CROSS-PLATFORM PROGRAMMING

Working program of the academic discipline (Syllabus)

Details of the educational component

Level of higher education	***First (Bachelor)***
Branch of knowledge	*12 Information Technologies*
Specialty	*121 Software Engineering*
Educational program	*Software Engineering of Multimedia and Information-Retrieval Systems*
Status of the educational component	*Elective*
Form of education	*Full-time*
A year of training	*3rd year of training, 6th semester*
Scope of the discipline	*Lectures: 36 hours, laboratory work: 18 hours, independent work: 66 hours.*
Semester control/ control measures	*Credit, modular control work, calendar control*
Schedule of classes	*According to the schedule for the autumn semester of the current academic year (rozklad.kpi.ua)*
Language of instructions	*English*
Information about head of the course / teachers	*Lecturer: Ph.D., associate professor, V. V. Tsurkan, v.v.tsurkan@gmail.com* *Laboratory work: Ph.D., associate professor, V.V. Tsurkan, v.v.tsurkan@gmail.com*
Course location	Google classroom. Access is given to registered students.

Program of educational component

1. Description of the educational component, its purpose, subject of study and learning outcomes

Studying the discipline "Cross-platform Programming" allows students to develop the competencies necessary for solving practical tasks of professional activities related to the development of cross-platform software.

The purpose of studying the discipline "Cross-platform Programming" is the formation of students' abilities to independently design and develop cross-platform software.

The subject of the discipline "Cross-platform programming" is methods, means of modeling and development of cross-platform software.

The study of the discipline "Cross-platform programming" contributes to the formation of students of ***professional competences (PC)*** *necessary for solving practical tasks of professional activities related to the development, improvement and operation of software:*

PC01 *Ability to identify, classify and formulate software requirements.*

PC02 *Ability to participate in software design, including its structure, behavior and functioning processes modeling (formal description).*

PC03 *Ability to develop software systems architectures, modules and components.*

PC05 *Ability to follow specifications, standards, rules and recommendations in the professional field during the life cycle processes implementation.*

PC06 *Ability to analyze, select and apply methods and tools to ensure information security (including cybersecurity).*

PC11 *Ability to implement phases and iterations of the life cycle of the software systems and information technology based on appropriate models and approaches to software development.*
PC14 *Ability to algorithmic and logical thinking.*
PC21 *Ability to identify, analyze and document software requirements for multimedia and information retrieval systems.*

The study of the discipline "Cross-Platform Programming" contributes to the formation in students of the following ***program learning outcomes (PLO)*** *according to the educational program:*
PLO03 *To know the software life cycle basic processes, phases and iterations.*
PLO05 *To know and apply relevant mathematical concepts, domain methods, system and object-oriented analysis and mathematical modeling for software development.*
PLO06 *Ability to select and use the appropriate task of software development methodology.*
PLO09 *To be able to use collecting, formulating and analyzing software requirements methods and tools.*
PLO10 *To conduct a pre-project survey of the subject area, system analysis of the design object.*
PLO11 *To select initial data for design, guided by formal methods of describing requirements and modeling.*
PLO12 *To apply effective approaches to software design in practice.*
PLO15 *To choose programming languages and development technologies to solve the problems of creating and ma intaining software.*
PLO17 *To be able to apply methods of component software development.*
PLO21 *To know the tools, analyze, select, skillfully apply the information security (including cybersecurity) and data integrity means in accordance with the applied tasks and software systems.*
PLO23 *To be able to document and present the software development results.*

2. Pre-requisites and post-requisites of the discipline (place in the structural and logical scheme of training according to the relevant educational program)

The successful study of the discipline "Cross-Platform Programming" is preceded by the study of the disciplines "Fundamentals of Programming", "Algorithms and Data Structures", "Fundamentals of Computer Systems and Networks", "Components of Software Engineering", "Databases" and "Programming" of the bachelor's training plan in the specialty 121 Software engineering.

The theoretical knowledge and practical skills obtained during the mastering of the discipline "Cross-Platform Programming" ensure the successful implementation of course projects and master's theses in the specialty 121 Software engineering.

3. Content of the academic discipline

The discipline "Cross-Platform Programming" involves the study of the following topics:
Topic 1. Specification of cross-platform software requirements.
Topic 2. Creating a logical structure of cross-platform software.
Topic 3. Creation of the physical structure of cross-platform software.
Topic 4. Implementation of the physical structure of cross-platform software.
Modular control work
Test

4. Educational materials and resources

Basic literature:

1. OMG® Unified Modeling Language® (OMG UML®). Version 2.5.1. URL: https://www.omg.org/spec/UML/2.5.1/PDF (accessed on: 01.06.2022).
2. OMG Systems Modeling Language (OMG SysML™). Version 1.6. URL: https://sysml.org/.res/docs/specs/OMGSysML-v1.6-19-11-01.pdf (accessed on: 01.06.2022).
3. Percival H., Gregory B. Architecture Patterns with Python: Enabling Test-Driven Development, Domain-Driven Design, and Event-Driven Microservices. Sebastopol, USA : O'Reilly Media, 2020. 304 p.
4. Aniche M. Effective Software Testing: A developer's guide. Shelter Island, USA: Manning, 2022. 328 p.

5. Threat Modeling. Process. URL: https://owasp.org/www-community/Threat_Modeling_Process (accessed on: 01.06.2022).
6. Shostack A. Threat Modeling: Designing for Security. Indianapolis: John Wiley & Sons, 2014. 590 p.
7. ISO/IEC 27005:2018. Information technology. Security techniques. Information security risk management. [Valid from 2018-06-10]. URL: https://www.iso.org/standard/75281.html (accessed on: 01.06.2022).

Additional literature:

8. Dennis A., Wixom B., Tegarden D. Systems Analysis and Design : An Object-Oriented Approach with UML. Hoboken, USA : Wiley, 2020. 544 p.
9. Holt J, Perry S. SysML for Systems Engineering: A model-based approach (Computing and Networks). London, United Kingdom : The Institution of Engineering and Technology, 2019. 880 p.
10. Python documentation. URL: https://docs.python.org/3/ (accessed on: 01.06.2022).
11. Myers G., Sandler C., Badgett T. The Art of Software Testing. Hoboken, USA : Wiley, 2011. 256 p.
12. Threat Modeling. URL: https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling (accessed on: 01.06.2022).
13. Tarandach I., Coles M. J. Threat Modeling. A Practical Guide for Development Teams. Sebastopol: O'Reilly Media, 2020, 201 p.
14. MITER ATT&CK. URL: https://attack.mitre.org/ (accessed on: 01.06.2022).

5. Educational content

1. Methods of mastering an educational discipline (educational component)

№	Type of training session	Description of the training session
Topic 1. Introduction to software security		
1	Lecture 1. Course content, introduction to cross-platform programming	Overview of course content. The concept of cross-platform as a property of software. Varieties of cross-platform software. Life cycle of cross-platform software development. Ways and means of achieving cross-platform software. Task on self-study: item 6 No. 1.
2	Lecture 2. Methods of determining functional requirements for cross-platform software	Concept of functional requirements for cross-platform software. The process of determining functional requirements for cross-platform software. Stages of determining functional requirements for cross-platform software. Presentation of options for using cross-platform software. Assignment on self-study: item 6 No. 2.
3	Laboratory work 1. Specification of functional requirements for cross-platform software	Task: to specify functional requirements for cross-platform software. Assignment on self-study: item 6 No. 3.
4	Lecture 3. Methods of determining non-functional requirements for cross-platform software	Security as a non-functional requirement for cross-platform software. The process of determining non-functional requirements for cross-platform software. Stages of determining non-functional requirements for cross-platform software. Assignment on self-study: item 6 No. 4.
5	Lecture 4. Methods of modeling security threats of cross-platform software	The concept of cross-platform software security threat model. The process of modeling security threats of cross-platform software. Stages of modeling security threats of cross-platform software. Characteristics of the STRIDE-DREAD combined method. Characteristics of the method of assessing the risks of information security and cyber security.

		Task on self-study: item 6 No. 5.
6	Laboratory work 2. Specification of non-functional requirements for cross-platform software	Task: to specify non-functional requirements (security) for cross-platform software. Assignment on self-study: item 6 No. 6.
Topic 2. Methods of modeling software security threats		
7	Lecture 5. Static logical structure of cross-platform software	Concept of static logical structure of cross-platform software. Ways of presenting the static logical structure of cross-platform software. Representation of entities of the subject area by classes. Assignment on self-study: item 6 No. 7.
8	Lecture 6. Presentation of the static logical structure of cross-platform software	Definition of classes, their attributes and methods (operations). Types of relations between classes. Definition of relations between classes. Graphical notation of representation of classes and relations between them. Examples of representation of classes and relations between them. Assignment on self-study: item 6 No. 8.
9	Laboratory work 3. Creating a static logical structure of cross-platform software	Task: to create a static logical structure of cross-platform software. Assignment on self-study: item 6 No. 9.
10	Lecture 7. Dynamic logical structure of cross-platform software	Concept of dynamic logical structure of cross-platform software. Ways of presenting the dynamic logical structure of cross-platform software. Cross-platform software behavior specification. Assignment on self-study: item 6 No. 10.
11	Lecture 8. Presentation of the dynamic logical structure of cross-platform software	Representation of cross-platform software behavior by activities. Definition of nodes and arcs of activity. Definition of nodes of activity management. Graphical notations of representation of activity. Examples of activity presentation. Task on self-study: item 6 No. 11.
12	Laboratory work 4. Creating a dynamic logical structure of cross-platform software	Task: to create a dynamic logical structure of cross-platform software. Task on self-study: item 6 No. 12.
13	Lecture 9. Interaction of elements of cross-platform software	Concept of interaction of elements of cross-platform software. Ways of representing the interaction of elements of cross-platform software. Specification of interaction of elements of cross-platform software. Assignment on self-study: item 6 No. 13.
14	Lecture 10. Presentation of the interaction of elements of cross-platform software	Representation of the interaction of cross-platform software elements with lifelines and messages between them. Determination of life lines. Defining messages between life lines. Graphic notation of the representation of life lines and messages

		between them. Examples of the representation of life lines and messages between them. *Assignment on self-study: item 6 No. 14.*
Topic 3. Creation of the physical structure of cross-platform software		
15	*Lecture 11. Physical structure of cross-platform software*	*The concept of the physical structure of cross-platform software. Ways of presenting the physical structure of cross-platform software. Specification of the physical structure of cross-platform software.* *Assignment on self-study: item 6 No. 15.*
16	*Lecture 12. Presentation of the physical structure of cross-platform software*	*Representation of the physical structure of cross-platform software by components. Identifying component ports and connectors. Determination of relations between components. Graphical notations of physical structure representation. Examples of physical structure representation.* *Assignment on self-study: item 6 No. 16.*
17	*Laboratory work 5. Creation of a physical structure of cross-platform software*	*The task: to create a physical structure of cross-platform software.* *Task on self-study: item 6 No. 17.*
18	*Lecture 13. Physical configuration of cross-platform software*	*Concept of physical configuration of cross-platform software. Ways of representing the physical configuration of cross-platform software. Specification of the physical configuration of cross-platform software.* *Assignment on self-study: item 6 No. 18.*
19	*Lecture 14. Presentation of the physical configuration of cross-platform software*	*Representation of the physical configuration of cross-platform software by nodes. Types of physical nodes. Stereotypes of physical nodes. Determination of physical nodes and relations between them. Graphical notations of the representation of the physical configuration. Examples of physical configuration representation.* *Assignment on self-study: item 6 No. 19.*
20	*Laboratory work 6. Creating a physical configuration of cross-platform software*	*Task: create a physical configuration of cross-platform software.* *Task on self-study: item 6 No. 20.*
Topic 4. Implementation of the physical structure of cross-platform software		
21	*Lecture 15. Means of implementing the physical structure of cross-platform software*	*Functional assignment of cross-platform software. Description of the logic of cross-platform software. Selection of means of implementing the physical structure of cross-platform software.* *Assignment on self-study: item 6 No. 21.*
22	*Lecture 16. The logic of cross-platform software*	*Structure of cross-platform software Elements of cross-platform software. Functions of elements of cross-platform software. Relationship between elements of cross-platform software. The logic of elements of cross-platform software.* *Task on self-study: item 6 No. 22.*

23	*Modular control work. Creation of a working project of a cross-platform program*	*Task: to create a working project of cross-platform software.* *Task on self-study: item 6 No. 23.*
24	*Lecture 17. Cross-platform software testing*	*The concept of cross-platform software testing. The life cycle of cross-platform software. Methods of testing cross-platform software. Peculiarities of cross-platform software testing.* *Task on self-study: item 6 No. 24.*
25	*Lecture 18. Test design of cross-platform software*	*The concept of cross-platform software test design. The concept of a test case. Attributes of test cases. Life cycle of test cases. Creation of test cases. Quality of test cases.* *Task on self-study: item 6 No. 25.*
26	*Laboratory work 7. Demonstration of a working project of cross-platform software*	*Task: to demonstrate a working project of cross-platform software.* *Task on self-study: item 6 No. 26.*
27	*Modular controul work*	*Task on self-study: item 6 No. 27.*

7. Independent work of a student/graduate student

The discipline "Cross-Platform Programming" is based on independent preparations for classroom classes on theoretical and practical topics.

№z з/p	*The name of the topic submitted for independent processing*	*Number of hours*	*literature*
1	*Preparation for the lecture 1*	*1,5*	*1–3; 8; 9*
2	*Preparation for the lecture 2*	*1,5*	*1; 2; 8; 9*
3	*Preparation for laboratory work 1*	*4*	*1; 2; 8; 9*
4	*Preparation for the lecture 3*	*1,5*	*1; 2; 5–9; 12–14*
5	*Preparation for the lecture 4*	*1,5*	*1; 2; 5–9; 12–14*
6	*Preparation for laboratory work 2*	*4*	*1; 2; 5–9; 12–14*
7	*Preparation for the lecture 5*	*1,5*	*1–3; 8; 9*
8	*Preparation for the lecture 6*	*1,5*	*1–3; 8; 9*
9	*Preparation for laboratory work 3*	*4*	*1–3; 8; 9*
10	*Preparation for the lecture 7*	*1,5*	*1–3; 8; 9*
11	*Preparation for the lecture 8*	*1,5*	*1–3; 8; 9*
12	*Preparation for laboratory work 4*	*4*	*1–3; 8; 9*
13	*Preparation for the lecture 9*	*1,5*	*1–3; 8; 9*
14	*Preparation for the lecture 10*	*1,5*	*1–3; 8; 9*
15	*Preparation for the lecture 11*	*1,5*	*1–3; 8; 9*
16	*Preparation for the lecture 12*	*1,5*	*1–3; 8; 9*
17	*Preparation for laboratory work 5*	*4*	*1–3; 8; 9*
18	*Preparation for the lecture 13*	*1,5*	*1–3; 8; 9*
19	*Preparation for the lecture 14*	*1,5*	*1–3; 8; 9*
20	*Preparation for laboratory work 6*	*4*	*1–3; 8; 9*
21	*Preparation for the lecture 15*	*1,5*	*1–3; 8–10*
22	*Preparation for the lecture 16*	*1,5*	*1–3; 8–10*

23	*Preparation for the Preparation for the modular control laboratory work*	*6*	*1–3; 8–10*
24	*Preparation for the lecture 17*	*1,5*	*3, 4, 11*
25	*Preparation for the lecture 18*	*1,5*	*3, 4, 11.*
26	*Preparation for laboratory work 7*	*3*	*3, 4, 11*
27	*Preparation for the test*	*6*	*1-14*

7. Policy and control

Attending lectures is mandatory.

Attending laboratory work classes may be occasional and as needed for consultation/protection of laboratory work.

Rules of behavior in classes: activity, respect for those present, turning off phones.

Adherence to the policy of academic integrity.

Rules for the protection of laboratory work: the work must be performed in accordance with the assigned tasks and according to the option chosen by the student.

8. Types of control and rating system for evaluating learning outcomes

During the semester, students perform 7 laboratory works.
The maximum number of points for each laboratory work: 5 points.

Points are awarded for the quality of performance and protection of laboratory work: 0-5 points.
Criteria for evaluating the quality of performance and protection:
5 points - the work is done qualitatively, in full, the answers are complete, well-argued;
4 points - the work is done qualitatively, in full, but has shortcomings, answers with minor errors;
3 points – the work is done with sufficient quality, in full, but contains significant shortcomings, answers with significant errors;
0 points - the work is not done well, not in full, the answers are either absent or incorrect.
The maximum number of points for performing and defending laboratory work:
5 points × 7 laboratory works = 35 points.

*The task of **modular control work is** to implement software security requirements. The answer is evaluated by 15 points.*
Evaluation criteria for modular test work:
14–15 points – the answer is correct, complete, well-argued;
12–13 points – the answer is generally correct, but has flaws;
9–11 points – there are significant errors in the answer;
0 points - there is no answer or the answer is incorrect.

The maximum number of points for a modular control work:

15 points × 1 task = 15 points.

The rating scale for the discipline is equal to:

R = RS = Rlab. works + R $_{\textit{modular control work}}$ *+ Rexam = 35 points + 15 points + 50 points = 100 points.*

Calendar control: is conducted twice a semester as a monitoring of the current state of fulfillment of the syllabus requirements.

At the first certification (8th week), the student receives "Passed" if his current rating is at least 10 points (50% of the maximum number of points that the student can receive before the first certification).

At the second certification (14th week), the student receives "Passed" if his current rating is at least 20 points (50% of the maximum number of points that the student can receive before the second certification).

Semester control: exam

Conditions for admission to semester control:

A prerequisite for a student's admission to the exam is a semester rating (RC) of at least 30 points. After passing the exam, a grade is assigned according to the table (Table of correspondence of rating points to grades on the university scale).

The exam task consists of 3 questions - 2 theoretical and 1 practical. The answer to each theory question is worth 15 points, and the answer to a practical question is worth 20 points.

Evaluation criteria for a theoretical question:

14–15 points – the answer is correct, complete, well-argued;

11–13 points – the answer is generally correct, but has flaws;

5–10 points – there are significant errors in the answer;

0 points - there is no answer or the answer is incorrect.

Evaluation criteria for a practical question:

17–20 points – the answer is correct, complete, well-argued;

12–16 points – the answer is generally correct, but has flaws;

5–11 points – there are significant errors in the answer;

0 points - there is no answer or the answer is incorrect.

Table of correspondence of rating points to grades on the university scale:

Scores	*Rating*
100-95	*Perfectly*
94-85	*Very good*
84-75	*Fine*
74-65	*Satisfactorily*
64-60	*Enough*
Less 60	*Unsatisfactorily*
Admission conditions not met	*Not allowed*

2. Additional information on the discipline (educational component)

The list of questions submitted for semester control is given in Appendix 1.

Working program of the academic discipline (syllabus):

Compiled by Ph.D., Associate Professor V.V. Tsurkan.

Adopted by Computer Systems Software Department (protocol № 8 from 25.01.23)

Approved by the Faculty Board of Methodology (protocol № 6 from 27.01.23)

Appendix 1. List of questions submitted for semester control

1. To characterize the concept of cross-platform as a property of software.

2. To characterize the types of cross-platform software.

3. Describe ways to achieve cross-platform software.

4. Describe the means of achieving cross-platform software.

5. Describe the methods of determining functional requirements for cross-platform software.

6. Describe the presentation of options for using cross-platform software.

7. Describe the methods of determining non-functional requirements for cross-platform software.

8. Describe the methods of modeling security threats of cross-platform software.

9. Describe the concept of static logical structure of cross-platform software.

10. Describe the presentation of the static logical structure of cross-platform software.

11. Describe the concept of dynamic logical structure of cross-platform software.

12. Describe the representation of the dynamic logical structure of cross-platform software.

13. Describe the concept of interaction of elements of cross-platform software.

14. Describe the representation of the interaction of elements of cross-platform software.

15. Describe the concept of the physical structure of cross-platform software.

16. Describe the representation of the physical structure of cross-platform software.

17. Describe the concept of physical configuration of cross-platform software.

18. Describe the representation of the physical configuration of cross-platform software.

19. Describe the implementation of the physical structure of cross-platform software.

20. Describe the process of testing cross-platform software.