Development of a simulator to determine personal financial strategies using machine learning

Dmytro S. Antoniuk[1], Tetiana A. Vakaliuk[1,2,3], Vladyslav V. Didkivskyi[1] and Oleksandr Yu. Vizghalov[1]

[1]Zhytomyr Polytechnic State University, 103 Chudnivsyka Str., Zhytomyr, 10005, Ukraine

[2]Institute of Information Technologies and Learning Tools of the NAES of Ukraine, 9 M. Berlynskoho Str., Kyiv, 04060, Ukraine

[3]Kryvyi Rih State Pedagogical University, 54 Gagarin Ave., Kryvyi Rih, 50086, Ukraine

Abstract

Personal finances are the own capital of an individual or family, which he manages independently. Personal finance management includes the ability to manage them, which requires separate training. The purpose of this work is to develop the mechanics of a personal finance simulator and build a system for determining personal financial strategies using machine learning. The development of a personal finance management simulator will allow in the future using it to teach the elements of managing such finances, even at school age. The developed software package consists of two parts: a personal finance management simulator and a system for determining financial strategies, which uses reinforced learning opportunities. The direction of future research is the integration of the recommendation system into the web application of the simulator. In addition, to improve the system and give flexibility in recommendations that could be adjusted to the current financial condition of the participant, it is necessary to pay attention to the use of "policy-based" methods of training with reinforcement, which are widely used in autopilot training systems for cars. Just as an autopilot cannot predict what may happen on the road in a second, can a person predict neither unexpected expenses nor profits (which is rare, but can happen).

Keywords

personal finance, simulator, business simulator, machine learning, development

1. Introduction

More and more questions and problems related to finances are becoming relevant for everyone. Therefore, it is very important not only to be aware of the latest economic and market developments but also to be competent in the ability to manage personal finances.

Personal finances are the own capital of an individual or family, which he manages independently. Personal finance management includes the ability to manage them, which requires separate training [1].

CS&SE@SW 2021: 4th Workshop for Young Scientists in Computer Science & Software Engineering, December 18, 2021, Kryvyi Rih, Ukraine

dmitry_antonyuk@yahoo.com (D. S. Antoniuk); tetianavakaliuk@gmail.com (T. A. Vakaliuk); v.didkivskyi@sana-commerce.com (V. V. Didkivskyi); aleksandrvizur@gmail.com (O. Yu. Vizghalov)

https://sites.google.com/view/neota (T. A. Vakaliuk)

0000-0001-7496-3553 (D. S. Antoniuk); 0000-0001-6825-4697 (T. A. Vakaliuk); 0000-0002-4615-7578 (V. V. Didkivskyi); 0000-0003-0985-4929 (O. Yu. Vizghalov)

Personal finance management and planning has a wide range of approaches dis-cussed in scientific and applied sources worldwide. Traditionally the problem is studied within the prism of efficient and safe personal fund liquidity management. Kaplan and Violante [2] discuss so-called "hand-to-mouth" consumption as the challenge to solve with improving personal finance competence while designing fiscal stimulus payments. The evidence of the importance of conscious management of personal finance was the fount and highlighted with extensive data analysis in the study of Olafsson and Pagel [3].

Behavioral finance approaches are getting popular in analyzing and consulting personal financial habits and management. The comprehensive study on the publications in the field of behavioral economics and behavioral finance highlights high interest and publication activity in the field, while concluding that despite behavioral finance has the higher publication activity and a broader network of authors, it concentrates on the definition of theoretical problems and pays less attention to finding solutions [4].

Application of the ICT to study, visualize and develop personal finance management is a reasonable and inevitable movement revealing itself in the modern digitalized and computerized world. The use of business simulations in higher education establishments highlights the effectiveness and efficiency of this active learning method to support traditional educational methods [5]. More sophisticated approaches of the use of artificial intelligence to employ behavioral finance apparatus in investment banking, back-end, and client-facing operations are being studied now. Königstorfer and Thalmann [6] conducted a structured literature review of the examples of the productive application of AI and behavioral finance to bank operations as well as the challenges to it.

The problem of development immersive, realistic, and ML-enabled educational simulation in the area of private finance competency development is not sufficiently developed and described. This work aims to contribute to this field of theoretical study and practical application.

The purpose of this work is to develop the mechanics of a personal finance simulator and build a system for determining effective personal financial strategies using machine learning. The methods of web programming, object-oriented programming, machine learning, analysis of scientific and scientific-practical publications, experimental analysis were used in the work.

The development of a personal finance management simulator will allow in the future to use it to teach the elements of such financial management, even at school age [1].

2. Results

Proper management of personal finances requires separate training. Therefore, in this work, it is necessary to develop a simulator that will allow users to develop skills in personal finance management.

Also in this work, it is necessary to develop a system for determining effective strategies for the development of the developed simulator. Performing this task and integrating such a system into a simulator would make it possible to give tips to users who could form in users an idea of how to better manage their finances.

To achieve this goal, the following tasks have been identified that relate to the functionality of the simulator:

1. *Develop a home page*. The page should contain the title and description of the simulation. To start the simulation, the user must enter the following data:

    - country;
    - currency, which will be considered the main currency of the user and cannot be changed during the simulation;
    - approximate salary and average monthly expenses.

    The user must be able to enable or disable the "assessment mode". The user must be able to start the simulation.
2. *Develop a simulation page*. This page should contain the following information/features:
    - data on salary and average monthly expenses indicated by the user on the home page;
    - the possibility of moving to the next week of the simulation;
    - the possibility of moving to the next month of the simulation;
    - the ability to stop the simulation and return to the home page;
    - information on user rewards by categories of savings and deposits (this information should be displayed for 24 weeks of the simulation and only if the user has enabled the "evaluation mode" on the home page);
    - information on funds on current accounts;
    - the ability to open a new account;
    - the ability to transfer funds from a current account to savings;
    - the ability to exchange currencies (this option must be available if the user has added at least one additional currency);
    - information on savings;
    - the ability to transfer funds from savings to the current account;
    - information on current exchange rates;
    - the opportunity to open a deposit in one of the available currencies;
    - information on open deposits;
    - opportunity to replenish an open deposit;
    - opportunity to take a loan;
    - information on open loans;
    - opportunity to invest in non-financial investments;
    - information on their contributions to non-financial investments;
    - information on the latest changes in the accounts for the last week;
    - information on the dynamics of changes in funds on the accounts during the entire time of the simulation;
    - information about the last executed transactions on all accounts.
3. *Develop a system of random events*, which is that at the beginning of each week the user may have an "unexpected event": illness, birth, etc. Each event should occur no more than once in 10 weeks, which will not distract the user too often from other features of the simulator.

4. *Develop a page with the results of the simulation*, which should be available when you reach the 60th week of the simulation. The participant should be able to get acquainted with the averages in his country, as well as view the averages of participants from other countries. The participant must also be able to leave feedback or suggestions in a special field. The information must be sent to the e-mail address of the site administrator.

2.1. Substantiation of technologies and means of software implementation

To implement the server part of the software package to create and conduct simulations to determine the effectiveness of personal financial strategies using machine learning capabilities, the object-oriented programming language C#, and modern technology for creating cloud applications ASP.NET Core 2.2.

Among the advantages of ASP.NET Core 2.2 [7]:

- high productivity and optimization;
- the ability to develop and run in Windows, macOS, and Linux;
- tools that simplify the process of modern web development;
- the only solution for creating a user web interface and Web API;
- arranged system of implementation of dependencies, which allows the components of the developed system to be weakly connected, to adhere to the principles of inversion of dependencies and a single duty.

During project development, the project was upgraded from ASP.NET Core 2.2 to a new version of ASP.NET Core 3.1. ASP.NET Core 3.1 includes many enhancements that reduce memory usage and improve bandwidth.

Framework SQLite technology was chosen to access the data, which allows you to work with the database using objects without having to write a large amount of code to access the data. SQLite is a lightweight relational database management system [8].

SQLite technology was chosen because of the following features: easy to use; does not require installation, as well as administration; the database is stored together with the application; speed of operations.

To implement the client part, the JavaScript programming language was chosen, which is the undisputed and unalterable leader in the development of client interfaces for web applications [9].

To create a user interface, the React library was used to create one-page sites. The purpose of React is to provide high speed, simplicity, and scalability [10].

React is often used with Redux, an open JavaScript library designed to manage the state of the program. Redux helps developers optimize program code. The main concept of Redux [11]:

1. *The only source of data.* Redux stores the state of the entire application in the object tree in one repository. One state tree makes it easy to debug or test a program; it also allows you to maintain the state of the application during development, to speed up the development cycle.

2. *The state is read-only.* The only way to change the state is to single out an action, an object that describes what happened. This ensures that neither views nor function callbacks will ever change status. Instead, they only express their intention to do so. All changes are centralized and occur one after another in a clear sequence. Because actions are simple objects, they can be registered, serialized, saved, and later played back for debugging or testing.
3. *The state changes with the help of pure functions.* Reducers are just functions that take the previous state and action and return to the next state. In the development process, reducers can be divided into smaller reducers that control certain parts of the state tree. Because reducers are just functions, they allow you to control the order in which they are sent, transfer additional data, or even create duplicate reducers for common tasks such as paging.

The open-source library Formik was chosen to build custom forms. Formik has the following advantages: saves and tracks changes in field values; organizes the check and processes the entered values; saves information about visited fields. Formik allows you to focus more on business logic and spend less time building forms.

To simplify the development of the user interface and design, a set of Material UI tools was used, which contains CSS and HTML templates for forms, buttons, and other interface components, as well as additional JavaScript extensions.

The developed simulator is a web solution that is deployed on the Internet. The Microsoft Azure cloud platform was used to deploy the site. Microsoft Azure is designed to develop cloud computing applications and is designed to simplify the process of creating online applications [12].

2.2. Identify options for using the simulator

The developed personal finance management simulator assumes the existence of the following roles: “Guest”; “Participant”.

To understand what opportunities are provided for users of the system, a description of the options for using the site. The user with the role “Guest” has the opportunity to enter information about himself and start the simulation. The “Guest” who started the simulation changes his role to “Participant”.

We will describe all the functionality of the simulator for comprehensive training of the user in personal finance. On this page, the user with the role of “Participant” has many opportunities to learn how to manage personal finances. The following is a list of information that a user with the role “Participant” can see, as well as opportunities that the participant can perform on the simulation page:

- Can see his salary and average monthly expenses, which the participant indicated on the home page.
- Can move to the next week of the simulation.
- Can move to the next month of the simulation.
- Can stop the simulation and return to the home page.

- Can see information about their rewards by categories of savings and deposits (this information is available from the 24th week of the simulation and only if the user has enabled the “evaluation mode” on the home page).
- Can see information about funds in current accounts.
- Can see information about savings.
- Can open an account in any currency of the world.
- Can transfer funds from a current account to savings.
- Can buy the currency of any country in the world (this option is available if the user has opened a current account in at least one other currency).
- Can transfer funds from savings to a current account.
- Can get acquainted with current exchange rates.
- Has the opportunity to open a deposit in one of the available currencies.
- Can see information about open deposits.
- Has the opportunity to replenish an open deposit.
- Has the opportunity to take a loan.
- Can see information about open loans.
- Has the opportunity to invest in non-financial investments.
- Can see information about their contributions to non-financial investments.
- Can see information about the latest changes in accounts for the last week.
- Can see information about the dynamics of changes in funds in the accounts throughout the simulation.
- Can see information about recent transactions on all accounts.

2.3. Software architecture

To develop a personal finance management simulator, a cloud architecture was chosen that allows access to the developed system using a centralized resource – the Internet.

The designed website provides a minimum number of pages, but those that contain a lot of complex functionality. The pages of a website provide constant dynamic changes that depend on the user’s actions. That is why a one-page architecture (SPA) was chosen for this application.

A one-page application (SPA) is a website that interacts with the user, dynamically rewriting the components of the current page with new data, which in most cases is provided by a web server, instead of the default browser loading entire new pages, as in multi-page applications (MPA). The SPA approach on the Internet is similar to the single-document interface (SDI) presentation technique popular in personal computer programs [13].

Among the main advantages of using SPA architecture: faster user interaction with the site; reduction of traffic; much greater opportunities to develop complex in terms of design functional components.

SPA moves logic from the server to the client, and the role of the webserver is transformed into a pure API that only provides data, often in JSON format. This architectural style is called Thin Server Architecture. In such a web application architecture, complexity is transferred from the server to the client, with the argument that this ultimately reduces the overall complexity of the system, making it flexible.

In figure 1 shows a diagram of a personal finance management simulator, which consists of the following components:

1. A client (web browser). Responsible for building views and displaying them to the client. Because the SPA architecture of the web application has been chosen, the client makes the first request to the server, which loads the initial page structure. The following requests are made to the webserver API, which returns the JSON data used by the client to modify individual view components.
2. FinanceSimulator ASP.NET WebApp – a web server that acts as an API, ie is responsible for storing and providing data to the client. The application consists of the following collections:
   2.1. FinanceSimulator.Web contains APIs that the client accesses to retrieve data or perform certain operations.
   2.2. FinanceSimulator.Core manages and stores data by accessing the SQLite data-base that is deployed with the web application. Contains the main classes of the simulator related to its business logic

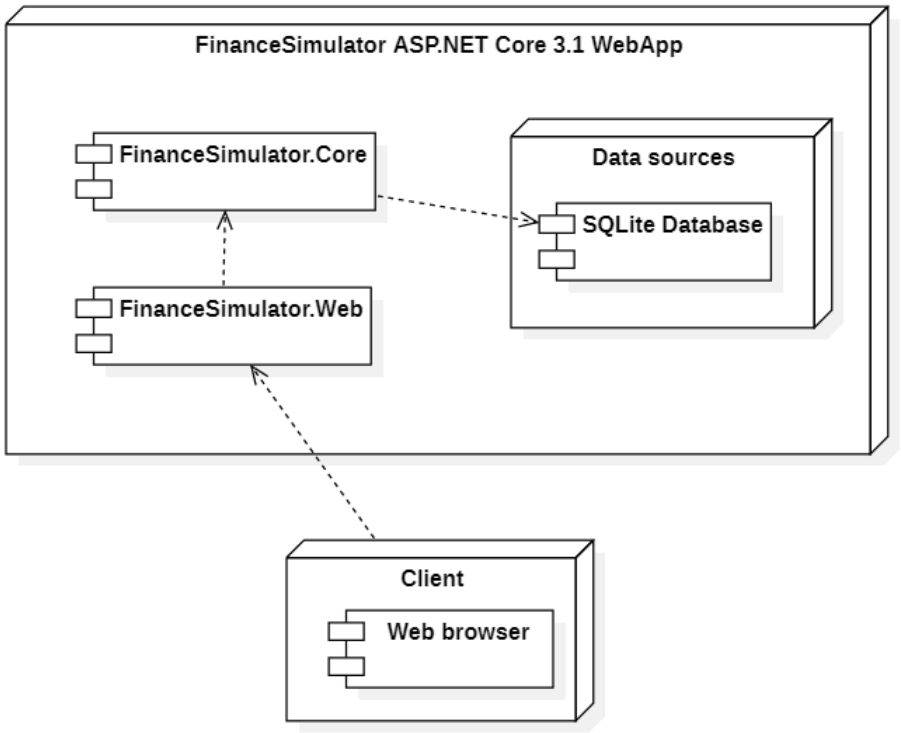

Figure 1: Diagram of simulator components.

One of the most popular modern libraries for creating one-page sites is React, which was chosen to implement a personal finance management simulator.

During the development of the project, it was also decided to use the architectural approach Flux. The main feature of Flux architecture is the one-way direction of data transfer between components. The architecture imposes restrictions on the data flow, in particular, eliminates the possibility of updating the state of the components them-selves. This approach makes the data flow predictable and makes it easier to track the causes of possible errors in the software [14].

The Flux architecture provides the following components [15]:

1. Actions – assistants that simplify the transfer of data from the manager.
2. Dispatcher – takes action and sends the load to registered handlers.
3. Stores – containers for program status and business logic in handlers registered in the manager.
4. Views – React components that collect the state of storage and pass it to child components through properties.

Nowadays, one of the most popular implementations of Flux architecture is the Redux library, which was used in the development of the simulator.

The .NET platform was chosen to create a program that could determine effective action strategies in a simulator using machine learning. As described in the previous section, this system will be created using the open-source library SharpRL, as Microsoft and its ML.NET platform do not yet have a ready-made solution to solve problems with reinforcement.

The program is a console application that is in the same solution as the simulator, but despite this, this system is a separate program that runs separately.

In figure 2 the structure of the decision tree with the developed program of finding financial strategies is presented.

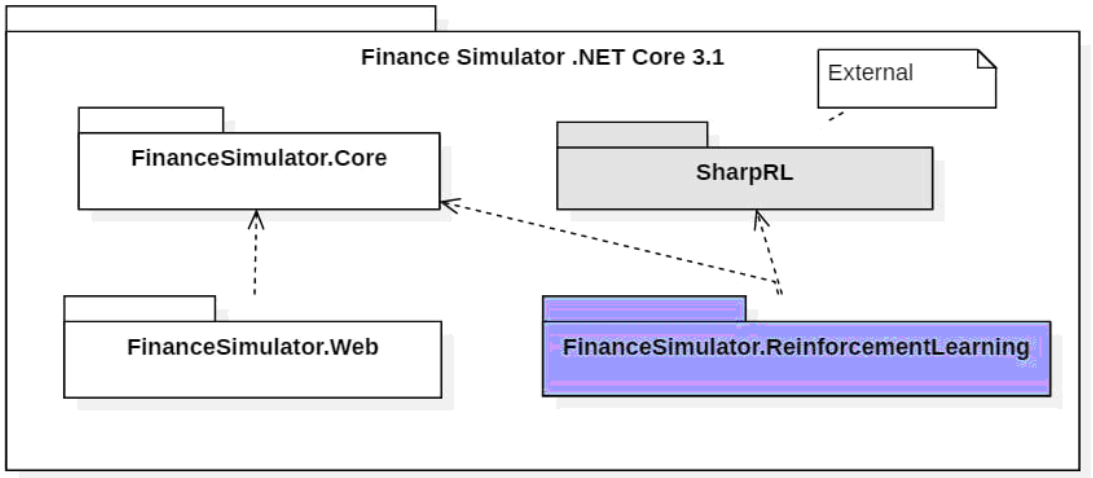

Figure 2: The structure of the solution in the form of assemblies.

As can be seen from the diagram in figure 2 the developed machine learning program uses a joint with the simulator assembly FinanceSimulator.Core, which is required to avoid duplication of code.

The SharpRL library contains basic classes and interfaces for programming reinforced model learning. The machine learning algorithm used in this library is Q-learning, which was described in the previous section (figure 3).

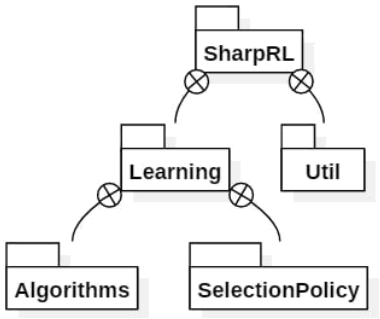

Figure 3: The structure of the adapted collection SharpRL.

To narrow the general space of possible actions in the simulator that can be done by the agent of the machine learning system, it was decided to allocate one type of action for each tool, for example, savings, deposits. For each type of instrument, it was necessary to create a list of possible changes in the balance in percent (if the action falls from 0% of changes in the balance, then nothing changes after its execution). Each week, the agent must perform one action for each tool. To implement this idea, it was necessary to partially change the source code of the library "SharpRL", which is why this external assembly was connected to the solution together with the source code.

Figure 4 shows a class diagram of the adapted SharpRL assembly, which describes the following classes and interfaces:

1. *Agent* – a class of system agents that implements the interface agent, which describes the entity that makes decisions in the environment. As a result of decision-making, they affect him and the environment. Thus the environment of the system is investigated.
2. *Environment* – a system environment-class that implements the environment interface. The environment can contain many agents who explore it by making decisions about the implementation of specific actions by the environment.
3. *Configuration* – a class that contains learning parameters that must be configured externally.

The learning process is carried out specified in the parameters of the number of simulations. Next, to obtain the final results of the training, there is an assessment in compliance with the policy of choice of actions of the agent.

2.4. Adjust simulator settings

The developed personal finance management simulator is a web application that has been deployed on the Internet [16].

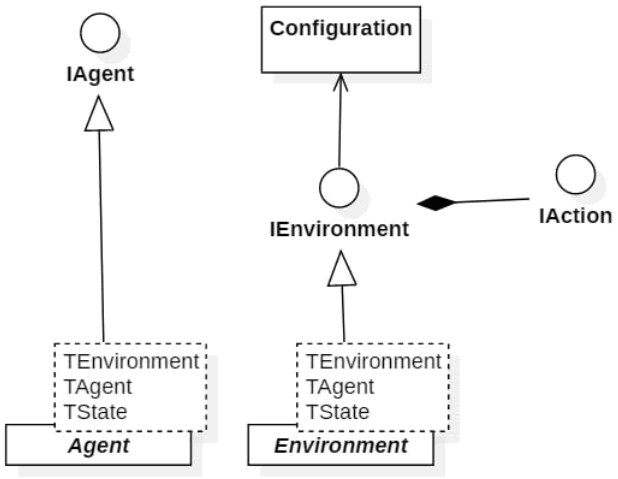

Figure 4: Class diagram of the adapted assembly SharpRL.

A Microsoft Imagine subscription was used to deploy the website, a program of Microsoft Corporation that offers students and graduate students free access to software design and development tools.

In figure 5 presents the site control panel, which provides various management and monitoring capabilities. As you can see from the figure, the site works stably without critical errors.

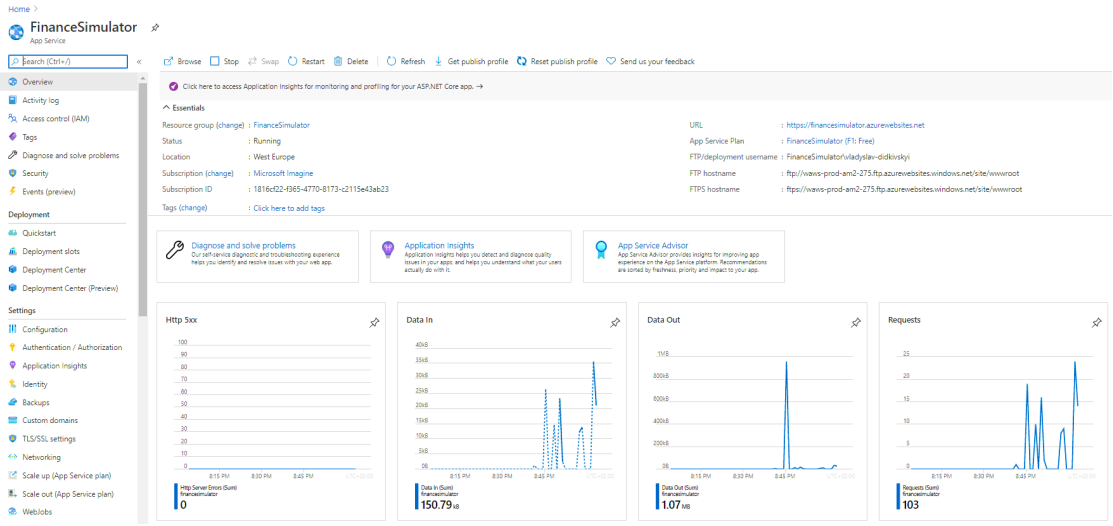

Figure 5: Microsoft Azure site control panel.

A deployment diagram was developed to represent the hardware components of the website and how the different parts of the system work with each other (figure 6).

The diagram shows the following components:

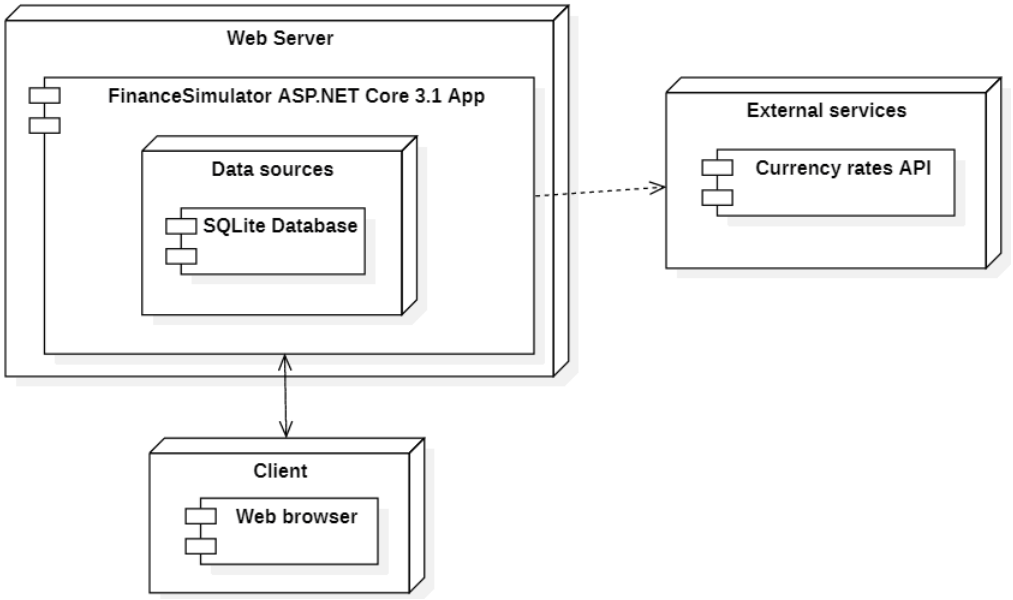

Figure 6: Deployment diagram of the developed simulator.

1. A *client*, which is a web browser, makes HTTP requests to a web server.
2. A *web server* that receives HTTP requests from the client and returns the re-quested data in response. The SQLite database is deployed with the web application.
3. *External service* to obtain information on current exchange rates.

A configuration mechanism has been developed for easy control of the simulator functionality. You can change the settings on the server without any interference with the source code. The settings consist of the following files:

- appsettings.json – contains settings for connection to the database, as well as a key to the API of the external system, which provides information about currency exchange rates;
- Countries.json – contains a set of settings about the countries that the user can select in the simulator;
- Currencies.json – contains a set of settings about currencies that the user can select in the simulator;
- Deposits.json – contains a set of settings about deposits that the user can open in the simulator;
- NonFinancialInvestments.json – contains a set of settings for non-financial in-vestments that the user can buy in the simulator;
- UserEvents.json – contains a set of settings about random events that may happen to the user in the simulator;
- UserQuiz – contains a set of test settings that the user must take within the time specified in the settings.

2.5. The structure of the simulator interface

Consider the part of the functionality that directly relates to the mechanics of the simulator. The first page that the user will see when opening the simulator site is a welcome page. This is the only page that a guest can see before starting the simulation. This page contains a set of fields that must be filled in by the user to start the simulation. The home page contains the following fields to fill in (figure 7):

1. *Country* – required only for grouping user data by country in the final graphs. The country is determined automatically depending on the location of the user, but he can change it.
2. *Currency* – the main currency of the user, in which he or she receives a salary and incurs daily expenses.
3. *Monthly salary and expenses.* The user can choose from several options or enter their data.
4. *Evaluation enables the option.* This option will allow the user to check the list of completed and uncompleted assessment tasks by different categories. This information is available after reaching 24 weeks of simulation. After 24 weeks, the user can still complete all tasks.

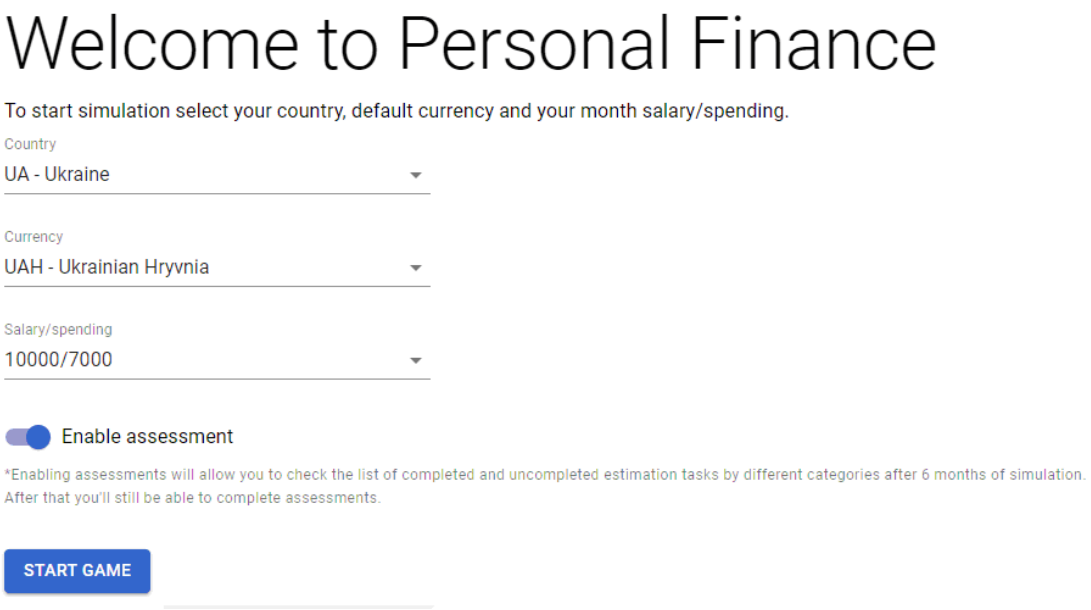

Figure 7: Home page of the simulator.

After the guest fills in all the data and clicks on the “Start game” button, he or she will change his or her role to “Participant” and will be redirected to the simulation page (figure 8).

The simulation page is divided into 3 main parts:

1. Site header contains only the site title and the number of the current simulation week.

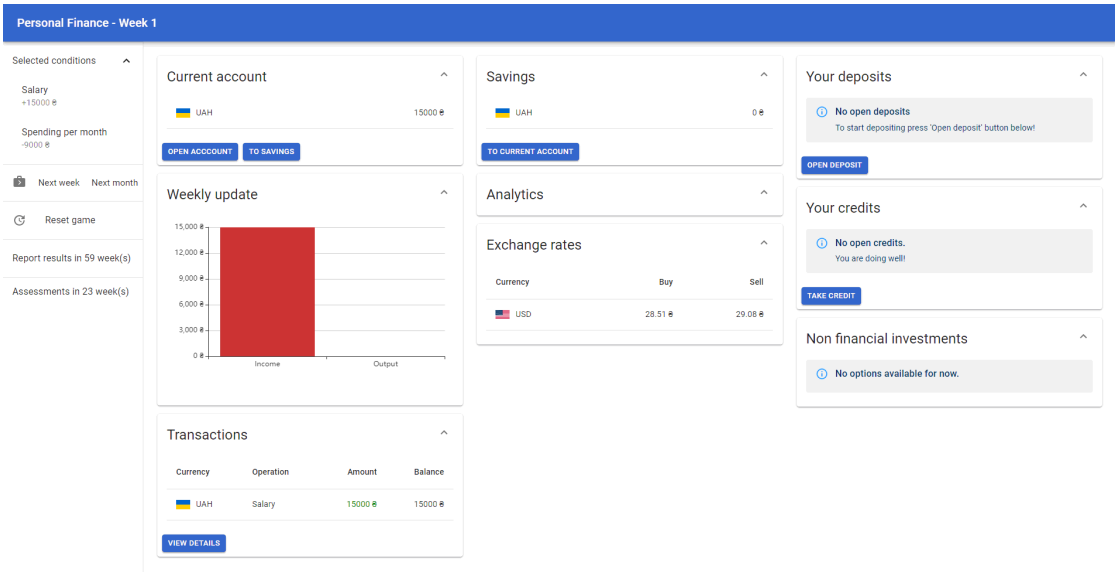

Figure 8: Simulation page.

2. The control panel, which is located on the left side of the window.
3. The main part with cards of separate functionalities of the simulator.

Consider in more detail the control panel of the simulator.
The simulator control panel contains:

1. User-entered data on monthly salaries and expenses.
2. “Next week” button moves to the next week of the simulation. After this transition, the amount of ¼ monthly expenses is automatically deducted from the user’s current balance. After the transition to the next month, the current balance of the user is automatically accrued salary.
3. “Next month” button moves to the next month. That is, this option does the same thing that will happen if you click 4 times on “Next week”.
4. The “Reset game” button, which completes the simulation, changes the user’s role from “Participant” to “Guest” and translates to the home page. To avoid the risk of losing the entire result of the simulation due to carelessness, the user must confirm the completion in an additional window that will appear after clicking on the “Reset game” (figure 9).
5. At 60 weeks, you will have the opportunity to review your simulation results (this option will be described in more detail below). Until the 60th week, information will be displayed on how many weeks are left before the opportunity opens.
6. At 24 weeks, you will be able to view the evaluation results if the "Enable assessment" option has been enabled on the home page. Here the user will be able to check the list of completed and uncompleted assessment tasks for different categories (figure 10). Until the 24th week, information will be displayed on how many weeks are left before the opportunity opens.

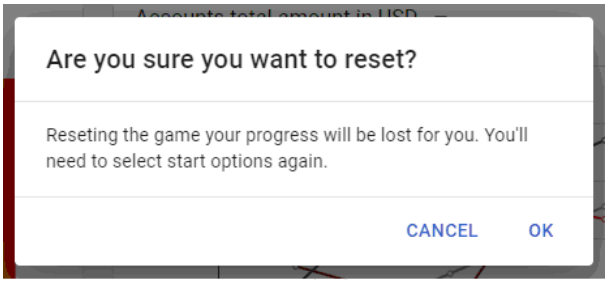

Figure 9: Example of confirmation of completion of the simulation.

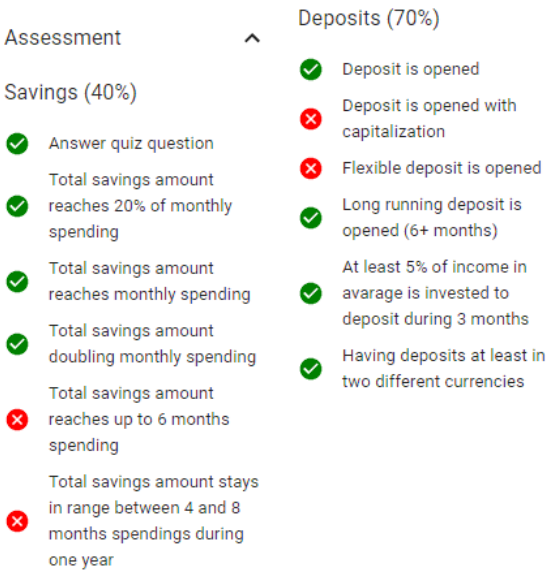

Figure 10: Example of the evaluation result.

3. Conclusions

The developed software package consists of two parts: a personal finance management simulator and a system for determining financial strategies, which uses reinforced learning opportunities.

The direction of future research is the integration of the recommendation system into the web application of the simulator. Also, to improve the system and give flexibility in recommendations that could be adjusted to the current financial condition of the participant, it is necessary to pay attention to the use of "policy-based" methods of training with reinforcement, which are widely used in autopilot training systems for cars. Just as an autopilot cannot predict what may happen on the road in a second, neither can a person predict unexpected expenses or profits (which is rare, but can happen). The built recommendation system based on the "value based" method assumes that the participant of simulation will follow absolutely all steps of the offered way. If any accidental event occurs that affects the financial situation or the participant simply decides at some point not to perform the actions specified by the system, in this case, the

recommendations will not lead to the intended result. That is why in the future it is advisable to research the possibility of applying "policy-based" methods to improve the finding of optimal strategies for personal finance management.

References

[1] D. S. Antoniuk, T. A. Vakaliuk, V. V. Didkivskyi, O. Y. Vizgalov, Necessity of the personal finance management simulation development, Innovative pedagogy 2 (2020) 208–212. doi:10.32843/2663-6085/2020/24-2.41.

[2] G. Kaplan, G. L. Violante, A model of the consumption response to fiscal stimulus payments, Econometrica 82 (2014) 1199–1239. doi:10.3982/ECTA10528.

[3] A. Olafsson, M. Pagel, The Liquid Hand-to-Mouth: Evidence from Personal Finance Management Software, The Review of Financial Studies 31 (2018) 4398–4446. doi:10.1093/rfs/hhy055.

[4] D. F. Costa, F. d. M. Carvalho, B. C. d. M. Moreira, Behavioral economics and behavioral finance: A bibliometric analysis of the scientific fields, Journal of Economic Surveys 33 (2019) 3–24. doi:10.1111/joes.12262.

[5] S. Palan, GIMS — Software for asset market experiments, Journal of Behavioral and Experimental Finance 5 (2015) 1–14. doi:10.1016/j.jbef.2015.02.001.

[6] F. Königstorfer, S. Thalmann, Applications of artificial intelligence in commercial banks – a research agenda for behavioral finance, Journal of Behavioral and Experimental Finance 27 (2020) 100352. doi:10.1016/j.jbef.2020.100352.

[7] Microsoft, ASP.NET documentation, 2022.

[8] SQLite Tutorial, SQLite Tutorial, 2021. URL: https://www.sqlitetutorial.net/.

[9] D. S. Shepiliev, S. O. Semerikov, Y. V. Yechkalo, V. V. Tkachuk, O. M. Markova, Y. O. Modlo, I. S. Mintii, M. M. Mintii, T. V. Selivanova, N. K. Maksyshko, T. A. Vakaliuk, V. V. Osadchyi, R. O. Tarasenko, S. M. Amelina, A. E. Kiv, Development of career guidance quests using WebAR, Journal of Physics: Conference Series 1840 (2021) 012028. doi:10.1088/1742-6596/1840/1/012028.

[10] Meta Platforms, Inc., React - A JavaScript library for building user interfaces, 2022. URL: https://reactjs.org.

[11] D. Abramov, the Redux documentation authors, Redux - A predictable state container for JavaScript apps, 2022. URL: https://redux.js.org.

[12] Microsoft, What is Azure?, 2022. URL: https://azure.microsoft.com/en-us/overview/what-is-azure.

[13] Mozilla, individual contributors, SPA (Single-page application), 2022. URL: https://developer.mozilla.org/en-US/docs/Glossary/SPA.

[14] A. Boduch, Flux Architecture, Packt Publishing, 2016.

[15] M. Kovalyova, Flux: Arkhitektura prilozhenii na React.js — vsestoronnee issledovanie, 2018. URL: https://medium.com/@marina.kovalyova/flux-the-react-js-application-architecture-773f515d068d.

[16] Personal finance, 2020. URL: https://financesimulator.azurewebsites.net/.