

# System zarządzania zespołem tworzącym gry komputerowe “Roofless Studio Team Management System”

Opracował:  
Jan Brzeziński

# Spis treści

<b>System zarządzania zespołem tworzącym gry komputerowe “Roofless Studio Team Management System”</b>	<b>1</b>
Spis treści	2
<b>1. Dziedzina problemu</b>	<b>3</b>
<b>2. Cel</b>	<b>4</b>
<b>3. Zakres odpowiedzialności systemu</b>	<b>5</b>
<b>4. Użytkownicy Systemu</b>	<b>6</b>
<b>5. Wymagania Użytkownika</b>	<b>7</b>
<b>6. Wymagania funkcjonalne</b>	<b>9</b>
<b>7. Opis Struktury Systemu</b>	<b>10</b>
<b>8. Analiza dynamiczna dla przypadków użycia</b>	<b>11</b>
<b>9. Analiza dynamiczna dla wybranej klasy obiektów</b>	<b>17</b>
<b>10. Struktura systemu z uwzględnieniem wyników analizy dynamicznej</b>	<b>18</b>
<b>11. Decyzje projektowe</b>	<b>19</b>
<b>12. Schemat logiczny systemu</b>	<b>20</b>
<b>13. Projekt GUI w oparciu o przypadki użycia</b>	<b>21</b>
<b>14. Wymagania Niefunkcjonalne</b>	<b>26</b>
<b>15. Opis Ewolucji Systemu</b>	<b>27</b>

# 1. Dziedzina problemu

System może znaleźć zastosowanie w przypadkach małych firm zajmujących się tworzeniem niezależnych gier komputerowych (tzw. “Gry Indie”). Produkcja gier komputerowych jest złożonym procesem, który wymaga skoordynowanej pracy zespołu. Jednak brak odpowiednich narzędzi i organizacji może prowadzić do chaosu, opóźnień i trudności w śledzeniu postępów projektów. System ten ma na celu rozwiązanie tych problemów poprzez zapewnienie efektywnego zarządzania zespołem, projektami i zadaniami, z którymi muszą się mierzyć członkowie zespołu.

## 2. Cel

Celem niniejszego systemu jest wsparcie zespołu poprzez udostępnienie aplikacji z przejrzystym i intuicyjnym interfejsem użytkownika. System umożliwia dodawanie nowych członków zespołu o określonych rolach i specjalizacjach, tworzenie zadań i projektów. Głównym celem jest umożliwienie przypisywania członków do zadań i projektów, co pozwala zespołowi na efektywne organizowanie pracy. Ponadto, system umożliwia generowanie raportów dotyczących postępów w pracy nad projektami oraz zgłaszanie ewentualnych błędów (bugów), zapewniając pełniejszą kontrolę nad procesem tworzenia gier komputerowych.

### 3. Zakres odpowiedzialności systemu

#### **Zarządzanie członkami zespołu:**

- System umożliwia dodawanie nowych członków zespołu, przechowując informacje takie jak imię, nazwisko, pseudonim, adres e-mail oraz dodatkowe dane związane z ich specjalizacją i rolą w zespole.
- Użytkownicy mają możliwość przeglądania, edytowania i usuwania informacji o członkach zespołu.
- System zapewnia możliwość przypisywania członków do konkretnych zadań i projektów, umożliwiając przejrzystość w podziale pracy.

#### **Zarządzanie zadaniami:**

- Użytkownicy mogą tworzyć nowe zadania, określając ich nazwę, opis, priorytet, termin wykonania i przypisując je do konkretnych członków zespołu.
- System umożliwia przeglądanie informacji o zadaniach, w tym ich statusu, postępu i przypisanego członka zespołu.
- Użytkownicy mają możliwość aktualizacji danych związanych z zadaniami, takich jak zmiana statusu, terminu wykonania itp.

#### **Raportowanie postępów i błędów:**

- System umożliwia generowanie raportów dotyczących postępów w pracy nad projektem, uwzględniających informacje o wykonanych zadaniach i ich statusie.
- Użytkownicy mogą tworzyć raporty z postępów dla konkretnych projektów.
- System zapewnia możliwość zgłaszania błędów (bugów) napotkanych podczas pracy nad grą, umożliwiając ich dokumentowanie, przypisywanie do konkretnych projektów.

#### **Zarządzanie projektami:**

- Użytkownicy mają możliwość tworzenia nowych projektów, określając ich nazwę, opis, terminy i informacje o kliencie, który zlecił dany projekt.
- System umożliwia przeglądanie i edytowanie informacji o projektach, w tym zmianę statusów, terminów, informacji o kliencie itp.
- Użytkownicy mogą przypisywać członków zespołu do projektów, zapewniając przypisanie odpowiednich zasobów do realizacji projektu.

#### **Zarządzanie testami:**

- System umożliwia przechowywanie informacji o testach wykonywanych w ramach projektów.
- Użytkownicy mogą dodawać informacje o testach, takie jak nazwa testu, priorytet, status, wyniki testu itp.
- System umożliwia przeglądanie informacji o testach.

## 4. Użytkownicy Systemu

- Członek zespołu: Programista
- Członek zespołu: Artysta
- Członek zespołu: Manager

## 5. Wymagania Użytkownika

System powinien przechowywać informacje o:

- Członkach zespołu
  - Dane personalne (imię, nazwisko, data urodzenia)
  - Pseudonim
  - Email
  - Rola pełniona w zespole:
    - Artysta:
      - Specjalizacja
      - Lista programów
      - Preferowany styl
      - Link do portfolio/showreel'u
      - dodatkowe umiejętności
    - Programista:
      - Specjalizacja
      - Lista technologii (stack technologiczny)
    - Manager:
- Projekcie
  - Nazwa projektu
  - Gatunek gry
  - Faza projektu
  - Data rozpoczęcia
  - Przewidywana data zakończenia
- Zadaniach
  - Nazwa zadania
  - Deadline
  - Opis
  - Poziom trudności
  - Priorytet
- Raportach
  - ID
  - Data
  - Autor
  - Treść
  - Opis błędu
  - Typ błędu
  - Priorytet
  - Lista wykonanych zadań
- Testach
  - Nazwa testu
  - Opis
  - Status
  - Priorytet
  - Wyniki testu
- Klientach
  - Dane osobowe

- Informacje dot. firmy/organizacji
- Adres email
- Numer telefonu

Oczekuje się, że system będzie pełnił rolę wsparcia dla użytkowników - członków zespołu - poprzez dostarczanie funkcjonalności, które umożliwią im wygodne zarządzanie projektami i zadaniami.

Istnieje kilka wniosków, które można wyciągnąć z tego oczekiwania:

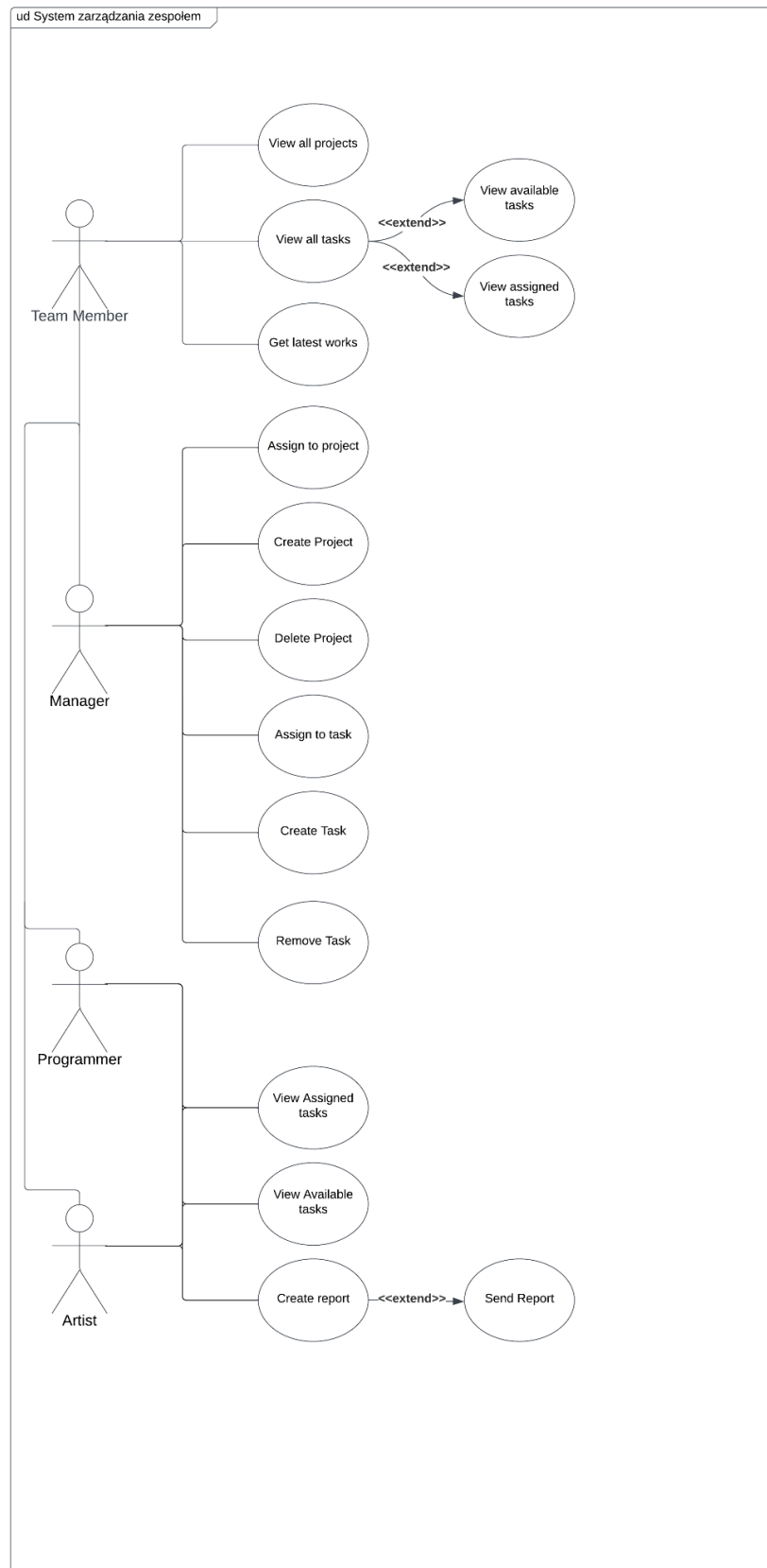
- System zapewnia dostępność i przeglądanie na bieżąco informacji o projektach i zadaniach. Dzięki temu każdy członek zespołu może łatwo uzyskać aktualne informacje na temat postępu prac, terminów i priorytetów. To umożliwia efektywną koordynację działań w zespole.
- Członkowie zespołu mają możliwość szczegółowego przeglądania informacji dotyczących projektów i zadań. Wszelkie istotne dane, takie jak opis, terminy, przypisane zasoby czy status, są dostępne w systemie. Dzięki temu użytkownicy mogą łatwo uzyskać pełny kontekst i zrozumienie dotyczące danego elementu pracy.
- Menedżer ma uprawnienia do tworzenia nowych projektów i zadań oraz przypisywania ich do odpowiednich członków zespołu. Jest w stanie skutecznie rozdzielać zadania i kontrolować postęp prac. To umożliwia efektywne rozplanowanie projektów i równomierne rozłożenie obciążenia w zespole.
- Menedżer ma również możliwość dodawania nowych członków do systemu. Dzięki temu może zarządzać składem zespołu. To ułatwia zarządzanie dynamicznymi zespołami i umożliwia elastyczne skalowanie zasobów ludzkich.

Aplikacja powinna spełniać następujące wymagania:

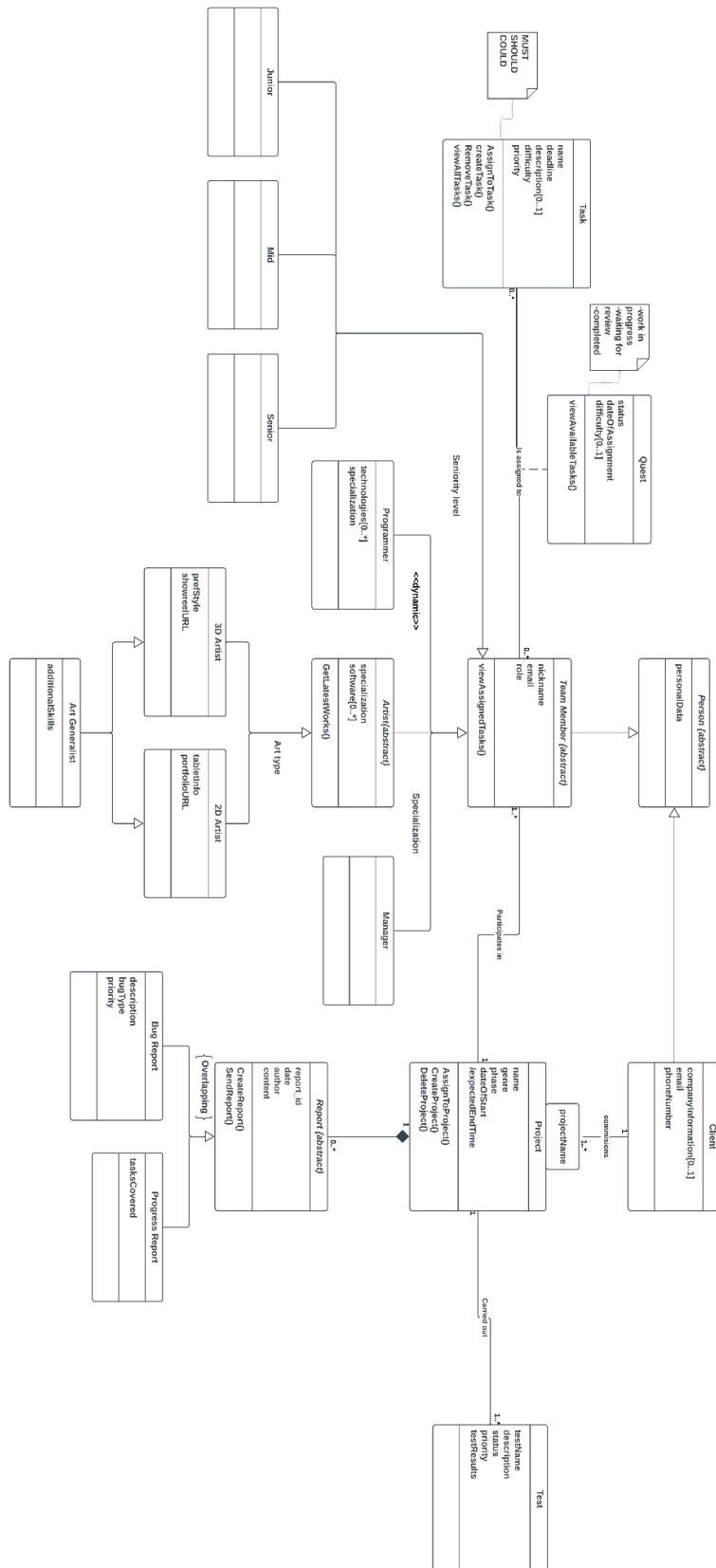
- Powinna działać na systemach operacyjnych Windows 7, 10, 11
- Powinna zostać zaimplementowana w języku Java, wersji 17
- W przypadku awarii, powinna zostać naprawiona w przeciągu max. 24h.



## 6. Wymagania funkcjonalne



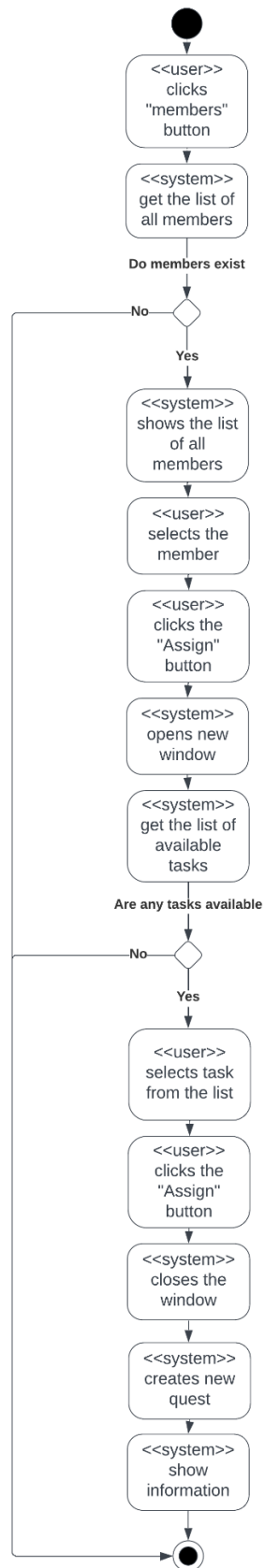
# 7. Opis Struktury Systemu

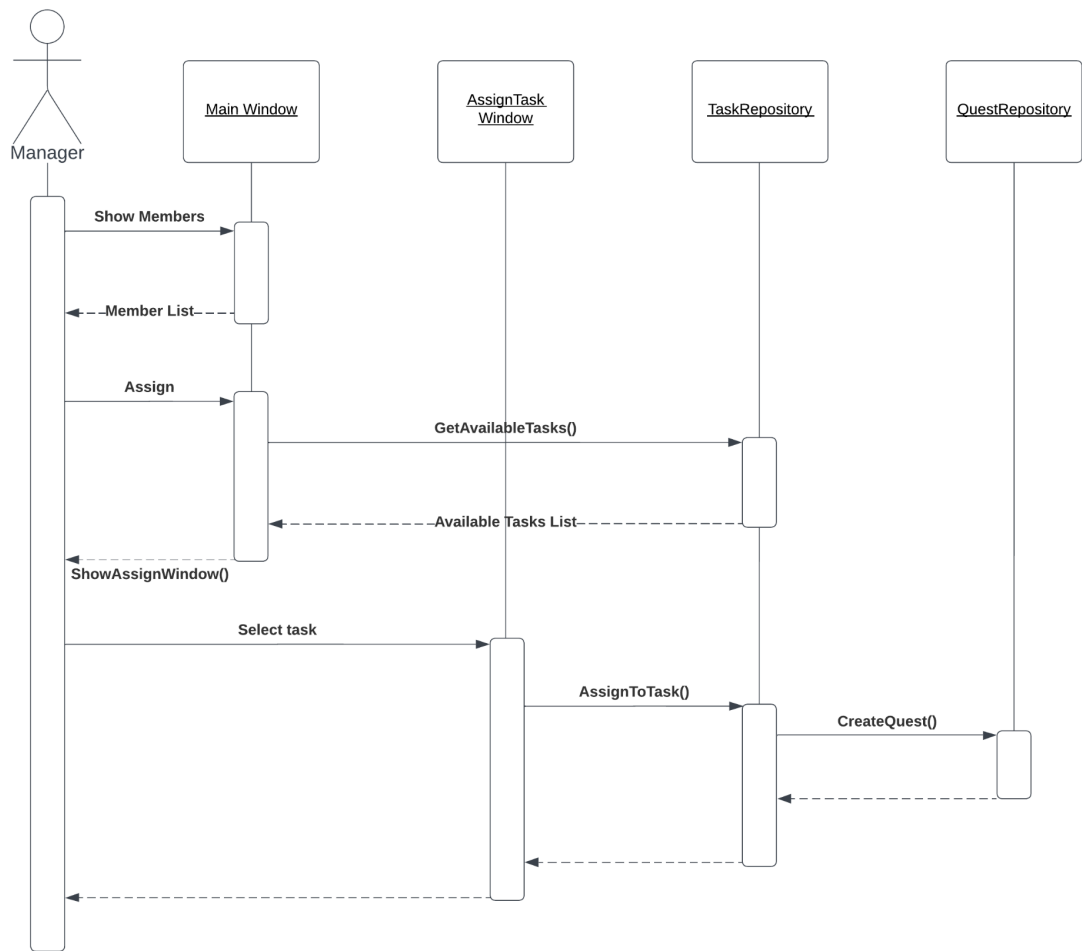


## 8. Analiza dynamiczna dla przypadków użycia

### Przypadek użycia: Przypisz członka zespołu do Taska (Nowy "Quest")

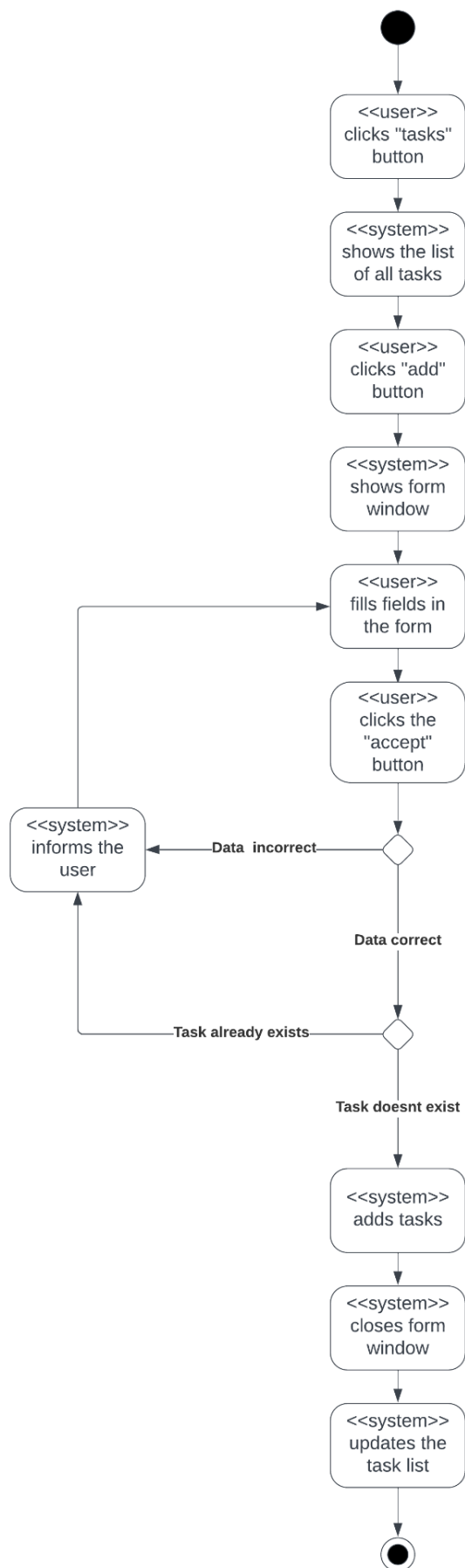
Warunek początkowy	Użytkownik zalogowany jako Manager z uprawnieniami Administratora
Główny przepływ zdarzeń	<ol style="list-style-type: none"><li>1. Użytkownik klika w przycisk "Members" wyświetlający listę wszystkich członków zespołu</li><li>2. Użytkownik wybiera z listy interesującego go członka zespołu</li><li>3. Użytkownik klika w przycisk "Assign"</li><li>4. Pojawia się nowe okno zawierające listę typu "Drop-Down" z listą dostępnych tasków</li><li>5. Użytkownik rozwija listę i wybiera interesujący go task</li><li>6. Użytkownik klika w przycisk "Assign"</li><li>7. Okno zamyka się, pojawia się informacja o przypisaniu.</li></ol>
Alternatywny przepływ zdarzeń	
Zakończenie	W dowolnym momencie

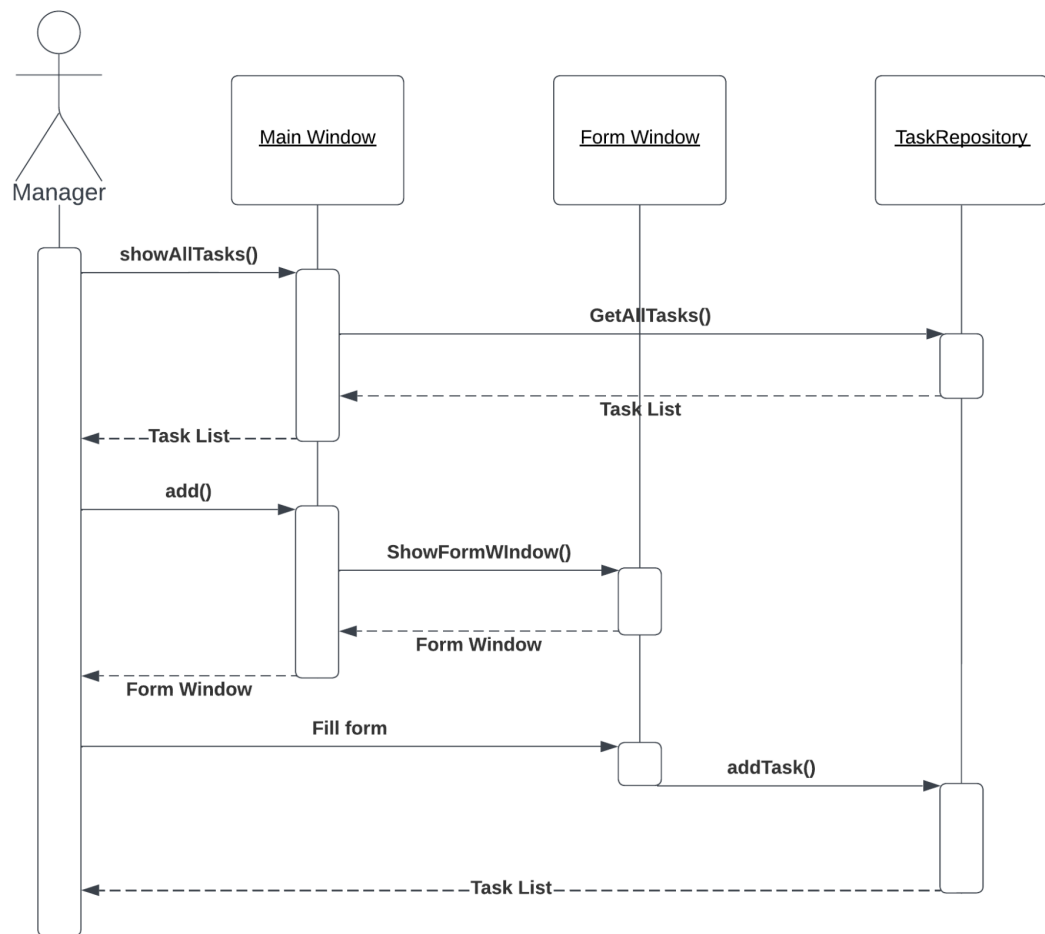




**Przypadek użycia: Dodaj task**

Warunek początkowy	Użytkownik zalogowany jako Manager z uprawnieniami Administratora
Główny przepływ zdarzeń	<ol style="list-style-type: none"><li>1. Użytkownik klika w przycisk “Tasks” znajdujący się nad oknem, w którym wyświetlają się listy w systemie</li><li>2. Użytkownik klika w przycisk “Add” znajdujący się w panelu obsługi listy</li><li>3. Pojawia się nowe okno z formularzem</li><li>4. Użytkownik wypełnia formularz</li><li>5. Użytkownik klika w przycisk “Accept”</li><li>6. Lista tasków aktualizuje się</li><li>7. Okno się zamyka, widoczna jest zaktualizowana lista tasków.</li><li>8. System kończy przypadek użycia</li></ol>
Alternatywny przepływ zdarzeń	
Zakończenie	W dowolnym momencie

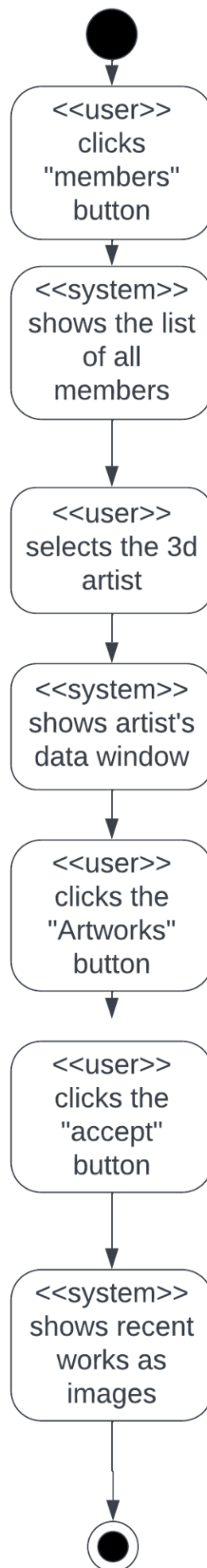


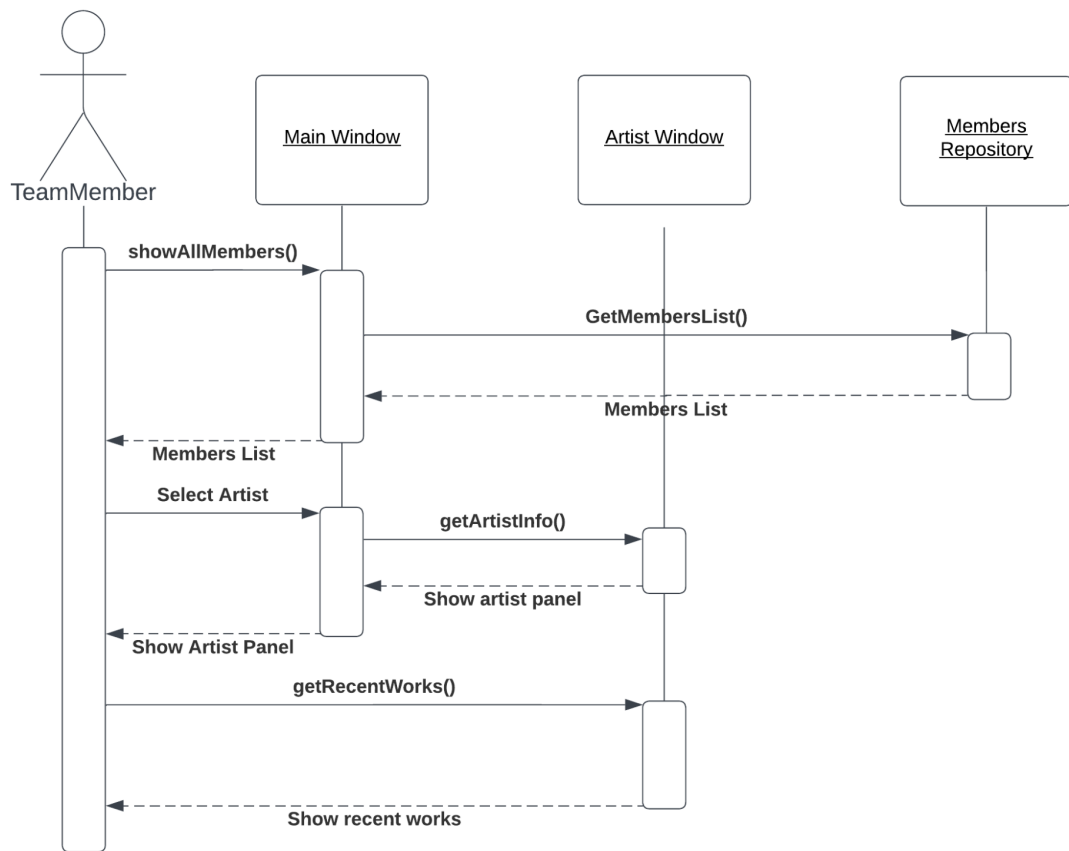




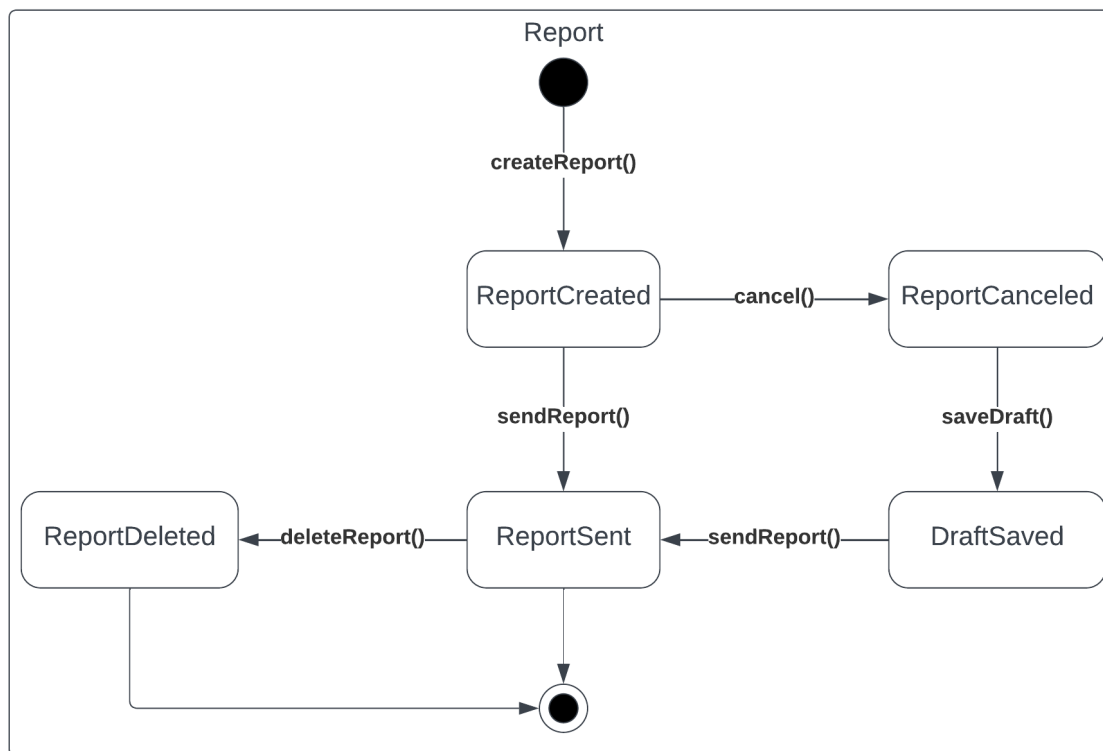
**Przypadek użycia: Pokaż najnowsze prace**

Warunek początkowy	Użytkownik zalogowany jako Artysta, Programista lub Manager z uprawnieniami do odczytu
Główny przepływ zdarzeń	<ol style="list-style-type: none"><li>1. Użytkownik klika w przycisk "Members"</li><li>2. Użytkownik wybiera z listy członka o roli "artist"</li><li>3. Użytkownik klika w interesującego go artystę</li><li>4. Listę na interfejsie zastępują informacje o danym artyście</li><li>5. Użytkownik klika w przycisk "Artworks"</li><li>6. Pod ogólnymi informacjami pokazują się najnowsze prace danego artysty</li><li>7. System kończy przypadek użycia</li></ol>
Alternatywny przepływ zdarzeń	
Zakończenie	W dowolnym momencie





## 9. Analiza dynamiczna dla wybranej klasy obiektów



W wyniku przeprowadzonej analizy dynamicznej, wykorzystującej obiekt klasy `Report`, ujawniono nową funkcjonalność w postaci wersji roboczej raportu. Aby umożliwić użytkownikowi (`TeamMember`) posiadającemu uprawnienia do edycji tworzenie nowych raportów, konieczne będzie udostępnienie dodatkowej metody, która umożliwi zapis raportu jako wersji roboczej (w przypadku anulowania).

Warto jednak zauważyć, że procedura usuwania raportu będzie możliwa jedynie po opublikowaniu raportu. Oznacza to, że użytkownik nie będzie mógł usunąć raportu, dopóki nie zostanie on opublikowany. Po wysłaniu raportu, użytkownik będzie miał możliwość usunięcia go, jeśli zajdzie taka potrzeba.

## 10. Struktura systemu z uwzględnieniem wyników analizy dynamicznej



# 11. Decyzje projektowe

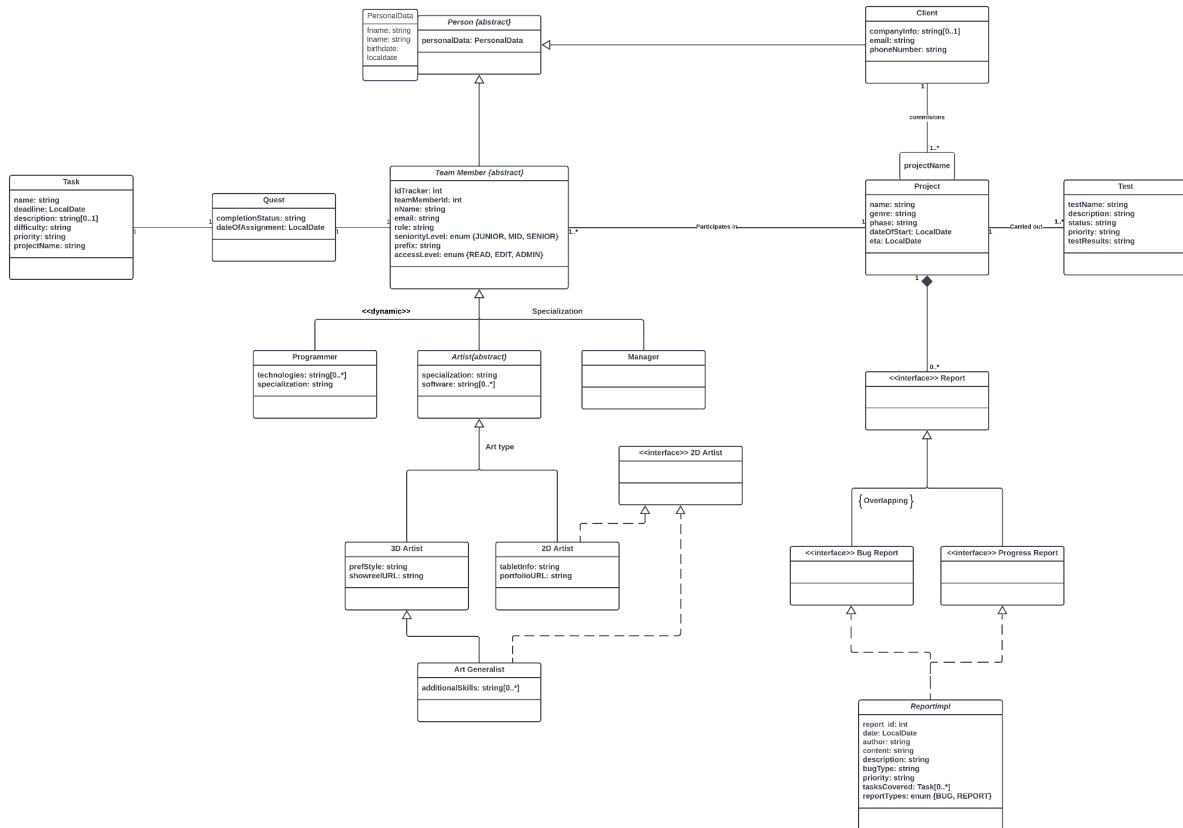
Implementacja konstrukcji:

- Atrybut pochodny - wyliczany na podstawie innych atrybutów (np. ETA)
- Asocjacja - związek między klasami (np. Team Member - Project)
- Związek wiele do wiele - Asocjacja składająca się łącznie z relacji między trzema klasami (dot. Tabela asocjacyjna "Quest")
- Dziedziczenie typu Overlapping - dziedziczenie, w którym klasy na siebie nachodzą (dot. Report)
- Dziedziczenie dynamiczne - dziedziczenie w którym obiekt może "zamienić się" w inny (dot. Team Member - Access <poziom dostępu>)
- Wielodziedziczenie - dziedziczenie, w którym klasa "dziedziczy" po dwóch różnych klasach (dot. Art Generalist)

Przechowywanie danych:

- Serializacja
  - Informacje o obiektach i ich asocjacjach będą przechowywane w plikach.
- Klasa ObjectPlus
  - Klasa pomocnicza definiująca zestaw statycznych metod, które umożliwiają tworzenie, zapisywanie i odczytywanie ekstensji obiektów. Każda klasa dziedzicząca po ObjectPlus może korzystać z tych funkcjonalności. Klasa utrzymuje mapę, gdzie kluczem jest klasa, a wartością jest lista obiektów tej klasy. Podczas tworzenia nowego obiektu klasy pochodnej, obiekt automatycznie jest dodawany do odpowiedniej ekstensji. Dodatkowo, umożliwia zapisywanie i odczytywanie ekstensji obiektów z pliku. Metody writeExtents() i readExtents() pozwalają na zapisanie wszystkich ekstensji do pliku .ser oraz odczytanie ich z powrotem. Istnieje również możliwość wyświetlenia zawartości ekstensji dla konkretnej klasy oraz pobrania wszystkich obiektów danej klasy z ekstensji.

## 12. Schemat logiczny systemu



## 13. Projekt GUI w oparciu o przypadki użycia

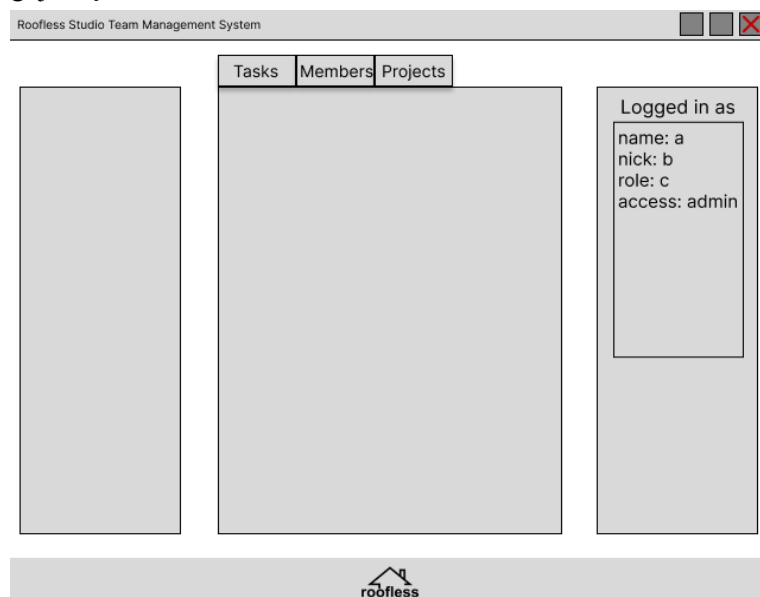
**Przypadek użycia: Przypisz członka zespołu do Taska (Nowy “Quest”)**

1. Użytkownik otwiera aplikację



The image shows a window titled "Roofless Studio Team Management System" with standard window controls (minimize, maximize, close). The main content area is titled "User Login" and contains three input fields: "Username", "Password", and a "LOGIN" button. Below the input fields is a horizontal bar with the "roofless" logo.

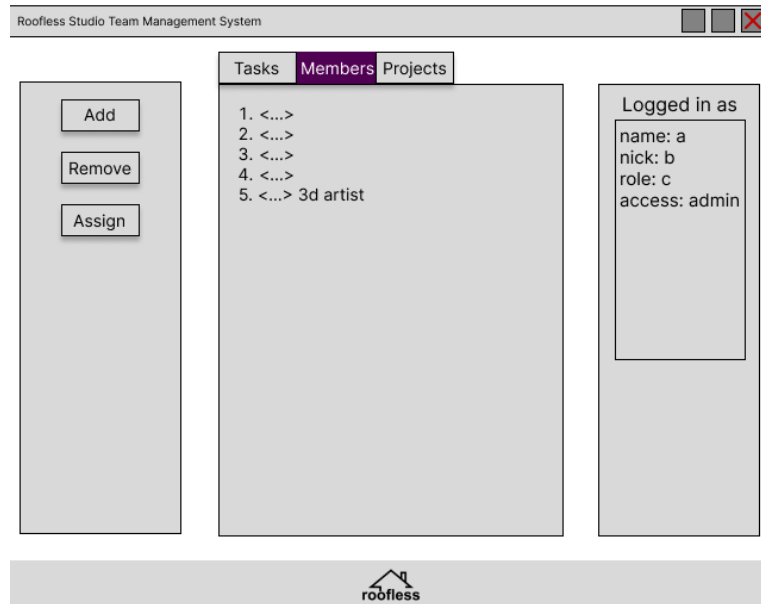
2. Użytkownik loguje się



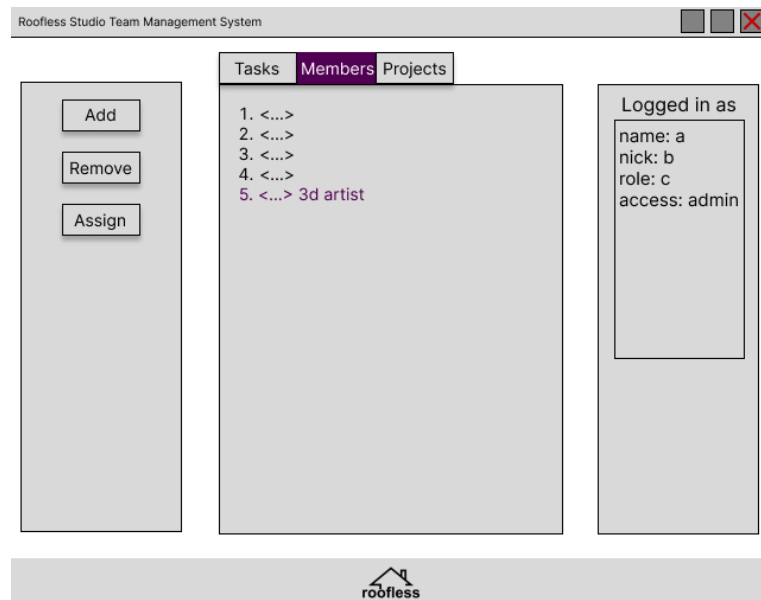
The image shows the main application window after login. The title bar is "Roofless Studio Team Management System". The main content area has three tabs: "Tasks", "Members", and "Projects". The "Tasks" tab is selected. To the right of the tabs is a "Logged in as" box containing the following information: name: a, nick: b, role: c, access: admin. Below the tabs is a horizontal bar with the "roofless" logo.



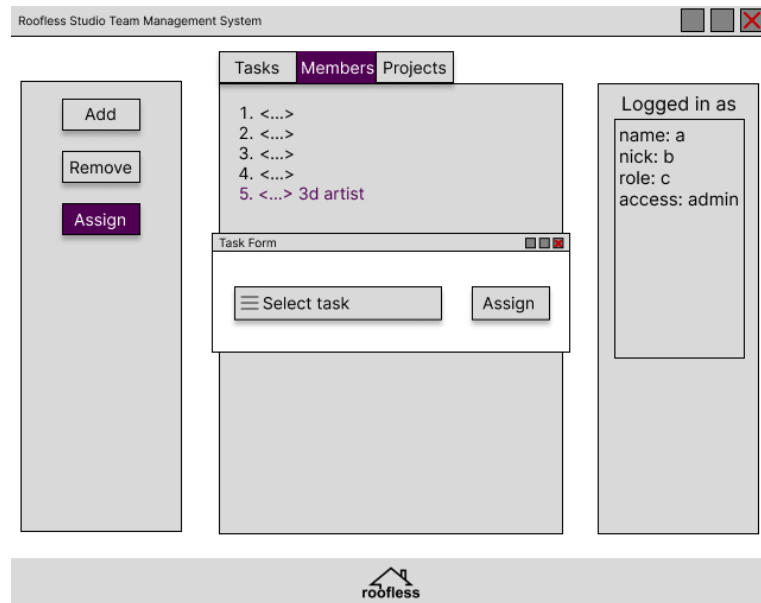
3. Użytkownik klika w przycisk “Members” wyświetlający listę wszystkich członków zespołu



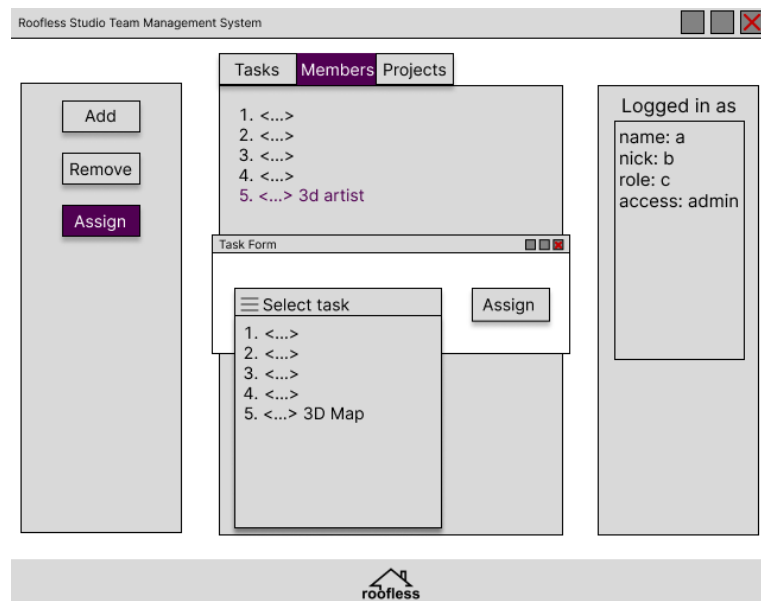
4. Użytkownik wybiera z listy interesującego go członka zespołu



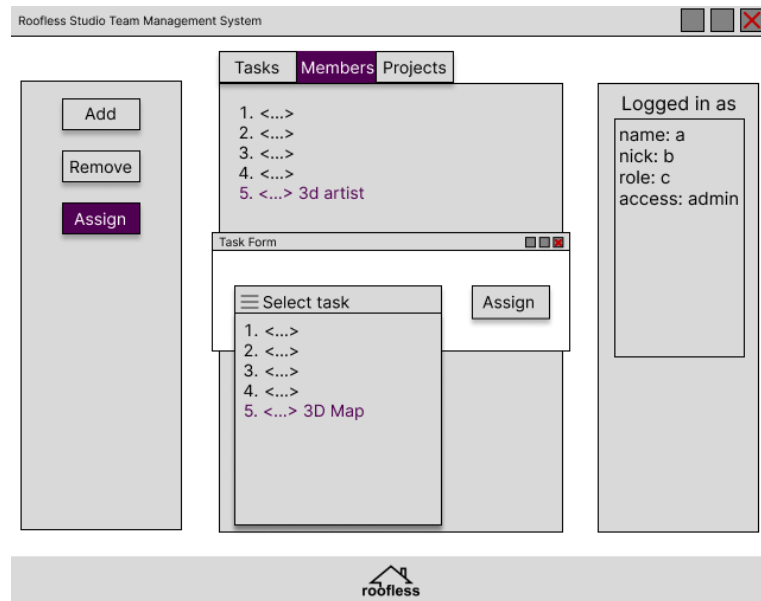
5. Użytkownik klika w przycisk “Assign”



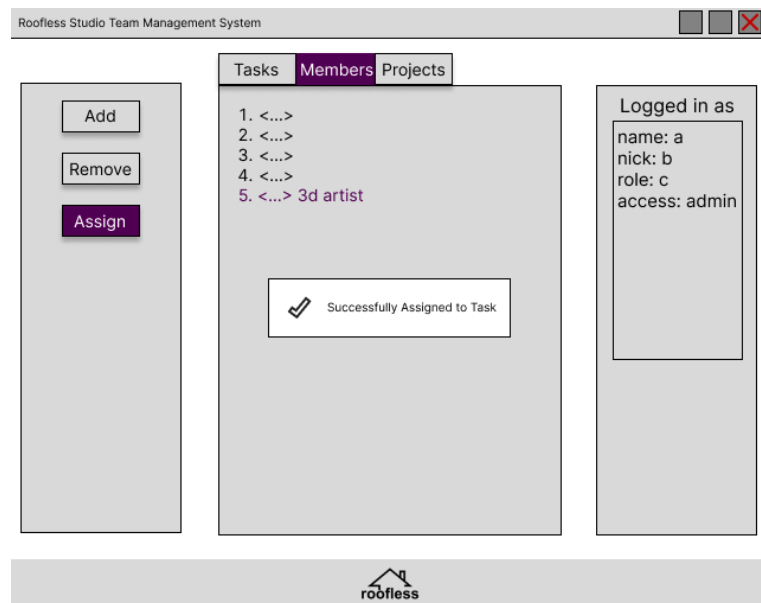
6. Pojawia się nowe okno zawierające listę typu “Drop-Down” z listą dostępnych tasków



7. Użytkownik rozwija listę i wybiera interesujący go task



8. Użytkownik klika w przycisk “Assign”  
9. Okno zamyka się, pojawia się informacja o przypisaniu.



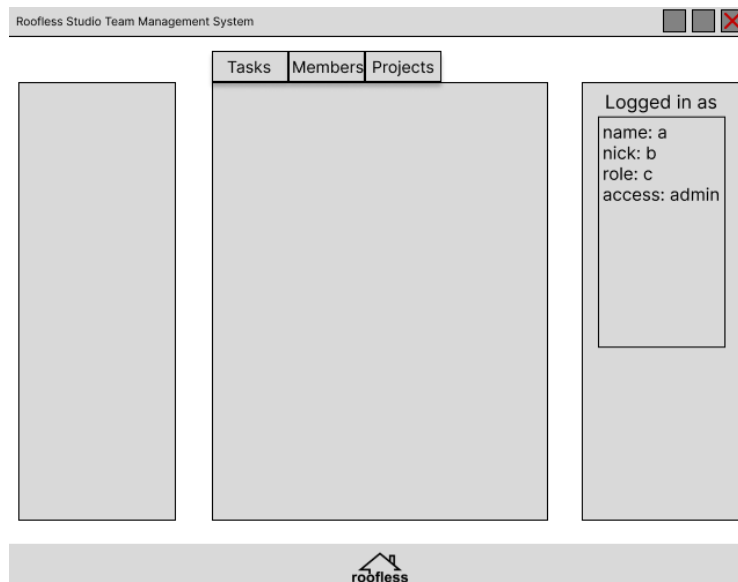
## Przypadek użycia: Dodaj task

1. Użytkownik otwiera aplikację



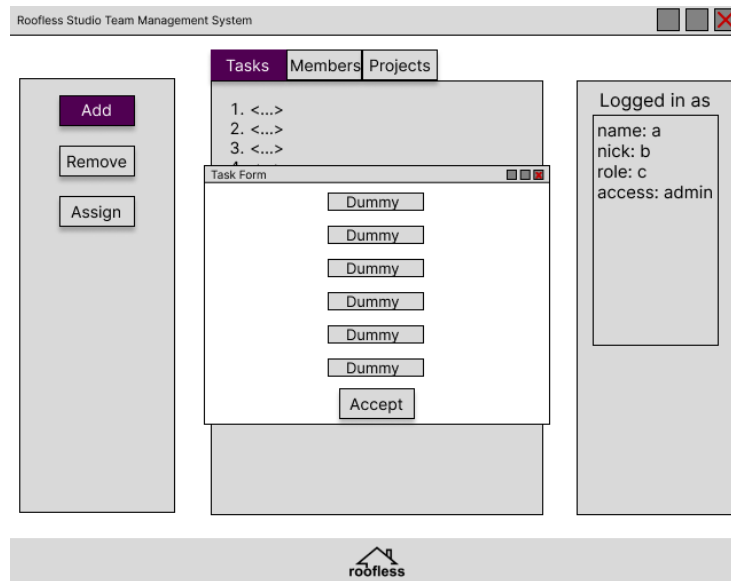
The image shows a window titled "Roofless Studio Team Management System" with standard Windows window controls (minimize, maximize, close). The main content area is titled "User Login" and contains three vertically stacked input fields: "Username", "Password", and a "LOGIN" button. Below the input fields is a horizontal bar with the "roofless" logo.

2. Użytkownik loguje się

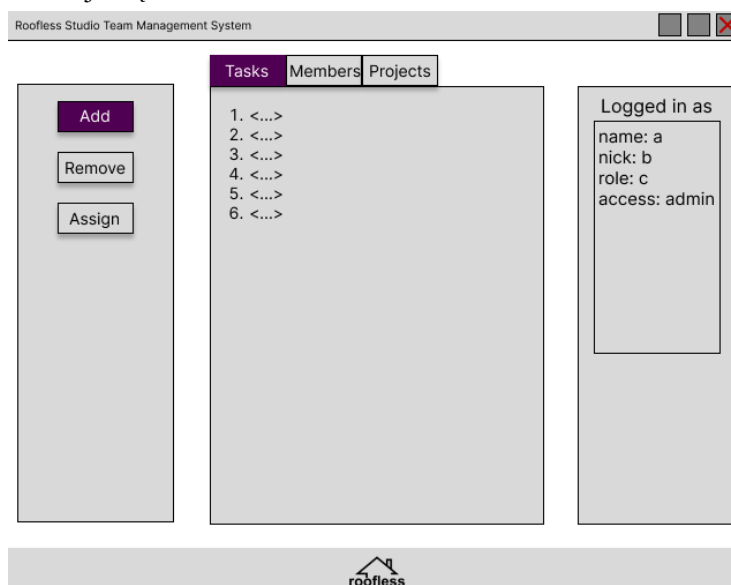


The image shows the main application window after a successful login. The window title is "Roofless Studio Team Management System". The main content area has a tabbed interface with three tabs: "Tasks", "Members", and "Projects". The "Tasks" tab is currently selected. To the right of the main content area is a sidebar with the text "Logged in as" followed by a box containing the user details: "name: a", "nick: b", "role: c", and "access: admin". Below the main content area is a horizontal bar with the "roofless" logo.

3. Użytkownik klika w przycisk "Tasks" wyświetlający listę wszystkich tasków
4. Użytkownik klika w przycisk "Add"



5. Pojawia się nowe okno z formularzem, który użytkownik wypełnia
6. Użytkownik klika w przycisk “Accept”
7. Lista tasków aktualizuje się



## Przypadek użycia: Pokaż najnowsze prace

1. Użytkownik otwiera aplikację



### User Login

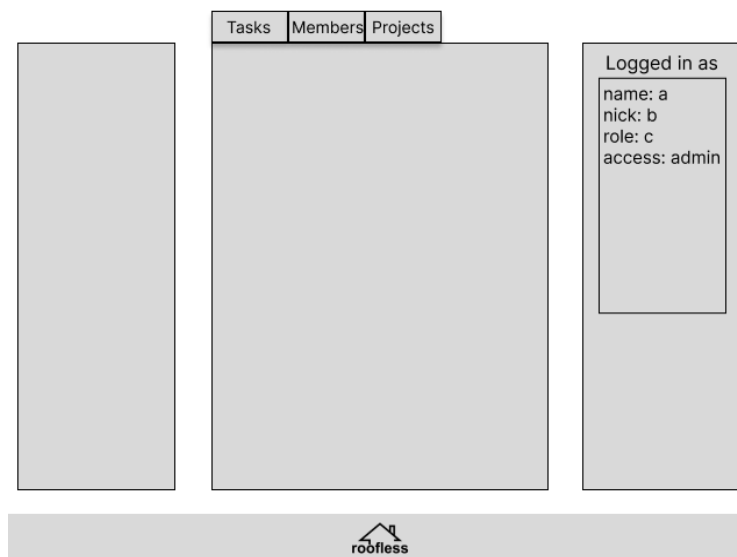
Username

Password

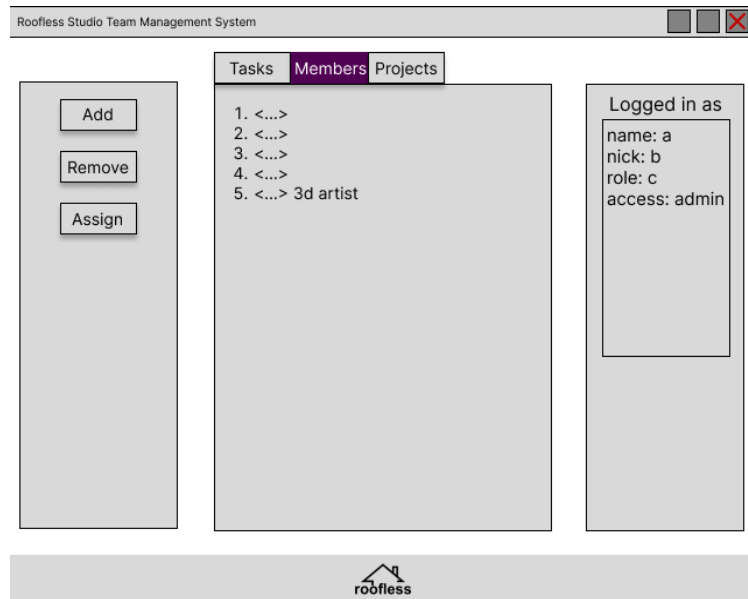
LOGIN



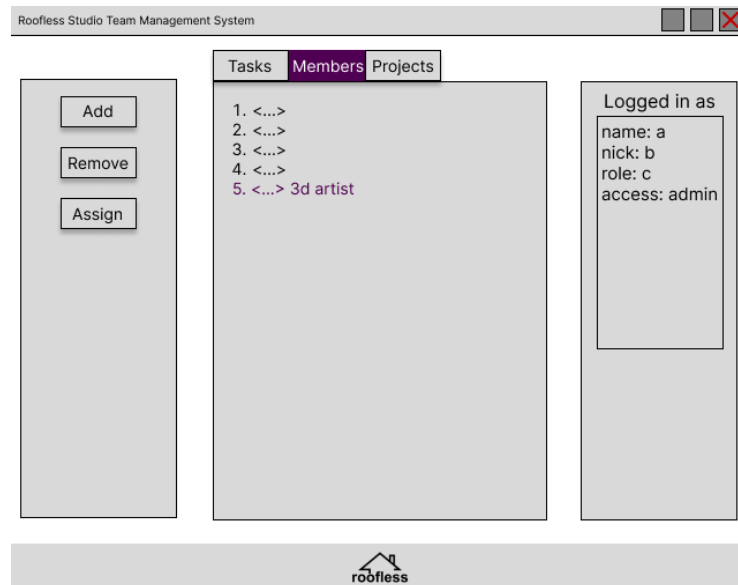
2. Użytkownik loguje się



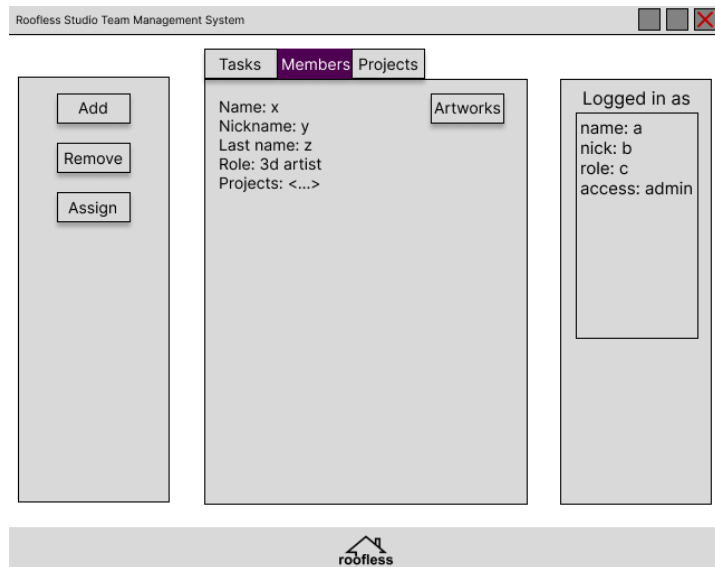
3. Użytkownik klika w przycisk “Members”



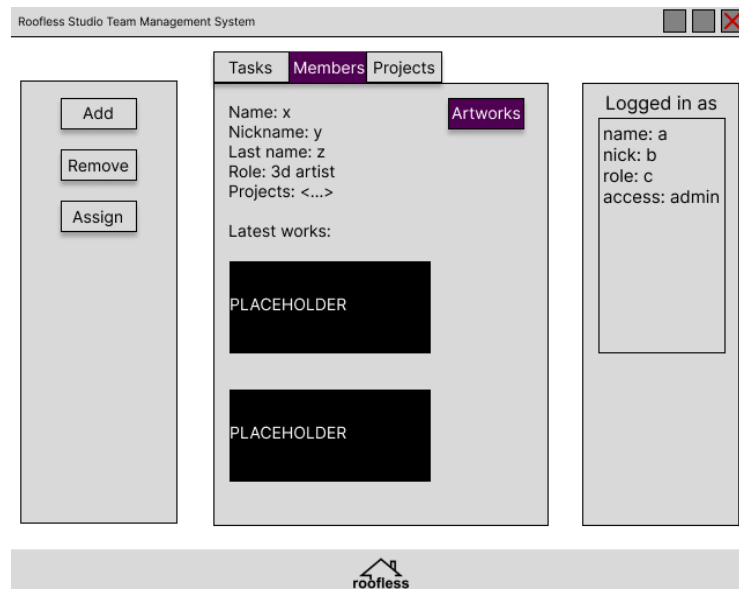
4. Użytkownik wybiera z listy członka o roli artysta



5. Użytkownik klika w przycisk “Artworks”



6. Najnowsze prace członka zespołu pokazują się pod ogólnymi informacjami na jego temat





## 14. Wymagania Niefunkcjonalne

- Interfejs użytkownika powinien być intuicyjny i prosty w obsłudze, nawet dla użytkowników bez doświadczenia w obszarze technologii.
- Status w przypadku klasy asocjacyjnej quest (Task - TeamMember) powinien być zmienną typu String i odpowiadać jednej z trzech opcji: Work in progress, waiting for review, completed.
- Content w przypadku klasy Report: nie więcej niż 1000 znaków.
- Description w przypadku klasy Task: nie więcej niż 150 znaków.
- Daty powinny być zmiennymi typu LocalDate

## 15. Opis Ewolucji Systemu

W przyszłości, system mógłby zostać rozwinięty o dodatkowe moduły jak:

- Moduł analizy danych: Ten moduł może dostarczyć zaawansowane narzędzia analityczne, które pomogą w ocenie wydajności zespołu i projektów.
- Moduł komunikacji: Ten moduł może zawierać zaawansowane narzędzia do komunikacji wewnątrz zespołu. Może obejmować funkcje czatu, powiadomień, dyskusji wątkowych.
- Moduł oceny i opinii: Ten moduł pozwala członkom zespołu oceniać i udzielać opinii na temat projektów, zadań, współpracy i innych członków zespołu. Może zawierać system oceny 360 stopni, w którym członkowie zespołu mogą oceniać siebie nawzajem.

Mógłby również wedle potrzeby zostać rozbudowany o dodatkowe typy członków zespołu:

- Tester
- Level Designer
- Sound Designer