KHMELNYTSKYI NATIONAL UNIVERSITY

QUALIFICATION WORK

Guidelines for students of the first (Bachelor's) level of higher education, Programme Subject Area 121 Software Engineering

.

Approved at the meeting of
the Department of Software Engineering,
Minutes No. 1 dated 01.09.2023

Khmelnytskyi 2023

Qualification work : Guidelines for students of the first (Bachelor's) level of higher education, Programme Subject Area 121 Software Engineering / Radelchuk H.I., Bedratyuk L.P. – Khmelnytskyi : KhNU, 2023. – 60 p.

Compilers: Radelchuk H.I., PhD, Assoc. Prof.;
Bedratyuk L.P., Dr. Sci., Prof.

Responsible for the issue: Bedratyuk L.P.

.

ВСТУП

Бакалавр – освітній ступінь, що здобувається на першому рівні вищої освіти, який відповідає шостому кваліфікаційному рівню Національної рамки кваліфікацій та присуджується здобувачеві вищої освіти в результаті успішного виконання ним освітньо-професійної програми (далі – освітня програма) підготовки бакалавра, яка завершується атестацією.

Нормативною формою атестації здобувачів першого (бакалаврського) рівня вищої освіти за спеціальністю 121 «Інженерія програмного забезпечення» є публічний захист ***кваліфікаційної роботи*** (КвР). КвР виконується відповідно до стандарту вищої освіти спеціальності та освітньої програми і передбачає розв'язання спеціалізованого завдання або практичної задачі інженерії програмного забезпечення із застосуванням теорій і методів інформаційних технологій.

Атестація здобувачів здійснюється Екзаменаційною комісією (ЕК) після повного виконання ними відповідної освітньої програми з метою встановлення відповідності набутих ними загальних та фахових компетентностей і засвоєних програмних результатів навчання вимогам, передбаченим відповідним рівнем Національної рамки кваліфікацій і нормативним змістом освітньої програми підготовки фахівців за спеціальністю «Інженерія програмного забезпечення», визначеного стандартом вищої освіти для цієї спеціальності.

Методичні настанови щодо виконання КвР містять вимоги до обсягу, структури, змісту, оформлення та захисту КвР з урахуванням специфіки спеціальності та майбутньої професійної діяльності бакалаврів, форми необхідних документів, приклади, зразки тощо, а також роз'яснення та рекомендації, які дозволяють студентам технічно грамотно виконати КвР та підготуватись до її захисту.

При виконанні КвР здобувачам слід дотримуватись вимог до її оформлення у відповідності до державних стандартів та нормативних документів університету.

Методичні настанови розроблені на підставі стандарту вищої освіти та освітньо-професійної програми спеціальності, навчального плану і відповідно до Положення про атестацію здобувачів вищої освіти та наукових ступенів у Хмельницькому національному університеті.

Укладачі сподіваються, що методичні поради допоможуть продуктивній роботі здобувачів, дозволять уникнути зайвих витрат часу під час виконання КвР та сприятимуть підвищенню їх якості.

INTRODUCTION

A ***Bachelor's degree*** is an educational level obtained at the first level of higher education, corresponding to the sixth qualification level of the National Qualifications Framework. It is awarded to a student as a result of successful completion of an educational-professional programme (hereinafter referred to as the educational programme) for Bachelor's preparation, culminating in an examination.

The regulated form of examination for students of the first (Bachelor's) level of higher education in the Programme Subject Area 121 Software Engineering is the public defence of a ***qualification work*** (QW). The QW is carried out in accordance with the higher education standard of the Programme Subject Area and the educational programme and involves solving a specialized task or practical problem of software engineering using theories and methods of information technology.

The examination of students is conducted by the Examination Commission (EC) after their full completion of the respective educational programme to determine the correspondence of their acquired general and professional competencies and the achieved learning outcomes to the requirements set by the respective level of the National Qualifications Framework and the normative content of the educational programme for training specialists in "Software Engineering", as defined by the higher education standard for this Programme Subject Area.

Guidelines for the preparation of the QW contain requirements for its volume, structure, content, design, and defence, taking into account the specifics of the Programme Subject Area and the future professional activity of the Bachelor, forms of necessary documents, examples, samples, etc., as well as explanations and recommendations that allow students to execute the QW and prepare for its defence technically competently.

When preparing the QW, students should adhere to its design requirements in accordance with state standards and regulatory documents of the university.

The guidelines are developed based on the higher education standard, the educational-professional programme of the Programme Subject Area, the curriculum, and in accordance with the Regulations on the certification of students and academic degrees at Khmelnytskyi National University.

The authors hope that the methodological advice will assist the productive work of the students, allow them to avoid unnecessary time expenditures during the preparation of the QW, and contribute to improving its quality.

1. OBJECTIVES AND TASKS OF THE QUALIFICATION WORK

The qualification work summarizes the student's education and characterizes their readiness for work in the field of "Software Engineering".

The purpose of the QW is to complete the training of a specialist in "Software Engineering", to evaluate their professional competencies during the defence before the Examination Commission of an independently executed project, to confirm the student's corresponding educational level, in particular, their ability to solve complex, specialized tasks or practical problems of software engineering (SE) characterized by complexity and uncertainty of conditions, using theories and methods of information technology.

In line with the purpose, the QW involves the student addressing one of the current practical issues in the field of SE, as well as obtaining a specific applied result in the form of a completed and functionally suitable software tool.

The ***main tasks*** of the QW are:

- Systematization, consolidation, and expansion of theoretical knowledge and practical skills in the profession;
- Development of skills and abilities to conduct targeted information searches in printed publications and on the Internet;
- Application of acquired knowledge and skills in solving specific technical, engineering, and production tasks in the field of information technology and SE;
- Mastery of modern design methodologies for mathematical, algorithmic, and software using contemporary methodologies, methods, and information technologies;
- Development of skills for independent work and mastering software development technologies for completed software products at all stages of the software life cycle (SLC);
- Development of analytical, graphical, and literary text presentation skills, design of relevant textual, software, and illustrative material in the form of project documentation, calculation and justification of adopted decisions, as well as gaining experience in presenting and publicly defending one's work;
- Gaining primary experience by students in the independent development of completed software products;
- Evaluating the student's readiness for independent professional activity under modern conditions.

Working on the QW involves reinforcing, improving, and expanding a range of ***competencies*** defined in the educational programme of the Programme Subject Area, including abilities: for abstract thinking, analysis, and synthesis; to apply knowledge in practical situations; to communicate in the state language both orally and in writing; to communicate in a foreign language both orally and in writing; to learn and master modern knowledge; ability to search, process, and analyse information from various sources; to work in a team; to act based on ethical considerations; to strive to preserve the environment; to act socially responsibly and consciously; to realize their rights and obligations as a member of society, to recognize the values of a civic (free democratic) society and the need for its sustainable development, the rule of law, human and citizen rights in Ukraine; to preserve and increase the moral, cultural, scientific values and achievements of society based on understanding the history and patterns of development of the subject area, its place in the general system of knowledge about nature and society and in the development of society, technology, and technologies, to use various types and forms of motor activity for active recreation and leading a healthy lifestyle; to identify, classify, and formulate software requirements; to participate in software design, including conducting modelling (formal description) of its structure, behaviour, and operating processes; to develop architectures, modules, and components of software systems; to formulate and ensure software quality requirements in accordance with customer requirements, technical specifications, and standards; to adhere to specifications, standards, rules, and recommendations in the professional field when implementing SLC processes; to analyse, select, and apply methods and tools to ensure information security (including cybersecurity); to possess knowledge about information data models and create software for storing, extracting, and processing data; to apply fundamental and interdisciplinary knowledge for the successful solution of SE tasks; to evaluate and take into account economic, social, technological, and ecological factors affecting the field of professional activity; to accumulate, process, and systematize professional knowledge regarding the creation and support of software and recognize the importance of lifelong learning; to implement phases and iterations of the SLC of software systems and information technologies based on relevant models and approaches to software development; to implement the system integration process, apply standards and change management procedures to support the integrity, overall functionality, and reliability of the software; to reasonably choose and master the toolkit for software development and support; ability for algorithmic and logical thinking; ability for parallel work on the stages of the SLC of software and effective interaction between the performers of the stages.

A student who has successfully completed the QW should achieve and refine the following ***learning outcomes***:

Analyse, purposefully search for, and select the necessary informational and reference resources and knowledge for solving professional tasks, considering modern scientific and technical achievements;

Know the code of professional ethics, understand the social significance and cultural aspects of software engineering, and adhere to them in professional activities; Know the main processes, phases, and iterations of the software life cycle; Know and apply professional standards and other regulatory documents in the field of SE; Know and apply relevant mathematical concepts, methods of domain, system, and object-oriented analysis, and mathematical modelling for software development; Ability to choose and use an appropriate methodology for software creation; Know and apply in practice fundamental concepts, paradigms, and main principles of the functioning of linguistic, instrumental, and computational tools of software engineering; Ability to develop a human-machine interface; Know and be able to use methods and tools for collecting, formulating, and analysing software requirements; conduct a pre-project survey of the subject area, system analysis of the design object; Choose input data for design, guided by formal methods of requirement description and modelling; Apply effective software design approaches in practice; Know and apply algorithm development methods, software construction, and data and knowledge structures; Apply domain analysis, design, testing, visualization, measurement, and software documentation software tools in practice; Motivatedly choose programming languages and development technologies for software creation and support tasks; Have skills in team development, coordination, design, and release of all types of software documentation; Ability to apply software component development methods; Know and be able to apply information technology for data processing, storage, and transmission; Know and be able to apply software verification and validation methods; Know approaches to software quality assessment and assurance; Know, analyse, select, and competently apply tools to ensure information security (including cybersecurity) and data integrity according to the solved applied tasks and created software systems; Know and be able to apply project management methods; Be able to document and present software development results; conduct an economic efficiency calculation of software systems; Be able to effectively collaborate with performers of various SLC stages, coordinating simultaneous work on stages; Know and be able to apply methodologies, tools, and strategies for resource, time, and communication management to ensure effective parallel work on different SLC stages; Professionally develop, process Ukrainian and English sources of the subject area, realize the need for lifelong learning to deepen acquired and acquire new professional knowledge in the field of software engineering, adapt to work in a specific profession, promote leading an active and healthy lifestyle as an effective component of professional development; Associate oneself as a member of civil society, the scietific community, recognize the rule of law, especially in professional activities, understand and be able to use one's rights and freedoms, show respect for the rights and freedoms of other individuals.

The QW should not contain academic plagiarism, fabrication, or falsification. To ensure accessibility and transparency, the QW is published in the institutional repository of Khmelnytskyi National University. The student - the author of the qualification work - is responsible for all information presented in the QW, the use of factual material and other information during the QW execution, and the validity and reliability of conclusions and positions.

2 *ORGANIZATION* OF THE EXECUTION OF THE QUALIFICATION WORK

According to the curriculum for the training of Bachelors majoring in "Software Engineering", the diploma design process includes: the execution of the QW, preliminary defence of the QW at the Software Engineering Department, and defence of the QW before the Examination Committee. The tentative calendar schedule for the execution of the QW is presented in Table 2.1.

Table 2.1 - Tentative calendar schedule for the execution of the QW.

Content of the work	Scheduled Period
1 Research of the business entity according to the QW topic; defining tasks and requirements for the software, developing technical specifications	1st – 3rd week
2 Design and development of the general architecture and structure of the software user interface; selection of software implementation tools	4th – 8th week
3 Software implementation and testing	9th – 12th week
4 Writing the explanatory note text and developing graphic materials	13th – 14th week
5 Preliminary defence of the QW	May (according to the schedule)
6 Final adjustments to the QW considering the supervisor's remarks; formatting the QW as a document according to requirements	15th week
7 Obtaining accompanying documents (supervisor's feedback, review, plagiarism check reports); undergoing standard control; binding (stitching) the explanatory note	16th week
8 Submitting the QW to the department; preparing the QW for placement in the university repository; preparing for and defending the QW	June(according to the schedule)

The qualification work (QW) is performed by the student independently under the supervision of a mentor. Leading professors of the Software Engineering department supervise the QW. They are appointed by the department and approved by the rector's order.

The student, while working on the QW, must:

– Choose and agree on the topic with the mentor and receive the QW task before starting the professional internship;

– Collect the necessary material for the work during the internship according to the QW topic and individual task;

– Follow the QW execution calendar plan; – Regularly attend consultations with the QW mentor;

– Undergo the preliminary defence of the QW at the department; – Format the QW according to requirements;

– Defend the QW in front of the Examination Committee (EC) according to the schedule.

The student has the right to:

– Receive consultations of any level regarding the QW;

– Use all the scientific and methodological materials available at the department and the university library.

The QW supervisor must:

– Set a consultation schedule (including online and video conferencing) and adhere to it;

– Advise students on various issues (choosing the QW topic, developing a design plan, selecting literature and other sources, executing and formatting the work, preparing the QW for defence, etc.);

– Define the phased deadlines for the work; – Monitor the progress and status of the QW;

– Periodically provide information about the progress of the individual work schedule of their assigned students to the department head;

– Inform at department meetings about students' adherence to the calendar plan;

– Check and evaluate the QW;

– Assist the student in preparing for the QW defence.

The supervisor has the right, in case of violation of the QW execution deadlines, low quality, or independent work, to suggest to the department head the expulsion of the student for not following the individual study plan.

***The department head must*:**

– Organize methodological and informational support for the QW execution;

– Monitor the consultation schedule;

– Consider the status of the QW execution at department meetings;

– Resolve disputes arising between the mentor and the student;

– Control the objectivity of the QW assessment;

– Approve the QW for defence.

The department head ***has the right*** not to allow the student to defend the QW if it does not meet the established requirements.

General control over the progress of the QW is carried out by the Software Engineering department.

3 THEMATICS **OF QUALIFICATION WORKS**

The theme of the Qualification Work (QW) should be oriented towards the research and development of software creation for various business entities: enterprises, insurance companies, financial and credit institutions, firms and companies of various business directions, educational institutions, service and entertainment sectors, document management, clerical work, etc.

The QW theme should meet the following requirements:

− Correspondence to the modern state of science, technology, methods, and tools for software development; relevance; practicality.

− Correspondence to the generalized object of the Bachelor's professional activity in the Programme Subject Area 121 Software Engineering.

− Complexity sufficient to demonstrate the theoretical knowledge and practical skills acquired by the student during the studies.

The title of the QW should be specific, containing the procedure of activity and the product to be obtained as a result of the QW execution.

The QW title should not contain abbreviations of words (phrases) or acronyms (except for generally accepted ones).

The QW is performed by the student independently and should have a completed character.

The QW theme can be implemented by several students (complex QW), but each QW should be completed within the theme. A complex QW is dedicated to solving two or more interrelated tasks united by a single goal. The title of such work consists of two sentences: "*General theme. The student's work theme*."

For students who combine studies with work, it is advisable to perform the QW based on the company where they work. According to the educational programme, students are recommended the following typical design directions:

− Software for automating the activities of various spheres (enterprises, organizations, companies, firms, etc.).

− Software for the operation of distributed reference-information systems.

− Development of client and/or server applications for creating and supporting modern websites.

− Software tools for mobile devices operating under different operating systems, etc.

Examples of typical QW themes:

1. Software for automating the planning and management of a company's advertising activities.
2. Software system for remote control of orders in catering establishments.

3. Web-oriented system for dry cleaning client management.
4. Web application for automating the dispatcher's workplace in a courier delivery service.
5. Reference-information system of a children's club (amusement park, sanatorium, travel agency, beauty salon, etc.).
6. Mobile version of an online store on the Android platform.
7. Network game Android application "Ping-pong" based on the cross-platform framework "Cocos2d-x".

The QW thematic is developed by the Software Engineering department at the beginning of the academic year according to the goals and tasks of the educational programme and labour market requests. The student also has the right to propose their own QW theme with justification of its feasibility and with the agreement of the supervisor.

The thematic of qualification works are approved by the rector's order on the submission of the head of the Software Engineering department. This order also appoints QW supervisors.

4 STRUCTURE **AND CONTENT OF THE QUALIFICATION WORK**

4.1 General requirements for the qualification work

The qualification work (QW) includes an explanatory note, a graphical part, and the developed software tool. The explanatory note is a textual document that provides justification, calculation, and description of the analytical, design, and software solutions adopted in the QW. It should reveal the creative concept of the project, include design models and methods, used algorithms and solution technologies, describe software implementation, conducted experiments (software testing), their analysis, conclusions, etc., and, if necessary, be accompanied by tables, illustrations, charts, etc.

The explanatory note consists of the following elements:

− Title page;
− Task for the qualification work;
− Abstract;
− Table of contents;
− Abbreviations and symbols (if necessary);
− Introduction;
− Main text sections;
− Conclusions;
− List of references;
− Appendices.

The explanatory note should have a total volume of 65–75 pages (excluding appendices). The volume of appendices is not regulated.

The graphical part of the QW includes drawings, diagrams, algorithms, models, etc., necessary for the student during the defence of the QW. The content and volume of the graphical material are agreed upon with the supervisor for each student individually in accordance with the QW topic and should correspond to the content of the work.

The graphical part is presented in the form of presentation slides and is presented using multimedia tools. With the agreement of the QW supervisor, the graphical part can also contain posters of size A2 or A3.

The content of the illustrative material should sufficiently reflect the main provisions of the QW that are presented for defence.

4.2 Requirements for structural elements of the explanatory note

The title page is the first page of the explanatory note. It contains the name of the university, faculty, and department, the title of the topic, information about the executor and the supervisor of the work, signatures of the executor and responsible persons (head of the department, supervisor, consultants), the year of execution, etc.

Example of forming the QW code:

QWSE.200135.01.06.SW

The first six digits in the code are the individual study plan number of the student (grade book number); the following numbers are the group number and the student's number in the group list.

The form of the title page is given in Appendix A.

The task for the qualification work is the starting document for the execution of the QW. This document is compiled by the QW supervisor according to the chosen topic and is issued to the student. The task is signed by the QW supervisor and the student. The task is approved by the head of the department.

The task for the QW is printed on both sides of an A4 sheet or is designed in a standard form.

The form of the task for the QW is given in Appendix B.

In the "***Annotation***", a concise description of the main aspects of the QW is provided.

The "***Annotation***" should contain:

– The topic of the QW;

– Surname, first name, and patronymic of the QW author;

– Surname, first name, and patronymic of the QW supervisor;

– The volume of the explanatory note and the graphical part, the number of figures, tables, appendices, and sources according to the list of references;

– A list of keywords;

– A concise description of the work performed;

– The author's signature and the date of submission of the QW for defence.

Keywords are presented in capital letters in a line with a direct order of words in the nominative singular, arranged alphabetically and separated by commas.

The main text of the "***Annotation***" should characterize the purpose of the QW, methods, and means of information technologies used to achieve the goal, contain brief information about the results achieved, their practical significance, the degree of implementation, and the scope of application, as well as conclusions and suggestions for the development of the object of development.

The volume of the "***Annotation***" is up to 500 words. It should be placed on a separate A4 sheet. A sample of the "Annotation" is provided in Appendix B.

The "***Table of Contents***" includes the titles of all structural elements, numbers, and titles of sections, subsections, and items of the main text and appendices, with an indication of page numbers containing the beginning of the relevant material. An example of the "***Table of Contents***" is provided in Appendix C.

The structural element "***Abbreviations and Symbols***" is optional; it should only be used if there are more than five abbreviations in the explanatory note. It is usually called the "***List of Abbreviations***". Lists of abbreviations should be arranged in a column alphabetically. Abbreviations are given on the left, and their decryption on the right. An example of the "***List of Abbreviations***" is provided in Appendix D.

In the "***Introduction***", trends in the development and state of the subject area (SA) are highlighted; the problem to which the QW is directed and the tasks that need to be solved are outlined; an assessment of the current state of the specific SE task being solved in the work and its relevance is also given. It is also necessary to justify the need to solve the task, the field of application, and the purpose of development.

The relevance of the topic is the importance, essential significance, and correspondence of the QW topic to the current needs of the industry, the prospects for its development, and the practical tasks of the relevant field of activity. The relevance of the QW topic can be determined by the objective need to create new software, the objective need for modernization of business processes in specific SAs, the commercial attractiveness of the QW results, etc.

The "***Introduction***" also formulates ***the goal and objectives*** of the QW. The formulation of the goal should logically follow from the justification of the relevance of the QW topic and reflect the desired final result. Therefore, the goal formulation should reflect WHY the developed software is needed, for example, "*Develop software ... that allows increasing the efficiency of ...*", or "*Create an* *information technology ... that ensures a reduction in resource intensity ...*", or "*Developing a web application to support decision-making during ...*".

Achieving the set goal is carried out by detailing it through a systematized plan of targeted actions - design tasks.

It is recommended to formulate tasks as follows: "*perform an analysis ...*", "*establish the features of the subject area ...*", "*analyse ...*", "*identify ...*", "*determine dependencies ...*", "*develop (architecture, models, algorithms, ...) ...*", "*perform software implementation ...*", "*conduct approbation ...*", etc.

When defining tasks, it should be remembered that none of them can repeat the goal or be broader than it. The goal is achieved by solving tasks, so each of them should advance the research towards the set goal. In the end, the result obtained from solving all tasks should meet the set goal.

The approximate volume of the "***Introduction***" is 2-3 pages.

The structural element "**Conclusions**" is final based on the results of the QW and should contain generalized conclusions of the work performed. It describes the methods and means of information technologies used to implement the set goal; a description of the works that were performed to solve the task (works are considered in their interrelation, following the sequence of their execution and determining the results obtained at each stage of the QW execution); describes the results obtained; a general conclusion is made based on the results of the work performed.

It is also advisable to formulate what advantages the implementation of the developed software will provide to users (for example, it will allow reducing time costs, saving human and financial resources, increase management efficiency, accelerating the speed and improving the quality of customer service, etc.). It is also worth noting in which other practical areas it is advisable to use the developed software, provide the results of implementation if they are obtained (implementation acts, conference reports, publications, etc.), and also assess possible directions for continuing the work.

All materials should be presented briefly as a summary of the work performed and also correspond to the defined tasks of the QW. The results can be formulated based on the conclusions made at the end of each section of the main text (see p. 4.3, p. 17), but they should not be replaced by a mechanical summary of these conclusions.

The approximate volume of the structural element "***Conclusions***" is 2-3 pages.

The structural element "***List of Reference Sources***" should contain a list of sources used in the QW. Such sources can be books, periodicals (journals), regulatory and technical documents (standards, patents, catalogues), electronic resources, etc. In this structural element, it is prohibited to indicate Wikipedia pages, Studopedia, essay websites, and other similar resources.

The minimum number of sources in the list is 40.

In the main text, all sources should be referenced, so the list of sources is arranged in the order of references to them.

Examples of reference sources according to the DSTU 8302:2015 standard are provided in Appendix E.

In the ***Appendices***, material that illustrates or supplements the main text of the document is placed (figures, large-format tables, algorithm descriptions, programme listings, software testing protocols, and other materials that help to more fully reveal the idea and ways of implementing the QW).

Appendices can have the following status: mandatory, reference.

The reference appendix provides reference information (for example, necessary document forms).

In the mandatory appendix, a presentation of certain provisions of the document is provided to avoid overloading the main text (for example, models and algorithms, programme listings, presentation materials, etc.).

In the main text of the document, all appendices should be referenced, so the appendices are arranged in the order of references to them.

All appendices must be listed in the "***Table of Contents***" with their designations and names (it is not necessary to indicate the status of the appendix).

4.3 Requirements for the sections of the main text of the explanatory note

4.3.1 Structure and content of the main text sections

Regardless of the QW topic, the sections of the main text of the explanatory note should reflect the main processes of the software life cycle: ***analysis, design, implementation, and testing***. This part should reflect the stages of software development and contain information about the SA, architecture and structure of the software, database structure, selected tools for software implementation, features of implementation and testing, necessary conditions and features of software application, etc.

The approximate total volume of the sections of the main text of the explanatory note is 55-65 pages.

According to the main processes of the life cycle, this part of the explanatory note should include the following three sections:

1 Research of the subject area and problem statement

2 Software design

3 Software implementation and testing

The structuring of each section is carried out by subsections/items/sub items and is coordinated with the QW supervisor.

Section 1 Research of the subject area and problem statement

The structure and content of this are typical for any QW topic. In this section, it is necessary to analyse the situation in the organizational, technical, and software provision of the chosen SA, conduct an analysis of literary and Internet sources, formulate a list of practical tasks that need to be performed during the QW, i.e., make a detailed problem statement.

The structure of this section can be as follows.

1.1 Content analysis of the subject area, its structural and functional features

1.2 Analysis of the existing software and hardware of the subject area

1.3 Definition of functional and non-functional requirements for software

1.4 Conclusions. Problem statement

In the subsection "***1.1 Content analysis of the subject area, its structural and functional features***", a review and description of the SA for which software development is planned should be carried out. The study of the SA is conducted to identify problems and unresolved issues from the point of view of implementing information technologies, automating production processes, processes of processing and transmitting information, etc. Based on the results of the SA analysis, the problem that will be solved with the help of future software is described.

The SA analysis can be accompanied by its model representations.

Note. Modelling methodologies and types of models for illustrating the SA description should be chosen individually, depending on the QW topic.

End users of the software and their information needs are also identified. For this, primary documents are analysed, and a list of their requisites for storage in tables of operational information is provided; the process of transforming a set of input information into output is described, and security aspects are analysed (identification and authentication of users), etc.

In the subsection "***1.2 Analysis of the existing software and hardware of the subject area***", an analysis of the software already used in the SA on the QW topic should be provided. The purpose of such a review is to study the experience of leading software development companies and use their solutions when performing the QW. This will ensure that the software developed in the project meets the current needs of the software product market.

During the analysis for each analogue, it is necessary to disclose the purpose of the software product, the developer company, the main interface windows, advantages and disadvantages, etc. The list of disadvantages and advantages of analogues is advisable to summarize in a comparative table. Knowing these features for a certain set of similar software, it is logical to choose the closest to those that need to be implemented in the developed software.

In the subsection "***1.3 Definition of functional and non-functional requirements for software***", based on the analysis of the SA, the requirements for the developed software should be defined and described, the consideration of which should be accompanied by the construction of a series of models.

Note: Today, there are many methodologies and visual representations that allow modelling requirements for software. In each specific case, it is necessary to determine the appropriateness of using certain techniques. The requirement analysis should reflect WHAT the software should do, abstracting from the details of its implementation, i.e., HOW it should do it.

The main tool for software requirements engineering within the object-oriented paradigm is the Unified Modelling Language (UML).

First and foremost, use-case models (UC) are developed. The UC diagram reflects the execution of specific tasks by users using software. When creating such a model, it is advisable first to compile short descriptions of software users (actors) and the services (UC) they need. Examples are given in Tables 4.1 and 4.2.

Actor	Short description
M anager	Places orders in production and monitors their execution. If necessary, one can make operational corrections to order descriptions and cancel orders. Workshop Master Assigns performers for each of the shift task jobs. Monitors the execution of jobs and reports to the dispatcher.
Di spatcher	Plans the loading of production resources in accordance with the queue of orders coming from managers. Sends plans (shift tasks) to the workshop. Analyses the execution of plans. Makes operational changes to plans.
M aster workshop	Assign performers for each of the shift task jobs. Monitors the execution of jobs and reports to the dispatcher.

Table 4.2 – Example description of use cases

Actor	The name of the UC	Description of UC
Manager	Order Registration	This UC allows the manager to send new orders to production.
Dispatcher	Planning a new order	Dispatcher places an order in the " tail " of the queue The dispatcher places the order in the plan at the "end" of the queue.
Workshop Master	Assignment of Performers	The workshop master assigns tasks from the shift task to the performers.

Each UC should be accompanied by its specification (description), which should reflect precondition (initial state); actions that trigger the UC; UC execution scenarios (main, alternative, and prohibited); UC completion conditions; the result obtained after the UC completion; post condition (final state).

To formalize the presentation of the UC scenario, UML activity or state diagrams can be used. The main, alternative, and prohibited paths of UC execution, interactions of actors with the system, and the information they exchange should be described.

Thus, as a result of modelling, a hierarchy of diagrams can be created that reflects various aspects of the developed software (structural, functional, behavioural, informational, etc.).

An example of a UC diagram is shown in Figure 4.1.

When analysing user interface requirements, it is necessary to determine the requirements for its appearance, the form of interaction with users, the requirements for access to the internal functionality of the software, etc.

In this subsection, it is recommended to provide only the main requirements necessary for understanding the problem statement. A detailed presentation of the software requirements can be formalized in the form of a document "***Technical Task***" (TT) and placed in the appendices.

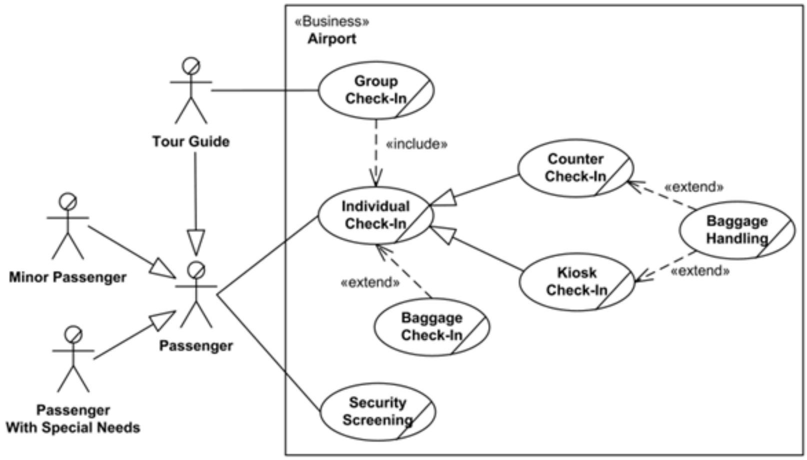


Figure 4.1 - Example of a UML use case diagram

The section should conclude with the subsection "1***.4 Conclusions. Problem Statement***" with brief conclusions and the formulation of design tasks.

The approximate volume of section 1 is 15-18 pages.

Section 2 Software Design

After analysing the software requirements and defining the requirements for the software, it is necessary to explore possible solutions to the set tasks. In other words, if at the analysis stage, the answer to the question "*WHAT should the software do?*" is required, then at the design stage, the answer to the question "*HOW to implement it?"* should be obtained.

During the project development, design decisions are justified that allow implementation of the technical task requirements, ensuring compatibility and interaction of various software components, etc. The models of the real system obtained at the analysis stage are expanded and adjusted in this section so that they can be implemented.

Software design, in general, is carried out in stages: ***preliminary design*** (architectural design) and ***detailed design*** (technical project).

The main tasks of software architecture development are:

- identification of software subsystems and mapping external software functions to them;
- defining ways of interaction between subsystems.

Typically, in general, four architectural software models are developed:

- a static structural model, which presents subsystems/components/modules that will be developed independently;
- a dynamic process model, which presents the organization of processes during software operation;
- an interface model that defines the services provided by each subsystem/component through a common interface;
- relationship models, which show the relationships between parts of the system (e.g., data flow between subsystems).

Software architecture development can be based on a specific architectural pattern, such as MVC (*Model-View-Controller*), client-server, etc.

Depending on the type of architecture, the project can be divided into levels, which will later require additional descriptions during decomposition and design (for example, components of an information system by functions can be divided into three levels: presentation level, business logic level, and data access level).

Software architecture design concludes with the creation of a description in which design decisions, logical and physical system structure, and ways of object interaction are fixed.

Note: The architectural design stage is not strictly mandatory. If the main design decisions are obvious, this stage can be excluded from the general sequence of works.

After designing the system structure and defining the principles of structure management, the subsystem decomposition into modules can be immediately performed. This work can be considered as an introduction to detailed design, which specifies architectural decisions. The task of decomposition is to determine the internal content of each subsystem (component). The result is the formation of the subsystem structure - a set of modules and their interaction relationships.

Considering the decisions made at this stage, further specification of functional specifications is carried out: programme structure, object model (class diagram), logical and physical database models, key methods and algorithms, user interface design, etc. For this, commonly accepted visualization tools should be used: schemes, UML diagrams, algorithm recording tools (pseudocode), etc.

The static model is represented by class diagrams and class specifications. The dynamic model represents cooperation, which are implementations of use cases by corresponding interaction diagrams. Logical data structure models are presented by class diagrams or "entity relationship". Logical modules are sub programmes (procedures, functions).

When developing the software structure, it is necessary to:

- identify the main modules and information resources (and also justify this choice) that the software product should consist of;
- describe the functional purpose of the main modules and information resources, their interconnection, as well as data exchange.

Modules serve as physical "containers" in which logical development classes and objects are declared. The class is the main "building" block of detailed design, and UML class diagrams are the main means of representing software structure.

It is recommended to build a tree of modules that reflect the structural scheme of the package: control modules (classes), modules performing auxiliary functions, working modules and other modules.

Software structure development can be based on a specific design pattern.

Note: In some languages (e.g., C++), a module is an independent language construct that facilitates the creation of separate design solutions. In some languages, there are no modules, and the only physical unit is a class.

In the user interface design, the following are defined:

- Interface elements necessary for executing use cases;
- The relationship between interface elements;
- The application of interface elements in various use cases, etc.

The user interface description should be accompanied by diagrams of screen forms, menus, dialogues, etc.

Also, in this section, an analysis of technologies and methods for software implementation is carried out, indicating their advantages and disadvantages in the context of their use for project implementation: platforms, frameworks, graphic "engines", database management systems (DBMS), programming languages, mark-up languages, database query languages, existing software development tools, etc. As a result of this analysis, the optimal option is determined.

At the end of the section, brief conclusions should be formulated.

The approximate volume of section 2 is 20-22 pages.

Section 3 Software Implementation and Testing

Software implementation involves creating a functional and viable software product based on the developed design.

In this section, issues directly related to software construction are discussed, namely:

- Description of software modules and their purpose, as well as the specifics of data transfer between them;
- Description of the implementation of programme modules;
- Description of the process of creating a database (if present in the project) with necessary illustrations;
- Description of the implementation of the user interface;
- User instructions (with relevant illustrations), which provide information on the principles and conditions for using the software (description of the user interface; sequence of user actions when working with the programme; conditions necessary for using the software, etc.);
- Technical specifications of the developed software (hardware and operating system (OS) requirements; required amount of RAM and external memory; special devices; additional programmes required for the software to function, etc.);
- Software testing.

Note: User instructions can be included in the appendices.

When presenting material to explain the specifics of model and algorithm implementation, module descriptions, classes, methods, SQL queries, etc., it is recommended to use fragments of programme code.

The full code of the programme is usually provided in the appendices.

In the software testing description, the testing strategy should be defined, and the chosen methods and testing methodology justified (methods of "black" and "white" box testing, functional and non-functional testing, performance testing, load testing, data level testing, usability testing, etc.), requirements for conducting experiments are formulated, the scope of each experiment according to functional specifications and constraints, etc. The ultimate goal is to compare expected and actual test results.

If necessary, testing levels are highlighted - module, integration, and system testing.

The tools used for ***automated testing*** (if any) should also be described.

Test cases should be realistic and sufficient to check the functionality and correctness of the software. Accordingly, for each software element, a list of all situations that need to be checked for both "correct" and "incorrect" input data should be compiled. The data used for software testing should be described in detail, indicating the source of their acquisition. Data from primary documents, regulatory reference documentation, as well as data generated automatically can be used.

In the expected results, the correct operation of the programme is always described. At the same time, the correct operation of the programme can fully anticipate messages about incorrect user actions or certain critical situations. For example, the message "*Unable to save the file at the specified path: not enough free space on the target medium*" is not a programme error but its correct operation. A programme error (in the same situation) would be the absence of such a message.

Usually, test cases are grouped into test sets. Use case scenarios can be considered a subtype of test case sets.

Note: It is not necessary to describe all test cases; it is enough to provide a few main examples.

In the section, the testing order should also be indicated, testing procedures with the expected results when the software works correctly should be provided, and a description of the main results obtained during software testing.

If the programme is supposed to work on different hardware or in different operating environments (on different platforms), it must be checked on all these platforms.

After executing the test cases, a test results report is created, which contains information on each executed test case and its outcome - success or failure.

For greater clarity, the testing summary data should be presented in tables and charts. An example of a testing scenario is shown in Table 4.3.

Table 4.3 - Example of a testing scenario (fragment)

Action	Programme Reaction	Result
1. Click on the "System" button and select the "System Management Applet" menu item	A login and password input window will appear	Correct
2. In the input window, enter the username "guest1" and password "guest". Then click the "Login" button	The "System Management Applet" window should close	*Incorrect.* The window is titled "System Management Application"
3. End the applet session by clicking the "Logout" button	The "System Management Applet" window should close	Correct

Note. Based on the testing results, a "Testing Act" document can be formed and included in the appendices.

At the end of the section, brief conclusions are presented regarding the results of the software implementation, as well as the degree of software operability and its compliance (non-compliance, partial compliance) with the technical specifications, for example:

"*After the conducted experiments ... it was found that ...";*

"*The basic functionality is implemented in accordance with the software requirements and is fully/partially operational*";

"*There are non-critical issues related to ...".*

The estimated volume of section 3 is 20-22 pages.

4.3.2 Elaboration of the main text sections for typical qualification work topics

The structure and content of the main sections will be considered using examples of typical qualification work topics.

Example 1. Software for automating business processes, management, company or organization activities, etc.

Any organization is a set of interacting elements (departments, processes, etc.), each of which can have its own structure. These elements are functionally connected, meaning they perform separate types of work within a single business process, exchanging information. Moreover, these elements interact with external systems. This situation is true for almost all organizations, regardless of their activities. Such a general view allows for the formulation of the main principles of building software systems revealed during the analysis of the subject area.

The structure and content of the explanatory note's main text sections can be as follows:

1 Research of the subject area and problem statement

This section characterizes the mission of the automation object (enterprise, process, etc.), presents the organizational structure of the object, characterizes business processes, existing software solutions, etc. Based on the analysis, the problem to be solved with the developed software is described, and tasks that need to be automated are identified. When analysing software requirements, it is necessary to provide a characterization of functional (business requirements, user requirements, system requirements, etc.) and non-functional requirements (business rules, quality attributes, external interface, constraints, etc.).

2 Software Design

This section addresses issues related to the selection and development of software architecture, project decomposition, description of object model interfaces, user interfaces, etc. This section is recommended to be structured as follows:

2.1 *Choice of architecture type and design patterns*

Here, the choice of architecture type and design patterns is described in detail. For this, several architectures are highlighted and analysed in detail; based on the analysis, the architecture that best meets the requirements is chosen. The choice is made based on comparisons and exclusions. Depending on the type of architecture and the complexity of the software, the project can be divided into appropriate levels.

2.2 Decomposition description

Here, the project's decomposition is carried out, which, depending on the chosen architecture type, is reflected in the description of general decompositions, module decomposition, decomposition into parallel processes, data decomposition, state transition model decomposition, I/O decomposition, etc. If the application is divided into several levels, the decomposition description should be done for each level separately.

2.3 Dependency description

Dependencies between the decomposition variants presented in item 2.2 are described. Such a description can consist of the following blocks: inter-module dependency block, inter-process dependency block, data dependency block, state dependency block, level dependency block, etc. If there are levels, inter-level dependencies should be described separately.

2.4 Interface description

Here, considering the developed software structure, the interfaces of the object model are described, where classes of one level can be described along with classes of another level (if available). This subsection can be divided into a description of inter-module interfaces, where the interfaces of existing packages are described, and a description of process interfaces (if all classes in all packages are open, the interfaces will consist of all classes and methods).

2.5 Detailed module design The stages of detailed module design are presented in the subsection, indicating their names and the structural elements included in them (each structural element should be presented and described). The description of structural elements should be based on class diagrams, sequence diagrams, state transition diagrams, and I/O. In addition, the detailed architectural decisions should include flowcharts and pseudocode, as well as a description of the user interface.

2.6 Detailed data design A description of the data structures that are not used when describing classes in the previous subsections is provided. Data flow diagrams, data table descriptions and their relationships, and infological and physical database models are presented. If there are no separate data structures in the chosen architectural solution, this subsection provides information about the data used in the modules and their processing.

2.7 Analysis and selection of application implementation technologies and methods

In this subsection, an overview, analysis, and justification of the choice of technologies and tools for software development are carried out.

2.8 Conclusions

In this subsection, brief conclusions are presented regarding the obtained software design results.

3 Software Implementation and Testing

This section addresses issues related to activities following the choice of architecture. The main goal of the section is to describe and implement the designed application. The developed architecture should be detailed, and its implementation is associated with software realization. Software realization involves creating a functional and operationally suitable software tool based on a detailed design. This section may include the following subsections:

3.1 Software Module Implementation

This subsection describes the implementation of all software modules. When presenting material to explain the peculiarities of implementing models, algorithms, modules, classes, methods, interfaces, etc., it is recommended to use fragments of programme code.

3.2 Database Development

Here, the process of creating a database (with necessary illustrations) and its connection to software modules should be described. This subsection should also contain descriptions of stored procedures, functions, and database triggers (if they were developed) with the source code provided in the appendices. If the created database model is intended to be shared by multiple applications or users, a modern DBMS supporting "client-server" network processing and ensuring efficient user access to database data should be used.

3.3 Technical and Software Requirements

This section specifies the technical characteristics of the developed software and the main requirements for hardware and software for its full and uninterrupted operation. The minimum composition and a brief description of the system software, additional software, drivers, special devices, etc., are provided.

3.4 Software Testing

In this subsection, software verification and validation are carried out. The subsection should contain a justification for the choice of testing methods, the development of test data sets, and an analysis of testing results. It is also advisable to include recommendations or suggestions for improving identified problem aspects. This subsection may contain the following points:

3.4.1 Selection and Justification of Application Testing Methods

In this point, an overview and analysis of testing methods and tools best suited for the developed software are conducted.

3.4.2 Software Validation and Verification

In this point, tests are presented through which the verification of application components and the correctness of the software's operation as a whole are implemented. Verification is carried out by checking the functions implemented in the finished software against the requirements specified in the technical specifications.

3.4.3 Analysis of Testing Results

In this point, summarized information related to the analysis of software testing results is provided. This information can be presented in the form of tables, charts, diagrams, flowcharts, etc.

3.5 Conclusions

In this subsection, brief summarizing conclusions are presented regarding the obtained results of software development and testing.

Example 2. Reference Information System

A Reference Information System (RIS) is an information repository that includes means for input, storage, protection, search, and presentation of messages. RISs solve tasks to provide the user with the most accurate (relevant) information on the topic of interest. They have several advantages and capabilities, including:

- the ability to compactly store significant amounts of information;
- the ability to structurally display stored information;
- the ability to quickly search for the required data, etc.

The RIS interface with the outside world, namely the information flows between the system and the external entities with which it should be connected, consists of information flows between users and the RIS itself, which processes and stores information in the database. The structure and content of the explanatory note's main text sections can be as follows:

1 Research of the subject area and problem statement

In this section, along with the general requirements for it, significant attention should be paid to the analysis and definition of the types and content of the information support of the developed RIS. A special role in this is given to the classification and coding of information, which ensures mutual information exchange between the user and the RIS. A description of the system users (for example, administrator, registered user, unregistered user, etc.) is also provided. To illustrate information flows, data flow diagrams can be used.

2 Design of the reference information system

The basis of any RIS is a database - a set of ordered information used in the operation of the RIS and aims to ensure mutual information exchange between the structural units of the RIS. In view of this, the RIS is designed. Special attention is also paid to the logic of the application's operation, the organization of interaction between the programme and the database, and between users and the RIS. It is recommended to structure this section as follows:

2.1 Designing System Architecture and Structure

This subsection describes the chosen RIS architecture and a detailed design of the modules, indicating their names and the structural elements they include. The description of structural elements should be based on class diagrams, sequence diagrams, state transition diagrams, and I/O diagrams. Additionally, it is appropriate to include flowcharts of algorithms and pseudocode in detailing architectural decisions. The interconnection scheme of modules (classes) and information resources should reflect the interconnection of software and information provision of the task complex and can be represented by several schemes, each corresponding to a certain mode. The functional structure of the RIS should be oriented towards the information needs of end-users, which change in market conditions, and reflect the content and specifics of the necessary functions.

2.2 Designing the Logical Model of the Database

To create a logical model of the database, it is primarily necessary to unify and standardize its components. At this stage, it is necessary to:

- Identify the list of entities, information about which will be stored in the database, and provide their description;
- Define the list of entity attributes and provide their description;
- Define the relationships between the database entities and provide their description;
- Define the types of relationships and necessary constraints;
- Define primary keys for each database object;
- Perform data normalization (up to the third normal form).

The logical database model should be built based on class diagrams (entity relationship diagrams can also be used). The class diagram should display entities, attributes, and associations necessary to describe the database structure.

2.3 Designing the User Interface

In this subsection, the design of the RIS content management system structure is described, as well as the interface operation scheme (including the logical organization of information on resource pages) for users with different access rights. If, according to the technical specifications, the RIS is a web-oriented system, then the logical structure of the website is developed, a description of the navigation system is provided, prototypes of the necessary web pages are created, preliminary design, etc. This information can be presented in the form of drawings, diagrams, etc.

2.4 Analysis and Selection of System Implementation Technologies and Methods

This subsection involves the analysis and justification of the choice of technologies and tools for RIS development. A brief analysis of programming languages is also carried out, and a DBMS is chosen with the necessary justification. If the created database model is intended to be shared by multiple applications or users, a modern DBMS supporting "client-server" network processing and ensuring efficient user access to database data should be used (e.g., Microsoft SQL Server, MySQL, etc.).

2.5 Conclusions

In this subsection, brief summarizing conclusions are presented regarding the obtained results of RIS design.

3 Software Implementation, Debugging, and System Testing

This section provides a detailed description of the RIS software implementation, i.e., a description of the system module's programme code and its components' structure. The stages of RIS implementation include describing the creation of the database, describing the composition, structure, content, and functions of the developed software, and mutual functioning (including based on the web interface). When presenting material to explain the peculiarities of implementing modules, classes, methods, algorithms, etc., it is recommended to use fragments of programme code. During software implementation, user error messages should be anticipated. This section may include the following subsections:

3.1 Database Implementation

This subsection addresses issues related to creating a database using the chosen DBMS (including version management, database updates, etc.). First, based on the constructed logical database model, this model is described in terms of the chosen DBMS, i.e., a physical database model is constructed. Then, explanations are provided for creating and populating database tables, descriptions of designed triggers, procedures, etc. When describing these elements, it is indicated which elements are implemented using the DBMS tools and which ones – using the chosen programming language.

3.2 System Module Implementation

In this subsection, a description of the software implementation of all RIS components is carried out, as well as the interface for users with different access rights. When presenting material to explain the peculiarities of implementing modules, classes, methods, algorithms, etc., it is recommended to use fragments of programme code. The subsection also describes the technology of interaction between the application, DBMS, and database.

3.3 User Manual

Here, issues are presented that contain a description of the conditions and principles of using the RIS. The RIS operation description should contain the following data: general information about the developed system; its functional purpose; input and output data; copies of interface windows, programme operation results, etc., as well as a sequence of user actions that ensure the loading, execution, and termination of the programme. In general, information should be provided that is sufficient to understand the main functions of the RIS and its operation.

3.4 Hardware and Software Requirements

This subsection indicates the minimum necessary for the normal operation of the RIS composition and characteristics of hardware, system software requirements, information about additional devices, etc. (these characteristics are described separately for the server and client parts of the application).

3.5 Debugging and System Testing

The process of debugging and testing the RIS is described. Achieved results are demonstrated by showing possible states of the programme and entering correct and incorrect data with the display of corresponding messages, which were anticipated during RIS development. Results are presented in the form of screen copies with explanations. This subsection may contain the following points:

3.5.1 Selection and Justification of System Testing Methods

In this subsection, an overview and analysis of testing methods and tools suitable for the developed system are conducted.

3.5.2 Verification and Validation of the System

To evaluate the practical aspects of the developed RIS, verification and validation are applied. During verification and validation, the programme is checked for the correct implementation of algorithms, the correct operation of the RIS for all sets of input data (checking data input formats, checking data types, checking data belonging to certain ranges, etc.), and searching and eliminating defects, as well as checking the implemented functions for compliance with the technical specifications.

3.5.3 Analysis of System Testing Results

This subsection provides a summary of the analysis of the RIS testing results. This information can be presented in the form of tables, graphs, diagrams, flowcharts, etc.

3.6 Deployment and Installation of the System

This subsection describes the process of setting up the server and client parts of the RIS. The server part is described separately, where (if necessary) the OS installation, DBMS installation and configuration, and component tuning (both in the OS and the DBMS) are provided. The process of transferring the database is described. For the client part, it is indicated how it is installed, configured, and accesses the server part, etc.

3.7 Conclusions

In this subsection, brief summarizing conclusions are presented regarding the obtained results of RIS development and testing.

Example 3. Web Application

A web application is a client-server application in which the client interacts with the web server using a browser. The logic of the web application is divided between the server and the client; data storage is mainly on the server; information exchange takes place over the network. The browser can be an implementation of "thin clients" - the application logic is focused on the server, and the browser's function is mainly to display information downloaded from the server over the network and send back user data. One of the advantages of this approach is that clients are not dependent on a specific user OS.

The structure and content of the main text sections of the explanatory note can be as follows:

1 Research of the Subject Area and Problem Statement

In this section, the subject area is described (purpose of the web application, roles, possible operations, etc.), and analysis of existing similar developments according to modern needs (especially business, economics, etc.). Also, the features and limitations of the developed application are described, and the justification for the development is provided.

2 Web Application Design

In this section, an analysis and selection of the architecture of the developed application, database structure design, server and/or client part design, and user interface design are conducted. Also, a description, schematic layouts (or .psd layouts) of the web application structure, a description of the navigation system, prototypes of the necessary web pages, and preliminary design are provided. This section is recommended to be structured as follows:

2.1 Analysis and Selection of Web Application Architecture

In this subsection, an analytical description and justification for the choice of the application's architecture are carried out. Since it's an online application that implies interaction, typically, a client-server architecture is chosen.

2.2 Description of Data Structure and Database Model

In this subsection, the database schema is reviewed, and all tables are presented with field types and explanations, as well as relationships between tables. The specified information should be presented in the form of a normalized database schema. The modelled database is visualized using an "entity relationship" diagram or class diagrams.

2.3 Designing the Server Part of the Web Application

In this subsection, the server is set up for the web application implementation, in particular, the methods of creating forms for sending data to the server and implementing scripts that process the received data (since the main task of the server part of the application is to receive and fully process data coming from the client).

2.4 Designing the Client Part of the Web Application

In this subsection, the design of the client part of the application is described, indicating the content management system structure and the interface operation scheme (including the logical organization of information on resource pages). The specified information can be presented in the form of drawings, diagrams, flowcharts, etc.

2.5 Creating a Web Application Layout and Design

In this subsection, the structure of the web application is designed, indicating the organization of data and the hierarchy of materials necessary for their presentation to the end user. The process of creating an application structure can be divided into two stages: 1) structuring information and 2) visual representation of the structure. Structuring information involves the need to classify and group various materials, combine them into categories or headings, and assign them names understandable to users. Attention is also paid to colour, graphic, and stylistic solutions.

2.6 Analysis and Selection of Web Application Implementation Technologies and Methods

In this subsection, an overview, analysis, and justification for the choice of technologies and tools for web application development are carried out: HTML mark-up language; DHTML components; cascading style sheets (CSS); document object model (DOM); methods of creating JavaScript scripts as part of web pages; managing page elements based on DOM; organizing user interaction based on events; jQuery library tools for organizing JavaScript and HTML interaction; PHP programming language and relevant frameworks; methods of interaction of web scripts and DBMS, query language to MySQL databases; an approach to building interactive user interfaces using Ajax technology; protocols and data exchange formats on the Internet and their processing methods, etc.

2.7 Conclusions

In this subsection, brief summarizing conclusions are presented regarding the obtained results of web application design.

3 Software Implementation and Testing of the Web Application

This section discusses issues directly related to the activity, the purpose of which is to prepare the project for implementation and its realization. It is appropriate to accompany the description of the web application development process and its components with fragments of the programme code and relevant illustrations ("screenshots"). It is also necessary to provide guides for both the administrator and the user. This section may include the following subsections.

3.1 Database Development

This subsection includes the creation of a database, connecting it to other application modules, version management, database updates, and other settings.

3.2 Development of Software Modules

This subsection describes the implementation of all planned application modules, including authorization modules, registration, payment systems (for instance, when developing an online store), etc.

3.3 User Guide

This subsection provides a step-by-step description of using the web application for different user categories (administrator, registered user, guest, etc.).

3.4 Technical Specifications of the Web Application

This subsection specifies the technical characteristics of the developed application (both its server and client parts) and the main requirements for its correct and uninterrupted operation.

3.5 Uploading the Web Application to Hosting

This subsection describes the placement of the web application on hosting. For the qualification work, it is sufficient to register on a free hosting (for example, https://000webhost.com/), place the web application files on it, and briefly describe the placement process.

3.6 Web Application Testing

This subsection describes the verification and validation of the web application. The subsection should justify the choice of testing methods, develop test data sets, and analyse the test results. It is advisable to include recommendations or suggestions for improving the identified problem aspects of the development subject. The subsection may contain the following points.

3.6.1 Selection and Justification of Web Application Testing Methods

This point reviews and analyses the testing methods and tools that are best suited for the application being developed.

3.6.2 Error Checking Using Unit Tests

This point presents tests that verify the application components. Examples of launched unit tests can be provided, showing whether the expected result matches the obtained one.

3.6.3 Validation and Verification of the Web Application

This point describes the validation of individual modules and the web application as a whole. Validation can be done using validators (for example, https://validator.w3.org/). Verification is done by checking the functions implemented in the application against the requirements specified in the technical specifications (or task statement).

3.6.4 Analysis of Web Application Testing Results

This provides a summary of the analysis of the web application testing results. It can be presented in the form of tables, graphs, diagrams, flowcharts, etc.

3.7 Conclusions

This subsection provides brief summary conclusions regarding the results obtained from the development and testing of the web application.

Example 4. Mobile Application

Developing software for mobile devices requires considering their limitations and capabilities. Mobile devices operate on batteries and have less powerful processors than PCs, and they also have additional features (e.g., geolocation and camera presence). Developers also have to consider screen sizes, various technical specifications, and equipment configurations due to the fierce competition of mobile software and changes in each platform. The structuring and content of the main text sections of the explanatory note can be as follows.

1 Research of the Subject Area and Task Setting

This section analyses the subject area, highlighting its boundaries. It also provides a review of existing mobile applications (close to the qualification work topic) developed in the last two years and justifies the feasibility of development.

The result of the analysis is the compilation of technical specifications, as well as a description of the features and limitations of the application.

2 Designing the Mobile Application

In this section, the application architecture is chosen, and maps or models are created and presented, visually demonstrating all the application functions. Prototypes are also created, reflecting all application screens and transition schemes between them. The presentation format can be arbitrary. Prototypes can be static or interactive. It is also necessary to design all application screens, depicting different states for all usage scenarios. This section is recommended to be structured as follows.

2.1 Architecture and Functional Structure of the Application

This subsection analyses and selects the architecture of the developed mobile application. The advantages and disadvantages of the chosen architectural solution are indicated, and its functional structure is schematically presented.

2.3 User Interface Design

In this subsection, the design of the user interface is discussed. The design of the interface is considered, taking into account the peculiarities of the dialogue structure that will take place between the user and the mobile device. This stage defines the high-level structure of screen layouts, as well as the flow, behaviour, and organization of the product. All elements of the graphical interface undergo usability testing to ensure that the design decisions made are ergonomic and allow users to solve their tasks effectively. The designed interface is presented in the form of structural diagrams, layouts, etc.

2.4 Development of the Mobile Application Algorithm

This subsection provides instructions that describe the sequence of executor actions to achieve the result of solving the task in a defined number of steps. When developing an algorithm, it is necessary to determine the complete set of initial data of the task (the initial state of the object), the purpose of creating the algorithm (the final state of the object) and the system of executor commands (a set of commands that the executor understands and can execute). The developed algorithm can be presented in the form of a flowchart, model, etc.

2.5 Creation of the Mobile Application Prototype

In this subsection, prototyping of the main parts of the application is carried out using modern tools. Based on the ready-made prototypes and documented decisions, a scheme for the mobile application's operation is built.

2.6 Analysis and Selection of Technologies and Methods for Application Implementation

This subsection analyses modern tools and technologies for performing the set task. The necessary frameworks, programming languages, etc., are chosen depending on the operating system and according to the needs, relevance, and feasibility of modern conditions.

2.7 Conclusions

In this subsection, brief summary conclusions are provided regarding the results obtained from designing the mobile application.

3 Software Implementation and Testing of the Mobile Application

This section describes the implementation and testing of the mobile application and its components (modules), accompanied, if necessary, by fragments of the programme code and "screenshots" of interface windows. Depending on the specified requirements, the implementation of the code part is carried out depending on the operating system (Android or iOS). It also describes the technical specifications of the developed application and the main requirements for its correct and uninterrupted operation. This section may include the following subsections.

3.1 Implementation of the Mobile Application Logic

This subsection describes the application's activities, defining the actions performed by the user. The process of the Activity class is also described and its creation, launch, and transition to other activities.

3.2 Implementation of the Mobile Application Layout

In this subsection, the appropriate container (layout) of the mobile application is chosen, and all its components are described (for example, the layout for an Android programme is built using a hierarchy of View and ViewGroup objects). The code part of the layout implementation can be provided according to the chosen container.

3.3 Database Development

If there is a database, this subsection should implement its full processing, including creation, version management, database updates, etc. In the application, when connecting to the database, it is necessary to indicate the means of supporting the database and its version. For example, Android has built-in database support - SQLite (all SQLite functions are supported, and an API wrapper with a compatible interface is provided). Android SQLite API is typical.

3.4 User Guide

This subsection examines detailed reference information for the user, as well as the features of the application's operation.

3.5 Technical Specifications of the Mobile Application This subsection should contain information regarding the technical specifications of the device for the guaranteed successful execution of the developed mobile application (version, OS, API, memory size, etc.).

3.6 Testing the Mobile Application

The mobile application is checked and tested for compliance with the technical specifications. The application is installed on devices and works as if it were in Google Play or AppStore. This section may contain the following points.

3.6.1 Analysis of Mobile Application Testing Methods

In this point, an analysis, selection, and justification of methods and tools for testing that are appropriate for mobile applications are carried out.

3.6.2 Testing the Mobile Application Using an Emulator

At this point, a description of the results of testing the mobile application using automated tools is carried out. Mobile application testing takes place in the development environment using an emulator. After that, the application is tested on the device. Emulators are an easy way to test an application on a mobile phone without physically using it.

3.6.3 Analysis of Mobile Application Testing Results

At this point, analytical information regarding the results of the conducted application testing is provided. The results can be presented in the form of diagrams, graphs, tables, etc.

3.7 Conclusions

In this subsection, brief summary conclusions are provided regarding the results obtained from the development and testing of the mobile application.

Note: In the case of the implementation of an atypical or combined qualification work topic by the student, the structure of the main text sections is agreed upon with the work supervisor (some subsections can be removed or new ones added). Subsections can also be divided into points/sub points if this is consistent with the general logic of the qualification work.

5 REQUIREMENTS FOR FORMATTING QUALIFICATION WORK

5.1 General requirements for explanatory note formatting

The general requirements for the explanatory note include a logical sequence of material presentation, clarity and specificity of design results, the essence of the task setting and the purpose of the qualification work, research methods, decisions made, substantiated conclusions, etc. The text should not be overloaded with low-information content or descriptions of well-known facts. The note's text should be clear, concise, and well-edited.

When stating mandatory requirements in the text, words and phrases such as *"must," "implies," "necessary," "needs to," "only allowed," "not allowed,"* etc., should be used. When stating other provisions, words like "*can be," "usually," "if necessary,*" etc., should be used. It is also permissible to use a narrative form of text, using words like *"apply," "consider" and "recommend*."

The explanatory note should not be written in the first person, e.g., *"I determined ...", "I believe ...", "It seems to me ...", "In my opinion ...*" etc. Instead, the text should be written in an impersonal form throughout. The text should adhere to the norms of the current spelling and vocabulary, using a business language style suitable for official documents.

The text should use terms, notations, and definitions established by current standards or, if absent, those generally accepted in scientific and technical literature.

In the document text, instead of numbers, their verbal value should be written (for example, "*four features*," but "*3 MHz*"); numbers 10, 11, ... are written in digits.

In the document text, except for formulas, tables, and figures, it is not allowed to use:

- the symbol "∅" as a designation of diameter (one should write the word "diameter"); when indicating the size or tolerance of the diameter in the drawings, the sign "∅" should be placed before its numerical value;
- without numerical values, mathematical signs, for example: ">" (more), "<" (less), "=" (equals), "≥" (greater than or equal to), "≤" (less than or equal to), "≠" (not equal to), as well as "№" (number), "%" (per cent) and "°C" (Celsius degree);
- the mathematical minus sign (–) before the negative value of the quantity (one should write the word "minus").

When indicating permissible deviations of specified norms and requirements, the phrases "should not be more than," "less than" and "should not exceed" should be used.

When indicating the maximum or minimum value of a quantity, the phrases "should be no more than" and "no less than" should be used.

In the document text, it is allowed to use:

- commonly accepted abbreviations: see. – see; min. – minimum; max. – maximum, etc.;
- abbreviations: abs. – absolute; p. – page; y. – year; UAH – hryvnia and other abbreviations used with numerical values.

Instead of abbreviations "etc." (and so on), "and others" (and such other), "and the like" (and the like), it is recommended to use "etc." If a special system of abbreviations of words or names is adopted in the text, a list of accepted abbreviations should be provided after the content in the "List of Abbreviations" structural element.

5.2 Requirements for text presentation

The document text, depending on its understanding by content, is divided into sections, subsections, items, and sub-items, which are numbered with Arabic numerals: sections – within the entire document, subsections – within each section, items – within each subsection, sub-items – within each item.

Each section should start on a new page.

The section number is recorded without a dot at the end. The subsection number should consist of the section number, a delimiter dot, and the subsection number; a dot is not placed at the end of the number. For example, 2.1 – the first subsection of the second section.

Items are numbered with Arabic numerals within each subsection. The item number should consist of the numbers of the section, subsection, and item, separated by dots (but without a dot at the end). For example, 2.1.3 – the third item of the first subsection of the second section.

The structural elements ***"Abstract," "Content," "List of Abbreviations," "Introduction," "Conclusions" and "List of References"*** are not numbered.

The headings of structural elements and sections should be printed with an indent in **CAPITAL LETTERS** in bold without a dot at the end. They can be placed in the middle of the line (in this case – without an indent).

Subsection, item, and sub-item headings should be printed with an indent starting with a capital letter without a dot at the end. Words cannot be broken by a hyphen in any headings.

The distance between the heading and the subsequent or previous text should be at least a double line spacing. The distance between the lines of the heading and between two headings should be the same as in the text.

The title of a subsection/item/sub-item should not be placed at the bottom of the page if less than two lines of text are placed after the title. If this happens, it is allowed (within individual pages) to change the line spacing, but not more than 0.02 (recommended multiplier values from 1.48 to 1.52).

5.3 Requirements for the Main Text Formatting

The requirements for formatting the explanatory note's text are regulated by the state standards of Ukraine and the university's regulatory documents.

The explanatory note of the qualification work should be formatted on A4 sheets (210 mm x 297 mm) using form frames; the main inscription is made according to the requirements of DSTU GOST 2.104:2006 (form 2 for the first page of "***Contents***", form 2a for subsequent sheets). If necessary, A3 sheets (297 mm x 420 mm) can be used.

The title page, Assignment for the qualification work, Abstract, and Appendices are formatted on regular A4 sheets (without form frames).

On forms, the distance from the form frame to the text boundaries at the beginning and end of lines should be no less than 3 mm (5 mm recommended).

The distance from the top or bottom line of the text to the top or bottom frame should be no less than 10 mm (10 mm recommended).

The distances of the form frame to the edges of the sheet should be: from the left edge – no less than 20 mm, from the right, top, and bottom – 5 mm.

For the title page, abstract, and appendices, the recommended page margins are the following: top and bottom – 20 mm, left – no less than 20 mm, right – 10 mm; for the assignment for the qualification work – all margins are 20 mm.

It is also recommended to adhere to the following requirements: main font – Times New Roman; font style – regular (except for the titles of structural elements and section headings); font-size – 14 pt.; font colour – black; line spacing – 1.5; text alignment – justified; paragraph indent – 1.25 cm.

It is not allowed to place only one word in the last line of a paragraph. If this happens, the paragraph text should be reformulated accordingly or use a denser character spacing (but no more than 0.2 pt.).

5.4 Page numbering of the explanatory note

The pages of the explanatory note should be numbered with Arabic numerals (without a dot at the end), maintaining continuous numbering throughout the entire document, including appendices. On forms, the page number is placed on the right in the *Sheet* field.

For appendices, the numbering continues, but the page number is placed in the top right header.

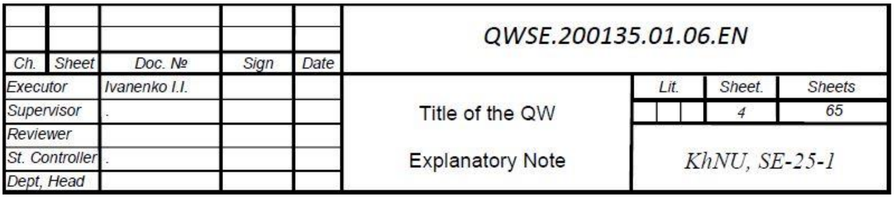
The title page is included in the general page numbering (with number 1), but the page number is not indicated on it.

The Assignment for the qualification work (with number 2) and "***Abstract***" (with number 3) are also included in the general page numbering of the document, but the page number is not indicated on them.

"***Contents***" is executed on a form frame; the main inscription is made according to the standard DSTU GOST 2.104:2006 – form 2 (with a "large" frame) for the first page of "***Contents***", form 2a (with a "small" frame) – for the next page of "***Contents***" (if any), and also for the following text pages.

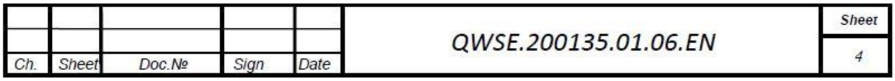
"***Contents***" is included in the total number of document pages. On the first sheet of "***Contents***", the page number (number 4 in the *Sheet* field) is indicated in the frame on the right, as well as the total number of pages of the explanatory note (in the *Sheets* field).

The sample of the main inscription according to form 2 for the first page of "***Contents***" and form 2a for the following text pages is shown in Figures 5.1 and 5.2, respectively.



					QWSE.200135.01.06.EN			
Ch.	Sheet	Doc. №	Sign	Date				
Executor		Ivanenko I.I.			Title of the QW	Lit.	Sheet.	Sheets
Supervisor		.					4	65
Reviewer					Explanatory Note	KhNU, SE-25-1		
St. Controller		.						
Dept, Head								

Figure 5.1 - Sample of Form 2 for the first page of "Contents"



					QWSE.200135.01.06.EN	Sheet
						4
Ch.	Sheet	Doc.№	Sign	Date		

Figure 5.2 - Sample of Form 2

5.5 Formatting of the Software Code and Its Fragments

Listings of software codes are typically placed in the appendices. Fragments of code that are key to solving the task can, if necessary, be presented in the main part of the explanatory note in the form of text for a better description of the software implementation. The programme text (as well as its fragment) should be easily "readable" due to proper structuring and formatting of the code, as well as the use of comments.

Code listings (or their fragments) are usually presented in a monospaced font (for example, **`Courier New`**) with left alignment without paragraph indents. It is permissible to reduce the font size (to 10 pt.) and also use a single line spacing.

A fragment of the software code is placed at a distance of one free line from the previous and subsequent text.

Example

Let's present the code of the software implementation for dynamic memory allocation for a one-dimensional array:

```
int *a, n;
printf("Enter the array size:  ");
scanf_s("%d", &n);
// Allocating memory for the array
a = (int *) malloc(n*sizeof(int));
// Checking memory allocation
if (!a)
{
   puts("Memory allocation error\n");
   _getch();
   return 1;
}
```

Note. Detailed requirements for formatting text documents (in particular, requirements for formatting lists, figures, tables, formulas, appendices, etc.) are set out in DSTU 3008:2015 and SOU 207.01:2017.

Detailed requirements for formatting bibliographic descriptions and references are provided in DSTU 8302:2015 and SOU 207.02:2017.

5.6 Requirements for Formatting the Graphic Part

Presentation materials are created in the form of slides using appropriate software tools (like Microsoft PowerPoint) and are intended for use with projection equipment during the defence of the qualification work.

Printed slides (format A4) are included in the appendices.

If the ***Task for the qualification work*** provides for a graphic part in the form of demonstration posters, it should be formatted on sheets of drafting paper of format A2 or A3 and accompanied by the main title and additional graphs according to the requirements of DSTU GOST 2.104:2006 - form 2.

The form indicates the topic of the qualification work and the title of the drawing.

Example of code formation:

QWSE.200135.01.06.E8

The graphic part is executed using modern computer tools and software packages (Case tools, Corel Draw, etc.). Sheets are numbered – the sheet number is indicated in the frame on the right in the *Sheet* field. (1, 2, ...). The total number of sheets is also indicated (in the *Sheets* field). The sheets are signed by the student, the supervisor of the qualification work, the norm controller, and the head of the department.

6 PREPARATION FOR THE DEFENCE OF THE QUALIFICATION WORK

6.1 Preparation of Documentation

For the defence, the student must submit to the department a completed and bound (in a "hard" cover) explanatory note with the signatures of responsible persons at least five days before the defence date; the sheets are bound in the order specified in paragraph 4.1 (p. 13). At the end of the work, the graphic part of the qualification work is attached (if available). The explanatory note is accompanied by the following documents:

1. Application for the assignment of the qualification work topic, signed by the student.
2. Application for checking the qualification work for borrowings (plagiarism), signed by the student.
3. Review from the supervisor of the qualification work.
4. Review of the qualification work.
5. Two references with the results of checking the qualification work for plagiarism.
6. Expert opinion on the admission of the qualification work to defence (based on the results of checking the work for plagiarism).
7. Other documents (a certificate of the implementation of the results of the qualification work, a scientific article, etc.), if any.

Note: The explanatory note, together with the developed software, is provided to the reviewer for independent evaluation. The review of the qualification work is entrusted to highly qualified specialists of production, scientific and design institutions, organizations, university staff, and other higher education institutions. The reviewer cannot be the Chairman or a member of the Examination Committee, as well as a teacher of the department where the student is defending.

Supporting documents are placed in an envelope glued to the inside of the cover of the explanatory note.

The admission of the qualification work to defence is certified by the signature of the head of the department on the title page of the explanatory note. The qualification work, not signed by the head of the department, is not allowed for defence.

Note: The head of the department has the right to admit the student to the defence of the qualification work in case of a negative review from the supervisor. A negative review is also not a reason for rejecting the qualification work from its defence.

6.2 Checking, Detecting Plagiarism, and Its Elimination

Checking the qualification work for academic plagiarism is carried out according to the "Regulations on the system of ensuring academic integrity at Khmelnytskyi National University". Based on the results of the check, the student receives references, which are attached to the qualification work.

The level of borrowings in the qualification work is checked no later than three days before the defence of the project and is carried out by a responsible person (hereinafter - responsible), appointed by the head of the department.

The functions of the responsible person are:

- uploading the qualification work (explanatory note) to the Anti-Plagiarism and Unicheck systems and checking it for plagiarism;
- processing references based on the results of the check;
- archiving the qualification work in the repository;
- maintaining the confidentiality of information regarding the qualification work.

The responsible person accepts the completed qualification work, signed by the supervisor, in printed form, as well as its electronic version in formats *.rtf, *.doc, *.docx, *.pdf. The responsible person conducts a selective check for a match between the printed and electronic versions of the qualification work. If the printed and electronic versions do not match, the qualification work is returned to the student to resolve the discrepancies.

After checking the explanatory note for plagiarism, the responsible person issues the appropriate references to the student.

If plagiarism is detected that exceeds the established norms, the student is not allowed to defend the qualification work until the violations are eliminated, and the plagiarism check is repeated.

The final decision on the fact of academic plagiarism in the qualification work of students and their admission to defence is made by the commission, which is formalized in the form of a relevant expert opinion.

6.3 Normative Control

The purpose of normative control is to ensure compliance with the standards, requirements, and rules established by standards and regulatory documents of the university in the qualification work documentation. To carry out normative control, the department appoints a responsible person - a normative controller.

Before submitting the qualification work for normative control, the materials must be printed and signed by the graduate and the supervisor of the qualification work. To ensure quality normative control and error correction, the explanatory note should be completed approximately seven days before the start of the qualification work defence.

Changes and corrections indicated by the normative controller and related to violations of current standards and other regulatory and technical documents are mandatory for inclusion in the qualification work documents.

The normative controller confirms the qualification work's compliance with current standards with his signature. It is not allowed to correct or change documents signed by the normative controller without his knowledge.

The normative controller is responsible for complying with the requirements of current standards and other regulatory and technical documents in the documentation at the same level as the developers of the qualification work documentation.

6.4 Report Preparation

The student's report should briefly and technically competently reflect the essence of the task set, justification of the relevance of the topic, the purpose and objectives of the qualification work, the purpose of development, the essence of the analysis carried out, the functional composition of the development, the tools chosen for software development, design results, the main characteristics of the developed software, conclusions (brief results of the entire design, possible ways of further improvement, use or implementation, etc.).

The report should include three interrelated conditional parts: ***introduction, main part***, and ***conclusions***.

The Introduction should start with an address to the members of the Examination Committee and the presentation of the qualification work topic, for example: "*Dear members of the examination committee! I present to you the qualification work on the topic ...*". Further, in the introduction, it is necessary to define the subject of the economy and the sphere of activity to which the qualification work topic relates, highlight the relevance of the development, indicate the purpose and main tasks of the design.

In ***the main part***, it is necessary to consider possible solutions to the problem briefly, explain how the problem was solved, justify the correctness of the decisions made, reveal the main design results, and demonstrate the level of problem-solving, as well as the main characteristics of the developed software.

The Conclusions should be focused on the main design results, achieving the purpose of the qualification work, the practical significance of the project, recommendations, etc. Conclusions and recommendations should be presented in a generalized form, avoiding excessive detail.

The report should end with the words: "*Thank you for your attention, the report is over.*"

The proposed report structure is generalized and can be specified and changed depending on the content of the qualification work, the results obtained, and the presented visual (demonstration) materials.

6.5 Preparation of Visual Materials

Visual (demonstration) materials should sequentially illustrate the report and ensure the full coverage of all provisions of the qualification work.

The first slide of the presentation should contain the name of the department, the topic of the qualification work, the surname and name of the student, the name, surname, academic degree, and academic title of the supervisor of the qualification work.

On the second slide, a clearly formulated problem statement, the purpose, and the tasks of the design are placed.

On the following slides (1-2), information is provided regarding the relevance of the task set, as well as the results of the analysis of existing solutions and conclusions made based on this analysis (a list of analogues of the developed software with an indication of their shortcomings and limitations).

Further, the slides contain the design results: methods, formulas, the content of the work performed (models, algorithms, structural diagram of the developed software, formats, and algorithms for data transmission and processing, interface scheme, "screenshots" of the windows of the developed software, etc.).

In conclusion, clearly and concisely formulated conclusions are given, indicating the results of the qualification work and their practical value.

The presentation and report should be coordinated in time. The student's speech should be designed for 10-15 minutes.

6.6 Preparation for Answering Questions

Questions during the defence of the qualification work are posed by members of the Examination Committee and present individuals, typically related to the topic of the qualification work and the results presented in the explanatory note and report. Therefore, the graduate should be well-versed in the project, understand the essence of the presented material, comprehend the principles of the developed programme, etc.

The number and nature of questions largely depend on the quality of the report. The student should be prepared for unexpected questions like: "*Why is this necessary?*" or those pertaining to details the student hasn’t addressed. It’s not advisable to answer with: "*That’s what the client wanted*" or "*I don’t know that*." It’s better to showcase one’s erudition and inventiveness, for instance: "*It wasn’t part of the qualification work’s objectives*."

Questions that are unclear to the student should be clarified.

6.7 Rehearsing the Report

Rehearsing the report is crucial in terms of determining the time required for it. During the report rehearsal, it's recommended to use all the equipment that will be needed during the defence of the qualification work, as well as all necessary visual (demonstration) materials. Special attention should be paid to preparing and setting up the necessary equipment for demonstrating the developed software. For reliability, it's also advisable to have a video demonstrating the operation of the developed programme.

7 DEFENCE OF THE QUALIFICATION WORK

The public defence of the qualification work (QW) marks the final stage of the student's efforts, during which the graduate is expected to showcase their professional qualities, demonstrate the results of their work, and present the developed project. The QW defence takes place before an ***Examination Committee***, formed by the rector's order.

Note: The defence of a comprehensive QW is typically scheduled and conducted in a single session of the Examination Committee. Students who have worked on a comprehensive project should be fully familiar with its general part and be prepared to answer questions from the Examination Committee members not only about their individual contributions but also about the general part of the QW.

Students are allowed to defend their QW if they have met all the requirements of the curriculum and programme, submitted their work to the department on time, received positive reviews and feedback from the QW supervisor, underwent standardization checks and plagiarism checks, and completed ***preliminary defence*** as per the set schedule.

On the day of the defence, the student must submit all QW documentation to the Examination Committee's secretary.

The QW defence is conducted in an open session of the Examination Committee with the participation of at least half of its members and the mandatory presence of its Chair.

The QW defence procedure is as follows:

- Introduction of the student and the submitted documents by the Examination Committee's secretary;
- The student's report on the essence of the QW, accompanied by slide or video demonstrations;
- Demonstration of the developed software;
- The student's responses to questions from the Examination Committee members;
- Presentation by the reviewer or announcement of their review;
- The student's responses to the reviewer's comments;
- Speech by the QW supervisor or presentation of their feedback;
- The student's responses to the supervisor's comments;
- Discussion of the project and its defence by the student, followed by the committee's decision regarding the overall evaluation of the QW.

The Examination Committee's secretary records the QW defence procedure. The Examination Committee's decision on the assessment of knowledge and skills demonstrated during the QW defence, as well as the awarding of the student's qualification and issuance of the diploma, is made in a closed session of the Examination Committee by open voting with a simple majority of the Examination Committee members. In case of a tie, the Chair's vote is decisive.

The results of the QW defence are announced on the same day after the completion of the relevant documents and meeting protocols of the Examination Committee and are recorded in the student's individual study plan (record book) and the examination statement with the signatures of all Examination Committee members.

Re-defence of the QW with the aim of improving the grade is not allowed.

A student who has successfully defended their QW is awarded the corresponding degree of higher education by the Examination Committee's decision, granted the relevant educational qualification, and issued a state-standard higher education document, as well as a European-standard diploma supplement.

A student who has fully completed the Bachelor's degree programme and received "excellent" grades for at least 75% of the educational components provided by the curriculum, and other components graded as "good" and defended the QW with a grade of "excellent", is issued a higher education document with a distinction mark. Mandatory participation in research work, confirmed by a motivated recommendation to the Examination Committee accepted at the department meeting, is a prerequisite for awarding a diploma with distinction. The final decision on including a distinction mark in the student's diploma is made by the Examination Committee based on the results of the assessment and considering all submitted materials. If the QW defence grade is unsatisfactory, the student is expelled from the university and receives an academic reference of the established form. Such an individual is allowed to re-defend the work within three years after leaving the university. A student who did not defend the QW for a valid reason, confirmed by relevant documents, may be given a new examination date within one year during the Examination Committee's operation by the university rector upon the faculty dean's recommendation. More detailed information is provided in the "Regulations on the certification of students and scientific degrees at Khmelnytskyi National University". In case of disagreement with the received grade, the student has the right to appeal. The appeal submission and review procedure is regulated by the "Regulations on the certification of students and scientific degrees at Khmelnytskyi National University". QWs, after their defence, are transferred to the university archive.

8 EVALUATION CRITERIA

In accordance with the "Regulations on Control and Evaluation of Learning Outcomes of Students at Khmelnytskyi National University", the QW is evaluated using the national four-point scale and the ECTS scale.

The QW evaluation system is based on the following parameters – quality assessment of the content of the explanatory note, its design, the developed software, and the defence of the QW.

Criteria for evaluating the content quality of the explanatory note:

– relevance of the topic and practical significance of the work;

– the QW content's correspondence to its theme;

– the work's alignment with the QW task;

– objective coverage of the issue with creative use of modern information sources;

– thoroughness of the subject area research;

– clarity and completeness of task formulation;

– presence of new ideas and solutions;

– justification for the choice of methods and means to solve the set task;

– level of design and software solutions and their justification;

– application of modern technologies and programming languages;

– clarity and quality of illustrative material;

– student's degree of independence;

– presence/absence of duplication, descriptive material, and stereotypical solutions that do not affect the essence of the results obtained.

Criteria for evaluating the design of the explanatory note:

– compliance with current standards;

– organic connection between textual and graphical material;

– overall and professional literacy, conciseness, and logical sequence of material presentation.

Criteria for evaluating the quality of the software:

– software's operability and functional suitability;

– software's alignment with the set task and specification requirements; – simplicity and convenience of the interface;

– possibilities for software implementation.

Criteria for evaluating the quality of the QW defence:

– quality and completeness of the report during the QW defence: report's relevance to the project's theme and objectives; mastery of the material, sequence, logic, literacy of material presentation; ability to justify decisions made in the project, briefly explain the purpose and operation of the developed software, draw conclusions, etc.;

– accuracy and completeness of answers during the QW defence: the ability to formulate a reasoned answer to questions, respond to non-standard (problematic) questions, and justify one's position in problematic situations.

A student receives an "***excellent***" grade if they have fully completed the QW adhering to all requirements, and during its defence, demonstrated a literate and logical presentation, accurate and complete answers to questions (including non-standard ones), deep and comprehensive mastery of the study material; ability to connect theory with practice, justify their judgments, draw conclusions; possession of diverse skills and competencies. The explanatory note fully meets the requirements for its content and design and reveals all project provisions. The developed software product meets the technical specifications and is fully functional; modern development tools have been used. The developed programme is uploaded to the GitHub repository with a link provided. The mobile application has been submitted to Google Play or AppStore for publication.

A "***good***" grade is awarded to a student when they have fully completed the QW adhering to all requirements, and during its defence, demonstrate a solid knowledge of the project material, present it competently and substantially, do not make significant inaccuracies in answers, correctly applies theoretical provisions when solving practical tasks, and possesses the necessary skills and techniques for their execution. The explanatory note largely meets the requirements and reveals the key provisions of the project. The developed software product meets the technical specifications and performs the main functions; modern development tools have been used.

A "***satisfactory***" grade is deserved by a student who has completed the QW as per the task but made inaccuracies in its execution; during the defence, they demonstrated knowledge of the main material to the extent necessary for professional activity; acquired and gained practical skills in the field, mainly copes with the execution of practical tasks, but makes violations of logical sequence in presenting the material, mistakes in answers to questions, and struggles when answering modified questions. The explanatory note mostly meets the requirements and reveals most of the QW provisions. The developed software performs most of the necessary functions, or its implementation is done in a simplified form (for example, a business card website).

An "***unsatisfactory***" grade is given when a student has poorly executed the QW, and during its defence, showed unsystematic knowledge, cannot distinguish between primary and secondary, makes mistakes in defining concepts, distorts their meaning, presents the material chaotically and uncertainly, and cannot apply knowledge when solving practical tasks. The explanatory note does not meet the requirements, insufficiently reveals the project provisions. The developed software product performs an insufficient number of functions or does not meet the technical specifications or goes beyond the QW theme.

LIST OF USED SOURCES

1. Стандарт вищої освіти України за спеціальністю 121 «Інженерія програмного забезпечення» для першого (бакалаврського) рівня вищої освіти. Київ : МОН України, 2018. 24 с.
2. Освітньо-професійна програма підготовки бакалавра зі спеціальності «Інженерія програмного забезпечення». URL: https://khmnu.edu.ua/121-ipz-b-op/ (дата звернення: 01.08.2023).
3. Положення про атестацію здобувачів вищої освіти та наукових ступенів у Хмельницькому національному університеті. URL: https://khmnu.edu.ua/wp-content/uploads/normatyvni-dokumenty/polozhennya/pro-atestacziyu.pdf (дата звернення: 06.08.2023)
4. Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті. URL: https://khmnu.edu.ua/wp-content/uploads/normatyvni-dokumenty/polozhennya/pro-systemu-zabezpechennya-akademichnoyi-dobrochesnosti.pdf (дата звернення: 25.07.2023).
5. Положення про контроль і оцінювання результатів навчання здобувачів вищої освіти у Хмельницькому національному університеті. URL: https://khmnu.edu.ua/wp-content/uploads/normatyvni-dokumenty/polozhennya/pro-kontrol-i-oczinyuvannya-rezultativ-navchannya.pdf (дата звернення: 26.07.2023).
6. ДСТУ ГОСТ 2.104:2006. Єдина система конструкторської документації. Основні написи. [Чинний від 2007–07–01]. Вид. офіц. Київ : Держспоживстандарт України. 2007. 23 с.
7. ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура та правила оформлювання. [Чинний від 2017–07–01]. Вид. офіц. Київ : ДП «УкрНДНЦ», 2016. 26 с. (Інформація та документація).
8. ДСТУ 8302:2015. Бібліографічне посилання. Загальні положення та правила складання. [Уведено вперше; чинний від 2016–07–01]. Вид. офіц. Київ : ДП «УкрНДНЦ», 2016. 17 с. (Інформація та документація).
9. ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень в українській мові. Загальні вимоги та правила [Чинний від 2013–08–22]. Вид. офіц. Київ : Мінекономрозвитку України, 2014. 18 с. (Інформація та документація).
10. СОУ 207.01:2017. Текстові документи. Загальні вимоги / Бойко Ю. М., Красильнікова Г. В., Першина Л. І., Косянчук Т. Ф. [2-ге вид., випр.]. Хмельницький : ХНУ, 2018. 45 с.
11. Бойко Ю. М., Першина Л. І. СОУ 207.02:2017. Бібліографічний запис. Загальні вимоги та правила складання. Хмельницький : ХНУ, 2017. 37 с.

APPENDICES

APPENDIX A
(reference)

TITLE PAGE OF THE EXPLANATORY NOTE

Khmelnytskyi National University
Faculty of Information Technology
Department of Software Engineering

QUALIFICATION WORK

____
Title of the work

____

Higher Education Level	_First (Bachelor's)_
Field of Study	_12 Information Technologies_
Programme Subject Area	_121 Software Engineering_
Educational programme	_Software Engineering_

____
Code

Executed by the student of the group ____	____________ Signature	____________________ First Name, SURNAME
Supervisor ________________________ Academic degree, title	____________ Signature	____________________ First Name, SURNAME
Normative Controller	____________ Signature	____________________ First Name, SURNAME

Approved for defence by:

Head of the Department of Software Engineering	____________ Signature	____________________ First Name, SURNAME

Date

Khmelnytskyi 202__

APPENDIX B
(reference)

ASSIGNMENT FORM FOR THE QUALIFICATION WORK

KHMELNYTSKYI NATIONAL UNIVERSITY

Faculty	Information Technology
Department	Software Engineering
Higher Education Level	First (Bachelor's)
Field of Study	12 Information Technologies
Programme Subject Area	121 Software Engineering
Educational programme	Software Engineering

APPROVED BY:
Head of the Department:
____
_________________ 202__

ASSIGNMENT
FOR THE QUALIFICATION WORK

____
Surname, name, patronymic of the student

1 Topic of the work____
____

Supervisor ____
____
Surname, name, patronymic, academic degree, academic title

Approved by the rector's order dated ___ ____ 202__ № __
2 Deadline for submitting the work to the department: ___________________
3 Initial data for the work: ____
____
4 Content of the explanatory note (list of issues to be developed): ______________
____
____
5 List of graphic material (with an indication of mandatory drawings):
____
____
____

Continuation of APPENDIX B

6 Consultants for sections of the qualification work

Section	Surname, initials, and position of the consultant	Signature, Date	
		assignment given	assignment accepted

7 Date of assignment issuance: _____ ___________ 202__

CALENDAR SCHEDULE

Name of stages (sections) of the qualification work	Deadline for completing the work stages	Note
1		
2		
3		
...		

Student ____________ ______________________
Signature Name, SURNAME

Supervisor ____________ ______________________
Signature Name, SURNAME

APPENDIX C
(reference)

EXAMPLE OF "ANNOTATION"

ANNOTATION

Topic of the qualification work: "Software Support for Decision Making in Software Product Marketing ".

Author of the work: Vasyl Semenovych Yakobchuk.

Work Supervisor: Radelchuk Halyna Ivanivna.

Explanatory note: 60 p., 12 figs., 5 tables, 4 appendices, 15 sources.

Graphic part: three A3 format posters.

CORPORATE MARKET, SOFTWARE PRODUCT, INTERNET MARKETING, DECISION-MAKING SYSTEM, PHP, APACHE, MYSQL.

The purpose of the work is to develop a set of models and software to support decision-making when promoting mass software products to the corporate market of small and medium-sized business entities using Internet marketing tools. The qualification work defines the specifics of marketing activities of IT companies specializing in the development and distribution of their software products; an analysis of methods and tools for promoting mass software products in the corporate market has been carried out; a functional model of organizing the process of promoting software products to the market, taking into account the specifics of using the Internet as the main communication channel, has been developed; models and algorithms have been developed to support decision-making at various stages of the process of promoting software products in the corporate market using Internet marketing tools. The PHP programming language, MySQL database server, and Apache web server were used to implement the software system. As a result of the qualification work, a software implementation of the decision support system was carried out when organizing the promotion of software products in the corporate market, as well as practical testing of the obtained results and implementation of the software system.

Student's signature Date

APPENDIX D
(reference)

EXAMPLE OF "CONTENTS"

CONTENTS

APPENDIX E
(reference)

EXAMPLE OF "LIST OF ABBREVIATIONS"

LIST OF ABBREVIATIONS

AWP	–	Automated Workplace
DB	–	Database
API	–	Application Programming Interface
LC	–	Life Cycle
IDE	–	Integrated Development Environment
SDK	–	Software Development Kit
MAS	–	Multi-Agent System
GUI	–	Graphical User Interface
QA	–	Quality Assurance
TS	–	Technical Specification
ACPI	–	Advanced Configuration and Power Interface
BIOS	–	Basic Input-Output System
CUDA	–	Compute Unified Device Architecture
ISDN	–	Integrated Services Digital Network
FAT	–	File Allocation Table
HTML	–	HyperText Mark-up Language
OLE	–	Object Linking and Embedding)
USB	–	Universal Serial Bus
CI/CD		Continuous Integration/Continuous Deployment
WWW	–	World Wide Web

APPENDIX F
(reference)

Examples of Bibliographical Entries

E.1 Books

Single author

1. Дичківська О. О. Інноваційний менеджмент : конспект лекцій. Київ : ДІА, 2018. 82 с.
2. Бондаренко В. Г. Історія України. Львів, 2020. 153 с.
3. Лазор О. Я. Державне управління у сфері реалізації екологічної політики в Україні: організаційно-правові засади : монографія. Львів : Ліга-Прес, 2019. 542 с.
4. Гурманова Л. І. Релігієзнавство : навч. посіб. 2-ге вид., перероб. та допов. Київ : ЦУЛ, 2018. 193 с.

Two authors

1. Мартиненко З. Е., Макар І. В. Управління підприємством: теоретико-методичні засади : монографія. Харків : Щедра садиба плюс, 2021. 296 с.
2. Мороз І. С., Василенко Н. Ю. Маркетинг : конспект лекцій. Київ : Молодь, 2018. 102 с.
3. Вердіна С. А., Волков А. А. Контролінг : навч. посіб. Вид. 3-тє., перероб. та допов. Херсон, 2020. 212 с.

Three authors

1. Тарнавська Г. Я., Марценюк Н. С., Герасимова Т. М. Фінанси : навч. посіб. Львів : Магнолія 2006, 2018. 412 с.
2. Пустовенко В. В., Максименко І. Л., Яким А.С. Безпека життєдіяльності : монографія. Харків : ХНПУ, 2022. 348 с.

Four authors

1. Інновації : навч. посіб. / Гуревич Д. Т., Чекан О. С., Грибан О. М., Макарова В. В. Запоріжжя : ЗНУ, 2018. 389 с.
2. Вища математика : конспект лекцій / Ткачук Т. С. та ін. Київ, 2021. 82 с.

Five or more authors

1. Операційний менеджмент : підручник / С. М. Поплавська та ін. Київ : ЦУЛ, 2021. 267 с.
2. Охорона праці : навч. посіб. / О. І. Подольська та ін. 2-ге вид. Київ : ЦУЛ, 2019. 264 с.

Автор(и) та редактор(и)/упорядники

1. Веретенко В. В. Міжнародний маркетинг : монографія / за заг. наук. ред. В. М. Марценюка. Київ, 2019. 374 с.

2. Освіта в Україні: виклики модернізації : зб. наук. пр. / редкол.: П. М. Марценюк (відп. ред.) та ін. Київ : Ін-т всесвітньої історії НАН України, 2019. 319 с.

E.2 Standards

1. ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура та правила оформлювання. [Чинний від 2017–07–01]. Вид. офіц. Київ : ДП «УкрНДНЦ», 2016. 26 с. (Інформація та документація).

2. СОУ 207.01:2017. Текстові документи. Загальні вимоги / Бойко Ю. М., Красильнікова Г. В., Першина Л. І., Косянчук Т. Ф. [2-ге вид., випр.]. Хмельницький : ХНУ, 2018. 45 с.

E.3 Part of a periodical (magazine, newspaper)

1. Капітанець С. О., Радельчук Г. І. Особливості логування даних та вплив типу додатка на вибір методології логування // Вісник ХНУ, серія Технічні науки. № 6 (315), Том 1. 2022. С. 98–101.

E.4 *Part of conference materials (abstracts of reports)*

1. Максименко Д. В. Методи оперативної діагностики виробничої діяльності підприємства // Зростання ролі бухгалтерського обліку в сучасній економіці : збірник тез та доповідей I Міжнародної науково-практичної конференції (м. Київ, 21 лютого 2018 р.) / відпов. за випуск Мельничук Б. В. Київ, 2013. С.331–335.

2. Капітанець С. О., Радельчук Г. І. Керування логуванням у програмних системах: концептуальні засади. *Розвиток наукової думки постіндустріального суспільства: сучасний дискурс (секція «Комп'ютерна та програмна інженерія»)* : матеріали II Міжнар. наук. конф., м. Львів, 18 листопада 2022 р. Вінниця : Європейська наукова платформа, 2022. С. 223–228.

E.5 *Electronic resources*

1. Україна очима дітей : фотовиставка. URL: http://www.kmu.gov.ua/control/uk/photogallery/gallery?galleryId=15725757& (дата звернення: 15.11.2020).

2. Хміль А. А. Функції державної служби за законодавством України // Юридичний науковий електронний журнал. 2019. № 5. С. 115–118. URL: http://lsej.org.ua/5_2017/32.pdf.

3. Положення про атестацію здобувачів вищої освіти та наукових ступенів у Хмельницькому національному університеті. URL: https://khmnu.edu.ua/wp-content/uploads/normatyvni-dokumenty/polozhennya/pro-atestacziyu.pdf (дата звернення: 06.08.2023).

CONTENTS