MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

SIMON KUZNETS KHARKIV NATIONAL UNIVERSITY OF ECONOMICS

Guidelines to writing a diploma project for students of higher education, specialty 121 "Software engineering" of the educational program "Software engineering" of the first (bachelor's) level

Authors: **Parfyonov Yu.E.**
PolyakovA. O.
Ushakova I. O.
Frolov O.V.

D.O. Bondarenko is responsible for the release.

Kharkiv
HNEU
2024

UDC 004.415 (07.034)
M54

Approved at a meeting of the Department of Information Systems.
Protocol No. 8 dated January 19, 2024.

Independent electronic text network publication

Authors : Yu. E. Parfyonov
A.O. Polyakov
I. O. Ushakova
O. V. Frolov

Methodical recommendations for writing a diploma project for students of higher education, specialty 121 "Software engineering" of the educational program "Software engineering" of the first (bachelor) level [Electronic resource] / Yu. E. Parfyonov, A. O. Polyakov, I. O. Ushakova, O. V. Frolov. – Kharkiv: KHNEU, 2024. - 65 p.

The issue of the organization of the diploma project is outlined, the requirements for the structure of the diploma project, methodical recommendations for the development of its structural elements are given.

It is recommended for applicants of the 121 "Software Engineering" specialty of the "Software Engineering" educational program of the first (bachelor's) level.

UDC 004.415 (07.034)

Introduction

Diploma project is one of the most important types of self-studies that completes the preparation of applicants for a bachelor's program, as well as the basis for conducting state certification of bachelors.

The main task of diploma project is to prepare the applicant to perform the tasks and duties provided for primary positions in a certain type of economic activity.

A bachelor's diploma project can be the initial stage of a master's thesis.

The methodical recommendations outline the general requirements for the organization and conduct of diploma project, the content, structure and scope of diploma projects, organization of work on the project, design and defense of the diploma project.

The content, scope and structure of the diploma project, which are given in the methodological recommendations, are typical and may be changed in some cases with the written permission of the graduation department.

The methodical recommendations are intended for applicants who are studying in specialties 121 "Software engineering" of the "Software engineering" educational program of the first (bachelor's) level.

The educational component "Diploma Project" is the final stage of the preparation of bachelors in the "Software Engineering" educational program. In accordance with the educational and professional program of training bachelors in software engineering, the discipline participates in the formation of learning outcomes and competencies, which are defined in the table. 1.

Table 1

Learning outcomes and competences formed by the educational discipline

Learning outcomes	Competences that must be mastered by a student of higher education
1	2
LO09	GC02
	GC03
	SK01
	SK04

The end of the table 1

1	2
LO11	SC01
	SC02
LO12	SC01
	SC02
LO14	SC04
	SC05
	SC13
LO15	SC10
	SC11
	SC13
LO16	GC10
	SC12
LO19	GC02
	SC04
LO20	GC02
	SC04
	SC09
LO23	GC03
	GC04
	SC10
LO24	GC02
	GC10
	SC09

where, GC02. Ability to apply knowledge in practical situations;

GC03. Ability to communicate in the state language both orally and in writing;

GC04. Ability to communicate in a foreign language both orally and in writing.

GC10. The ability to act socially responsibly and consciously;

SC01. Ability to identify, classify and formulate software requirements;

SC02. Ability to participate in software design, including modeling (formal description) of its structure, behavior and functioning processes;

SC04. Ability to formulate and ensure software quality requirements in accordance with customer requirements, terms of reference and standards;

SC05. Ability to comply with specifications, standards, rules and guidelines in the professional field when implementing life cycle processes;

SC09. Ability to evaluate and take into account economic, social, technological and environmental factors affecting the field of professional activity;

SC10. The ability to accumulate, process, and systematize professional knowledge about creating and maintaining software and recognize the importance of life long learning;

SC11. Ability to implement phases and iterations of the life cycle of software systems and information technologies based on appropriate software development models and approaches;

SC12. Ability to carry out the system integration process, apply change management standards and procedures to maintain the integrity, overall functionality and reliability of the software;

SC13. Ability to reasonably choose and master tools for software development and maintenance.

LO09. Know and be able to use methods and tools for collecting, formulating and analyzing software requirements;

LO11. Select input data for design, guided by formal methods of requirements description and modelling;

LO12. Apply effective software design approaches in practice;

LO14. Apply in practice software tools for domain analysis, design, testing, visualization, measurement and documentation of software;

LO15. Motivated choice of programming languages and development technologies to solve developing and maintaining software the problems;

LO16. To have skills in team development, approval, design and release of all types of program documentation;

LO19. Know and be able to apply software verification and validation methods;

LO 20. Know approaches to software quality assessment and assurance;

LO23. Be able to document and present the results of software development;

LO24. Be able to calculate the economic efficiency of software systems.

1. The purpose and tasks of diploma project

Diploma project is the final stage of bachelor's training.

The diploma project is a qualifying work designed to objectively monitor the degree of formation of graduates' abilities to solve typical activity tasks that belong to design (design and construction) and executive (technological, technical) production functions.

The goal of diploma project is to generalize and systematize the knowledge and practical skills of the applicants, which they acquired during the study of educational disciplines of humanitarian and socio-economic training; mathematical and natural science training; professional and practical training. In the process of working on a diploma project, applicants acquire skills in the analysis of scientific and technical, normative and reference literature, the use of state standards, the drafting of an explanatory note to the project, and the practical application of knowledge when making specific project decisions.

The tasks of diploma project are:

systematization, consolidation and expansion of theoretical knowledge and practical skills in the direction of preparation, application of this knowledge and skills in the process of completing specific tasks of the diploma project;

development and consolidation of independent work skills;

improving the ability to use modern programming systems, to solve engineering tasks for the design of information systems and their elements, using modern methodologies, information technologies, to conduct computer modeling, as well as the ability to process and systematize the results of research, using computer technology and appropriate tools;

determination of compliance of the level of training of the graduate with the requirements of the educational and qualification characteristics of the bachelor, his readiness and ability to work independently in the conditions of the market economy, modern production, progress of science and technology.

Carrying out the diploma project, the acquirer must make full use of acquired knowledge of information technologies and computer equipment, intelligent systems and knowledge bases, available packages, methods and means of mathematical information processing; combine theoretical knowledge with production experience gained during practice; use the achievements of domestic and world science and technology ; take into account the technical and economic indicators of the functioning of the created software and information systems and complexes; perform the design of selected technical solutions at a high theoretical and professional level; competently, fully and at the same time succinctly state your decisions in an explanatory note.

During the defense of the diploma project, the applicant must concisely convey the main content of the work, emphasizing its relevance and novelty, the possibilities of its practical application, present the technical solutions adopted in it with arguments and justify the obtained results.

The diploma project is an independent work of the applicant. The author is responsible for all project solutions developed in it, as well as the correctness, validity of calculations and proper design of its materials.

A candidate who has completed a full course of study and passed all the credits and exams provided for in the curriculum, i.e. has fulfilled all the requirements of the curriculum in terms of training, is admitted to the diploma project.

2. Organization of the diploma project

The applicant may be assigned a topic of the diploma project from the list of recommended topics. He is also given the right to independently choose a topic, taking into account his inclinations and opportunities to apply the acquired knowledge to the fullest extent. If the topic is proposed by the applicant, it must be discussed and agreed with the head of the diploma project.

For approval of the chosen topic, the applicant submits an application to the head of the Department of Information Systems. A sample statement is provided in Appendix A.

The direct management of the diploma project is entrusted to the teachers of the department, who are appointed by the head of Department of Information Systems.

The mentor of the diploma project assigns the applicant; helps the applicant in drawing up a calendar plan; conducts consultations; controls the project execution process according to the calendar plan; recommends the acquirer to scientific and technical literature and normative and reference sources on the topic of the project; checks work materials; carries out a preliminary hearing of the results of the diploma project.

The recipient completes the diploma project independently. This requires a clear organization of its work from the moment of choosing the topic of the work to its defense.

At the initial stage, the applicant must familiarize himself with the main publications on the topic of the diploma project and make a list of them.

Based on the study of literary sources, which should include both monographs, textbooks and study guides, articles in periodicals, as well as patent materials, scientific and technical reports, abstract publications, the applicant must clearly imagine what has been done in the theoretical and applied aspects of the topic of the diploma project, as well as get acquainted in detail with similar solutions in the relevant field. Based on the results of this work, an analytical review (comparative analysis) is drawn up, from which the selected research methods should logically follow.

After studying the literary sources, the applicant draws up a preliminary plan for the implementation of the diploma project, discusses it with the supervisor. In the process of discussion, the initial data for designing and the terms regulating the work of the acquirer are clarified. After that, the acquirer draws up a detailed plan of work on the project, agrees it with the manager and starts designing. During the implementation of the diploma project, the applicant must regularly attend the supervisor's consultations, submit working materials to him for review in accordance with the plan-schedule of the project's stages .

The control of the head of the diploma project does not release the winner from full responsibility for the validity of the decisions made, compliance with standards, deadlines for the implementation of the calendar plan.

At the meetings of the *Information Systems Department, reports from the heads of diploma projects about the progress of the calendar plans are regularly heard. Applicants who do not adhere to the project implementation schedule or are significantly behind in its implementation are invited to report to the department meeting.*

3. Structure, content and scope of the diploma project. General requirements for diploma projects

Diploma project consists of an explanatory note and other mandatory materials (diagrams, diagrams, graphs of dependencies, tables, drawings, program listings, etc.) developed according to the task.

The volume of the explanatory note is 50-70 printed pages of A4 format (without appendices).

The general structure of the explanatory note *of the diploma project and the recommended number of pages* are given in the table. 2 *.*

Table 2

The structure of the explanatory note of the diploma project

Structural elements of the explanatory note	Number of pages
Title page	1
Tasks for *the diploma project*	2
Abstract	1 - 2
Content	1
List of conventional abbreviations	1
Introduction	1 - 2
Chapter 1	9 - 12
Section 2	14 - 19
Section 3	14 - 20
Chapter 4	8 - 10
Conclusions	1 - 2
references	2 - 3
Appendices	

The explanatory note is printed on sheets of A4 paper. Its text should be written using the Times New Roman typeface (point 14), with a line spacing of 1.2.

Explanatory note page margins: 30 mm from the left edge of the sheet, 10 mm from the right edge of the sheet, 20 mm each from the top and bottom edges of the sheet.

Paragraph indentation should be the same throughout the text and equal to 1.27 cm.

Alignment of the main text is done "by width".

Detailed requirements for drawing up an explanatory note are given in the relevant methodological recommendations [20].

The diploma project is aimed at the analysis, modeling, forecasting of information processes or management of such processes in the economy (as an example, you can consider a separate enterprise, organization, industry or geographical region). They involve the selection of a specific subject area for analysis and research. The implementation of such projects involves the analysis of the subject area based on the study of special literature and familiarization with information processes directly at enterprises and organizations.

The substantive part of the project explanatory note must contain:

1. *Analysis of the subject area;*
2. *Specification of software requirements;*
3. *Design and technical solutions;*
4. *Software testing.*

The structure of the substantive part of diploma projects of a practical nature , which must be followed, is given in *Appendix G.*

4. Methodological recommendations for the development of structural elements of the explanatory note

4.1. General recommendations for the development of the explanatory note of the diploma project

The general requirements for the text of the explanatory note are the logical sequence of the presentation of the material, the clarity and specificity of the presentation of the theoretical and practical results of the work, the essence of the assignment and the purpose of the work, research methods, decisions made, the provenance of the conclusions and the validity of the recommendations. In the text of the explanatory note, it is necessary to adhere to a single terminology. It should not be overloaded with uninformative material, description of well-known data, derivation of formulas, etc. It is necessary to refer to sources of information. In the text of the explanatory note, the mathematical apparatus used and the results of calculations performed using a PC should be indicated.

The text of the explanatory note should not be written in the first person, it is better to use the impersonal form (for example, "calculated", "found") for the entire text in the specified case and tense.

When laying out the material, you should not use:

colloquial turns;

slang words and phrases;

different terms for the same concept;

foreign words and terms if there are equivalent words and terms in the Ukrainian language;

abbreviations of words and phrases, in addition to those established by the rules of spelling and regulatory documents.

The first page of the explanatory note is **the title page.** He contains the following data:

information about the performer of the work - a legal entity (organization) or an individual;

full name of the document;

signatures of responsible persons;

the year of drafting the explanatory note;

A sample title page of the diploma project is given in Appendix D.

The abstract is a summary of the content of the explanatory note, containing the main factual information and conclusions necessary for the initial familiarization with it. *The abstract is written in Ukrainian and English.*

The abstract should be concise, informative and contain information that allows decisions to be made about the feasibility of reading the explanatory note.

The abstract should contain:

information about the volume of the explanatory note, the number of illustrations, tables, appendices, the number of sources according to the list of references (all information is given, including the data of the appendices);

abstract text;

list of keywords.

The text of the abstract should reflect the information provided in the explanatory note in the following sequence:

object of research or development;

The purpose of the work;

research methods and equipment;

results and their novelty;

main technological and operational characteristics and indicators;

relationship with other works;

recommendations on the use of work results;

field of application;

significance of work and conclusions;

predictive assumptions about the development of the object of research or development.

The abstract should be no more than 500 words long, and preferably on one page of A4 format.

Keywords are intended to reveal the essence of the project and to disseminate information about the development. They are placed after the abstract text. The list of keywords contains from 5 to 15 words (phrases), printed in capital letters in the nominative case on a line separated by commas.

Examples of essays are given in Appendix E.

The content includes: introduction; names of all sections, subsections and points of the main part of the explanatory note; conclusions; References; names of appendices and page numbers that contain the beginning of the material.

An example of the content of the explanatory note is given in Appendix D.

If the explanatory note uses little-known abbreviations, new symbols, designations, etc., then it should contain **a list of conventional abbreviations,** which is presented in the form of a separate list, placed before the introduction. Despite this, for the first appearance of these elements in the text of the document, their transcript is provided.

If special terms, abbreviations, symbols, designations are repeated less than three times in the work, the list is not compiled, and their transcription is provided in the text at the first mention.

An example of a list of conditional abbreviations is given in [19].

The introduction is a reflection of the work, so it should be carefully processed. It is better to form the introduction after completing the main text of the explanatory note.

In the introduction to the diploma the project needs to identify and formulate a business problem that has arisen at the enterprise, justify the relevance of the project topic for solving this problem on the basis of the developed module or system. Briefly describe the functionality of the module or system. It is necessary to characterize the technical and software platforms for the development of the automated module. It is necessary to formulate the goal and task of the project, to define the object and subject area of the design. It is also necessary to provide information on the design tools used in the diploma project and the possible fields of application of the results obtained in the diploma project.

Conclusions for the work are a summary of the results of the entire work. This part is of particular importance, since the final results of the work should be presented here.

The explanatory note must contain conclusions for each stage of the completed project and for the work in general, which must be correlated with the goal and task of diploma project.

It is necessary to note the practical value of the work results, to give recommendations for further improvement of the design object. Noting the practical value of the obtained results, it is important to outline the degree of their readiness for use, the scope of use, as well as to provide concise information on the implementation of research results, indicating the names of the organizations in which the implementation was carried out, forms of implementation, details of documents, etc. If the diploma project is implemented at the enterprise, then a certificate or act of implementation is attached to the diploma project.

They provide information on where and when the obtained results were tested (at which conferences and seminars they were presented), the publications of the recipient are listed based on the materials of the diploma project.

The list of used sources should contain information about the literary sources used in the process of developing the project.

The bibliography is a register of used sources on the topic of the project in the broadest possible sense. Therefore, one should not limit oneself to only cited literature. The list should include all materials that were read, reviewed, analyzed in the process of working on the diploma project and related to its topic. It is advisable to provide the sources as completely as possible, remembering that the bibliographic list for the project is a summary of the study of the problem and a prerequisite for further scientific research.

The list of used sources is drawn up in accordance with DSTU 8302:2015 "Bibliographic reference. General requirements and rules of compilation" [9].

The list of used sources must contain at least 30 sources. It is presented in the original language, placed in alphabetical order of the first authors' surnames or titles, and numbered in ascending order. Numbering is continuous.

The works of one author are arranged alphabetically by title or in the chronology of their writing. The alphabetical list is arranged alphabetically in the following sequence:

literature of the combined Cyrillic alphabet, for sources in languages with Cyrillic graphics (Ukrainian, Bulgarian, etc.);

literature in the Latin alphabet;

electronic resources in the same sequence as printed editions (first in Cyrillic and then in Latin).

The list of used sources must contain the surname and initials of the author, the full name of the source, the city of publishing, the publisher and the year of publication, the number of pages or links to pages, etc. The total volume of the book in pages is indicated if the reference is made to it in its entirety, pages (from ... to) are noted if the reference refers to a separate part of the literary source.

Appendices contain material that is necessary for the completeness of the explanatory note, but cannot be consistently placed in its main part due to the large volume or for other reasons **.**

Illustrations (diagrams of business processes, dialogue scenarios, etc.), tables, intermediate mathematical proofs, formulas and calculations, auxiliary text, etc. can be made in the form of attachments.

4.2. Recommendations for the development of sections of the main part of the explanatory note of the diploma project

CHAPTER 1. ANALYSIS OF THE SUBJECT FIELD <NAME>

The purpose of Section 1 is a detailed study of the main aspects and characteristics of the subject area of the project, namely, the definition of the content and modeling of the subject area, a review of literary sources regarding existing solutions and analogues, positioning of the software product.

Subsection 1.1. Definition of content of the subject area

Defining the content of the subject area provides context and understanding of the problem or problem to be solved. If the topic of the diploma is related to business, then this subsection provides a brief description of the areas of activity of the management object (enterprise, organization); the business problem to be solved is defined. Otherwise, the subject area is determined without reference to the enterprise.

Here it is necessary to define:

development goals and objectives;

context and scope. Context is the boundaries within which processes are performed and related events occur, and the resources necessary for their appropriate interpretation; scopes are usually defined in terms of product, type of customer or industry;

connections and interactions with other subject areas. A subject area can be complex and include different elements that are related to each other;

key concepts and terms specific to this subject area. It can include many interrelated concepts, processes, subjects and objects;

changes and evolution of the subject area. The subject area may be subject to change and development over time. New technologies, research and requirements can influence the subject area, leading to its change and development .

Subsection 1.2. Modeling of the subject area

Modeling of the subject area involves the creation of a system of models that simulate the structure and functioning of the studied subject area and meet the basic requirements - to be adequate to this area.

In subsection 1.2. necessary:

to determine the composition of the company's divisions and the connections between them, to develop a diagram of the company's organizational structure;

to develop a diagram of the organizational structure of the unit (units) related to the defined business processes;

determine the composition of functions included in the business process, develop a function tree diagram;

develop a business process management model and describe it (Table 3).

Table 3

Characteristics of the business process "Name"

The name of the characteristic	The value of the characteristic
The name of the business process	
Main participants*	
Input event	
Incoming messages / documents**	
Weekend event	
Outgoing messages / documents**	
Business Process Client***	

*for each participant, indicate the structural division, position, his role in the business process,

** provide a list of documents

*** process using process information

CASE tools are used to model the subject area.

In the process of modeling, it is necessary to distinguish the transactional component of the business process, which ensures the collection, accumulation and processing of quantitative data about the current state of the management object, as well as the analytical component, which provides the analysis of quantitative indicators formed into transactional components.

The analytical component of the business process should ensure the study of quantitative indicators in various sections and dimensions (by time periods, by goods, by customers, divisions, etc.).

Conducting such a multi-faceted analysis will provide informational support for decision-making aimed at solving the identified problem.

Subsection 1.3. Review and analysis of literary sources regarding existing solutions and analogues

In subsection 1.3. it is necessary to conduct a review of literary sources regarding existing approaches to the development of software for a correspondingly defined problem, goals and tasks that relate to business requirements and principles, certain technologies, platforms, libraries, etc. During the review, references should be made to relevant sources (for example, in the work [..] the authors conduct a systematic review; the author touches on (covers) such problems (questions, facts); the work [..] of the author concerns; the author in the work [..] draws a conclusion (summarizes, says, asserts), points out (analyzes, characterizes, reveals) shortcomings (contradictions, essence), describes, names, formulates, puts forward (hypothesis, question), makes an assumption, stops, emphasizes, asserts, proves etc. based on the results of the review, a conclusion is made regarding the use of better approaches to development.

It is necessary to choose several of the most popular analog software products designed to implement the functionality of the subject area and determine their characteristics, such as:

development company;
product versions;
Operating System;
basic functionality;
user interface;
user assistance;
usage price per month/year;
availability of a free version / plan;
etc.

The results of the analysis are given in the table (Table 4):

Table 4

Characteristics of similar software products

The name of the software product	<Product Name 1>	<Product Name 2>	. . .
Development company			
Product versions			
Operating System			
Basic functionality			
User interface			
Help for the user			
The price of use			
Availability of free version / plan			
. . .			

For each of the software products, provide and briefly describe the screen forms characterizing the main functionality of the product.

At the end of the subsection, draw a conclusion about the possibility of using certain approaches to software development and the experience of leading companies developing software products in the development of project solutions.

Subsection 1.4. Positioning of the created software (application, module, system)

Positioning is the determination of how your software (application, module, system) fits into the IT market, and how you provide unique value to the target audience. Positioning defines how your software product differs from the competition and why your customers need it, so it's one of the most important steps to success.

The positioning contains a brief description of the following characteristics:

business benefits,

problem definition,

determination of the position of the product being created.

Clause 1.4.1. Business benefits

Business benefits are a summary of the benefits achieved by the project:

relevance, i.e. importance, significance, demand for the product being created today;

purpose, that is, how the product being developed will satisfy the needs of the main users and other interested parties and how it will be implemented.

Clause 1.4.2. Problem definition

When defining the problem - a summary of the problem that is solved by the project is given. To define the problem, use the following format (Table 5).

Table 5

Problem definition

Problem	<Description of the problem>
touches	<Stakeholders affected by the problem>
Its consequence is	<What is the impact of the problem>
Successful solution	<List of some key benefits of a successful solution>

Clause 1.4.3. Definition of position

When defining the position of the product being created, the unique position in the market that the software product intends to fill is described at the highest level. The definition of the position is described in the following format (Table 6)

Table 6

Defining the position of the software product

For	<target customer>
Which	<determination of needs and opportunities>
<Product name>	is <software product category>
Which	<definition of the key advantage (the reason that prompts to purchase / use the product>
Unlike	<main competitive alternative>
Our product	<definition of the main difference>

SECTION 2. SOFTWARE REQUIREMENTS SPECIFICATION

The purpose of Chapter 2 is to develop and detail the requirements for the software (application, module, system, etc.) being developed. This section contains a glossary, development of use cases, specifications of functional and non-functional requirements according to the RUP methodology. If a different approach to the specification of requirements is used, then the structure of the section may be different in agreement with the manager.

Subsection 2.1. Glossary

The glossary is a dictionary of the main terms used in the project. This document is the first result of a conceptual analysis of the subject area. The glossary can be considered as a document certifying the common understanding of the basic terminology by the Customer and the Developer.

In addition, the glossary is a starting point for building more detailed models of the subject area, which at the stage of implementation of the information system form the basis of the object model (for object-oriented applications) and the data model (for generating the database schema).

The glossary must be submitted in the form of a table. 7.

Table 7

Glossary

Term	Description of the term
1. Basic concepts and categories of the subject area and the project	
2. System users	
3. Input and output documents	

Subsection 2.2. Development of use cases

Development of use cases contains:

- diagram of use cases;
- specification of use cases.

Clause 2.2.1. Use case diagram

A use case diagram represents the functionality that will be implemented in the software product. A use case can be considered as a function implemented by the system. However, any feature must have value and provide an opportunity to get the end result for the end user of the product or service. Therefore, when specifying the use case, among all the functionality of the system, only that functionality is singled out, which:

- useful to a specific end user;
- allows the user to receive specific finished results.

Before proceeding to the actual specification of requirements in the form of a use case, the RUP usually compiles a register (list) of actors (Table 8) and use cases (Table 9).

An actor is an actor, an entity external to the designed system that interacts with the system and uses its functionality to achieve a certain goal or solve private tasks. The actor will usually be the system user. In addition to the user, another software system, hardware device, time can be considered as an actor. Searching for system actors usually boils down to analyzing the roles of different users. The choice of actors depends on their functional responsibilities, demarcation of access, ways of using the information system.

A use case is a description of a sequence of actions that a system can perform in response to external influences of actors. The purpose of a use case is to define a finished aspect or piece of behavior of a system that has its own behavior without revealing its internal structure.

Table 8

List of actors

Actor	Brief description

Table 9

List of usage options

Option of use	Brief description of the use case	Actor	Main/auxiliary

The next step is to decompose the main use cases to describe them in more detail. Additional use cases detailing the main purpose are considered (for example, for the main use case "Formation of an order", additional options "Create an order", "Calculate a discount", etc.) are allocated. In a use case diagram, the relationships between primary use cases and secondary use cases are defined by generalization, inclusion, and extension relationships. The results of the decomposition should be presented in the table. 10.

Table 10

Decomposition of use cases

The main option using	Detailed usage option	Communication type

Next, a diagram of use cases in the environment of a particular CASE tool is created.

Clause 2.2.2. Specification of use cases

Each use case should have a description. In the diploma project, you should provide a description of the use cases that implement the main functionality (usually in addition to maintaining directories) in the form of a table (Table 11).

Table 11

Use case <Name>

Option of use	<ID><Use case name, in verb form with explanatory words>
Context of use	<Description of a set of functions combined by a context>
The characters	<List of actors>
Prerequisites	<Description of actions to be performed before executing the use case>
Trigger	<Description of the event that triggers the execution of the use case>

End of table 11

Post-conditions	<Set of states guaranteed by the system regardless of the success of the use case execution>
The main stream	1. Actor <action> 2. System <action> 3. Actor <action> . . .
Expansion	4a. The name of the extension point for a specific script 4a.1. System <action> 4a.2. System <action> . . .

The name is a short phrase in the form of a verb in the indefinite form of the perfect form or a verbal noun that reflects the purpose.

Context - concisely describes in one paragraph what the option should do and what the final result is expected from it.

The active persons are:

Main (primary) actors.

Supporting (secondary) actors.

A prerequisite is a use case that must be fulfilled in order for this case to be executed. The prerequisite describes the state in which the system must be before the execution of the variant begins.

A trigger is a domain event that causes a use case to be used. Sometimes a trigger precedes the first step of a use case, and sometimes it is the first step itself.

Posthumous – a usage option that must be executed after the execution of this option; this is the state in which the system should be after the completion of the option; it is what is guaranteed to the participating actors, regardless of the success of the execution of the given option. For example, in the event of a failed transaction, all data that was in the system before it started is kept unchanged.

Events described by preconditions or postconditions must be states that the user can observe.

The main flow - the numbered steps of typical actions starting from the trigger until achieving a guarantee of success are listed as a certain scenario of interaction between the system and the user.

Extensions are exceptions to the steps of the main flow, which are processed according to their algorithms according to the business logic of the process. An extension starts with an extension point name that begins with a number step of the main flow of adding a letter of the Latin alphabet (a, b, c), followed by a sequence of actions of the system and the user.

Subsection 2.3. Specification of functional requirements

To specify the functional requirements defined in the previous paragraph, it is necessary to define their attributes:

priority – filled in by the analyst, shows the priority of implementing the requirement for the client. It is used in project management and determines the priority of requirement development. Possible values: mandatory, recommended, optional (optional);

difficulty – filled in by the project manager, shows the level of labor costs associated with the implementation of the requirement. It is used in project management and affects the sequence of development. The difficulty of fulfilling the requirement can be expressed in the form of labor intensity and indicate the number of man-days required for its implementation or in the form of scale values: high, medium, low. High difficulty is the likelihood that the requirement is very expensive in terms of resources or money. It must be performed first or it is abandoned;

influence on architecture - filled in by the architect. Indicates whether the use cases will affect the core of the software architecture. The attribute can take the value: yes, if it affects the main part of architectural decisions, or no - if it does not affect. Used to prioritize project execution by use cases. It is necessary to establish in advance which use cases affect the main part of the architectural solutions. These use cases are implemented first.

contact – filled in by the analyst, identifies the interested person who can provide the necessary information about the requirement (contact name). Used to ensure that developers can get the information they need to implement a requirement. The Performer attribute is often used instead of Contact.

If one person acts as a contact in the project, this column can be omitted, but the contact must be identified once in the text of this item. The specification of functional requirements is given in the table (Table 12)

Table 1 2

Specification of functional requirements

Requirem ent ID	Name of requirement (use option)	Requirement attributes			
		Priority	Difficulty	Influence on architecture	Contact / Performer

UC1					
...					

Subsection 2.4. Specification of non-functional requirements

Non-functional requirements can be divided into the following groups:

1. Applicability:

the time required to train ordinary and experienced users;

measurable response time for typical tasks;

basic requirements for applicability of the new system relative to other systems known to users;

requirements to conform to common applicability standards, such as the IBM User Interface Standards or the Microsoft Graphical User Interface Standards for Windows.

2. Reliability:

Availability - determines the % of available time (xx.xx %), usage time, time spent on maintenance, disruption of work mode, etc.

average uptime - usually defined in hours, but can also be defined in days, months or years;

average service life before repair - how long the system is allowed to work before it needs to be serviced;

accuracy - defines the bit rate (resolution) and accuracy (according to some known standard) that are required in the system's output data;

maximum error or defect rate - usually expressed in terms of the number of errors per thousand lines of code or the number of errors per functional unit.

3. Operating characteristics.

Here it is necessary to highlight the specific characteristics of performance, speed, capacity, use of system resources. Where possible, reference should be made to related uses by name.

transaction speed (average value, maximum);

performance (for example, the number of transactions per second);

capacity (for example, the number of customers or transactions that the system can accommodate);

reduced performance modes (which is an acceptable mode of operation when the system has become somewhat worse);

resource usage: memory, disk space, communications, etc.

4. Serviceability

This group includes all requirements that extend the usability or reliability of the system being developed, including coding standards, naming conventions, class libraries, and support utilities.

5. Design restrictions.

These requirements should contain all the design limitations to the system being created. Design constraints are decisions that have been formulated as binding and must be firmly adhered to. Examples can be programming languages, requirements for programming technology, mandatory use of development tools, architectural and design limitations of purchased components, class libraries, etc.

6. Requirements for documentation intended for the user and for the help system.

Describe requirements, if any, for interactive user documentation, help system, warning messages, etc.

7. Purchased components.

Describes all purchased components to be used by the system, any applicable licenses or restrictions on use, and any information about compatibility and/or interoperability or interface standards.

8. Interfaces.

Define the interfaces that must be supported by the application. It should contain adequate specifications, protocols, ports and logical addresses, etc., so that the software can be developed and tested for compliance with the requirements of the interfaces.

8.1. User interfaces.

Describes the user interfaces that must be implemented by the software.

8.2. Hardware interfaces.

All hardware interfaces that should be supported by software are defined, including logical structure, physical addresses, expected behavior, etc.

8.3. Software interfaces.

Software interfaces with other components of the software system are described. These may be purchased components, reusable components from another application program, or components developed for subsystems outside the context of these software requirements specifications, but with which this application program must interact.

8.4. Communication interfaces.

Describes all communication interfaces to other systems or devices such as local networks, remote serial devices, etc.

9. Licensing requirements.

Any mandatory licensing requirements or other usage restriction requirements that must be met by the software are identified.

10. Copyright Notice.

Any necessary legal disclaimers, warranties, copyright notices, succession rights, trademarks or logos for the software are described.

11. Used standards

References are given to all used standards and to specific sections of such standards that relate to the described system. For example, it can be legal and regulatory standards, quality standards, industry standards for applicability, interoperability, internationalization, compatibility with the operating system, etc.

In the specification of non-functional requirements, only those requirements that are essential to the project are specified.

The specification of non-functional requirements with their attributes should be listed in the table (Table 13)

Tables this 13

Specification of non-functional requirements

Requirement ID	The name of the requirement	Requirement attributes		
		Priority	Difficulty	Contact
1. Applicability				
SUPP1				
2. Reliability				
. . .				

Section 2.5. User interface design

The user interface is a kind of communication channel through which the user and the program interact. To create an effective interface, you need to understand what tasks users will solve with the help of this program and what interface requirements users may have.

General principles of user interface design:

1. The program should help to perform tasks.

This means that the interface should be easy to learn, not a hurdle the user has to overcome to get started.

2. While working with the program, the user should not feel discomfort.

To implement this principle, it is necessary:

ensure that the results of as many "incorrect" user actions as possible are checked, but do not do it everywhere;

to tell the user exactly what to do and display informational messages in situations when it is really necessary;

provide advanced users with the ability to disable the output of informational messages;

it is good to think about the content of the messages displayed to the user.

The heuristic rules of Jacob Nielsen, an authoritative American expert in the field of interface design, are widely known [38]:

1. System status visibility.
2. Correspondence between the system and the real world.

3. User management and freedom of their actions.
4. Consistency and standards.
5. Preventing errors.
6. Recognition, not recall.
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Helping users identify and correct errors.
10. Help and documentation.

Form design

Forms are the "building blocks" of the user interface.

To create a well-designed form, you need to understand its purpose, how and when to use it, as well as its connections with other elements of the program.

A special type of forms are forms designed for data entry. During development, the main attention should be paid to the speed of their use.

To speed up the data entry process, it is advisable to:

1. If possible, use the same form to add and edit data.
2. Assign keyboard shortcuts for commands.
3. Do not force the user to "jump" from one part of the form to another.
4. Do not make the data entry process dependent on the content of individual control elements of the form.
5. Use means of feedback to the user.

Another important part of form development is the creation of meaningful and effective menus. Here are some important recommendations:

1. Follow standard conventions for menu item placement.
2. Group menu items in a logical order and by content.
3. Use separator lines to group items in drop-down menus.
4. Avoid redundant menus.
5. Avoid top-level menu items that do not have drop-down menus.
6. Remember to use the <…> symbol to indicate menu items that activate dialog boxes.
7. If possible, use keyboard equivalents of commands and "hot" keys.

Here are some recommendations for designing a Web user interface:

1. Minimize the effort a user must make to make navigational decisions.
2. Use one screen whenever possible. This helps users to complete as many tasks as possible without having to navigate too much.

3. When creating multiple pages, combine them into sections. Make it easy to move between sections with a main navigation control and an additional section navigation control.

4. Make sure that key elements such as menus, headings, and other information displayed on all pages are visually consistent.

5. Always think about future expansion. If new functions are expected to be added to the software system in the future, it is first necessary to decide how the navigation will be expanded to effectively add new screens.

In this unit, you need to develop the graphical user interface of the application in the form of a wireframe or mockup, which is used to obtain the approval of the stakeholders for the proposed concept.

The user interface (UI - English User interface) ensures the transfer of information between the human user and the hardware and software components of the computer system.

The purpose of designing the interface is to obtain an early reaction of users to the proposed system concept. Figma, PhotoShop, Marvel, Pencil Project, etc. are used as tools.

Wireframe (English wireframe, frame) – a structural diagram of the arrangement of interface elements with an example of content (text, illustrations, tables) and highlighting, which partially reflects the work with the product. A wireframe is a low-fidelity design image. It should clearly show:

main groups of content (What?);

information structure (Where?);

description and basic visualization of interaction between the interface and the user (How?).

Wireframe does not contain graphic design (visual design). Interface elements are presented in a simplified form, for example, using "fillers" - rectangles crossed by criss-cross lines for images. That's why wireframes are usually called low-fidelity (lo-fi) data. The scheme is supplemented by a text document describing the logic of the product's operation and the user's interaction with it.

A mockup is a medium or highly detailed static representation of a design. It conveys the structure of information, visualizes content and demonstrates basic functionality in the form of static images. Allows you to understand what the final product will look like.

Mockup is a non-clickable, but more visually designed layout (almost a design). He has:

show the information structure;
visualize content;
demonstrate basic functionality in statics;
give an opportunity to evaluate the visual side of the project.

SECTION 3. DESIGN AND TECHNICAL SOLUTIONS

Subsection 3.1. Justification of the architecture of the software system

Reasonable creation of system architecture is design at the highest level. A logical architecture describes a system in terms of its fundamental organization in the form of components, modules, packages, program classes, and subsystems.

The ISO/IEC/IEEE 42010:2022 standard [22] defines architecture as " the fundamental concepts or properties of a system in its environment, embodied in its elements, connections, and principles of its design and evolution."

The main idea of the architecture is to reduce the complexity of the perception of the system due to the separation of powers and the creation of a clear structure. The architecture and design of the software allows you to create a clear structure, according to which it is convenient for programmers to work. Its quality depends on how easy software maintenance, changes, additions and support will be.

The term "Architecture" also refers to the documentation of software architecture. Documenting the software architecture simplifies the process of communication between interested parties, allows you to record decisions about the high-level design of the system made at the early stages of design, and allows you to reuse components of this design and design templates in other projects.

Thus, architecture is a set of elements that have a certain form (properties and restrictions imposed on the elements), and their justification. The justification records the motives for choosing a certain architectural style, elements and limitations. Justification is necessary at the stage of creating the architecture, and is useful in the future. Because an architecture has a set of properties that allow it to satisfy requirements, and not knowing these requirements can lead to changes that break the architecture, but architecture includes properties, not requirements. The architecture should be built to best meet the requirements for the system being created, according to the "form meets function" principle. Therefore, the design of the software architecture is a process that is performed after the stage of analysis and formulation of requirements.

The task of the software architecture design stage is to transform the system requirements into software requirements and build the system architecture based on them. The construction of the system architecture is carried out by determining the goals of the system, its input and output data, decomposing the system into subsystems, components or modules and developing its overall structure. The design of the system architecture can be carried out by various methods (standardized, object-oriented, component-based, etc.), each of which offers its own way of building the architecture, namely, the definition of conceptual, object, and other models with the help of appropriate structural elements (block - schemes, graphs, diagrams, etc.).

The stage of architectural design of information systems can be reflected in terms of architecture description in the chosen architecture description language (Architecture Description Language (ADL)). Such languages include [60] - ArchiMate, Architecture Analysis & Design Language, C4 model (software), Darwin (ADL), EAST-ADL, Wright (ADL). Most of these languages are based on the UML language.

If the architecture description language is UML, then the architecture design is displayed by describing the process of object-oriented decomposition of the system to the level of the list of subsystems and their connections, describing its logical structure in the form of component packages (component diagram).

So, the substantiation of the architecture in a simplified form can be divided into three stages:

1) definition of important requirements for architecture;
2) architectural design (Architectural Design);
3) architectural documentation.

In this subsection of the explanatory note, the acquirer must highlight the adopted architectural decisions, which must be presented in the form of a table - see table 14.

Based on the analysis of the architectural requirements, it is necessary to determine the main principles and solutions that will be the basis of the future application. These are primarily reference architecture, deployment patterns, architectural patterns (styles), architectural tactics, internal and external components. Each decision must be justified with an indication of the requirements that are satisfied in the form of identifiers of the requirements that were assigned in section 2 (tables 12, 13).

In the reference to the requirements, those of the functional requirements that were defined at the previous stage, as those that have an impact on the architecture of the software system (for example, the presence of data flows, the need to transform data, receive data from external sources, etc.) are added. Also in the references to requirements may include quality attributes - those non-functional requirements that can affect the architecture - system extensibility, its availability, the ability to modify, speed, security, etc. At the end, the references should display the architectural constraints from the group of design constraints that were formed in subsection 2.4.

Table 14

Adopted architectural decisions

Architectural solution	Justification of the decision	Link on request

This subsection ends with architectural views designed in the selected architectural design language (for example, UML diagrams of packages (components)) with a description of the purpose of structural units.

Subsection 3.2. Software design

Clause 3.2.1. Selection of means of project implementation

The choice of means of project implementation involves the justification of the programming language(s), the software environment of the designed system (primarily, the operating systems on which the software will work), the database management system, frameworks and libraries, data caching tools, message brokers, etc. .

In clause 3.2.1. it is necessary to substantiate the adopted technical solutions (technology stack) for the implementation of the project, namely:

selection of the main programming language, or if necessary, several programming languages;

selection of the main framework and auxiliary libraries;

selection of a database management system;

software environment and other tools used in the project.

For each technical solution, there should be indications of its purpose and alternative options that were considered. It is recommended to present the selection results in the form of a table (Table 15).

Table 15

Selection of means of project implementation

The name of the tool	Appointment	Alternative options considered	Justification of the choice

Clause 3.2.2. UML class diagram (UML activity diagram)

Software System means software under development. It can be a large collection of many software components, a single program or part of a program.

This point of the explanatory note should contain:

1. *For object-oriented software systems* - a UML class diagram (Class Diagram) that implements the basic business logic of the software system, and its brief description, in which at least the purpose of each class must be specified. *For other software systems* - UML activity diagram (Activity Diagram), which reflects the basic business logic of the software system, and its brief description.
2. A link to the program listing containing the source code that corresponds to the UML diagrams given in point 1. The program listing itself should be in one of the appendices to the explanatory note.

Next, there are methodological recommendations for designing a software system.

Recommendations for the development of a UML class diagram (Class Diagram) that implements the basic business logic of the software system.

A UML class diagram is used to represent the static structure of the software system model. It can reflect, in particular, various relationships between separate entities of the subject area, such as objects and subsystems, and also describes their internal structure and types of relationships.

A class in the UML language is used to denote a set of objects that have the same structure, behavior and relationships with objects of other classes. A class may not have instances or objects. In this case, it is called an abstract class. Graphically, the class is represented in the form of a rectangle, which can additionally be divided into sections by horizontal lines. These sections can specify the class name, attributes, and operations.

Mandatory element of class designation is its name. A class name must be unique within a package described by some set of class diagrams. It is indicated in the first upper section of the rectangle. It is recommended to use nouns written without spaces as class names. It is necessary to remember that it is the names of the classes that will form the dictionary of the subject area. Examples of class names can be such nouns as "Employee", "Company", "Manager", "Client", "Seller", "Manager", "Office" and those that are directly related to the subject area and functional purpose of the designed systems.

In the second section from the top of the class rectangle, its attributes are recorded. Each class attribute corresponds to a separate line of text consisting of the attribute's visibility quantifier, the attribute's name, its multiplicity, the attribute's value type, and possibly its output value.

The visibility quantifier can take one of three possible values:

1. Public. An attribute with this scope is accessible from any other package class in which the chart is defined.
2. Protected. An attribute with this scope is not available to all classes except subclasses of this class.
3. Closed (private). An attribute with this scope is not available to all other classes without exception.

An attribute name is a string of text that is used as an identifier for the corresponding attribute and must therefore be unique within that class. The attribute name is the only required element of the attribute syntax.

The multiplicity of an attribute is characterized by the total number of specific attributes of a certain type that are part of a separate class.

An attribute type is an expression, the semantics of which is determined by the specification language of the corresponding model. In UML notation, the attribute type is sometimes defined depending on the programming language that is intended to be used to implement that model. In the simplest case, the attribute type is indicated by a string of text that has a meaningful meaning within the package or model to which the class in question refers.

Class operations are recorded in the third section from the top of the rectangle. An operation is some service that provides any instance of a class on a certain request. The set of operations characterizes the functional aspect of class behavior. Each operation of the class corresponds to a separate string consisting of the operation visibility quantifier, the name of the operation, an expression of the type returned by the operation.

An operation name is a string of text that is used as an identifier for the corresponding operation and must therefore be unique within that class.

In addition to the internal organization or structure of classes, the corresponding diagram indicates various relationships between classes. At the same time, the set of types of such relationships is fixed in the UML language and defined by the semantics of their types.

The basic relations in the UML language are:

1. Dependency relationship.
2. Association relations.
3. The relation of generalization.
4. Realization relationship.

Recommendations for the development of a UML activity diagram (Activity Diagram), which reflects the basic business logic of the software system.

Activity diagrams are used to show the flow of control in a system and the steps involved in executing a use case. Using activity diagrams, you can model sequential and parallel activities. An activity diagram focuses on the conditions and sequence of execution of some flow.

An activity diagram depicts the flow of control from a start point to an end point, showing the different decision paths that exist during the execution of an activity. It is used to model the workflow taking into account conditions, constraints, sequential and simultaneous actions.

To develop an activity chart, you need to determine:

1. Initial state and final state.
2. Intermediate actions necessary to reach the final state from the initial state.
3. Conditions or constraints that prompt the system to change control flow.

Clause 3.2.3. File structure of the software product (application) project

In this point, it is necessary to outline the content of the file structure of the developed software product, that is, to describe the purpose of the folders and files that make up the source code of the project.

The structure is presented in the form of lists, where each item defines a separate structural unit and provides a description of the purpose and content of this unit. If the software product has a distributed architecture (for example, client-server), then the structure of its client and server parts is given separately. It is recommended to start with the folder structure of the project (part of the project), which can be depicted in the form of a "tree". And, further, display the contents of the folders in the form of lists.

An example of a description of the contents of the folder:

The config folder contains the following files:

- .editorconfig — configuration file of the working IDE on which the project was executed;
- .eslintignore — exception configuration file for the es-lint plugin;
- .eslintrc — es-lint plugin rule configuration file;
- .gitignore — exception configuration file for the version control system;
- .prettierrc — prettier plugin rules configuration file;
- package-lock.json — technical information about installed project modules;
- package.json — a list of project dependencies and supporting scripts.

Subsection 3.3. Data model design

The initial data for designing a data model are:

description of the functional scheme of the enterprise (if available) for which the project is being implemented;

identification of the place and functions of a specific unit in the structure of the entire organization (enterprise);

description of information processing technology within the framework of this subject area of a specific unit with an indication of input and output information, tasks to be solved, functions and work regulations of information processing executors, frequency of performing information processing functions;

analysis of available means of automation of information processing within the framework of this subject area (hardware and software, DBMS);

substantiation of the necessity and possibility of developing an information processing automation module and a database of this module as a constituent element of the organization's general data infrastructure;

analysis and description of tasks that are automated in the module (subsystem) being developed.

The logical data model (logical data model) defines: a set of supported types of data structures; set of valid operations on supported data structures; a set of general data integrity rules that explicitly or implicitly determine the correct state of the database or their changes;

The database consists of two parts: transactional and analytical. The characteristic features of the transactional part are:

1) relational structure (mainly);
2) the possibility of accumulating significant volumes of actual data;
3) the possibility of adding, deleting, editing records;
4) no aggregated (calculated) data;
5) is used to perform various accounting operations;
6) is the basis for the development of the analytical part of the database.

The analytical part of the database is used in the process of operational analysis of information and development of models for decision support systems. The characteristic features of the analytical part of the database are:

1) multidimensional structure (support of MOLAP, ROLAP, HOLAP models);
2) storage of aggregated data;
3) the absence of the possibility of performing data deletion and editing operations and their accumulation, mainly according to chronology.

In the process of developing the transactional part of the database, you can use structural or object-oriented modeling, using appropriate CASE tools: ErWin Data Modeler, IBM Rational Software, etc. When building a data warehouse, it is necessary to justify the choice of storage model (MOLAP or ROLAP cubic model: "star" or "snowflake").

If the topic of the diploma project is related to the creation of complex transactional databases for the information system, it is recommended to include the design of conceptual, logical and physical data models in this unit. In other cases, you can limit yourself to the design of conceptual and logical data models.

Clause 3.3.1. Conceptual infographic design

In this section, the construction of a database-independent data model is performed, which includes the creation of a data dictionary and a global informational data model.

Data dictionary. On the basis of the analysis of incoming and outgoing documents, a model is built for mapping the set of details of the outgoing and incoming documents to a set of data elements to be stored in the database, then the collected information is brought to a form convenient for design. For this, a data dictionary is compiled (Table 16).

Table 1 6

Data dictionary

No for/p	Name element	Identifier	Type and length	Appointment element

Data elements of the dictionary are listed in alphabetical order under the "Name of the element" column in order to further remove homonyms and duplicate elements. If the dictionary contains many items, it is moved to the application. In the "Element assignment" field, you must specify whether the storage element is actual or calculated.

During the design of a global informational data model, it is necessary to identify equivalent entities and their merger, identify categories and synthesis of generalizing entities, identify and eliminate duplication of attributes and relationships. A graphical representation of the global model is built in the form of ERD (IDEF1X notation), class diagram or other notations.

For all entities of the developed system, specifications of integrity constraints and operational rules should be given, namely:

1) limitation of entity attributes (Table 17);
2) restriction of tuples;
3) limitation of uniqueness;
4) dynamic limitations;
5) other restrictions;
6) operational rules;
7) referential integrity rules.

Table 17

Restrictions on entity attributes

No. for/p	The name of the attribute or aggregate	Limits / permissible values	Structure (format)	Condition	Default value

Clause 3.3.2. Designing a logical (physical) data model

The design of the logical data model includes: a graphical representation of the logical model in the form of an ERD (in IDEF1X notation), class diagrams, etc.

Justification of the properties of the base model data includes: a description of the means, tools, and methods used to ensure the following properties of the database: functional completeness; minimal redundancy; database integrity (table, referential integrity, provided by keys and triggers, etc.); consistency; topicality; security; renewability; logical and physical independence; efficiency.

For the case of using relational DBMS, a logical model is presented in the form of an ER diagram, which includes entities (tables), attributes (columns / fields) and relations (keys). If ORM technology is used, the ER diagram is replaced by a class diagram, which represents the classes that represent the data model. When using a NoSQL DBMS (for example, MongoDB), a data schema for collections and relationships between collections, if any, are provided.

Subsection 3.4. Protection of information

The information protection unit involves the development of elements of the security policy for the information system and the software product being developed. The information system security policy should include the following items:

List of resources and circulating information (data) that are critical (confidential) for this information system. Each resource or information should be accompanied by a brief justification of why it should be classified as confidential data. Information covered by the following laws should be highlighted here:

the Law of Ukraine "On Information", which defines the concept of confidential information [40];

the Law of Ukraine "On the Protection of Personal Data", which defines what personal data is and regulates the rules of its processing [40];

the Law of Ukraine "On Banks and Banking Activity", which regulates the rules for using banking information [41].

The list of confidential information circulating in the IS is recommended to be drawn up in the form of tables (see Tables 18 - 19). For each type of information, indicate the business processes in which it is used, and in the "justification" column, indicate why it should be classified as confidential information.

Table 18

List of confidential information circulating in the system

Names of information	The name of the business process in which it is used	Justification

Table 19

List of critically important information system resources

The name of the resource	The name of the business process in which it is used	Justification

1. List of persons having access to confidential information circulating in the information system. In this point, the access rights to the database resources for each user, methods and types of access to the information system network resources should be described, as well as the availability of the resource from the Internet. In terms of availability of resources on the network and the Internet, the necessary ports and protocols required for the proper functioning of the resource must be specified.

The list of persons who process confidential information should be presented in the form of a table (see Table 20). To reduce the number of lines, you can group information that has the same access rights.

Table 20

List of users who have access to confidential information and critical resources

IS user	Confidential information or resource	Access rights					
		Reading	Record	Renewal	Removal	Archiving	Restoration
Administrator	Database passwords	+	+	+	-	+	+

2. Analyze the information system from the point of view of a cryptanalyst. Identify the weak components of the system and form a list of possible security threats. Security threats should be classified in relation to basic security services, such as: confidentiality, integrity, availability, involvement, observability, authenticity, etc. The list of threats is presented in the form of table 21.

Table 21

List of information system security threats

The name of the threat	Where it occurs (business process, resource)	To which resource or information is directed	Which security service is violated

3. In Table 19, for each threat, the purpose of the threat and the possible place of occurrence / penetration of the threat should be indicated. The goal of the threat may be certain system resources, the operation of which may be disrupted in cases of its passage. This is done for the possibility of assessing the damage caused, as well as for the correct choice of protection mechanisms. The point of origin / penetration is a system resource or business process in which a threat can occur or through which a threat can penetrate. The identification of such thin places in the IS will allow to reasonably apply security mechanisms for the relevant system resources.

4. Determine with respect to each security mechanism, by what means it can be implemented and what threats it is aimed at blocking. Protection means are defined as specific software products or specialized service functions of the OS, Web servers, databases. The list of security mechanisms and means of protection is presented in the form of table 22.

Table 2 2

List of security mechanisms covering IS security threats

Safety mechanism	A means of protection	Security threat

5. Give a detailed description of the means of protection used in the software product being developed. It should be noted here the cryptographic protection methods used, user authentication methods, the way passwords are stored in the system, authentication tools when connecting to the database, the use of secure SSL protocols, and so on.

6. For systems designed for many users, it is necessary to consider the possibility of implementing a monitoring security service in the form of logging of the main actions of the user when working with the database and the main events of the system, such as connecting to the database, logging in and out of the user, etc.

Section 3.5. The process of CI continuous integration and continuous delivery of CD software or its components

This subsection addresses the CI/CD processes that apply to software that has been designed and developed. The content and structure of this unit can be adjusted depending on the specific project in agreement with the head of the diploma project.

The method of continuous integration (continues integration, SI) requires the developer to periodically commit code changes to the GIT version repository. It is recommended to publish edits at least once a day. The approach helps to more easily and quickly detect errors and other problems with the quality of development, to eliminate them in a timely manner, without delay in the main processes. The goal of using continuous integration is to provide a consistent and automated way to build and test your application.

Continuous Delivery (Continue Delivery automates the process of implementing the application and making changes to the code, to the prepared server infrastructure. Developers create software using environments suitable for certain stages of work. The key feature of the software is the automation of the process of delivery of changes for all used environments and the implementation of necessary additional mechanisms (for example: sending a request to the server, executing SQL queries, setting messages or restarting, etc.).

CI / CD tools. The idea of continuous integration is based on the use of tools that support this process. One of them is the source code management system (SCM). The software is used to track changes in the source code, helps development teams to combine changes made by different developers (systems – Bitbucket, Github, GitLab).

Other tools are systems supporting creation, testing and implementation in continuous integration mode (systems – Gitlab CI, Jenkins, CircleCI, Travis and others).

The development process begins with choosing a branching strategy in Git. A branching strategy is an important working tool during software development. There are several main strategies:

GitHub Flow;

GitFlow;

Forking Workflow;

GitLab Flow;

Trunk Based Development.

As part of these strategies, it is necessary to accept which branches will be used, for what purposes, how often the code will be merged, when the release will be formed, which additional concepts can still be used. For this, a list of types of branches is compiled (Table 23).

Table 23

Data dictionary

No	Branch name / template	Appointment	Frequency of integration

Infrastructure.

In the work, it is necessary to justify the choice of the type of infrastructure local (On-Premise) or cloud (Cloud) according to the main factors that have the greatest influence on the choice of solution (see tab. 24).

Table 2 4

Key factors of on-premise or cloud infrastructure

Factor	Local environment (on-premise)	Cloud environment (cloud)	What needs to be done
1	2	3	4
costs	are responsible for the ongoing costs of server hardware, power and space.	you only need to pay for the resources you need, without any maintenance and upkeep costs, and the price adjusts depending on how much capacity is consumed	you need to calculate infrastructure support costs and average daily, monthly and quarterly resource consumption for each environment: DEV, QA, PROD

control and security	the business stores all its data and has complete control over what happens to it. Due to special privacy requirements, companies/organizations from highly regulated industries are often reluctant to move to the cloud.	many companies and suppliers are concerned with the problem of data ownership; data and encryption keys reside with your third-party provider and if the unexpected happens and there is downtime, there is a chance that you will not be able to access this data in time	identify infrastructure components such as transmission channels, data processing and storage services; specify the relevant data protection requirements in these components (confidentiality, integrity, availability, observability); provide security mechanisms that ensure these requirements
realization	The company is responsible for supporting the solution and all related processes	Resources are placed on the territory of the service provider, a private, public or hybrid cloud	Describe which resources are hosted (virtual machines with different operating systems and their characteristics), which servers or clusters are used (Docker Swarm, Kubernetes)

The end of the table. 2 4

1	2	3	4
regulatory requirements	for government companies, as well as for a number of businesses whose industry and regulatory documents are strictly regulated, it is very important that data storage clearly meets the requirements of the law.	must exercise due diligence and ensure that their third-party supplier complies with all the various regulatory requirements in their industry; Confidential data must be protected and the confidentiality of customers, partners and employees must be ensured; A service-level agreement is important to ensure that the parties, the supplier and the customer, understand a single standard of service and services, including accessibility requirements.	to analyze the regulatory requirements imposed on your type of company/organization regarding data storage rules, for this it is necessary to be guided by national and international laws and legal acts such as the Law of Ukraine "On the Protection of Personal Data", the General Data Protection Regulation (*GDPR*) and others.

The project should consider how infrastructure is built for each environment. Show which part of the infrastructure will be created and maintained by the DevOps team, that is, will form the platform, and which part will be configured in accordance with the deployment of software services, which is influenced by the team of developers and testers. The platform can be provided either by a Cloud provider or deployed on local infrastructure. One or more infrastructures to be used for the respective environment should be justified.

IaaS (Infrastructure-as-a-Service). According to this model, the consumer receives information and technological resources - virtual servers with a certain computing power and memory volumes. All the "hardware" is handled by the provider. It installs software on it to create virtual machines, but does not install and maintain user software. The provider controls only the physical and virtual infrastructure. Examples of IaaS: IBM Softlayer, Hetzner Cloud, Microsoft Azure, Amazon EC2, GigaCloud, GCP, and for local infrastructure Huper-V Windows Server, KVM Ubuntu Server, Cent OS Server, Debian and others.

PaaS (Platform-as-a-Service) – in this case, the cloud provider provides access to operating systems, development and testing tools, and database management systems. The provider controls not only servers, data storage systems and computing power, but also offers the user a choice of certain platforms and tools for managing them. Examples of PaaS: Google App Engine, IBM Bluemix, Microsoft Azure, VMWare Cloud Foundry, and for local infrastructure deployment of various clusters as a platform: cluster databases, cluster web servers.

Containers-as-a-Service (CaaS) – virtualization of computing power, resources and tools in one container. The service also allows you to manage components using a virtual environment. Examples of CaaS: Docker, Podman, Containerd, LXC, Docker Swarm, Kubernetes.

SaaS (Software-as-a-Service) is when applications and services are developed and maintained by a provider, placed in the cloud and offered to the end user through a browser or an application on his PC. The client only pays the subscription fee (or uses the service free of charge), updating and technical support of programs is handled by the provider. Examples of SaaS: file storage (S3, Dropbox, Google Disk, One Drive), office document package for work (Google Doc, Microsoft Office 365).

Properties such as cost of ownership and operation, performance, scalability, availability, reliability, scalability, security, management and monitoring should be considered when choosing an infrastructure.

Table 25

Characteristics of infrastructures

No	Infrastructure property	Appointment	Type of infrastructure	The environment in which it will be used

The next step is to consider the types of environments and provide their description, such as the local environment of the developer (local environment or work environment), the development environment (DEV ENV), the testing environment (QA environment) and the production environment (production environment).

The developer's local environment is the developer's workstation on which the developer works directly and on which all the necessary components are installed to support the development process. These can be implementations, IDEs, plugins for IDEs, servers, a set of libraries, text editors, and other development tools.

A QA Environment, also known as a test environment, is a test setup consisting of firmware, software, hardware, and related network configuration. All tests are performed on this test bench, which contains all the infrastructure necessary for conducting tests.

A production environment is the location where software or products have been put into operation for use by intended users. When something goes into production, all bugs should already be fixed and the product or update should work perfectly.

For each environment, it is necessary to describe all services that will be deployed on the basis of the selected infrastructure and their characteristics (settings). The description of environments is given in table. 26.

Table 26

Characteristics of environments

No	Environment	Service	End points	Appointment

DevOps pipeline .

A DevOps pipeline is a set of automated processes and tools that enable developers and operations professionals to work together to build and deploy code in a production environment. While the DevOps pipeline can vary by organization, it typically includes build automation/continuous integration, automated testing, validation, and reporting. It may also contain one or more manually operated gates that require human intervention before the code can proceed.

Each project has its own set of technologies that can affect the process. The conveyor can be represented in the form of a diagram in Fig. 1.

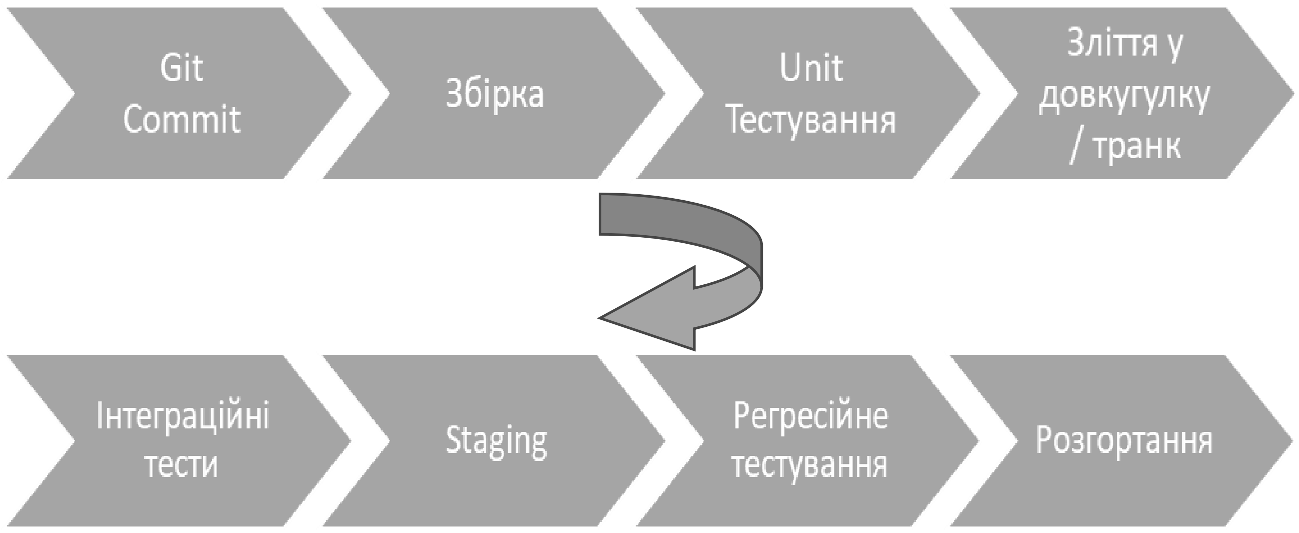

Fig. 1. An example of the stages of the DevOps pipeline.

Although each pipeline is unique, most projects use similar fundamental components. Each step is evaluated for success before proceeding to the next stage of the pipeline. In the event of a failure, the pipeline stops and feedback is provided to the developer.

In the project, you need to describe each stage of the pipeline that you consider necessary for the deployment of your application. Describe scripts, settings of relevant tools, provide instructions for launching the DevOps pipeline. All these settings can also be stored in the corresponding Git repository. To build a DevOps pipeline, such systems as: Jenkns, GitLab CI, GitHub Actions can be used. Corresponding scripts should be added to the annexes of the diploma project.

SECTION 4. SOFTWARE TESTING

The purpose of Section 4 is to create the documentation required for software testing and analysis of test results, including checklists, test cases, test data sets, as well as defect reports and test results reports. Taking into account the topic of the diploma project and upon agreement with the supervisor, the results of other types of testing may be presented in this section.

Subsection 4.1. Creation of checklists

Checklists for smoke testing and critical path testing should be compiled in this section. A checklist is a list that contains the necessary checks (test cases) during software testing. Purpose of the checklist:

1. Do not miss the necessary tests.
2. Divide tasks according to skill levels.

3. Keep reports and test results.

Clause 4.1.1. Checklist for smoke testing

Smoke testing (Smoke test) is aimed at checking the most important, the most important, the most key functionality, the failure of which makes the very idea of using the software pointless. A deeper test should not be conducted until the smoke tests are 100% complete!

To create a checklist for smoke testing, you need to determine:

what your software is for;

what are the most obvious simple steps to take to get into it;

what are the minimum important steps and in what sequence should be performed to achieve the goal.

A checklist for smoke testing is created in the table (Table 27).

Table 27

Checklist for smoke testing

Number	Description	<Environment>	Status	Comment
1				
2				
. . .				

The list of tests should be minimal - up to 10-15, and be linear, without hierarchical arrangement.

Validation environment. As a rule, several columns intended for testing on a separate platform are added to each checklist. The name of these columns indicates the name of the device, browser and its version, etc.

During testing, the status is indicated against each test item in the checklist. Examples of statuses are given in the table. 28.

Table 28

Test statuses

No	Satus	Comment
1	2	3
1	Passed	The check was passed successfully, no bugs were found
2	Failed/ Unsuccessful	One or more bugs found

3	Blocked/ Blocked	Unable to verify because one of the bugs is blocking the current verification
4	Not run	Not yet verified
5	Skipped	There will be no verification for any reason. For example, the current functionality is not yet implemented.
6	In Progress	The current item the tester is working on

Based on the created checklist, it is necessary to check the created software. In the process of passing the tests, change the status of the test in accordance with the table. 12. When detecting defects, insert a note to the corresponding item with a link to the page with the bug. All errors found during this testing phase must be corrected!

Clause 4.1.2. Checklist for critical path testing

Critical path testing aimed at investigating the functionality used by typical users in typical daily activities. There is a majority of users who use some subset of the program's features most often. It is these functions that need to be checked after smoke testing. If for some reason the program does not perform these functions or performs them incorrectly, many users will not be able to achieve many of their goals. The threshold value of successfully passing the critical path test is lower than in smoke testing, but still quite high (usually, about 70-90% - depending on the nature of the project).

To create a checklist for testing the critical path, it is necessary to determine the main steps in working with the software that most users perform to achieve the main goal of its use, and then write down the sequence of actions that must be performed at each step.

A checklist for testing the critical path is created in the table (Table 29).

Table 29

Checklist for critical path testing

Number	Description	Status	Comment
1			
1.1			
1.2			
. . .			
2			
2.1			
. . .			

Subsection 4.2. Creation of test cases

To test the critical path, test cases must be developed in accordance with the checks specified in the checklist. The description of test cases for testing the critical path should be submitted in the table according to the template (Table 30).

Table 30

Template for creating test cases

No	Requirement No	Name	Priority	Description of the test case	Expected result	Comment

Requirements for the content of the test case.

Identifier.

Unique.

Meaningful (if the software allows).

A requirement related to the test case.

Indicates the main requirement for which the test case was created.

The presence of this field improves such a property of the test case as tracking.

Name (essence) of the test case

It is designed to simplify the quick understanding of the main idea of the test case without referring to its other attributes.

This field is most informative when viewing a list of test cases.

Each test case should always have a name! Test cases without names are absolutely not allowed under any circumstances!

Priority.

Shows the importance of the test case.

Can be expressed by letters (A, B, C, D, E), numbers (1, 2, 3, 4, 5), words ("extremely high", "high", "average", "low", "extremely low") or in another convenient way.

May correlate with:

the importance of the requirement;

the potential importance of the defect, which the test case is aimed at finding.

The description of the test case contains:

input data required to execute the test case.

They allow to describe everything that must be prepared before the start of the test case execution, for example, the state of the database, the state of the file system and its objects, and so on.

Everything described in this field is prepared without the use of a test program: if there are problems here, you cannot write a software defect report;

test case steps.

The steps of the test case describe the sequence of actions that must be implemented during the execution of the test case, and are numbered.

Expected results.

The expected results for each step of the test case describe the reaction of the program to the actions described in the steps of the test case. The step number corresponds to the result number.

Section 4.3. Creating input data sets for testing

For test cases that check the operation of the software with input data sets, you need to create a test set for data input based on equivalence classes and boundary conditions.

An equivalence class is a set of data that is processed in the same way and leads to the same result.

Boundary conditions are places (values) in which one equivalence class passes into another.

Signs of equivalence of test cases:

test cases are aimed at finding the same error;

if one of the test cases detects an error, the others will most likely detect it too;

if one of the test cases does not detect an error, the others most likely will not either;

test cases use similar sets of input data;

for creating test cases, the same operations are performed;

test cases generate the same output or bring the software to the same state;

all test cases trigger the same error handling block ("error handling block");

none of the test cases causes the error handling block to trigger.

Equivalence classes must be specified when creating test data sets:

by field length: for unacceptable length and acceptable length;

by characters: for invalid and valid characters.

Based on the selected equivalence classes, determine data sets for positive and negative testing. Submit the results in the table (Table 31).

Table 31

Data sets for testing

	Positive test cases		Negative test cases			
Value						
Explanation						

For positive test cases it is:

a line of minimum acceptable length;

string of the maximum allowable length.

For negative test cases it is:

string of unacceptable length beyond the lower limit;

line of unacceptable length beyond the upper limit;

a string of acceptable length with invalid characters

you can add string with invalid length / invalid characters for reliability.

Section 4.4. Defect report

Based on the results of software critical path testing, a defect report is created based on a checklist, test cases, and data sets. Defect report - a document that provides information about the expected and actual test results and describes and determines the priority of the detected defect, as well as contributes to its elimination. When a defect is detected, it is absolutely necessary to copy a screenshot of the defect and insert it or a link to it in the report's appendices (table 32).

Table 32

Defect report

Identifier	Brief description	Detailed description	Steps of reproduction	Creativity	Importance	Urgency	Symptom	It was possible to bypass the news	Comment	Appendices

A defect is a discrepancy between the expected and the actual result.

The expected result is the behavior of the system described in the requirements.

The actual result is the behavior of the system, which is observed during the testing process.

Requirements for the content of the defect report.

Identifier.

- Unique.
- Meaningful (if the software allows).

Brief description.

In a concise form, it gives a comprehensive answer to the question "What happened?" "Where did it happen"? "Under what conditions did this happen?"

For example, "missing logo on welcome page if user is an administrator."

Detailed description.

Provides detailed information about the defect, as well as (required!)

- description of the actual result,
- description of the expected result,
- link on request (if applicable).

Playback steps.

Describe the actions that must be performed to reproduce the defect. This information in the defect report is extremely important. It allows the developer to quickly reproduce and eliminate the problem.

Reproducibility.

Shows whether the defect is reproduced always ("always") or only sometimes ("sometimes"). Defects that are always reproduced are much easier to diagnose and fix.

Recommendation : reproduce your steps at least 2-3 times before writing that the defect is always reproduced. To prove that the defect exists is the task of the tester! Therefore, as soon as a defect is detected, make a copy of the screen. Even if you yourself will no longer be able to reproduce this defect, perhaps colleagues will understand what is the matter from the resulting picture.

Importance.

Shows the degree of damage caused to the project by the existence of a defect. Typical values:

Critical. These are the worst defects that lead to application or operating system crash, serious database corruption, web server or application server crash.

High (major). Serious defects, such as: loss of user data, crash of a significant part of the functionality of the application, crash of the browser or other client, etc.

Medium (medium). Defects affecting a small set of program functions. As a rule, such defects can be "bypassed", i.e. perform the necessary actions in another way, which does not lead to the occurrence of the defect.

Low (minor). Defects that do not directly interfere with the operation of the software. As a rule, this includes all kinds of cosmetic defects, errors, etc.

Urgency.

Shows how quickly the defect should be eliminated. Typical values:

Highest (ASAP, as soon as possible). Assigned to a defect, the presence of which makes it impossible to continue working on the project or transfer the current version of the project to the customer.

High (high). Assigned to a defect that needs to be fixed as soon as possible.

Normal (normal). Assigned to a defect that should be fixed in the order of the general queue.

Low (low). Assigned to the defect that should be dealt with last (when and if there is time left).

Symptom.

Allows to classify defects according to their typical manifestation. Typical symptom values:

cosmetic defect (cosmetic flaw);

damage / loss of data (data corruption / loss);

documentation issue;

incorrect operation (incorrect operation);

installation problem (installation problem);

localization issue (localization issue);

unimplemented functionality (missing feature);

the problem of scalability (scalability);

low performance (slow performance);

system crash (system crash);

unexpected behavior (unexpected behavior);

unfriendly behavior (unfriendly behavior);

variance from specs;

proposal for improvement (enhancement).

Ability to bypass.

Shows whether there is an alternative sequence of actions, the implementation of which would allow the user to achieve the set goal (bypassing the occurrence of the defect).

Comment.

It can contain any data useful for understanding and correcting the defect.

Appendices.

Contains a list of applications attached to the defect report (crashing screenshots, files, etc.).

Section 4.5. Test results report

In this subsection, it is necessary to summarize the results of the test in the form of a report, which contains:

1. Brief description. In an extremely concise form, it reflects the main achievements, problems, conclusions and recommendations based on the results of testing. For example, during the reporting period, 100% of test cases of smoke testing and XX% of test cases of critical path testing were successfully completed, XX% of high importance tests were implemented correctly, etc.;

2. Description of the testing process. Description of the tasks that were completed during the reporting period. This describes the environment in which testing was performed (OS, software environments, etc.), how certain types of testing were performed (manual or automated), whether retesting was performed and what its result was;

3. Statistics on defects. The statistics are presented in the form of a table , which presents data on the defects detected throughout the project (with classification by stages of the life cycle and importance) (Table 33). This section also contains a graph showing the number of defects found (total and by importance) by time periods;

Table 33

Statistics on defects

Status	the total number of	Number by importance			
		low	average	High	Critical
Found					
Corrected					

Checked					
Reopened					
rejected					

4. Recommendations. Reasoned conclusions about the obtained results and, if necessary, recommendations for a further strategy for improving the quality of the software are presented.

5. Procedure for submission to defense and defense diploma project

5.1. **Advance protection diploma project**

In order to identify the applicant's readiness for defense, a preliminary defense of the diploma project is carried out. Preliminary protection consists of two parts:

a report with a presentation of a fully completed project;

demonstration of the software product.

The purpose of the preliminary defense is to check the applicant's readiness for the defense in accordance with the requirements of the graduation department, to evaluate the scope of the submitted project and the quality of its execution and design. Regardless of the degree of readiness of the project, the acquirer must appear for the preliminary defense.

For the preliminary defense, the graduation department determines the composition of the commissions and draws up the preliminary defense schedule.

The following are submitted for preliminary protection:

completed and timely approved task;

a fully prepared, but not bound, explanatory note with the signatures of the applicant, manager and consultants on the assignment;

finished software product and a video clip of its operation;

report plan and presentation, agreed with the manager.

On the basis of the applicant's report, his answers to the questions, the results of the inspection of the explanatory note, the presentation, the commission determines recommendations to the applicant regarding:

reports;

answers to questions;

content of the explanatory note;

drawing up an explanatory note;

presentations;

software product demonstrations.

Admission to the defense is possible in case of a positive assessment of both types of preliminary defense (explanatory note; program part). Applicants who have not passed the preliminary defense are not allowed to the defense.

5.2. Submission **of the diploma project** ***for defense***

After the preliminary defense and elimination of deficiencies with *the completed* diploma the project *is submitted to the project manager. He finally checks the compliance of the completed work with the task and the relevant requirements, prepares a submission (written feedback), in which he gives a description of the work of the applicant .*

Completed diploma the project *, signed by its manager, is subject to regulatory control. After regulatory control* , the diploma project is signed by the head of the department, the diploma project *is sent* for review.

5.3. ***Defense*** **of the diploma project**

No later than a day before the defense, the applicant submits the diploma project to the secretary of the State *Examination Commission* (SEC). A handout on the work performed for each member of the DEC, which contains a printout of the presentation slides, is mandatory.

Other materials that characterize the scientific and practical value of the completed thesis can be submitted to the DEC project *, namely:*

printed articles on the topic of work;

documents that characterize the practical value of the development of the acquirer ;

documents indicating the practical application of the work (signed by officials);

layouts, product samples, etc.

Protection of diplomas projects *is held at a meeting of the DEC.*

Defense of one diploma the project *should preferably not exceed 30 minutes. The applicant is given no more than 10 minutes to give a report on the project .*

The defense of a complex diploma project is, for the most part, planned and conducted at one meeting of the DEC, and the student who defends first is instructed to report both on the general part of the work and on the individual part with an increase (if necessary) in the time for the report. All applicants who performed complex work must be fully aware of the general part of the work and be ready to answer questions from the commission members not only about the individual, but also about the general part of the work.

The applicant's report should consist of three main parts, namely: introduction, main part and conclusions.

In the introduction, it is necessary to note the relevance of the project topic, give a general analysis of the state of the problem and formulate the main tasks, the solution of which was connected with the implementation of the project.

In the main part of the report, in a concise form, it is necessary to provide a report on the content of the completed developments, show the effectiveness of the adopted technical solutions, and provide a brief report on the results obtained.

In the final part of the report, it is necessary to draw general conclusions and give recommendations regarding the possible scope of application of the design object, list publications by the topic of the work, provide information on implementation.

The report must be accompanied by links to the electronic presentation, which is shown by the applicant . The presentation should contain the following slides:

title slide with initial data on the diploma project;

content of the presentation (with links to relevant slides);

the relevance of the topic and the purpose of the diploma project;

a model of the organizational structure of the enterprise, a division of the enterprise;

business process management model;

UML diagram of use cases;

wireframe or mockup of the user interface project;

mathematical (logical) statement;

filled out forms of outgoing and incoming documents, diagrams, maps;

logical and physical models of the database;

UML-diagram of classes (Class Diagram), which implement the basic business logic of the software system, or UML-diagram of activity (Activity Diagram), which reflects the basic business logic of the software system;

UML-diagram of states (State Diagram), in which elements of the graphical user interface can be located;

the results of software testing and its security testing;

used tools and technologies;

conclusions based on the results of the diploma project;

approval of the results of the diploma project;

final slide.

Demonstration material in the form of a video may be additionally used during the defense.

After the report , the winner briefly answers the questions of the EC members. Then the review is read. The student is given the opportunity to respond to the reviewer's remarks.

After the defense of all declared winners, the commission conducts a closed discussion of each defense and evaluates it according to the evaluation criteria. At the same time, the level of the work performed and the developed software product, the quality of the design of the explanatory note, the level of scientific, practical and theoretical training of the applicant , the rhythm of work on the project, the presence of publications, speeches at conferences, etc. are taken into account.

of the diploma defense project are brought to the attention of the acquirers after the completion of the work of the DEC.

Recommended Books

The main

1. Бородкіна І.Л. Інженерія програмного забезпечення : навч. посіб. / І.Л. Бородкіна, Г.О. Бородкін. - Київ. : ЦУЛ, 2019. - 204 с.
2. Козак О.Л. Опорний конспект лекцій з курсу «Аналіз вимог до програмного забезпечення» для студентів напрямку підготовки «Програмна інженерія» / О.Л. Козак. – Тернопіль, 2021. – 56 с.
3. Кучеров Д.П. Інженерія програмного забезпечення : навчальний посібник / Д.П. Кучеров, Є.Б. Артамонов. - Київ. : НАУ, 2017. - 386 с.
4. Проектування інформаційних систем: Загальні питання теорії проектування ІС (конспект лекцій) [Електронний ресурс]: навч. посіб. для студ. спеціальності 122 «Комп’ютерні науки» / КПІ ім. Ігоря Сікорського; уклад.: О. С. Коваленко, Л. М. Добровська. – Електронні текстові дані (1 файл: 2,02 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. – 192 с.
5. Ушакова І. О. Лабораторний практикум з системного аналізу та проектування інформаційних систем [Електронний ресурс] : навчальний посібник / І. О. Ушакова, І. Б. Медведєва. – Харків : ХНЕУ ім. С. Кузнеця, 2022. – 251 с.
6. Бази даних : лабораторний практикум для студентів галузі знань 12 "Інформаційні технології" першого (бакалаврського) рівня [Електронний ресурс] / укл. В. В. Федько, В. П. Бурдаєв; Харківський національний економічний університет ім. С. Кузнеця. - Х. : ХНЕУ ім. С. Кузнеця, 2019. - 229 с.
7. Якість програмного забезпечення та тестування: базовий курс. Навчальний посібник / За ред. Крепич С.Я., Співак І.Я. / для бакалаврів галузі знань 12 «Інформаційні технології» спеціальності 121 «Інженерія програмного забезпечення». – Тернопіль: ФОП Паляниця В.А., 2020. – 478 с.

Additional

8. Дейт К. Дж. Введення в системи баз даних / К. Дж. Дейт. – 6-е вид. – Київ : Діалектика, 2020 - 1328 с.

9. ДСТУ 8302:2015 "Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання" - Київ : Держстандарт України, 2016. — 16 с.

10. ДСТУ ISO 9000:2015. Системи управління якістю. Основні положення та словник термінів: чинний з 01.07.2016 – Київ. : УкрНДНЦ, 2016. – 49 с.

11. ДСТУ ISO/IEC 25000:2016. Інженерія систем і програмних засобів. Вимоги до якості систем і програмних засобів та її оцінювання (SQuaRE). Настанова до SQuaRE: чинний з 01.01.2018. – Київ. : УкрНДНЦ.

12. ДСТУ ISO/IEC 25010:2016. Інженерія систем і програмних засобів. Вимоги до якості систем і програмних засобів та її оцінювання (SQuaRE). Моделі якості системи та програмних засобів: чинний з 01.01.2018. – Київ. : УкрНДНЦ.

13. ДСТУ ISO/IEC 25020:2016. Інженерія систем і програмних засобів. Вимоги до якості систем і програмних засобів та її оцінювання (SQuaRE). Рамкова модель і настанова щодо вимірювання: чинний з 01.01.2018. – Київ. : УкрНДНЦ.

14. ДСТУ ISO/IEC 25022:2019 (ISO/IEC 25022:2016, IDT) Інженерія систем і програмних засобів. Вимоги до якості систем програмних засобів та їхнього оцінювання (SQuaRE). Вимірювання якості під час застосування. : чинний з 01.01.2020. – Київ. : УкрНДНЦ.

15. ДСТУ ISO/IEC 25023:2019. Інженерія систем і програмних засобів. Вимоги до якості систем програмних засобів та їхнього оцінювання (SQuaRE). Вимірювання якості систем та програмних продуктів: чинний з 01.11.2019. – Київ. : УкрНДНЦ.

16. ДСТУ ISO/IEC 25040:2016 (ISO/IEC 25040:2011, IDT) Інженерія систем і програмних засобів. Вимоги до якості систем і програмних засобів та її оцінювання (SQuaRE). Процес оцінювання: чинний з 01.01.2018. – Київ. : УкрНДНЦ.

17. ДСТУ ISO/IEC/IEEE 15939:2018. Інженерія систем і програмних засобів. Процес вимірювання: чинний з 01.01.2019. – Київ. : УкрНДНЦ.

18. Життєвий цикл програмного забезпечення : навчальний посібник / Є.В. Левус, Т.А. Марусенкова, О.О. Нитребич. - Львів. : Видавництво Львівської політехніки, 2017. – 208 с.

19. Методичні рекомендації до оформлення звітів, курсових проєктів та дипломних робіт (проєктів) для студентів спеціальності 121 "Інженерія програмного забезпечення", 122 "Комп'ютерні науки", 126

"Інформаційні системи і технології": [Електронний ресурс] / уклад. І.О.Ушакова, Г.О. Плеханова, О.М. Беседовський. – Х. : ХНЕУ ім. С. Кузнеця, 2021. – 48 с.

20. Ушакова І. О. Проектування інформаційних систем : Практикум / І. О. Ушакова. – Харків : Вид. ХНЕУ ім. С. Кузнеця, 2015. - 250 с.

21. Developing Information Systems: Practical guidance for IT professional / P. Thompson, D. Paul, A. Paul et al. – London : BCS Learning & Development Limited, 2014. – 206 p.

22. ISO/IEC/IEEE: Systems and Software Engineering – Architecture Description. ISO/IEC/IEEE 42010:2022. - [Electronic resource]. – Access mode - http://www.iso-architecture.org/42010/index.html.

23. Lampathaki F. Business process modelling. Business process reengineering // F.Lampathaki, S. Koussouris, J. Psarras. – Zografou : NTUA, 2013. – 89 p.

24. Standard glossary of terms used in Software Testing. Version 3.1. All Terms [Electronic resource]. – ISTQB, 2024. – 82 p. – Access mode - https://astqb.org/assets/documents/Glossary-of-Software-Testing-Terms-v3.pdf

25. Wiegers K. Software Requirements Essentials: Core Practices for Successful Business Analysis; 1st Edition / K. Wiegers, C. Hokanson. – Addison-Wesley Professional, 2023.– 208 p.

Information resources

26. Інформаційний портал CRM [Електронний ресурс]. — Режим доступу : https://www.crm.com.ua

27. Державна служба статистики України [Електронний ресурс]. — Режим доступу : https://www.ukrstat.gov.ua/.

28. Спільноста розробників dou.ua [Електронний ресурс]. – Режим доступу : https://dou.ua/.

29. Gartner, Inc. [Electronic resource]. — Access mode : http://www.gartner.com/.

30. IBM [Електронний ресурс]. Режим доступу : — http://www.ibm.com/ua-en.

31. Microsoft [Електронний ресурс]. — Режим доступу : http: //www.microsoft.com/.

32. Agile alliance [Electronic resource]. – Access mode : http://www.agilealliance.org.

33. ITC Online [Electronic resource]. — Access mode : http://itc.ua/.

34. Object Management Group [Electronic resource]. – Access mode : http://www.omg.org.

35. SCRUM [Electronic resource]. – Access mode : http://www.scrum.org.

36. UML [Electronic resource]. – Access mode : http://www.uml.org/.

37. Try QA [Electronic resource]. – Access mode : http://tryqa.com/.

38. 10 usability heuristics for user interface design. - [Electronic resource]. – Access mode - https://www.nngroup.com/articles/ten-usability-heuristics/.

39. Про інформацію : Закон України від 02.10.1992 № 2657-XII: станом на 21 березня 2023 р. - — Режим доступу : https://zakon.rada.gov.ua/laws/show/2657-12#Text (дата звернення: 20.01.2024).

40. Про захист персональних даних: Закон України від 01.06.2010 р. № 2297-VI: станом на 27 жовт. 2022 р. — Режим доступу : https://zakon.rada.gov.ua/laws/show/2297-17#Text (дата звернення: 20.01.2024).

41. Про банки і банківську діяльність: Закон України від 07.12.2000 р. № 2121-III: станом на 1 січня 2024 р. — Режим доступу : https://zakon.rada.gov.ua/laws/show/2121-14#Text (дата звернення: 20.01.2024).

42. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). – Режим доступу: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679 (дата звернення: 12.02.2024).

Appendices

Appendix A

A sample of the student's application for approval of the topic of the diploma project

To the head of the department
information systems
Associate Professor Bondarenko
D.O.
student of the 4th year <number>
of the group
<P. I. B student>

STATEMENT

Please approve the topic of the diploma project <Topic name>.

<Date> <Signature of the student> <P. I. B student>

Head
diploma project

<Position

of science degree, academic title> <Head's signature> <P. I. B. of the head>

Appendix B

The structure of the substantive part of the diploma project of a practical nature

No. z/p	*The name of the structural element*
1.	ANALYSIS OF SUBJECT FIELD <NAME OF SUBJECT FIELD>
1.1.	Definition of content of the subject area
1.2.	Modeling of the subject area
1.3.	Review and analysis of literary sources regarding existing solutions and analogues
1.4.	Positioning of the created software (application, module, system)
1.4.1.	Business benefits
1.4.2.	Problem definition
1.4.3.	Definition of position
2.	SOFTWARE REQUIREMENTS SPECIFICATION
2.1.	*Glossary*
2.2.	*Development of use cases*
2.2.1.	*Use case diagram*
2.2.2.	*Specification of use cases*
2.3.	*Specification of* functional requirements
2.4.	*Specification of non-functional requirements*
2.5.	User interface design
3.	*DESIGN AND TECHNICAL SOLUTIONS*
3.1.	*Justification of the architecture of the software system*
3.2.	*Software design*
3.2.1.	*Selection of means of project implementation*
3.2.2 .	*UML class diagram (UML activity diagram)*
3.2.3.	*File structure of the software product (application) project*
3.3.	*Data model design*
3.3.1.	Conceptual infographic design
3.3.2.	Designing a logical (physical) data model
3.4.	*Protection of information*
3.5.	*The process of CI continuous integration and continuous delivery of CD software or its components*
4.	*SOFTWARE TESTING*
4.1.	*Creation of checklists*
4.1.1.	*Checklist for smoke testing.*
4.1.2.	*Checklist for critical path testing*
4.2.	*Creation of test cases*
4.3.	*Creating input data sets for testing*
4.4.	*Defect report*
4.5.	*Test results report*

Appendix C

FROM the title page of the diploma project

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
KHARKIV NATIONAL ECONOMIC UNIVERSITY
THE NAMES OF SEMEN KUZNETSYA

FACULTY OF INFORMATION TECHNOLOGIES

DEPARTMENT OF INFORMATION SYSTEMS

Level of higher education	First (undergraduate)
Specialty	Software engineering
Educational program	Software engineering
Group	<Group code>

DIPLOMA PROJECT

on the topic: "Topic name"

Performed by: student <Student's name SURNAME>

Supervisor: <academic degree>, <academic title> <First name SURNAME>

Reviewer: <position> <First Name SURNAME>

Kharkiv - 202_ year

Appendix E

Examples of essays

1. For research projects

ABSTRACT

Explanatory note to the diploma project: 81 pages, 34 figures, 6 tables, 2 appendices, 35 sources.

The object of research is the methods of mathematical modeling of information systems.

The purpose of the work is to research approaches to the development of simulation models of information systems functioning processes and automation of the simulation modeling process.

As development methods, the analysis method for researching existing modeling methods, the synthesis method for combining the advantages of existing modeling methods, modeling methods for presenting and researching the processes of functioning of information systems, the method of comparative analysis for assessing the adequacy of the model of the processes of information systems functioning were chosen.

As a result of the work, a rational approach to the development of simulation models of the functioning of information systems was substantiated, and a software tool for automating simulation modeling was developed, which allows creating simulation models of the processes of functioning of information systems and researching them.

The results of the research can be used in research institutions and divisions of enterprises engaged in the development of simulation models.

INFORMATION SYSTEM, E-NETWORK, MODEL ADEQUACY, SIMULATION MODEL, MATHEMATICAL SCHEME, SIMULATION METHODS

ABSTRACT

The bachelor's thesis report: 81 pages, 34 figures, 6 tables, 2 appendices, 35 sources.

The object of the research is the methods of mathematical modeling of information systems.

The purpose of the work is to research the approaches to the development of simulation models of the functioning processes of information systems and automation of the simulation process.

The research methods are analysis to research existing methods of modeling, synthesis to unite the advantages of the available methods of modeling, modeling to present and research the functioning processes of information systems, comparison to estimate the adequacy of the models.

As a result of the work, a rational approach to the development of the simulation models of the functioning processes of the information systems has been substantiated. Also a software product has been developed, which helps to create simulation models of the functioning processes of information systems and research the models.

The results of the research can be used in research establishments and the departments of enterprises which develop simulation models.

INFORMATION SYSTEM, E-NET, MODEL ADEQUACY, SIMULATION MODEL, MATHEMATICAL SCHEME, METHODS OF MODELING

ABSTRACT

Explanatory note for the diploma project: 90 pages, 30 figures, 20 tables, 12 appendices, 55 sources.

Design objects are functional elementb, architecture, information and software of the module for monitoring warehouse deliveries of "SIGMA" LLC.

The purpose of the design is to create a module "Monitoring of warehouse deliveries".

The design method is the use of software systems ARIS Toolset, IBM Rational, Microsoft Visual Studio.

The created module makes it possible to increase the reliability of the accounting of goods in the warehouse, the reasonableness and efficiency of decision-making during the management of the supply of goods and their stocks in the warehouse.

The results of the development can be implemented in commercial enterprises.

SUPPLY CHAIN MANAGEMENT SYSTEM, OBJECT-ORIENTED DESIGN, CASE DIAGRAMS, DATA BASE, WAREHOUSE DELIVERY MONITORING, WEB SERVICE

ABSTRACT

The bachelor's thesis report: 90 pages, 30 figures, 20 tables, 12 appendices, 55 sources.

The objects of designing are functional elements, architecture and software of a module which provides monitoring of warehouse supplies.

The purpose of designing is to create "The monitoring of warehouse supplies" software module for Sigma Ltd.

The method of designing is using the ARIS Toolset, IBM Rational, Microsoft Visual Studio software systems.

The advantage of the developed module is the increasing of reliability of goods accounting, validity and timeliness of decision-making.

The obtained results can be applied at commercial enterprises.

SUPPLY CHAIN MANAGEMENT SYSTEM, OBJECT ORIENTED DESIGN, CASE DIAGRAMS, DATABASE, MONITORING OF WAREHOUSE SUPPLIES, WEB SERVICE

Appendix F

Sample content design

CONTENT

CONTENT

EDUCATIONAL EDITION

Guidelines before the performance of qualification works for students of higher education specialty 121 "Software engineering" of the educational program "Software Engineering" of the first (bachelor's) level

Independent electronic text network publication

Compilers: Yurii Eduardovych **Parfyonov**
Andriy Oleksandrovich **Polyakov**
Iryna Oleksiivna **Ushakova**
Oleg **Frolov**

D. O. Bondarenko is responsible for the publication

Editor V. Yu. Stepanenko

Corrector

Plan 2024. Pos. No. 82 EV. Volume 80 p.

Publisher and manufacturer — Khneu, 61166, Kharkiv, ave. Sciences, 9-A

Certificate of entry of the subject of the publishing business into the State Register, ***Dk No. 4853 dated February 20, 2015.***