KHERSON STATE UNIVERSITY

Faculty of Computer Science, Physics and Mathematics
Department of Computer Science and Software Engineering

Master's Degree Programme in Software Engineering

Development and audit of smart contracts of the educational platform

Supervisor:
Doctor of Physical and Mathematical sciences,
Prof. Volodymyr Peschanenko

Author:
Master's student
Konnova Olga

Reviewer:
PhD (Information Technology),
CEO of GARUDA.AI
Yuliia Tarasich

Kherson – Ivano-Frankivsk – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики

Кафедра комп'ютерних наук та програмної інженерії

РОЗРОБКА ТА АУДИТ СМАРТ-КОНТРАКТІВ ОСВІТНЬОЇ ПЛАТФОРМИ

Кваліфікаційна робота (проєкт)
на здобуття ступеня вищої освіти «магістр»

Виконала: здобувачка 2 курсу 241М групи

Спеціальності 121 Інженерія програмного забезпечення

Освітньо-професійної (наукової) програми Інженерія програмного забезпечення
Коннова Ольга Владиславівна

Керівник: доктор фізико-математичних наук, професор
Песчаненко Володимир Сергійович

Рецензент: докторка філософії (Інформаційні технології), докторантка Інституту кібернетики імені В.М.Глушкова НАН України, директорка ТОВ «ГАРУДА.АІ»
Тарасіч Юлія

Херсон – Івано-Франківськ – 2023

АНОТАЦІЯ

Онлайн освітні платформи почали з'являтися досить давно, але вони практично не розглядалися як варіант повної заміни традиційного офлайн-навчання у школах та університетах. Однак пандемія Covid-19 повністю змінила цю ситуацію. Студенти почали вчитися віддалено та виникла потреба в інструментах, які могли б підтримати цей процес. Однією з головних переваг освітніх онлайн-платформ є те, що вони дозволяють студентам здобувати якісну освіту незалежно від географічного положення та фізичного доступу до навчальних закладів. Тому не лише школярі чи студенти, а й працюючі люди, які бажають освоїти нову спеціальність чи підвищити свою кваліфікацію, можуть навчатися у зручний для них час.

Перехід до онлайн освіти, зростання освітніх платформ та збільшення обсягу освітніх даних створюють серйозні проблеми, такі як безпека даних, автентифікація та управління фінансами. Блокчейн може стати потужним інструментом підвищення надійності та безпеки платформ онлайн-навчання. Він забезпечує незмінність даних, захищає їх від кібератак, забезпечує прозорість та безпеку для всіх учасників системи. Це може допомогти зміцнити довіру до онлайн-освіти та зробити її більш доступною та надійною для всіх. І одним із важливих компонентів блокчейну є смарт-контракти.

Смарт-контракти відкривають нові освітні можливості, перетворюючи традиційні системи навчання на безпечне та прозоре середовище. Однак із зростанням використання смарт-контрактів у додатках бізнес-процесів виникає необхідність перевірки безпеки. Смарт-контракти ґрунтуються на програмному коді, тому можуть містити помилки, що призводять до некоректного виконання контракту. Оскільки сфера використання смарт-контрактів часто пов'язана із фінансами, ціна таких помилок може бути досить високою.

Актуальність дослідження полягає у необхідності створення надійних та безпечних смарт-контрактів для підтримки освітнього процесу, реєстрації студентів, визначення їх успішності, розподілу фінансових ресурсів та багатьох інших аспектів освіти. Частиною мети цього проекту є перевірка розроблених смарт-контрактів на наявність вразливостей, коректність виконання та відповідність специфікаціям.

Зв'язок роботи з науковими програмами, планами, темами. Тема роботи знаходиться у сфері наукових досліджень кафедри комп'ютерних наук та програмної інженерії: «Кібербезпека та інсерційне моделювання» (проф. Песчаненко В.С.) та «Криптоекономіка та блокчейн» (проф. Песчаненко В.С., Кобець В.М.).

Об'єкт дослідження – смарт-контракти для освітньої платформи.

Предмет дослідження – процес розробки та аудиту смарт-контрактів.

Мета дослідження: розроблення смарт-контрактів, які регулюють основні бізнес-процеси освітньої платформи. Проведення аудиту смарт-контрактів з метою виявлення потенційних вразливостей, помилок у логіці, а також перевірка відповідності вимогам.

Досягнення мети дослідження передбачає розв'язання таких завдань:

- Огляд можливостей та переваг використання блокчейну та смарт-контрактів в освітній діяльності.
- Визначення основних вимог до смарт-контрактів освітньої платформи та бізнес-процесів, які можуть керуватися смарт-контрактами.
- Вибір блокчейн платформи та інструментів для реалізації смарт-контрактів.
- Реалізація смарт-контрактів на мові програмування TEAL на блокчейні Algorand.

- Аналіз коду смарт-контрактів для виявлення можливих вразливостей та хибної поведінки.

Публікації. О.В. Співаковський , М. О. Вінник, М. Ю. Полторацький, О. В. Коннова. К84 Співаковський О.В. Криптоекономіка. Навч.- метод. посіб. / О.В. Співаковський, М.О. Вінник, М.Ю. Полторацький, О.В. Коннова. – Херсон: Херсонський державний університет, 2023. – 162 с.

ОСНОВНИЙ ЗМІСТ РОБОТИ

У **вступі** обґрунтовано актуальність теми, визначено об’єкт та предмет дослідження, мету та завдання.

У першому розділі – **“Overview of blockchain technology and the possibilities of its use in education”** (Огляд технології блокчейн та можливостей її використання в освіті) – було розглянуто поняття блокчейн і смарт-контракти. Ми описали принципи роботи цих технологій, а також переваги, які вони надають. Також ми аналізували можливості використання блокчейну та смарт-контрактів в освіті. Основні способи використання блокчейну в освіті:

- зберігання академічних даних;
- створення курсів з використанням блокчейну;
- стимулювання студентів та викладачів для отримання кращих результатів;
- захист авторських прав і антиплагіат.

Ми описали покращення, які ці технології можуть внести в навчальний процес. Серед основних переваг, які дає використання смарт-контрактів у навчальному процесі:

- автоматизація рутинних процесів;

- забезпечення високого рівня довіри між учасниками освітнього процесу;
- зменшення ризику помилок, пов'язаних з ручною працею;
- забезпечення високого рівня захисту.

У другому розділі – **“Determination of smart contract requirements and selection of tools”** (Визначення вимог до смарт-контрактів та вибір інструментів) – було визначено основні ролі користувачів на освітній платформі та дії кожного з них. Слід зазначити, що увага була приділена тим діям, які регулюватимуться смарт-контрактами. Докладно описано процес створення та розміщення курсу на платформі.

Ми також визначили платформу та інструменти для розробки смарт-контрактів для освітньої платформи. Таким чином, у якості блокчейн-платформи було обрано Algorand. Ми розглянули особливості смарт-контрактів Algorand, а також їх різновиди: смарт-контракти без збереження та зі збереженням стану, а також основні відмінності між ними. Нами було розглянуто особливості мови програмування для розробки смарт-контрактів Algorand - TEAL (Transaction Execution Approval Language).

У третьому розділі – **“Implementation and audit of smart contracts”** (Реалізація та аудит смарт-контрактів) – нами було детально розглянуто процес реалізації смарт-контрактів із збереженням стану на TEAL та їх взаємодію з клієнтським додатком за допомогою Algorand JS SDK. Описано структуру смарт-контрактів та основні методи їх роботи. У ході роботи було створено смарт-контракти для управління курсами та управління винагородами.

Додаток CourseManagement необхідний для управління основними процесами на платформі, пов'язаними з курсом: створенням курсу, розрахунком його вартості, реєстрацією студентів на курс, збереженням балів студентів, розрахунком суми стипендії для студентів та винагороди для викладачів. Смарт-контракт RewardsManagement контролює розподіл

винагород серед користувачів. Він також функціонує як рахунок умовного депонування.

Також ми описали основні вразливості, які можуть існувати у коді смарт-контрактів, створених на TEAL. Ми розглянули основні підходи щодо верифікації коду смарт-контрактів. У цій роботі ми перевірили розроблені смарт-контракти на наявність описаних вразливостей. Ми також використали інструмент статичного аналізу коду смарт-контрактів на TEAL – Tealer, який дозволяє нам шукати вразливості у коді контракту.

ВИСНОВКИ

У цій роботі ми розглянули можливості використання блокчейну та, зокрема, смарт-контрактів, в освіті. Ми проаналізували особливості цих технологій, а також переваги, які вони надають.

Це дослідження було здійснено для розгляду процесу розробки смарт-контрактів для освітньої платформи на основі блокчейну. Смарт-контракти, що розробляються, дозволяють керувати процесом реєстрації студентів на курс, зберігати їх бали в блокчейні, розраховувати розмір стипендій для студентів та винагород для викладачів, а також розподіляти інші фінансові винагороди серед користувачів платформи.

У ході нашої роботи ми визначили основні ролі користувачів на освітній платформі та дії кожного з них, виділивши варіанти використання, які регулюються смарт-контрактами.

Algorand був обраний у якості блокчейн-платформа для розробки смарт-контрактів. Це один з найшвидших блокчейнів, який надає розширені можливості створення смарт-контрактів з низькими комісіями за транзакції. У якості мови для написання коду смарт-контракту для освітньої платформи було обрано TEAL.

Ми розробили два смарт-контракти: CourseManagement та RewardsManagement. Додаток CourseManagement управляє основними

процесами на платформі, пов'язаними з курсом: створенням курсу, розрахунком його вартості, реєстрацією студентів на курс, збереженням балів студентів у блокчейні, розрахунком суми стипендії для студентів та винагороди для викладачів. Програма RewardsManagement контролює розподіл фінансових винагород серед користувачів платформи. Він також функціонує як рахунок умовного депонування.

У цій роботі ми описали основні вразливості у смарт-контрактах, написаних мовою TEAL. Ми проаналізували причини їх виникнення та можливості усунення. Розроблені смарт-контракти також протестували на наявність описаних вразливостей. Ми використали інструмент статичного аналізу коду для смарт-контрактів на TEAL — Tealer, який дозволяє шукати вразливості в коді контракту.

LIST OF ABBREVIATIONS

P2P network — Peer-to-peer network

PoW — Proof of Work consensus mechanism

PoS — Proof of Stake consensus protocol

PPoS — Pure Proof-of-Stake consensus algorithm used by Algorand blockchain

AVM — Algorand Virtual Machine

ASA — Algorand Standart Asset

ASC — Algorand Smart Contract

DLT — Distributed Ledger Technology

TEAL — Transaction Execution Approval Language

SDK — Software Development Kit

CONTENT

INTRODUCTION

Online educational platforms began to appear quite a long time ago, but they were almost not considered as an option to completely replace traditional offline education in schools and universities. However, the Covid-19 pandemic has completely changed this situation. Students began to study remotely and there was a need for tools that could support this process. One of the main advantages of online educational platforms is that they allow students to receive quality education regardless of geographic location and physical access to educational institutions. Therefore, not only schoolchildren or students, but also working people who would like to learn a new specialty or improve their qualifications can study at a time convenient for them.

The shift to online and distance education, the growth of educational platforms and the increase in the amount of educational data pose serious challenges such as data security, authentication, and financial management. Blockchain can be a powerful tool to improve the reliability and security of online learning platforms. It ensures data immutability, protects it from cyber attacks, and ensures transparency and security for all system participants. This can help build trust in online education and make it more accessible and reliable for everyone. And one of the important components of the blockchain is smart contracts.

Smart contracts open up new educational opportunities, turning traditional learning systems into secure and transparent environments. However, with the growing use of smart contracts in business process applications, there is a need for security validation. Smart contracts are based on software code, so they can contain errors that lead to incorrect execution of the contract. Since the area of use of smart contracts is often related to finance, the cost of such errors can be quite high.

Relevance of the study lies in the need to create reliable and secure smart contracts to support the educational process, student registration, determining

their success, distribution of financial resources, and many other aspects of education. Part of the aim of this project is to check the developed smart contracts for vulnerabilities, correct execution, and compliance with specifications.

The topic of our work is within the scope of scientific research of the Department of Computer Sciences and Software Engineering: "Cybersecurity and Insertion Modeling" (Prof. V.S. Peschanenko) and "Cryptoeconomics and Blockchain" (Prof. V.S. Peschanenko, Prof. V. M. Kobets).

Object of study – smart contracts for the educational platform.

Subject of study – the process of developing and auditing smart contracts.

The purpose of the study – development of smart contracts that regulate the main business processes of the educational platform. Auditing smart contracts to identify potential vulnerabilities, logic errors, and compliance checks.

Achieving the purpose of the study involves solving the following ***tasks***:

- An overview of the possibilities and advantages of using blockchain and smart contracts in educational activities.
- Defining the basic requirements for smart contracts of the educational platform and business processes that can be managed by smart contracts.
- Choosing a blockchain platform and tools for implementing smart contracts.
- Implementation of smart contracts in the TEAL programming language on the Algorand blockchain.
- Analysis of smart contract code to identify possible vulnerabilities and misbehavior.

Research methods. The following methods were used to reach the purposes of the study: informational and logical analysis, case study, description, modeling, and experiment.

Publications. Spivakovs′kyy O.V., Vinnyk M. O., Poltorats′kyy M. YU., Konnova O. V. Kryptoekonomika: navch.- metod. posib. Kherson: Khersons′kyy derzhavnyy universytet, 2023. – 162 s. (Spivakovskyi O.V., Vinnyk M.O., Poltoratskyi M.Yu., Konnova O.V. Cryptoeconomics: Textbook. Kherson: Kherson State University, 2023. – 162 p.)

KonnovaO., LetychevskyiO., PeschanenkoV., & PoltoratskyiM. (2024). AN ALGEBRAIC APPROACH TO THE VERIFICATION OF SMART CONTRACTS IN TEAL. Journal of Information Technologies in Education (ITE), (54).

Structure of the study. The thesis consists of the list of abbreviations, introduction, three chapters, conclusion, and list of sources.

CHAPTER 1
OVERVIEW OF BLOCKCHAIN TECHNOLOGY AND THE POSSIBILITIES OF ITS USE IN EDUCATION

1.1 Blockchain and smart contracts

Blockchain has introduced many new opportunities to the world and provided huge benefits through increased transparency, distributed ledgers and decentralization. This technology was first described by a group of researchers in 1991 and was put into practice only in 2008, when an unknown user under the pseudonym Satoshi Nakamoto published a technical description of his cryptocurrency protocol [1]. This technology has opened up new opportunities to get rid of intermediaries and ensure a high level of security in many areas of activity.

1.2.1 Blockchain

In the Cryptopedia glossary, the definition of blockchain is as follows “A blockchain is a public ledger of transactions that is maintained and verified by a decentralized, peer-to-peer (P2P) network of computers that adhere to a consensus mechanism to confirm data. Each computer in a blockchain network maintains its own copy of the shared record, making it nearly impossible for a single computer to alter past transactions or for malicious actors to overwhelm the network. Sufficiently decentralized blockchains do not rely on centralized authorities or intermediaries to transact globally, securely, verifiably, and quickly, making technology like cryptocurrency possible.” [2]

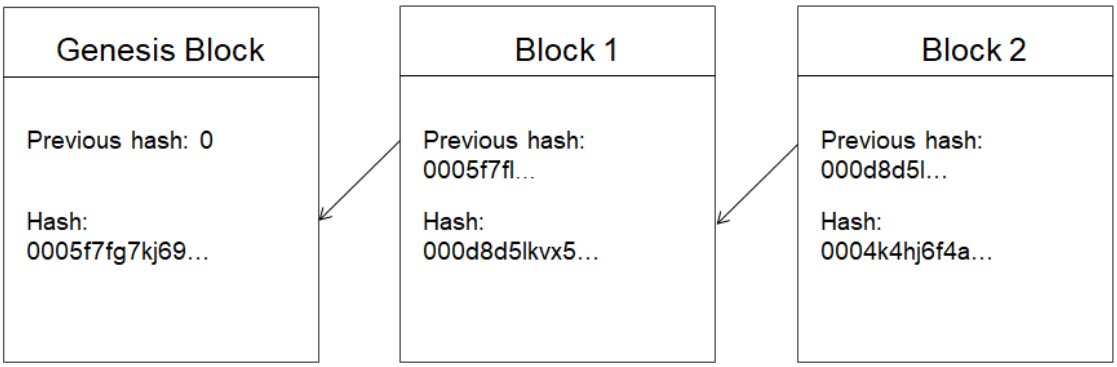
The blockchain concept has certain features:

- Decentralization: All data stored inside the blockchain does not belong to one person. There is no single point of control or failure, which in turn makes the blockchain more secure and resilient to attacks or data leakage.

- Transparency: All data stored on the blockchain is visible to everyone who is part of the network. This simplifies the tracking and verification of transactions and ensures their accuracy.

- Immutability: All data within the blockchain cannot be changed thanks to the cryptographic hash function. This “creates a permanent record of all transactions that can be verified by anyone with access to the blockchain network” [3].

Data stored on the blockchain is secure and immutable, thanks to cryptography. Each block is referenced by a unique character string generated by a cryptographic hash function. Each block is linked to the previous block (known as the parent block) by storing the parent's hash (Pic. 1.1). Therefore, any changes made to the contents of a block will change the hash of the block. Thus, falsifying data in any block of the blockchain will change the hash of all subsequent blocks.



Pic. 1.1 — Link between blocks

In addition, all transactions in a block are verified and agreed upon by a consensus mechanism, ensuring that every transaction is valid and accurate. Therefore, the user cannot change transaction records.

A consensus algorithm is a mechanism that is used in blockchain systems to agree on changes made to the distributed ledger. It ensures that none of the network participants can arbitrarily add, delete, or change the data contained in the registry.

At the moment, there are several consensus algorithms. The two most common ones are:

- Proof-of-Work (PoW) requires the participant to prove that work has been done and provide proof in the form of a puzzle solution. This proof gives the node the right to add a block for the transaction to the blockchain. Typically, a proof of work involves solving a mathematically complex puzzle using cryptographic techniques. Whoever solves the puzzle first gives the proof, and after that, the block will be added. This process is called mining and requires large computing resources, i.e., a long processing time and high-power consumption of computers. This mechanism is used by Bitcoin and Dogecoin for their BTC and DOGE currencies.

- Proof-of-Stake (PoS) requires the participating validator to have a certain percentage of the stake in the network. It is believed that this node is interested in maintaining the reliability of the network. This algorithm is used by Cardano, Solana, and Avalanche for their ADA, SOL, and AVAX currencies, respectively.

Other consensus algorithms include Delegated Proof-of-Stake, Proof-of-Importance, Proof-of-Activity, Proof-of-Burn, Proof-of-Capacity and others.

- **Advantages and disadvantages of blockchain**

Considering the principle of operation of the blockchain as well as its features, we can highlight the following advantages of this technology:

- Ensuring a high level of security: the three principles of blockchain — cryptography, decentralization, and consensus — allow developers to provide a secure system that is almost impossible to interfere with. Data is also stored in blocks on the computers of many users. This reduces the risk of hacker attacks as well as technical failures.
- Ensuring independence by eliminating intermediaries: in the blockchain network, transactions occur without intermediaries, as well as third parties; that is, the blockchain is not managed or controlled by a bank or government, which means there is no possibility of interference in the process.
- Ensuring transparency: blockchain transactions are typically public and visible to all network participants. This provides a high level of transparency and allows participants to verify and confirm transactions.
- Ability to create smart contracts: blockchain allows developers to create smart contracts that are automatically executed when specified conditions are met. This can simplify many business processes and make them more transparent.

Along with a lot of advantages, blockchain technology still has some limitations:

- Difficulty in changing data: Once data is added, it is quite difficult to modify it. Although stability is considered an advantage of blockchain, there are situations in which it can be a disadvantage.
- Limited scalability: public blockchains such as Bitcoin or Ethereum can experience limited throughput and latency when the network is heavily loaded.
- Lack of universal standards of legal regulation: many countries have not yet defined clear norms and rules regarding the use of blockchain technologies and cryptocurrencies. Also, smart contracts that function on

the blockchain may have legal significance, but laws do not always take their specifics into account.

- Environmental issues: Some blockchains, especially Bitcoin, require significant computing resources, resulting in serious carbon emissions and environmental impacts.

1.2.2 Smart contracts

One of the main components of the blockchain are smart contracts. The concept of smart contracts was first introduced by Nick Szabo in the 1990s. However, they became widespread with the advent of the Ethereum platform, the concept of which was described in 2013.

“Smart contracts are simply programs stored on a blockchain that run when predetermined conditions are met. They typically are used to automate the execution of an agreement so that all participants can be immediately certain of the outcome, without any intermediary’s involvement or time loss.” [4] It is thanks to smart contracts that the decentralized nature of the network is ensured.

Each smart contract has two components:

- Code: A set of rules and functions programmed into the contract. It defines how the smart contract should be executed and under what conditions. It defines the functionality of the smart contract and often cannot be changed once it is added to the blockchain.
- State: A set of data and variables that are stored on the blockchain and describe the current state of this contract. This may include account balances, timestamps, other contract or account addresses, transaction IDs, and other information.

Smart contracts are developed using programming languages. The choice of a specific language depends on the selected blockchain. For example, the

Solidity and Vyper languages are used to develop smart contracts for the Ethereum blockchain, and the Rust programming language is used for the Solana, NEAR, or Polkadot blockchains. Other languages for developing smart contracts include C++, JavaScript, Yul, Python, and others. Regardless of the blockchain and programming language, the main goal of smart contracts is to ensure the security and transparency of processes. That is why it is important that the contract code does not contain vulnerabilities and is executed in accordance with the requirements.

There are also several types of smart contracts:

- Smart Legal Contracts (SLC). These contracts combine legal logic with the technicalities of blockchain and smart contracts to provide more secure, reliable, and efficient agreements between parties. They are legally binding and require the parties to fulfill their contractual obligations. Smart legal contracts can be used to perform cryptocurrency transactions as well as to register real estate or other applications. Such smart contracts underlie DeFi projects, cryptocurrency exchanges, and NFT marketplaces.
- Decentralized Autonomous Organizations (DAO). These are organizations that are managed by smart contracts on the blockchain and operate autonomously, without centralized control. The rules of the organization and the rights of participants are encoded in smart contracts that cannot be changed without the consent of other participants.
- Application Logic Contracts (ALC). ALCs contain application code that enables communication between different devices. They allow interaction and communication between different devices, for example, through the integration of the Internet of Things (IoT) with blockchain technology. ALCs are an important component of multifunctional smart contracts and mostly operate within a control program.

Smart contracts have huge potential in many areas of activity. They help to automate many business processes, get rid of intermediaries in transactions, and reduce the cost of human labor. The following main areas of use of smart contracts can be highlighted:

- Cryptocurrency and Finance: Smart contracts are used to create and manage cryptocurrencies and tokens.
- Real estate: Smart contracts can be used to automate the processes of buying, selling, and renting real estate, as well as to store and manage property documents.
- Medicine and healthcare: Smart contracts can be used to manage access to medical data, medical prescriptions, etc.
- Voting: Smart contracts can ensure security and prevent vote fraud during elections and voting.

Also, one of the main sectors where smart contracts can be useful is education. Next, we will consider in more detail the possibilities of using blockchain, and smart contracts in particular, in education. As well as the benefits that this approach will bring to this area.

1.2 Analysis of blockchain use cases in education

Education in the classic form with offline classes in educational institutions still remains the main approach to teaching, but the Covid-19 pandemic and the war in Ukraine have shown that online education can almost completely replace offline classes at school or university. Online schools and platforms for online courses are becoming an increasingly popular and sometimes even necessary alternative to offline education. However, such online educational platforms can also become a supplement and a good tool for already existing educational institutions.

Traditional offline education at a school or university has many advantages but also certain disadvantages:

- Geographic restrictions: Students must be physically located at the university or school, which leads to restrictions in the choice of educational institutions.
- High Costs: Traditional education can be expensive through fees for courses, textbooks, accommodations and other expenses. Students also spend time and money getting to university or school.
- Limited opportunities for individualization: Study groups in universities and schools are usually quite large in terms of the number of students. Because of this, it is difficult to ensure an individual approach to learning.
- Insufficient motivation of teachers: in ordinary schools and universities, the salary level of teachers does not depend much on the quality of their teaching, so teachers are not motivated to produce higher quality educational content.

Many people choose the flexibility and accessibility of online education, which is why online educational platforms are becoming increasingly popular.

And the technology that can significantly improve such online platforms is blockchain.

1.2.1 Blockchain use cases in education

Blockchain can improve education by increasing transparency by providing a decentralized and secure way to store, validate, and share educational data. It is also important to remember about stimulating learning through the use of tokenomic models with rewarding students for good results. The online orientation of the blockchain is an important alternative to the modern educational process in terms of not only platforms for online classes but also effective and high-quality assessment, and document management with the possibility of constant updating and improvement.

The main uses of blockchain in education are:

- Storage of academic records: blockchain can serve as a reliable database, where academic achievements, certificates, diplomas, and other qualifications can be saved. Storing student achievement data on the blockchain allows employers and other stakeholders to verify the authenticity and validity of these documents.
- Stimulating students and teachers for better results: incentive and reward systems using tokenomics can help significantly improve students' academic success rates. Teachers can use tokens to reward their students in the online learning environment for completing modules or other assignments. In addition, the gamification component of learning methodology and tokenization could significantly change the teaching and learning process.
- Creation of courses using the blockchain: blockchain can serve as a basis for creating decentralized platforms for online learning where experts and educators can provide their services. This allows

students and teachers to interact directly, minimizing intermediaries and reducing the cost of access to education. Smart contracts can verify the completion of tasks and distribute crypto tokens to students and teachers.

- Copyright protection and anti-plagiarism: Blockchain can be used to create systems that track and verify the originality of scientific papers and educational materials, helping to fight plagiarism and protect copyrights.

1.2.2 Literature review on the use of blockchain in education

The use of blockchain in education is still in its early stages: only a few institutions are using this technology. The first blockchain technology in education was officially used in 2017 at the University of Nicosia (UNIC), which decided to modernize and simplify the process of storing any documents on specialization (diploma, certificate, research paper). The University of Surrey (UniS) developed the ARCHANGEL system, which is considered one of the first projects in the field of education, created on the basis of distributed ledger technology [5].

Opportunities and problems associated with the use of blockchain in education are considered in the following works:

Yan Ma and Yiming Fang in the paper [6] comprehensively summarize the recent applications of blockchain in education, especially those related to learning records, certificate issuance and management, as well as a decentralized educational ecosystem. Technical and non-technical problems of using blockchain in education were also discussed.

The paper [7] studied the key factors that influence the decision of educational institutions to use blockchain technology for e-learning. The authors also proposed an extended model of the Technology Acceptance Model for the

implementation of blockchain technology in the educational process. As a result of the study, the authors found that compatibility, trialability and relative advantage have a significant impact on the use of DLT in an educational institution.

The main features and technical principles of the application of blockchain technology are discussed in the work [8]. Han Sun and co-authors also propose a solution to the problems of online education based on blockchain technology. The article considers the possibility of a full recording of the learning trajectory, trusted certification of learning results and decentralized sharing of education resources.

The work [9] presents research and coverage of the experience of practical use of blockchain technology in the education in Ukraine, in particular in the sector of learning management systems. Among the possible options for using blockchain in education, the author highlights the following: certification of learning outcomes, accreditation of educational programs, security of learning management systems, management of learning outcomes and rewards.

Special attention is paid to the use of smart contracts in education. The paper [10] considers the possibility of creating a secure system based on smart contracts for the examination system of a large university with a large number of affiliated colleges. This article also analyzed various areas of smart contracts and security issues. In the work [11] authors describe the education digital authentication system based on blockchain technology with the use of smart contracts. Using this authentication approach will help protect sensitive information from unauthorized interference or data theft.

1.2.3 Advantages of using smart contracts in education

We can highlight the following benefits of using smart contracts in education:

- automation of routine processes, such as registration for courses, issuance of certificates, testing, and other administrative tasks. This allows teachers and students to pay more attention to the educational process.
- the possibility of creating automatic regular payments for courses or materials. This allows users to reduce payment processing costs and ensures timely payment.
- ensuring a high level of trust between participants in the educational process, since operations are performed automatically in accordance with the logic and conditions prescribed in the contract. The smart contract code is open and any user can access it.
- reducing the risk of errors associated with manual data entry due to process automation using smart contracts.
- providing a high level of protection due to the fact that the data in smart contracts is cryptographically protected, which ensures reliability and security.

Thus, the use of smart contracts can improve the quality and accessibility of education, make most processes more rational, and provide more efficient and convenient interaction between all participants in the educational process.

In this work, we will consider smart contracts for the educational platform. They will regulate all processes on the platform, such as registration, distribution of rewards for students and teachers, calculation of teacher rankings, and so on.

Conclusions to the chapter:

In this section, we considered the concepts of blockchain and smart contracts. We have described the principles of operation of these technologies as well as the advantages they provide. We also analyzed the possibilities of using blockchain and smart contracts in education. The main uses of blockchain in education are:

- Storage of academic records.
- Stimulating students and teachers for better results.
- Creation of courses using the blockchain.
- Copyright protection and anti-plagiarism.

We have described the improvements that these technologies can bring to the educational process. Among the main advantages that the use of smart contracts brings to the educational process are:

- Automation of routine processes.
- Ensuring a high level of trust between participants in the educational process.
- Reducing the risk of errors associated with manual labor.
- Providing a high level of protection

CHAPTER 2

DETERMINATION OF SMART CONTRACTS REQUIREMENTS AND SELECTION OF TOOLS

2.1 Requirements for the smart contracts and design

2.1.1 Educational Platform

An online education platform is seen as “an integrated set of interactive online services that provides the teachers, learners, parents information, tools, and resources to support and enhance educational delivery and management.” [12]

In this work, an educational blockchain-based platform using tokenomics is considered. It will allow teachers, students, and other stakeholders to interact to provide a better educational process, and will also stimulate them to be more productive through the use of a system of incentives and rewards.

Teachers can create their courses and post them on the platform, students can register for them and study, and employers can look for potential employees among the best students. This way, each participant can benefit from the interaction as well as be rewarded for their success.

The platform will use smart contracts to automatically record and save student grades, ensuring their reliability and accessibility to all stakeholders, and to distribute scholarships, salaries, and other types of financial rewards.

The model of education tokenization was described in more detail in the work [13]. The formalization of the model was also done to find “modeling errors, shortcomings or possible contradictions; search for effective system scenarios” [13].

Next, we will consider the main roles of users on the platform, as well as the interaction between them. This will allow us to identify and describe the

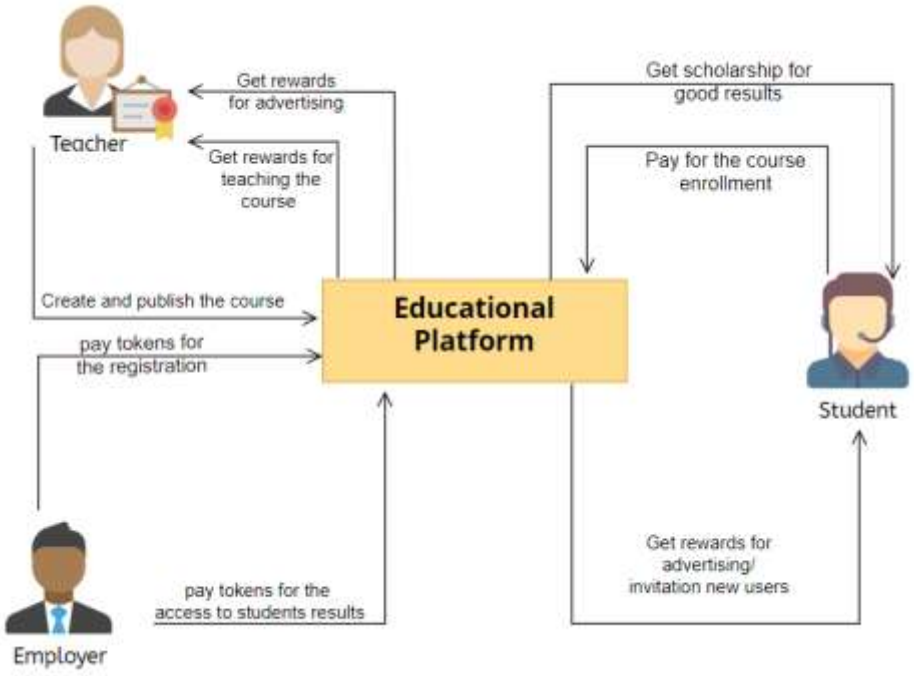
main processes and requirements for them, for the further creation of smart contracts.

2.1.2 Definition of user roles on the platform

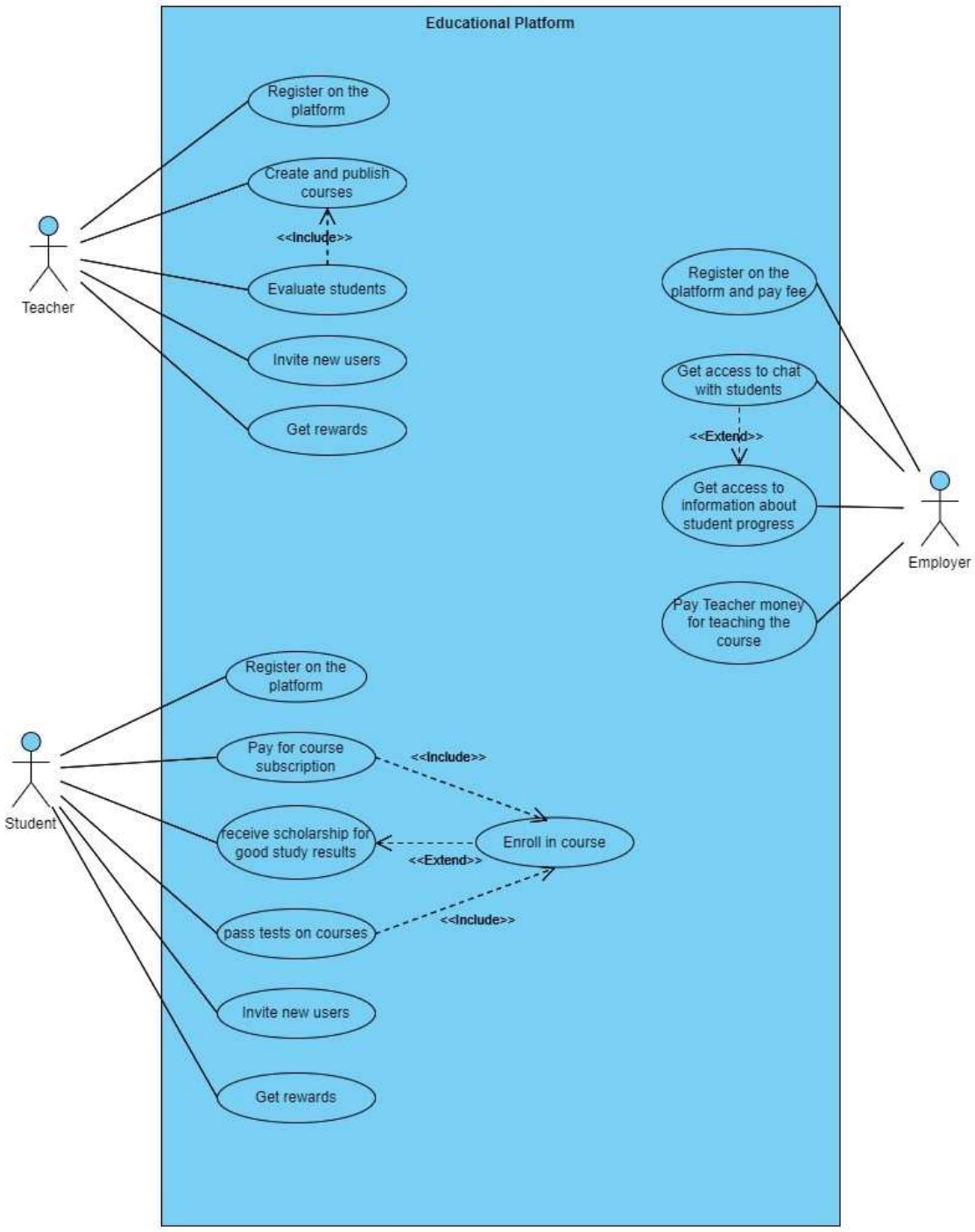
During the work on the platform design, we identified the following user roles:

- Teacher — the user who can create and host educational courses on the platform and evaluate students for completing courses.
- Student — the user who can choose courses and enroll in them by paying a certain amount of money.
- Employer — the user who can access student performance information. Also gets access to the chat for the opportunity to discuss cooperation with the student. Thus, the employer can choose a suitable candidate and offer him a job.

The principle of user interaction with the platform is presented in the diagram (Pic. 2.1). User roles on the platform, as well as their main actions, are presented in the Use Case diagram (Pic. 2.2).



Pic. 2.1 — User interaction with the platform



Pic. 2.2 — Users actions on the platform

Next, we will describe users` actions on the platform in more detail. Attention is paid to those use cases that will be handled by smart contracts.

Teacher:

- can create and host courses on the platform by paying a fixed fee to the platform;
- can evaluate the student's achievements while studying. Evaluation can occur at intervals determined by the teacher (for example, at the end of each month);
- can invite other people to become users of the platform. The teacher will receive a reward for this. The reward is given after the invitee registers for the course with any teacher (for a student) or after creating his own course (for a teacher).

Student:

- can choose the course on the platform and enroll in it. In this case, the student must pay the course fee set by the teacher to the platform.
- can study course materials, participate in online classes, and take tests. Depending on the student's performance during the course, the student may receive a scholarship (based on his grades in the course).
- can also invite new users to the platform. The student will receive a reward for this. The reward is given after the invitee registers for the course with any teacher (for a student) or after creating his own course (for a teacher).
- can take a survey on the platform and receive rewards for it. The survey may concern a specific course that the student took (taken after completion of the course) and the teacher who taught the course. And also about the quality of the platform as a whole, the convenience of new developments, and so on.

Employer:

- can register on the platform. In this case, the employer must pay an appropriate fee to the platform. It depends on the number of staff of the company or firm.

  Company sizes are ranked:

  1. Small: less than 50 employees;
  2. Medium: from 50 to 249 employees;
  3. Large: from 250 to several thousand employees.

- can pay tokens to be able to access student success information. Fee amount is calculated by formula:

  fee = (studentAmount*amountImpactFactor) + (teacherRating*ratingImpactFactor) (1)

  where:

  - studentAmount — number of students enrolled in the course;
  - amountImpactFactor — coefficient that determines the influence of the number of students on the fee size;
  - teacherRating — Teacher`s Rating value on the platform;
  - ratingImpactFactor — coefficient that determines the influence of the Rating value on the fee size;

  Also gets access to the chat for the opportunity to discuss cooperation with the student. If he wants to get the opportunity to communicate with the student, he must pay the fee:

  fee = studentScore*10 (2)

  where

  - studentScore — the student's current grade in this course.

- can order a teacher to create a course he needs to find a candidate for a job or improve the qualifications of his employees. In this case, he must pay a fee:

  fee = teacherRating*(creationCourseFee+courseType) (3)

  where:

  - teacherRating — Rating on the platform of the teacher who will create the course;
  - creationCourseFee — base fee for creating a course on the platform;
  - courseType — course type (will be described in more detail in the section 2.1.3).

  The funds are distributed as follows: 10% to the reserve, 90% to the Reward account.

  Considering the Rating value when calculating the fee value will allow the employer to choose whether he wants to order a course at a higher price, but from a teacher with a higher rating, or from a teacher with a lower rating at a lower price. This will also encourage teachers to create better courses to improve their Ratings.

Next, we will describe in more detail the processes that will be controlled by smart contracts on the platform.

2.1.3 Course creation and publication on the platform

The main process of the educational platform is the creation of a course by the teacher and its publication on the platform. As well as recruiting students for the course, training, testing, and completing the course. Next, we provide a more detailed description of this process on the educational platform that is being developed:

1) The teacher starts creating the course and indicates the basic information about the course: name, description, duration (number of modules), number of participants, type (asynchronous (without video materials, only text materials and presentations), asynchronous (with teacher's video materials), synchronous (online lessons via video conferencing)).
2) The system offers the teacher the optimal cost of the course depending on the type, duration of the course and the rating of the teacher.

   courseCost = (basicModuleCost*typeCoef*durationCoef*ratingCoef)* moduleNumber (4)

   where:

   - basicModuleCost — basic cost of one course module;
   - typeCoef — coefficient for the course type;
   - durationCoef — course duration coefficient;
   - ratingCoef — coefficient that depends on the value of the teacher's Rating on the platform;
   - moduleNumber — number of modules in the course.

   Automatic calculation of the course cost has the following advantages:

   - Objectivity: Calculating costs based on specific factors such as course type, length and teacher rating makes the cost setting process more objective and reasonable.
   - Stimulate improvement in teaching quality: Considering teacher ratings when calculating the cost can incentivize teachers to improve the quality of their courses and receive higher payment.

3) Next, the teacher clicks "Create course". At the same time, he must pay the fee to the wallet of the platform. The transaction parameters (recipient address, payment amount, etc.) are checked by the smart contract to ensure that it is correct.

The process of creating a course by a teacher is described in detail in the sequence diagram (Pic. 2.3).

4) The Student enrolls in the course. At the same time, the smart contract checks whether the course is still available for recording. If yes then the student must pay the cost of the course for at least the first module (10% of the funds go to the reserve, the remaining funds go to the Reward Account).

   The cost of one module is calculated using the formula:

   moduleCost = courseCost/ moduleNumber (5)

   where:

- courseCost — cost of the entire course, calculated when it was created by formula (4);
- moduleNumber — number of modules in the course.

The process of registering a student for a course is described in detail in the sequence diagram (Pic. 2.4).

5) Testing takes place at the time set by the teacher (for example, at the end of each month). In accordance with the scores received, students are awarded a scholarship to their account (from the Reward Account).

   Student success level coefficients:

   1. Excellent —10
   2. Good — 8
   3. Not Bad — 6
   4. Bad — 4
   5. Very Bad — 2

Thus, the amount of the scholarship for the student is calculated using the following formula (2):

scholarship= moduleCost*successCoef (6)

where:

- moduleCost — the price for one module of the course. Calculated by formula (5);

- successCoef — student success level coefficient.

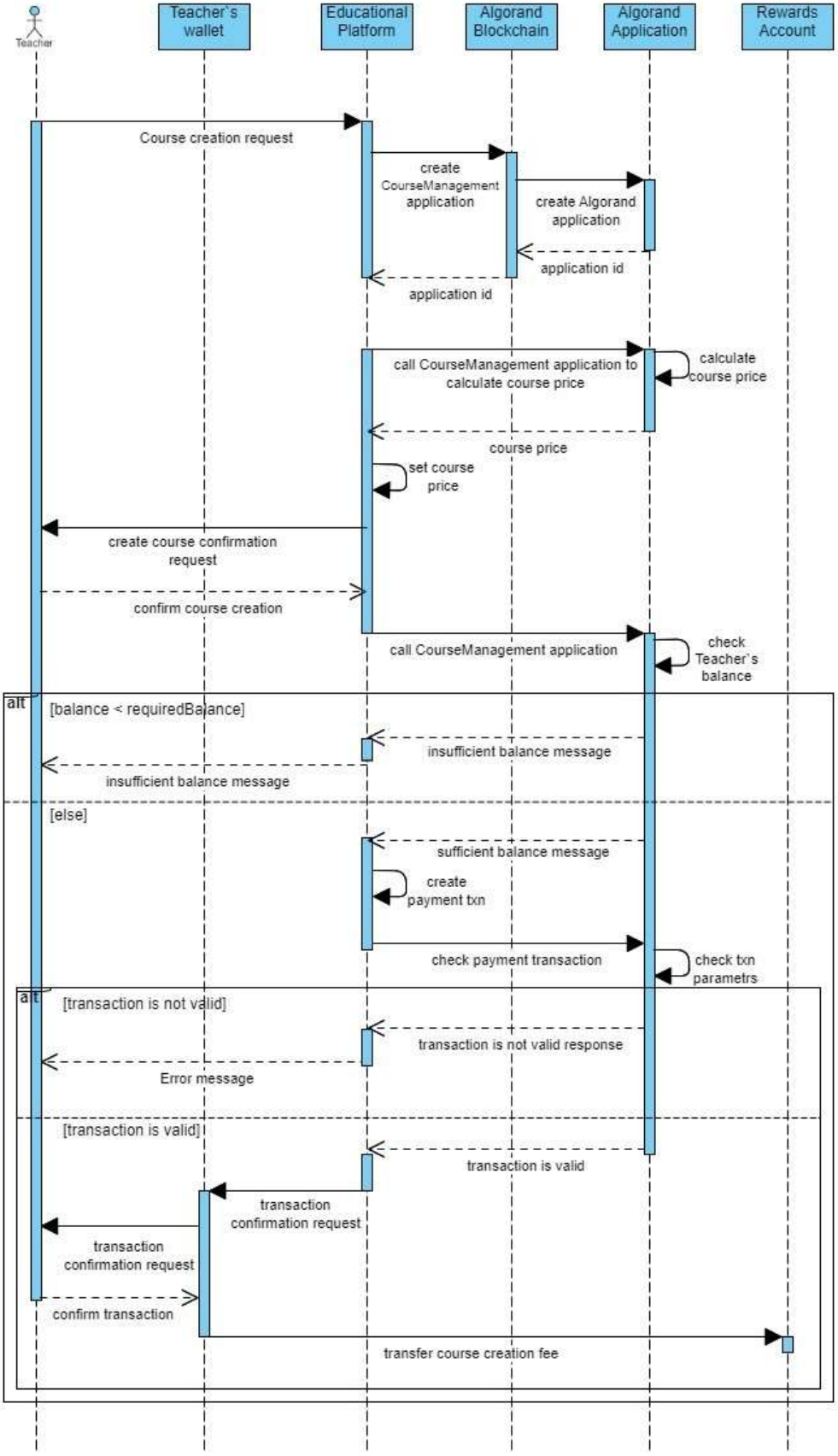
The current score of the student and the amount of his scholarship are stored in the Blockchain.

The advantage of storing scores on the blockchain is that this technology is a reliable mechanism that guarantees the integrity and objectivity of the data. Therefore, scores stored on the blockchain cannot be changed or falsified. This approach also helps to simplify administrative processes and reduce bureaucracy.

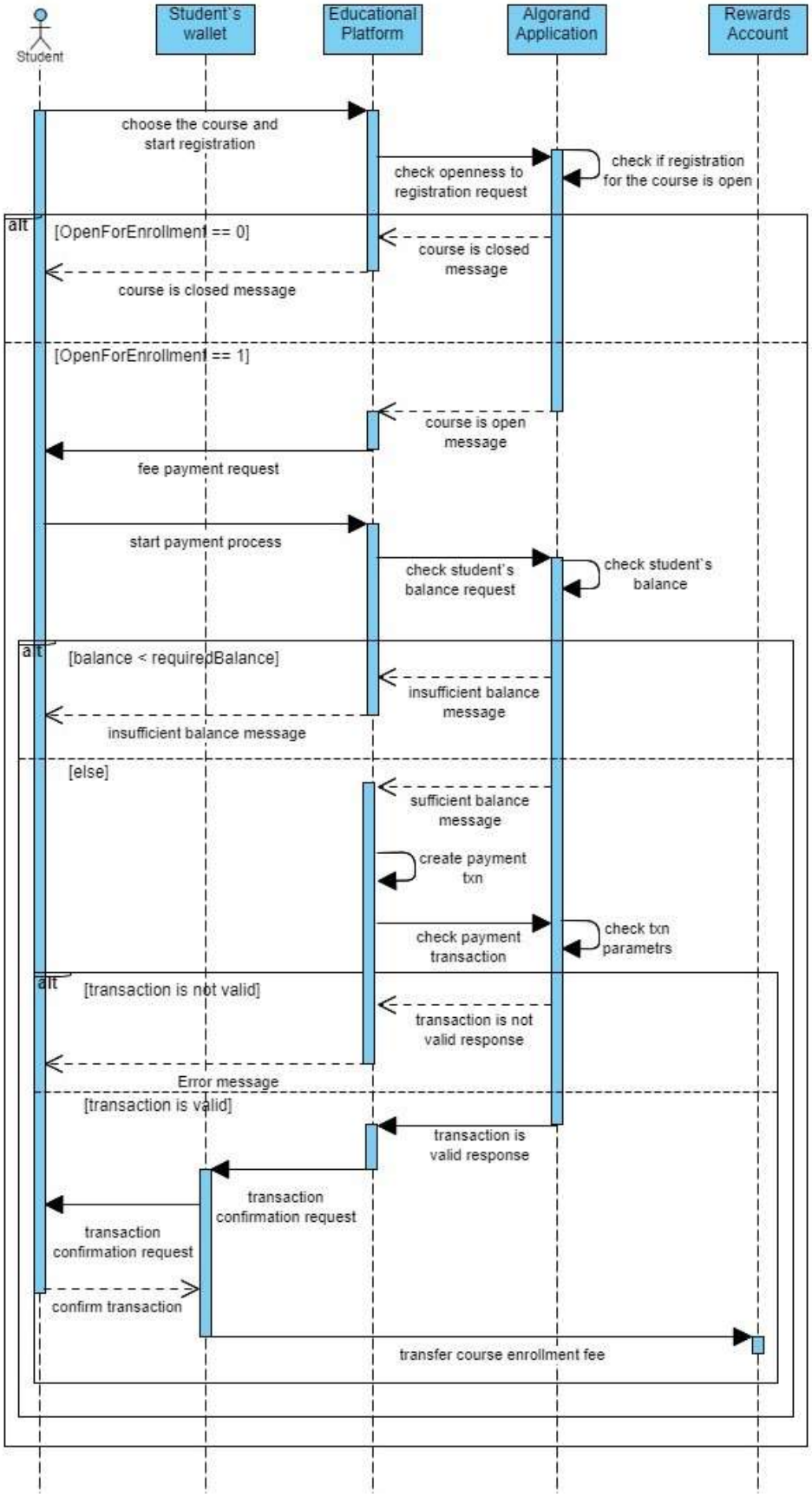
6) The teacher will receive a monthly payment from the Reward account in the amount which is calculated by next formula:
   teacherReward = countStudent*moduleCost*0.9 (7)
   where:
   - countStudent — number of students who have registered for this course;
   - moduleCost — the price for one module of the course.
7) At the end of the course, students leave feedback about the teacher. Based on their assessment, the teacher's Rating on the platform is recalculated. When recalculating the Rating, the type of course and duration are also considered.
8) Students also receive the reward for giving feedback.

The full process from course creation to completion is shown in the diagram (Pic. 2.5).

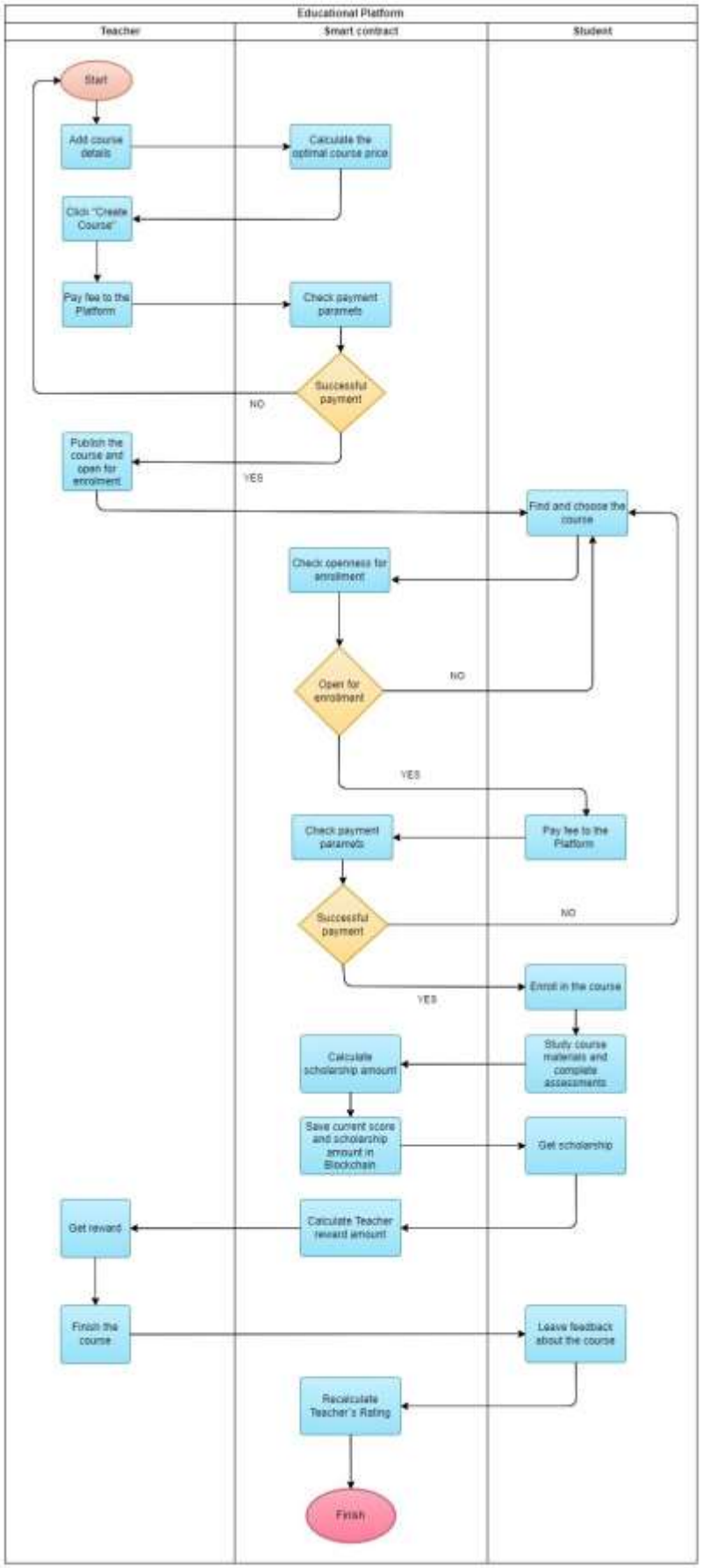
It is worth noting that payments on the platform will be made using the platform's own token, which will be implemented as an Algorand standard asset [14].



Pic. 2.3 — Course creation by Teacher



Pic. 2.4 — Process of student registration for a course



Pic. 2.5 — Educational platform flow diagram

2.2 Tools for smart contracts implementation

In this section, we will consider the tools that will be used to develop smart contracts for the educational platform.

As a blockchain for the implementation of the educational platform, we have chosen Algorand [15]. It was introduced in 2017 by MIT professor Silvio Micali [16]. Algorand is a high performance blockchain platform with fast and low-cost transactions powered by its own consensus algorithm — Pure Proof-of-Stake (PPoS). It uses its own virtual machine Algorand Virtual Machine (AVM). This blockchain supports smart contracts, decentralized applications, and the issuance of digital assets. Algorand uses its native cryptocurrency Algo.

2.2.1 Algorand Smart Contracts

Algorand Smart Contracts (ASC1) are small programs that perform various functions on the blockchain and operate at layer 1 [17]. Algorand supports two types of smart contracts: stateful and stateless (smart signatures). Stateful smart contracts allow us to store data in global and local storage. Also, the contract code is stored on the blockchain network and can be viewed at any time. The smart contract itself, after being deployed on the network, is called an application and has its own id. The smart signature is sent to the blockchain with the transaction. Its logic accepts or rejects the transaction.

The main differences between smart contracts and smart signatures in Algorand are listed below in Table 2.2.

Table 2.2

Difference between smart signatures and smart contracts in Algorand Blockchain

	Smart signatures	**Smart contacts**
Purpose of use	Have two use cases: signature authority delegation and contract accounts (escrow account)	Used to create complex decentralized applications that can manage assets, perform fund transfers, verify transaction parameters etc.
Principle of operation	As delegated authority: As escrow account: the compiled program has its own unique address and can function as an Algorand account. Users can transfer funds to it, as well as withdraw	The user creates an application call transaction and sends it to the network. The corresponding smart contract code processes the transaction, either accepting or rejecting it
Access to network data	Does not have access to information about blocks and transactions	Smart contracts are executed on the blockchain, have access to transaction information and blockchain state such as balances and contract state
Complexity	Small-sized programs of low complexity	Programs of high complexity, may contain

		branches, loops, conditional statements, etc.
Size	1000 bytes	1 kb

Thus, smart contracts are aimed at automating complex agreements and operations, while smart signatures are aimed at ensuring the secure execution of transactions on the Algorand network.

2.2.2 TEAL as language for Algorand Smart Contracts

Smart contracts and smart signatures for the Algorand blockchain are written in the language that is called TEAL (Transaction Execution Approval Language). It is a low-level, assembly-like language that is interpreted by the Algorand Virtual Machine (AVM) [17]. TEAL programs are processed line by line, pushing and popping values on the stack. TEAL supports a limited set of data types. These can be bytes or unsigned 64-bit integers. TEAL provides a set of operators that operate on those values on the stack [18]. TEAL has more limited potential functionality such as no support for recursive logic, however this makes smart contracts safer to write and execute.

A smart contract written in TEAL can be compared to a class in object-oriented programming. Then the application that is created and resides on the blockchain can be compared to an instance of a class.

As we already mentioned, TEAL is a stack-based language. This means that the program processes all actions requested by the transaction from which it was called if and only if the last value on the stack is 1. If so, the TEAL program returns true, and the transaction is processed. For every other value left on the stack, false is returned and the transaction will fail [18].

TEAL is a restricted computing language. It is designed to perform simple checks on the status of a transaction, not complex calculations.

Each operation in the TEAL program has an associated cost that counts towards the total cost of executing the program. Smart contracts are given a total opcode budget of 700. If the program exceeds this budget, the application call will fail. This approach allows the Algorand blockchain to set a fixed fee per application call, rather than charging a fee based on the computational cost of calling a smart contract.

2.2.3 Algorand JS SDK

For interaction of the platform with the Algorand blockchain, Algorand JS SDK is used. The Algorand JavaScript Software Development Kit [19] is a set of tools, libraries and functions that help developers interact with the Algorand blockchain network using the JavaScript programming language.

This SDK provides the following key functionalities for interaction with the blockchain:

- Account creation and management. Algorand is an account based blockchain platform. The SDK provides tools for creating and managing accounts on the Algorand blockchain. SDK also allows users to get information about the account (balance, account assets, etc.)
- Transaction creation. SDK allows users to create transactions and send them to the Algorand network. These can be transactions of transferring funds to other accounts, making applications (smart contracts) calls and others.
- Interaction with smart contracts. SDK makes it possible to create applications, trigger smart contract methods, pass arguments to the application, and so on. This is done using the application creation and application call transactions.
- Getting information about the current state of the blockchain. The SDK can be used to obtain information about the current state of the Algorand

blockchain, including information about blocks, transactions, and accounts.

In the next section we will look in detail at the features of developing smart contracts on the Algorand blockchain using the TEAL language.

Conclusions to the chapter:

In this section, we defined the main roles of users on the educational platform and the actions of each of them. It should be noted that attention was paid to those actions that will be regulated by smart contracts. The process of creating and placing a course on the platform was described in detail.

We have also defined the platform and tools for developing smart contracts for the educational platform. Thus, Algorand was chosen as the blockchain platform. We considered the features of Algorand smart contracts, as well as their varieties: stateless and stateful smart contracts, and the main differences between them. We considered the features of the programming language for the development of Algorand smart contracts — TEAL (Transaction Execution Approval Language) — low-level, assembly-like language.

CHAPTER 3

IMPLEMENTATION AND AUDIT OF SMART CONTRACTS

3.1 Implementation of smart contracts for the educational platform

In this section, we will take a detailed look at the process of creating smart contracts and sending them to the network. We will also describe the process of auditing smart contract code for the educational platform.

In the process of developing smart contracts, the following main stages can be distinguished:

1. Define, document and analyze the requirements for smart contracts that need to be developed.
2. Writing smart contracts code.
3. Verification of smart contract code using special tools.
4. Correcting errors and eliminating vulnerabilities discovered during code verification.
5. Testing the correct execution of smart contracts.
6. Deployment of smart contracts in the blockchain network.

3.1.1 Development of the Smart contract for course management

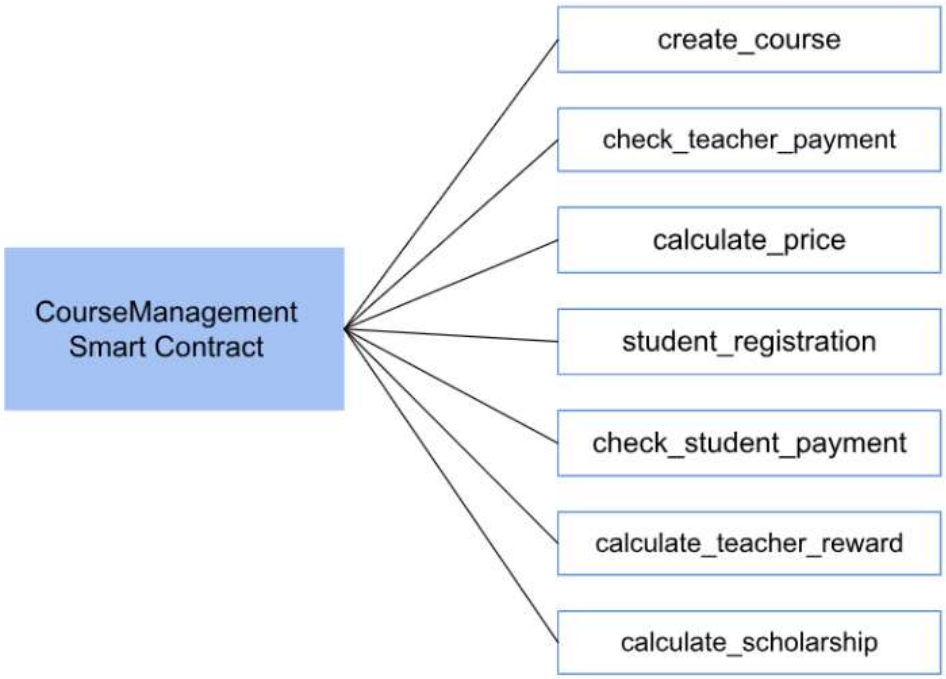
Now we will look at the implementation of the stateful smart contracts and their interaction with the client application through the SDK. In this project, we will use JavaScript SDK for Algorand.

The Smart Contracts code consists of the ApprovalProgram and the ClearStateProgram.

- ApprovalProgram code defines the logic and conditions for approving or rejecting transactions that trigger a contract.
- ClearStateProgram code is used to define the logic that is executed at the end of the contract's lifecycle.

In the ApprovalProgram, we define variables that will be stored in global storage. It should also be noted that the global state of stateful smart contract is limited to 64 key-value pairs, and the local state is limited to 16 key-value pairs for each individual account that interacts with it. The size of each key/value pair is limited to 128 bytes.

CourseManagement Application will regulate the basic processes of creating and managing courses.



Pic. 3.1 — CourseManagement application methods

Next, we specify the key-value-pairs that the CourseManagement application contains in its global and local storages.

Global State Schema for CourseManagement Application:

- CourseTitle — the course name specified by the teacher during course creation.
- Type — type of course: asynchronous (without video, only text materials, presentations, etc.), asynchronous (with video materials from the teacher),

synchronous (online video lessons with students). Also specified by the teacher during course creation.

- ModulesNumber — number of the modules in the course.
- EnrollmentLimit — maximum number of students who can enroll in a course.
- OpenForEnrollment — indicates whether the course is available for enrollment of new students. The value is checked when new students are recruited (1 – registration is open, 0 – registration is closed).
- EnrolledStudentsNumber — number of students who have enrolled in the course.
- CoursePrice — the cost a student must pay to take a course.
- TypeCoefficient — coefficient, which depends on the project type. It is used when calculating project cost.
- DurationCoefficient — coefficient, which depends on the project duration. It is also used when calculating project cost.
- RatingCoefficient — coefficient, which depends on the Teacher`s Rating value.

Local State Schema for CourseManagement Application:

- CurrentScore — the current value of the student's success results. The indicator is used to calculate the amount of the student's scholarship.
- Scholarship — the current amount of scholarship that the student will receive per month.

It's worth noting that these schemas are immutable after creation.

- **Minimum balance requirements**

Creating new applications increases the minimum account balance requirements. This minimum balance is necessary to cover the storage cost for

the TEAL application state, which includes the storage of global and local variables, as well as any other data associated with the application.

Global storage is actually stored in the creator account, so that account is responsible for the global storage minimum balance.

The minimum balance increases with each asset contained in the account (regardless of whether the asset was created or owned by the account), and with each application created or registered in the account. The minimum balance requirement is reduced when the application created by this account is removed from the network.

Thus, when creating this application, the minimum balance requirements for the account that creates it will increase as follows:

- 100,000 microAlgos — base fee for each page requested.
- 25,000 + 3,500 = 28,500 for each Uint variable in the global storage = 10*28 500 = 285 000 microAlgos.

The creator of the Application would have its minimum balance raised by 385 500 microAlgos = 0.3855 Algo.

Algorand smart contracts interact with the backend of the platform using transactions and application calls. Application calls allow external entities to invoke specific functions or methods defined within the smart contract. These calls can pass arguments, trigger specific actions or calculations within the contract, and return results or updated state.

- **Application Creation**

We define global storage variables when creating an application. In TEAL code this happens as follows:

```
#pragma version 6
//If app id == 0, must be creation call
txn ApplicationID
int 0
==
bz not_creation

byte "CourseTitle"
txna ApplicationArgs 0
app_global_put

byte "Type"
txna ApplicationArgs 1
app_global_put

byte "ModulesNumber"
txna ApplicationArgs 2
btoi
app_global_put

byte "EnrollmentLimit"
txna ApplicationArgs 3
btoi
app_global_put

byte "EnrolledStudentsNumber"
int 0
app_global_put

byte "OpenForEnrollment"
int 1
app_global_put

byte "CoursePrice"
int 0
app_global_put

byte "TeacherRating"
txna ApplicationArgs 4
btoi
app_global_put

byte "TypeCoefficient "
int 0
app_global_put

byte "DurationCoefficient"
int 0
app_global_put

byte "RatingCoefficient"
int 0
app_global_put

b done
```

Listing 3.1 — Definition of global variables during application creation

Thus, at the beginning we check that ApplicationID is 0. If this statement is correct, then we move on to the part of the code that is responsible for creating the Application and allocating global state variables.

In this code snippet, the values of the "CourseTitle", "Type", "ModulesNumber", "EnrollmentLimit", "CoursePrice" and "TeacherRating" variables are set by the application creator as arguments when creating the CourseManagement Application. When creating the application, we set the value of the "OpenForEnrollment" variable to 1. This means that the course is open for student enrollment. "EnrolledStudentsNumber" variable default value is 0. The default values of the coefficient variables are also set to 0.

After that, we move to a label called “done”, which will return and approve the application creation transaction.

- **ApplicationCall transactions**

From the backend, we construct an application call transaction using the Algorand SDK.

There are several types of ApplicationCall transactions:

- NoOp: the most used transaction type. It allows the developer to make application calls to execute the ApprovalProgram.
- OptIn: this transaction allows the user to start participating in the smart contract (application). Allows the user to use the application's local state.
- DeleteApplication: this operation is used to remove a smart contract (application) from the Algorand network blockchain. Once this transaction is completed, the smart contract will no longer be available.
- UpdateApplication: this transaction allows the creator to make changes to the code, parameters, or other aspects of the smart contract after it has been deployed. This can be useful in cases where the contract needs to be updated or modified to fix bugs, ensure security, or extend the functionality of the contract.

- CloseOut: with this transaction the user can stop using the application. This will lead to removing the application's local state from the user account.
- ClearState: this transaction allows the user to clear the local state of the application, even if the application was deleted by the creator.

All transaction types except DeleteApplication are part of the ApprovalProgram. In our contract, we check the transaction type and jump to the appropriate branch depending on the type. Next, we are giving the snippet of TEAL code for jumping between program branches:

```
txn OnCompletion
int OptIn
==
bnz handle_optin

txn OnCompletion
int NoOp
==
bnz handle_noop

…
```

Listing 3.2 — Jumping between program branches

Operation “bnz target” — jump to TARGET if the last element of the stack is non-zero.

“handle_optin” branch is called when the ApplicationCall type is OptIn. This occurs when the student enrolls in the course since the user account that wants to use its local state for the application must subscribe to the smart contract.

```
handle_optin:
byte "OpenForEnrollment"
int 1
==
bz failed

int 0
byte "CurrentScore"
int 0
app_local_put

int 0
byte "Scholarship"
int 0
app_local_put

byte "EnrolledStudentsNumber"
app_global_get
int 1
+
dup
store 0
byte "EnrolledStudentsNumber"
load 0
app_global_put

byte "EnrolledStudentsNumber"
app_global_get
store 1
byte "EnrollmentLimit"
app_global_get
load 1
==
bnz close_enrollment
b done
```

Listing 3.3 — OptIn ApplicationCall branch

In this snippet, we check if there are places available to enroll in the course (OpenForEnrollment == 1), allocate variables "CurrentScore" and

"Scholarship" and initialize them with default values. We also increment the value of the "EnrolledStudentsNumber" variable and check if the course is closed for enrollment of new students (EnrolledStudentsNumber == EnrollmentLimit). Then we set the “OpenForEnrollment” value to 0.

NoOp type of the ApplicationCall transactions forms the main part of the smart contract Approval program. The CourseManagement Application contains 6 methods that can be executed depending on the argument passed to the transaction:

1) set_coefficient: method for setting course coefficient values depending on its type, duration and Teacher rating. AppCall transaction argument — "SetCoefficient" of byte type.
2) calculate_price: course price calculation method. AppCall transaction argument — "CalculateCoursePrice" of byte type.
3) evaluate_student: method that is called when the teacher evaluates the student. AppCall transaction argument — "EvaluateStudent" of byte type.
4) calculate_teacher_reward — teacher reward amount calculation method. AppCall transaction argument — "CalculateTeacherReward" of byte type.
5) check_teacher_payment: method that is called when it is necessary to verify the validity of the transaction created when a teacher pays a commission to the platform for creating the course. AppCall transaction argument — "CheckTeacherPayment" of byte type.
6) check_student_payment: method that is called when it is necessary to verify the validity of the transaction created when a student pays a commission to the platform for enrolling in the course. AppCall transaction argument — "CheckStudentPayment" of byte type.

The following is a “handle_noop” snippet:

```
1  handle_noop:
2  txna ApplicationArgs 0
3  byte "CalculateCoursePrice"
4  ==
5  bnz calculate_price_with_coef
6
7  txna ApplicationArgs 0
8  byte "EvaluateStudent"
9  ==
10 bnz evaluate_student
   …
   err
```

Listing 3.4 — NoOp ApplicationCall branch

The cost of the course is calculated (according to formula (4), described in the previous section) using the TEAL code as follows:

```
1       calculate_price:
2       byte "TypeCoefficient"
3       app_global_put
4       int 100
5       *
6       dup
7       store 0
8       byte "DurationCoefficient"
9       app_global_get
10      load 0
11      *
12      dup
13      store 1
14      byte "RatingCoefficient"
15      app_global_get
16      load 1
17      *
18      dup
19      store 2
20      byte "CoursePrice"
21      load 2
22      app_global_put
23      b done
```

Listing 3.5 — Course cost calculation

To validate a payment transaction, we can use Algorand's atomic transactions, which allows multiple transactions to be sent at the same time, and if any of the transactions fail, then they all fail. To check the parameters of payment transactions created in the JS SDK, we can group this transaction with the stateful smart contract call and send them simultaneously. In TEAL we check values of the transaction in the group as follows:

```
1  check_teacher_payment:
2  global GroupSize
3  int 2
4  ==
5  assert
6
7  gtxn 0 AssetAmount
8  int 50
9  ==
10 assert
11
12 gtxn 0 XferAsset
13 int 106326517
14 ==
15 assert
16
17 gtxn 0 AssetReceiver
18 addr
   2ERPXALR7IESFS6FWX45CDA4Z3447SPNMC6YQHF2UXOXRRNLX5KZAMT4DI
19 ==
20 assert
21
22 gtxn 0 RekeyTo
23 global ZeroAddress
24 ==
25 assert
26 b done
```

Listing 3.6 — Checking payment transaction parameters

In this example, we checked the parameters of the transaction that is created when a teacher pays a commission to the platform for creating a course.

We checked that the transaction amount is equal to the established commission amount, the address of the recipient of the funds is the platform address, and the asset ID corresponds to the platform token id that is used for payment.

In the backend, we can analyze the transaction result to determine if the smart contract executed successfully and to extract any relevant data or state changes resulting from the contract's execution.

When a teacher evaluates a student, the student's score is saved in the application's local state. The amount of the scholarship for a given student is also calculated. The scholarship is paid to the student from the Rewards account. Thus, the CourseManagement smart contract must make an inner transaction call to the application, which is responsible for distributing rewards (it is also an escrow account).

Calculation of the amount of the student's scholarship according to the formula (6) on the TEAL code is as follows:

```
1    calculate_scholarship:
2    byte "CoursePrice"
3    app_global_get
4    store 0
5    load 0
6    byte "ModulesNumber"
7    app_global_get
8    /
9    dup
10   store 1
11   int 0
12   byte "CurrentScore"
13   app_local_get
14   load 1
15   *
16   dup
17   store 2
18   int 0
19   byte "Scholarship"
20   load 2
21   app_local_put
```

Listing 3.7 — Calculation of student scholarship amount

Creating the inner transaction to call the Rewards application from CourseManagement smart contract is done as follows:

```
 //call reward smart contract
1   itxn_begin
2
3   int appl
4   itxn_field TypeEnum
5
6   txn Applications 1
7   itxn_field ApplicationID
8
9   int NoOp
10  itxn_field OnCompletion
11
12  byte "PayStudentScholarship"
13  itxn_field ApplicationArgs
14
15  int 0
16  byte "Scholarship"
17  app_local_get
18  itxn_field ApplicationArgs
19
20  itxn_submit
21  b done
```

Listing 3.8 — Inner transaction to call the Rewards application

As arguments when calling the Rewards application, we pass the name of the method that needs to be called in the smart contract, and the amount of the scholarship that needs to be paid to the student.

In this example, we are calling one smart contract from another. It should be noted that “Contract-to-Contract Calling” has some features and limitations [31]:

- Contract A can call Contract B, which calls Contract C, etc., but call depth is limited to 8.

- The number of inner transactions is limited to 256 per transaction group. This includes both the inner transaction of contract A that calls contract B, and those inner transactions that contract B can execute.
- The fee for inner transactions, as well as for any other transactions in Algorand, is 0.01 Algo. If smart contract A makes a call to smart contracts B and C, the commission amount will be equal to 0.3 Algo.
- Smart contract A cannot make a call to smart contract B if its code contains a call to smart contract A. This is done in order to avoid "re-entrancy" vulnerability.

When the application is being deleted, the `handle_deleteapp` method is triggered.

```
1 handle_deleteapp:
2 txn Sender
3 global CreatorAddress
4 ==
5 bz failed
6
7 failed:
8 int 0
9 return
```

Listing 3.9 — DeleteApplication ApplicationCall branch

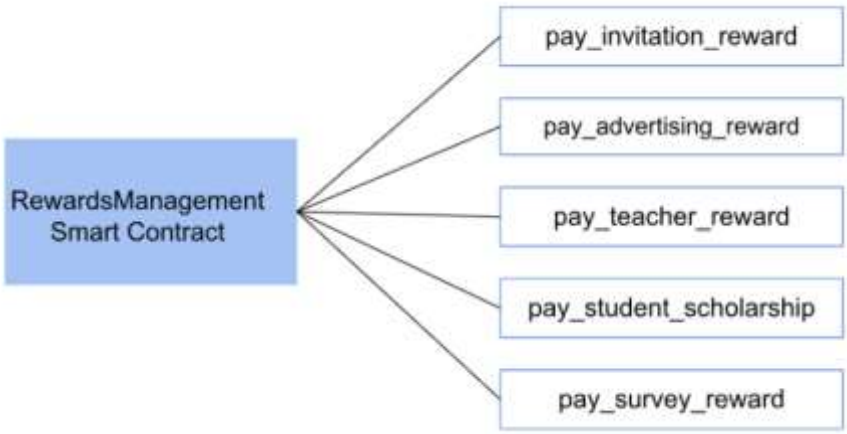
If the transaction sender address does not match the application creator address, the application call will fail. This approach allows only the app creator, such as the course creator, to delete the application. The smart contract logic will reject the DeleteApplication transaction if it is submitted by any other user.

3.1.2 Development of the Smart contract for rewards distribution

The Reward account will be implemented as a smart contract. Once deployed, the smart contract has its own address and can function as an escrow account.

An escrow account is an account in which funds are locked until some predetermined event occurs or a certain set of conditions are met. The conditions that determine when funds must be sent are encoded and thus enforced by the logic of the contract account itself. This eliminates the need for a centralized authority to determine whether a condition has been met and then moderate the transaction [20].

Thus, all payments from users will be credited to this account. Next, the smart contract will have methods for creating inner transactions for paying scholarships to students and awards to teachers.



Pic. 3.3 — RewardsManagement smart contract methods

- **Reward payment for inviting a new participant**

When an already registered user invites a new user to the platform, and the latter enrolls in any course (student) or creates his own course (teacher), the

inviting user can receive a fixed reward. Then the RewardsManagement smart contract is called, in which the `pay_invitation_reward` method is triggered.

```
1       pay_invitation_reward:
2       itxn_begin
3
4       int axfer
5       itxn_field TypeEnum
6
7       int 106326517
8       itxn_field XferAsset
9
10      txn Accounts 1
11      itxn_field AssetReceiver
12
13      int 20
14      itxn_field AssetAmount
15
16      int 0
17      itxn_field Fee
18
19      itxn_submit
```

Listing 3.10 — Inner transaction for paying rewards to users for inviting a new user

This method creates the inner transaction that transfers a fixed amount of the asset from the Rewards account to the user's account.

The parameters of this transaction are: the type of transaction (in this case, it is `axfer` — asset transfer transactions), the id of the asset being transferred (`106326517` — id of the platform payment token), the address of the recipient of the asset (`txn Accounts 1` — the address is passed as an argument to the application call transaction in Algorand JS SDK), the amount of the asset being transferred.

- **Payment of the students scholarship**

When the student's scholarship amount is calculated in the CourseManagement application, it creates the inner transaction that makes a call

to the RewardsManagement application and triggers the `pay_student_scholarship` method.

```
1       pay_student_scholarship:
2       itxn_begin
3
4       int axfer
5       itxn_field TypeEnum
6
7       int 106326517
8       itxn_field XferAsset
9
10      txn Accounts 1
11      itxn_field AssetReceiver
12
13      txna ApplicationArgs 1
14      itxn_field AssetAmount
15
16      itxn_submit
17
18      b done
```

Listing 3.11 — Inner transaction to pay the scholarship to the student

3.1.3 Creating test accounts

In order to be able to create an application on the Algorand network and interact with a smart contract, we need to have test accounts that must have a certain amount of Algo on their balance in order to be able to pay transaction fees. It is possible to create such a test account using Algorand SDK with the following code:

```
    const generatedAccount = algosdk.generateAccount();
    const passphrase =
algosdk.secretKeyToMnemonic(generatedAccount.sk);
```

Listing 3.12 — Creating an account using algosdk

To finance a test wallet, we can transfer funds from another funded wallet or use the Algorand dispenser [21].

To make it easier to test smart contracts on the Testnet, we can use the private account key for an Algorand account to retrieve account information from the blockchain or sign transactions. To work with assets on the Mainnet, it is necessary to create an account using a trusted wallet, for example Pera wallet [22] for Algorand. This will allow users to manage their accounts and assets safely and correctly. Since the platform being developed will use its own token, it is also necessary to set up accounts for this asset.

To be able to receive an Algorand asset, the user must “opt-in” to receive it by sending a 0 amount of the ASA to himself (to the account that will receive the asset). This approach protects users from spam assets, preventing unknown assets or assets that are not whitelisted from being sent to the user without his approval.

3.1.4 Deployment of smart contracts

To test smart contracts, we can use the Algorand test network — Testnet. In it developers can perform the same operations as in the real network (create own assets, transfer funds, create and call applications, etc.), but without having to spend real Algos and pay transaction fees.

Once deployed, the instance of the smart contract on the network is called an application and is given an application ID. Additionally, every smart contract has a unique Algorand address that is generated from this specific ID.

- **CourseManagement SC**

To deploy the CourseManagement smart contract in the network, *makeApplicationCreateTxn* is created:

```
    const arg0 = Encodeuint8arr('CourseTitle');
    const arg1 = Encodeuint8arr('Asynchronous with video
material');
    const arg2 = algosdk.encodeUint64(3); //ModulesNumber
    const arg3 = algosdk.encodeUint64(20); //EnrollmentLimit
    const arg4 = algosdk.encodeUint64(teacherRating);
```

```
    const                          from                          =
"2ERPXALR7IESFS6FWX45CDA4Z3447SPNMC6YQHF2UXOXRRNLX5KZAMT4DI";
      const onComplete = algosdk.OnApplicationComplete.NoOpOC;
    const           approvalProgram           =           await
getBasicProgramBytes('CreateCourse.teal');
    const            clearProgram            =            await
getBasicProgramBytes('clear.teal');
    //txn params
    const numLocalInts = 3;
    const numLocalByteSlices = 0; //The numLocalByteSlices variable is set
to 0, indicating that there are no local byte slice variables required for the smart contract.
    const numGlobalInts = 10; // The numGlobalInts variable is set to 10,
indicating that the smart contract requires 10 global integer variables.
    const numGlobalByteSlices = 2;
    const appArgs = [arg0, arg1, arg2, arg3, arg4]; //The appArgs
variable is an array that contains the arguments to be passed to the smart contract during the
application call.
    // get suggested params
    const           suggestedParams           =           await
algodClient.getTransactionParams().do();

    // create the application creation transaction
    const createTxn = algosdk.makeApplicationCreateTxn(
          from,
          suggestedParams,
          onComplete,
          approvalProgram,
          clearProgram,
          numLocalInts,
          numLocalByteSlices,
          numGlobalInts,
          numGlobalByteSlices,
          appArgs
     );

    // send the transaction
    const signedCreateTxn = createTxn.signTxn(system.sk);
    const { txId: createTxId } = await algodClient
            .sendRawTransaction(signedCreateTxn)
            .do();
```

Listing 3.13 — Creating a create application transaction

The function returns the ID of the created application. Next, the ID is used for application calls. The ID can also be used to view the code of the created application, as well as the values that are stored in its global state using the blockchain explorer. One of these explorers is AlgoExplorer [23].

Pic. 3.2 — CourseManagement Application Global State in Algo Explorer

Pic. 3.3 — CourseManagement Approval Program in AlgoExplorer

- **RewardsManagement smart contract**

To create the Rewards application, we also have to create *makeApplicationCreateTxn* transaction. Once the RewardsManagement SC is

deployed, it must be funded as it functions as an escrow account and has a minimum balance requirement. To do this, we need to get the escrow account address and transfer Algo there (the minimum balance for any Algorand account is 0.1Algo). Since the platform will use its own token for payment, escrow must also be set up to work with it (opt-in the asset).

Obtaining an escrow account address using Algosdk:

```
const escrowAddress = algosdk.getApplicationAddress(appId);
```

To fund the escrow account, we create a regular payment transaction using Algorand JS SDK, where the escrow account address will be specified as the receiver:

```
let sender = system.addr;
let receiver = escrowAddress;
let suggestedParams = await
algodClient.getTransactionParams().do();
let amount = 200000;     //200000 MicroAlgo = 0.2 Algo

const txn =
algosdk.makePaymentTxnWithSuggestedParamsFromObject({
          from: sender,
          suggestedParams,
          to: receiver,
          amount: amount,
      });

const signedTxn = txn.signTxn(system.sk);
const { txId: createTxId } = await algodClient
        .sendRawTransaction(signedTxn)
        .do();
```

Listing 3.14 — Creating a payment transaction to transfer Algo to the escrow account

We are transferring 0.2 Algo because we are going to set up the escrow to work with the platform asset. For an account with two assets, the minimum balance is 0.2 Algo.

To opt-in the escrow account to the asset, we must create a payment transaction and send 0 of the asset amount to that account. It will increase the account minimum balance by 100,000 microAlgos.

```
1       opt_in_escrow:
2       // opt-in to the asset
3       itxn_begin
4
5       int axfer
6       itxn_field TypeEnum
7
8       txn Assets 0
9       itxn_field XferAsset
10
11      global CurrentApplicationAddress
12      itxn_field AssetReceiver
13
14      int 0
15      itxn_field AssetAmount
16
17      itxn_submit
18      b done
```

Listing 3.15 — Opting-in the escrow account to the platform asset

After this, any account can transfer the platform asset to the escrow account.

3.1.5 Interaction with the smart contract

To trigger the execution of the smart contract logic, we use application calls. They can include any necessary arguments or data required by the smart contract. From the backend, we create an application call transaction. Specify the application ID of the stateful smart contract and additional parameters required by the contract's logic.

When called, smart contracts on Algorand gain access to certain external resources, so it is necessary to determine which addresses, applications and

assets the contract will interact with (these are called “foreign” assets/accounts/applications).

Example of calling the ‘*CalculateCoursePrice’* method in the CourseManagement Application:

```
    const appArgs = [Encodeuint8arr('CalculateCoursePrice')];

    const suggestedParams = await
algodClient.getTransactionParams().do();

    const appForeignAssets = [106326517]; //id of the Platform token

    const appAccounts = [accAddress]; //appAccounts variable is assigned
an array containing accAddress, representing the account associated with the application.

    const foreignApps = [foreignAppId]; //foreignApps  variable is
assigned an array containing foreignAppId, representing the application associated with the
current application.

    const callTxn = algosdk.makeApplicationNoOpTxn(

                system.addr,

                suggestedParams,

                appId,

                appArgs,

                appAccounts,

                foreignApps,

                appForeignAssets

          );

const signedCallTxn = callTxn.signTxn(system.sk);

const { txId: callTxnId } = await algodClient

            .sendRawTransaction(signedCallTxn)

            .do();
```

Listing 3.16 — CourseManagement Application NoOp transaction

When a student enrolls in the course, he must also agree to interact with the application. For this purpose, the Opt-In transaction is created as follows:

```
const suggestedParams = await
algodClient.getTransactionParams().do();

const appOptInTxn = algosdk.makeApplicationOptInTxnFromObject({

    from: studentAccount.addr,

    appIndex: appId,

    suggestedParams,

  });

const signedCallTxn = appOptInTxn.signTxn(studentAccount.sk);

const { txId: callTxnId } = await algodClient

          .sendRawTransaction(signedCallTxn)

          .do();
```

Listing 3.17 — Opt-in transaction

In this case, the sender of the transaction is the student who must opt-in to the application. The local storage in the contract will be associated with his address.

Next, we will look at the main vulnerabilities that can occur in smart contracts written in TEAL. We will also check the code of the smart contracts described in this section for the presence of these vulnerabilities.

3.2 Smart contracts code audit

Smart contracts are based on program code, any error in the code leads to incorrect execution of the contract. This can be especially dangerous when it comes to financial transactions or other actions, the cost of errors of which is high enough. Errors in smart contracts can lead to large financial losses and a breach of user confidence in the platform. That is why the analysis of smart contracts is an essential part of the development of any decentralized application. To ensure the reliability, security, and correct execution of smart

contracts, it is necessary to audit the code. Verification of smart contracts allows developers to check the correctness of their code and protect them from possible errors.

A smart contract audit is a procedure that involves checking and testing the smart contract code to identify possible problems or vulnerabilities. Smart contracts audit is a rather complex and voluminous task. It includes the following stages: checking the smart contract specification, analysis of the contract code, functionality, and logic of its operation, and manual verification of the smart contract.

The purpose of smart contract auditing is to find any behavior that may have caused the code to fail or provide an unexpected result to the user. It is performed to identify potential vulnerabilities, programming errors, weaknesses, and other issues that could lead to unwanted consequences or security risks.

Some of the main reasons for the importance of smart contracts audit are:

- Security and Quality Improvement: Smart contract code review reveals vulnerabilities, bugs, and weaknesses that can be exploited by hackers to attack or gain access to assets and data, which can lead to loss of funds. Auditing improves the quality and reliability of contracts, which reduces the risk of failures, unexpected behavior, or loss of assets.
- Compliance: An audit helps to ensure that the contract operates in accordance with the requirements and specifications defined by the customer, performs the necessary operations, and ensures that the platform functions correctly.
- Transparency and trust: Smart contract code audits provide platform users with confidence in reliability, security, and compliance.

Next, we will take a closer look at the vulnerabilities that are most often found in Algorand smart contracts (Table 3.1). They are described in more detail in [24]. Also, some new vulnerabilities typical for Algorand smart contracts are highlighted in the work [25].

Table 3.1

Vulnerabilities in the Algorand smart contracts

Vulnerability	**Description**
Missing Group Size Check	If the application does not check the size of a group of transactions, attackers can add their own transactions to the group, which can lead to the loss of funds if these transactions involve the transfer of assets. Consider the example: removing lines 2-5 in Listing 3.6.
Missing Access Control	If the smart contract code does not contain checks for application calls such as UpdateApplication and DeleteApplication, an attacker can

	update the application code or completely delete it.
Missing Asset ID Verification	If the smart contract code does not check the ID of the asset that the contract interacts with (accepts, sends, etc), an attacker can manipulate the logic of the contract by passing a fake, less or more valuable asset instead of the correct asset. Consider the example: removing lines 12-15 in Listing 3.6. The attacker can then create a valid transfer transaction for more valuable assets.
Missing Inner Transaction Fee Check	If the fee amount for an inner transaction is not explicitly set in the smart contract, an attacker can create operations that perform inner transactions and burn the application balance in the form of fees. Consider the example: removing lines 16-17 in Listing 3.10.
Missing RekeyTo Parameter Check	If there is no check for the RekeyTo parameter, then an attacker can set it on his public address, and will be able to directly control the contract account assets or take over the signature account. More details about the

	possibility of Rekeying can be found in the Algorand documentation [26].
Missing Transaction Receiver Check	To validate payment transactions in Algorand, developers can use an atomic transaction group to link payment transactions and application call transactions (Part 3.1.1, Listing 3.6). If a smart transaction does not verify the receiver of a payment transaction or asset transfer transaction, an attacker can use this to specify a different address for the recipient of the transaction. Consider the example: removing lines 17-20 in Listing 3.6.
Overflow/underflow	The AVM default panics on overflows, underflows, or division by zero**.** It halts execution and fails the transaction. To prevent this error from occurring, it is possible to add restrictions on the values of variables that are involved in operations that can possibly result in the overflow or underflow.

There are different approaches to analyzing smart contract code, including:

- Formal verification — method that uses mathematical proof and logic to verify the correctness of a smart contract. When implementing smart contracts, formal verification can prove that the business logic of the contract satisfies the predefined specification. This is done by creating formal specifications that describe the characteristics of a smart contract and checking the compliance of the formal model with the smart contract specification. The specifications must include all the properties of the contract and define how it should operate under different conditions. Formal verification uses many approaches. Theorem Proving, Model Checking, and Runtime Verification are widely used in the context of smart contracts [30].
- Symbolic execution is one of methods used for formal verification that allows developers to automatically explore all possible branches of program execution using symbolic values instead of specific inputs. These values determine which paths should be executed. In the context of smart contracts, this method allows developers to explore all possible paths and conditions in the contract code to discover potential vulnerabilities or misbehavior without actually executing the contract on the blockchain network. This method also makes it possible to detect parts of the code that are not used in the smart contract (Dead Code) and can be potentially vulnerable.
- Static analysis — method that checks the source or bytecode of a smart contract before execution. Static analyzers can detect common vulnerabilities in smart contracts.

In this work, to audit the code of smart contracts, we will use Tealer — TEAL static analyzer with a set of vulnerability detectors for fast contract verification [27]. It analyzes the Teal program and creates its CFG (Control

Flow Graph). The analyzer comes with a set of vulnerability detectors and printers, allowing developers to quickly review contracts.

Tealer printers provides following functionality:

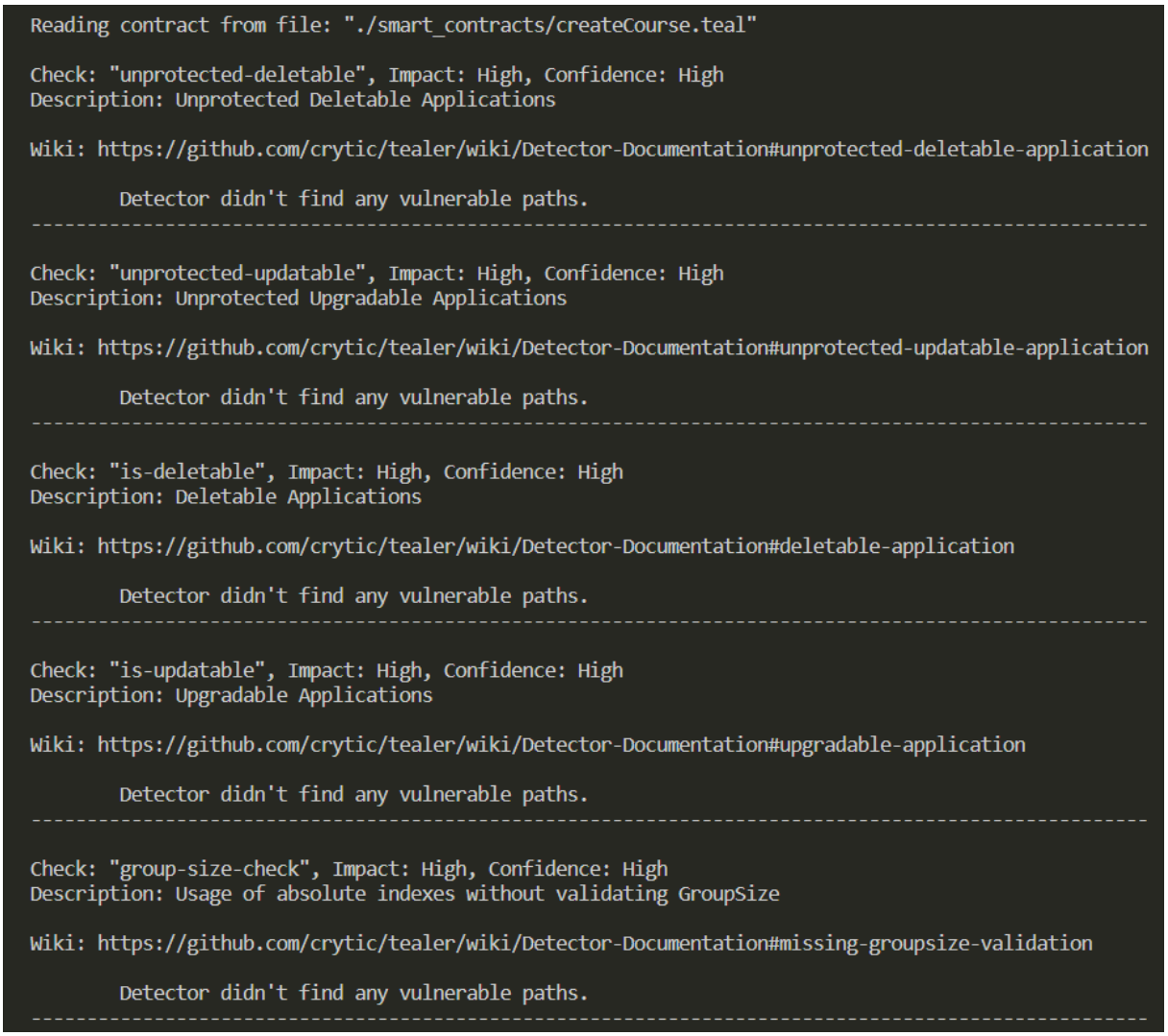
- cfg: Export the CFG of the contract to a dot file;
- human-summary: Print a human-readable summary of the contract;
- function-cfg: Export the CFG of each subroutine in the contract;
- call-graph: Export the call-graph of the contract to a dot file [27].

Tealer provides following detectors:

- is-deletable: check if the stateful application can be deleted by sending an DeleteApplication type application call;
- is-updatable: check if the stateful application can be updated by sending an UpdateApplication type application call;
- unprotected-deletable: check if the stateful application can be deleted by anyone;
- unprotected-updatable: check if the stateful application can be updated by anyone;
- group-size-check: check missing GroupSize validation;
- can-close-account: check missing AssetCloseTo field validation [28].

Thus, using this tool, we can check our smart contract for some of the vulnerabilities described in the Table 3.1, namely: missing Access Control while updating or deleting the application, missing Group Size Check.

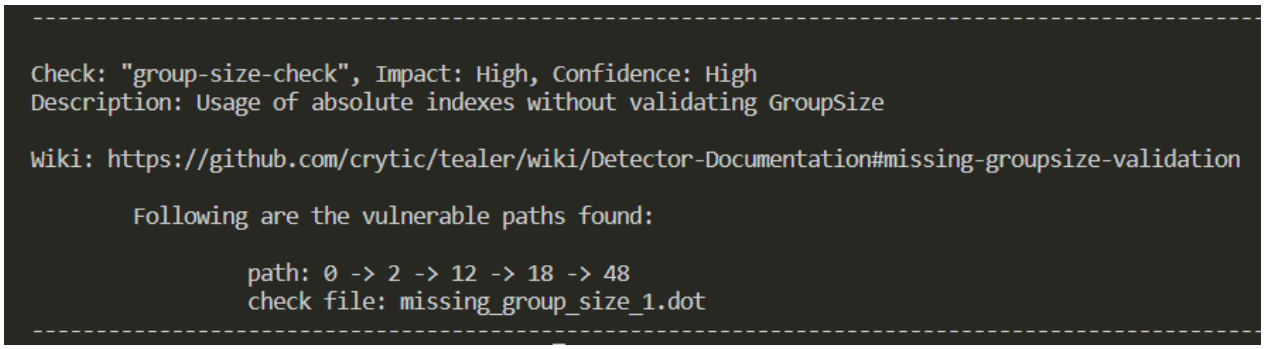
Below is the result of checking the CourseManagement smart contract using the Tealer tool (Pic. 3.4).



Pic. 3.4 — The result of checking the CourseManagement smart contract using the Tealer for vulnerabilities

As we can see, these vulnerabilities were not found in the smart contract.

If the smart contract code contains some vulnerabilities, the Tealer tool will show the vulnerable path. We deliberately removed transaction group check when checking payment transaction parameters (Listing 3.6) to show the vulnerable path detected by the Tealer in the smart contract (Pic. 3.5).



Pic. 3.5 — Vulnerable path checking payment transaction parameters

To explore this path, we can build the CourseManagement smart contract data flow graph using the “print-cfg” printer of Tealer tool.

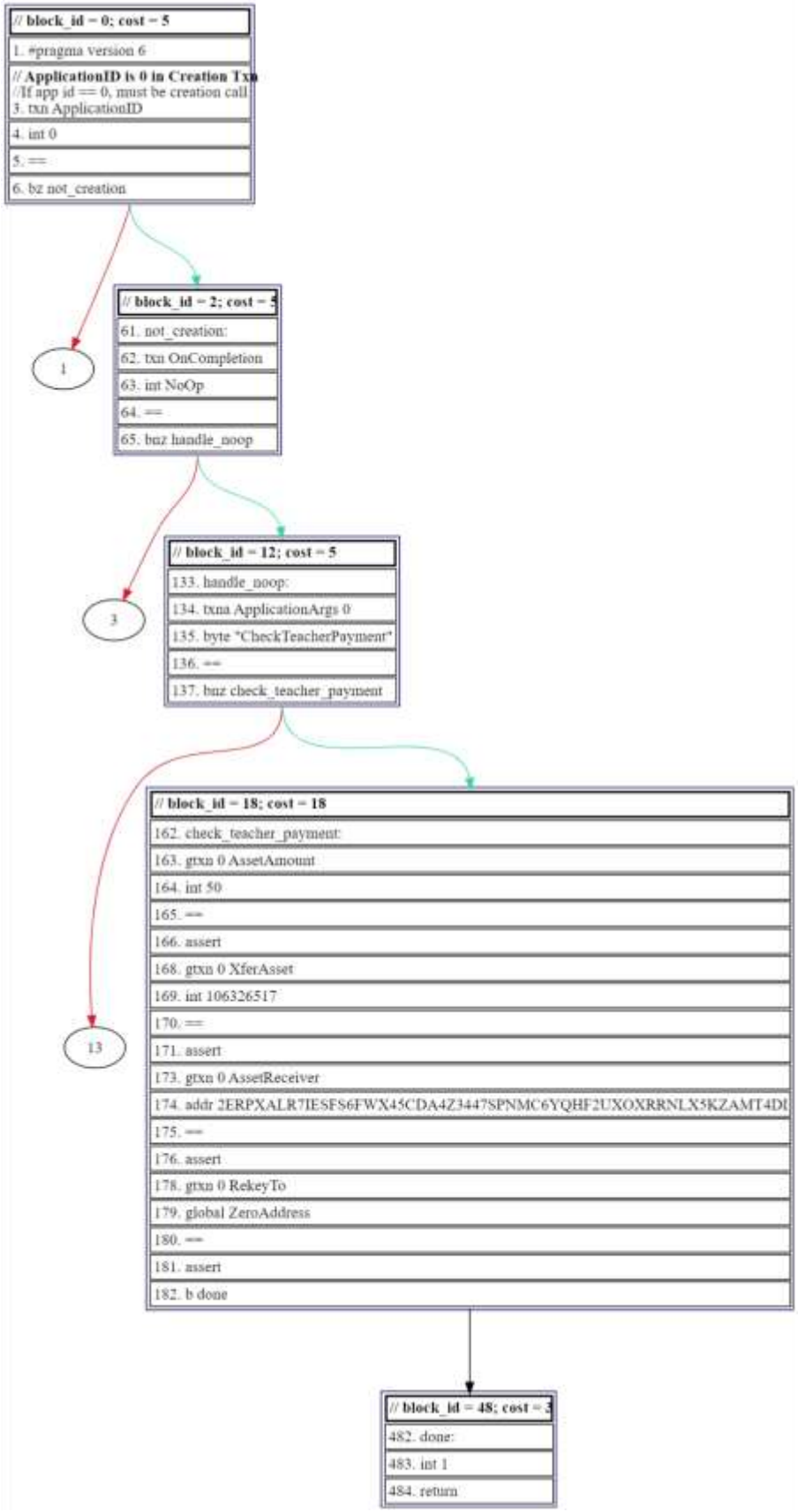
CFG provides a visual representation of the control flow within a smart contract. This can be useful for developers to understand the logical flow of a program and identify potential problems or vulnerabilities.

Figure 3.6 shows a vulnerable path that was detected in the smart contract.

Additional information has been added to each basic block in the graph:

- block_id: id, which is used to identify the basic block in the contract. This makes it easier to analyze the structure and logic of the contract and also makes it easier to understand the results of the detector`s work.
- cost: opcode cost of the basic block execution.

Complete CFG for the RewardsManagement smart contract is provided in Appendices A.



Pic. 3.6 — CFG of the vulnerable path in the CourseManagement SC

Conclusions to the chapter.

In this section, we looked in detail at the process of implementing stateful smart contracts on TEAL and their interaction with the client application using the Algorand JS SDK. The structure of smart contracts and the main methods of their work were described. In the course of the work, smart contracts for course management and rewards management were created.

The CourseManagement application is necessary to manage the main processes on the platform that are associated with the course: creating a course, calculating its cost, registering students for the course, saving student scores, and calculating the sum of scholarships for students and rewards for teachers. The RewardsManagement smart contract controls the distribution of rewards to users. It also functions as an escrow account.

We also described the main vulnerabilities that may exist in the code of smart contracts created on TEAL. We examined the main approaches to verifying the code of smart contracts. In this work, we analyzed the created smart contracts for the presence of the described vulnerabilities. We also used the static code analysis tool for smart contracts on TEAL — Tealer, which allows us to search for vulnerabilities and explore malicious paths in the contract code.

CONCLUSIONS

In this thesis, we considered the possibilities of using blockchain, and smart contracts in particular, in education. We analyzed the features of these technologies as well as the benefits they provide.

This study was undertaken to consider the process of developing smart contracts for a blockchain-based educational platform. The smart contracts being developed make it possible to manage the process of registering students for a course, save their scores in the blockchain, calculate the amount of scholarships for students and rewards for teachers, and distribute other financial rewards to platform users.

In the course of our work, we identified the main roles of users on the educational platform and the actions of each of them, highlighting those use cases that are regulated by smart contracts.

Algorand was chosen as a blockchain platform for the development of smart contracts. It is one of the fastest, low-cost, carbon-negative blockchains that has advanced smart contract capabilities with low transaction fees. TEAL was chosen as the language for writing smart contract code for the educational platform.

We developed two smart contracts: CourseManagement and RewardsManagement. CourseManagement application manages the main processes on the platform related to the course: creating a course, calculating its cost, registering students for the course, saving student scores in blockchain, calculating the amount of scholarships for students and rewards for teachers. RewardsManagement application controls the distribution of financial rewards among platform users. It also functions as an escrow account.

In this work, we described the main vulnerabilities in smart contracts written in the TEAL language. We analyzed the causes of their occurrence and the possibility of eliminating these vulnerabilities. The developed smart

contracts were also tested for the described vulnerabilities. We also used the Tealer tool to check the code of our smart contracts for the presence of malicious paths.

REFERENCES

[1] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Decentralized Business Review, 21260.

[2] Cryptopedia: website. URL: https://www.gemini.com/cryptopedia/glossary

[3] What Is Blockchain and How Does It Work? (September, 2023). Medium: website. URL: https://medium.com/@zahidsardarsardarali/what-is-blockchain-and-how-does-it-work-6515fa916d58

[4] What are smart contracts on blockchain? IBM: website. URL: https://www.ibm.com/topics/smart-contracts

[5] ARCHANGEL - Trusted Archives of Digital Public Records. Surrey Blockchain: website. URL: https://blockchain.surrey.ac.uk/projects/archangel.html

[6] Ma, Y., & Fang, Y. (2020). Current Status, Issues, and Challenges of Blockchain Applications in Education. International Journal of Emerging Technologies in Learning (iJET), 15(12), pp. 20–31. https://doi.org/10.3991/ijet.v15i12.13797

[7] Ullah, N., Mugahed Al-Rahmi, W., Alzahrani, A. I., Alfarraj, O., & Alblehai, F. M. (2021). Blockchain technology adoption in smart learning environments. Sustainability, 13(4), 1801.

[8] Sun, H., Wang, X., & Wang, X. (2018). Application of Blockchain Technology in Online Education. International Journal of Emerging Technologies in Learning (iJET), 13(10), pp. 252–259. https://doi.org/10.3991/ijet.v13i10.9455

[9] Yalanetskyi, V. (2023). BLOCKCHAIN-BASED LEARNING MANAGEMENT SYSTEMS. Electronic Professional Scientific Edition «Cybersecurity: Education, Science, Technique», 3(19), 56–68. https://doi.org/10.28925/2663-4023.2023.19.5668

[10] Samanta, A. K., Sarkar, B. B., & Chaki, N. (2021). A blockchain-based smart contract towards developing secured university examination system. Journal of Data, Information and Management, 3, 237-249.

[11] Sanni, M. I., & Apriliasari, D. (2021). Blockchain Technology Application: Authentication System in Digital Education. Aptisi Transactions on Technopreneurship (ATT), 3(2), 151-163.

[12] What is Educational Platform. IGI Global: website. URL: https://www.igi-global.com/dictionary/emerging-platform-education/42260#:~:text=An%20integrated%20set%20of%20interactive,enhance%20educational%20delivery%20and%20management

[13] Konnova. O. V. "Using algebraic programming methods to analyze models of education tokenization". URL: http://ekhsuir.kspu.edu/handle/123456789/16484

[14] Algorand Standard Assets (ASAs). Algorand Developer Portal: website. URL: https://developer.algorand.org/docs/get-details/asa/ (date of application: October 2023)

[15] The world's most powerful and sustainable blockchain. Algorand: website. URL: https://algorand.com/

[16] Chen, J., & Micali, S. (2016). Algorand. arXiv preprint arXiv:1607.01341.

[17] Introduction. Algorand Developer Portal: website. URL: https://developer.algorand.org/docs/get-details/dapps/smart-contracts/ (date of application: September 2023)

[18] The smart contract language. Algorand Developer Portal: website. URL: https://developer.algorand.org/docs/get-details/dapps/avm/teal/ (date of application: September 2023)

[19] js-algorand-sdk. GitHub: website. URL: https://github.com/algorand/js-algorand-sdk

[20] DeFi to FutureFi. Algorand: website. URL: https://algorand.com/resources/defi

[21] Algorand dispenser. URL: https://bank.testnet.algorand.network/

[22] Pera wallet. URL: https://perawallet.app/

[23] AlgoExplorer. URL: https://algoexplorer.io/

[24] (Not So) Smart Contracts. Trail of Bits: website. URL: https://secure-contracts.com/not-so-smart-contracts/algorand/index.html (date of application: September 2023)

[25] Sun, Z., Luo, X., & Zhang, Y. (2023). Panda: Security analysis of algorand smart contracts. In 32nd USENIX Security Symposium (USENIX Security 23) (pp. 1811-1828).

[26] Rekeying. Algorand Developer Portal: website. URL: https://developer.algorand.org/docs/get-details/accounts/rekey/ (date of application: September 2023)

[27] Tealer. GitHub: website. URL: https://github.com/crytic/tealer (date of application: August 2023)

[28] Vara Prasad Bandaru. (2022, February 09). Detector Documentation. https://github.com/crytic/tealer/wiki/Detector-Documentation#missing-assetcloseto-field-validation

[29] Bartoletti, M., Bracciali, A., Lepore, C., Scalas, A., & Zunino, R. (2021). A formal model of Algorand smart contracts. In Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part I 25 (pp. 93-114). Springer Berlin Heidelberg.

[30] Corwin Smith. (March, 2023). Formal Verification of smart contracts. Ethereum: website. URL: https://ethereum.org/en/developers/docs/smart-contracts/formal-verification/

[31] Anne Kenyon. (Febriary 28, 2022). Hello? Contract Calling. Algorand: website. URL: https://algorand.com/resources/blog/hello-contract-calling

[32] Rautenberg M. J., Rezabek F. (2022). A Case Study of Security Vulnerabilities in Smart Contracts. Seminar IITM SS 22, Network Architectures and Services. URL: https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2022-11-1/NET-2022-11-1_10.pdf

[33] Espina V. B. (2020). Symbolic execution. OpenZeppelin: website. URL: https://forum.openzeppelin.com/t/symbolic-execution/2158

[34] Mohanta, B. K., Panda, S. S., & Jena, D. (2018, July). An overview of smart contract and use cases in blockchain technology. In 2018 9th international conference on computing, communication and networking technologies (ICCCNT) (pp. 1-4). IEEE.

[35] Khan, S. N., Loukil, F., Ghedira-Guegan, C., Benkhelifa, E., & Bani-Hani, A. (2021). Blockchain smart contracts: Applications, challenges, and future trends. Peer-to-peer Networking and Applications, 14, 2901-2925.

[36] Wang, S., Ouyang, L., Yuan, Y., Ni, X., Han, X., & Wang, F. Y. (2019). Blockchain-enabled smart contracts: architecture, applications, and future trends. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 49(11), 2266-2277.

[37] Sathya, A. R., Panda, S. K., & Hanumanthakari, S. (2021). Enabling smart education system using blockchain technology. In Blockchain Technology: Applications and Challenges (pp. 169-177). Cham: Springer International Publishing.

[38] Cheng, J. C., Lee, N. Y., Chi, C., & Chen, Y. H. (2018, April). Blockchain and smart contract for digital certificate. In 2018 IEEE

international conference on applied system invention (ICASI) (pp. 1046-1051). IEEE.

[39] He, D., Deng, Z., Zhang, Y., Chan, S., Cheng, Y., & Guizani, N. (2020). Smart contract vulnerability analysis and security audit. IEEE Network, 34(5), 276-282.

[40] Raimundo, R., & Rosário, A. (2021). Blockchain system in the higher education. European Journal of Investigation in Health, Psychology and Education, 11(1), 276-293.

Appendix A