MISKOLCI EGYETEM

GÉPÉSZMÉRNÖKI ÉS INFORMATIKAI KAR

Automatizálási és Infokommunikációs Intézet

Beszámoló szakmai gyakorlatról (GEIAL237B)

Pribilián Péter

G25E5O

Vállalat neve: Joyson Safety Systems Hungary Kft.

Üzemi instruktor neve: Kristó Gábor

Üzemi instruktor beosztása: Tesztüzem vezető

Üzemi instruktor telefonszáma: 06 46 407 976

Szakmai gyakorlat időtartama: 2019. július 01. – 2019. augusztus 09.

Miskolc, 2019

Tartalomjegyzék:

1. Bevezetés 3

1.1. Cég bemutatás 3

1.2. A munkám során használt számítógép paraméterei..... 3

1.3. Használt programok 3

1.4. Megismert szakmai ismeretek 4

2. Fő munkáim 4

2.1. Időrendi munkanapló 4

2.2. Bejelentkezés, kijelentkezés, regisztráció 6

2.3. Kapacitástervező oldal..... 7

2.4. Normaidő táblázat..... 10

2.5. CSV fájl letöltése,admin jogosultsággal kapcsolatos módosítások 12

2.6. Beregisztrált felhasználók listája 14

2.7. Tesztelés, dokumentáció..... 15

3. Összegzés, elsajátított ismeretek 16

1. Bevezetés

2019. július 01.-től 2019 augusztus 09.-ig mérnök informatikusként a miskolci telephelyű Joyson Safety Systems Hungary Kft. cég, Tesztelés területén töltöttem a szakmai gyakorlatomat.

1.1 Cég bemutatás

A Joyson Safety Systems Kft. elődjét 1961-ben alapították Breed Corporation néven. Fő központja az Egyesült Államokban található Auburn Hills város Michigan tagállamában található. A Joyson Safety Systems (korábban Key Safety Systems) a Ningbo Joyson Electronic Corporation leányvállalata. A cég számos telephellyel rendelkezik a világban, többek között Magyarországon is (Miskolc). A folyamatos terjeszkedés és fejlődés miatt mára 25 országban 98 telephellyel van jelen a Joyson Safety Systems Kft., munkavállalóinak száma pedig meghaladja az 50 ezer főt.

A Joyson Safety Systems a világ autóipari biztonsági berendezéseinek piacvezető gyártója, feladata biztonsági rendszerek és alkatrészek fejlesztése és gyártása. Széles termékportfóliójában megtalálhatóak autókormányok, légzsákok és gázgenerátorok, biztonsági övek, műanyag alkatrészek, biztonsági gyermekülések, elektronikai termékek és szenzorok is. Ezeket a termékeket szinte minden jelentősebb gépjárműgyártó cégnek szállítja világszerte.

Munkám során a fő feladatom egy olyan tesztkapacitás tervező szoftver létrehozása volt amely megkönnyíti a kapacitástervezéssel foglalkozó beosztottak munkáját azáltal, egy webes felületre kell felvinni a teszteléssel kapcsolatos adatokat és adatbázisba tárolódnak el az adatok az adatvesztés kiküszöbölésének érdekében.

1.2 A munkám során használt számítógép paraméterei:

- Típus: HP Compaq Pro 6300 Microtower
- CPU: Intel Core i3 3220 4x3.30GHz
- RAM: 4GB/1600MHz DDR-3
- HDD: 500GB SATA3
- Operációs rendszer: Windows 7 (64 bites)

1.3 Használt programok:

- XAMPP webszerver-szoftvercsomag
- phpMyAdmin
- Atom Editor
- Notepad++
- Microsoft Word 2017, Excel 2017

1.4 Megismert szakmai ismeretek:

- PHP, MySQL,
- JQuery, JavaScript, HTML, CSS
- Microsoft Word

2. Főbb munkáim

A szakmai gyakorlati időtartam alatt az volt a fő feladatom, hogy létrehozzak egy olyan légzsák tesztkapacitás és raktárkapacitás tervező programot, azonban az idő rövidsége miatt a program légzsák tesztkapacitás tervező részét tudtam csak létrehozni. A gyakorlati munka eredménye a tesztkapacitás tervező program tesztkamrákra lebontva naptárszerkezettel, ami figyelembe veszi a gépi és emberi kapacitást, kiegészítve bejelentkezéssel, illetve kijelentkezéssel, amit könnyedén elérhetnek és használhatnak a hálózatba bekötött számítógépek. A feladataim közé tartozott a kapacitástervező program bővítése egy normaidő táblázattal, amely a tesztek elvégzéséhez szükséges időt tartalmazza. Illetve egy olyan oldal létrehozása amely tartalmazza a beregisztrált felhasználók listáját.
Az alábbiakban a gyakorlat ideje alatt elvégzett főbb feladataimat részletezem.

2.1 Időrendi munkanapló

A szakmai gyakorlat ideje alatt elvégzett munka időrendi beosztása, hetekben megadva:

1. hét: - Apache és MySQL szerver létrehozása és ehhez a megfelelő szoftver megkeresése (XAMPP).
- A bejelentkezési, regisztrációs felület megtervezése.
- A bejelentkezéshez, regisztációhoz a szükséges SQL tábla (users) és adatbázis (register) létrehozása.
- A bejelentkezési, regisztrációs oldal és főoldal létrehozása(login.php, registration.php, index.php) mappában

2. hét: - Kapacitástervező oldal megtervezése
- A kapacitástáblázathoz szükséges SQL tábla (user) és adatbázis (company) létrehozása.
- Külön mappa létrehozása a kapacitástervező oldal fájljainak
- Kapacitástáblázat megtervezése és leprogramozása.
- Kapacitástáblázat kibővítése a “Másolás” “Szerkesztés”, “Törlés”, “Új tesztadat bevitele” és Keresés funkciókkal.

- A Szerkesztéshez és az Új tesztadat beviteléhez egy-egy beviteli oldal készítése (bevitel.php, edit.php) és az ezekhez tartozó olyan oldalak elkészítése amelyek az adatbázisba történő eltárolásért felelnek(add.php, update.php).
- A Törléshez létrehoztam a “delete.php” fájlt ami az adatbázisból való törlést végezte el.

3. hét: - Kapacitástervező naptárfelület megtervezése

- Üres naptárfelület létrehozása, miniatűr naptár létrehozása.
- Naptár fejléc beállítása (hét, nap), különböző megjelenítés és naptártábla beosztás hétre és napra(day_index.php, week_index.php).
- Miniatűr naptár beállítása úgy, hogy nap szerinti megjelenítési oldal esetén az aktuális nap legyen kijelölve a miniatűr naptáron, hét szerinti megjelenítés esetén az aktuális hét legyen kijelölve a miniatűr naptáron.
- Naptártábla felosztása idő szerint.
- Naptáresemény létrehozása és elmentése egy helyi adatbázis fájlba. Ehhez a “create.php” fájl létrehozása.
- Naptáresemény mozgatásának, átméretezésének és törlésének megoldása úgy, hogy módosuljon az adatbázis. Ezekhez a “resize.php”, “move.php” és “delete.php” fájlok létrehozása.
- A kapacitásoldal többszörösítése mind a 13 tesztkamrára, napi és heti megjelenítéssel.

4. hét: - Normaidő táblázat megtervezése.

- A normaidő táblázathoz szükséges SQL tábla (norma.php) létrehozása a “company” adatbázisban.
- Külön mappa létrehozása a normaidő oldal fájljainak
- A normaidő táblázat létrehozása (norma.php) és kibővítése a “Szerkesztés”, “Törlés” és “Adat bevitele” funkciókkal.
- A Szerkesztéshez és az Adat beviteléhez egy-egy beviteli oldal készítése (adatbevitel.php, edit.php) és az ezekhez tartozó olyan oldalak elkészítése amelyek az adatbázisba történő eltárolásért felelnek(add.php, update.php).

5. hét: - A kapacitástáblázat és normaidő táblázat adatainak .csv kiterjesztésű fájlba kiírása, az ehhez szükséges fájl (export.php) leprogramozása.

- A kapacitástáblázat és a normaidő táblázatnál annak beállítása, hogy csak admin tudjon adatot bevinni, módosítani, és törölni

- Külön mappa létrehozása a beregisztrált felhasználókat tartalmazó oldal fájljainak
- Beregisztrált felhasználókat tartalmazó táblázat megtervezése (user.php) úgy, hogy a felhasználókat lekérdezze az adatbázisból.
- Beregisztrált felhasznlálók törlésének megoldása a létrehozott delete.php fájl segítségével

6. hét: - A létrehozott alkalmazás tesztelése a hálózat egyes gépein
- A létrehozott alkalmazás dokumentációjának elkészítése

2.2 Bejelentkezés, kijelentkezés, regisztráció

Mielőtt az alkalmazást elkezdtem volna fejleszteni azelőtt találnom kellett egy olyan programot amellyel lokálisan tudok futtatni PHP fájlokat. Én erre a XAMPP programot választottam mivel az Apache és MySQL szerverek könnyen aktiválhatóak külön konfiguráció nélkül. A program készítése előtt elsajátítottam a PHP, JQuery és JavaScript programozási nyelveket.

Ezután nekiálltam a bejelentkezési és regisztrációs felület létrehozásának amihez az Atom Editor-t használtam.

Első lépésként létrehoztam a phpMyAdmin alkalmazásban egy “register” SQL adatbázist és benne egy “users” SQL táblát amely tartalmazza a felhasználói azonosító (id), felhasználónév (username), e-mail cím (email), jelszó(password) és létrehozási datum(trn_date) mezőket. Ezután a kapcsolatot építettem ki az alábbi PHP kóddal egy db.php fájlba. Ha sikertelen volt a kapcsolat kiépítése akkor error-t írt ki a böngésző:

„<?php $con = mysqli_connect("localhost","root","","register"); if (mysqli_connect_errno())
{ echo "Failed to connect to MySQL: " . mysqli_connect_error();}
?>”

A mysqli_connect() felépítése a következő:
“mysqli_connect("útvonal","felhasználónév","jelszó","adatbázis");”

Az útvonal tetszőleges link vagy ip cím lehet, az adatbázis pedig csak létező adatbázis lehet. A felhasználónév alapértelmezés szerint “root”, a jelszó pedig alapértelmezés szerint üres.

Majd ezután létrehoztam egy login, logout és index nevű PHP fájlokat a bejelentkezéshez, kijelentkezéshez és főoldalhoz.

A login php fejlécében megadtam a “require(‘db.php’)” parancsot ami arra szolgál, hogy a db.php tartalmát meghívja és alkalmazza valamint elindítottam a session-t a session_start() paranccsal. Ellenőriztettem, hogy nem-e üres a felhasználónév az “if (isset($_REQUEST[‘username’]))” paranccsal majd a

“$query = "INSERT into `users` (username, password, email, trn_date)VALUES ('$username', '".md5($password)."', '$email', '$trn_date')";” sorral eltároltam a bejelentkezési adatot az adatbázisban.
A felhasználónevet, jelszót az “input type=text” elemekkel vittem be az adatbázisba submit gombbal.

Ezután létrehoztam a regisztrációs php fájlt. A regisztrációs php fájl (registration.php) tartalma nagyban hasonlít a bejelentkezéses php fájlra, mindkettőnél ugyanúgy szükség van a “require('db.php');” és “session_start();” sorokra a kapcsolódáshoz és a felhasználónév ellenőrzés után ugyanúgy a fenti “$query” sorral tároltam el a regisztrációs adatot az adatbázisban. A felhasználónevet, jelszót és az email címet az “input type=text” elemekkel vittem be az adatbázisba submit gombbal.

Ezt követően létrehoztam az auth.php fájlt azért, hogy a főoldalon kiírathassam a bejelentkezett felhasználó nevét. Ennek tartalma: “<?php session_start();

if(!isset($_SESSION["username"])){
header("Location:login.php");exit();
}?>”

Ezt követően létrehoztam a főoldalt(index.php), ahol elhelyeztem a “include("auth.php");” kódot a felhasználónév kiíratására és linkeket helyeztem el a tesztkamrákra, normaidő táblázatra, dokumentáció letöltésére, kijelentkezésre, és beregisztrált felhasználókra kilistázására vonatkozóan.

A kijelentkezést a “logout.php”fájlban a session_destroy() paranccsal oldottam meg.
A logout.php tartalma:
“<?php
session_start();
if(session_destroy()){
header("Location: login.php");}
?>”

A főoldal, a kijelentkezési és bejelentkezési oldal kinézetét CSS fájl-al oldottam meg.

2.3 Kapacitástervező oldal

A következő feladat volt a kapacitástervező oldal megtervezése és megvalósítása úgy, hogy lehessen adatot törölni, módosítani és létrehozni.
Először létrehoztam egy “company” nevű adatbázist SQL-ben és benne egy “user” nevű táblát. Ebben a táblában tároltam el a tesztelésre vonatkozó összes adatot.

Ezek a következőek: id, sorrend, dátum, tesztkezdet, tesztvég, napszak, program neve, mérnöknév, megrendelésszám, óraszám(a tesztvég és tesztkezdet közötti idő), teszthelyiség, autótípus, teszttípus, légzsáktípus, kameraszám(a teszteléskor használt kamerák száma), megjegyzés.

Ezt követően a fájlok elkülönítése miatt a kapacitástervező oldal elkészítését egy másik mappába kezdtem el.

Ezután a db.php fájlhoz hasonlóan létrehoztam a kapcsolatot az adatbázissal és ezután létrehoztam egy kapacitástáblázatot az előbb felsorolt adatokkal, valamint létrehoztam minden sorhoz egy “Törlés” gombot amellyel törölni lehetett az adott teszteseményt, egy “Másolás” gombot amellyel a teszteseményösszes adatát vágólapra lehetett helyezni, egy “Szerkesztés” gombot amellyel a teszteseményt lehetett szerkeszteni illetve egy “Új tesztadat bevitele” gombot amellyel új teszteseményt lehetett bevinni az adatbázisba.

A táblázat sorainak lekérdezését úgy oldottam meg hogy a <td> tag-ek közé a „<?php echo $row[' ']; ?>” sort adtam meg úgy hogy a négyzetes zárójelben a „user” tábla SQL elemeinek nevét írtam be minden elem esetén.

Mivel a táblázat méretére nem adtam meg korlátozást, ezért hogy átlátható legyen a táblázat sok bejegyzés esetén is, létrehoztam egy szövegdobozt amibe beírva bármilyen információt, rögtön kiszűri a táblázat adatait a kulcsszó alapján és megjeleníti a találatokat.

Plusz funkcióként az oldalhoz létrehoztam egy gombot amely visszairányította a felhasználót a főoldalra.

Ezt követően létrehoztam egy kapacitástáblázat beviteli és szerkesztési weboldalt ami arra szolgál, hogy a teszteseményeket külön-külön tudjuk létrehozni és szerkeszteni. Mindkettő oldalon (bevitel.php és edit.php) form-ok segítségével oldottam meg a bevitelt vagy módosítást submit gombokkal kiegészítve, illetve mindkét oldalhoz létrehoztamegy olyan gombot amely visszairányítja a felhasználót a főoldalra és egy olyan gombot amely visszairányítja a felhasználót a kapacitástervező oldalra. Mindkét oldalon mindkét formban létrehoztam egy action-t ami a beviteli oldal esetén az “add.php”-t hajtja végre, szerkesztés esetén az “update.php”-t hajtja végre.

A beviteli oldalhoz létrehoztam egy “add.php”, a szerkesztési oldalhoz pedig egy “update.php” fájlt. Ezekben a fájlok tartalmazzák a kódot ami szükséges a tesztesemény adatbázisban való módosítására is.

Mindkettő fájl felépítése hasonló: A db.php fájlhoz hasonlóan létrehoztam a kapcsolatot az adatbázissal de más adatbázis névvel:
” $conn = mysqli_connect("localhost","root","","company");”

Majd ezután megadtam azt hogy az összes bevitt érték a neki megfelelő táblamezőhöz kerüljenek bejegyzésre (pl.: $napszak=$_POST[‘napszak’]). Eddig megegyezik az add.php és az update.php tartalma.

Majd ezután az “add.php” esetén

“mysqli_query($conn, “insert into ‘user’ (…)values(…))” kódot hoztam létre zárójelben az összes tesztelésre vonatkozó adat felsorolásával a tesztesemény létrehozásához.

Majd ezután az “update.php” esetén

“mysqli_query($conn, “update ‘user’ set datum=’$datum’…” kódot hoztam létre az összes tesztelésre vonatkozó adat felsorolásával a tesztesemény szerkesztéséhez.

Ezután a Törlés gombhoz létrehoztam a ”delete.php” fájlt amely végrehajtotta a tesztesemény törlését, ha a felhasználó rákattintott a gombra. A db.php fájlhoz hasonlóan létrehoztam a kapcsolatot az adatbázissal (company nevű adatbázis) és a “mysqli_query($conn,"delete from `user` where id='$id'");” sorral elvégezte az adott tesztesemény törlését a program ha rákattintottak a “Törlés” gombra.

Miután megvalósítottam a kapacitástáblázat összes funkcióját, ezután hozzáláttam magának a naptárszerkezetnek a létrehozásához.

A naptárszerkezet létrehozásához úgy kezdtem hozzá, hogy egy olyan felületet képzeltem el a weblapon amely be van osztva idő szerint és ki lehet jelölni egy miniatűr naptáron, hogy melyik nap jelenjen meg és dinamikusan, a felület egy adott időtartamának kijelölésével lehessen megadni a tesztesemény adatait.

Először annak kezdtem neki hogy két részre osszam a kapacitástervező oldalt, egyik rész a kapacitástáblázatnak és a másik rész kapacitástervező naptárnak és a miniatűr naptárnak.

Ezt követően beállítottam a naptár fejlécét úgy, hogy ha kijelöltem egy napot a miniatűr naptáron akkor arra a napra frissült a fejléc. Ezután létrehoztam még egy oldalt(week_index.php) de ott a megjelenítést hét alapúra valtoztattam úgy hogy az üres területet 7 egyenlő oszlopra osztottam fel. Ezen az oldalon a miniatűr táblázatban az egész sor kijelölésével frissült a naptártábla fejlécein az adott dátum.

Majd ezt követően beállítottam, hogy a miniatűr naptáron a napot megjelenítő oldal esetén az aktuális nap legyen kijelölve. Ugyanezt beállítottam a hetet megjelenítő oldal esetén is.

Ezután egy oszlopot hozzáadtam a naptártáblázat elejéhez és felosztottam az üres területet, úgy hogy minden sor egy félórát jelöljön.

Miután végeztem a naptár beállításaival azután hozzáláttam ahhoz, hogy eseményt hozzak létre úgy, hogy annak a kezdete, vége , és a bevitt adatok bekerüljenek egy helyi adatbázisba. Ehhez először létrehoztam egy SQL fájlt ami az eseményeket tárolja (events). Az SQL fájl mezői: id,name, start, end Majd ezt követően létrehoztam egy „create.php” fájlt ami arra szolgált hogy az adatbázisba elmentse a létrehozott eseményt. A „create.php” fájlban a „require_once 'db.php';” kódsorral kapcsolódtam az adatbázishoz. A következő sorral az

adatbázisba való tárolást oldottam meg: „$insert = "INSERT INTO events (name, start, end) VALUES (:name, :start, :end)";”. Ha létrehoztunk egy eseményt a naptártáblában akkor felugrott egy ablak ahol megadhattuk a tesztelés adatait.

Ezután hozzáláttam hogy megoldjam ahhoz, hogy naptártáblában a tesztesemény mozgatását, átméretezését és törlését megvalósítsam. A mozgatáshoz létrehoztam egy „move.php”, az átméretezéshez egy „resize.php”, a törléshez egy „delete.php fájlt”. Mindhárom fájlban a „require_once 'db.php'; ” kódsorral kapcsolódtam az adatbázishoz. A „move.php” és „resize.php” oldalak esetén a mozgatást és az átméretezést az „$insert = " UPDATE events SET start = :start, end = :end WHERE id = :id"; ” sorral oldottam meg. A törlést a „delete.php” esetén a „$insert = "DELETE FROM events WHERE id = :id";” sorral végeztem el.

Miután elvégeztem ezeket a feladatokat, azután kész volt egy olyan felület ahová a kapacitástáblázatból átmásolhatóak voltak az adatok a naptártáblába tesztesemény létrehozása esetén mind nap szerinti, mind hét szerinti megjelenítés esetén.

Ezt követően sokszorosítottam a létrejött kapacitástervező oldalt a napi és heti megjelenítést is beleértve és minden oldalon megjelenítettem az összes kamra linkjét a könnyű elérhetőség érdekében.

Minden tesztkamrához különböző naptártáblát hoztam létre, de a kapacitástáblázatot úgy alakítottam ki, hogy az közös legyen, így nem kell a különböző kamrák között ugrálni, ahhoz hogy az összes teszteseményt lássuk a kapacitástáblázatban.

A kapacitástervező oldalak kinézetét CSS fájl-al oldottam meg

2.4 Normaidő táblázat

A következő feladat az az volt, hogy hozzak létre egy olyan oldalt, ami eltárolja és kiszámolja különböző kritériumok (pl.: tesztkamra, teszttípus, klímakamra kezdő és végállapota) alapján a tesztek elvégzéséhez szükséges időtartamot.

Először létrehoztam SQL-ben egy “norma” nevű táblát a “company” adatbázisban. Ebben a táblában tároltam el a normaidő kiszámításához összes adatot.

Ezek a következőek: id, megrendelésszám, napszak, cop, fejlesztés, tesztidő, tesztelőszám, szorzó, végtesztszám, teszthelyiség, kezdőklíma, végklíma, kezdődarab, végdarab, átállási idő, összidő_nappal, összidő_éjszaka.

A „napszak” az adott napszakot jelöli, a “cop” mező a sorozat-ellenőrzéses teszteket jelenti, a “fejlesztés” a fejlesztéses teszteket jelenti, a “tesztidő” a teszteléshez szükséges alapidő ami a teszttípus alapján számolódik ki, a

“tesztelőszám” a tesztelést végző személyek száma, a “szorzó” a tesztelést végző személyek alapján kerül meghatározásra,a „végtesztszám” az az érték ami a tesztelőszám, teszttípus , szorzó kezdődarab és végdarab alapján számolódik ki, a “teszthelyiség” az adott helyiség ahol a tesztet végzik, a “kezdőklíma” a kamra kezdőállapota, a “végklíma” a kamra végállapota, a “kezdődarab” a kezdőklímán végzett tesztek száma, a “végdarab” a végklímán végzett tesztek száma, az “átállási idő” a kezdő és végállapot között eltelt idő, az „összidő_nappal” az átállási idő + végtesztszám eredménye ha a napszak „Nappal”-ra van állítva, az „összidő_éjszaka” az átállási idő + végtesztszám eredménye ha a napszak „Éjszaka” –ra van állítva.

Ezt követően a fájlok elkülönítése miatt a normaidő táblázatot tartalmazó oldal elkészítését egy másik mappába kezdtem el.

Első lépésként létrehoztam a kapcsolatot az adatbázissal a „norma.php” fájlban a “$conn = mysqli_connect("localhost","root","","company");” sorral majd lekérdeztem az adatokat a

“$query=mysqli_query($conn,"select * from `norma`"); ” sorral. Valamint létrehoztam minden sorhoz egy “Törlés” gombot amellyel törölni lehetett az adott rekordot a táblázatból, egy “Szerkesztés” gombot amellyel a bevitt normaidő rekordot lehetett szerkeszteni illetve egy “Adat bevitele” gombot amellyel új teszteseményt lehetett bevinni az adatbázisba.

Ezt követően létrehoztam egy táblázatot amely fejléceinek a fent említett mezőneveket adtam meg. A táblázat sorainak lekérdezését úgy oldottam meg hogy a <td> tag-ek közé a „<?php echo $row[' ']; ?>” sort adtam meg úgy hogy a négyzetes zárójelben a „norma” tábla SQL elemeinek nevét írtam be minden elem esetén.

Mivel a táblázat méretére nem adtam meg korlátozást, ezért hogy átlátható legyen a táblázat sok tesztelem esetén is, létrehoztam egy szövegdobozt amibe beírva bármilyen információt, rögtön kiszűri a táblázat adatait a kulcsszó alapján és megjeleníti a találatokat.

Plusz funkcióként az oldalhoz létrehoztam egy olyan gombot amely visszairányította a felhasználót a főoldalra.

Ezt követően létrehoztam egy normaidő táblázat beviteli és szerkesztési weboldalt ami arra szolgál, hogy a normaidő táblázat rekordjait külön-külön tudjuk létrehozni és szerkeszteni. Mindkettő oldalon (adatbevitel.php és edit.php) form-ok segítségével oldottam meg a bevitelt vagy módosítást submit gombokkal kiegészítve, illetve mindkét oldalhoz létrehoztam egy olyan gombot amely visszaállítja az űrlapot, egy olyan gombot amely visszairányítja a felhasználót a főoldalra és egy olyan gombot amely visszairányítja a felhasználót a normaidő táblázat oldalára. Mindkét oldalon mindkét formban létrehoztam egy action-t ami a beviteli oldal esetén az “add.php”-t hajtja végre, szerkesztés esetén az “update.php”-t hajtja végre.

A beviteli oldalhoz létrehoztam egy “add.php”, a szerkesztési oldalhoz pedig egy “update.php” fájlt. Ezekben a fájlok tartalmazzák a kódot ami szükséges a normaidő táblázat rekordjainak adatbázisban való módosítására is.

Mindkettő fájl felépítése hasonló: A norma.php fájlhoz hasonlóan létrehoztam a kapcsolatot az adatbázissal de más adatbázis névvel:
” $conn = mysqli_connect("localhost","root","","company");”

Majd ezután megadtam azt hogy az összes bevitt érték a neki megfelelő táblamezőhöz kerüljenek bejegyzésre (pl.: $cop=$_POST[‘cop’]).
Eddig megegyezik az add.php és az update.php tartalma.

Majd ezután az “add.php” esetén

“mysqli_query($conn, “insert into ‘norma’ (cop, ...)values('$cop', …));” kódot hoztam létre zárójelben az összes normaidőre vonatkozó adat felsorolásával a tesztesemény létrehozásához.

Majd ezután az “update.php” esetén

“mysqli_query($conn,“update ‘norma’ set napszak=’$napszak’, ...);” kódot hoztam létre az összes normaidőre vonatkozó adat felsorolásával a tesztesemény szerkesztéséhez.

Ezután a Törlés gombhoz létrehoztam a ”delete.php” fájlt amely végrehajtotta a táblázat rekordjának törlését, ha a felhasználó rákattintott a gombra. A norma.php fájlhoz hasonlóan létrehoztam a kapcsolatot az adatbázissal (company nevű adatbázis) és a “mysqli_query($conn,"delete from `user` where id='$id'");” sorral elvégezte az a táblázat adott rekordjának törlését a program, ha rákattintottak a “Törlés” gombra.

A normaidő táblázatot tartalmazó oldal kinézetét CSS fájl-al oldottam meg.

2.5 CSV fájl letöltése, admin jogosultságokkal kapcsolatos módosítások

A következő feladat az volt hogy meg kellett oldani azt hogy a kapacitástáblázat és a normaidő táblázat adatai lementhetőek legyenek egy *.csv kiterjesztésű fájlba. Először minden kamra kapacitástervező oldalára vonatkozóan hoztam létre egy „export.php” fájlt ami a kapcsolat felépítése után („” $conn = mysqli_connect("localhost","root","","company");”) lekérdezte a kapacitástervező SQL tábla adatait($query = "SELECT * FROM user";) és ha a sorok száma nagyobb volt mint 0
(if (mysqli_num_rows($result) > 0) {
while ($row = mysqli_fetch_assoc($result))
{$norma[]=$row}), akkor kiírja fájlba az „fputcsv(array(...))” parancssal.

Majd ezután a normaidő táblázat mappájában is létrehoztam egy „export.php” oldalt ami a kapcsolat felépítése után („” $conn = mysqli_connect("localhost","root","","company");”) lekérdezte a kapacitástervező SQL tábla adatait($query = "SELECT * FROM norma";) és ha a sorok száma nagyobb volt mint 0
(if (mysqli_num_rows($result) > 0) {
while ($row = mysqli_fetch_assoc($result))
{$norma[]=$row}), akkor kiírja fájlba az „fputcsv(array(...))” parancssal.

A következő feladat az volt, hogy a kapacitástáblázatnál és a normaidő táblázatnál olyan beállítást kellett végrehajtani, hogy csak admin tudjon adatot bevinni, módosítani, és törölni.

Ezt úgy tudtam megoldani a kapacitástervező oldalak esetében, hogy módosítottam a mappában lévő összes „add.php”, „delete.php” és „edit.php” fájlokat úgy, hogy megadtam egy hivatkozást a korábban létrehozott „auth.php” fájlra az „include '../auth.php';” paranccsal majd a lekérdezéseket(mysqli_query(...)) egy if ciklusban helyeztem el: „if($_SESSION['username'] == 'admin'){}” , így minden módosítás, törlés, létrehozás esetén csak akkor hajtódott végre a művelet, ha az admin van bejelentkezve, egyébként csak frissítette az oldalt. Mivel több kapacitástervező oldal létezett (egészen pontosan 13, mivel ennyi kamra áll rendelkezésre a légzsák teszteléshez) ezért mindegyiknél végig kellett hajtani ezeket a módosításokat. A fájlba íráshoz egy „CSV fájl kiírása” gombot használtam amit lenyomva választhattam a mentés és a megnyitás opciók közül.

A normaidő táblázatos oldal esetén a kapacitástervező oldalakhoz hasonlóan oldottam meg az adminisztrátori jogosultsággal kapcsolatos feladatot: Módosítottam a mappában lévő „add.php”, „delete.php” és „edit.php” fájlokat úgy, hogy megadtam egy hivatkozást a korábban létrehozott „auth.php” fájlra az „include '../auth.php';” paranccsal majd a lekérdezéseket (mysqli_query(...)) egy if ciklusban helyeztem el: „if($_SESSION['username'] == 'admin'){}” , így minden módosítás, törlés, létrehozás esetén csak akkor hajtódott végre a művelet, ha az admin van bejelentkezve, egyébként csak frissítette az oldalt. A fájlba íráshoz egy „CSV fájl kiírása” gombot használtam amit lenyomva választhattam a mentés és a megnyitás opciók közül.

Az adminisztrátori jogosultsággal kapcsolatos módosításokat elvégezve a kapacitástervező oldalakon és a normaidő táblázatos oldalon, elkészültem a program kapacitástervező és normaidő táblázatos részével.

2.6 Beregisztrált felhasználók listája

A következő feladat az volt, hogy létrehozzak egy olyan felületet ami kilistázza a felhasználók listáját és ha adminisztrátori jogosultsággal rendelkezik a bejelentkezett személy akkor képes legyen törölni a táblázatból és egyúttal az adatbázisból is.

Először a fájlok elkülönítése miatt a bejelentkezett felhasználók listáját tartalmazó oldal elkészítését egy másik mappába kezdtem el.

Majd létrehoztam egy „user.php” fájlt. A fájlban első lépésként létrehoztam a kapcsolatot a „register” adatbázisban lévő „users” mappához: “$conn = mysqli_connect("localhost","root","","register");” sorral majd lekérdeztem az adatokat a
“$query=mysqli_query($conn,"select * from `users`"); ” sorral.

Ezt követően létrehoztam egy táblázatot amelyet a „users” SQL tábla adataival töltöttem fel.
A táblázat sorainak lekérdezését úgy oldottam meg hogy a <td> tag-ek közé a „<?php echo $row[' ']; ?>” sort adtam meg úgy hogy a négyzetes zárójelben a „users” tábla SQL elemeinek nevét írtam be minden elem esetén (pl.: <?php echo $row['username']; ?>).
Minden sorhoz létrehoztam egy „Törlés” gombot amire kattintva a felhasználó(ha rendelkezik a megfelelő jogosultsággal) ki tudja törölni az adott felhasználót a táblázatból és az adatbázisból.

A Törlés gombokhoz létrehoztam a ”delete.php” fájlt amely végrehajtotta a táblázatból a kijelölt felhasználó törlését, ha a felhasználó rákattintott a gombra. A „user.php” fájlhoz hasonlóan létrehoztam a kapcsolatot az adatbázissal (register nevű adatbázis) és a “mysqli_query($conn,"delete from `users` where id='$id'");” sorral elvégezte az a táblázat adott sorának törlését a program, ha rákattintottak a sorhoz tartozó “Törlés” gombra.

Az adminisztrátori jogosultsággal kapcsolatos módosításokat úgy oldottam meg, hogy a „delete.php” fájlban megadtam egy hivatkozást a korábban létrehozott „auth.php” fájlra az „include '../auth.php';” paranccsal majd a lekérdezést (mysqli_query(...)) egy if ciklusban helyeztem el: „if($_SESSION['username'] == 'admin'){}” így minden törlés esetén csak akkor hajtódott végre a művelet, ha az admin van bejelentkezve, egyébként csak frissítette az oldalt.

A bejelentkezett felhasználók listáját tartalmazó oldal kinézetét CSS fájl-al oldottam meg.

Miután elvégeztem a szükséges adminisztrátori jogosultsággal kapcsolatis módosításokat, azután teljesítettem ezt a feladatrészt is.
Ezzel megcsináltam a programot, ezután már csak a tesztelés és dokumentáció elkészítése maradt hátra.

2.7 Tesztelés, dokumentáció

Miután készen voltam a programmal, azt teszteltem sikeresen, hogy hogyan működik a program a hálózat többi gépén. Ezután elhelyeztem egy olyan gépen a programot amely állandóan bekapcsolt állapotban van, így a felhasználók bármikor el tudják érni a programot.

Ezt követően részletes dokumentációt készítettem a program működéséről, a szerver használati instrukciókról és a lehetséges hibákról valamint ezek javításáról.

3. Összegzés, elsajátított ismeretek

A szakmai gyakorlat ideje alatt, úgy hiszem elég sok ismeretanyagot sajátítottam el, mint például a PHP nyelv, amit ezelőtt gyakorlatban nagyon ritkán használtam.

Szerintem nagyon hasznos volt a tapasztalatszerzés céljából, hogy egy önálló projektet meg tudtam valósítani a cég segítségével. Illetve, elég fontosnak tartom azt, hogy elsajátítottam azt hogy hogyan készítsek dokumentációt egy komplex programról.

Biztos vagyok benne, hogy a szakmai gyakorlat alatt elsajátított összes tudás és ismeret nagy hasznomra fog válni a jövőben.