**AUTOR PROJEKTU:**

**Taras Kuts**

**Spis treści**

1 Wstęp 2

Treść zadania 2

Dodatkowe założenia 2

Serwer 2

Klient 2

2 Użyte protokoły 2

Protokół komunikacji klienta z serwerem 3

Wstęp 3

Technologia 3

Funkcjonalność: 3

Dokładny opis funkcjonalny poszczególnych metod/komend 3

Przypadki użycia 4

Aplikacja klienta 11

Wstęp 12

Moduł Algorytm 12

Aplikacja serwera 12

Wstęp 12

Moduł Algorytm 12

Przechowywane dane 12

Wstęp 12

Interfejs danych 12

Spis komunikatów błędów 12

Planowany podział prac 13

* **Wstęp**
* ***Treść zadania***

Aplikacja umożliwia tworzenie i przeprowadzanie egzaminu testowego. 3 rodzaje użytkowników:

administrator, egzaminator i student. Administrator tworzy grupy studenckie i przypisuje do nich studentów. Egzaminator wprowadza egzamin testowy (wszystkie pytania zamknięte) podając dla

wszystkich pytań:

Treść pytania,

Propozycje odpowiedzi (co najwyżej 5),

Poprawną odpowiedź (tylko 1).

Egzaminator przeprowadza egzamin udostępniając go wybranej grupie studentów na określony czas

oraz uruchamia automatyczne sprawdzanie wyników testu dla poszczególnych studentów.

Po sprawdzeniu wynik testu jest przypisywany poszczególnym studentom.

Student loguje się, wprowadza odpowiedzi na poszczególne pytania i ogląda wyniki

* **Serwer**

Serwer wykonany zostanie w języku „C”. Lista studentow,lista testow,lista grup,lista pytan,lista odpowiedzi,lista prawidlowych odpowiedzi będą przechowywane w roznych plikach. Serwer po otrzymaniu zapytania od klienta będzie sprawdzac, który użytkownik jest zalogowany i odpowiednio otrzyma swoje menu(Admin,Egzamentatir,Student).

* **Klient**

Klient będzie aplikacją napisaną w języku „C”. Będzie wysyłał zapytanie do serwera z danymi jaki beda obrobioni na serverze(Authorization-Menu).

* **Użyte protokoły**
* ***Protokół komunikacji klienta z serwerem TCP***
* **Wstęp**

TCP / IP określa sposób wymiany danych za pośrednictwem internetu poprzez zapewnienie bezpośredniego związku, który określa, jak powinna być podzielona na pakiety, adresowana przekazana, transportowana i uzyskana w miejscu przeznaczenia

* ***Technologia***

Podstawowym założeniem modelu TCP/IP jest podział całego zagadnienia komunikacji sieciowej na szereg współpracujących ze sobą warstw.

Każda z nich może być tworzona przez programistów zupełnie niezależnie, jeżeli narzucimy pewne protokoły według których wymieniają się one informacjami. Założenia modelu TCP/IP są pod względem organizacji warstw zbliżone do modelu OSI. Jednak liczba warstw jest mniejsza i bardziej odzwierciedla prawdziwą strukturę Internetu. Model TCP/IP składa się z czterech warstw.

Warstwa:

* Warstwa aplikacji
* Warstwa transportowa
* Warstwa Internetu
* Warstwa dostępu do sieci
* ***Funkcjonalność:***

Protokół ten realizuje następującą funkcjonalność:

Aby protokół TCP wystartował musimy z serwerem jawnie nawiązać kontakt, wysyłamy żądanie i czekamy na odpowiedź, że mamy połączenie. TCP nie jest najszybsze, jednak przy pojedynczych żądaniach nie ma potrzeby, aby działało jak najszybciej.

* **Dokładny opis funkcjonalny poszczególnych metod/komend**

Klient:

* Metoda socket() służy do utworzenia socketu, zwraca socket
* Metoda connect() tworzy polaczenie z serwerem, zwraca informacje czy się udało
* Metoda temp() sluze zabierac enter na koncu
* Metoda studentPanel() loguje na studenta i pozwala go opcje
* Metoda adminPanel() loguje na admina i pozwala go opcje
* Metoda CreateGroup() tworze grupy dla testow(Admin)
* Metoda Add() dodaje uzytkownikow(Admin)
* Metoda AddToGroup() dodaje studentow do grup(Admin)
* Metoda egzaminatorPanel() loguje na egzaminatora i pozwala go opcje
* Metoda AddingAnswers() dodaje odpowiedzi(Egzaminator)
* Metoda rightAnswer() zapisuje prawidlowa odpowiedz(Egzaminator)
* Metoda ChooseCroupOfStudents() wybiera grupe(Egzaminator)
* Metoda Stars() sluze dla szyfrowania hasla pod wygladom "\*"
* Metoda ChooseExamForSstudent() napisana logika studentPanel
* Metoda Getch sluze dla pomocy w szyfrowaniu hasla

Serwer:

* Metoda socketCreate() służy do utworzenia socketu, zwraca socket
* Metoda listenClient() czeka na zadanie od klienta
* Metoda acceptClients() przejmuje klientow
* Metoda portBinding() binduje port
* Metoda authorization() sluze dla ustalenia kto byl login(Admin,Egzamenator,Student)
* Metoda checkLogin() odpowiada za sprawdzenia Logina
* Metoda newConnection() odpowiada za obsluge nowych TCP polaczen
* Metoda getAllList() odpowiada za zapisywanie wszystkich list
* Metoda getStudList() odpowiada za zapisywanie stud list
* Metoda getExList() odpowiada za zapisywanie exams list
* Metoda getGroupList() odpowiada za zapisywanie GroupList
* **Przypadki użycia**

Wlaczamy server jezeli otrzymujemy komunikat ze wlaczania servera bylo sukcesne z tego wychodzi ze server jest wlacznony.Teraz mozemy wlaczac klienta i probowac nawiazac polaczenia(jezeli otrzymujemy komunika ze polaczenie jest nawiazane) automatycznie wyskakuje login i password do wpisania.Wpisujemy login i password jaki sa wysylane do servera gdzie server sprawdza jaki rozdzaj uzytkownika jest zalogowany i zwraca numer odpowiednio jezeli(1 - Admin,2 - Egzaminator,3- Student).Ten numer przyjmuje klient i zalezno od numera wlacza menu.N-d jezeli otrzymujemy numer "1" wlaczamy Admin panel w ktora wchodze nastepne mozliwosci:Create group - czyli admin moze stworzyc grupe i nazwac jak chce,Remove - moze usuniac grupe,Add to list - moze dodawac studentow do listy,StudList - moze zobaczyc liste studentow jaki sa,ExamList - moze zobaczy liste examinow,AllList - moze zobaczyc wszystkie listy i oczywiscie ze jest 'logout' po nacisnienciu jakiego uzytkownik zobacze znowu login i password do wprowadzania.)Teraz jezeli klient otrzymuje numer "2" wlaczamy Egzaminator panel gdzie mamy dostep do nastepujacych funkcij(Start to create test - po nacisnieciu tej knopki egzaminator ma mozliwosc wprowadzic pytania,odpowiedzi i prawidlowa odpowiedz.Spierwszu on wprowadza ilosc pytan i potem w cykli on wprowadza tresc pytania odrazu przyklady odpowiedzej i poprawnu odpowiedz,Choose group - egzaminator wybiera groupu jakiej udostepniaje egzamin,StudList - moze zobaczyc liste studentow jaki sa,ExamList - moze zobaczy liste examinow,AllList - moze zobaczyc wszystkie listy i oczywiscie ze jest 'logout' po nacisnienciu jakiego uzytkownik zobacze znowu login i password do wprowadzania.)I jezeli otrzymujemy numer '3' wlaczamy Student panel gdzie mamy dostep do nastepujacych funkcij(Choose exam - student moze wybrac exam przy warunku ze ten student jest w liscie studentow i tej grupie jest udostepniony egzamin, jezeli tak to student zaczyna swoj egzamin gdzie wkoncu on moze zobaczyc swoj wynik,Logout - oczywiscie ze jest 'logout' po nacisnienciu jakiego uzytkownik zobacze znowu login i password do wprowadzania.)

* **Aplikacja klienta**
* ***Wstęp***

Client.c obsługuje zapyty klienta , a dokładnie wprowadzenie login i password wysyla na server i otrzymuje wynik z pomoca jakiego wybiera ktore menu startowac . Celem jest poprawne zalogowanie i otrzymujesz nalezne menu opcje.

Admin(Tworzy grupy studenckie i przypisuje do nich studentow)

Egzaminator(Wprowadza egzamin testowy podajac dla wszystkich pytan:tresc pytania,propozycje odpowiedzi,poprawna odpowiedz,przeprowadza egzamin udostepniajac go wybranej grupie studentow)

Student(Wprowadza odpowiedzi na pytania i oglada wyniki)

* ***Algorytm***

1)Klient tworzy socket -> connect do serwera -> wywoła funkcje „*Authorization*”

Ta funkcja przyjmuję socket dla dalszego wykorzystania.

2)Authorization prosze wpisać login i haslo do buffera. W tej chwili można otrzymac opcje oczekujące (admin,egzaminator,student)

3)Kiedy dostales 'permission' mozesz skorzystac jedna z opcji

* **Aplikacja serwera**
* ***Wstęp***

Serwer sprawdzaje do jakiego typu nalezy uzytkownik ->Przetworzenia informacji-> Zapisuje informacje w rozne pliki i wysyla na klient

* ***Algorytm***

1)Serwer po otrzymaniu danych,sprawdzaje jaki typ uzytkownika to jest,wysyla go Klientu

2)Przyjmuje dane i zapisuje wszystko w pliki

* **Przechowywane dane**
* Dane są przychowywane w nastepujacy sposob:
* 
* 
* .
* **Interfejs danych**
* ***Spis komunikatów błędów***

**Socked creation failed**

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd == -1)

{

printf("socket creation failed...\n");

exit(0);

}

**Failed binding**

if ((bind(sockfd, (SA\*)&servaddr, sizeof(servaddr))) != 0)

{

printf("socket bind failed...\n");

exit(0);

}

**Listening**

if ((listen(sockfd, 5)) != 0)

{

printf("Listen failed...\n");

exit(0);

}

* **Odebranie pakietu od klienta**
* connfd = accept(sockfd, (SA\*)&cli, &len);
* if (connfd < 0){
* printf("server acccept failed...\n");
* exit(0);
* }

**Plik jest pusty**

if(pFile==NULL)

{

perror("Error opening file.");

}

**Wprowadzony numer istnieje**

**Numer musi być 8 cyfrowy**