Principali informazioni sull'insegnamento

Denominazione dell'insegnamento	**Integrazione e Test di Sistemi Software**	
Corso di studio	Informatica e Tecnologie per la Produzione del Software	
Anno Accademico	2025/26	
Crediti formativi universitari (CFU) / European Credit Transfer and Accumulation System (ECTS)		6 CFU
Settore Scientifico Disciplinare	ING-INF/05	
Lingua di erogazione	Italiano	
Anno di corso	Terzo	
Periodo di erogazione	1^semestre, le date esatte sono riportate nel manifesto/regolamento	
Obbligo di frequenza	No, ma la frequenza è fortemente raccomandata	
Sito web del corso di studio	https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea/informatica-tps-270/laurea-triennale-in-informatica-e-tecnologie-per-la-produzione-del-software-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, 6^ piano.
Sede virtuale	Piattaforma elearning - 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)	Martedì 12:00 - 13:00 (su appuntamento da concordare per email con il docente)

Syllabus

Obiettivi formativi	L'insegnamento si propone di introdurre le metodologie, strategie, tecniche e strumenti di integrazione e testing che concorrono alla Verifica e Validazione del software. Obiettivo dell'insegnamento è favorire l'acquisizione di competenze utili sia allo sviluppo di software di qualità che per la sua valutazione.
Prerequisiti	Le seguenti conoscenze preliminari facilitano ed accelerano la comprensione degli argomenti dell'insegnamento: - da Programmazione II: fondamenti della programmazione ad oggetti (Java), gestione delle eccezioni in Java, lambda expression e Stream, familiarità di utilizzo con un ambiente di sviluppo (un IDE, tipo Eclipse, IntelliJ, ecc.).
Contenuti di insegnamento (Programma)	- **Introduzione al Software Testing (5h)**: Validazione e Verifica – Terminologia - Come selezionare le tipologie di test più adeguate - Test automation – La piramide dei test - **Testing Black Box (8h)**. Testing basato sui requisiti e sugli scenari dei casi d'uso - Test di tipo funzionale – identificare test case per i valori soglia (boundary test) - **Testing White Box (8h)** Code coverage - criteri di code coverage - Strumenti per la misura automatica della copertura del codice - Test di tipo strutturale - **Design Contracts (3h)** Definire le pre-condizioni e post-condizioni nei contratti, differenza tra contratti e validazione del software - **Property-based testing (5h)** Cosa sono e differenza con i test di tipo strutturale e funzionale - **Mocking framework (3h)** Dummies, fakes, stubs, spies e mocks - Test double - **Static testing (3h)** Differenza tra testing dinamico e statico. Tecniche di testing statico e suoi benefici. - **Test-driven development (TDD) (3h)** Cosa si intende per TDD - Quando usare l'approccio TDD e quando non usarlo - **Design per la testabilità e Qualità del codice di test (5h)** Come scrivere codice di programmazione che favorisca la testabilità dello stesso. Best practice per la scrittura di codice di test di qualità e che sia manutenibile – Test smell - **Documentazione di test (1h)** Documento di strategia dei test – Test Plan – Test Design Specification Document - **Integration and System Test (6h)** Testare diversi componenti del software – Testing di SQL e Database – Test di Sistemi- Analisi costi/benefici - **JUnit (7h)**: Introduzione a JUnit - Implementazione di test di unità con JUnit su programmi Java – Assunzioni e asserzioni – Testing delle eccezioni – Test dinamici e parametrici – Testing Data Driven con JUnit - **LLM-based testing (2h)** Scrivere codice di test avvalendosi dell'uso di Large Language Models (LLMs) – Verifica della correttezza del codice – Correzione degli errori - **Integrazione dei Sistemi Software (3h)** Diverse architetture e processi funzionali all'integrazione dei sistemi software (es. architetture a microservizi)
Testi di riferimento	Testo di riferimento: Effective Software Testing (A developer's guide) Mauricio Aniche. Manning. ISBN 9781633439931, 2022 (NB: Vengono trattati tutti i capitoli del libro) Testo consigliato: G. J. Myers, T. Badgett, C. Sandler. The Art of Software Testing. Wiley (Cap.1, 2, 4, 20, 21, 22, 24) Materiale messo a disposizione dal docente su piattaforma e-learning

	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	Sono disponibili sulla piattaforma e-learning di UNIBA: - Slide usate a lezione (caricate dopo ogni lezione) - Codici degli esercizi svolti durante i laboratori Il docente indica sempre alla fine delle slide di ogni lezione il testo e il capitolo del libro a cui la lezione fa riferimento. Inoltre, sempre alla fine delle slide vengono indicati i riferimenti ad eventuali altri materiali bibliografici (white paper, articoli scientifici, guide, ecc.), repository di codice ed eventuali approfondimenti.

Organizzazione della didattica

Ore

Totali	Didattica frontale	Pratica (laboratorio, progetto, esercitazione, altro)	Studio individuale
150 ore	32 ore	30 ore	88 ore

CFU/ETCS

6 CFU	4 CFU	2 CFU	

Metodi didattici

- Lezioni frontali condotte con l'ausilio di slide proiettate in aula e rese disponibili tramite la piattaforma di e-learning;
- Svolgimento in aula di due tipologie di esercitazione: (a) esercizi svolti interamente dal docente con indicazione delle soluzioni; (b) esercitazioni guidate in cui gli studenti risolvono da soli, ma supervisionati dal docente, problemi relativi al testing di sistemi software.

Entrambe le tipologie di esercitazione sono svolte con l'obiettivo di acquisire dimestichezza con le metodologie di software testing da applicare successivamente al caso di studio da svolgere preferibilmente in gruppo.

Seminari svolti da aziende che illustrano casi di studio reali riguardanti gli argomenti del corso.

Risultati di apprendimento previsti

Conoscenza e capacità di comprensione

	▪ Comprensione del concetto di qualità del software e dei concetti di verifica e validazione dello stesso ▪ Conoscenza delle diverse tecniche di integrazione e testing dei sistemi software (ad es. white-box e black-box), metodologie di sviluppo (ad es. sviluppo test driven) e degli strumenti a supporto degli sviluppatori per l'integrazione e testing di tali sistemi
Conoscenza e capacità di comprensione applicate	▪ Saper applicare le best practice del software testing per garantire lo sviluppo di software di qualità (robusto, affidabile, ecc.). ▪ Saper applicare i concetti, le tecniche e le metodologie di integrazione e testing di sistemi software, nonché utilizzare gli strumenti di supporto agli sviluppatori.
Competenze trasversali	**Autonomia di giudizio** 1) Acquisire autonomia di giudizio sulle scelte da effettuarsi relativamente allo sviluppo di test case suite e alle tecniche di integrazione e testing da adottare. **Abilità comunicative** 2) Capacità di comunicare sia in maniera orale che tramite documenti scritti (ad es. test case suite) la soluzione proposta; saper modulare la comunicazione rispetto ai diversi stakeholder (tecnici e non) di un progetto software usando la giusta terminologia. **Capacità di apprendere in modo autonomo** 3) Sviluppare capacità di intraprendere in autonomia ulteriori approfondimenti su argomenti attinenti all'integrazione e testing di sistemi software.

Valutazione	
Modalità di verifica dell'apprendimento	La verifica dei risultati formativi raggiunti avviene durante l'esame finale, che prevede una prova scritta con domande a risposta chiusa e domande a risposta aperta. La durata della prova scritta è fissata in circa 1h. Il voto della prova scritta sarà in trentesimi. Inoltre, è prevista la presentazione in aula di un lavoro di gruppo (facoltativo): ▪ Si presenta e si discute in maniera critica il lavoro sviluppato in gruppo; ▪ Si verificano le competenze acquisite durante il corso; ▪ Si accertano le capacità espositive dello studente e la capacità di motivare le scelte progettuali. Il lavoro di gruppo va concordato con il docente e inizia ad essere sviluppato sotto la guida del docente durante le ore di laboratorio.

	Il lavoro di gruppo è facoltativo e contribuisce, se svolto, alla votazione finale dell'esame conferendo un bonus da 1 a 3.
Criteri di valutazione	• **Conoscenza e capacità di comprensione:** Si valuta attraverso la prova scritta, e l'eventuale lavoro di gruppo, la conoscenza e la capacità di comprensione da parte dello studente dei concetti, delle tecniche, degli strumenti e delle metodologie di integrazione e testing di sistemi software affrontati a lezione • **Conoscenza e capacità di comprensione applicate:** Si valuta attraverso la prova scritta, e l'eventuale lavoro di gruppo, la capacità di saper applicare le best practice per garantire lo sviluppo di software di qualità, oltre a saper applicare i concetti, le tecniche e le metodologie di integrazione e testing di sistemi software, nonché utilizzare gli strumenti di supporto • **Autonomia di giudizio:** Si valuta attraverso la prova scritta, e l'eventuale lavoro di gruppo, la capacità di effettuare autonomamente scelte relativamente allo sviluppo di test case suite e alle tecniche di integrazione e testing da adottare. • **Abilità comunicative:** Si valuta attraverso la prova scritta, e l'eventuale lavoro di gruppo, la capacità di comunicare la soluzione proposta usando la giusta terminologia. • **Capacità di apprendere:** Si valuta attraverso la prova scritta, e l'eventuale lavoro di gruppo, la capacità di applicare i concetti appresi durante il corso a nuovi casi di test non analizzati a lezione
Criteri di misurazione dell'apprendimento e di attribuzione del voto finale	Il voto finale è dato dalla votazione della prova scritta, a cui si sommerà eventualmente la votazione del lavoro di gruppo, se svolto dallo studente.
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 ADA: https://elearning.di.uniba.it/course/view.php?id=1086

Main information on the course		
Course name	Integration and Testing of Software Systems	
Degree	Computer Science and Software Production Technologies	
Academic year	2025/26	
European Credit Transfer and Accumulation System (ECTS), in Italian Crediti Formativi Universitari (CFU)		6 CFU (each CFU corresponds to 25 hours (h) of student's time); CFU are of type T1, T2 or T3 T1 = 32 h lecture T2 = 30 h practice T3 = 88 h individual study
Settore Scientifico Disciplinare	ING-INF/05	
Course language	Italian	
Course year	Third	
Course period	First Semester - exact dates can be found in the didactic regulations	
Course attendance requirement	None, but it is highly recommended to attend classes	
Website of the Degree	https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea/informatica-tps-270/laurea-triennale-in-informatica-e-tecnologie-per-la-produzione-del-software-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 No. 616, 6th floor
e-learning platform	https://elearning.di.uniba.it/
Teacher's homepage	https://www.uniba.it/it/docenti/ragone-azzurra
Office hours	Tuesday 12:00 - 13:00 (by appointment to be arranged via email with the professor)

Syllabus	
Course goals	The course aims to introduce the methodologies, strategies, techniques, and tools for software integration and testing that contribute to software verification and validation. The goal is to help students acquire skills useful for both developing quality software and its evaluation.
Prerequisites/requirements	The following preliminary knowledge facilitates and accelerates understanding of the course topics: From Programming II: Fundamentals of object-oriented programming (Java), exception handling in Java, lambda expressions and Streams, familiarity with a development environment (an IDE like Eclipse, IntelliJ, etc.).
Course program	• **Introduction to Software Testing (5h):** Validation and Verification – Terminology - How to select appropriate test types - Test automation – Test pyramid • **Black Box Testing (8h):** Requirement-based and use case scenario testing - Functional testing – Identifying test cases for boundary values

	• **White Box Testing (8h):** Code coverage - Code coverage criteria - Tools for automatic measurement of code coverage - Structural testing • **Design Contracts (3h):** Defining pre-conditions and post-conditions in contracts - Difference between contracts and software validation • **Property-based Testing (5h):** What they are and how they differ from structural and functional tests • **Mocking Framework (2h):** Dummies, fakes, stubs, spies, and mocks - Test doubles • **Static Testing (3h):** Difference between dynamic and static testing. Static testing techniques and benefits. • **Test-Driven Development (TDD) (3h):** What is TDD - When to use and not use the TDD approach • **Design for Testability and Quality of Test Code (5h):** How to write programming code that promotes testability. Best practices for writing quality and maintainable test code – Test smells • **Test Documentation (1h):** Test strategy document – Test Plan – Test Design Specification Document • **Integration and System Test (6h):** Testing various software components – SQL and Database testing – System testing- Cost/benefit analysis • **JUnit (7h):** Introduction to JUnit - Implementing unit tests with JUnit on Java programs – Assumptions and assertions – Exception testing – Dynamic and parametric testing – Data-driven testing with JUnit • **LLM-based testing (2 hours)** Writing test code using Large Language Models (LLMs) – Verifying code correctness – Correcting errors • **Software System Integration (3h):** Different architectures and processes for software system integration (e.g., microservices architectures)
Books of reference	• Main text: "Effective Software Testing: A Developer's Guide" by Mauricio Aniche. Manning, ISBN 9781633439931, 2022 (All chapters of the book are covered) • Recommended text: G. J. Myers, T. Badgett, C. Sandler. "The Art of Software Testing." Wiley (Chapters 1, 2, 4, 20, 21, 22, 24) • Materials provided by the instructor on the e-learning platform
Notes to the books	Available on the UNIBA e-learning platform: • Slides used in class (uploaded after each lesson) • Codes of exercises performed during laboratories The instructor always indicates at the end of the slides of each lesson the text and chapter of the book to which the lesson refers. Additionally, at the end of the slides, references to any other bibliographic materials (white papers, scientific articles, guides, etc.), code repositories, and any further insights are provided.

Organization of the didactic activities

Hours				
Total	Lectures	Practice sessions	Project work	Individual study
150 hours	32 hours	30 hours	0 hours	88 hours
CFU/ETCS				
6 CFU	4 CFU	2 CFU	0 CFU	

Teaching methods	
	• Lectures conducted with the aid of slides projected in class and made available through the e-learning platform • Two types of exercises conducted in class: (a) exercises entirely performed by the instructor with indications of the solutions; (b) guided exercises where students solve problems related to software system testing independently but supervised by the instructor. Both types of exercises aim to familiarize with software testing methodologies to be subsequently applied to case studies, preferably carried out in groups. • Seminars conducted by companies illustrating real case studies related to the course topics.

Expected learning outcomes	
Knowledge and understanding	• Understanding the concept of software quality and the concepts of verification and validation. • Knowledge of various integration and testing techniques for software systems (e.g., white-box and black-box), development methodologies (e.g., test-driven development), and tools supporting developers in the integration and testing of these systems.
Applying knowledge and understanding	Applying best practices in software testing to ensure the development of quality software (robust, reliable, etc.). Applying concepts, techniques, and methodologies of software system integration and testing and using support tools for developers.
Other skills	*Making judgements* Acquiring the autonomy to make decisions regarding the development of test case suites and the integration and testing techniques to adopt. *Communication* Ability to communicate both orally and in writing (e.g., test case suites) the proposed solution; ability to modulate communication to the different stakeholders (technical and non-technical) of a software project using the right terminology. *Learning skills* Developing the ability to undertake further in-depth studies independently on topics related to the integration and testing of software systems.

Assessment	
Assessment methods	The assessment of the learning outcomes achieved is carried out during the final exam, which includes a written test with closed and open questions. The duration of the written test is set at about 1 hour.

	The grade of the written test will be in thirtieths. Additionally, there is a presentation in class of a group project (optional): - presenting and critically discussing the work developed in the group - verifying the skills acquired during the course; - assessing the student's presentation skills and the ability to justify design choices. The group project must be agreed upon with the instructor and begins to be developed under the instructor's guidance during lab hours. The group project is optional and contributes to the final exam score, giving a bonus of 1 to 3 points if carried out.
Evaluation criteria	• **Knowledge and Understanding:** Assessed through the written test and the optional group project, evaluating the student's knowledge and understanding of the concepts, techniques, tools, and methodologies of software system integration and testing addressed in class. • **Applied Knowledge and Understanding:** Assessed through the written test and the optional group project, evaluating the ability to apply best practices to ensure the development of quality software and the ability to apply concepts, techniques, and methodologies of software system integration and testing, as well as using support tools. • **Autonomy of Judgment:** Assessed through the written test and the optional group project, evaluating the ability to independently make decisions regarding the development of test case suites and the integration and testing techniques to adopt. • **Communication Skills:** Assessed through the written test and the optional group project, evaluating the ability to communicate the proposed solution using the right terminology. • **Learning Skills:** Assessed through the written test and the optional group project, evaluating the ability to apply the concepts learned during the course to new test cases not analyzed in class.
Measurements and final grade	The final grade is given by the score of the written test plus the possible score of the group project if carried out by the student.
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 established and managed exclusively by the instructors of the respective 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.di.uniba.it/ Teaching schedules are available here: ● https://programmi.di.uniba.it/ Information that all students should know is written in the Teaching Regulations and Study Manifestos available on the website: ● https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea/corsi-di-laurea Students are suggested to be wary of information and materials circulating on unofficial sites or social groups, as they are often found to be unreliable, incorrect

	or incomplete. If you have any doubts, ask for a meeting with the lecturer in accordance with the reception arrangements. _____ Link to the course on the ADA department's e-learning platform: https://elearning.di.uniba.it/course/view.php?id=1086