MINISTRY OF EDUCATION, CULTURE AND RESEARCH OF REPUBLIC OF MOLDOVA
TECHNICAL UNIVERSITY OF MOLDOVA
FACULTY OF COMPUTERS, INFORMATICS AND MICROELECTRONICS
DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATICS

TombTally: The Final Funds Calculator

DSA Individual Work Project report

Course coordinator: PhD, Associate Prof., Natalia Burlacu
Student: Maxim Alexei, FAF-232

Chișinău, 2024

Content

Introduction

Picture a busy city with crowded streets, cars moving slowly and horns honking. Traffic jams are more than just inconveniences; they cause stress, waste time, and harm the environment. Imagine being late for work because your bus is stuck in traffic, or missing time with your family because you're trapped in a sea of brake lights.

Across the world, traffic jams are a major problem, polluting the air and draining our resources. But what if there was a way to improve the flow of traffic in our city streets? The concept of "green lines" could be the solution.

By optimizing traffic light timings and creating dedicated paths of green lights, we can revolutionize how traffic moves through cities. This innovative approach offers the potential to reduce congestion, cut down on travel time, and improve the overall quality of life for everyone living in the city. This project examines the issue of traffic congestion and explores how implementing green lines can provide a solution. By calculating the most efficient green and red timing for traffic lights at crucial intersections and connecting them with green corridors, we aim to transform how people move around our city. Let's delve deeper into the concept of green lines and discover how they could be the key to unlocking smoother, more efficient urban transportation.

Problem-Solution Analysis

Problem Description

To begin with, in the context of financial planning, individuals face formidable challenges in accurately calculating and understanding their retirement, disability, and descendant pension benefits. These challenges are rooted in several key complexities:

- *Diverse Pension Types:* Pension types vary widely, while being characterized by intricate rules, eligibility criteria, and benefit structures. This variability makes it difficult for individuals to navigate and comprehend their entitlements, leading to confusion and uncertainty.
- *Limited Financial Literacy:* Many individuals lack sufficient financial knowledge and understanding of pension-related concepts such as, contribution rates, benefit calculations, and the impact of various life events on pension benefits.
- *Complexity of Retirement Planning:* Retirement planning involves numerous interconnected factors including current savings, investment strategies, retirement age, expected lifespan, and desired lifestyle in retirement. Understanding and projecting the implications of these variables on future financial security requires specialized tools and expertise.
- *Evolving Family Dynamics:* Changes in family structures, such as blended families and diverse dependents, add complexity to descendant pension planning. Determining beneficiaries and understanding pension implications for family members can be challenging.
- *Importance of Long-Term Financial Security:* Effective planning for retirement, disability, and descendant pensions is crucial for ensuring financial stability over the long term. Inadequate planning may lead to financial hardship during retirement or unforeseen life events.

Moreover, the budget allocation for pensions in 2024, totaling 28 billion moldovan lei with an 11% increase from the previous year, highlights the pressing importance of addressing challenges in pension planning and management[1]. In Moldova, the state is set to distribute this budget among 530,643 pensioners receiving age limit pensions, with each pensioner estimated to receive an average of 3,965 lei annually. This substantial financial commitment underscores the critical need for an efficient and sustainable pension calculator that can accurately compute and allocate resources to support retirees and ensure their financial well-being. As the number of pensioners and pension expenditures continue to grow, the effective management and optimization of pension funds become imperative to maintain fiscal stability and meet the evolving needs of an aging population.

Therefore, recognizing the intricacies, relevant reasons, and challenges inherent in retirement, disability, and descendant pension planning is essential to developing tailored and accessible solutions. By

elucidating these challenges, this problem description sets the stage for the development of comprehensive tools and resources aimed at enhancing financial literacy and promoting informed decision-making among individuals seeking to secure their financial futures.

Existent Solutions

In addressing the complexities of pension planning, two primary existent solutions have emerged to assist individuals in navigating their retirement finances.

The first solution involves the online pension calculator provided by the National House of Social Insurance (CNAS), designed to estimate future retirement pension amounts based on personal social insurance account data[2].

As we can observe, this solution has lots of functionality, as it lets users log in, see their current retirement plan status and other characteristics.

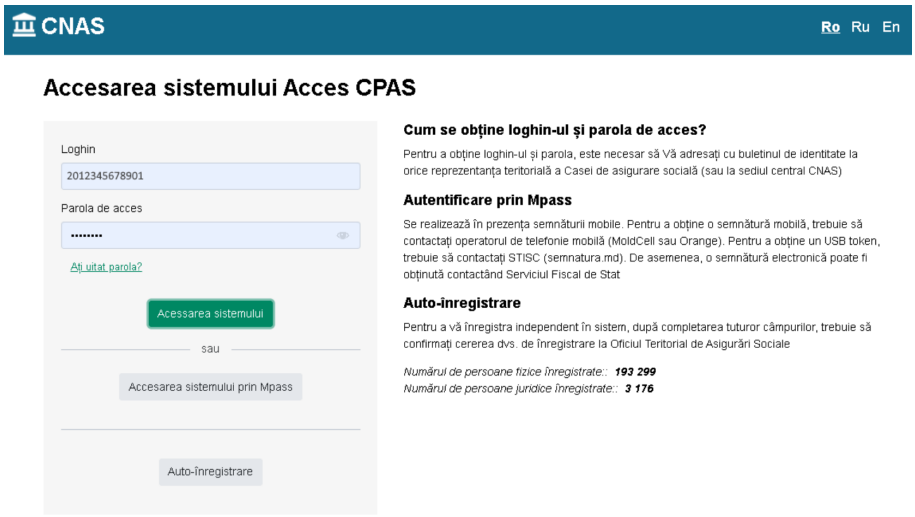

Figure 1.2.1 - Pension Calculator by CNAS - Log In Page[2]

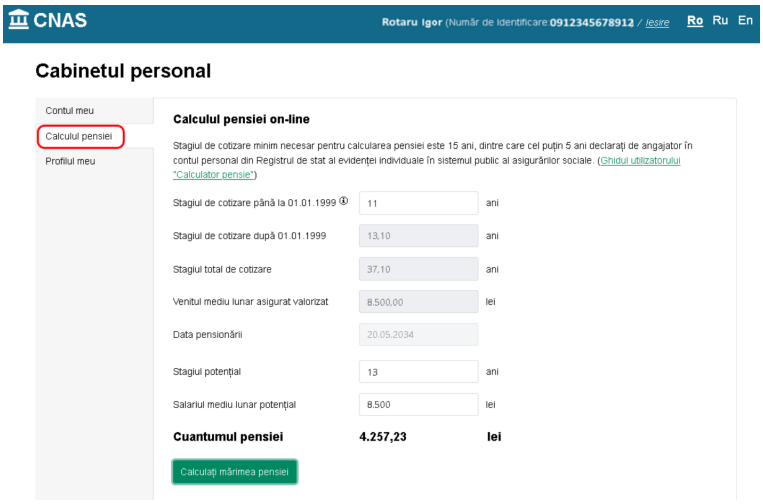

Figure 1.2.2 - Pension Calculator by CNAS - Dashboard Page[2]

Despite its potential benefits, accessibility issues arise due to the requirement for in-person registration at CNAS headquarters for login credentials, hindering its widespread adoption.

The second solution focuses on a tool that enable users to calculate their retirement age based on their current age, which is rather simple and it grants no valuable insights regarding the amount of money that could be received as a pension by one or another[3].

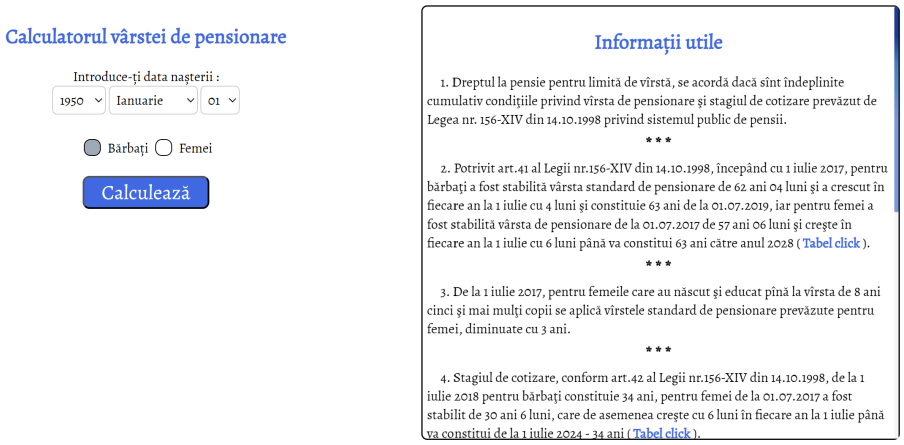

Figure 1.2.3 - Simple Retirement Age Calculator by CNAS[3]

By examining these existing solutions, we can identify areas for improvement and innovation to enhance accessibility and effectiveness in supporting individuals' pension planning needs, as some clear flaws are visible and this topic is indeed relevant as stated in the problem description section.

Solution Proposal

After analysing the state of the problem, regarding computing one's pension amount in Moldova and the existent solutions, TombTally was shaped, which is a meticulously designed platform aimed at revolutionizing pension planning by providing comprehensive and user-friendly tools for calculating various pension types. This innovative solution addresses the complexities and challenges individuals face in understanding and projecting their retirement, disability, and descendant pension benefits. TombTally stands out for its intuitive usability, catering to a diverse range of users seeking accurate and personalized pension planning assistance.

The main source of inspiration behind the development of TombTally was *the document regarding the Decision of the Government of the Republic of Moldova to Approve the Regulation on the Method of Calculating Pensions and the Method of Confirming the Contribution Period for Determining pensions*[4]. This comprehensive regulation provided a wealth of formulas, laws, regulations, and requirements essential for accurately computing pensions based on individual parameters. The clarity and detail outlined in this government decision served as a foundational resource for designing TombTally, ensuring that the platform incorporates the latest legal and mathematical frameworks to deliver precise and reliable pension calcula-

tions. By leveraging this regulatory framework, TombTally aims to streamline pension planning processes and empower users with the knowledge and tools necessary to make informed financial decisions about their retirement, disability, and descendant pensions. The commitment to aligning with government standards underscores TombTally's dedication to accuracy, compliance, and effectiveness in assisting individuals in securing their financial futures.

Thus, the implementation of TombTally promises several impactful benefits:

- *Comprehensive Pension Calculation:* Offers the ability to compute the three main types of pensions—retirement, disability, and descendant pensions—based on individual parameters and legal requirements outlined in the government regulation.
- *Advanced Formulas Implementation:* Implements sophisticated formulas, including the $V_{(}av)$ (average insured monthly valued income) formula, where all the monthly incomes are stored, for precise pension calculations rather than relying on user estimates or approximations.
- *Direct C Code Execution:* TombTally runs actual C code, leveraging low-level programming for robust and efficient computations, ensuring accuracy and performance.
- *Modern and Intuitive UI:* Boasts a slick, intuitive, and modern user interface (UI) that simplifies the pension planning process and enhances user experience.
- *Utilization of Data Structures:* Makes use of structs and other data structures to organize and manage pension-related data efficiently.
- *Textual Database for Data Storage:* Saves user data securely in a textual database, ensuring data integrity and accessibility.
- *Data Modification Capabilities:* Allows users to modify existing data and inputs, enabling flexibility and adaptability in pension planning scenarios.
- *Well-Defined Input/Output Methods:* Implements clear and structured input/output methods, guiding users through the pension calculation process seamlessly.
- *Greater Accessibility:* TombTally's user-friendly interface and online accessibility democratize pension planning, making it accessible to individuals from all walks of life.
- *Long-Term Financial Security:* By facilitating accurate projections and recommendations, TombTally contributes to long-term financial security and stability for users and their families.

User Persona

To develop a comprehensive understanding of our users and their diverse needs, we have created detailed user personas that represent key demographics and profiles. These personas enable us to empathize with our users, identify their goals, challenges, and expectations, and tailor our solutions to address their specific requirements effectively.

Elena Marin *(Figure 1.4.1)* represents a user profile that values simplicity, accuracy, and indepen-

dence in pension planning. TombTally's tailored features and user-centric design make it an ideal solution for Elena to optimize her pension benefits and achieve financial security in retirement.

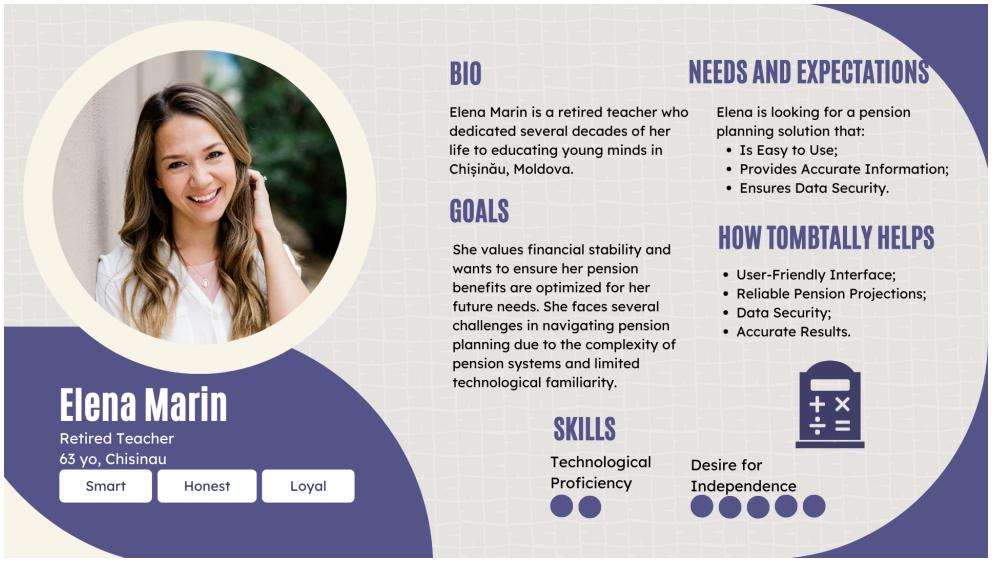

Figure 1.4.1 - User Persona - Elena Marin

Andrei Rusu *(Figure 1.4.2)* represents a user profile characterized by technical proficiency, forward-thinking mindset, and a proactive approach to financial planning. TombTally's tailored features and comprehensive capabilities make it an ideal solution for Andrei to calculate and optimize his pension benefits as he progresses in his career and plans for a secure financial future.

Figure 1.4.2 - User Persona - Andrei Rusu

By delving into the lives and motivations of our users, we can develop personalized experiences and user-centric solutions that resonate with individuals like Elena Marin and Andrei Rusu.

In continuity, we will cocus on the Value Proposition Canvas *(Figure 1.4.3)* that involves outlining

the key elements that define the value our product, TombTally, offers to specific customer segments.

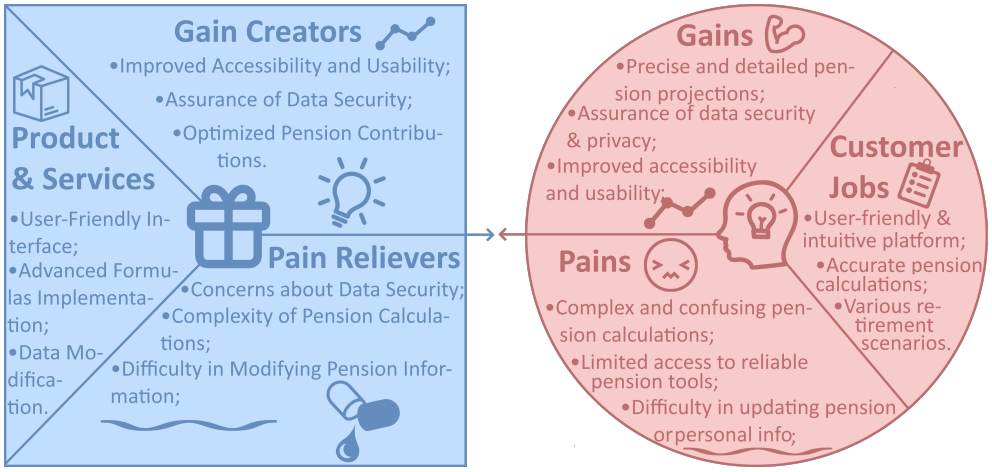

Figure 1.4.3 - Value Proposition Canvas

Customer Jobs

- *Accurate pension calculations:* Providing precise and reliable projections of retirement benefits.
- *User-friendly and intuitive platform:* Ensuring ease of use and navigation for pension planning.
- *Various retirement scenarios:* Allowing exploration of different scenarios to optimize pension strategies.

Gains

- *Precise and detailed pension projections:* Delivering accurate and comprehensive pension benefit forecasts.
- *Assurance of data security & privacy:* Safeguarding personal and sensitive pension information.
- *Improved accessibility and usability:* Enhancing the overall user experience and accessibility of pension planning tools.

Pains

- *Complexity of pension calculations:* Addressing the challenges associated with complex pension computations.
- *Limited access to reliable pension tools:* Providing accessible and dependable pension planning resources.
- *Difficulty in updating pension/personal info:* Offering easy-to-use features for modifying and managing pension data.

Gain Creators

- *Improved Accessibility and Usability:* Enhancing accessibility and usability to empower users in pension planning.
- *Assurance of Data Security:* Instilling confidence through robust data security measures.

- *Optimized Pension Contributions:* Enabling users to make informed decisions to optimize pension contributions.

  Product & Services

- *User-Friendly Interface:* Offering a modern and intuitive interface for easy navigation and interaction.
- *Advanced Formulas Implementation:* Utilizing sophisticated formulas for accurate and advanced pension calculations.
- *Data Modification Capabilities:* Providing features for updating and managing pension and personal information.

  Pain Relievers

- *Difficulty in Modifying Pension Information:* Simplifying the process of updating and managing pension details.
- *Complexity of Pension Calculations:* Streamlining complex calculations to enhance understanding and usability.
- *Concerns about Data Security:* Addressing user concerns by ensuring robust data security and privacy measures.

In the following sections, we will delve into the technical details of TombTally, exploring key components that drive the platform's functionality and effectiveness in pension calculation. This exploration will include a deep dive into the computation of formulas used for accurate pension projections, providing insights into the underlying calculations and methodologies. Additionally, we will examine use case diagrams that illustrate how TombTally supports various scenarios and user interactions, offering a comprehensive view of the platform's capabilities. Furthermore, we will explore the file structure and organization of TombTally, detailing how data is managed and stored securely to ensure user privacy and accessibility. By examining these technical aspects, we aim to provide a comprehensive understanding of TombTally's infrastructure and operational framework, highlighting its innovation and sophistication in delivering valuable pension planning and computing solutions.

Technical Details

Pension Types & Formulas

In this section, we will analyze the three main types of pensions that TombTally focuses on, namely the Age Limit Pension, Disability Pension, and Descendant Pension, as defined by government regulations. We will delve into the specific formulas and methodologies prescribed by the government for calculating these pensions, providing detailed insights into the parameters and variables that determine pension eligibility and benefit amounts. By understanding the intricacies of each pension type and its corresponding formula, the readers of this report will gain clarity on how TombTally accurately computes pension projections and facilitates informed decision-making. This analysis will showcase TombTally's adherence to regulatory standards and its commitment to providing users with reliable and comprehensive pension planning/computing functions tailored to their individual needs and circumstances.

The first type of pension we will focus on is the Age Limit one, which requires two different formulas depending on the scenario:

For the stages of contribution BEFORE 1999:

$$\mathbf{P} = \mathbf{P}_{\text{min}} \times \frac{Tt}{2Tmin} \times \mathbf{K}_{\text{np}}, where$$

P - pension amount;

P_{min} - the minimum pension for the age limit on the date of application for establishing the pension;

T_t - total stages of contribution, in years;

T_{min} - minimum stages of contribution that grants the right to pension (15 years);

K_{np} - the coefficient of the professional level, that's connected to:

1.0 - for farmers, unskilled workers (skill category 1-2) and unqualified auxiliary staff;

1.2 - for medium-skilled workers (skill category 3-4);

1.5 - for highly skilled workers (skill category 5-8) and specialists with specialized secondary education;

1.8 - for specialists with higher education;

2.0 - for management staff at structural subdivision level;

3.0 - for heads of units and their deputies.

For stages of contributions made AFTER 1999, if they were at least of 5 years:

$$\mathbf{P} = 1,35\%\text{x } \mathrm{T_t} \text{ x } \mathrm{V_{av}} \text{ , } where$$

P - pension amount;

1.35% - the accumulation rate for the contribution years, made after January 1, 1999, calculated as a percentage;

V_{av} - average insured monthly income, determined according to annex no.1 to the Law on the public pension system, as provided for in Chapter V of the Regulation;

T_t - total stages of contribution, in years.

In addition, if the amount of the pension calculated for the full period of contribution does not reach the minimum amount, a minimum pension is established.

Moreover, if the pension calculated under the conditions of a contribution period incomplete is below the minimum pension level, the insured is granted a pension calculated, which cannot be lower than the minimum pension reduced proportionally contribution period made according to the formula:

$$\mathbf{P} = \mathbf{P}_{\min} \times \frac{Tt}{Tl}, where$$

P_{min} - minimum pension amount;

T_t - total stages of contribution, in years;

T_l - the necessary complete stage of contribution during the date of establishing the pension;

T_t/T_l - the proportion between the total stages of contribution and the necessary completed one.

In continuity, we will have a look at the how the Disability pension is computed, and how many degrees of disability exist. In case, the insured falls under a degree of disability confirms the conditions of contribution period, in relation to age, according to art. 20 of the Law on the public pension system, the amount of the pension is determined based on art. 21 of the average insured monthly income achieved after January 1, 1999 and valued, taking into account the total contribution period.

The disability pension, in relation to the degree of disability, is calculated according to the formulas:

For Severe Disability:

$$\mathrm{P} = 0,42 \text{ x } \mathrm{V_{av}} + \frac{\mathrm{Tt}}{\mathrm{Tmax}} \text{x } \mathrm{V_{av}} \text{ x } 0,1$$

For Pronounced Disability:

$$\mathrm{P} = 0,35 \text{ x } \mathrm{V_{av}} + \frac{\mathrm{Tt}}{\mathrm{Tmax}} \text{x } \mathrm{V_{av}} \text{ x } 0,1$$

For Average Disability:

$$\mathrm{P} = 0,20 \text{ x } \mathrm{V_{av}} + \frac{\mathrm{Tt}}{\mathrm{Tmax}} \text{x } \mathrm{V_{av}} \text{ x } 0,1 \text{ , } where$$

P - pension amount;

T_t - total stages of contribution, in years;

T_{max} - the maximum potential contribution period (stages of contribution) from the age of 18 up to the retirement ages established in art. 41 para. (1), but not older than 40 years.

If the calculated amount of the disability pension is more less than the amount of the minimum pension, the minimum pension is granted which constitutes:

75% - from the amount of the guaranteed minimum monthly income established by law - in the case of a Severe Disability;

70% - from the amount of the guaranteed minimum monthly income established by law - in the case of a Pronounced Disability;

50% - from the amount of the guaranteed minimum monthly income established by law - in the case of a Average Disability.

The amount of the disability pension is calculated according to the formulas indicated in point 15 of this Regulation only if the insured placed in a degree of disability completed after January 1, 1999 an internship of contribution for at least 2 years.

Age of being diagnosed the disability	**Stages of Contribution (Years)**
Until 23 years old	2
23-29 years old	4
29-33 years old	7
33-37 years old	10
37-41 years old	13
Over 41 years old	15

Table 2.1.1 - Required stages of contribution based on the age, in order to compute the Disability Pension[5]

People who, starting with January 1, 1999, have not completed an internship contribution or the contribution period achieved after this date is less than 2 years the minimum pension is granted in the amounts indicated in point 16 hereof Regulations.

The last available pension for computing on TombTally is the Descendant one. The survivor/descendant's pension is granted if the deceased person was receiving an old-age pension or a disability pension or met the conditions for obtaining a disability pension.

The descendant's pension is granted to the following categories:

1. Children up to the age of 18 or, if they continue their studies in full-time educational institutions (secondary, specialized medium and higher), until they finish, without exceeding the age of 23, it

is calculated in the amount of 75% for each descendant, but not less than the amount of the social allowance for the loss of the breadwinner;

2. To the surviving spouse if, at the time of the breadwinner's death or within 5 years after the death, he reached retirement age or was assigned to the degree of severe or accentuated disability, had at least 15 years of marriage with the deceased person and did not remarry, in the amount of 50%;
3. To the surviving spouse or the guardian (curator) who takes care of children under the age of 3 of the deceased breadwinner, during the periods of non-employment or being on leave to care for the child up to the age of 3, in the amount of 50%.

UML Diagrams

Also, In the course of creating TombTally, there have been employed Unified Modeling Language (UML) Sequence diagrams, the ones below will show how the user, app, server, and database interact, in order to provide information that users need from our app. They could be best used for representing the Log In and Sign Up sequences, as the other actions are better described by other types of diagrams.

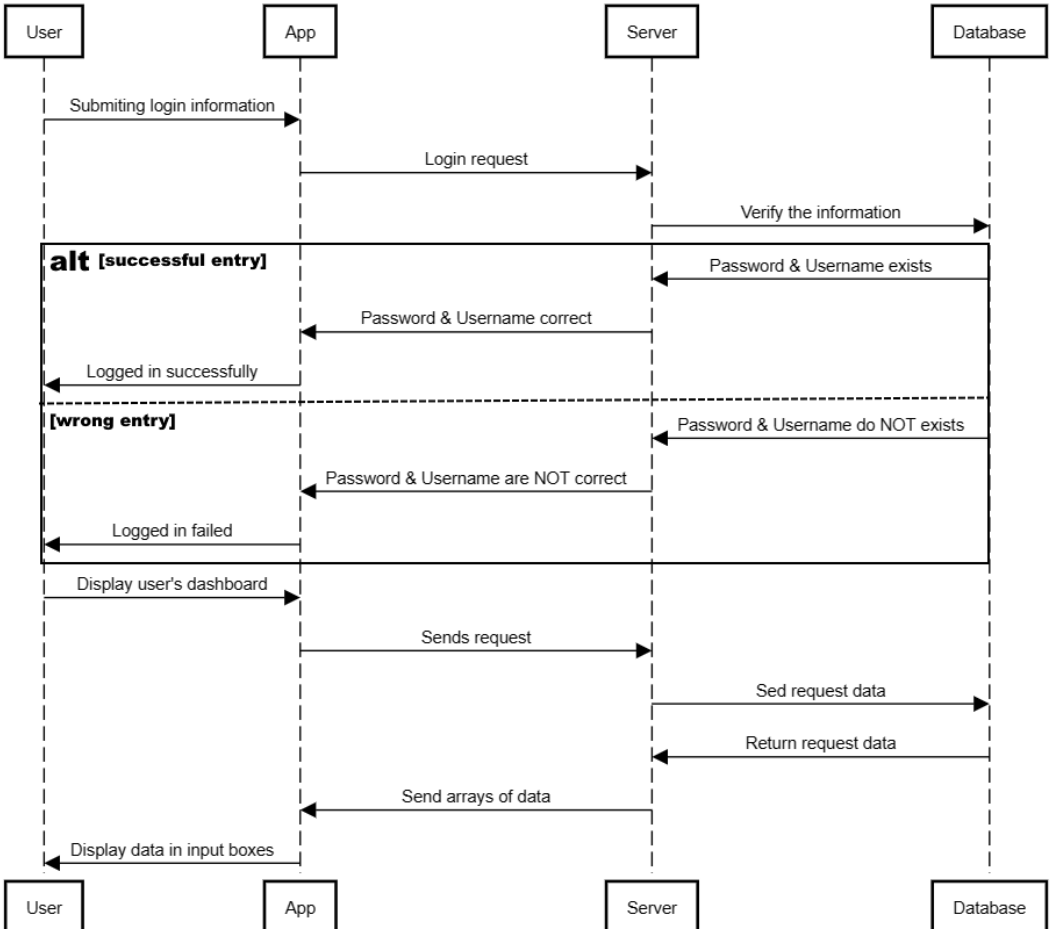

Figure 2.2.1 - Sequence Diagram - Logging in and displaying user's Dashboard

When a user tries to log in *(Figure 2.2.1)* he needs to introduce his login information (username, pass- word). This information is sent to the server and then to the database to verify if this user exists. If

the information is correct, the user is logged in successfully and the app will display the user's Dashboard. For this, it will send a request to the server and then to the database to get the list of categories with the following data (profile data, contributory data, salary data, non-contributory data). This list is sent to the app through the server and processed in the following way to display it in a user-friendly manner.

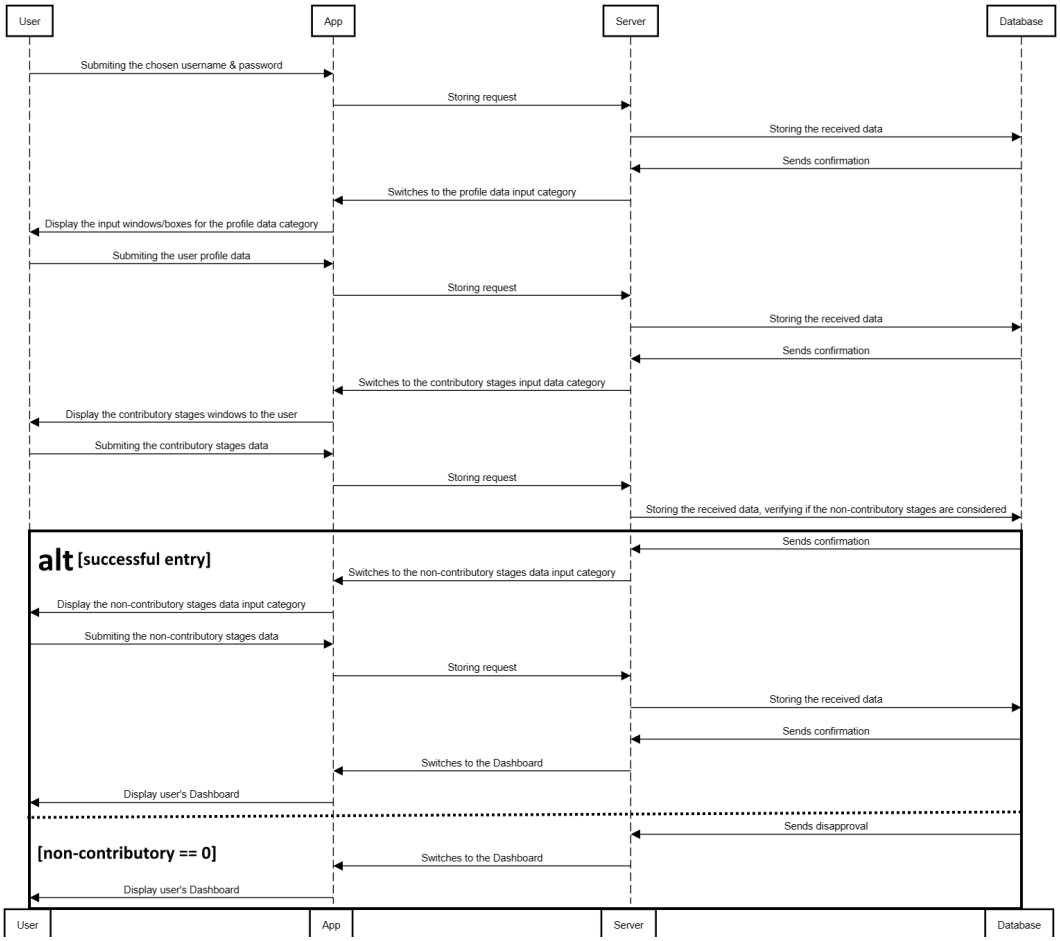

Figure 2.2.2 - Sequence Diagram - Signing in and displaying user's Dashboard

Another sequence diagram, *(Figure 2.2.2)*, reveals that a user has to sign up for the platform, in order to access the dashboard for the first time. It also shows the main interactions between the user, the app, the server and the database, regarding saving the personal data of the users in text files in the database. Moreover, a request regarding whether the non-contributory data should be taken into account or not is sent to the server and the database, so based on the received answer the non-contributory windows with the input fields could pop out or not.

Furthermore, a Unified Modeling Language (UML) Activity Diagram is another important behavioral diagram to describe dynamic aspects of the system. The Activity Diagrams below will show the dynamic aspects of some processes during the work activity of our application, TombTally, revealing the main flow when we are executing the C core program that lies in the server that's accessed by sending a JavaScript request to the Flask based server and executing the executable, C based file that will output our

pension's amount depending on the stored output. The output is fetched from the text files from the user's folder and displayed nicely on the dashboard's foreground.

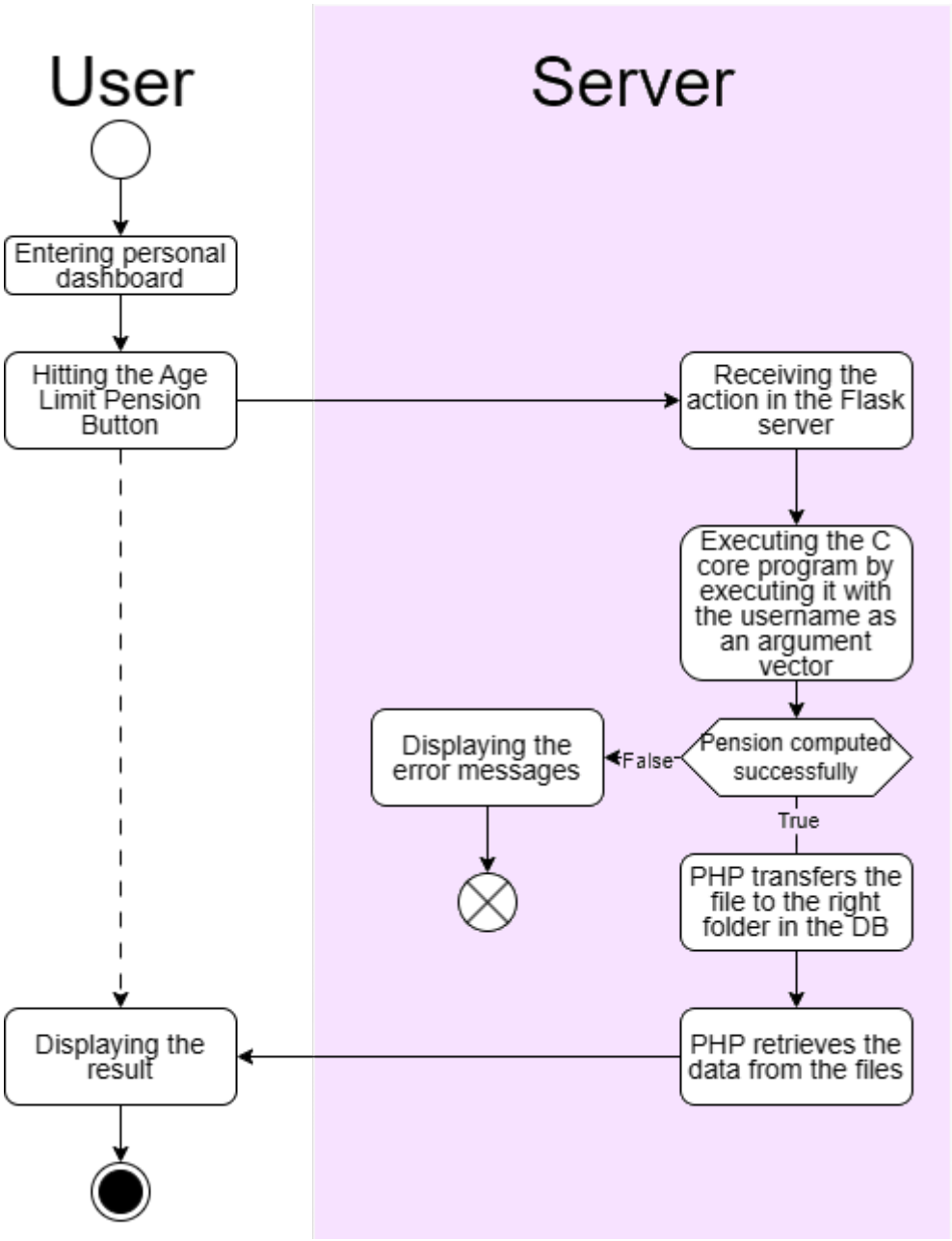

Figure 2.2.3 - Activity Diagram - Computing the Age Limit Pension

In the Activity Diagram from above, *(Figure 2.2.3)*, we can observe how the Age Limit Pension's result gets on the user's screen by using TombTally. If we had to comment on it, we can observe that once a user has entered the personal dashboard, he can hit the Age Limit Pension button, which computes the Age Limit Pension, and on the user's current screen the result will be displayed. However, although, it takes only a few seconds for the result to be displayed, and even though the user has to take only a few steps in order to get the result, on the Server side of the app a lot more complex processes are taking place. Once the the user hits the Age Limit Pension button, the request is received by the Flask based server, that also

executes the C program that computes the actual pension. Futher, if the pension fails to be computed an error message will be output, otherwise the PHP server will transfer the text file where the pension was saved by the C program, into the user's folder in the database, and after it retrieves the data from the files and it displays the result on the user's dashboard.

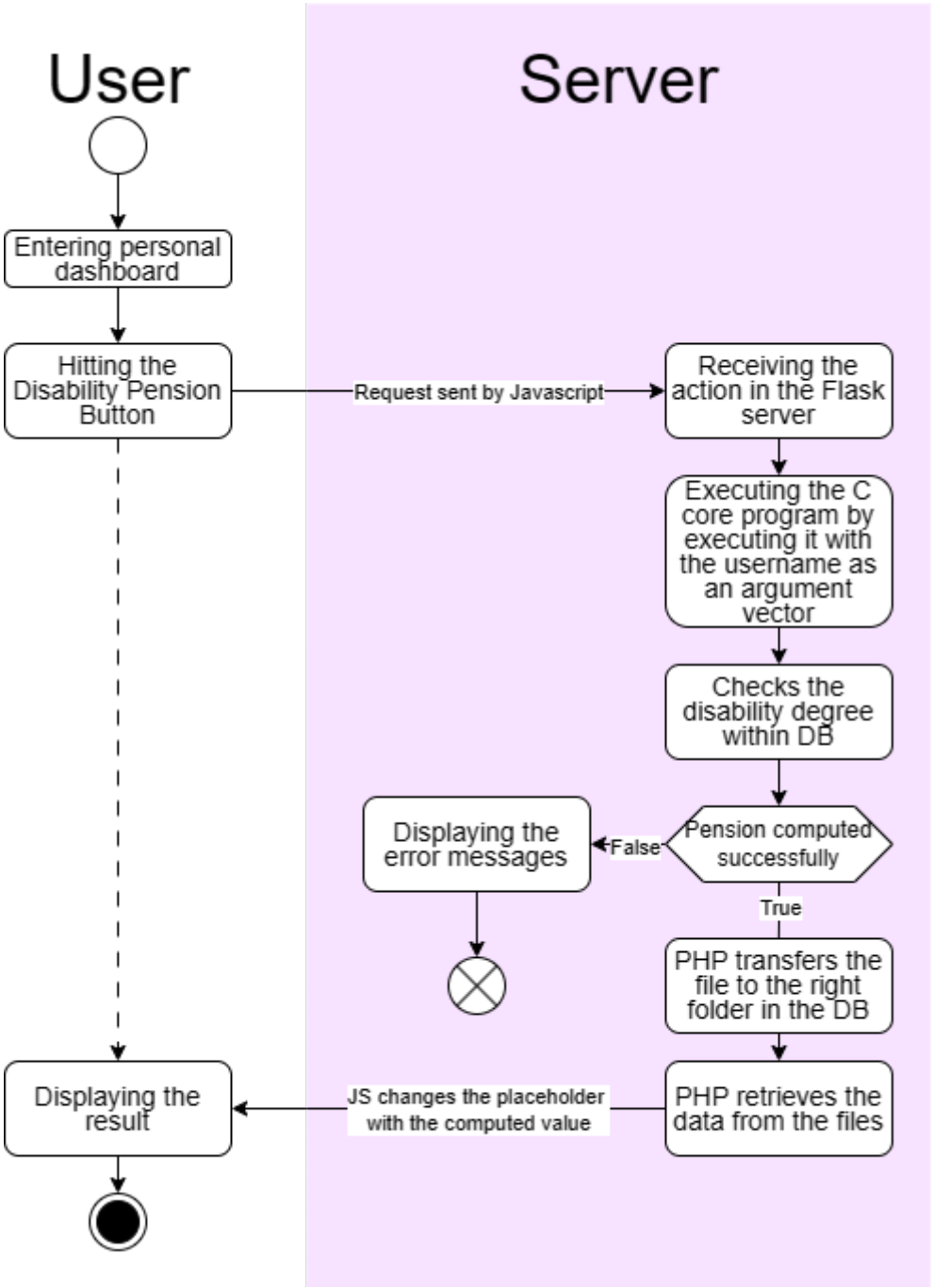

Figure 2.2.4 - Activity Diagram - Computing the Disability Pension

Another Activity Diagram, *(Figure 2.2.4)*, is the one that represents how the Disability Pension gets computed once the users hits the Disability Pension Button on the dashboard. After that action is done a request is sent to the Flask server by the Javascript function, which runs the C program and since the second

vector argument is ”2”, we would get as an output the Disability Pension. However, before reaching the output the C program, it also, checks the degree of the disability that was stored in the *user_login.txt* file, and based on that it will use the right rate towards computing the Disability Pension. In the end, if the program execution fails, we would get an error, otherwise, the PHP transfers the pension's result to the database from where we fetch using PHP the pension's amount and display it on the user's dashboard.

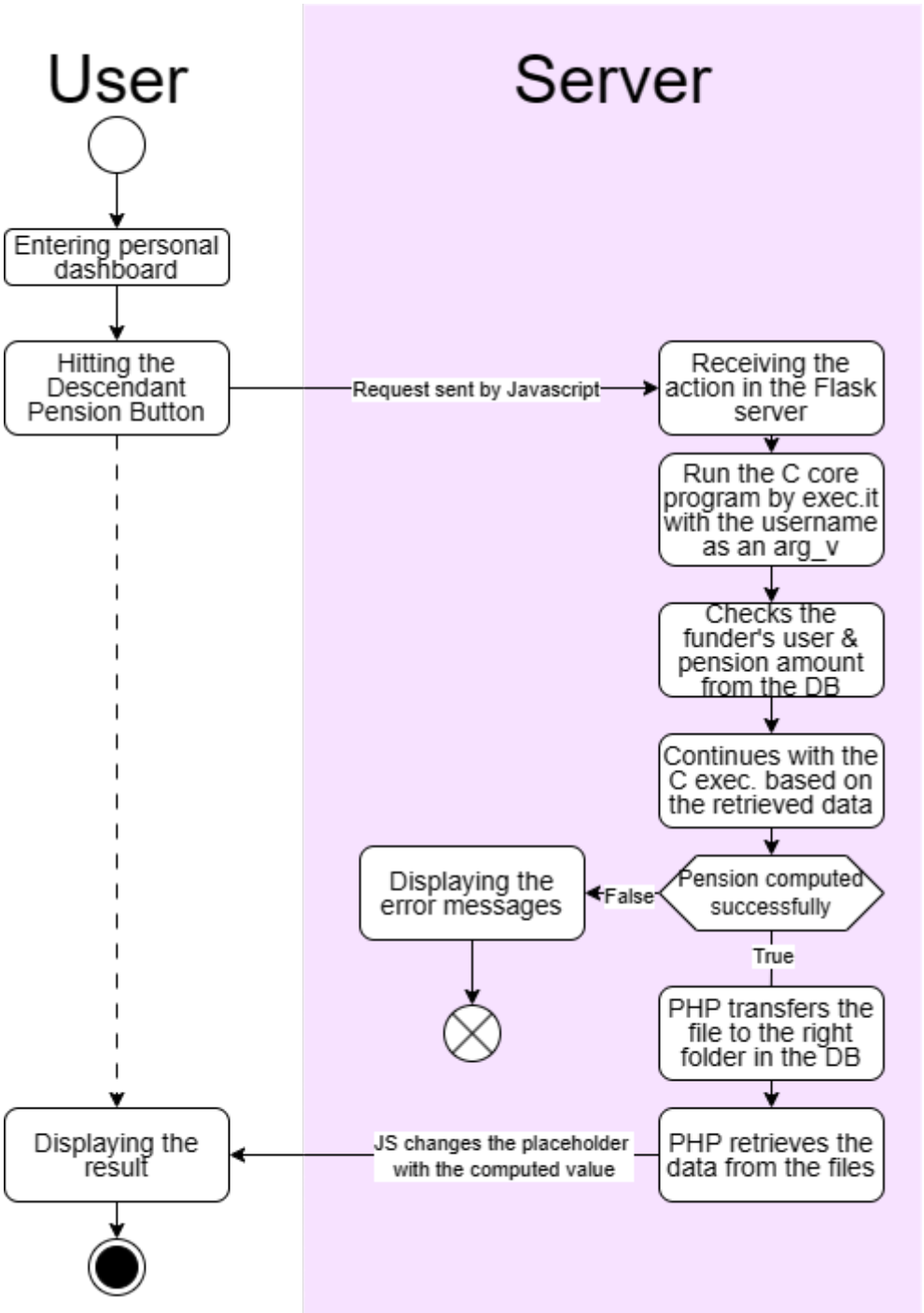

Figure 2.2.5 - Activity Diagram - Computing the Descendant Pension

The last Activity Diagram, we will focus on, *(Figure 2.2.5)*, is the one that represents how the computation of the Descendant/Survivor's pension takes place. As the previous pension types, the user hits

the button that's supposed to compute it, then the request is sent to the server where the C file gets executed. Nevertheless, this time based on the inserted funder's username, Age Limit Pension, and Disability Pension, we will use those values in combination with the descendant pension criteria, to compute the required pension, that's saved in a text file, from where the data is retrieved and displayed on the dashboard with the help of the PHP and Javascript.

Next, we will have a look at the Use Cases Diagram, that describes the high-level functions and scope of our system. This diagram also identifies the interactions between the system and its actor, in our case the user. The use cases and actor in the use-case diagram describes what our system does and how the actors use it, but not how the system operates internally.

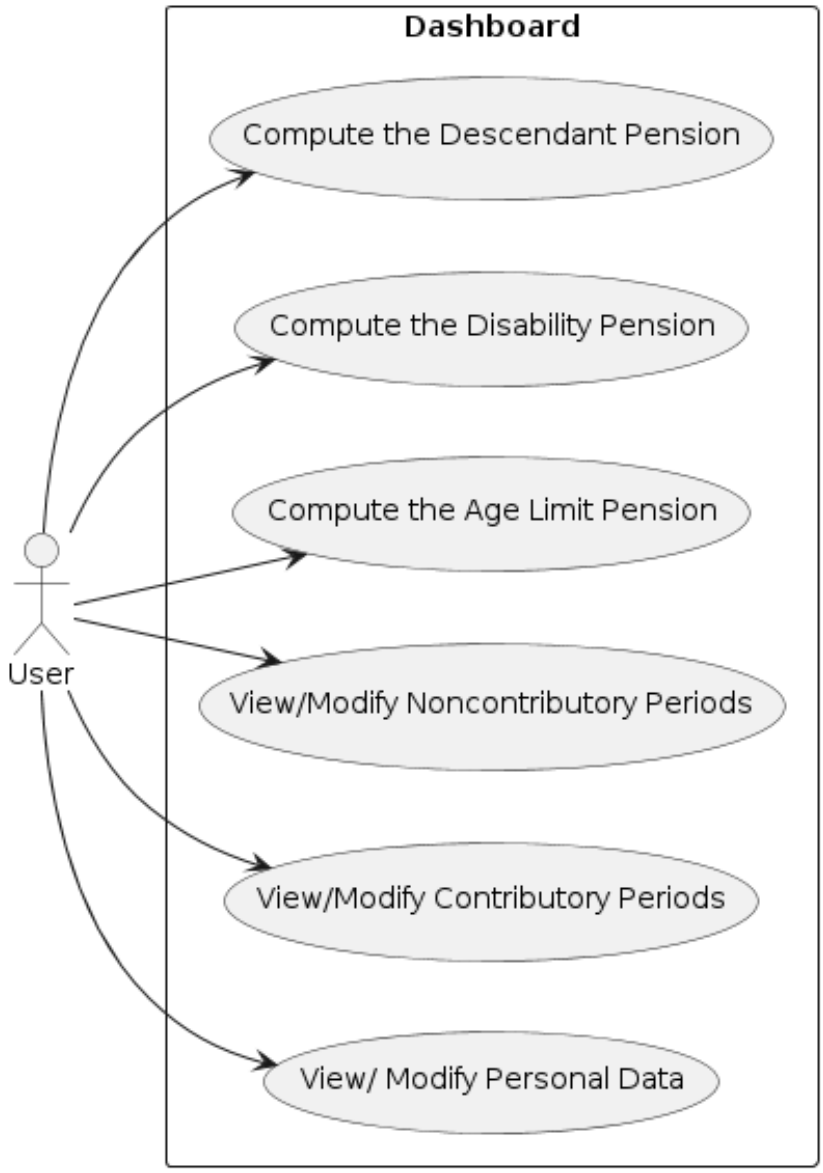

Figure 2.2.6 - Use Cases Diagram - Once on the Dashboard Page

In the Use Cases diagram from above, *(Figure 2.2.5)*, we can observe how the user could interact

with TombTally, once they are on the Dashboard page. As we can observe the main functionalities are the fact that they can compute the three main types of pensions: Age Limit Pension, Disability Pension, and the Descendant/Survivor's one. Furthermore, the user can view and modify the non-contributory stages, which is a feature that is not existent in many solutions that were presented by other people. Moreover, the user can view and update the contributory periods, and on top of that, they can also modify the salary data from those periods, that was rather complex to implement. And last but not least, they can change their personal data that will have an impact once the pensions are computed again.

File Organization

In the current section, we will examine the file organization of TombTally, examining the purpose and functionality of each code file within the application. This analysis will highlight the meticulous and well-thought-out organization of files, showcasing how different components of TombTally interact to deliver seamless pension calculation functionality. By comprehending the file structure, we will uncover the specific roles and responsibilities of key code files, such as modules for pension calculations, user interface components, data storage management, and more. Understanding the file organization of TombTally provides valuable insights into the underlying architecture that supports the platform's robustness, scalability, and maintainability.

Figure 2.3.1 - The main languages used to create TombTally

The main languages used to create TombTally, as seen in the picture *(Figure 2.3.1)*, are:

- *C* - that is used as the core of TombTally, since the main program that computes the pensions, and solves the stated problem, is written in this low-level language, as it is a great way to achieving fast mathematical computations and work with the data stored from the database;

- *HTML5* - this markup language, allowed us to create the bare bone skeleton of the graphical graphical interface for our platform, on top of which we developed different functions, interactive parts and a pleasant design;
- *CSS* - this style sheet language helped us in creating a slick and modern interface for my platform, giving it an unique look and a n up-to-date look;
- *Javascript* - this programming language aided us in giving functionality to most of the buttons on TombTally, by either communicating to the PHP or Flask server what files to run or what data store, or by saving the input data in various arrays, or it was used in making the UI more dynamic, rather than leaving it just as static;
- *PHP* - was mainly used to run the main server, also, the PHP files they are the ones that take the data from the Javascript arrays and store them in specific files in the users' folders. Moreover with their help we can load the data from the database and display the computed pensions on the dashboard;
- *Python* - mostly Flask, was used to create another server that was running on the same port as the PHP one so that we could run the executable files that were written in C in pair with the argument vectors, which are the username of the user and the number of the pension we are interested in.

Now let's examine the file organization of TombTally and analyze briefly some of the files and in depth the other, plus we will see what they do and why they are needed.

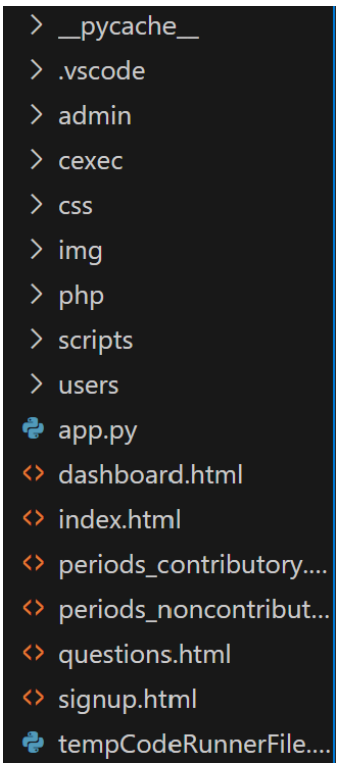

Figure 2.3.2 - The TombTally's Content

As we can observe in *(Figure 2.3.2)*, everything is organized in a nice form, the main files that are

outside folders are the HTML5 ones and the flask server file. Talking about the html files, the *index.html* is the bare bone skeleton of the log in page, where *signup.html* is self explanatory by its name, showing that it's the skeleton of the sign up page. In continuity, the *questions.html* is the file that acts as the back bone of our questionnaire used to gather the personal data regarding the user, such as name, surname, and others. Futhermore, *periods_contributory.html* & *periods_noncontributory.html* are the pages for gathering data regarding the stages of contribution and the salary and respectively the page used to save the data about the non-contributory periods done by the user. And last but not least, the *dashboard.html* is the main page of our platform where the pension amount gets computed and the result gets displayed.

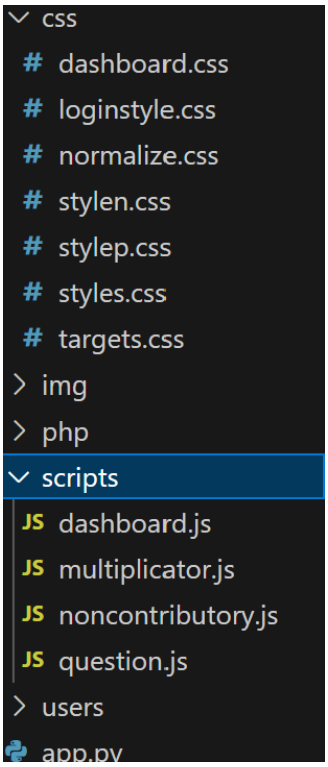

Figure 2.3.3 - The *css* and *scripts* folders' content

In the figure *(Figure 2.3.3)*, we can see that in the *css* folder are store the cascading style sheets, that are linked to the html files, and in order to see which one is connected to which html file, we would have to analyze the content of the HyperText Markup Language files. Moreover, in the *scripts* folder are stored the Javascript files, that are also linked to the html files, the *question.js* is used in the form that gathers the data of the users regarding the personal information; the *multiplicator.js* is used on the page where the contributory-stages and salary related data are stored in arrays and futher transfered into the database, the file it's called in this way as it contains complex functions that multiply some html divisions regarding the input of data, based on some input parameters. The *noncontributory.js* file contains the main functions regarding storing the data regarding the non-contributory periods of the users. And finally, the *dashboard.js* file, that's also the biggest in terms of the amount of code lines it contains and complexity, has the functions

required to update/modify/view the stored data and also the ones that trigger the execution of the C files with the help of the Flask and the PHP server.

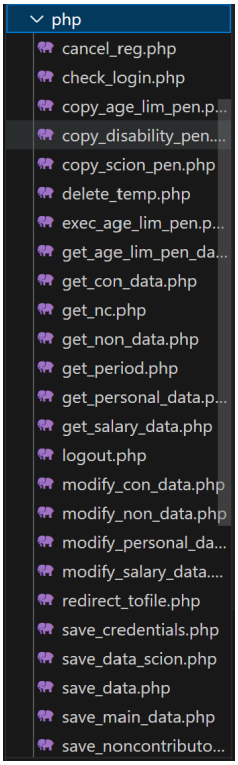

Figure 2.3.4 - The *php* folder' content

The screenshot from above, *(Figure 2.3.4)*, it does indeed contain a lot of files, as our app does require lots of interaction with the database. The php files that start with *save_* are mainly used to save the elements of the Javascript arrays into text files in the DB, while the ones that begin with *modify_* are used to modify/update the existent elements/fields within the database, and the ones that start with *get_* have the functionality of retrieving data from the DB and displaying it on the screen, at the same time the files that are named with *copy_* are being implemented with the aim of transferring the text files that were output by the C program into the right place of the database.

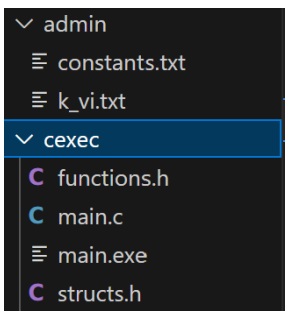

Figure 2.3.5 - The *admin* and *cexec* folders' content

In *(Figure 2.3.5)*, is displayed the content of two important folders, the first one being the *admin*

folder, where we have two files, in *constants.txt* are stored some data regarding the minimum pension of a specific year or the pension age, that's futher used in the C program for the computation, and in *k_vi.txt* is stored the data regarding the valorization coefficients of insured income that were set by the government[6].

In the *cexec* folder we can notice that we have the core of our app, the *main.c* file and the header files, *structs.h* in which are stored the struct declarations and *functions.h* where the functions are stored. After compiling the C file we can run the *main.exe* executable that computes the pension.

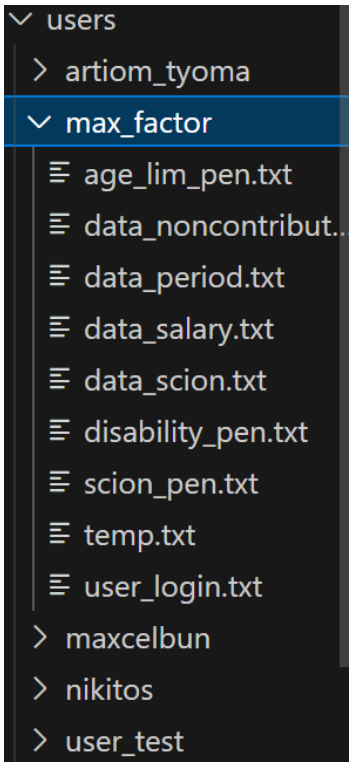

Figure 2.3.6 - The *users* folder's content

The last folder we will look at is the *users* folder, *(Figure 2.3.6)*, which is our database. In this folder, once a user creates an account, a folder is also created in the *users* folder with the user's username as a name for the folder, and while the user goes through the sign up process, it's folder in the database gets gradually bigger in terms of content. Therefore, in the *user_login.txt* file we have the user's username, password and personal data stored, in the *data_period.txt* we have the information regarding the contributory periods of the user, in the *data_noncontributory.txt* we have the data about the non-contributory stages, and in the *data_salary.txt* are stored all the salaries the user has received during his contributory periods. Moreover, the some files get added after executing the C program, such as *age_lim_pen.txt* in which we have the Age Limit Pension amount to be received by the user, the date he got the pension rights and the average insured monthly income, while in the *disability_pen.txt* we have the information related to the Disability Pension amount and the average insured monthly income, and in the *scion_pen.txt* we have the data regarding the Descendant/Survivor's pension amount to be received, and in the *data_scion.txt* is stored

the funder's username, his Age Limit Pension and Disability pension, that's used in the computation of the Descendant/Survivor's pension. Moreover, all of the data is modifiable through the dashboard page, and once a modification happens it also updates the data from the database.

Visual Representation

In this section we will focus on the significant part of the UI design of the application that will be represented in different simple parts so that everyone can perceive the idea of *TombTally*. The following pictures will represent the deep functionality of the platform.

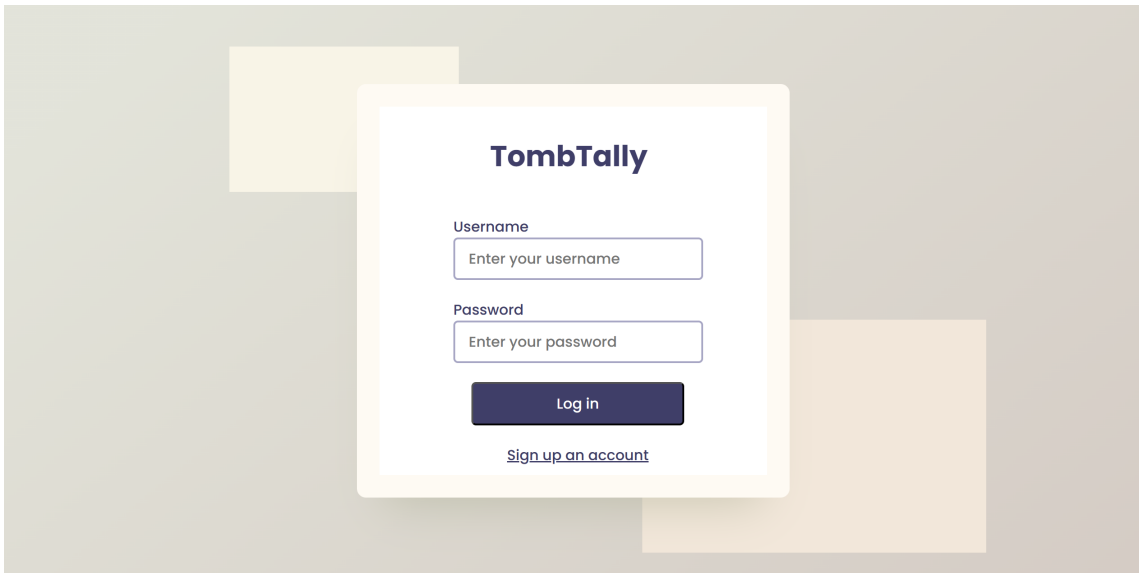

Figure 2.4.1 - The Log In Page

This Log In page, *(Figure 2.4.1)*, is very similar to the Sign Up one, here users can input their username and password in order to access the dashboard.

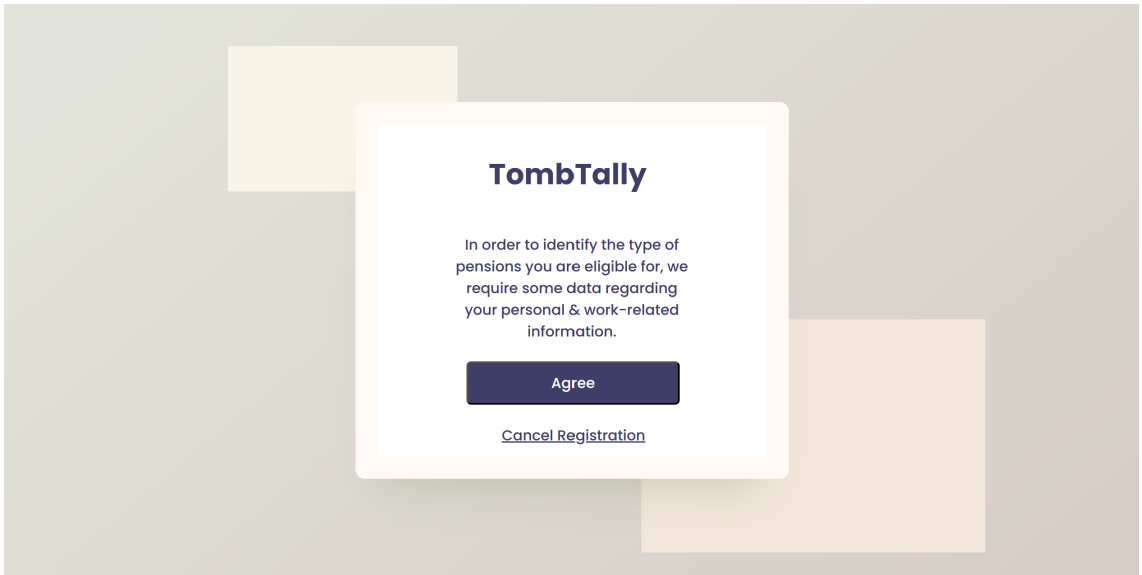

Figure 2.4.2 - The Start of the Signing Up Process

This page, *(Figure 2.4.2)*, is only visible upon attempting to sign up on TombTally, the user having to agree with the terms and conditions of gathering personal & work related data about them. If the user does not accept the terms and conditions, he can simply hit *Cancel Registration* which will deleted the recorded data during the signing up process, so that he could either leave the app or have a fresh start.

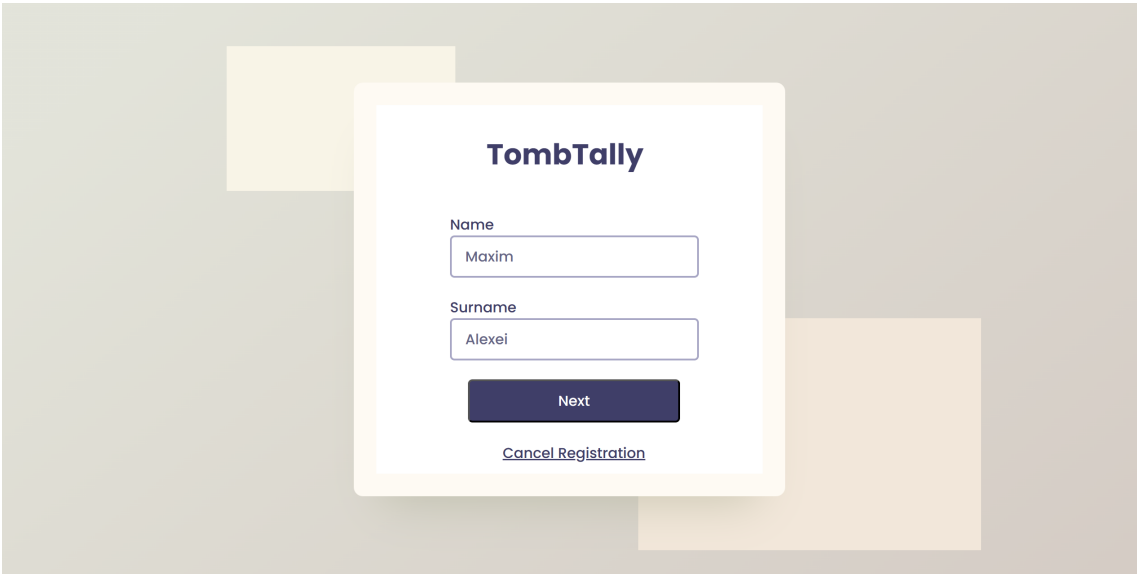

Figure 2.4.3 - Sign Up Process - Entering the Name and the Surname

Next, above we can observe, *(Figure 2.4.3)*, that the user has to input their birth name and surname, which gets stored in the database in their personal folder. This can be used by the officials in confirming their identity together with the help of other parameters.

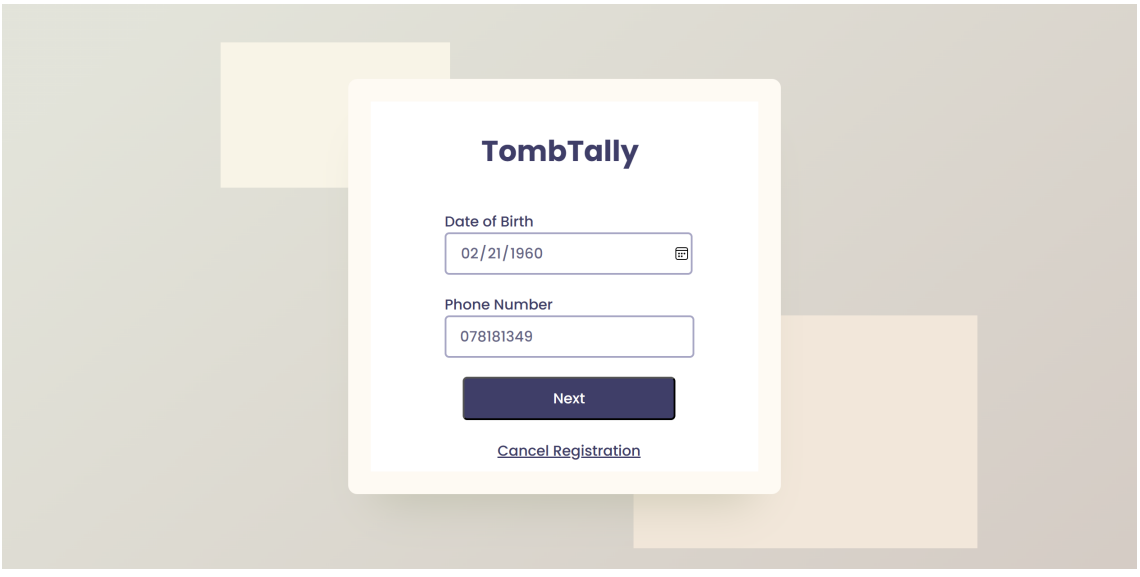

Figure 2.4.4 - Sign Up Process - Entering the Birth Date and the Phone Number

On this window, *(Figure 2.4.4)*, the user has to insert their birth date, that is used later to compute their age and then what their retirement date will be. On top of that, the phone number could be used by the

officials to contact the potential pension benefiters, in case there are some issues or if they have to inform them regarding any changes.

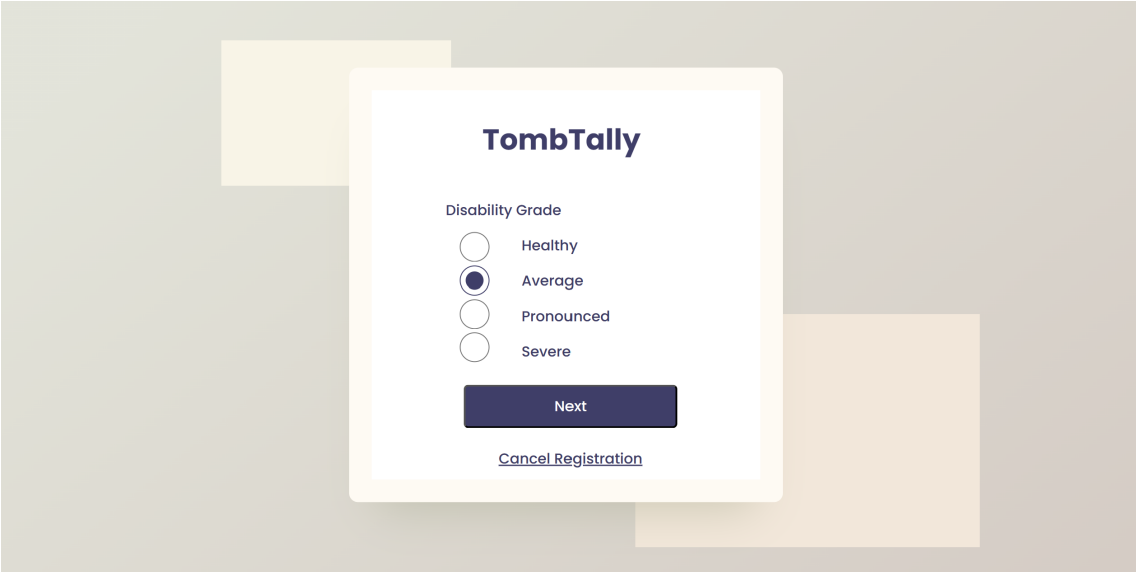

Figure 2.4.5 - Sign Up Process - Choosing the Disability Degree

Futher on, the user will eventually stumble across this window, *(Figure 2.4.5)*,that will make them choose whether they were tagged as disabled or healthy, and in case they are indeed suffering of disability, they can choose the degree of disability they have, which ranges from Average to Pronounced and to Severe, which is the worst case. This data will have a vital role in our disability pension computation, but also if the user will be named as a funder of someone's descendant/survivor's pension. On top of that, if a disability grade is chosen, the user will also have to insert the date they've been diagnosed as disabled.

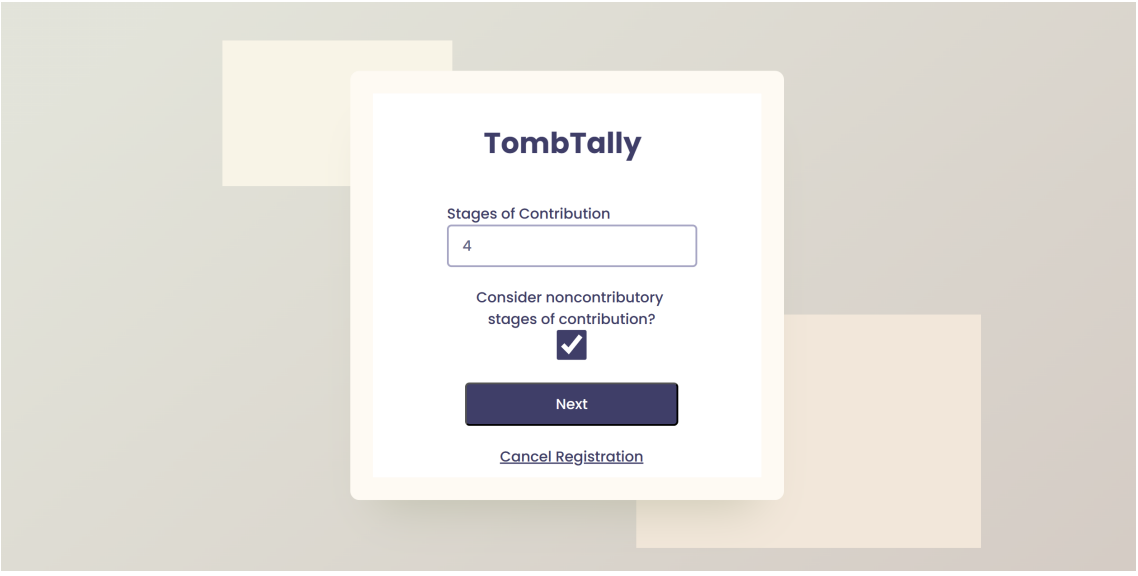

Figure 2.4.6 - Sign Up Process - Entering the stages of contribution and choosing whether the non-contributory stages should be considered or not

The window from above, *(Figure 2.4.6)*, is the last window of the collection of fields that have the aim of gathering the personal information regarding the user itself, this information until and including it, is saved in the *user_login.txt* file in the database. As we can see the user has to input the amount of contributory stages/periods he has been working for and paid taxes to the state, and whether he desires to take in the consideration the amount of contributory periods or not.

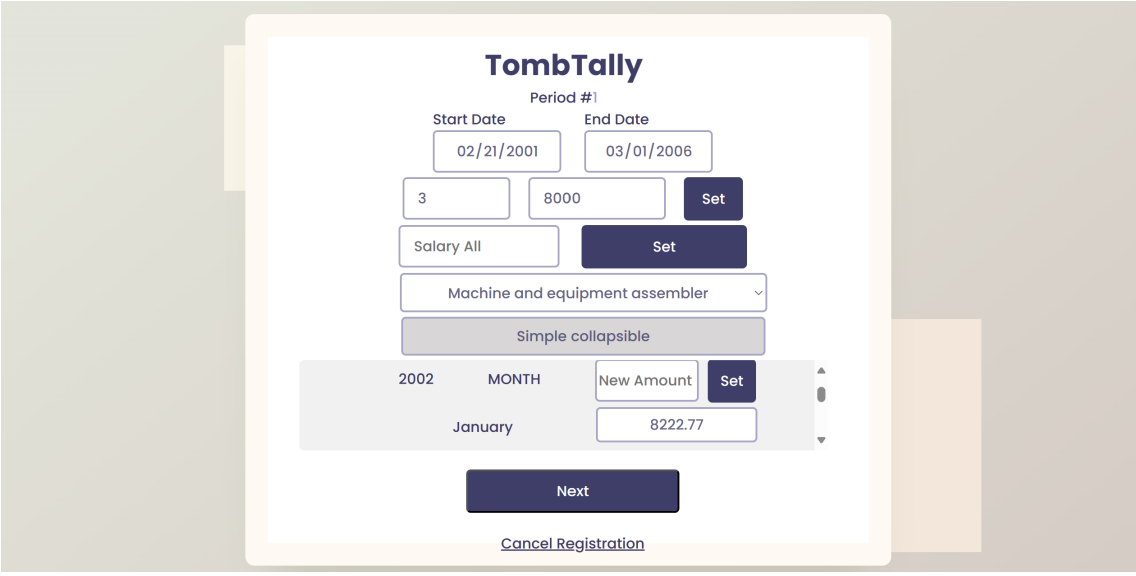

Figure 2.4.7 - Sign Up Process - Entering the Data Regarding the Period of Contribution

In the picture from above, *(Figure 2.4.7)*, we can observe how the UI of the page where the user has to input the data regarding the comtribution period looks like. We can observe the period number we are on, which increments by one once we hit ”Next”. As we can see the user is required to enter a date range, set a salary and choose the type of profession he had during this period.

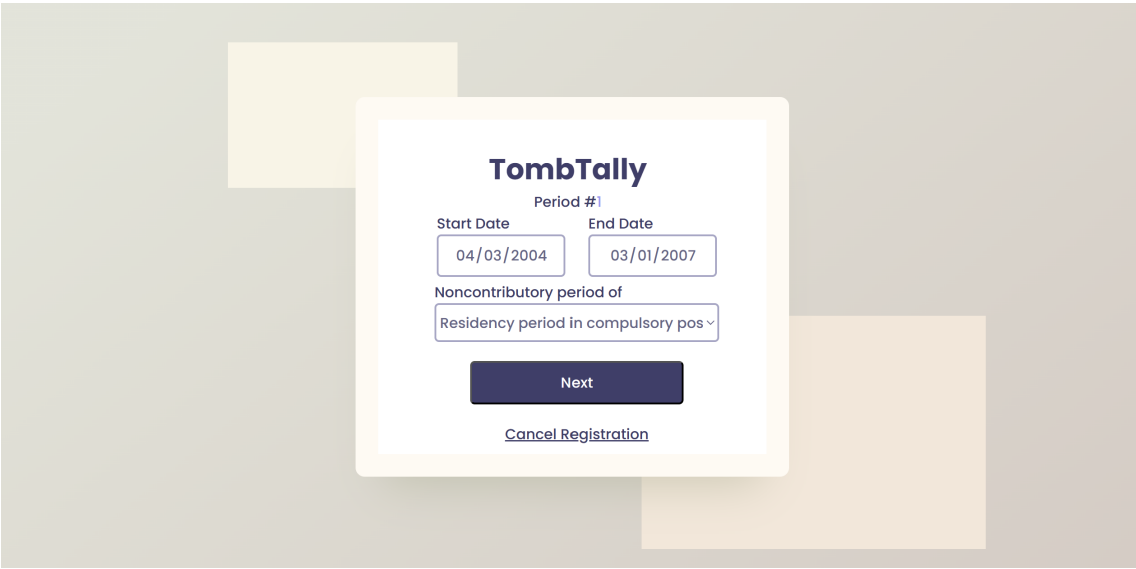

Figure 2.4.8 - Sign Up Process - Entering the data regarding the Non-Contributory Periods

In continuity, this window, *(Figure 2.4.8)*, is only reachable if the user has gone through all of the contributory periods that were chosen, and they have ticked the box that lets them input and non-contributory periods data. As we can see they have to chose a specific range of dates when the non-contributory periods took place and the type of activity they were doing during those dates.

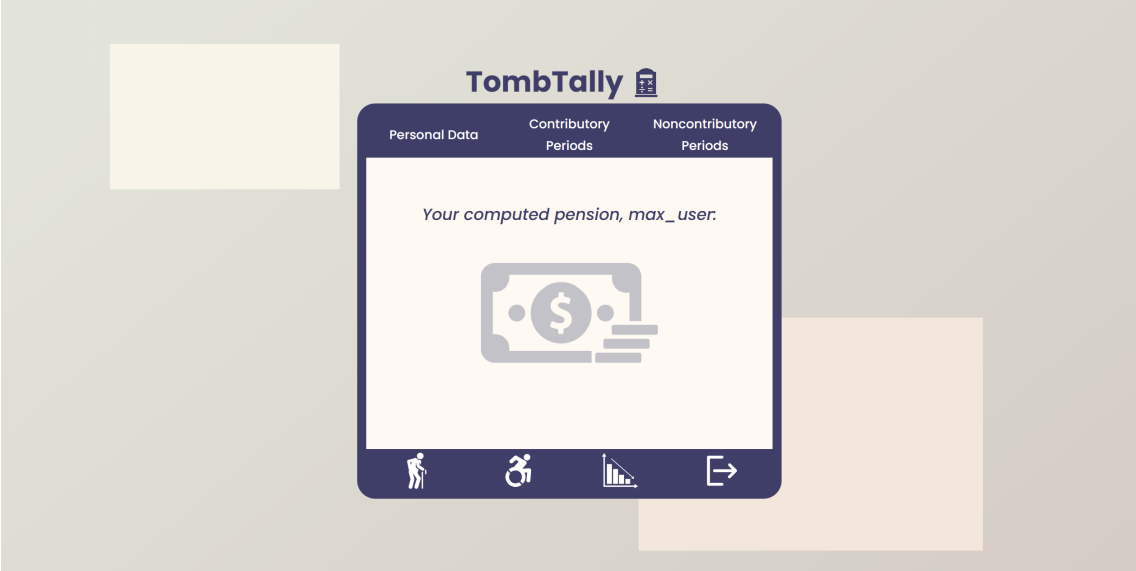

Figure 2.4.9 - Dashboard - Viewing the dashboard

Finally, once the user is done with the signing up process they can access the dashboard, *(Figure 2.4.9)*, or it can simply be accessed if the user logs in with an existent account.

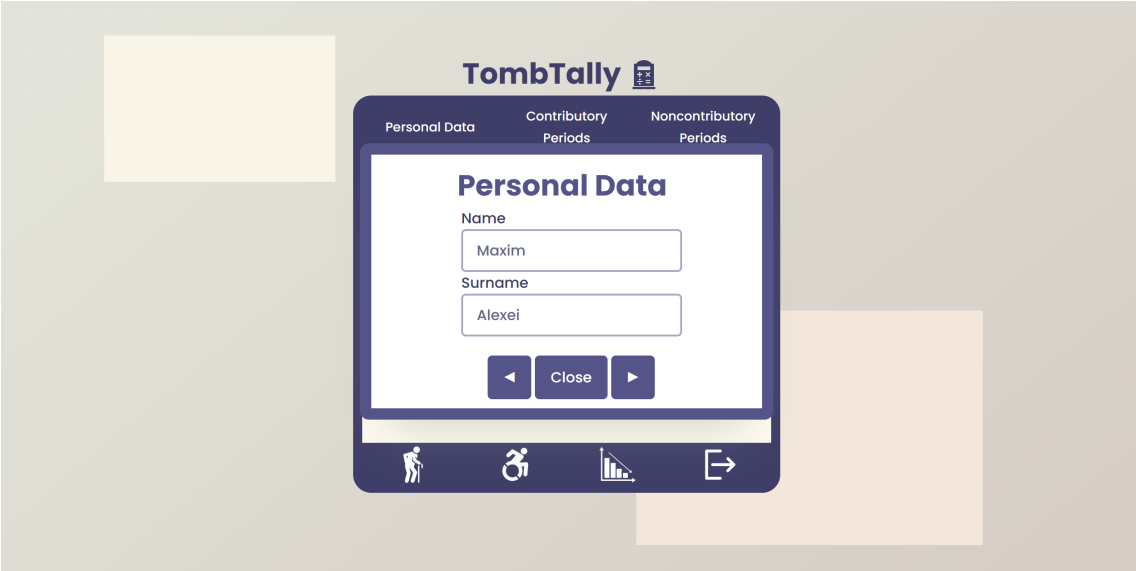

Figure 2.4.10 - Dashboard - Viewing/Modifying the Personal Data

While on the dashboard page, the user can also open the Personal Data window, *(Figure 2.4.10)*, where they can view the current saved personal data from the database. On top of that, they have the amazing possibility of modifying the information that will automatically save it into the database once *Close* button

is pressed. This is valid for the Contributory Periods, *(Figure 2.4.11)*, and for the Non-Contributory Periods windows, *(Figure 2.4.12)*, which gives a lot of flexibility regarding the input data to the user, making the whole experience of using TombTally quite pleasant.

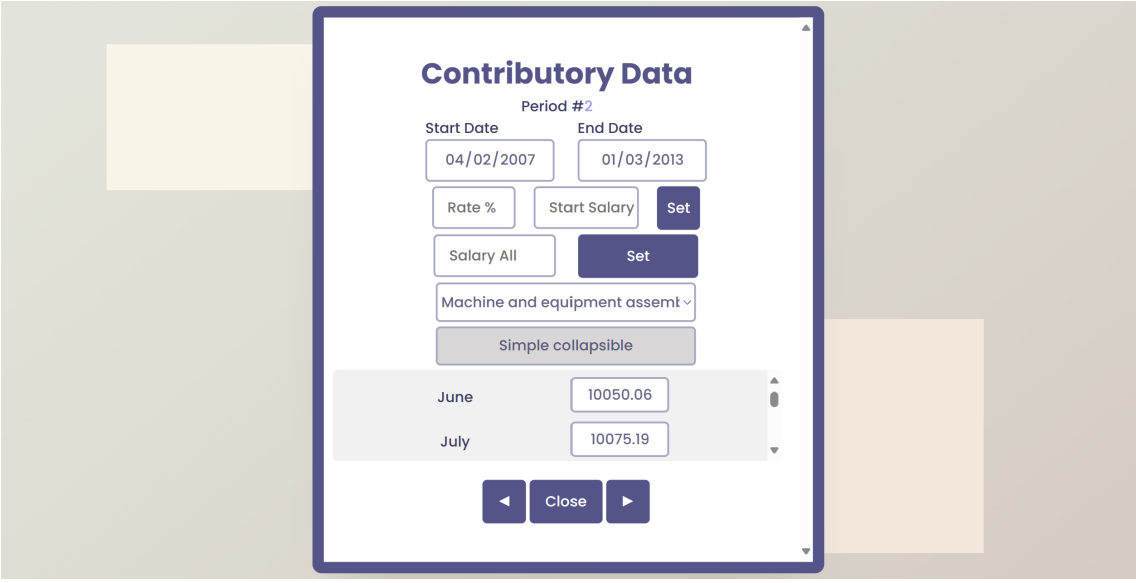

Figure 2.4.11 - Dashboard - Viewing/Modifying the Contributory Data

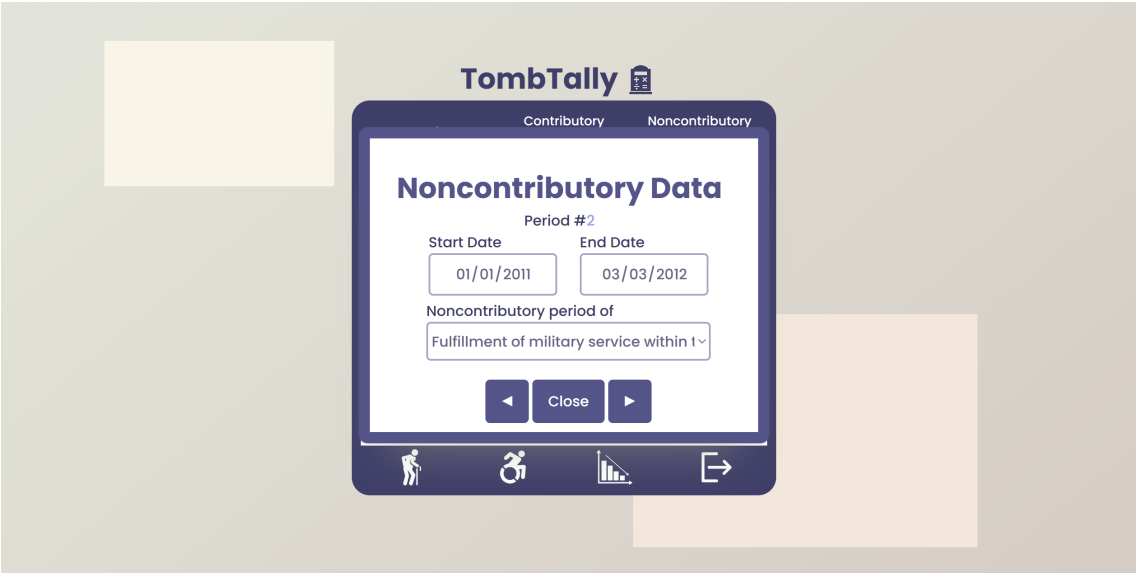

Figure 2.4.12 - Dashboard - Viewing/Modifying the Non-Contributory Data

The following pictures will show us how the Age Limit Pension amount is displayed on the screen, *(Figure 2.4.13)*, and how the Disability Pension, *(Figure 2.4.14)*, is displayed as well, together with other parameters. The data is taken from the database, more specifically from the logged user's folder, where the text files were saved after the execution of the core C based program that does all of computation based on the formulas that were stipulated above. This data is rather accurate, if the input is right and realistic as

well, as the program follows with great accuracy the mathematical formulas. Therefore, other parameters than only the pension amount are displayed on the dashboard, which gives the users more information about their pension planning.

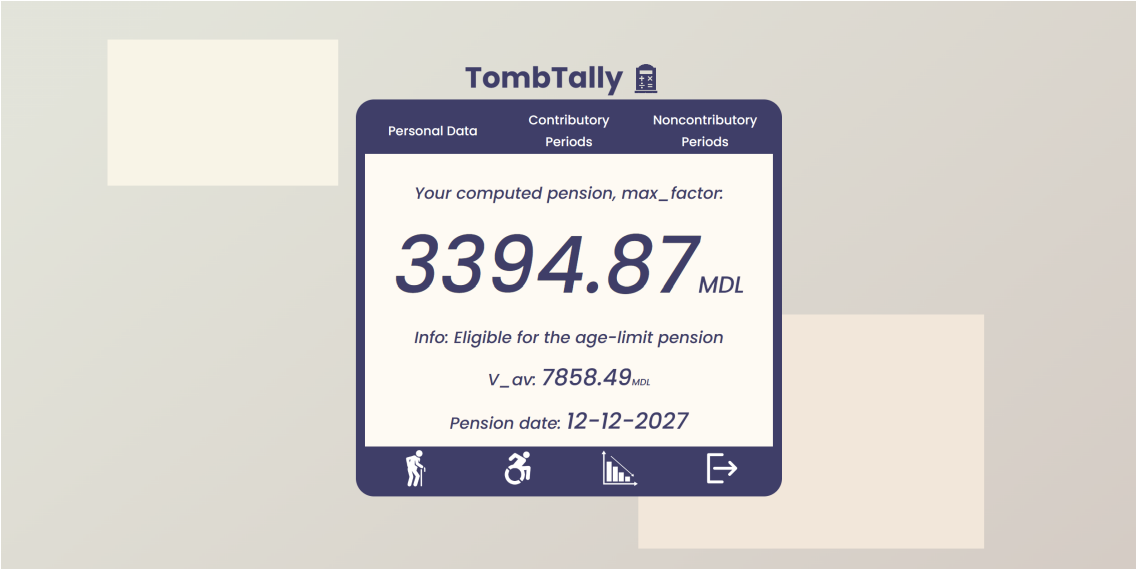

Figure 2.4.13 - Dashboard - Viewing the computed Age Limit Pension

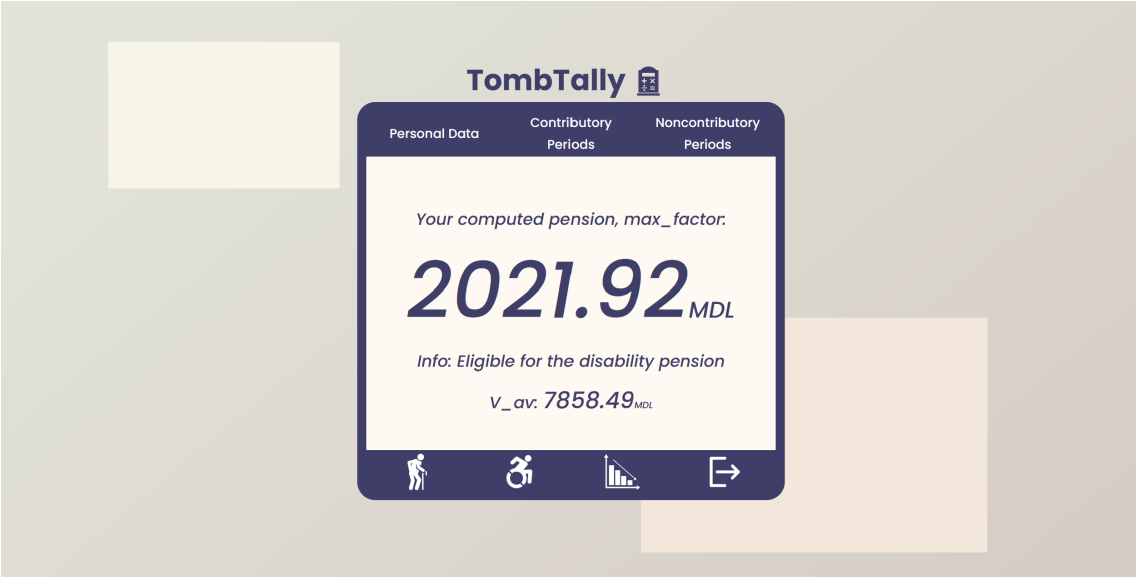

Figure 2.4.14 - Dashboard - Viewing the computed Disability Pension

In continuity, we will see that in order to display the amount of the Descendant/Survivor's Pension is rather different from the Age Limit and Disability pension, as the user has to enter the funder's username that they will take a percentage from their Age Limit or Disability pension, as seen in the picture below *(Figure 2.5.1)*, in order to compute their Descendant pension. On top of that, they have to pick a criteria that best describes the reason and it justifies why they have to receive the Descendant Pension. After that, once they press the *Close* button, the C file will be executed, computing the Descendant/Survivor's pension

amount, that will finally get displayed on the user's dashboard, *(Figure 2.4.16)*, just like the Age Limit pension or the Disability one.

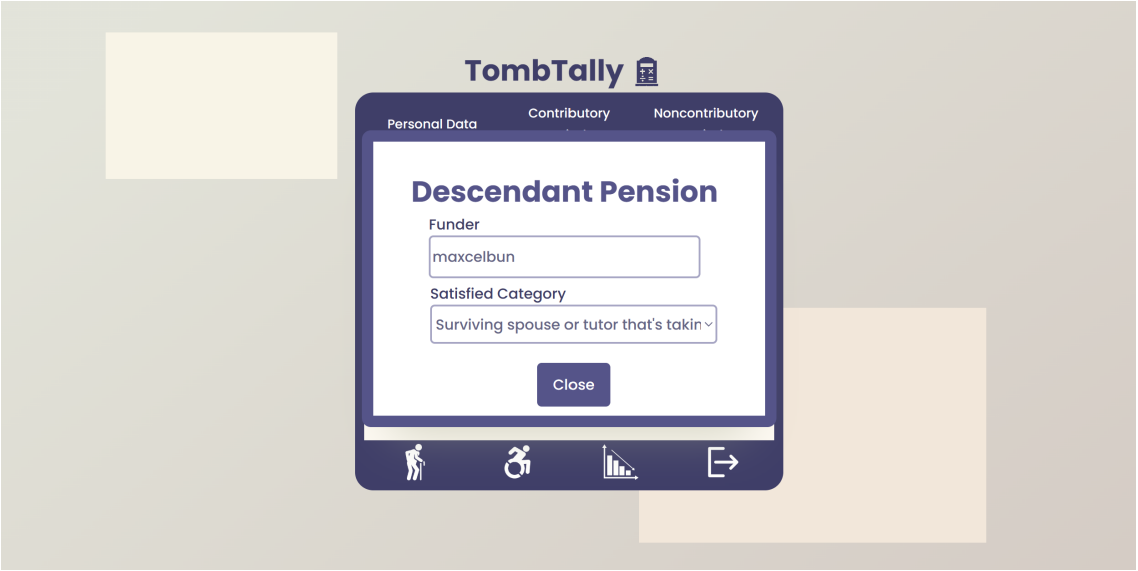

Figure 2.4.15 - Dashboard - Selecting the Criteria and Input the Funder's Username for the Descendant Pension

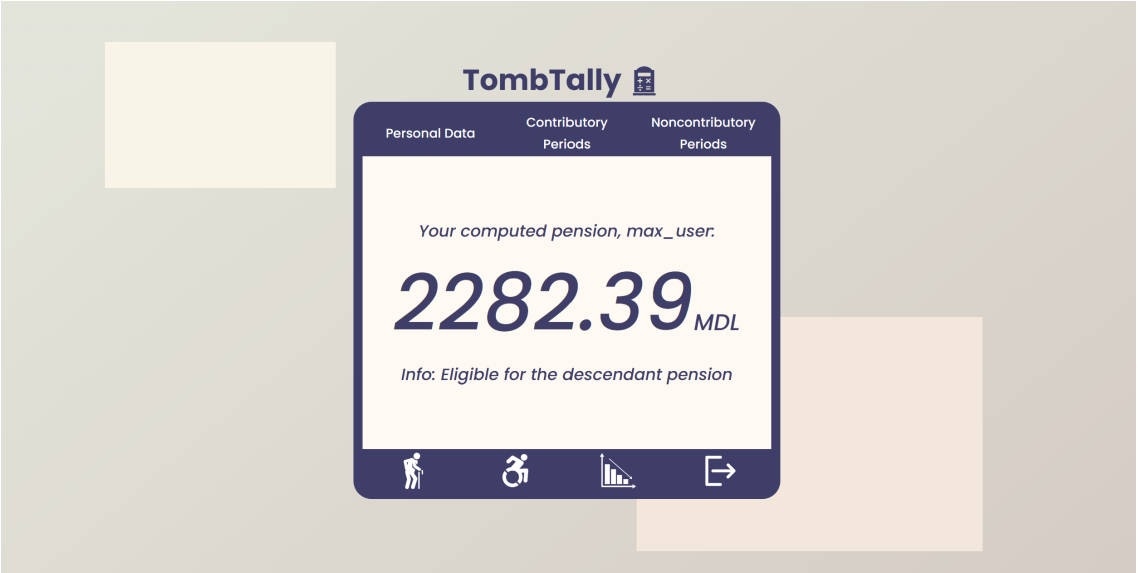

Figure 2.4.16 - Dashboard - Viewing the computed Descendant/Survivor's Pension

Therefore, those have been the main Visual User Interface elements of TombTally that could be presented through png formats. Besides the static images, if we had to actually use the app, we would also observe some pleasant fade-in, fade-out animations with it once we switch the windows or on clicking the buttons. Also, when the pension gets computed a loading animation plays over the dashboard, that's a pleasant touch, which could improve the overall user experience with TombTally.

Source Code

This section's main purpose, is to clarify the fact that TombTally is open source and that it's code can be used in any other projects. Moreover, below we can see the Github repository[7] being shared.

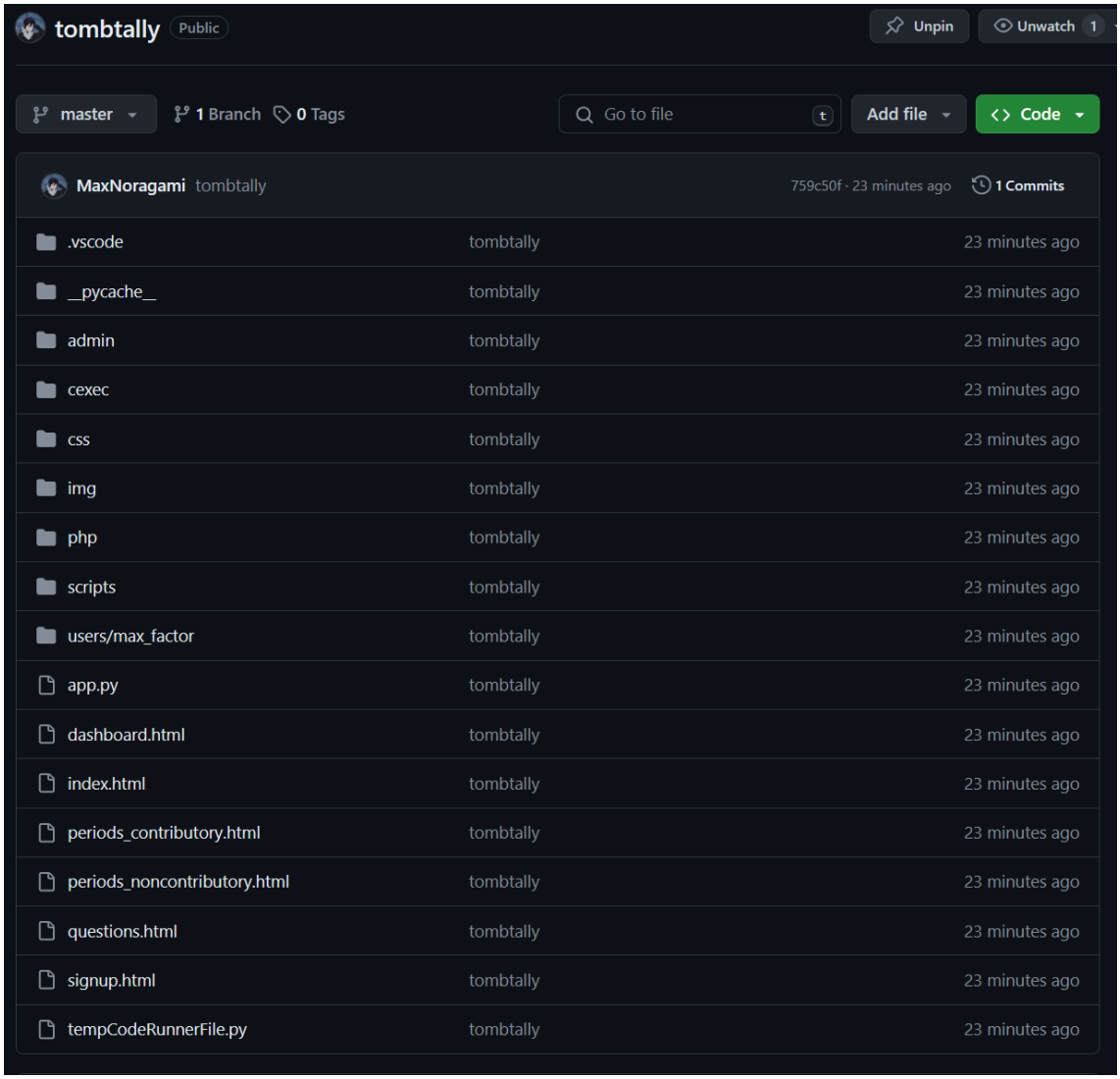

Figure 2.5.1 - The Github page where the source code can be accessed[7]

Above, we can observe that all of files that TombTally consists of were uploaded on the Github repository[7], that has the same name as our platform. There is also a *readme.md* file that can give the users some intructions regarding how the program should be run and what are the requirements so that the execution will go smoothly with no issues.

¿ Introduction ¿ Scenarios ¿ Error Handling ¿ Future Plans ¿ Conclusions

Conclusions

To sum up, dealing with the problem of traffic jams requires new and creative solutions that can improve cities around the world. Using strategies like green corridors is a good way to reduce traffic and make transportation systems work better.

By adjusting traffic lights and creating green corridors, cities can have smoother traffic and more reliable transportation. This not only saves time stuck in traffic but also reduces air pollution and helps create a more sustainable future for our cities.

Looking ahead, it's important to adopt green corridors and other smart transportation ideas. By working together and using technology, cities can become places where traffic moves smoothly, making it easier for people to get around. Let's work together to create a better, greener future for everyone.

Bibliography

[1] Ziarul de Gardă. ”Who (actually) will benefit from the minimum pension of 2778 lei and the average pension for the age limit of 3926 lei”. https://www.zdg.md/stiri/cine-de-fapt-va-beneficia-de-pensia-minima-de-2778-de-lei-si-de-pensia-medie-pentru

[2] CNAS. ”Pension Calculator User Guide”. https://acces.cnas.md/ac/resources/doc/Ghidul%20utilizatorului%20Calculator%20pensie.pdf.

[3] CNAS. ”Retirement age calculator”. https://cnas.md/calc/index.php.

[4] gov.md. ”Regulation on the Method of Calculating Pensions”. https://gov.md/sites/default/files/document/attachments/intr24_43.pdf.

[5] CNAS. ”Disability Pension”. http://www.cnas.md/tabview.php?l=ro&idc=535.

[6] Legis.md. ”Valorization coefficients of insured income for 1999-2023”. https://www.legis.md/UserFiles/Image/RO/2024/mo118-121md/anexa%20nr_3_165.docx.

[7] Github. ”TombTally”. https://github.com/MaxNoragami/tombtally/tree/master.