

**Universitatea Tehnică “Gheorghe Asachi” din Iași**  
**Facultatea de Automatică și Calculatoare**  
**Domeniul Calculatoare și Tehnologia Informației**  
**Specializarea Tehnologia Informației**

## **MANAGEMENTUL PROIECTELOR SOFTWARE PROIECT**

**Tarlev Mateo, Grupa 1409B**  
**Budacă Marius-Andrei, Grupa 1409B**  
**Pintilie Răzvan-Nicolae, Grupa 1409B**  
**Dumea Cristian, Grupa 1409B**  
**Imbrea Daniel, Grupa 1409B**  
**Ciocan Stefan-Robert, Grupa 1408B**  
**Bejinaru Mihnea-George, Grupa 1408B**  
**Chirca Radu-Iulian, Grupa 1408B**

**An universitar 2023-2024**

# **Etapa 1**

## **Consolidarea echipei**

### **Echipa 7**

#### **Tarlev Mateo (Project Manager)**

The Leadership Motivation Assessment: 56 (This implies a strong motivation to lead)

How Good Is Your Time Management: 50 (You're managing your time very effectively)

Test de autocunoastere

1. A1 B3 D2 F1 G1 H2
2. A2 D2 E1 F3 H2
3. B1 C2 D1 E1 G3 H2
4. A3 B2 F1 G2 H2
5. C3 D3 E3 F1
6. B3 C2 E3 F1 G1
7. A1 B2 C1 D1 F2 G2 H1

REZULTAT: Lucrător al echipei

#### **Budacă Andrei (Lead Developer)**

The Leadership Motivation Assessment: 53 (This implies some uncertainty about your motivation to lead)

How Good Is Your Time Management: 58 (You're managing your time very effectively)

Test de autocunoastere

1. A2 B1 C4 F1 G1 H1
2. B1 E7 F2
3. B3 D3 F3 G1
4. A1 C3 D3 F3
5. A3 B3 C1 D1 H2
6. B2 C1 E3 F3 G1
7. A3 B1 C3 F3

REZULTAT: Modelator

#### **Pintilie Razvan-Nicolae (Software Developer)**

The Leadership Motivation Assessment: 48 (This implies some uncertainty about your motivation to lead)

How Good Is Your Time Management: 56 (You're managing your time very effectively)

Test de autocunoastere

1. B3 C3 F3 H1
2. A3 G3 H4
3. A3 D1 F1 G3 H2
4. B5 F2 H3
5. A2 B2 C1 E2 F1 H2
6. B3 C2 F3 G1 H1
7. C3 E3 F2 G2

REZULTAT: Coordonator

### **Dumea Cristian (Software Developer)**

The Leadership Motivation Assessment: 50 (This implies some uncertainty about your motivation to lead)

How Good Is Your Time Management: 51 (You're managing your time very effectively)

Test de autocunoastere

1. A4 B2 E2 H1
2. A2 C2 F2 H4
3. B2 D2 E2 F2 H2
4. C2 D2 E2 G2 H1
5. A2 B2 C2 E2 G2
6. B2 C2 E2 F2
7. A1 B3 E1 G4 H1

REZULTAT: Lucrător al echipei

### **Imbrea Daniel (Tester)**

The Leadership Motivation Assessment: 39 (This implies some uncertainty about your motivation to lead)

How Good Is Your Time Management: 39 (You're good at some things but there is room for improvement elsewhere)

Test de autocunoastere

1. C5 E4 G1
2. A2 C1 D3 E1 G2 H1
3. A3 D4 E1 F1 G1
4. C5 G1 H4
5. B2 D2 F2 G2 H2
6. A5 B3 D2

7. A5 D1 E2 G2

REZULTAT: Inovator

### **Ciocan Stefan-Robert (Software Developer)**

The Leadership Motivation Assessment: 42 (This implies some uncertainty about your motivation to lead)

How Good Is Your Time Management: 58 (You're managing your time very effectively)

Test de autocunoastere

1. B2 C5 E2 G1
2. C5 G3 H2
3. C2 D4 E2 G2
4. A1 B2 C4 H3
5. A2 B2 D2 H4
6. B3 C3 G4
7. D6 E2 F2

REZULTAT: Inovator

### **Bejinaru Mihnea-George (Software Engineer)**

The Leadership Motivation Assessment: 41 (This implies some uncertainty about your motivation to lead)

How Good Is Your Time Management: 55 (You're managing your time very effectively. Still, check the sections below to see if there's anything you can tweak to make this even better.)

Test de autocunoastere

1. A1 B2 C1 E2 G3 H1
2. A4 E3 G2 H1
3. B2 C1 D1 F2 G3 H1
4. A2 B1 C1 D1 F2 H3
5. A2 B2 C1 E1 G4 H2
6. A3 C1 D2 E2 F2
7. A2 B1 E3 F2 H2

REZULTAT: CW (Lucrator al echipei/companiei) / Implementator

### **Chirca Radu-Iulian (Software Developer)**

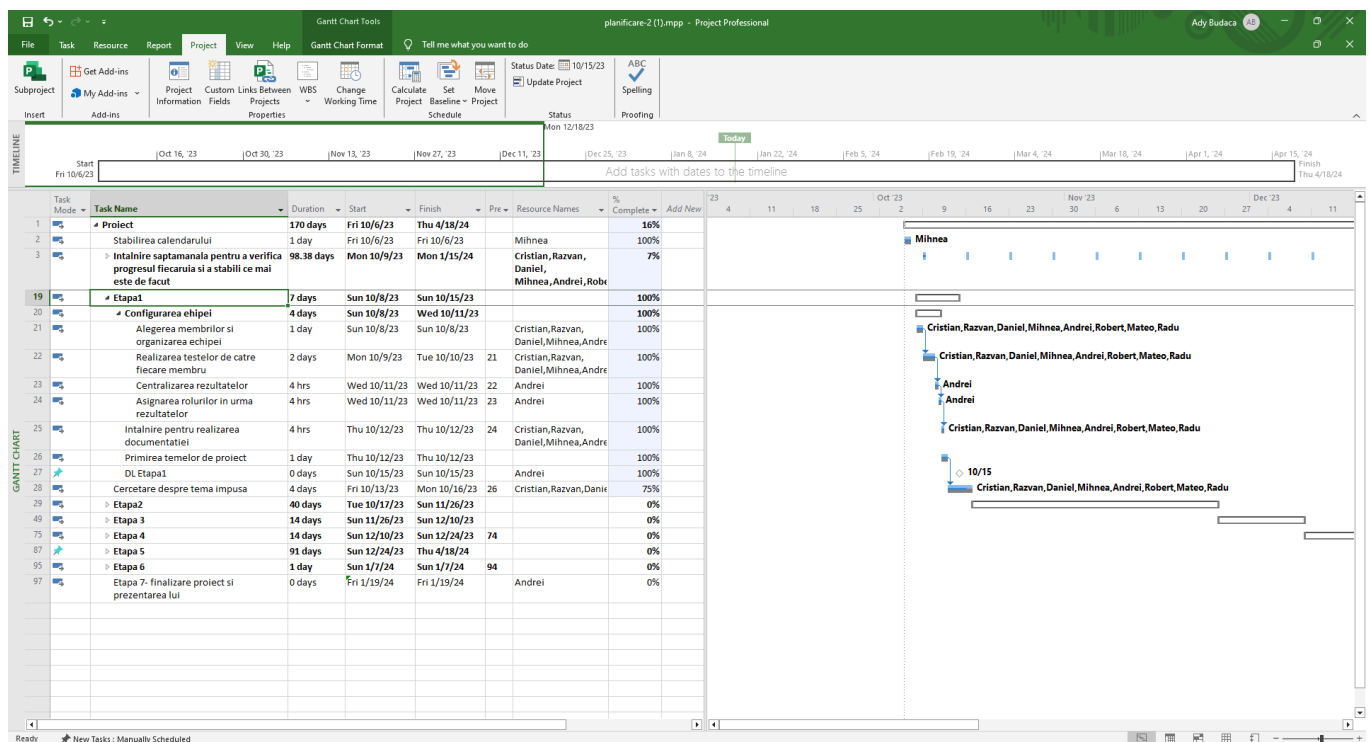
The Leadership Motivation Assessment: 50 (This implies some uncertainty about your motivation to lead)

How Good Is Your Time Management: 39 (You're good at some things but there is room for improvement elsewhere)

## Test de autocunoastere

1. A2 C2 D1 F1 G1 H3
2. A3 B1 C1 D2 G1 H2
3. A1 B1 C1 E3 G1 H3
4. A2 B3 C2 D1 G1 H1
5. A1 B3 C1 D2 E1 F1 H1
6. B2 C1 E2 F1 G2 H2
7. A1 C1 D2 E6

REZULTAT: Lucrător al echipei



## Etapa 2

# Software Requirements Specification

### 2.1. Purpose of the Document

Scopul documentului este de a descrie pașii necesari rezolvării problemei de proiectare a unei RNA cu 2 straturi care sa aproximeze functia  $\phi: \mathbb{R} \rightarrow \mathbb{R}$ ,  $\phi(u) = 1 + e^{-\sin 2\pi u}$

### 2.2. Scope of the System

Documentul este împărțit în 3 capitole.

- Primul capitol oferă o descriere generală a produsului software despre situația inițială, scopul proiectului, contextul și beneficiile proiectului.
- Al doilea capitol listează cerințele funcționale pe care ar trebui să le îndeplinească produsul software. Deci, descrie actorii, granița sistemului și cazurile de utilizare.
- Ultimul capitol expune cerințele non-funcționale ale aplicației, cum ar fi: performanță, siguranță, probleme de securitate etc.

## **2.3 General description of the product**

### **2.3.1 The current situation**

În cadrul acestui proiect, se dorește realizarea unei rețele neuronale artificiale (RNA) cu două straturi, având un strat ascuns cu 's' neuroni ce utilizează funcția de activare tangentă sigmoidă (tansig), și un strat de ieșire cu un singur neuron ce folosește o funcție de activare liniară.

Scopul acestei soluții este să aproximeze o funcție matematică dată, în acest caz, funcția  $\varphi(u) = 1 + e^{-(\sin(2\pi u - 2))}$ , unde 'u' este un număr real. RNA va fi antrenată pe un set de date specific, cu tipare de intrare selectate astfel încât să acopere cazurile pentru  $N = 51$  și  $N = 101$ , conform formulei  $u_i = (i-1)/(N-1)$ , unde 'i' variază de la 1 la N.

### **2.3.2 Purpose of the product**

În situația actuală, metodele tradiționale pot întâmpina dificultăți în a surprinde modelele intricate și non-liniare ale acestei funcții, fapt care motivează dezvoltarea unei soluții specializate.

Scopul produsului constă în a furniza o unealtă computațională capabilă să învețe și să aproximeze relația complexă descrisă de funcția matematică specificată.

Produsul își propune să prezinte o soluție eficientă și precisă pentru prezicerea ieșirii în funcție de intrare, bazându-se pe modelele învățate din datele de antrenare.

### **2.3.3 Product context**

Contextul produsului se axează pe crearea unei arhitecturi de rețea neuronală adaptată special pentru aproximarea funcțiilor matematice.

Produsul este conceput ca o soluție independentă, punând accent pe adaptabilitatea sa la seturi de date diverse și capacitatea de a gestiona funcții matematice cu modele non-liniare.

Contextul produsului se extinde către aplicații unde prezicerea precisă a ieșirii în funcție de intrare este crucială, precum în modelele matematice și sarcinile de analiză a datelor.

### **2.3.4 Benefits**

Beneficiile acestei soluții constau în capacitatea sa de a oferi o aproximare precisă și eficientă a unor funcții matematice complexe. Prin utilizarea puterii rețelelor neuronale și a unei arhitecturi adaptate, produsul oferă o modalitate de a surprinde modele intricate care ar putea fi dificil de obținut cu metode tradiționale.

Flexibilitatea rețelei neuronale permite adaptarea la diverse seturi de date și funcții matematice, îmbunătățind astfel utilitatea sa în diferite domenii.

Beneficiile includ precizia crescută în predicții, eficiența sporită în aproximarea funcțiilor și aplicabilitatea într-o gamă largă de sarcini de modelare matematică.

## 2.4 Functional requirements

### 2.4.1 Actors

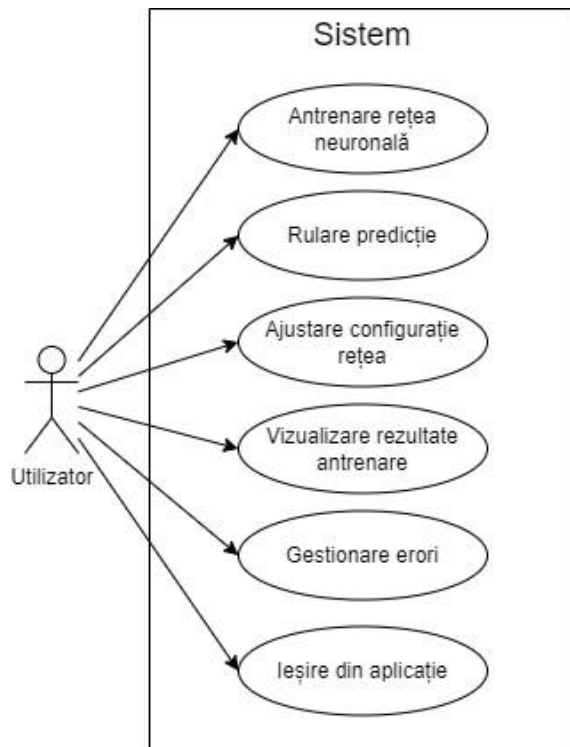
Acest capitol conține descrierea tuturor entităților care interacționează cu sistemul.

#### 2.4.1.1 Utilizator

Principalul actor în sistem este Utilizatorul, care interacționează cu software-ul pentru proiectarea unei rețele neuronale artificiale (RNA) cu două straturi folosind Matlab. Utilizatorul efectuează una dintre următoarele operațiuni:

- Antrenare rețea neuronală
- Rulare predicție
- Ajustare configurație rețea
- Vizualizare rezultate antrenare
- Gestionare erori
- Ieșire din aplicație

#### 2.4.2 System boundary



#### 2.4.3 Use cases description

Descrierea cazelor de utilizare subliniază diferitele scenarii în care sistemul de rețea neuronală artificială (RNA) este folosit, detaliind interacțiunile dintre sistem și utilizatorii săi.

Actorul principal implicat în aceste cazuri de utilizare este utilizatorul care folosește mediul de dezvoltare (Matlab).

#### 2.4.3.1 Antrenare rețea neuronală

- **Actorii:** Utilizator
- **Descriere:** Utilizatorul inițiază procesul de antrenare furnizând sistemului RNA un set de date. Sistemul utilizează mediul Matlab pentru a antrena rețeaua neuronală conform configurației specificate. Acest lucru implică ajustarea ponderilor și a bazelor rețelei pentru a aproxima funcția matematică  $\phi(u)$ .
- **Precondiții:** Matlab este instalat și accesibil.
- **Postcondiții:** Rețeaua neuronală este antrenată, iar sistemul este pregătit pentru predicții.

#### 2.4.3.2 Rulare predicție

- **Actorii:** Utilizator
- **Descriere:** După faza de antrenare, utilizatorul poate introduce o valoare nouă 'u,' iar sistemul, folosind rețeaua neuronală antrenată, furnizează o predicție pentru valoarea corespunzătoare  $\phi(u)$ .
- **Precondiții:** Rețeaua neuronală este antrenată.
- **Postcondiții:** Sistemul returnează rezultatul prevăzut pentru valoarea dată.

#### 2.4.3.3 Ajustare configurație rețea

- **Actorii:** Utilizator
- **Descriere:** Utilizatorul are opțiunea de a modifica configurația rețelei neuronale, cum ar fi numărul de neuroni ascunși sau alte parametri, pentru a observa impactul asupra rezultatelor de antrenare și predicție.
- **Precondiții:** Matlab este instalat și accesibil.
- **Postcondiții:** Rețeaua neuronală este reconfigurată în funcție de ajustările utilizatorului.

#### 2.4.3.4 Vizualizare rezultate antrenare

- **Actorii:** Utilizator
- **Descriere:** Utilizatorul poate accesa și revizui rezultatele procesului de antrenare, inclusiv metrice de performanță, acuratețe și eventuale erori sau avertismente.
- **Precondiții:** Rețeaua neuronală a fost antrenată.
- **Postcondiții:** Utilizatorul obține informații privind eficacitatea procesului de antrenare.

#### 2.4.3.5 Gestionare erori

- **Actorii:** Utilizator



- **Descriere:** În cazul apariției erorilor în timpul antrenării sau predicției, sistemul furnizează mesaje informative de eroare pentru a ajuta utilizatorul să înțeleagă și să rezolve problemele.
- **Precondiții:** Erori apar în timpul antrenării sau predicției.
- **Postcondiții:** Utilizatorul este informat despre natura erorii și poate lua măsuri corective.

#### 2.4.3.6 Iesire din aplicatie

- **Actorii:** Utilizator
- **Descriere:** Utilizatorul poate ieși din aplicație în mod corespunzător, asigurându-se că orice configurații sau rezultate nesalvate sunt gestionate în mod adecvat.
- **Precondiții:** Aplicația rulează.
- **Postcondiții:** Aplicația se închide, iar resursele sunt eliberate.

### 2.5. Non-functional requirements

#### 2.5.1 User Interface Requirements

Utilizatorul trebuie să fie capabil să ruleze un script folosind Matlab.

#### 2.5.2 Performance Requirements

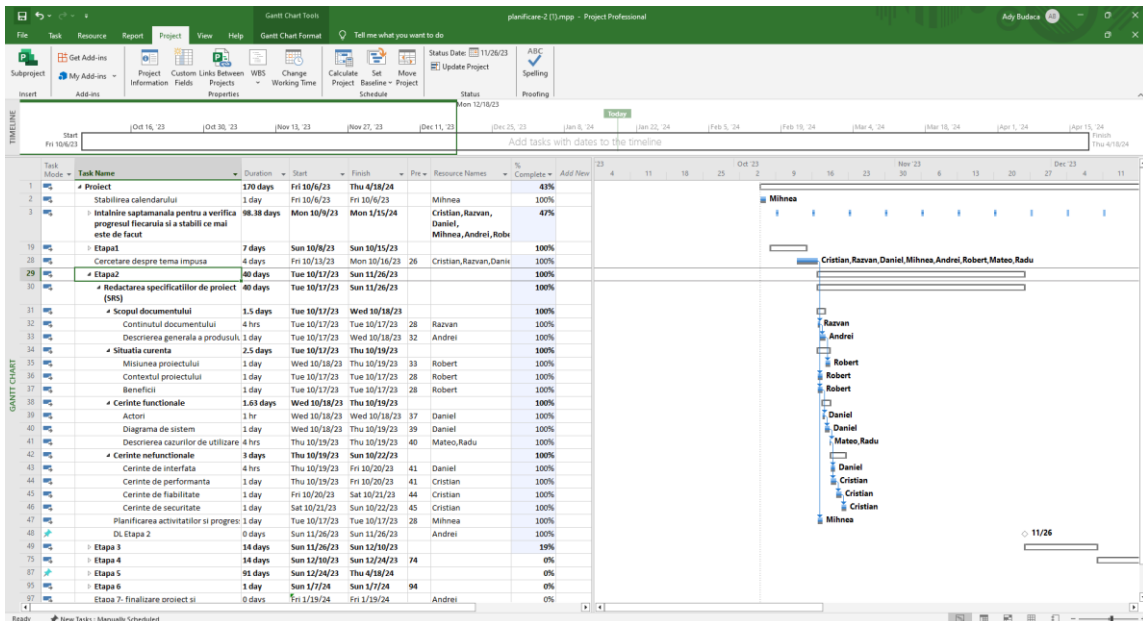
Programul trebuie să poată rula pe orice sistem care suportă Matlab.

#### 2.5.3 Availability & Reliability

Programul trebuie să furnizeze întotdeauna rezultate corecte și, în caz de eroare umană, să semnaleze acest lucru.

#### 2.5.4 Security Requirements

Deși aplicația operează local, pe o mașină gazdă, securizarea acesteia nu este necesară.



# **Etapa 3**

## **Documentul de Proiectare a Soluției aplicației Software (Software Design Document)**

### **3.1 Scopul documentului**

Acest proiect constă în furnizarea unei descrieri comprehensive a soluției proiectate pentru implementarea rețelei neuronale artificiale (RNA) în cadrul sistemului software dedicat. Acest document va servi drept ghid detaliat pentru echipa de dezvoltare responsabilă de construirea și implementarea rețelei neuronale artificiale cu două straturi, formată dintr-un strat ascuns cu 's' neuroni ce utilizează funcția de activare tangentă sigmoidă (tansig), și un strat de ieșire cu un singur neuron ce folosește o funcție de activare liniară. În esență, acest document va oferi clarificări și direcții precise pentru dezvoltarea eficientă și corectă a proiectului axat pe realizarea acestei structuri neuronale specifice.

### **3.2 Conținutul documentului**

Documentul este alcătuit din patru secțiuni:

#### **1. Modelul Datelor:**

Această secțiune oferă o prezentare detaliată a principalelor structuri de date utilizate în cadrul proiectului. Ea expune modul în care datele vor fi organizate și gestionate în cadrul rețelei neuronale artificiale, facilitând înțelegerea procesului de manipulare a informațiilor.

#### **2. Modelul Arhitectural și Modelul Componentelor:**

În această secțiune, se vor descrie șabloanele arhitecturale utilizate pentru implementarea rețelei neuronale artificiale. Se vor evidenția aspectele cheie ale arhitecturii sistemului, precum și detaliile privind fiecare componentă a acesteia. Această secțiune va oferi o perspectivă cuprinzătoare asupra modului în care diverse elemente ale proiectului interacționează și contribuie la funcționarea generală a rețelei neuronale.

#### **3. Modelul Interfeței cu Utilizatorul:**

Această secțiune se axează pe prezentarea interfeței cu utilizatorul dezvoltate în cadrul proiectului. Se va explora modul în care utilizatorii interacționează cu rețeaua neuronală și se vor ilustra succesiunile de ferestre și elemente grafice asociate pentru a facilita înțelegerea experienței utilizatorului.

#### 4. Elemente de Testare:

În ultima secțiune, se vor expune componentele critice ale proiectului și se vor aborda alternativele de proiectare a acestora. Aceasta include strategii și criterii de testare pentru a asigura funcționarea corectă și eficientă a rețelei neuronale artificiale implementate.

### 3.3 Modelul datelor

#### 3.3.1 Structuri de date globale

În cadrul proiectului nostru, structura de date globală reprezintă o instanță a scriptului dedicat rețelei neuronale artificiale. Aceasta servește ca punct de intrare esențial în program, definind parametrii și configurările necesare pentru funcționarea rețelei.

#### 3.3.2 Structuri de Date de Legătură:

Scriptul principal în comunicare directă cu alte scripturi este esențial pentru atingerea obiectivelor proiectului nostru. Aceasta se realizează prin apelul direct al funcțiilor definite în cadrul acestor scripturi, facilitând schimbul de informații între diferitele componente ale rețelei neuronale.

#### 3.3.3 Structuri de Date Temporare:

Pentru implementarea algoritmului specific rețelei neuronale artificiale, programul va utiliza matrici pentru a stoca informațiile temporare referitoare la părinți și copii. Acest aspect este crucial pentru funcționarea eficientă a algoritmului genetic utilizat în cadrul proiectului.

#### 3.3.4 Formatul Fișierelor Utilizate:

În ceea ce privește formatul fișierelor, aplicația noastră se bazează pe fișiere de tip Matlab. Aceste fișiere conțin informațiile esențiale pentru funcționarea rețelei neuronale și sunt gestionate în conformitate cu standardele specifice Matlab.

#### 3.3.5 Descrierea Bazei de Date:

În contextul proiectului nostru, nu se utilizează o bază de date dedicată, deoarece toate datele necesare pentru funcționarea rețelei neuronale sunt stocate în memorie. Acest aspect simplifică gestionarea datelor și optimizează procesul de acces la informații în cadrul aplicației noastre.

### 3.4 Modelul arhitectural si modelul componentelor

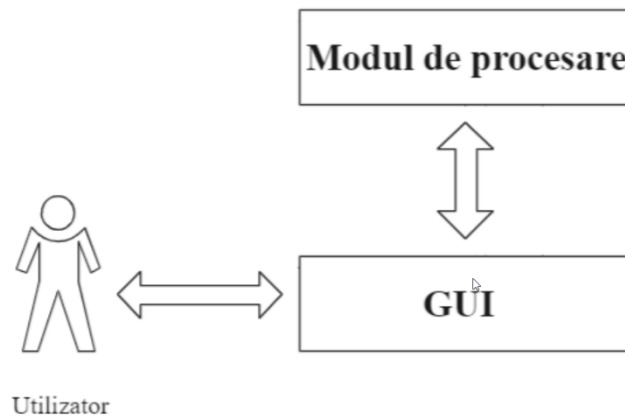
#### 3.4.1 Arhitectura sistemului

##### 3.4.1.1 Sabloane arhitecturale folosite

Arhitectura sistemului nostru se bazează pe șablonul event-driven architecture. Această abordare oferă o structură flexibilă, permitând interacțiuni eficiente între diferitele componente ale sistemului, în special în contextul interfeței cu utilizatorul.

##### 3.4.1.2 Diagrama de arhitectură

Diagrama de arhitectură de mai jos descrie componentele arhitecturii aplicației și relațiile de interacțiune dintre acestea.



### 3.4.2 Descrierea componentelor

Aplicația este formată din următoarele module interconectate:

- **Modulul GUI(Graphical User Interface)**

Este responsabil pentru adunarea informațiilor de intrare de la utilizator și afișarea rezultatelor

- **Modulul de procesare al datelor**

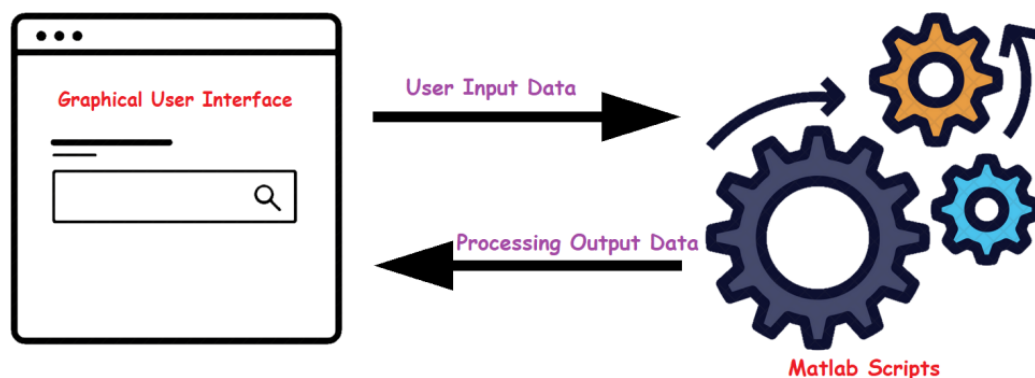
Este responsabil cu adaptarea continuă a datelor așa încât să se ajungă la soluția optimă.

### 3.4.3 Restricțiile de implementare

Ambele module vor fi implementate în mediul de dezvoltare Matlab R2016A folosind limbajul Matlab.

### 3.4.4 Interacțiunea dintre componente

Componenta de procesare va prelua datele de la modulul GUI, le va modifica și apoi le va trimite înapoi la GUI pentru afișarea soluției.



## 3.5 Modelul interfeței cu utilizatorul

În cadrul modulului GUI, prin intermediul unei ferestre de GUI, utilizatorul va putea introduce datele necesare pentru procesare și va primi rezultatele acestora în fereastra respectivă. De asemenea, utilizatorul va putea rula script-ul și prin intermediul consolei din matlab.

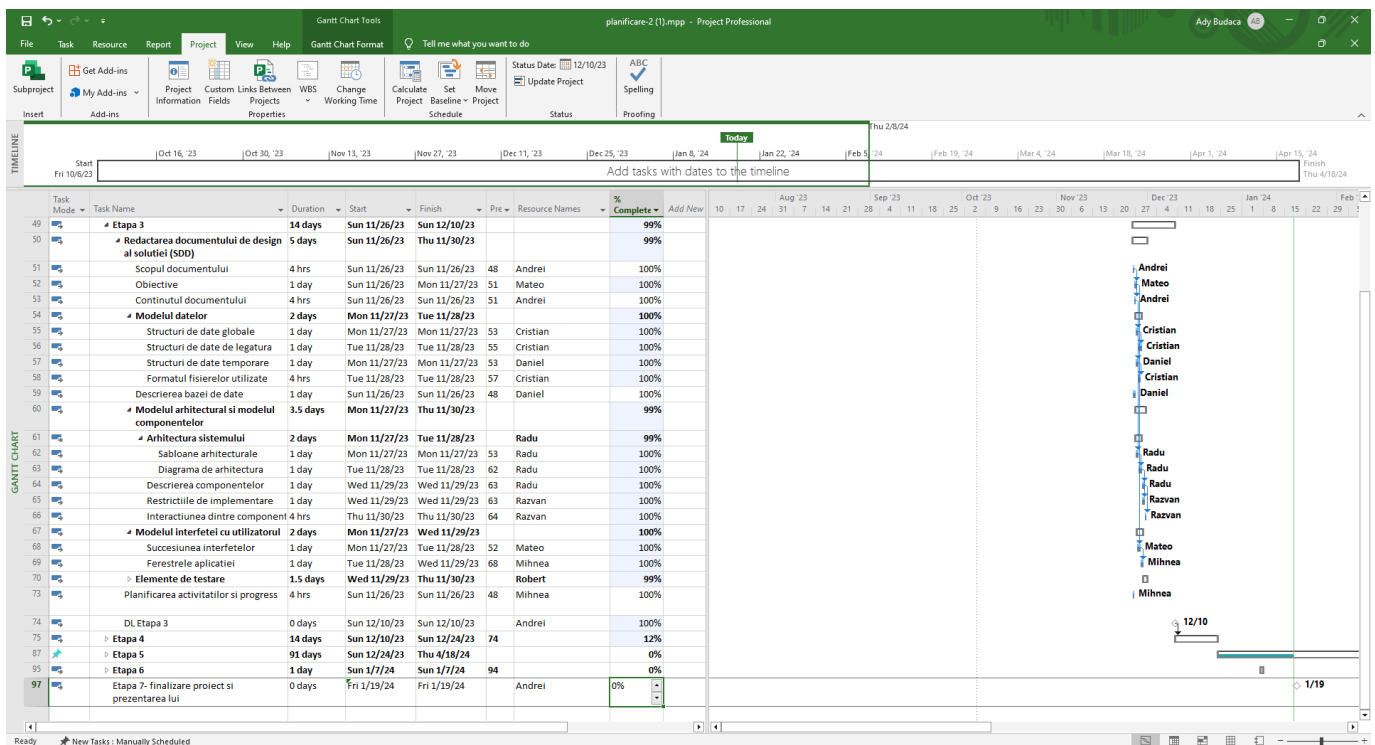
## 3.6 Elemente de testare

### 3.6.1 Componente critice

Performanța modulului de procesare a datelor are o influență decisivă asupra vitezei de răspuns și reprezintă componenta care trebuie să fie cea mai optimizată.

### 3.6.2 Alternative

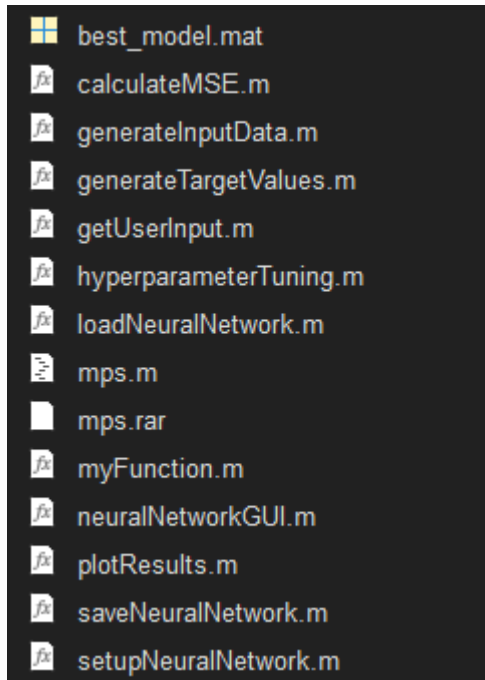
Pentru a crește viteza de procesare a datelor, se pot face simulări cu diferite valori ale parametrilor programului și păstrarea celor care produc cele mai rapide rezultate.



# Etapa 4

## Implementarea proiectului

### 4.1 Structura proiect



În cadrul proiectului am folosit următoarele funcții:

#### 4.1.1 Pentru Algoritm

- `my_function(x)` – returnează valoarea funcției cerute pentru un cromozom

```
% myFunction.m
% This function calculates the values of the given mathematical expression
% for the provided input values.
% Input:
%   - u: Input values
% Output:
%   - result: Calculated values based on the mathematical expression

function result = myFunction(u)
    % Mathematical expression
    result = 1 + exp(-u .* sin(2 * pi * u - 2));
end
```

- `calculateMSE(test_ui, predicted_values, myFunction)` – returnează eroarea medie pătratică dintre valoarea prezisă și valoarea țintă

```

% calculateMSE.m
% This function calculates the mean squared error between predicted and target values
% Input:
%   - test_ui: Test input data
%   - predicted_values: Predicted values from the neural network
%   - myFunction: Target function for comparison
% Output:
%   - mse: Mean squared error

function mse = calculateMSE(test_ui, predicted_values, myFunction)
    % Calculate the mean squared error
    mse = mean((myFunction(test_ui.') - predicted_values).^2);
end

```

- generateInputData(N) – generează date inițiale pentru rețeaua neuronală

```

% generateInputData.m
% This function generates input data for the neural network training.
% Input:
%   - N: Number of data points
% Output:
%   - ui: Generated input data

function ui = generateInputData(N)
    % Generate linearly spaced input values between 0 and 1
    ui = (0:(N-1))/(N-1);
    % Reshape ui to a column vector
    ui = ui(:);
end

```

- generateTargetValues(ui, myFunction) – calculează valorile țintă

```

% generateTargetValues.m
% This function generates target values using a given function.
% Input:
%   - ui: Input data
%   - myFunction: Function to generate target values
% Output:
%   - target_values: Generated target values

function target_values = generateTargetValues(ui, myFunction)
    % Generate target values using the provided function
    target_values = myFunction(ui);
end

```

- hyperparameterTuning(ui, target\_values, hidden\_layer\_neurons\_range, learning\_rate\_range, epochs\_range) – calculează și setează valorile ideale pentru numărul de neuroni din stratul ascuns, rata de învățare și numărul de epoci

```

% hyperparameterTuning.m
function [best_net, best_params, best_mse] = hyperparameterTuning(ui, target_values, hidden_layer_neurons_range, learning_rate_range, epochs_range)

% Specify ranges for hyperparameters
% hidden_layer_neurons_range = [5, 10, 15];
% learning_rate_range = [0.001, 0.01, 0.1];
% epochs_range = [500, 1000, 1500];

best_mse = Inf;
best_params = [];

% Iterate over hyperparameter combinations
for s = hidden_layer_neurons_range
    for lr = learning_rate_range
        for epochs = epochs_range
            % Set up the neural network architecture using a separate function
            net = setupNeuralNetwork(s, numel(ui), lr, epochs);

            % Train the neural network using a separate function
            [net, tr] = trainNeuralNetwork(net, ui, target_values);

            disp(['Number of Epochs: ' num2str(tr.num_epochs)]);
            disp(['Final Training Error: ' num2str(tr.best_perf)]);

            % Test the neural network on the test input data using a separate function
            predicted_values = testNeuralNetwork(net, linspace(0, 1, numel(ui)).');

            % Calculate the mean squared error using a separate function
            mse = calculateMSE(ui, predicted_values, @myFunction);

            % Update best hyperparameters if the current model is better
            if mse < best_mse
                best_mse = mse;
                best_params = [s, lr, epochs];
            end
        end
    end
end

disp(['Best hyperparameters: Hidden Neurons=' num2str(best_params(1)) ', Learning Rate=' num2str(best_params(2)) ', Epochs=' num2str(best_params(3))]);

% Set up the neural network with the best hyperparameters
best_net = setupNeuralNetwork(best_params(1), numel(ui), best_params(2), best_params(3));

% Train the neural network with the best hyperparameters
[best_net, ~] = trainNeuralNetwork(best_net, ui, target_values);

end

```

- `loadNeuralNetwork(filename)` - încarcă rețeaua neuronală într-o variabilă numită `net`

```

% loadNeuralNetwork.m
% Load a trained neural network from a MAT-file.
% Input:
% - filename: Name of the MAT-file containing the saved network
% Output:
% - net: Loaded neural network object

function net = loadNeuralNetwork(filename)
    % Load the neural network from a MAT-file
    loadedData = load(filename);
    net = loadedData.net;
end

```

- `saveNeuralNetwork(net, filename)` – salvează rețeaua neuronală într-un fișier care poate fi încărcat ulterior

```

% saveNeuralNetwork.m
% Save the trained neural network to a MAT-file.
% Input:
% - net: Trained neural network object
% - filename: Name of the MAT-file to save the network to

function saveNeuralNetwork(net, filename)
    % Save the neural network to a MAT-file
    save(filename, 'net');
end

```

- `setupNeuralNetwork(s, inputSize, learningRate, epochs)` – inițializează rețeaua neuronală și permite alegerea unei funcții de transfer



```

% setupNeuralNetwork.m
% This function sets up a feedforward neural network with specified parameters.
% Input:
%   - s: Number of neurons in the hidden layer
%   - inputSize: Size of the input layer
% Output:
%   - net: Configured neural network

function net = setupNeuralNetwork(s, inputSize, learningRate, epochs)
    % Create a feedforward neural network
    net = feedforwardnet(s);

    % Set the transfer function of the output layer to linear
    net.layers{2}.transferFcn = 'purelin';

    % Choose diff ones
    % Tansig (Hyperbolic Tangent Sigmoid):
    % 'tansig': Hyperbolic tangent sigmoid transfer function.

    % Logsig (Logarithmic Sigmoid):
    % 'logsig': Logarithmic sigmoid transfer function.

    % Purelin (Linear):
    % 'purelin': Pure linear transfer function.

    % Hardlim (Hard Limit):
    % 'hardlim': Hard limit transfer function.

    % Hardlims (Symmetric Hard Limit):
    % 'hardlims': Symmetric hard limit transfer function.

    % Satlin (Saturating Linear):
    % 'satlin': Saturating linear transfer function.

    % Satlins (Symmetric Saturating Linear):
    % 'satlins': Symmetric saturating linear transfer function.

    % Poslin (Positive Linear):
    % 'poslin': Positive linear transfer function.

    % Compet (Competitive):
    % 'compet': Competitive transfer function.

    % Set the size of the input layer
    net.inputs{1}.size = inputSize;

    % Set the learning rate
    net.trainParam.lr = learningRate;

    % Set the epochs
    net.trainParam.epochs = epochs;
end

```

## 4.1.2 Pentru GUI

- Crearea elementelor din interfață

```
function neuralNetworkGUI()
N = 51;

% Generate input data using a separate function
ui = generateInputData(N);

% Generate target values using a separate function and the provided myFunction
target_values = generateTargetValues(ui, @myFunction);

% Create the main figure window
fig = figure('Name', 'Neural Network Hyperparameter Tuning GUI', 'Position', [100, 100, 800, 600]);

% Create axes for plotting (right side)
ax = axes('Parent', fig, 'Position', [0.5, 0.1, 0.4, 0.8]);

% Create dropdown menus for selecting hyperparameters (left side)
uicontrol('Style', 'text', 'String', 'Select Hidden Neurons:', 'Position', [-10, 520, 250, 30]);
hidden_neurons_edit = uicontrol('Style', 'edit', 'String', '', 'Position', [190, 530, 150, 30]);

uicontrol('Style', 'text', 'String', 'Select Learning Rate:', 'Position', [-10, 470, 250, 30]);
learning_rate_edit = uicontrol('Style', 'edit', 'String', '', 'Position', [190, 480, 150, 30]);

uicontrol('Style', 'text', 'String', 'Select Epochs:', 'Position', [-10, 410, 250, 30]);
epochs_edit = uicontrol('Style', 'edit', 'String', '', 'Position', [190, 420, 150, 30]);
% Create a button for initiating hyperparameter tuning
start_button = uicontrol('Style', 'pushbutton', 'String', 'Start Hyperparameter Tuning', 'Position', [70, 350, 250, 30], 'Callback', @startTuning);

result_textbox = uicontrol('Style', 'edit', 'String', '', 'Position', [70, 80, 270, 240], 'Max', 10, 'HorizontalAlignment', 'left');
```

- Crearea callback-ului pentru pornirea antrenării

```
% Callback function for the start button
function startTuning(~, ~)
% Get selected hyperparameters from dropdown menus
hidden_neurons_str = hidden_neurons_edit.String;
hidden_neurons = str2num(hidden_neurons_str);
if isempty(hidden_neurons) || ~isvector(hidden_neurons) || any(hidden_neurons < 0)
    error('Please enter a valid vector of hidden neurons.', 'Invalid Input', 'modal');
    return;
end
learning_rate_str = learning_rate_edit.String;
learning_rate = str2double(learning_rate_str);

epochs_str = epochs_edit.String;
epochs = str2num(epochs_str);

% Perform hyperparameter tuning using selected values
[best_net, best_params, best_mse] = hyperparameterTuning(ui, target_values, hidden_neurons, learning_rate, epochs);

% Generate test input data for evaluation
test_ui = linspace(0, 1, numel(ui)).';

% Test the neural network on the test input data using a separate function
predicted_values = testNeuralNetwork(best_net, test_ui);

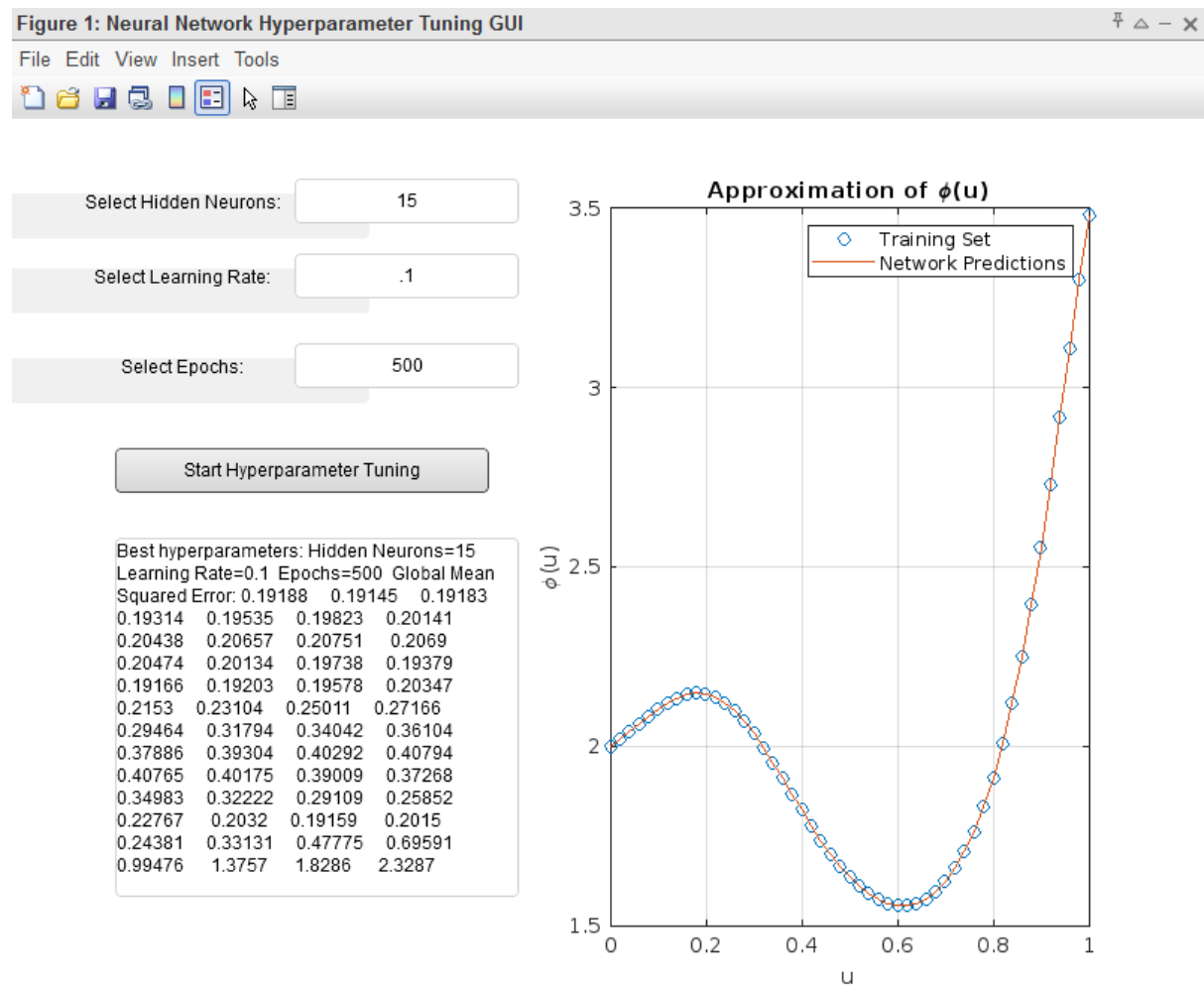
% Calculate the mean squared error using a separate function
mse = calculateMSE(test_ui, predicted_values, @myFunction);

% Plot the results on the axes (right side)
plot(ax, ui, target_values, 'o', test_ui, predicted_values, '-');
title(ax, 'Approximation of \phi(u)');
xlabel(ax, 'u');
ylabel(ax, '\phi(u)');
legend(ax, 'Training Set', 'Network Predictions');
grid(ax, 'on');
result_str = ['Best hyperparameters: Hidden Neurons=' num2str(best_params(1)) ' Learning Rate=' num2str(best_params(2)) ' Epochs=' num2str(best_params(3))];
result_str2=[' Global Mean Squared Error: ' num2str(mse)];

% Setează textul în TextBox pentru afișare
result_textbox.String = [result_str result_str2];
% Display the results
disp(['Global Mean Squared Error: ' num2str(mse)]);

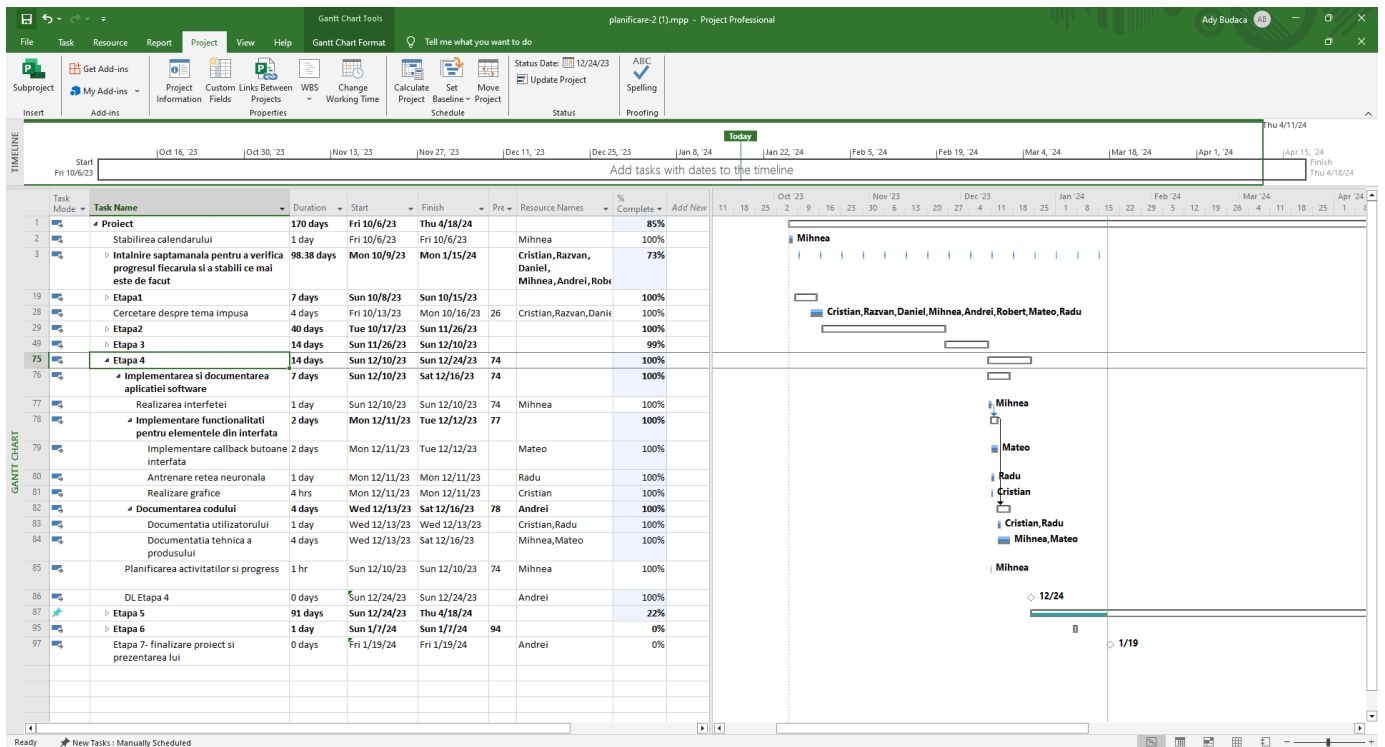
end
end
```

### 4.1.3 Exemple rulare GUI:



În cadrul output-ului GUI este prezentat:

- Setul de valori care au fost folosite pentru a antrena rețeaua
- Valorile prezise de rețeaua neuronală
- Eroarea medie globală



# Etapa 5

## Testarea Aplicatiei

### 5.1 Plan de testare

Pentru a asigura buna funcționalitate a aplicației și pentru a determina dacă aceasta oferă rezultate precise, am ales testarea unitară a modulelor importante care iau parte la logica acesteia.

Interfața a fost proiectată astfel încât să se evite introducerea unor date invalide, precum input-uri ce nu sunt valori numerice sau caractere nepermise care ar putea afecta în mod negativ funcționarea componentei logice a aplicației.

## 5.2 Scenarii de test

Testarea metodei *generateInputData(N)*:

Pentru testarea acestei metode s-a avut în vedere validarea datelor returnate pentru diferiți parametri de intrare. S-a urmărit atât dimensiunea vectorului returnat, cât și corectitudinea elementelor acestuia.

Testarea metodei *calculateMSE(test\_ui, predicted\_values, myFunction)*:

Pentru testarea acestei metode s-a avut în vedere corectitudinea calculării erorii mediei pătratice comparând rezultatele cu rezultatele funcției *immse* din Matlab.

Testarea metodei *my\_function(x)*:

Pentru testarea acestei metode s-a avut în vedere corectitudinea calculării rezultatului funcției evaluate. S-a avut în vedere testarea funcției cu parametru pozitiv, negativ și egal cu zero.

Testarea metodei *hyperparameterTuning(ui, target\_values, hidden\_layer\_neurons\_range, learning\_rate\_range, epochs\_range)*:

Pentru testarea acestei metode s-a avut în vedere corectitudinea rezultatului funcției evaluate. S-a avut în vedere corectitudinea tuturor valorilor returnate astfel:

- verificarea MSE-ului returnat să aibe o valoare diferită de infinit;
- verificarea vectorului parametrilor optimi să nu fie vid;
- verificarea rețelei create să nu fie vidă;

## 5.3 Rapoarte cu rezultatele testelor

*generateInputData(N)*:

```
Running TestGenerateInputData
...
Done TestGenerateInputData
-----

ans =

1x3 TestResult array with properties:

    Name
    Passed
    Failed
    Incomplete
    Duration
    Details

Totals:
    3 Passed, 0 Failed, 0 Incomplete.
    0.03246 seconds testing time.
```

*calculateMSE(test\_ui, predicted\_values, myFunction):*

```
Running TestCalculateMSE
...
Done TestCalculateMSE
-----

ans =

1x2 TestResult array with properties:

    Name
    Passed
    Failed
    Incomplete
    Duration
    Details

Totals:
    2 Passed, 0 Failed, 0 Incomplete.
    0.37759 seconds testing time.
```

*my\_function(x):*

```
Running TestMyFunction
...
Done TestMyFunction
-----

ans =

1x3 TestResult array with properties:

    Name
    Passed
    Failed
    Incomplete
    Duration
    Details

Totals:
    3 Passed, 0 Failed, 0 Incomplete.
    0.093539 seconds testing time.
```

*hyperparameterTuning(ui, target\_values, hidden\_layer\_neurons\_range,  
learning\_rate\_range, epochs\_range):*

```
Running TestHyperparameterTuning
Final Training Error: 0
Number of Epochs: 0
Final Training Error: 0
Best hyperparameters: Hidden Neurons=2, Learning Rate=0.1, Epochs=10
Final Training Error: 0
.
Done TestHyperparameterTuning
-----

ans =

TestResult with properties:

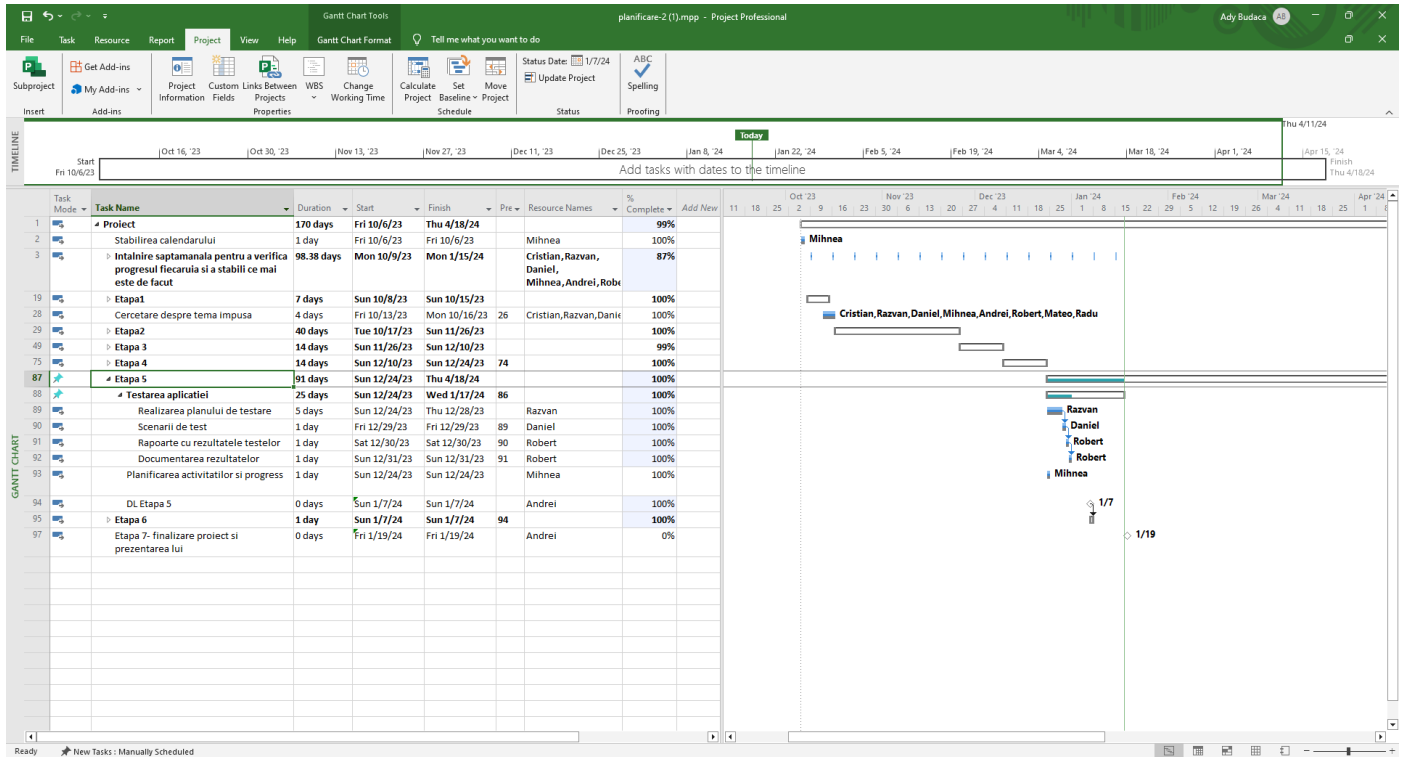
    Name: 'TestHyperparameterTuning/testHyperparameterTuningWithValidInput'
    Passed: 1
    Failed: 0
    Incomplete: 0
    Duration: 0.9883
    Details: [1x1 struct]

Totals:
    1 Passed, 0 Failed, 0 Incomplete.
    0.98828 seconds testing time.
```

## 5.4 Documentarea rezultatelor

În urma efectuării testelor am ajuns la următoarele concluzii: ☐

- Programul funcționează fără erori. ☐
- Interfața grafică este ușor de utilizat.



# RNA cu 2 straturi pentru aproximarea unei funcții

Tarlev Mateo (**Project Manager**)

Budacă Andrei (**Lead Developer**)

Bejinariu Mihnea-George (**Software Engineer**)

Pintilie Răzvan-Nicolae (**Software Developer**)

Dumea Cristian (**Software Developer**)

Ciocan Stefan-Robert (**Software Developer**)

Chirica Radu-Iulian (**Software Developer**)

Imbrea Daniel (**Lead Tester**)

Proiectati o RNA cu 2 straturi (strat ascuns cu  $s$  neuroni cu functii de activare tansig si strat de iesire cu 1 neuron liniar) care sa aproximeze functia:

$$\varphi: R \rightarrow R, \varphi(u) = 1 + e^{-u} \sin\left(2\pi u - \frac{\pi}{2}\right)$$

Tiparele de intrare din setul de antrenare se aleg astfel:

$$u(i) = \frac{i-1}{N-1}, i = 1, \dots, N$$


Se vor considera cazurile  $N = 51$  si  $N = 101$ . Trebuie alese valori convenabile pentru numar neuroni ascusi, rata învățare, numar epoci antrenare. Aprecierea eficientei se va face în termenii erorii patraticice globale obtinute în urma antrenarii si a numarului de iteratii necesare pentru atingerea unui prag impus pentru eroare. Se va urmări si comportarea rețelei antrenate pe setul de date de testare:

$$u = 0: \left(\frac{0.5}{N}\right): 1$$






# Purpose of the product



În situația actuală, metodele tradiționale pot întâmpina dificultăți în a surprinde modelele non-liniare ale acestei funcții, fapt care motivează dezvoltarea unei soluții specializate.



Scopul produsului constă în a furniza o unealtă computațională capabilă să învețe și să aproximeze relația complexă descrisă de funcția matematică specificată.

Produsul își propune să prezinte o soluție eficientă și precisă pentru prezicerea ieșirii în funcție de intrare, bazându-se pe modelele învățate din datele de antrenare.

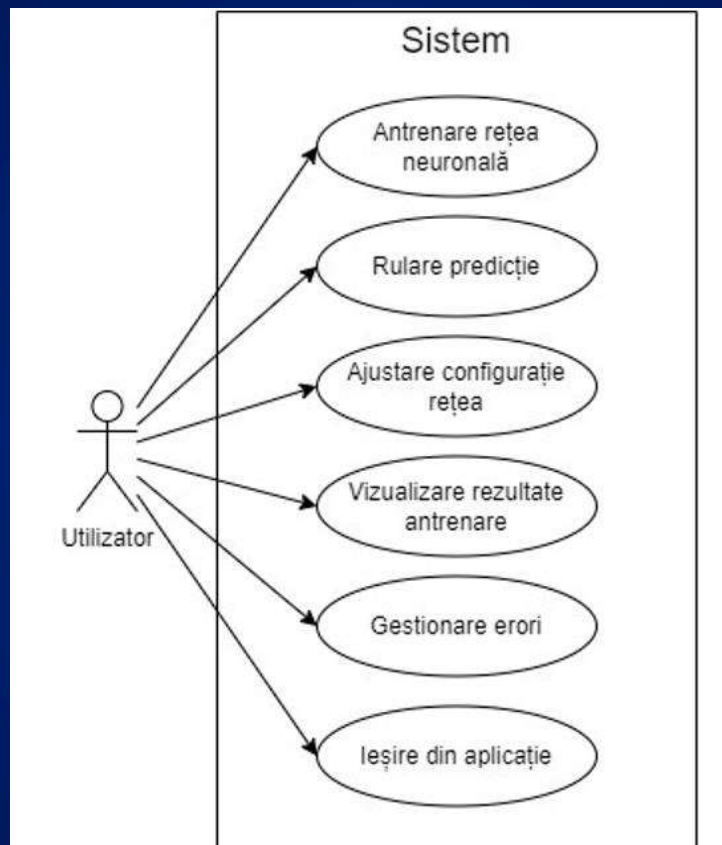


# Functional requirements

**UTILIZATORUL** este principalul actor în sistem, care interacționează cu software-ul pentru proiectarea unei rețele neuronale artificiale (RNA) cu două straturi folosind Matlab.

Utilizatorul efectuează una dintre următoarele operațiuni:

- **Antrenare rețea neuronală**
- **Rulare predicție**
- **Ajustare configurație rețea**
- **Vizualizare rezultate antrenare**
- **Gestionare erori**
- **Ieșire din aplicație**



# **Nonfunctional requirements**

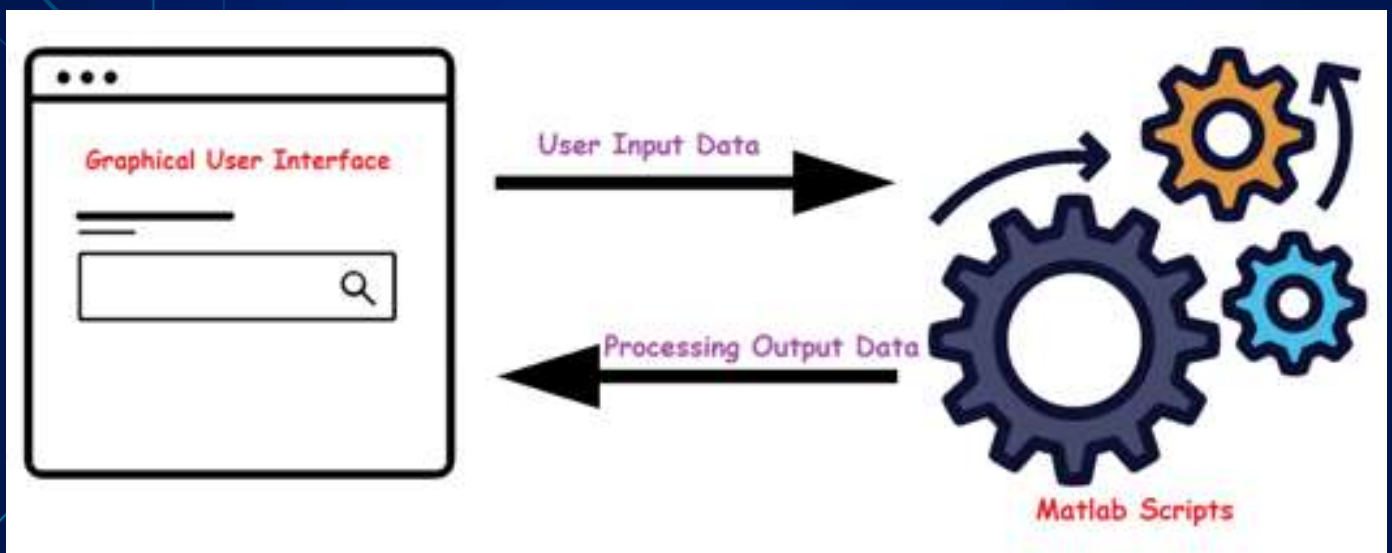
- **Utilizatorul trebuie să fie capabil să ruleze un script folosind Matlab**
- **Programul trebuie să poată rula pe orice sistem care suportă Matlab**
- **Programul trebuie să furnizeze întotdeauna rezultate corecte și, în caz de eroare umană, să semnaleze acest lucru**
- **Deoarece aplicația operează local, pe o mașină gazdă, securizarea acesteia nu este necesară**

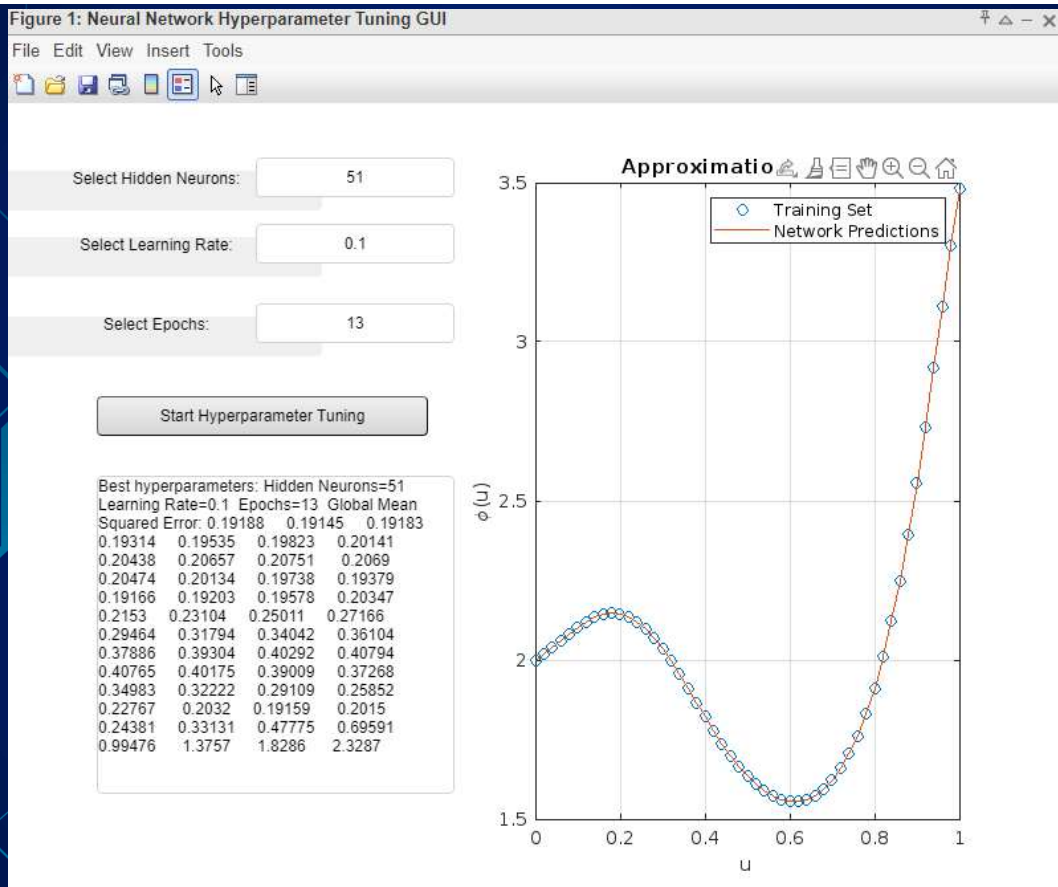
# Componentele aplicației

- **Modulul GUI**
  - este responsabil de adunarea informațiilor de intrare de la utilizator și afișarea rezultatelor
- **Modulul de procesare al datelor**
  - este responsabil de adaptarea continuă a datelor așa încât să se ajungă la soluția optimă

# Interacțiunea dintre componente

Componenta de procesare va prelua datele de la modulul GUI, le va modifica și apoi le va trimite înapoi la GUI pentru afișarea soluției.





# Code Examples



- **generateInputData(N)** – generează date inițiale pentru rețeaua neuronală

```
% generateInputData.m
% This function generates input data for the neural network training.
% Input:
%   - N: Number of data points
% Output:
%   - ui: Generated input data

function ui = generateInputData(N)
    % Generate linearly spaced input values between 0 and 1
    ui = (0:(N-1))/(N-1);
    % Reshape ui to a column vector
    ui = ui(:);
end
```

- **generateTargetValues(ui, myFunction)** – calculează valorile țintă

```
% generateTargetValues.m
% This function generates target values using a given function.
% Input:
%   - ui: Input data
%   - myFunction: Function to generate target values
% Output:
%   - target_values: Generated target values

function target_values = generateTargetValues(ui, myFunction)
    % Generate target values using the provided function
    target_values = myFunction(ui);
end
```



# Tests

# Examples

## Testare calculare corectă a MSE

```
classdef TestCalculateMSE < matlab.unittest.TestCase
    methods (TestClassSetup)
        % Shared setup for the entire test class
    end

    methods (TestMethodSetup)
        % Setup for each test
    end

    methods (Test)

        function testCalculateMSEWithZeroInput(testCase)
            N = 51;
            test_ui = zeros(N, 1);
            predicted_values = zeros(N, 1);

            mse_actual = calculateMSE(test_ui, predicted_values, @myFunction);
            mse_expected = mean((myFunction(test_ui') - predicted_values).^2);
            testCase.verifyEqual(mse_actual, mse_expected);
        end

        function testCalculateMSEWithValidInput(testCase)
            N = 51;
            test_ui = generateInputData(N);
            predicted_values = myFunction(test_ui');

            mse_actual = calculateMSE(test_ui, predicted_values, @myFunction);
            mse_expected = mean((myFunction(test_ui') - predicted_values).^2);
            testCase.verifyEqual(mse_actual, mse_expected);
        end
    end
end
```

## Testare funcționalitate myFunction

```
classdef TestCalculateMSE < matlab.unittest.TestCase
    methods (TestClassSetup)
        % Shared setup for the entire test class
    end

    methods (TestMethodSetup)
        % Setup for each test
    end

    methods (Test)

        function testCalculateMSEWithZeroInput(testCase)
            N = 51;
            test_ui = zeros(N, 1);
            predicted_values = zeros(N, 1);

            mse_actual = calculateMSE(test_ui, predicted_values, @myFunction);
            mse_expected = mean((myFunction(test_ui') - predicted_values).^2);
            testCase.verifyEqual(mse_actual, mse_expected);
        end

        function testCalculateMSEWithValidInput(testCase)
            N = 51;
            test_ui = generateInputData(N);
            predicted_values = myFunction(test_ui');

            mse_actual = calculateMSE(test_ui, predicted_values, @myFunction);
            mse_expected = mean((myFunction(test_ui') - predicted_values).^2);
            testCase.verifyEqual(mse_actual, mse_expected);
        end
    end
end
```

# Managementul Proiectului

