European Software
Skills Alliance.

ESSA Learning programmes

ANNEX I
Junior Developer EQF 4/5

30 November 2023

Status: Final version

ESSA Learning programme – Junior Developer EQF 4/5, 2023.

Deliverable 10 – ESSA Learning Programmes & Materials – ANNEX I

This document is a draft version and is subject to change after review coordinated by the European Education and Culture Executive Agency (EACEA).

Authors: Federica D'Armini (Adecco/Mylia)

Editors/Reviewers: Alessandro Prunesti (Adecco/Mylia), Chiara Longobardi (Adecco/Mylia), Carmel Somers (DTSL), Niels Sellings (Digital Europe), Ants Sils (BCS Koolitus), Merje Vaide (BCS Koolitus), FOR FINAL VERSION: all ESSA WP4 involved partners

softwareskills.eu

Legal Disclaimer

Co-funded by the Erasmus+ Programme of the European Union

About ESSA

The European Software Skills Alliance (ESSA) is a four-year transnational project funded under the EU's Erasmus+ programme. It ensures the skills needs of the rapidly evolving Software sector can be met — today and tomorrow.

ESSA provides current and future software professionals, learning providers and organisations with software needs with the educational and training instruments they need to meet the demand for software skills in Europe.

ESSA will develop a European Software Skills Strategy and learning programmes for Europe. It will address skill mismatches and shortages by analysing the sector in depth and delivering future-proof curricula and mobility solutions; tailored to the European software sector's reality and needs.

Project partners

The ESSA consortium is led by DIGITALEUROPE. It is composed of academic and non-academic partners from the education, training, and software sectors.

View all project partners: ESSA Partners | ESSA Associated Partners

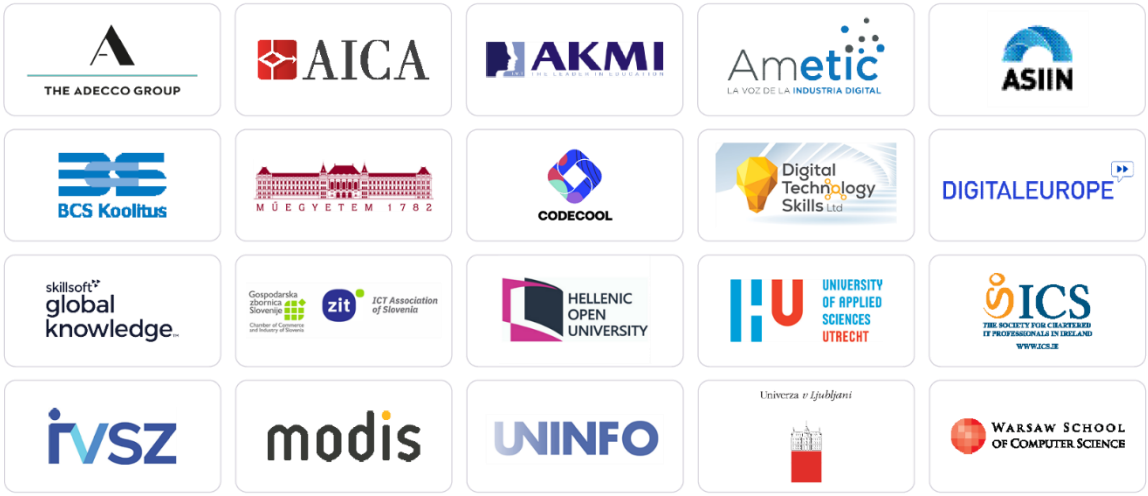

Table of Contents

List of abbreviations and acronyms

Abbreviation	Term
e-CF, EN 16234-1	European e-Competence Framework, European Norm 16234 - Part 1: Framework
ECTS	European Credit Transfer and Accumulation System
EQF	European Qualifications Framework
ESSA	European Software Skills Alliance
LO	Learning Outcome
PLO	Programme Learning Outcome

1 Junior Developer EQF 4/5 – ESSA Learning Programme

1.1 Students with ICT background

Executive summary

The Learning programme is being designed by Adecco Formazione (Mylia) (IT) to develop the skills of students from Technical Institutes, University students and professionals committed in upskilling or reskilling paths with a prior basic knowledge of the topic.

The Learning programme provides participants with the knowledge necessary for software development at junior level. Programming languages, techniques and practices for frontend and backend development. The main support tools, the fundamentals of software testing and project management and team collaboration skills are explored to ensure maximum productivity in business contexts.

Targeted Institutions: Higher Education and VET providers.

The recommended Learning programme is articulated in fourteen (14) Learning Units, for a total of 200 hours of study and 8 ECTS.

The recommended delivery method is the Virtual Classroom.

1.1.1 PLO 1. Application Design [e-2]

1. PLO Application Design [e-2]	
The learner has demonstrated capability *→ to interpret a design for a software application or component*	
Unit learning outcomes	Explains and distinguishes basic principles and terminology of software design (e.g., phases in the design process, common techniques, deliverables)
	Describes principles of user interface design
	Reads design models and diagrams (e.g., ERD, UML)
	Interprets a basic database design
	Interprets a design for an application or software component

1.1.1.1 Duration of Study

Recommended duration: starting from n.0,5 ECTS

Often integrated with studies of PLOs: PLO 2 Application Development [e-2]

1.1.1.2 Recommendations for Micro-credentials

Not applicable.

1.1.1.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Virtual Classroom ☒
- Work placement ☒

Recommended delivery methods:

- Lecture ☒ up to 60%
- Coding Training Lab delivered by individual/team project work ☒ up to 40%

Additional comments
It is recommended to deepen the topics presented in the Learning Units by reading publications dedicated to the various topics, reading websites specialized in programming, watching online tutorials and downloading materials useful for practical exercises from official sources.

1.1.1.4 WBL and Follow-up Reinforcement

After learning the basic principles, terminology, and models of software design, the study should focus on analysing and simulating real work-life-like tasks as, for example the student:

- Specifies a design for an application or software (component), taking into account certain basic constraints/ requirements;
- Checks whether the design meets requirements/ wishes and, if necessary, makes proposals to adapt the design;
- Interprets and employs the software (component)/ application design and design patterns.

1.1.1.5 Important (new) approaches and technologies to consider

Before starts to coding, the learner should decide if he/she wants to start off with a set of codes and stick with them (deductive coding), or come up with the codes as he/she read what he/she read see in his/her data (inductive), or take a combination approach.

1.1.1.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
Collects, formalises and validates functional and non-functional requirements;	Exam: The candidate must evaluate which programming language is most suitable for the development of specific digital services.	n/a

Checks whether the design meets requirements/ wishes and, if necessary, makes proposals to adapt the design	Exam: The candidate must evaluate which programming language is most suitable for the development of specific digital services.	n/a
Specifies a design for an application or software (component), taking into account certain basic constraints/ requirements	Exam: The candidate must evaluate which programming language is most suitable for the development of specific digital services.	n/a
Interprets and employs the software (component)/ application design and design patterns.	Exam: The candidate must evaluate which programming language is most suitable for the development of specific digital services.	n/a
Collects, formalises and validates functional and non-functional requirements	Assignment: Practical activity. The learner is asked to build the best User Experience for the web application developed in the previous modules.	n/a
Drafts functional and technical design	Assignment: Practical activity. The learner is asked to build the best User Experience for the web application developed in the previous modules.	n/a
Specifies a design for an application or software (component), taking into account certain basic constraints/ requirements	Assignment: Practical activity. The learner is asked to build the best User Experience for the web application developed in the previous modules.	n/a
Checks whether the design meets requirements/ wishes and, if necessary, makes proposals to adapt the design.	Assignment: Practical activity. The learner is asked to build the best User Experience for the web application developed in the previous modules.	n/a

1.1.2 Learning Resources - PLO 1. Application Design [e-2]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Overview of the Main Programming Languages and key differences	5	8 hours	The didactic approach would be aimed to allows participants to understand the characteristics and differences between the main programming languages, using practical examples, such as viewing and analyzing programming code.	1 exam. The candidate must evaluate which programming language is most suitable for the development of specific digital services.	Overview of the Main Programming Languages and key differences	Virtual classroom, Workshop and lecture guides	Junior Developer_EQF_4_5_Overview of the main programming languages.pptx Junior Developer_EQF_4_5_Cloud and Virtualization.pptx
Principles of UI/UX Design. Adobe XD, Zepling.	5	12 hours	The didactic approach would be aimed to allow participants to understand the main UI/UX design	Assigmment: practical activity. The student is asked to build the best User Experience for the web application	Principles of UI UX Design	Virtual classroom, Workshop and lecture guides	Junior Developer_EQF_4_5_Principles of UI-UX Design.pptx

		principles, using pratical examples, such as viewing and analyzing programming code.	developed in the previous modules.			

1.1.3 PLO 2. Application Development [e-2]

2. PLO Application Development [e-2]	
The learner has demonstrated capability *→ to systematically develop a simple software application or component* *→ to propose modifications to an existing solution* *→ to document the development activities*	
Unit learning outcomes	Explains and distinguishes common software development methods (e.g., waterfall, iterative, agile), techniques (e.g., object-oriented) and tools (e.g., IDE, CASE; multimedia integration tools; app development tools)
	Describes common programming principles and terminology (e.g., secure programming)
	Explains concepts and principles of databases, data structures and query languages (e.g., SQL)
	Participates in a development process and applies a common software development method (e.g., agile)
	Creates a simple relational database
	Writes code and related documentation to it, by using a common programming language and applying coding conventions (e.g., Java, Javascript, PHP, Python; clean coding principle)
	Creates a simple working software component or application, taking into account architecture, design requirements and other possible constraints (e.g., installability) applying relevant tools and techniques (e.g., object-oriented programming; IDE, CASE; editors, compilers, version control tools)
	Modifies an existing software application or component

1.1.3.1 Duration of Study

Recommended duration: starting from n. 4 ECTS

Often integrated with studies of PLOs: 1- 3-4-5

1.1.3.2 Recommendations for Micro-credentials

Not Applicable

1.1.3.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Virtual Classroom ☒

Additional comments
n/a

Recommended delivery methods:

- Lecture ☒ up to 60%
- Coding Training Lab delivered by individual/team project work ☒ up to 40%

Additional comments
It is recommended to deepen the topics presented in the Learning Units by reading publications dedicated to the various topics, reading websites specialized in programming, watching online tutorials and downloading materials useful for practical exercises from reliable sources.

1.1.3.4 WBL and Follow-up Reinforcement

After learning the basic principles, terminology, and models of Application Development, the study should focus on analysing and simulating real work-life-like tasks as, for example, the student:

- Participates in a development process and applies a common software development method;
- Writes code and related documentation, by using a common programming language and applying coding conventions (e.g., Java, Javascript, PHP, Python; clean coding principle)

1.1.3.5 Important (new) approaches and technologies to consider

Not applicable

1.1.3.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
Explains and distinguishes common software development methods (e.g., waterfall, iterative, agile), techniques (e.g., object-oriented) and tools (e.g., IDE, CASE; multimedia integration tools; app development tools)	1 exam. The candidate must evaluate which programming language is most suitable for the development of specific digital services.	n/a
Describes common programming principles and terminology (e.g., secure programming)	1 exam. The candidate must evaluate which programming language is most suitable for the development of specific digital services.	n/a
Explains concepts and principles of databases, data structures, and query languages (e.g., SQL)	1 exam. The candidate is asked to set up a database to support a web application	n/a
Participates in a development process and applies a common software development method (e.g., agile)	1 exam. The candidate must evaluate which programming language is most suitable for the development of specific digital services.	n/a
Creates a simple relational database	1 exam. The candidate is asked to set up a database to support a web application	n/a

Writes code and related documentation to it, by using a common programming language and applying coding conventions	Assignment: practical activity. The student is asked to create a web application using HTML5, CSS3 and Bootstrap.	n/a
Creates a simple working software component or application, taking into account architecture, design requirements and possible other constraints (e.g., installability) applying relevant tools and techniques	Assignment: practical activity. The student is asked to create a web application using HTML5, CSS3 and Bootstrap.	n/a
Modifies an existing software application or component	Assignment: practical activity. The student is asked to modify a web application using HTML5, CSS3 and Bootstrap.	n/a

1.1.4 Learning Resources – PLO 2. Application Development [e-2]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Overview of the Main Programming Languages and key differences;	5	8 hours	The didactic approach would be aimed to allows participants to understand the main programming languages and their applications using practical examples, such as viewing and analyzing programming code.	1 exam. The candidate must evaluate which programming language is most suitable for the development of specific digital services.	Overview of the Main Programming Languages and key differences	Workshop and lecture guides	Junior Developer_EQF_4_5_Overview of the main programming languages.pptx
HTML5, CSS3, BOOTSTRAP	5	24 hours	The didactic approach would be aimed to allows participants to understand the HTML and CSS	Assignment: practical activity. The student is asked to create a web application	• HTML5, CSS3, Bootstrap	Workshop and lecture guides	Junior Developer_EQF_4_5_HTML.pptx Junior Developer_EQF_4_5_CSS.pptx

			code and relative application, using practical examples, such as viewing and analyzing programming code.	using HTML5, CSS3 and Bootstrap.	• Exercise: JuniorDEVProj_HTML-CSS-JS		
Javascript, AJAX, Typescript, *GIT*	5	24 hours	The didactic approach would be aimed to allows participants to understand the main applications of javascript, ajax, typescript, using practical examples, such as viewing and analyzing programming code.	Assignment: practical activity. The student is asked to develop a web application using frontend coding (Javascript, AJAX, Typescript) and related tools.	• Javascript Ajax • Java at main	Workshop and lecture guides	https://learn.softwareskills.eu/wp-content/uploads/2023/11/Javascript-Ajax.pptx https://learn.softwareskills.eu/wp-content/uploads/2023/12/ESSA-Junior-Devloper_JAVA.pptx
Backend – Coding and development tools: Java 11, Spring Boot, Sprint Data,	5	16 hours	The didactic approach would be aimed to allows participants to understand the	Assignment: practical activity. The student is asked to build and test a	• Backend – Java 11	Workshop and lecture guides	• Backend – Java 11.pptx

Hibernate, Ex Java: Junit, Mockito.			main applications of those sets of code using practical examples, such as viewing and analyzing programming code.	backend-side application using Java 11 and related development tools.	• Spring at main		• JuniorDEVProj_Spring at main · MaSTERmIKK_JuniorDEVProj · GitHub
Backend – Coding and development tools: PHP, Laravel, Eloquent	5	16 hours	The didactic approach would be aimed to allows participants to understand the main applications of those sets of code using practical examples, such as viewing and analyzing programming code.	Assignment: practical activity. The student is asked to build and test a backend-side application using PHP and related development tools.	PHP Lavarel Eloquent	Workshop and lecture guides	PHP Lavarel Eloquent.pptx
Backend – Coding and development tools: Fundamentals	5	16 hours	The didactic approach would be aimed to allows participants to	Assignment: practical activity. The student is asked to build	• Back End – Objects – Ruby – Phyton - NodeJS	Workshop and lecture guides	• Back End – Objects – Ruby – Phyton -NodeJS.pptx

of Ruby, Python, NodeJS			understand the main applications of those sets of code using practical examples, such as viewing and analyzing programming code.	a backend-side application using Ruby, Python and NodeJS.	• NodeJS at Main		• JuniorDEVProj_NodeJS at main · MaSTERmIKK_JuniorDEVProj · GitHub
Agile Project Management, SCRUM and collaboration tools	5	8 hours	The didactic approach would be aimed to allows participants to understand the Agile and SCRUM culture and framework using practical examples and exercises.	Assignment: practical activity. The student is invited to apply the Agile methodology in the development of a web application.	Agile PM and SCRUM	Workshop and lecture guides	Agile PM and SCRUM.pptx

1.1.5 PLO 3. Component Integration [e-2]

3. PLO Component Integration [e-2]	
The learner has demonstrated capability *→ to integrate efficiently a software application or component into an existing system* *→ to document the installation activities*	
Unit learning outcomes	Explains and distinguishes common methods, techniques and tools related to efficient integration
	Describes the interplay between and compatibility of system components
	Carries out installation and configuration activities, applying common methods, techniques and tools related to efficient integration (e.g., packaging and distribution, virtualisation, containerisation)
	Monitors and tests the connectivity of integrated systems
	Writes an installation report

1.1.5.1 Duration of Study

Recommended duration: starting from n.4 ECTS

Often integrated with studies of PLOs: 1 - 5

1.1.5.2 Recommendations for Micro-credentials

Not Applicable

1.1.5.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Virtual Classroom ☒

Additional comments

It is recommended to deepen the topics presented in the Learning Units by reading publications dedicated to the various topics, reading websites specialized in programming, watching online tutorials and downloading materials useful for practical exercises from reliable sources.

Recommended delivery methods

- Lecture ☒ up to 60%
- Coding Training Lab delivered by individual/team project work ☒ up to 40%

1.1.5.4 WBL and Follow-up Reinforcement

After learning the basic principles, terminology, and models of Component Integration, the study should focus on analysing and simulating real work-life-like tasks as, for example, the student:

- Participates in a development process and applies a common software development method;
- Writes code and related documentation, by using a common programming language and applying coding conventions;
- Describes the interplay between and compatibility of system components;
- Monitors and tests the connectivity of integrated systems.

1.1.5.5 Important (new) approaches and technologies to consider

Not Applicable

1.1.5.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
Creates basic database design	1 exam. The candidate is asked to set up a database to support a web application	n/a
Creates a working software component/ application by applying design requirements and eventually other constraints/ requirements;	Assignment: practical activity. The student is asked to create a web application using HTML5, CSS3 and Bootstrap.	n/a
Modifies an existing software component/ application, in order to optimize application	The student is asked to optimize a web application using frontend coding and related tools.	n/a

1.1.6 Learning Resources – PLO 3. Component Integration [e-2]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Entity-Relationship Model, SQL, MySql	5	8 hours	The didactic approach would be aimed to allows participants to understand the main applications of those sets of code using practical examples, such as viewing and analyzing programming code.	1 exam. The candidate is asked to set up a database to support a web application	Entity-Relationship Model – SQL – MySql	Workshop and lecture guides	Junior Developer_EQF_4_5_Entity-Relationship Model, SQL, MySql.pptx
HTML5, CSS3, BOOTSTRAP	5	24 hours	The didactic approach would be aimed to allows participants to understand the main applications of those sets of	Assignment: practical activity. The student is asked to create a web application using HTML5, CSS3 and Bootstrap.	• HTML5 CSS3 Bootstrap	Workshop and lecture guides	Junior Developer_EQF_4_5_HTML.pptx Junior Developer_EQF_4_5_CSS.pptx

			code using practical examples, such as viewing and analyzing programming code.		• Exercise: HTML-Css-Js		
Javascript, AJAX, Typescript, GIT.	5	24 hours	The didactic approach would be aimed to allows participants to understand the main applications of those sets of code using practical examples, such as viewing and analyzing programming code.	Assignment: practical activity. The student is asked to develop a web application using frontend coding (Javascript, AJAX, Typescript) and related tools.	• Javascript – Ajax • Exercise: Java at main	Workshop and lecture guides	• Javascript – Ajax.pptx • JuniorDEVProj_Java at main · MaSTERmIKK_JuniorDEVProj · GitHub
Angular, React. Jest, Mocha, Selenium	5	24 hours	The didactic approach would be aimed to allows participants to understand the main applications of	Assignment: practical activity. The student is asked to develop a web application using Angular and tools such	Angular – React – Jest – Mocha – Selenium	Workshop and lecture guides	Angular – React – Jest – Mocha – Selenium.pptx

			those sets of code using practical examples, such as viewing and analyzing programming code.	as: React, Jest, Mocha, Selenium.			
Principles of UI/UX Design. Adobe XD, Zepling.	5	12 hours	The didactic approach would be aimed to allows participants to understand the main applications of those sets of code using practical examples, such as viewing and analyzing programming code.	Assignment: practical activity. The student is asked to build the best User Experience for the web application developed in the previous modules.	Principles of UI UX Design – Adobe XD – Zepling	Workshop and lecture guides	https://learn.softwareskills.eu/wp-content/uploads/2023/11/Principles-of-UI-UX-Design.pptx
Java 11, spring boot, spring data/ Hibernate. Ex. Java: Junit, Mockito	5	16 hours	The didactic approach would be aimed to allows participants to understand the main	Assignment: practical activity. The student is asked to build and test a backend-side application	• Backend – Java 11	Workshop and lecture guides	java.pptx

			applications of those sets of code using practical examples, such as viewing and analyzing programming code.	using Java 11 and related development tools.	• Exercise: Java at main		• JuniorDEVProj_Java at main · MaSTERmIKK_JuniorDEVProj · GitHub
BACKEND: php, Laravel, Eloquent	5	16 hours	The didactic approach would be aimed to allows participants to understand the main applications of those sets of code using practical examples, such as viewing and analyzing programming code.	Assignment: practical activity. The student is asked to build and test a backend-side application using PHP and related development tools.	PHP – Lavarel – Eloquent	Workshop and lecture guides	PHP Lavarel Eloquent.pptx
Introduction to STLC (software testing Life cycle). Unit test, end to end test.	5	8 hours	The didactic approach would be aimed to allows participants to understand the	Assignment: practical activity. The student is asked to test an application	Introduction to STLC	Workshop and lecture guides	Introduction to STLC – Unit tests – end-to-end tests.pptx

		main applications of STLC using practical examples, such as viewing and analyzing programming code.	developed in the previous modules.			

1.1.7 PLO 4. Testing [e-2]

4. PLO Testing [e-2]	
The learner has demonstrated capability *→ to test a software application or component* *→ to document test outcomes*	
Unit learning outcomes	Explains and distinguishes principles of software testing, common testing methods, techniques, and tools
	Writes an (automated) test on a piece of code
	Performs common test activities, applying testing and debugging techniques and tools
	Records and interprets test outcomes and writes test result documentation/ test report

1.1.7.1 Duration of Study

Recommended duration: starting from n. 0,5 ECTS

Often integrated with studies of PLOs: not applicable

1.1.7.2 Recommendations for Micro-credentials

Not Applicable

1.1.7.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Virtual Classroom ☒

Additional comments

n/a

Recommended delivery methods:

- Lecture ☒ up to 60%
- Coding Training Lab delivered by individual/team project work ☒ up to 40%

Additional comments

It is recommended to deepen the topics presented in the Learning Units by reading publications dedicated to software testing procedures, reading websites specialized in STLC, watching online tutorials and downloading materials useful for practical exercises from reliable sources.

1.1.7.4 WBL and Follow-up Reinforcement

After learning the basic principles, terminology, and models of Software Testing, the study should focus on analysing and simulating real work-life-like tasks as, for example:

- Define and explain appropriate test methods, techniques, and tools.
- Explain and write (parts of) testing related documentation, such as a test plan, test strategy/approach, test case, test script, test scenario, test conditions.
- Setup a test environment.

1.1.7.5 Important (new) approaches and technologies to consider

Not Applicable

1.1.7.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
Explains and writes (parts of) testing related documentation, such as a test plan, test strategy/approach, test case, test script, test scenario, test conditions.	Assignment: practical activity. The student is asked to test a web application	N/A
Configures a test environment.	Assignment: practical activity. The student is asked to test a web application	N/A
Executes associated test cases and performs test activities related to different sorts of common tests.	Assignment: practical activity. The student is asked to test a web application	N/A
Writes test result documentation/ test report.	Assignment: practical activity. The student is asked to test a web application	

1.1.8 Learning Resources – PLO 4. Testing [e-2]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Fundamentals of software testing : Introduction to STLC, Unit testing, end to end testing	5	8 hours	The didactic approach would be aimed to allows participants to understand the main applications of STLC using practical examples, such as viewing and analyzing programming code.	Assignment: practical activity. The student is asked to test an application developed in the previous modules.	Introduction to STLC – Unit test – end to and test	Workshop and lecture guides	Introduction to STLC – Unit tests – end-to-end tests.pptx

1.1.9 PLO 5. Documentation Production [e-2]

5. PLO Documentation Production [e-2]	
The learner has demonstrated capability *→ to draft technical documentation*	
Unit learning outcomes	Describes types of technical documentation
	Provides different (parts of) common technical documents, using appropriate tools (e.g., software documentation tools)

1.1.9.1 Duration of Study

Recommended duration: starting from n. 2 ECTS

Often integrated with studies of PLOs: 3 – 4 - 5

1.1.9.2 Recommendations for Micro-credentials

Not applicable

1.1.9.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Virtual Classroom ☒

Additional comments

n/a

Recommended delivery methods:

- Lecture ☒ up to 60%
- Training Lab delivered by individual/team project work ☒ up to 40%

Additional comments

It is recommended to deepen the topics presented in the Learning Units by reading publications dedicated to the various topics, reading websites specialized in software development, watching online tutorials and downloading materials useful for practical exercises from reliable sources.

1.1.9.4 WBL and Follow-up Reinforcement

After learning the basic principles, terminology, and models of Application Development, the study should focus on analysing and simulating real work-life-like tasks as, for example the student:

- Describes types of technical documentation;

- Provides different (parts of) common technical documents, using appropriate tools (e.g., software documentation tools).

1.1.9.5 Important (new) approaches and technologies to consider

Not Applicable

1.1.9.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
Specifies a design for an application or software (component), taking into account certain basic constraints/ requirements;	Assignment: practical activity. The student is asked to write a software documentation.	Specifies a design for an application or software (component), taking into account certain basic constraints/ requirements; Writes code and related documentation to it Applies version management
Writes documentation to related to the coding activity.	Assignment: practical activity. The student is asked to write a software documentation.	Specifies a design for an application or software (component), taking into account certain basic constraints/ requirements; Writes code and related documentation to it Applies version management

1.1.10 Learning Resources – PLO 5. Documentation Production [e-2]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Entity-Relationship Model, SQL, MySql	5	16 hours	The didactic approach would be aimed to allows participants to understand the main applications of those sets of code using practical examples, such as viewing and analyzing programming code.	1 exam. The candidate is asked to set up a database to support a web application	Entity-Relationship Model – SQL – MySQL	Workshop and lecture guides	Junior Developer_EQF_4_5_Entity-Relationship Model, SQL, MySql.pptx
HTML5, CSS3, BOOTSTRAP	5	24 hours	The didactic approach would be aimed to allows participants to understand the main applications of those sets of	Assignment: practical activity. The student is asked to create a web application using HTML5,	HTML5 CSS3 Bootstrap	• Workshop and lecture guides • Exercise: Html-Css-JS at main	Junior Developer_EQF_4_5_HTML.pptx Junior Developer_EQF_4_5_CSS.pptx

			code using practical examples, such as viewing and analyzing programming code.	CSS3 and Bootstrap.			
Javascript, AJAX, Typescript, GIT	5	24 hours	The didactic approach would be aimed to allows participants to understand the main applications of those sets of code using practical examples, such as viewing and analyzing programming code.		• Javascript Ajax • Java at main	Workshop and lecture guides	java.pptx • JuniorDEVProj_Java at main · MaSTERmIKK_JuniorDEVProj · GitHub

1.1.11 PLO 6. Problem management [e-2]

6. PLO Problem management [e-2]	
The learner has demonstrated capability *→ to act systematically in handling incidents and problems*	
Unit learning outcomes	Systematically resolves or escalates incidents and problems, resulting in a solved incident e.g., by applying techniques and tools for troubleshooting such as diagnostic tools

1.1.11.1 Duration of Study

Recommended duration: starting from n.0,5 ECTS

Often integrated with studies of PLOs: 4

1.1.11.2 Recommendations for Micro-credentials

Not Applicable

1.1.11.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Virtual Classroom ☒

Additional comments

Recommended delivery methods:

- Lecture ☒ up to 80%
- Training Lab delivered by individual/team project work ☒ up to 20%

Additional comments

It is recommended to deepen the topics presented in the Learning Units by reading publications dedicated to the various topics, reading websites specialized in coding and web development, watching online tutorials and downloading materials useful for practical exercises from reliable sources.

1.1.11.4 WBL and Follow-up Reinforcement

After learning the basic principles, terminology, and models of Application Development, the study should focus on analysing and simulating real work-life-like tasks as, for example, the student:

- Participates in a development process and solve common problems applied to the software development.

1.1.11.5 Important (new) approaches and technologies to consider

Not Applicable

1.1.11.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
Defines and explains appropriate test methods, techniques, and tools.	Assignment: practical activity. The student is asked to test a web application developed in the previous Learning Unit.	n/a
Explains and writes (parts of) testing related documentation, such as a test plan, test strategy/approach, test case, test script, test scenario, test conditions.	Assignment: practical activity. The student is asked to test a web application developed in the previous Learning Unit.	n/a
Configures a test environment	Assignment: practical activity. The student is asked to test a web application developed in the previous Learning Unit.	n/a
Writes test result documentation/ test report	Assignment: practical activity. The student is asked to test a web application developed in the previous Learning Unit.	n/a

1.1.12 Learning Resources – PLO 6. Problem Management [e-2]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Fundamentals of software testing: Introduction to STLC, Unit testing, end to end testing.	5	8 hours	By alternating between theory and practical activities, students will be able to understand the fundamentals of software testing, with reference to software testing life cycle and to Unit Testing, software carried out during the development of an application.	Assignment: practical activity. The student is asked to test a web application developed in the previous modules.	Introduction to STLC	Workshop and lecture guides	Introduction to STLC – Unit tests – end-to-end tests.pptx

1.1.13 PLO 7. Professional related competences [EQF5]

7. PLO Profession related competences [EQF5]	
The learner has demonstrated capability *→ to apply profession related skills*	
Unit learning outcomes	Masters common ICT knowledge
	Explains the principles, related concepts, advantages and disadvantages of a new technology. Applies and reports on basic methods, techniques and tools related to a new technology.
	Applies and reports on measures, methods, tools and techniques related to security
	Applies and reports on measures, methods, tools and techniques related to software lifecycle processes
	Is aware of basic ethical considerations and issues

1.1.13.1 Duration of Study

Recommended duration: starting from n.1 ECTS

Often integrated with studies of PLOs: 1 – 8 - 9

1.1.13.2 Recommendations for Micro-credentials

This PLO should be an integral part of the initial studies for students with no prior knowledge of professional-related competences useful to work in complex organizations embedded in innovative markets.

1.1.13.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Presence Classroom ☐
- Virtual Classroom ☒
- Blended ☐
- e-learning ☐
- Work placement ☐

Additional comments

n/a

Recommended delivery methods:

- Lecture ☒ up to 100%

Additional comments

It is recommended to deepen the topics presented in the Learning Units by reading publications dedicated to the various topics, reading websites specialized in innovation, project management and team collaboration.

1.1.13.4 WBL and Follow-up Reinforcement

Not applicable

1.1.13.5 Important (new) approaches and technologies to consider

- Business Process and Business Architecture understanding and mapping tools
- Product/service design and innovation management

1.1.13.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
Masters common ICT knowledge	1 test (multiple choice questions) on the characteristics of the main IT technologies.	n/a
Works in project settings, applies project management methods and tools	Assignment: practical activity. The student is invited to apply the Agile methodology in the development of a web application.	n/a
[Security skills] Applies and reports on methods, tools and techniques related to security	Assignment: practical activity. The student is asked to test the security of a web application	n/a
[Software life cycle skills] Applies and reports on methods, tools and techniques related to software lifecycle processes	Assignment: practical activity. The student is asked to test a web application developed in the previous modules.	n/a
[Ethical awareness skills] Is aware of basic ethical considerations and issues	1 test (multiple choice questions) on the characteristics of the main IT technologies.	n/a

1.1.14 Learning Resources – PLO 7. Profession related competence [EQF5]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Introduction to ICT and Digital Transformation tech enablers	5	4 hours	The didactic approach would be aimed to allows participants to understand ICT culture using practical example and storytelling of case histories.	1 test (multiple choice questions) on the characteristics of the main IT technologies.	Introduction to ICT and Digital Transformation tech enablers	Workshop and lecture guides	Introduction to ICT and Digital Transformation tech enablers.pptx
Agile Project Management + Scrum + collaboration tools	5	8 hours	The didactic approach would be aimed to allows participants to understand the Agile and SCRUM culture and framework using practical examples and exercises.	Assignment: practical activity. The student is invited to apply the Agile methodology in the development of a web application.	Agile PM and SCRUM	Workshop and lecture guides	Agile PM and SCRUM.pptx
Fundamentals of cybersecurity	5	8 hours		Assignment: practical activity. The student is required to take a	Fundamentals of Cybersecurity	Workshop and lecture guides	Fundamentals of Cybersecurity.pptx

				computer security test on a series of web applications.			
Introduction to STLC (software testing Life cycle). Unit test, end to end test.	5	8 hours	The didactic approach would be aimed to allows participants to understand the main applications of STLC using practical examples, such as viewing and analyzing programming code.	Assignment: practical activity. The student is asked to test a web application developed in the previous modules.		Workshop and lecture guides	Introduction to STLC – Unit tests – end-to-end tests.pptx

1.1.15 PLO 8. Soft competences [EQF5]

8. PLO Soft competences [EQF5]	
The learner has demonstrated capability *→ to apply soft skills*	
Unit learning outcomes	Works together with others in a team
	Communicates with peers, colleagues, supervisors and/or relevant others, appropriately to the context, using conventions that are relevant to professional practice. Explains and gives instruction.
	Masters the English language at level B2. Can understand the main ideas of complex text on both concrete and abstract topics, including technical discussions in his/her field of specialisation
	Distinguishes and analyses fairly complex and unpredictable problems. Solves these problems systematically and in a creative way, using existing procedures and guidelines and own solutions by identifying and using data.
	Exercises self-management within the guidelines of contexts that are usually predictable, but are subject to change. Is able to cope with limited change and to adapt to a certain level of variety in the workplace. Copes with pressure and stress setbacks and maintains composure. Shows some initiative and carries responsibility for the results of own activities, work and or study. Works correctly and carefully.
	Realises learning and personal development on request, where necessary with support, through self-reflection and external- and self-evaluation of own (learning) results.

1.1.15.1 Duration of Study

Recommended duration: starting from n.0,5 ECTS

Often integrated with studies of PLOs: 9

1.1.15.2 Recommendations for Micro-credentials

This PLO should be an integral part of the initial studies for students with no prior knowledge of professional-related competences useful to work in complex organizations embedded in innovative business.

1.1.15.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Presence Classroom ☐
- Virtual Classroom ☒
- Blended ☐
- e-learning ☐
- Work placement ☐

Additional comments

n/a

Recommended delivery methods:

- Lecture ☒ up to 100%

Additional comments
It is recommended to deepen the topics presented in the Learning Units by reading publications dedicated to the various topics, reading websites specialized in programming, watching online tutorials and downloading materials useful for practical exercises from reliable sources.

1.1.15.4 WBL and Follow-up Reinforcement

Not applicable

1.1.15.5 Important (new) approaches and technologies to consider

Not applicable

1.1.15.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
Teamwork skills] Manages teamwork processes and facilitates collaboration to reach common objectives, e.g., handles conflicts, negotiates, motivates, and persuades.	The student is called to work on a development project, creating a work group and collaborating with other people through the use of enterprise social collaboration tools.	n/a
Communication skills] Communicates with peers, colleagues, supervisors and or relevant other, specialists and non-specialists, and clients, appropriately to the scientific and professional community, using conventions which are relevant. Applies communication to the objective and the target group.	The student is called to work on a development project, creating a work group and collaborating with other people through the use of enterprise social collaboration tools. The student is called to work on a development project, creating a work group and collaborating with other people through the use of enterprise social collaboration tools.	n/a
Problem solving skills] Identifies and analyses complex and unpredictable problems Solves these problems in a tactical, strategic and creative way by selecting and using data and by using one's creativity, flexibility and inventiveness.	The student is called to work on a development project, creating a work group and collaborating with other people through the use of enterprise social collaboration tools.	n/a
[Self-management skills] Realises personal development on one's own initiative, by reflecting on and evaluating personal (learning) results.	The student is called to work on a development project, creating a work group and collaborating with other people through the use of enterprise social collaboration tools.	n/a

1.1.16 Learning Resources – PLO 8. Soft competences [EQF5]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Team Collaboration (soft Skill)	5	8 hours	The didactic approach would be aimed to allows participants to understand the main team collaboration principles and practices, using practical examples, case study analysis and exercises.	Assignment: practical activity. The student is called to work on a development project, creating a work group and collaborating with other people through the use of enterprise social collaboration tools.	Team Collaboration and file versioning	• Formative quizzes • Workshop and lecture guides	Team Collaboration and file versioning.pptx
Introduction to STLC (software testing Life cycle)	5	8 hours	The didactic approach would be aimed to allows participants to understand the main applications of STLC using practical examples, such as viewing and	Assignment: practical activity. The student is asked to test a web application developed in the previous modules.	Introduction to STLC	• Formative quizzes • Workshop and lecture guides	Introduction to STLC – Unit tests – end-to-end tests.pptx

		analyzing programming code.				

1.1.17 PLO 9. Functionning in organisations [EQF5]

9. PLO Functioning in organisations [EQF5]	
The learner has demonstrated capability *→ to function in an organisational context*	
Unit learning outcomes	Explains the basics of organisation theory and behaviour
	Describes the relationship between business and IT
	Works in an organisational context under specific direction with limited autonomy and responsibility e.g., at the level of a trainee, junior or assistant
	Works in project settings, applies project management methods and tools
	Writes a report on functioning in the organisation

1.1.17.1 Duration of Study

Recommended duration: starting from n.0,5 ECTS

Often integrated with studies of PLOs: 8

1.1.17.2 Recommendations for Micro-credentials

This PLO should be an integral part of the initial studies for students with no prior knowledge of teamwork collaboration.

1.1.17.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Virtual Classroom ☒

Additional comments

n/a

Recommended delivery methods:

- Lecture ☒ up to 100%

Additional comments

It is recommended to deepen the topics presented in the Learning Units by reading publications dedicated to the various topics, reading websites specialized in teamwork collaboration and communication & collaboration tools and platforms.

1.1.17.4 WBL and Follow-up Reinforcement

Not Applicable

1.1.17.5 Important (new) approaches and technologies to consider

Not Applicable

1.1.17.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
Explains basics of organisation theory and behaviour	Assignment: practical activity. The student is called to work on a development project, creating a work group and collaborating with other people through the use of enterprise social collaboration tools.	n/a
Works in an organisational context under specific direction with limited autonomy and responsibility	Assignment: practical activity. The student is called to work on a development project, creating a work group and collaborating with other people through the use of enterprise social collaboration tools.	n/a

1.1.18 Learning Resources – 9. PLO Functionning in organisation [EQF5]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Team Collaboration	5	8 hours	The didactic approach would be aimed to allows participants to understand the main team collaboration principles and practices, using practical examples, case study analysis and exercises.	Assignment: practical activity. The student is called to work on a development project, creating a work group and collaborating with other people through the use of enterprise social collaboration tools.	Team collaboration and file versioning	Workshop and lecture guides	Team Collaboration and file versioning.pptx
ICT and Digital Transformation tech enablers	5	8 hours	The didactic approach would be aimed to allows participants to understand ICT culture using practical example and storytelling of case histories.	1 test (multiple choice questions) on the characteristics of the main IT technologies.	Introduction to ICT and Digital Transformation Tech Enablers	Workshop and lecture guides	Introduction to ICT and Digital Transformation tech enablers.pptx

1.2 Unemployed adults and young aged 16-29 y.o.

Executive summary

The Learning Programme is being designed by BCS Koolitus Training (ESTONIA) for beginners taking their first steps to become a Junior software developer. The course provides learners knowledge and skills to create a simple web application. The curriculum is part of Jr Developer EQF4 level.

It provides learners (also in terms of reskilling programme) with different level and previous experience (i.e., everybody, who has interest and some initial experience in software development or web development) knowledge and skills to create a simple web application.

Targeted Institutions: Higher Education and VET providers.

Requirements for starting studies:

- belongs to the target group of the project - an adult who has been unemployed and inactive for at least six months or a young person aged 16 to 29 years who is not currently working or studying;
- suitable for the project based on motivation;
- has a desire to take up a job in a position that requires the professional skills of a web developer.

The training course consists of 14 units:

- Software development - introduction into programming, software development methods, agile development methods
- Basic web technologies: (HTML; CSS; JavaScript; Php; MySQL)
- Website visualization and prototyping, UI/UX
- Servers and network management
- Teamwork - how to work in teams, teamwork basics, best practices
- Work clubs - developing social and communication skills when applying for a job and in the work environment
- Mentoring - learners work with their own project individually or in teams

Total: 7 ECTS (160 hours)

Delivery methods are presence in classroom, virtual classroom, blended learning.

1.2.1 PLO 1. Application Design [e-2]

1. PLO Application Design [e-2]	
The learner has demonstrated capability *→ to interpret a design for a software application or component*	
Unit learning outcomes	Explains and distinguishes basic principles and terminology of software design (e.g., phases in the design process, common techniques, deliverables)
	Describes principles of user interface design
	Reads design models and diagrams (e.g., ERD, UML)
	Interprets a basic database design
	Interprets a design for an application or software component

1.2.1.1 Duration of Study

Recommended duration: starting from 0.5 ECTS.

Often integrated with studies of PLOs: n/a

1.2.1.2 Recommendations for Micro-credentials

This course is designed as one micro-credential course and it cannot be split into smaller integral units.

1.2.1.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Presence Classroom ☒
- Virtual Classroom ☒
- Blended ☒

Additional comments

Recommended delivery methods:

- Lecture ☒ up to 20%
- Practical tasks and exercises ☒ up to 80%

Additional comments

Lectures, e-learning are recommended for learning the basic principles, terminology, and models of software design. These should be reinforced through practical tasks, case studies, individual/team-projects.

1.2.1.4 WBL and Follow-up Reinforcement

After learning the basic principles, terminology, and models of software design, the study should focus on analysing and simulating real work-life-like tasks as, for example:

- Designing and creating different webpage examples as real-life-like customer project

1.2.1.5 Important (new) approaches and technologies to consider

n/a

1.2.1.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
Explains and distinguishes basic principles and terminology of software design (e.g., phases in the design process, common techniques, deliverables)	Test	n/a
Describes principles of user interface design	Practical exercises	n/a
Reads design models and diagrams (e.g., ERD, UML)	Practical exercises	n/a
Interprets a basic database design	Practical exercises	n/a

1.2.2 Learning Resources - PLO 1. Application Design [e-2]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Basics of programming	4	4 hours	Blended	Report	Agile Development	Lecture (80%) Discussion (20%)	https://learn.softwareskills.eu/wp-content/uploads/2023/07/Developer-4-5-Agile-development.pdf
CSS3	4	8 hours	Blended	Practical tasks	CSS	Practical exercises	https://learn.softwareskills.eu/wp-content/uploads/2023/12/Junior-Developer_EQF_4_5_CSS.pptx
Creating User Stories	4	4 hours	Blended	Practical tasks	User stories	Teamwork	https://learn.softwareskills.eu/wp-content/uploads/2023/07/Developer-4-5-User-Stories-1.pdf

1.2.3 PLO 2. Application Development [e-2]

2. PLO Application Development [e-2]	
The learner has demonstrated capability *→ to systematically develop a simple software application or component* *→ to propose modifications to an existing solution* *→ to document the development activities*	
Unit learning outcomes	Explains and distinguishes common software development methods (e.g., waterfall, iterative, agile), techniques (e.g., object-oriented) and tools (e.g., IDE, CASE; multimedia integration tools; app development tools)
	Describes common programming principles and terminology (e.g., secure programming)
	Explains concepts and principles of databases, data structures and query languages (e.g., SQL)
	Participates in a development process and applies a common software development method (e.g., agile)
	Creates a simple relational database
	Writes code and related documentation to it, by using a common programming language and applying coding conventions (e.g., Java, Javascript, PHP, Python; clean coding principle)
	Creates a simple working software component or application, taking into account architecture, design requirements and other possible constraints (e.g., installability) applying relevant tools and techniques (e.g., object-oriented programming; IDE, CASE; editors, compilers, version control tools)
	Modifies an existing software application or component

1.2.3.1 Duration of Study

Recommended duration: starting from n.3 ECTS

Often integrated with studies of PLOs: n/a

1.2.3.2 Recommendations for Micro-credentials

This course is designed as one micro-credential course and it cannot be split into smaller integral units.

1.2.3.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Presence Classroom ☒
- Virtual Classroom ☒
- Blended ☒

Additional comments
n/a

Recommended delivery methods:

• Lecture	☒ up to 15 %
• Practical exercises	☒ up to 85 %

Additional comments

Lectures, e-learning are recommended for learning the basic principles, terminology, and models of software design. These should be reinforced through practical tasks, case studies, individual/team-projects.

1.2.3.4 WBL and Follow-up Reinforcement

After learning the basic principles, terminology, and models of software design, the study should focus on analysing and simulating real work-life-like tasks as, for example:

- Designing and creating different web application examples as real-life-like customer project

1.2.3.5 Important (new) approaches and technologies to consider

Not Applicable

1.2.3.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
Explains and distinguishes common software development methods (e.g., waterfall, iterative, agile), techniques (e.g., object-oriented) and tools (e.g., IDE, CASE; multimedia integration tools; app development tools)	Test	n/a
Describes common programming principles and terminology (e.g., secure programming)	Test	n/a
Explains concepts and principles of databases, data structures and query languages (e.g., SQL)	Practical tasks	n/a
Participates in a development process and applies a common software development method (e.g., agile)	Practical tasks	
Creates a simple relational database	Practical tasks	n/a
Writes code and related documentation to it, by using a common programming language and applying coding conventions (e.g., Java, Javascript, PHP, Python; clean coding principle)	Practical tasks	n/a
Creates a simple working software component or application, taking into account architecture, design requirements and other possible constraints (e.g., installability) applying relevant tools and techniques (e.g., object-oriented programming; IDE, CASE; editors, compilers, version control tools)	Practical tasks	n/a
Modifies an existing software application or component	Practical tasks	n/a

1.2.4 Learning Resources - PLO 2. Application Development [e-2]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Basics of programming	*4*	4 hours	Blended	*Discussion*		Lecture (80%) Discussion (20%)	
Agile development methodologies	4	4 hours	Blended			Lecture (80%) Discussion (20%)	https://learn.softwareskills.eu/wp-content/uploads/2023/07/Developer-4-5-Agile-development.pdf
HTML5	4	8 hours	Blended	Practical tasks	HTML	Practical exercises	https://learn.softwareskills.eu/wp-content/uploads/2023/07/Developer-4-5-HTML5.pdf
JavaScript	4	12 hours	Blended	Practical tasks	JavaScript	Practical exercises	https://learn.softwareskills.eu/wp-content/uploads/2023/12/ESSA-Junior-Devloper_JAVA.pptx
Php	4	8 hours	Blended	Practical tasks	PHP	Practical exercises	https://learn.softwareskills.eu/wp-content/uploads/2023/12/ESSA-Junior-Developer_PHP.pptx
MySQL	4	8 hours	Blended	Practical tasks	SQL	Practical exercises	https://learn.softwareskills.eu/wp-content/uploads/2023/07/Developer-4-5-SQL.pdf

1.2.5 PLO 4. Testing [e-2]

4. PLO Testing [e-2]	
The learner has demonstrated capability *→ to test a software application or component* *→ to document test outcomes*	
Unit learning outcomes	Explains and distinguishes principles of software testing, common testing methods, techniques, and tools
	Writes an (automated) test on a piece of code
	Performs common test activities, applying testing and debugging techniques and tools
	Records and interprets test outcomes and writes test result documentation/ test report

1.2.5.1 Duration of Study

Recommended duration: starting from n.1 ECTS

Often integrated with studies of PLOs: n/a

1.2.5.2 Recommendations for Micro-credentials

This course is designed as one micro-credential course and it cannot be split into smaller integral units.

1.2.5.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Presence Classroom ☒
- Virtual Classroom ☒
- Blended ☒

Additional comments

Recommended delivery methods:

Case study. Individual/team project ☒ integrated with other modules

Additional comments

Lectures, e-learning are recommended for learning the basic principles, terminology, and models of software design. These should be reinforced through practical tasks, case studies, individual/team-projects.

1.2.5.4 WBL and Follow-up Reinforcement

- Designing and creating different web application examples as real-life-like customer project

1.2.5.5 Important (new) approaches and technologies to consider

Not Applicable

1.2.5.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
Explains and distinguishes principles of software testing, common testing methods, techniques, and tools	Test, practical tasks	n/a

1.2.6 Learning Resources - PLO 4. Testing [e-2]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Basics of programming	*4*	4 hours	*Blended*	*Test*		Lecture (80%) Discussion (20%)	https://learn.softwareskills.eu/wp-content/uploads/2023/07/Developer-4-5-Agile-development.pdf
JavaScript	*4*	12 hours	*Blended*	*Practical tasks*	*JavaScript*	*Practical exercises*	https://learn.softwareskills.eu/wp-content/uploads/2023/11/plo3Javascript-Ajax.pptx
Php	*4*	8 hours	*Blended*	*Practical tasks*	*PHP*	*Practical exercises*	https://learn.softwareskills.eu/wp-content/uploads/2023/07/Developer-4-5-PHP.pdf

1.2.7 PLO 8. Soft competences [EQF5]

8. PLO Soft competences [EQF5]	
The learner has demonstrated capability *→ to apply soft skills*	
Unit learning outcomes	Works together with others in a team
	Communicates with peers, colleagues, supervisors and/or relevant others, appropriately to the context, using conventions that are relevant to professional practice. Explains and gives instruction.
	Masters the English language at level B2. Can understand the main ideas of complex text on both concrete and abstract topics, including technical discussions in his/her field of specialisation
	Distinguishes and analyses fairly complex and unpredictable problems. Solves these problems systematically and in a creative way, using existing procedures and guidelines and own solutions by identifying and using data.
	Exercises self-management within the guidelines of contexts that are usually predictable, but are subject to change. Is able to cope with limited change and to adapt to a certain level of variety in the workplace. Copes with pressure and stress setbacks and maintains composure. Shows some initiative and carries responsibility for the results of own activities, work and or study. Works correctly and carefully.
	Realises learning and personal development on request, where necessary with support, through self-reflection and external- and self-evaluation of own (learning) results.

1.2.7.1 Duration of Study

Recommended duration: integrated with all the other study modules

Often integrated with studies of PLOs: n/a

1.2.7.2 Recommendations for Micro-credentials

Not Applicable

1.2.7.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Blended ☒

Additional comments

n/a

Recommended delivery methods:

- Teamwork ☒

Additional comments

Lectures, e-learning are recommended for learning the basic principles, terminology, and models of software design. These should be reinforced through practical tasks, case studies, individual/team-projects.

1.2.7.4 WBL and Follow-up Reinforcement

Related with other study units.

1.2.7.5 Important (new) approaches and technologies to consider

Not Applicable

1.2.7.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
Works together with others in a team	Practical tasks	n/a
Communicates with peers, colleagues, supervisors and/or relevant others, appropriately to the context, using conventions that are relevant to professional practice. Explains and gives instruction.	Project	n/a

1.2.8 Learning Resources - PLO 8. Soft competences [EQF5]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Teamwork	*4*	8 hours	*Blended*	Self-reflection report	General skills	Teamwork	https://learn.softwareskills.eu/wp-content/uploads/2024/01/General-skills.pptx

1.3 Workers in upskilling/reskilling paths

Executive summary

The Learning Programme is being designed by Digital Technology Skills Limited (IE).

The Junior Developer Course, a 22-week programme, crafted for employed professionals looking to reskill and pursue a career in software development. Delivered by an experienced academic, this course seamlessly integrates online learning with in-person sessions at a university campus.

The programme is structured into nine learning units, each carefully designed to provide a comprehensive foundation in software development. Participants will have the opportunity to engage in both collaborative group work and guided individual learning, ensuring a well-rounded and tailored educational experience.

This 22-week program, led by an academic professor, is tailored for entry-level junior developers. It serves as an accelerated pathway for employed individuals seeking to reskill for a career in software development or enhance their existing roles with valuable software development skills. The program focuses on providing a comprehensive foundation in coding and design, equipping participants with the expertise needed in the dynamic field of software development.

Candidates must be at least 18 years old and be unemployed

Applicants must be Irish or EU/EEA nationals and must be resident in the Republic of Ireland

Targeted Institutions: Higher Education and VET providers.

1.3.1 PLO 1. Application Design [e-2]

1. PLO Application Design [e-2]	
The learner has demonstrated capability *→ to interpret a design for a software application or component*	
Unit learning outcomes	Explains and distinguishes basic principles and terminology of software design (e.g., phases in the design process, common techniques, deliverables)
	Describes principles of user interface design
	Reads design models and diagrams (e.g., ERD, UML)
	Interprets a basic database design
	Interprets a design for an application or software component

1.3.1.1 Duration of Study

Recommended duration: 16 hours

Often integrated with studies of PLOs: PLO2 Application Development

1.3.1.2 Recommendations for Micro-credentials

One distinct unit – possibly suitable for micro credentials

1.3.1.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Presence Classroom ☒
- Virtual Classroom ☒
- e-learning ☒

Additional comments

n/a

Recommended delivery methods:

- Lecture ☒ 50%
- Case study. Individual/team project ☒ 30%
  e-Learning ☒ 20%

Additional comments

Lectures, e-learning are recommended for learning the basic principles, terminology, and models of software application design. These are reinforced through two assignment options.

1.3.1.4 WBL and Follow-up Reinforcement

After learning the basic principles, terminology, and models of software design, the study should focus on analysing and simulating real work-life-like tasks as, for example:

The combination of both assignments provides an opportunity for students' to engage in a real life customer based project.

1.3.1.5 Important (new) approaches and technologies to consider

Including a guest speaker from industry or academic would enhance the learners knowledge and afford them the opportunity to ask questions and gain insights into the role of a software developer in a business context.

1.3.1.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
Explains and distinguishes basic principles and terminology of software design (e.g., phases in the design process, common techniques, deliverables)	Design Process Assignment	Validates knowledge acquired of application design principles
Describes principles of user interface design	Design an App	Validates knowledge of acquired application design principles and the usage of design tools. Also affords students' the

	opportunity to present their results (transversal skills validation)

1.3.2 Learning Resources - 1. PLO 1. Application Design [e-2]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Application Design		14 hours or 18 hours if both assessments are used	Lecture, e-Learning and assignment(s)	Design Process Assignment Design an Application Assignment	ESSA_Junior_Developer_Application_Design.pptx	Tutor Led	https://learn.softwareskills.eu/wp-content/uploads/2024/01/ESSA_Junior_Developer_Application_Design-1-2.pdf
Application Design				Recommended Reading	**Reading Materials & Supports** **9 Key Elements of Application Design** https://centogram.com/2022/01/28/9-key-elements-of-application-design/ **Software Architecture and Software Design** https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3772387 **How has design thinking being used and integrated into software development activities? A systematic mapping** https://www.sciencedirect.com/science/article/abs/pii/S0164121222000024	e-Learning	

1.3.3 PLO 2. Application Development [e-2]

2. PLO Application Development [e-2]	
The learner has demonstrated capability *→ to systematically develop a simple software application or component* *→ to propose modifications to an existing solution* *→ to document the development activities*	
Unit learning outcomes	Explains and distinguishes common software development methods (e.g., waterfall, iterative, agile), techniques (e.g., object-oriented) and tools (e.g., IDE, CASE; multimedia integration tools; app development tools)
	Describes common programming principles and terminology (e.g., secure programming)
	Explains concepts and principles of databases, data structures and query languages (e.g., SQL)
	Participates in a development process and applies a common software development method (e.g., agile)
	Creates a simple relational database
	Writes code and related documentation to it, by using a common programming language and applying coding conventions (e.g., Java, Javascript, PHP, Python; clean coding principle)
	Creates a simple working software component or application, taking into account architecture, design requirements and other possible constraints (e.g., installability) applying relevant tools and techniques (e.g., object-oriented programming; IDE, CASE; editors, compilers, version control tools)
	Modifies an existing software application or component

1.3.3.1 Duration of Study

Recommended duration: 140 hours

Often integrated with studies of PLOs: PLO1 Application Design, PLO3 Component Integration, PLO4 Testing, PLO5 Documentation Production, PLO6 Problem Management, PLO New Technology, PLO7Profession Related Competences, PLO9 Functioning in Organisations).

1.3.3.2 Recommendations for Micro-credentials

May be suitable for micro-credentials in specific application development skills

1.3.3.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Presence Classroom ☒
- Virtual Classroom ☒
- e-learning ☒

Additional comments

n/a

Recommended delivery methods:

- Lecture ☒ 70%
- Case study. Individual/team project ☒ 20%
- eLearning ☒ 10%

Additional comments
Lectures and e-learning are recommended for learning the basic principles and terminology associated with application design. By engaging in one or both assignments, students' have the opportunity to put the learning in this unit into practice.

1.3.3.4 WBL and Follow-up Reinforcement

Through the Industry Related Software Development Project (Software Pathway Project) students' have the opportunity to engage in a coding project which mirrors a software project within a work environment and build on their learning through this unit of application design.

1.3.3.5 Important (new) approaches and technologies to consider

Including a guest speaker from industry or academic would enhance the learners knowledge and afford them the opportunity to ask questions and gain insights into the role of a software developer in a business context.

1.3.3.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
Explains and distinguishes common software development methods (e.g., waterfall, iterative, agile), techniques (e.g., object-oriented) and tools (e.g., IDE, CASE; multimedia integration tools; app development tools)	Software Coding Exercise (Python)	Modify existing python code snippet – testing knowledge and understanding of the coding environment and the code itself
Describes common programming principles and terminology (e.g., secure programming)	Software Pathway Project (Industry Project)	Validates software design, development, implementation, testing and deployment

1.3.4 Learning Resources - PLO 2. Application Development [e-2]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Application Development		30 hours	Lecture	Python Assignment	ESSA_Junior_Developer_application_Development.pptx	Lecture, e-Learning	https://learn.softwareskills.eu/wp-content/uploads/2024/01/ESSA_Junior_Developer_Application_Development.pdf
Application Development		24 hours	Lecture		04_HTML5 CSS3 Bootstrap.ppx	Lecture	https://learn.softwareskills.eu/wp-content/uploads/2023/11/plo3HTML5-CSS3-Bootstrap.pptx
Application Development		24 hours	Lecture		05_Javascript-Ajax.pptx	Lecture	https://learn.softwareskills.eu/wp-content/uploads/2023/07/Developer-4-5-JavaScript.pdf
Application Development		24 hours	Lecture		06_Angular – React – Jest – Mocha - Selenium	Lecture	https://learn.softwareskills.eu/wp-content/uploads/2023/11/Angular-React-Jest-Mocha-Selenium.pptx
Application Development		16 hours	Lecture		08_Backend-Java 11.pptx	Lecture	https://learn.softwareskills.eu/wp-content/uploads/2023/

							07/Developer-4-5-JavaScript.pdf
Application Development		16 hours	Lecture		09_PHP Lavarel Eloquent_.pptx	Lecture	https://learn.softwareskills.eu/wp-content/uploads/2023/07/Developer-4-5-PHP.pdf
Application Development		24 hours	Lecture		10_Back end – Objects – Ruby – Python - NodeJS.pptx	Lecture	https://learn.softwareskills.eu/wp-content/uploads/2023/11/Backend-Java-11.pptx
Application Development		6 hours	eLearning	None	**Reading Materials & Supports** PEP 8 – Style Guide for Python Code https://peps.python.org/pep-0008/Conduct one-on-one interviews with users to observe their interactions with the product How to Write Beautiful		

					Python Code With PEP 8 https://realpython.com/python-pep8/ Lesson 2. Clean Code Syntax for Python: Introduction to PEP 8 Style Guide https://www.earthdatascience.org/courses/intro-to-earth-data-science/write-efficient-python-code/intro-to-clean-code/python-pep-8-style-guide/ **Noteworthy 30+ Learning Resources For Developers 2023** https://dev.to/theme_selection/learning-		

					resources-for-developers-165d **3 Must Have Tools for Every Junior Software Developer** https://launchacademy.com/blog/3-must-have-tools-for-every-junior-software-developer **Codecademy**: Offers interactive coding courses in various programming languages. FreeCodeCamp: Provides free online coding lessons and coding challenges. Udemy: Offers a wide range of programming courses taught by industry experts.		

					Coursera: Provides online courses from top universities and institutions		
Industry Related Software Development Project		4 weeks	Assignment	Industry Project	ESSA_Software_Pathway_Project_Junior_Developer_(EQF4_7).docx	Tutor Assigned	n/a

1.3.5 PLO 3. Component Integration [e-2]

3. PLO Component Integration [e-2]	
The learner has demonstrated capability *→ to integrate efficiently a software application or component into an existing system* *→ to document the installation activities*	
Unit learning outcomes	Explains and distinguishes common methods, techniques and tools related to efficient integration
	Describes the interplay between and compatibility of system components
	Carries out installation and configuration activities, applying common methods, techniques and tools related to efficient integration (e.g., packaging and distribution, virtualisation, containerisation)
	Monitors and tests the connectivity of integrated systems
	Writes an installation report

1.3.5.1 Duration of Study

Recommended duration: 8 hours

Often integrated with studies of PLOs: Aligns with end of year Project PLOs 2, 4, 7, 8, 9

1.3.5.2 Recommendations for Micro-credentials

- May be suitable for micro-credential

1.3.5.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Presence Classroom ☒
- Virtual Classroom ☒
- e-learning ☒

Additional comments

n/a

Recommended delivery methods:

- Lecture ☒ up to 50%
- eLearning/self-study ☒ up to 50%

Additional comments

Lectures and e-learning are recommended for learning the basic principles, terminology, and models component integration. These are reinforced through practical in-class assignments and through the Industry Software Project referenced in PLOs 7,8,9.

1.3.5.4 WBL and Follow-up Reinforcement

After learning the basic principles, terminology, and models of software component architectures, the knowledge gained can form part of the final year project

1.3.5.5 Important (new) approaches and technologies to consider

Including a guest speaker from industry or academic would enhance the learners knowledge and afford them the opportunity to ask questions and gain insights into the role of a software developer in a business context.

1.3.5.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
Explains and distinguishes common methods, techniques and tools related to efficient integration	Written Assignment (10 Questions to be answered)	Assesses key concepts related to efficient software component integration, including methods, techniques, architectures, tools, and best practices
Describes the interplay between and compatibility of system components	Component Installation and Configuration Multiple Choice Quiz	Eight multiple choice questions to assess knowledge and understanding of component installation and configuration

1.3.6 Learning Resources - PLO 3. Component Integration [e-2]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Component Integration		8 hours	Virtual Classroom	Assignment – 10 Questions & Multiple choice quiz	*ESSA_Junior_Developer_COMPONENT_INTEGRATION.pptx* *ESSA_Learning_Programmes_and_Materials_JUNIOR DEVELOPER_(EQF4_7).docx*	Virtual Classroom via Tutor	https://learn.softwareskills.eu/wp-content/uploads/2023/11/ESSA_Learning-Programmes-and-Materials-EQF-4-7-Component-Integration.pptx
Component Integration			e-Learning		*ESSA_Junior_Developer_COMPONENT_INTEGRATION.pptx* **Recommended Reading:** "Software Architecture in Practice" by Len Bass, Paul Clements, and Rick Kazman: This book provides an overview of software architecture, including different architectural styles and patterns. It also covers topics such as architectural design, documentation, and evaluation "Building Microservices: Designing Fine-Grained Systems" by Sam Newman: This book focuses on microservices architecture, including design principles, implementation strategies, and deployment techniques "Enterprise Integration Patterns" by Gregor Hohpe and Bobby Woolf: This book provides an overview of	e-Learning	https://learn.softwareskills.eu/wp-content/uploads/2023/11/ESSA_Learning-Programmes-and-Materials-EQF-4-7-Component-Integration.pptx

					different integration patterns and technologies, including messaging systems, service-oriented architecture, and enterprise application integration "Kubernetes in Action" by Marko Luksa: This book provides an introduction to Kubernetes, including installation, deployment, and management of containerized applications Docker Deep Dive" by Nigel Poulton: This book provides a comprehensive overview of Docker, including containerization concepts, image creation, and container management "Cloud Native Infrastructure: Patterns for Scalable Infrastructure and Applications in a Dynamic Environment" by Justin Garrison and Kris Nova: This book covers the basics of cloud-native infrastructure, including containerization, microservices, and serverless architecture "Service-Oriented Architecture: Concepts, Technology, and Design" by Thomas Erl: This book provides a comprehensive overview of service-oriented architecture, including design principles, implementation strategies, and deployment techniques "The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations" by Gene Kim, Jez Humble, Patrick		

					Debois, and John Willis: This book covers the principles and practices of DevOps, including continuous integration, delivery, and deployment "The Practice of System and Network Administration" by Thomas A. Limoncelli, Christina J. Hogan, and Strata R. Chalup: This book provides a comprehensive guide to system and network administration, including installation and configuration of software component "Windows Internals, Part 1: System architecture, processes, threads, memory management, and more" by Mark E. Russinovich and David A. Solomon: This book provides a detailed look at the Windows operating system's internals, including the installation and configuration of software component "Linux Administration Handbook" by Evi Nemeth, Garth Snyder, Trent R. Hein, and Ben Whaley: This book provides a comprehensive guide to Linux system administration, including the installation and configuration of software components "Cloud Computing: From Beginning to End" by Ray J. Rafaels: This book provides an overview of cloud computing, including the installation and configuration of software components in cloud environments		

					"Network Warrior: Everything You Need to Know That Wasn't on the CCNA Exam" by Gary A. Donahue: This book provides a practical guide to network administration, including the installation and configuration of software components on networks "Software Interoperability: Frameworks for Addressing Challenges" by Arunava Paul, Nilanjan Banerjee, and Sajal K. Das (IEEE Communications Surveys & Tutorials, 2012). "A Model-Based Approach to the Design of Interoperable Software Systems" by Hassan Gomaa (IEEE Transactions on Software Engineering, 2004) "Interoperability in Healthcare Information Systems: Standards, Approaches, and Challenges" by Miltiadis D. Lytras, Ernesto Damiani, and Patricia Ordóñez de Pablos (IGI Global, 2010) "Interoperability of Enterprise Software and Applications" edited by Yingxu Wang, Athanasios Tsadiras, and Richard Hill (Springer, 2005). "The Importance of Interoperability in the Age of the Cloud" by Michael Mimoso (Communications of the ACM, 2011).		

1.3.7 PLO 4. Testing [e-2]

4. PLO Testing [e-2]	
The learner has demonstrated capability *→ to test a software application or component* *→ to document test outcomes*	
Unit learning outcomes	Explains and distinguishes principles of software testing, common testing methods, techniques, and tools
	Writes an (automated) test on a piece of code
	Performs common test activities, applying testing and debugging techniques and tools
	Records and interprets test outcomes and writes test result documentation/ test report

1.3.7.1 Duration of Study

Recommended duration: 24 hours

Often integrated with studies of PLOs: PLO2 Application Development, PLO3 Component Integration, PLO7 Profession Related Competencies, PLO8 Soft Competencies and PLO9 Functioning in Organisations.

1.3.7.2 Recommendations for Micro-credentials

- *Potential to be considered for Micro-credentials.*

1.3.7.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Presence Classroom ☒
- Virtual Classroom ☒
- e-learning ☒

Additional comments

n/a

Recommended delivery methods:

- Lecture ☒ up to 70%
- Case study. Individual/team project ☒ 15%
- e-Learning ☒ 15%

Additional comments

Lectures and e-learning are supplemented with two assignments. It is advisable that students' practice software testing to gain competency in automation tools and techniques.

1.3.7.4 WBL and Follow-up Reinforcement

After learning the basic principles, terminology, models and tools associated with software testing, the knowledge gained can form part of the final year project.

1.3.7.5 Important (new) approaches and technologies to consider

Including a guest speaker from industry or academic would enhance the learners knowledge and afford them the opportunity to ask questions and gain insights into the role of a software developer in a business context.

1.3.7.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
Explains and distinguishes principles of software testing, common testing methods, techniques, and tools	10 Questions	Assesses student's understanding and knowledge of testing principles and methods
Writes an (automated) test on a piece of code	Test Automation Assignment	Assesses the student's practical application of the knowledge, methods and tools associated with software testing

1.3.8 Learning Resources - PLO 4. Testing [e-2]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Software Testing		24	Lecture & eLearning	Two assignments	ESSA_Junior_Developer_SOFTWARE_TESTING.pptx	Tutor and eLearning	https://learn.softwareskills.eu/wp-content/uploads/2023/12/ESSA-Test-Specialist_Fundamentals-of-software-testing.pptx
Software Testing					**Optional Additional Reading**: • Effective Testing with RSpec 3: Build Ruby Apps with Confidence" by Myron Marston and Ian Dees. This book covers the basics of testing with RSpec, a popular testing framework for Ruby applications. • "Testing Python: Applying Unit Testing, TDD, BDD and Acceptance Testing" by David Sale. This book covers various testing techniques for Python applications, including unit testing, test-driven development (TDD), behavior-driven development (BDD), and acceptance testing. • "Agile Testing: A Practical Guide for Testers and Agile Teams" by Lisa Crispin and Janet Gregory. This book provides an overview of agile testing, including how it differs from traditional testing methods and best		https://learn.softwareskills.eu/wp-content/uploads/2023/12/ESSA-Test-Specialist_Fundamentals-of-software-testing.pptx

					practices for testing in an agile environment. • "The Art of Unit Testing: With Examples in .NET" by Roy Osherove. This book covers the basics of unit testing with examples in .NET, including how to write effective unit tests and how to use mocking frameworks to isolate dependencies. • "Selenium WebDriver Recipes in C#: Second Edition" by Zhimin Zhan. This book provides practical examples of using Selenium WebDriver, a popular automated testing tool, to test web applications in C#.		

1.3.9 PLO 5. Documentation Production [e-2]

5. PLO Documentation Production [e-2]	
The learner has demonstrated capability *→ to draft technical documentation*	
Unit learning outcomes	Describes types of technical documentation
	Provides different (parts of) common technical documents, using appropriate tools (e.g., software documentation tools)

1.3.9.1 Duration of Study

Recommended duration: 12 hours

Often integrated with studies of PLOs: PLOs 2 Application Development, PLOs 7,8,9

1.3.9.2 Recommendations for Micro-credentials

- *May be suitable for Micro-credentials*

1.3.9.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Presence Classroom ☒
- Virtual Classroom ☒
- e-learning ☒

Additional comments

n/a

Recommended delivery methods:

- Lecture ☒ 70%
- Case study. Individual/team project ☒ 30%

Additional comments

Lectures, e-learning are recommended for learning the basic principles, terminology, and models of software design. These should be reinforced through practical tasks, case studies, individual/team-projects.

1.3.9.4 WBL and Follow-up Reinforcement

After learning the basic principles, terminology, and models of software development documentation, the knowledge gained can form part of the final year project.

1.3.9.5 Important (new) approaches and technologies to consider

Including a guest speaker from industry or academic would enhance the learners knowledge and afford them the opportunity to ask questions and gain insights into the role of a software developer in a business context.

1.3.9.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
Describes types of technical documentation	Assignment	Assignment validates the learning and knowledge acquired by the student by applying these to the creation of documentation using software documentation tools

1.3.10 Learning Resources - 5. PLO Documentation Production [e-2]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Software Documentation		12 hours	Virtual classroom	Software documentation assignment using software documentation tools	ESSA_Juniot_Developer_SOFTWARE_DOCUMENTATION.pptx	Virtual Classroom	https://learn.softwareskills.eu/wp-content/uploads/2024/01/ESSA_Junior_Developer_SOFTWARE_DOCUMENTATION.pdf

1.3.11 PLO 6. Problem management [e-2]

6. PLO Problem management [e-2]	
The learner has demonstrated capability *→ to act systematically in handling incidents and problems*	
Unit learning outcomes	Systematically resolves or escalates incidents and problems, resulting in a solved incident e.g., by applying techniques and tools for troubleshooting such as diagnostic tools

1.3.11.1 Duration of Study

Recommended duration: 12 hours

Often integrated with studies of PLOs: PLOs2, 7 ,8 9

1.3.11.2 Recommendations for Micro-credentials

Potentially could form part of a micro-credential.

1.3.11.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Presence Classroom ☒
- Virtual Classroom ☒

Additional comments

n/a

Recommended delivery methods:

- Lecture ☒ 80%
- Case study. Individual/team project ☒ 10%
- e-Learning ☒ 10%

Additional comments

Lectures, e-learning are recommended for learning the basic principles, terminology, and models of software problem management. These may be reinforced through practical tasks including class discussion, the case study and a multiple choice quiz available for this learning Unit.

1.3.11.4 WBL and Follow-up Reinforcement

After learning the basic principles, terminology, and processes in software problem management the knowledge gained can form part of class projects.

1.3.11.5 Important (new) approaches and technologies to consider

Including a guest speaker from industry or academic would enhance the learners knowledge and afford them the opportunity to ask questions and gain insights into the role of a software developer in a business context.

1.3.11.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
Systematically resolves or escalates incidents and problems, resulting in a solved incident e.g., by applying techniques and tools for troubleshooting such as diagnostic tools	Case Study	Validates students' take the correct approaches to problem management
	Multiple Choice Quiz	Validates the students' knowledge and learning of the aspects of software problem management processes and tasks

1.3.12 Learning Resources - PLO 6. Problem Management [e-2]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Problem Management		12 hours	Lecture & eLearning	Case study & Multiple choice quiz	ESSA_Junior_Developer_Problem_Management ESSA_Junior_Developer_Problem_Solving_Case_Study(EQF4).docx ESSA_Junior_Developer_Problem_Solving_Quiz(EQF4).docx ESSA_New Technologies_Junior Developer.pptx	Lecture	https://learn.softwareskills.eu/wp-content/uploads/2024/01/ESSA_Junior_Developer_Problem_Management.pdf
Problem Management			Additional Reading		**Additional Reading** **Agile for when things go wrong: the missing piece of your incident response plan** • https://www.atlassian.com/agile/software-development/incident-response **Incident Management: A Lost Art in the World of Software Engineering** • https://betterprogramming.pub/incident-management-a-lost-art-in-		

					the-world-of-software-engineering-f1bcac95a03 **Building an Effective Incident Management Process** • https://www.infoq.com/articles/effective-incident-management/		

1.3.13 PLO 7. Professional related competences [EQF5]

7. PLO Profession related competences [EQF5]	
The learner has demonstrated capability *→ to apply profession related skills*	
Unit learning outcomes	Masters common ICT knowledge
	Explains the principles, related concepts, advantages and disadvantages of a new technology. Applies and reports on basic methods, techniques and tools related to a new technology.
	Applies and reports on measures, methods, tools and techniques related to security
	Applies and reports on measures, methods, tools and techniques related to software lifecycle processes
	Is aware of basic ethical considerations and issues

1.3.13.1 Duration of Study

Recommended duration: 4 Weeks (across PLOs 7,8,9)

Integrated with studies of PLOs: PLOs 2 Application Development; 3 Component Integration; 4 Testing; 5 Documentation Production; 6 New Technology; 7 Profession Related Competencies; and 9 Functioning in the Organisation

1.3.13.2 Recommendations for Micro-credentials

This PLO should be an integral part of the initial studies for students with no prior knowledge of software development as it supports them in seeking employment after the programme and develops transversal skills which will be important in an industry setting.

The format is likely unsuitable for Micro-credentials as it comprises a range of supports and activities rather formalized learning content

1.3.13.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Overall Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Presence Classroom ☒
- Virtual Classroom ☒
- Industry Mentor ☒

Additional comments
Professional related competencies are demonstrated through the students' interactions in the classroom, with the Industry Mentor and in the presentation of their final project to a panel consisting tutors, industry and government representatives.

Recommended delivery methods:

- Case study. Individual/team project ☒ 70%
- Industry Mentor ☒ 30%

Additional comments

Additional supports are provided to assist students' with their CV and interview preparation, to support their professional competency development.

1.3.13.4 WBL and Follow-up Reinforcement

The student project - "Software Pathway Project" three PLOs "PLOs 7 Profession Related Competencies; 8 Soft Competencies and 9 Functioning in the Organisation"

1.3.13.5 Important (new) approaches and technologies to consider

Including a guest speaker from industry or academic would enhance the learners knowledge and afford them the opportunity to ask questions and gain insights into the role of a software developer in a business context.

1.3.13.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
	Final Team Project Presentation	Validates each student's ability to work together in a team to create a software application from start to finish. This involves all aspects of software developmet from understanding the requirements, engaging in the analysis and design and also includes implementation, testing and documentation.
	Mentor Assessment	Validates how the students' engaged in the mentoring and Software Pathway Project and the outcome of the project prior to the student presentation to the tutor, industry and government panel for overall assessment

1.3.14 Learning Resources - PLO 7. Profession related competence [EQF5]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Industry Related Software Development Project		4 weeks	Student Project	*Tutor, Industry and government panel assessment*	ESSA_Software_Pathway_JUNIOR_DEVELOPER_(EQF4_7).docx Note document will need to be updated based on reviews and whether it will be used.	Assignment	https://learn.softwareskills.eu/wp-content/uploads/2024/01/ESSA_Software_Pathway_Project_JUNIOR_DEVELOPER.docx

1.3.15 PLO 8. Soft competences [EQF5]

8. PLO Soft competences [EQF5]	
The learner has demonstrated capability *→ to apply soft skills*	
Unit learning outcomes	Works together with others in a team
	Communicates with peers, colleagues, supervisors and/or relevant others, appropriately to the context, using conventions that are relevant to professional practice. Explains and gives instruction.
	Masters the English language at level B2. Can understand the main ideas of complex text on both concrete and abstract topics, including technical discussions in his/her field of specialisation
	Distinguishes and analyses fairly complex and unpredictable problems. Solves these problems systematically and in a creative way, using existing procedures and guidelines and own solutions by identifying and using data.
	Exercises self-management within the guidelines of contexts that are usually predictable, but are subject to change. Is able to cope with limited change and to adapt to a certain level of variety in the workplace. Copes with pressure and stress setbacks and maintains composure. Shows some initiative and carries responsibility for the results of own activities, work and or study. Works correctly and carefully.
	Realises learning and personal development on request, where necessary with support, through self-reflection and external- and self-evaluation of own (learning) results.

1.3.15.1 Duration of Study

Recommended duration: 22 weeks (duration of the programme)

Integrated with studies of PLOs: PLOs 2 Application Development; 3 Component Integration; 4 Testing; 5 Documentation Production; 6 New Technology; 7 Profession Related Competencies; and 9 Functioning in the Organisation

1.3.15.2 Recommendations for Micro-credentials

The format is likely unsuitable for Micro-credentials as it comprises a range of supports and activities rather formalized learning content

1.3.15.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Presence Classroom ☒
- Virtual Classroom ☒
- e-learning ☒

Additional comments

Recommended delivery methods:

- Case study. Individual/team project ☒ 70%
- e-Learning ☒ 30%

Additional comments
Lectures and e-learning are recommended for learning the basic principles, terminology, and models of software development. Soft competencies are gained through the practical experience of engaging in class discussions, engaging in the industry mentoring programme and participating in the Industry Software Project.

1.3.15.4 WBL and Follow-up Reinforcement

The student project - "Software Pathway Project" combined with Industry Mentoring addresses three PLOs "PLOs 7 Profession Related Competencies; 8 Soft Competencies and 9 Functioning in the Organisation" in addition to building on the knowledge gained through the other PLOs

Soft skills are demonstrated through a range of activities over the course of the programme, including group work, active participation in the classroom setting, and attending guest lectures.

1.3.15.5 Important (new) approaches and technologies to consider

Including a guest speaker from industry or academic would enhance the learners knowledge and afford them the opportunity to ask questions and gain insights into the role of a software developer in a business context.

1.3.15.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
	Group Final Project	Validation by an industry and teaching panel of the final project, knowledge attained and project output
	Mentoring Review	Feedback from Industry Mentor on engagement through the mentoring process and suitability for entry level role in software development

1.3.16 Learning Resources - PLO 8. Soft competences [EQF5]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Industry Related Software Development Project		*4 weeks*	*Team Project*	*Tutor, Industry and government panel assessed*	ESSA_Software_Pathway_Project_JUNIOR_DEVELOPER_(EQF4_7).docx	*Assignment by Tutor*	https://learn.softwareskills.eu/wp-content/uploads/2024/01/ESSA_Software_Pathway_Project_JUNIOR_DEVELOPER.docx

1.3.17 PLO 9. Functionning in organisations [EQF5]

9. PLO Functioning in organisations [EQF5]	
The learner has demonstrated capability *→ to function in an organisational context*	
Unit learning outcomes	9.1 Explains the basics of organisation theory and behaviour
	9.2 Describes the relationship between business and IT
	9.3 Works in an organisational context under specific direction with limited autonomy and responsibility e.g., at the level of a trainee, junior or assistant
	9.4 Works in project settings, applies project management methods and tools
	9.5 Writes a report on functioning in the organisation

1.3.17.1 Duration of Study

Recommended duration: 4 Weeks (across PLOs 7,8,9)

Integrated with studies of PLOs: PLOs 2 Application Development; 3 Component Integration; 4 Testing; 5 Documentation Production; 6 New Technology; 7 Profession Related Competencies; and 9 Functioning in the Organisation

1.3.17.2 Recommendations for Micro-credentials

The format is likely unsuitable for Micro-credentials as it comprises a range of supports and activities rather formalized learning content

1.3.17.3 Recommendations on Didactical Approach, Delivery Methods and Training Environment

Recommended didactical approach:

- Presence Classroom ☒
- e-learning ☒

Additional comments

n/a

Recommended delivery methods:

- Case study. Individual/team project ☒ 70%
- Industry Mentor ☒ 30%

Additional comments

Lectures, e-learning are recommended for learning the basic principles, terminology, and models of software development. These are reinforced through the practical tasks undertaken during the course, participating in the industry mentoring programme and engaging in the final group project (Software Pathway Project)

1.3.17.4 WBL and Follow-up Reinforcement

The student project - "Software Pathway Project" addresses three PLOs "PLOs 7 Profession Related Competencies; 8 Soft Competencies and 9 Functioning in the Organisation"

1.3.17.5 Important (new) approaches and technologies to consider

Including a guest speaker from industry or academic would enhance the learners knowledge and afford them the opportunity to ask questions and gain insights into the role of a software developer in a business context.

1.3.17.6 Assessment

Unit learning outcome	Assessment method	Validation of prior acquired competences (skills and knowledge)
1.1	Group Final Project	Validation by an industry and teaching panel of the final project, knowledge attained and project output
1.2	Mentoring Review	Feedback from Industry Mentor on engagement through the mentoring process and suitability for entry level role in software development

1.3.18 Learning Resources -PLO 9. Functionning in organisation [EQF5]

LEARNING UNIT	EQF	Duration	Didactical Approach	ASSESSMENT	Title of the Learning material	Delivery method of the learning material	Quick link to learning materials
Industry Related Software Development Project		*4 weeks*	*Team Project*	*Tutor*	*ESSA_Software_ Pathway_Project_JUNIOR_DEVELOPER_(EQF4_7).docx*	*Assignment by Tutor*	https://learn.softwareskills.eu/wp-content/uploads/2024/01/ESSA_Software_Pathway_Project_JUNIOR_DEVELOPER.docx

www.softwareskills.eu

Co-funded by the
Erasmus+ Programme
of the European Union