'insegnamento di Ingegneria del Software riguarda l'analisi, il progetto e la realizzazione di sistemi software applicando i principi dell'Ingegneria del Software, nonché metodologie e tecniche di sviluppo di sistemi software. Ciò include la progettazione di una applicazione d'impresa a partire dalla raccolta dei requisiti. |
| **Prerequisiti** | Lo studente deve avere familiarità con almeno un linguaggio di programmazione e con le strutture di dati fondamentali.<br>Le seguenti conoscenze preliminari facilitano ed accelerano la comprensione degli argomenti dell'insegnamento:<br>• da Programmazione: Linguaggi imperativi, capacità di sviluppo di programmi in un linguaggio di programmazione (es. C);<br>• da Linguaggi di Programmazione: comprensione della relazione tra problemi, algoritmi, linguaggi formali e linguaggi di programmazione; sintassi e semantica di un linguaggio di programmazione;<br>• da Laboratorio di Informatica: Algoritmi fondamentali, Progettazione modulare; |
| **Contenuti di insegnamento (Programma)** | **Introduzione all'Ingegneria del Software (ore 9 + 1 esercitazione)**<br>- Visione d'insieme<br>- I tipi di prodotti software<br>- Processi di sviluppo software<br>- Qualità dei prodotti<br>- Problemi dell'ingegneria del software<br><br>**Principi dell'Ingegneria del Software (ore 6)**<br>- Applicabilità dei principi |

| | |
|---|---|
| | - Rigore e formalità, Separazione degli interessi, Modularità, Astrazione, Generalità, Incrementalità<br><br>**Analisi dei requisiti (ore 8 + 2 esercitazione)**<br>- Concetti generali<br>- Specifiche dei Requisiti<br>- Specifiche Software<br><br>**Processi Agili (ore 7)**<br>- Sviluppo Agile del Software<br>- Metodologia SCRUM<br><br>**Progetto Software (ore 4 + 2 esercitazione)**<br>- Concetti Generali<br>- Elementi di Base sui Processi<br>- Linee guida di progetto (Information Hiding)<br>- Processo di progettazione SW<br><br>**Linguaggio di modellazione di un sistema software – UML (ore 10 + 3 esercitazione)**<br>- Overview<br>- Diagramma dei casi d'uso: casi d'uso, scenari, relazioni<br>- Diagramma delle classi: classi, oggetti, relazioni<br>- Diagramma di sequenza<br>- Diagramma delle componenti<br>- Diagramma di deployment<br>- Stereotipi UML: Approccio BCE (Boundary – Control – Entity)<br>- Esempi di modellazione UML<br><br>**Stili Architetturali (ore 4 + 1 esercitazione)**<br>- Principi Generali<br>- Stili Architetturali<br>- Object Oriented<br><br>**Design Pattern (ore 3)**<br>- Pattern di creazione<br>- Pattern strutturali<br>- Pattern comportamentali<br><br>**Strumenti di Supporto allo sviluppo (ore 2 + 1 esercitazione)**<br>- Application Lifecycle Management (ALM)<br>- Configuration Management<br><br>**Caso di studio (ore 3 + 5 esercitazione)**<br>- Introduzione caso di studio<br>- Verifica dei requisiti/user stories caso di studio<br>- Esempi e best practices<br>- Progettazione caso di studio |
| **Testi di riferimento** | ● Carlo Ghezzi, Medhi Jazayeri, Dino Mandrioli "Ingegneria del Software - Fondamenti e Principi, 2a edizione" Pearson Prentice Hall, 2004.<br>○ Capitolo 1: Ingegneria del Software: visione d'insieme<br>○ Capitolo 2: Il software: natura e qualità<br>○ Capitolo 3: Principi dell'ingegneria del software<br>● Ian Sommerville "Ingegneria del Software", 10a ed. Pearson, 2017<br>○ Capitolo 2: Processi Software<br>○ Capitolo 3: Sviluppo Agile del Software<br>○ Capitolo 5: Modelli di sistema<br>○ Capitolo 6: Progettazione architetturale<br>○ Capitolo 7: Progettazione e implementazione |

| | |
|---|---|
| | • Martin Fowler "UML distilled. Guida rapida al linguaggio di modellazione standard" (4 ed.). Pearson Addison Wesley, 2010.<br>  ○ Capitolo 1: Introduzione<br>  ○ Capitolo 3: Diagramma delle classi: concetti fondamentali<br>  ○ Capitolo 4: Diagramma di sequenza<br>  ○ Capitolo 5: Diagramma delle classi: concetti avanzati<br>  ○ Capitolo 8: Diagramma di deployment<br>  ○ Capitolo 9: Casi d'uso<br>  ○ Capitolo 14: Diagramma delle componenti<br><br>Gli studenti che lo desiderano possono ottenere i testi in prestito dalla Biblioteca. Può convenire verificarne la disponibilità mediante il Sistema Bibliotecario di Ateneo https://opac.uniba.it/easyweb/w8018/index.php? e contattare la biblioteca per concordare il prestito. |
| **Note ai testi di riferimento** | I testi di riferimento sono integrati con slide, dispense del docente e altro materiale didattico messi a disposizione degli studenti sulla piattaforma di e-learning usata dal CdS.<br><br>Testi consigliati per approfondire specifici argomenti:<br>• Jim Arlow, Ila Neustadt "UML 2 e Unified Process – Analisi e progettazione Object-Oriented, 2a edizione", McGraw-Hill, 2014 (per diagrammi UML)<br>• Steven John Metsker, "Design pattern in Java: manuale pratico". Pearson: Addison Wesley, 2003 (per pattern software) |

## Organizzazione della didattica

| **Ore** | | | |
|---|---|---|---|
| Totali | Didattica frontale | Esercitazione guidate + Progetto | Studio individuale |
| 225 ore | 56 ore | 15+25 ore di laboratorio ed esercitazioni guidate | 129 ore |
| **CFU/ETCS** | | | |
| 9 CFU | 7 CFU | 1 + 1P CFU | |

## Metodi didattici

| | |
|---|---|
| | Lezioni frontali con l'ausilio di slide che riportano esempi per illustrare gli argomenti trattati.<br>Esercitazioni pratiche sull'utilizzo dei vari principi e tecniche presentate a lezione attraverso esercizi da svolgere singolarmente.<br>Un progetto da svolgere preferibilmente in gruppo.<br>Utilizzo della piattaforma di e-learning del Dipartimento di Informatica per la distribuzione del materiale e per le interazioni tra docenti e studenti durante e dopo il corso. |

| **Risultati di apprendimento previsti** | |
|---|---|
| **Conoscenza e capacità di comprensione** | • Il principale risultato di apprendimento previsto è la conoscenza relativa a principi, paradigmi, metodologie, tecniche e tecnologie fondamentali per l'analisi e progettazione in team di sistemi software di medie-grandi dimensioni supportati da strumenti allo stato della pratica.<br>• Tali conoscenze mirano anche a fornire allo studente le competenze necessarie nella produzione e manutenzione di software applicativo per le applicazioni d'impresa.<br><br>Lo studente acquisisce tale conoscenza sia attraverso le lezioni frontali e la partecipazione a seminari tematici erogati durante il corso, sia attraverso esercitazioni che gli consente di mettere in pratica e verificare quanto appreso, acquisendo così consapevolezza della capacità di comprensione e di come migliorare l'applicazione delle tecniche apprese. |
| **Conoscenza e capacità di comprensione applicate** | • Per consentire allo studente di applicare le conoscenze per lo sviluppo (produzione e manutenzione) delle Applicazioni d'Impresa, si svolgono in aula sia esercitazioni individuali che collettive.<br>• Allo studente è richiesto di sviluppare un progetto, nel quale è necessario applicare i principi di ingegneria del software, le metodologie e le tecniche presentate a lezione, selezionando quelle più adeguate allo specifico caso. La valutazione di tale progetto contribuisce alla valutazione finale dello studente e quindi al voto conseguito all'esame di profitto. |
| **Competenze trasversali** | **Autonomia di giudizio**<br>• Acquisire una significativa autonomia nell'operare le opportune scelte durante l'analisi, la progettazione e lo sviluppo del sistema software oggetto del progetto.<br>• Acquisire la capacità di lavorare in team per lo sviluppo del sistema software e verificare i risultati ottenuti. Le esercitazioni che si svolgono durante il corso contribuiscono al raggiungimento di tali competenze grazie anche alla discussione di tali scelte con il docente.<br>• L'autonomia di giudizio è parte della valutazione finale dello studente e tiene conto delle discussioni avvenute durante le lezioni, delle esercitazioni e della presentazione del progetto.<br><br>**Abilità comunicative**<br>• Illustrare il risultato di esercizi svolti, autonomamente o in gruppo, con l'obiettivo di sviluppare le sue abilità comunicative.<br>• La presentazione e discussione del progetto sviluppato in gruppo è parte della prova orale d'esame e consente allo studente di mostrare le proprie abilità comunicative.<br><br>**Capacità di apprendere in modo autonomo**<br>• Per stimolare la capacità di apprendere in modo autonomo, allo studente è richiesto di approfondire specifici argomenti oppure è invitato a partecipare a seminari tenuti da altri docenti, interni o in visita al dipartimento, sui quali lo studente deve poi presentare durante le lezioni, e riportare in sede d'esame. |

## Valutazione

**Modalità di verifica dell'apprendimento**

La verifica dei risultati formativi raggiunti avviene durante l'esame finale, che prevede:

- Un colloquio orale in cui si presenta e si discute il progetto sviluppato in gruppo e si verificano le competenze acquisite durante il corso e le capacità espositive dello studente.
- Una prova scritta con domande a risposta multipla, chiuse e/o aperte.

Il risultato di ciascuna prova superata è valido per l'intero anno accademico in corso (8 appelli d'esame).

Criteri di valutazione

- **Conoscenza e capacità di comprensione**
  - Lo studente dovrà essere in grado di applicare correttamente principi e metodologie per lo sviluppo del progetto, validare l'appropriatezza delle tecniche usate, produrre una documentazione chiara ed esaustiva.
- **Conoscenza e capacità di comprensione applicate**
  - Si valuta la presentazione del progetto per verificare le competenze acquisite dallo studente e la sua capacità di sintesi nonché la chiarezza di esposizione, la capacità di fare confronti significativi tra metodologie, tecniche e tecnologie diverse adottate e riportare un proprio giudizio critico.
- **Autonomia di giudizio**
  - Lo studente dovrà essere in grado di applicare opportune soluzioni per lo sviluppo del sistema software.
  - Si valuta la presentazione del progetto per verificare le competenze acquisite dallo studente e la sua capacità di sintesi nonché la chiarezza di esposizione, la capacità di fare confronti significativi tra metodologie, tecniche e tecnologie diverse adottate e riportare un proprio giudizio critico.
  - Si valuta la prova scritta con domande a risposta multipla, chiuse e/o aperte, per accertare le conoscenze di base dello studente.
  - Il voto del progetto e la sua presentazione concorrono al 70% del voto complessivo dell'esame e la prova scritta al rimanente 30%.
- **Abilità comunicative**
  - Lo studente dovrà essere in grado di produrre una documentazione chiara e contenente le informazioni necessarie per il sistema software sviluppato.
- **Capacità di apprendere**
  - Lo studente dovrà essere in grado di tradurre autonomamente il progetto in un sistema software.

Criteri di misurazione dell'apprendimento e di attribuzione del voto finale

| Voto | Descrittori |
|---|---|
| < 18 insufficiente | Conoscenze frammentarie e superficiali dei contenuti, errori nell'applicare i concetti, descrizione carente. |
| 18 - 20 | Conoscenze dei contenuti sufficienti ma generali, descrizione semplice, incertezze nell'applicazione di concetti teorici. |
| 21 - 23 | Conoscenze dei contenuti appropriate ma non approfondite, capacità di applicare i concetti teorici, capacità di presentare i contenuti in modo semplice. |
| 24 - 25 | Conoscenze dei contenuti appropriate ed ampie, discreta capacità di applicazione delle conoscenze, capacità di presentare i contenuti in modo articolato. |

| | | |
|---|---|---|
| | 26 - 27 | Conoscenze dei contenuti precise e complete, buona capacità di applicare le conoscenze, capacità di analisi, descrizione chiara e corretta. |
| | 28 - 29 | Conoscenze dei contenuti ampie, complete ed approfondite, buona applicazione dei contenuti, buona capacità di analisi e di sintesi, descrizione sicura e corretta. |
| | 30<br>30 e lode | Conoscenze dei contenuti molto ampie, complete ed approfondite, capacità ben consolidata di applicare i contenuti, ottima capacità di analisi, di sintesi e di collegamenti interdisciplinari, padronanza di descrizione. |

## Altro

Si suggerisce agli studenti di affidarsi esclusivamente alle informazioni/comunicazioni fornite sui siti ufficiali del Dipartimento di Informatica, ovvero sui gruppi social solo se costituiti e amministrati esclusivamente dai docenti dei relativi insegnamenti:

- https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea/corsi-di-laurea
- https://www.uniba.it/it/ricerca/dipartimenti/informatica
- https://elearning.di.uniba.it/

I programmi degli insegnamenti sono disponibili qui:

- https://programmi.di.uniba.it/

Le informazioni che tutti gli studenti dovrebbero conoscere sono scritte nei Regolamenti didattici e manifesti degli studi disponibili nel sito:

- https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea/corsi-di-laurea

Si suggerisce agli studenti di diffidare delle informazioni e dei materiali circolanti su siti o gruppi social non ufficiali, poiché spesso sono risultati non affidabili, non corretti o incompleti. Per ogni dubbio, chiedere un incontro al docente secondo le modalità previste per il ricevimento.

---

Link al corso sulla piattaforma e-learning del dipartimento:

- https://elearning.uniba.it/course/view.php?id=5210

## Main information on the course

| | | |
|---|---|---|
| Course name | Software Engineering | |
| Degree | Computer Science and Digital Communication | |
| Academic year | 2024/25 | |
| European Credit Transfer and Accumulation System (ECTS), in Italian Crediti Formativi Universitari (CFU) | | 9 CFU |
| Settore Scientifico Disciplinare | INF/01 - Computer Science | |
| Course language | Italian | |
| Course year | Second | |
| Course period | 2nd semester (exact dates are specified in the curriculum/regulations) | |
| Course attendance requirement | Attendance is strongly recommended | |
| Website of the Degree | https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea/informatica-icd-taranto-270/laurea-triennale-in-informatica-e-comunicazione-digitale-sede-di-taranto-d.m.-270 | |

## Teacher(s)



| | |
|---|---|
| Name and Surname | Azzurra Ragone |
| email | azzurra.ragone@uniba.it |
| phone | +39 080-5443289 (int. 3289) |
| office | Department of Computer Science, Via Orabona 4, 70125 Bari. Room 619, 6th floor |
| e-learning platform | E-LEARNING Platform - https://elearning.uniba.it/ |
| Teacher's homepage | https://www.uniba.it/it/docenti/ragone-azzurra |
| Office hours | (To be confirmed)<br>Wednesday 12:30 - 13:30 (by appointment) |



## Syllabus

| | |
|---|---|
| **Course goals** | The Software Engineering course covers the analysis, design, and implementation of software systems using software engineering principles, methodologies, and development techniques. This includes the design of an enterprise application starting with gathering requirements. |
| **Prerequisites/requirements** | Students must be familiar with at least one programming language and fundamental data structures. The following preliminary knowledge facilitates and accelerates understanding of the course topics:<br>• From Programming: Imperative languages, ability to develop programs in a programming language (e.g., C);<br>• From Programming Languages: understanding the relationship between problems, algorithms, formal languages, and programming languages; syntax and semantics of a programming language;<br>• From Computer Lab: Fundamental algorithms, modular design. |
| **Course program** | **Introduction to Software Engineering (9 hours + 1 exercise)**<br>- Overview<br>- Types of software products<br>- Software development processes<br>- Product quality<br>- Software engineering issues |

| | |
|---|---|
| | **Software Engineering Principles (6 hours)**<br>- Applicability of principles<br>- Rigour and formality<br>- Separation of concerns<br>- Modularity<br>- Abstraction<br>- Generality<br>- Incrementality<br><br>**Requirements Analysis (8 hours + 2 exercises)**<br>- General concepts<br>- Requirement specifications<br>- Software specifications<br><br>**Agile Processes (7 hours)**<br>- Agile Software Development<br>- SCRUM methodology<br><br>**Software Design (4 hours + 2 exercises)**<br>- General concepts<br>- Basic process elements<br>- Design guidelines (Information Hiding)<br>- SW design process<br><br>**Software System Modelling Language – UML (10 hours + 3 exercises)**<br>- Overview<br>- Use case diagram: use cases, scenarios, relationships<br>- Class diagram: classes, objects, relationships<br>- Sequence diagram<br>- Component diagram<br>- Deployment diagram<br>- UML stereotypes: BCE (Boundary – Control -Entity) approach<br>- UML modelling examples<br><br>**Architectural Styles (4 hours + 1 exercise)**<br>- General principles<br>- Architectural styles<br>- Object Oriented<br><br>**Design Patterns (3 hours)**<br>- Creational patterns<br>- Structural patterns<br>- Behavioral patterns<br><br>**Development Support Tools (2 hours + 1 exercise)**<br>- Application Lifecycle Management (ALM)<br>- Configuration Management<br><br>**Case Study (3 hours + 5 exercises)**<br>- Introduction to the case study<br>- Requirements/user stories verification<br>- Examples and best practices<br>- Case study design |
| **Books of reference** | • Carlo Ghezzi, Medhi Jazayeri, Dino Mandrioli, “Fundamentals of Software Engineering”, 2nd ed. Pearson Prentice Hall, 2004<br>○ Chapter 1: Overview of Software Engineering<br>○ Chapter 2: Software: Nature and Quality<br>○ Chapter 3: Software Engineering Principles<br>• Ian Sommerville, “Software Engineering”, 10th ed. Pearson, 2017 |

| | |
|---|---|
| | ○ Chapter 2: Software Processes<br>○ Chapter 3: Agile Software Development<br>○ Chapter 5: System Models<br>○ Chapter 6: Architectural Design<br>○ Chapter 7: Design and Implementation<br>• Martin Fowler, "UML distilled. A Brief Guide to the Standard Modeling Language" (4th ed.). Pearson Addison Wesley, 2010<br>○ Chapter 1: Introduction<br>○ Chapter 3: Class Diagram: Basic Concepts<br>○ Chapter 4: Sequence Diagram<br>○ Chapter 5: Class Diagram: Advanced Concepts<br>○ Chapter 8: Deployment Diagram<br>○ Chapter 9: Use Cases<br>○ Chapter 14: Component Diagram<br><br>Students can borrow these texts from the library. It is advisable to check availability through the University Library System (https://opac.uniba.it/easyweb/w8018/index.php?) and contact the library for lending. |
| **Notes to the books** | The reference texts are supplemented with slides, teacher's notes, and other teaching materials made available to students on the e-learning platform used by the degree program.<br><br>Recommended texts for further study on specific topics:<br>• Jim Arlow, Ila Neustadt, "UML 2 and the Unified Process – Object-Oriented Analysis and Design", 2nd ed. McGraw-Hill, 2014 (for UML diagrams).<br>• Steven John Metsker, "Design Patterns in Java: A Practical Guide". Pearson Addison Wesley, 2003 (for software patterns). |

## Organization of the didactic activities

| **Hours** | | | | |
|---|---|---|---|---|
| Total | Lectures | Practice sessions | Project work | Individual study |
| 225 hours | 56 hours | 15 hours | 25 hours | 129 hours |
| **CFU/ETCS** | | | | |
| 9 CFU | 7 CFU | 1 CFU | 1 CFU | |

## Teaching methods

- Lectures with the aid of slides illustrating the discussed topics with examples.
- Practical exercises on using the various principles and techniques presented during the lectures through individual exercises.
- A project, preferably to be carried out in a group.
- Use of the Department of Computer Science's e-learning platform for distributing materials and facilitating interactions between teachers and students during and after the course.

| **Expected learning outcomes** | |
|---|---|
| **Knowledge and understanding** | • The primary learning outcome is knowledge of principles, paradigms, methodologies, techniques, and technologies fundamental for analyzing and designing medium-to-large-scale software systems in teams, supported by state-of-the-art tools.<br>• This knowledge aims to equip students with the skills necessary for producing and maintaining enterprise application software.<br><br>Students acquire this knowledge through lectures and thematic seminars during the course and through practical exercises that allow them to practice and verify what they have learned, gaining awareness of their understanding and how to improve the application of learned techniques. |
| **Applying knowledge and understanding** | • To enable students to apply their knowledge in developing (producing and maintaining) Enterprise Applications, both individual and collective exercises are conducted in the classroom.<br>• Students are required to develop a project in which they must apply the principles of software engineering, methodologies, and techniques presented in the lectures, selecting the most appropriate ones for the specific case. The evaluation of this project contributes to the final assessment of the student and, consequently, to the grade achieved in the exam. |
| **Other skills** | *Making judgements*<br>• Gain significant autonomy in making appropriate choices during the analysis, design, and development of the software system under study.<br>• Acquire the ability to work in a team for software system development and verify the results obtained. The exercises conducted during the course contribute to achieving these skills, thanks also to the discussion of these choices with the teacher.<br>• Autonomy of judgment is part of the final assessment of the student, taking into account the discussions held during lectures, exercises, and project presentation.<br><br>*Communication*<br>• Illustrate the results of exercises carried out independently or in a group with the aim of developing communication skills.<br>• The presentation and discussion of the project developed in a group are part of the oral exam and allow the student to demonstrate their communication skills.<br><br>*Learning skills*<br>• To stimulate the ability to learn independently, students are required to delve into specific topics or are invited to attend seminars given by other internal or visiting lecturers, on which they must then present during lectures and report in the exam. |

| Assessment | |
|---|---|
| **Assessment methods** | The assessment of the achieved learning outcomes occurs during the final exam, which includes:<br>• An oral interview presenting and discussing the group-developed project, verifying the knowledge acquired during the course and the student's presentation skills.<br>• A written test with multiple-choice and/or open-ended questions.<br>The result of each passed test is valid for the entire current academic year (8 exam sessions). |
| **Evaluation criteria** | **Knowledge and Understanding**<br>The student must be able to correctly apply principles and methodologies for project development, validate the appropriateness of the techniques used, and produce clear and exhaustive documentation.<br><br>**Applied Knowledge and Understanding**<br>The project presentation is evaluated to verify the student's acquired skills, summarization ability, and clarity of presentation, as well as the ability to make significant comparisons between different methodologies, techniques, and technologies adopted and to provide their critical judgment.<br><br>**Autonomy of Judgment**<br>• The student must be able to apply appropriate solutions for software system development.<br>• The project presentation is evaluated to verify the student's acquired skills, summarization ability, and clarity of presentation, as well as the ability to make significant comparisons between different methodologies, techniques, and technologies adopted and to provide their critical judgment.<br>• The written test with multiple-choice and/or open-ended questions is evaluated to verify the student's basic knowledge.<br>• The project grade and its presentation account for 70% of the overall exam grade, and the written test accounts for the remaining 30%.<br><br>**Communication Skills**<br>The student must be able to produce clear documentation containing the necessary information for the developed software system.<br><br>**Learning Ability**<br>The student must be able to translate knowledge independently into a software system. |
| Measurements and final grade | <table><tr><th>Grade</th><th>Descriptors</th></tr><tr><td>< 18 insufficient</td><td>Fragmentary and superficial content knowledge, errors in applying concepts, poor description.</td></tr><tr><td>18-20</td><td>Sufficient but general content knowledge, simple description, uncertainties in applying theoretical concepts.</td></tr><tr><td>21-23</td><td>Appropriate but not deep content knowledge, ability to apply theoretical concepts, ability to present content simply.</td></tr></table> |

| | |
|---|---|
| 24-25 | Appropriate and broad content knowledge, fair ability to apply knowledge, ability to present content articulately. |
| 26-27 | Precise and complete content knowledge, good ability to apply knowledge, clear and correct description. |
| 28-29 | Broad, complete, and deep content knowledge, good content application, good analysis and synthesis ability, confident and correct description. |
| 30<br>30 e lode | Very broad, complete, and deep content knowledge, well-established content application ability, excellent analysis, synthesis, and interdisciplinary connections, mastery of description. |

## Further information

Students are advised to rely exclusively on information/communications provided on the official websites of the Department of Computer Science or on social groups only if formed and administered exclusively by the lecturers of the related courses:

- https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea/corsi-di-laurea
- https://www.uniba.it/it/ricerca/dipartimenti/informatica
- https://elearning.uniba.it/

The course programs are available here:

- https://elearning.uniba.it/

The information that all students should know is written in the Teaching Regulations and study posters available on the site:

- https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea/corsi-di-laurea

Students are advised to be cautious of information and materials circulating on unofficial sites or social groups as they are often unreliable, incorrect, or incomplete. For any doubts, request a meeting with the instructor according to the office hour arrangements.

Link to the course on the e-learning platform of the University E-Learning Center:

- https://elearning.uniba.it/course/view.php?id=5210