Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра “Системи автоматизованого проектування”

**ЗВІТ**

про практику за темою бакалаврської кваліфікаційної роботи

студента ІV курсу групи КН-44

Дацка Любомира Андрійовича

**База практики** Кафедра САПР НУ «Львівська політехніка»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(назва та адреса підприємства)

**Термін практики** з « » квітня 2018 р. до « » травня 2018 р.

**Керівники практики**

від кафедри \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_к.т.н., доцент, Фармага І.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (наук.ст, вч.звання, прізвище, ім’я, по батькові)

**Керівник бакалаврської кваліфікаційної роботи**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_ас.кафедри, Дупак Б.П.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (наук.ст, вч.звання, прізвище, ім’я, по батькові)

**Тема:** Розроблення автоматизованої підсистеми для допомоги оператору при розподіленні автомобілів швидкої допомоги

**ОЦІНКА** \_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Дата** \_\_\_\_\_\_\_\_\_\_\_\_\_

Львів – 2018

**Зміст**

1. Опис бази практики……………………………………………………..……3

2. Апаратні та програмні засоби комп’ютерних систем……………….….….3

3. Опис проблеми та огляд існуючих аналогів системи ……………………..4

4. Опис вимог…………………………………………………………………..8

5. Тестування роботи системи……………………………………………….14

6. Приклади реалізації……………………………………………………….15

Висновки…..…………………………………………………………………....19

Список використаної літератури та посилань ………………………………20

Додаток А………………………………………………………………………21

**1. Опис бази практики**

Проходження практики здійснювалося на кафедрі САПР НУ «Львівська політехніка» .

Кафедра "Системи автоматизованого проектування" (САПР) створена в 1989 році як профілююча з метою підготовки інженерів за спеціальністю "Системи автоматизованого проектування" в складі радіотехнічного факультету.

З травня 2000 року завідувачем кафедри САПР є Лобур Михайло Васильович.

Територіально кафедра розташована в IV навчальному корпусі (318-324 аудиторії). В розпорядженні кафедри 5 навчальних лабораторій, оснащених сучасними комп'ютерами (107 комп'ютерів), які підключені в загально-університетську мережу з можливістю доступу до Інтернету.

.

**2. Апаратні та програмні засоби комп’ютерних систем**

Проходження практики та виконання індивідуального завдання здійснювалося на ПК з такими технічними характеристиками:

Операційна Система

Windows 10 Enterprise 64-bit

CPU

Intel Core i5 3570K @ 3.40GHz 45 °C

Ivy Bridge 22nm технологія

Пам`ять

8,00ГБ двойний -Канал DDR3 @ 686МГц (9-9-9-27)

Материнська Плата

ASUSTeK COMPUTER INC. P8Z77-V LK (LGA1155) 28 °C

Графіка

IPS234 (1920x1080@60Hz)

SAMSUNG (1920x1080@60Hz)

8192МбATI AMD Radeon R9 390 Series (MSI) 62 °C

Storage

931GB Seagate ST1000DM003-9YN162 (SATA) 24 °C

74GB INTEL SSDSA2M080G2GN (SSD)

931GB Seagate Backup+ RD USB Device (USB (SATA)) 31 °C

Оптичні Приводи

не знайдено опціальних дисків

Аудіо

AMD High Definition Audio Device

Також для виконання завдання було встановлено набір програмних засобів:

* avast!
* Microsoft Office 2013
* Adobe Reader XI (11.0.11)
* DjVu Reader

3. Опис проблеми та огляд існуючих аналогів системи

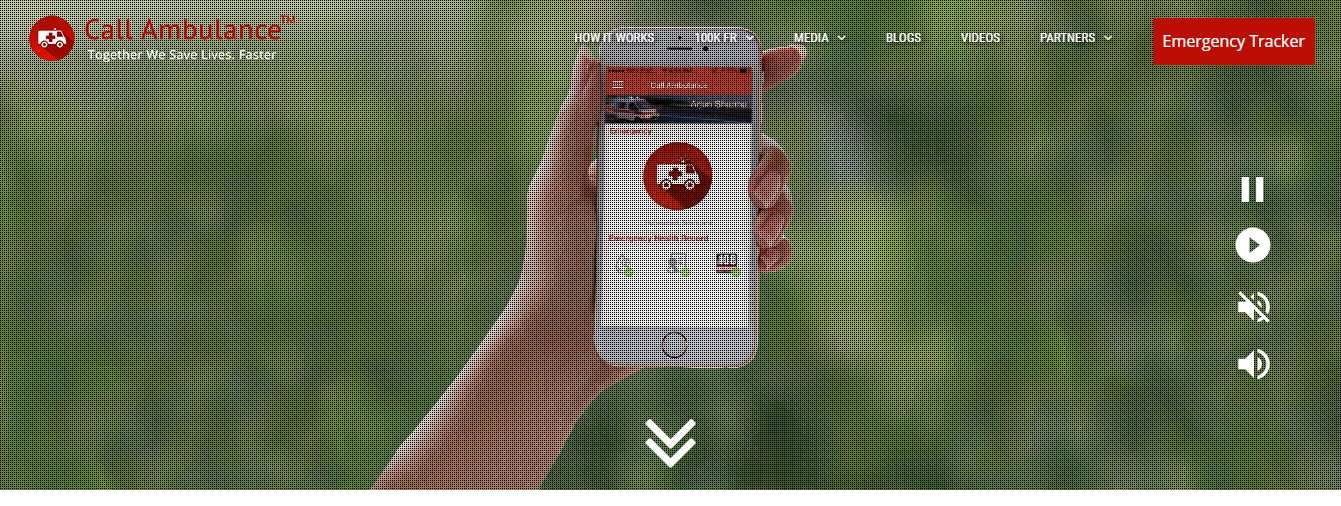
*3.1 Опис проблеми*

Моєю ідеєю є створення сервісу, який пригодиться для оператора при розподіленні автомобілів швидкої допомоги. Припустимо, що в одному районі поступає 10 викликів з різних вулиць, але в розпорядженні лікарні є лише 3 автомобілі швидкої допомоги. Оператор фіксує ці звернення , вводить дані в програму, відслідковується машина, яка знаходиться найближче до місця виклику і вона виїжджає.

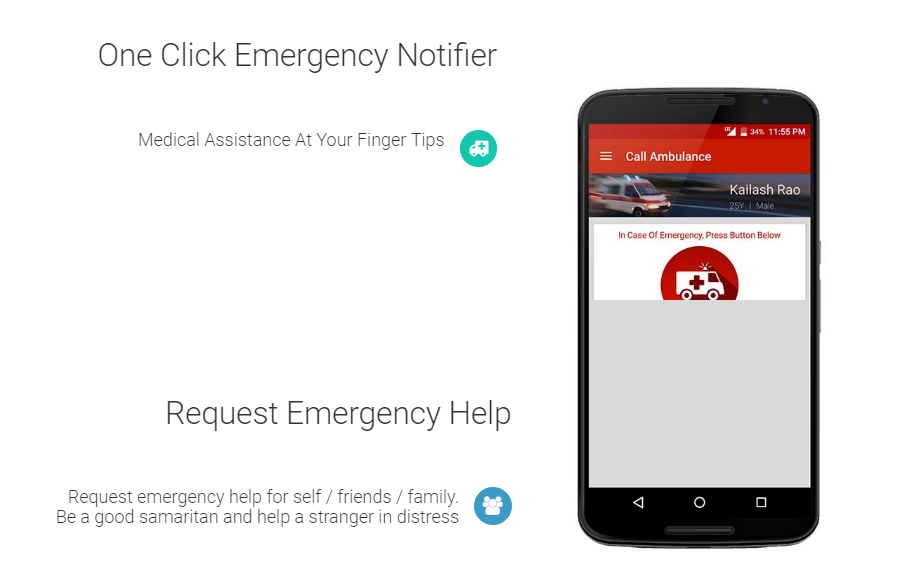
*3.2 Аналіз існуючих програмних засобів*

Сьогодні існує один відомий сервіс, який схожий до мого завдання – Call ambulance. Це додаток, який дозволяє швидко повідомити всіх про пригоду, прокласти шлях водію до місцязнаходження пацієнта, знає твою групу крові, а також повідомляє чергового лікаря про пригоду

Нижче представлені скріншоти з описом цього сервісу

****

****

****

**4. Опис вимог**

*4.1 Функціональні вимоги*

Система працює в режимі оператора та пацієнта.

У режимі оператора має доступ до таких функцій:

* Позначати місце виклику;
* Прокладати шлях водію автомобіля до місця події;
* Редагувати пріоритетність викликів;

У режимі оператора має доступ до таких функцій :

* Вхід в систему;
* Додавання персональної інформації;
* Додавання членів сім’ї та лікаря

*4.2 Нефункціональні вимоги*

Вимоги до продукту :

* Комп’ютер-сервер, на якому буде розміщена база даних повинен мати процесор з частотою не менше 2.0 GHz, вінчестер об’ємом 1 ТB та оперативною пам’яттю не менше 8 Gb. Операційна система MS Windows із встановленою програмою Міcrosoft SQL server 2008 SP1.
* Локальна мережа, вихід в Інтернет.
* Швидкодія Інтернету повинна бути не меншою 256 Кбіт/сек.
* Наявність веб-браузера, який підтримується операційною системою

Вимоги до процесу:

* Використання протоколу TCP/IP.
* Використання протоколу HTTP/HTTPS.
* Можливість одночасної роботи в системі ~ 100 клієнтів.
* Можливість одночасної обробки ~ 50 запитів.

Зовнішні вимоги:

* Операційна система МS Windows.
* Мова інтерфейсу – англійська, українська, російська.
* Комп’ютер-сервер.
* Локальна мережа, вихід в Інтернет.
* Тип мережі: Ethernet або Fast Ethernet.
* Швидкодія Інтернету повинна бути не меншою 256 Кбіт/сек.

*Таблиця 1. Опис зовнішніх вимог*

|  |  |
| --- | --- |
| Об’єм | Можливість одночасної роботи в системі ~ 10000 клієнтів. |
| Швидкість реакції системи на запит | Можливість одночасної обробки ~ 1000 запитів. |
| Апаратні засоби | Апаратна частина серевера:   * Процесор Intel Core I5 3570K * Вінчестер 2 TB * Відеокарта 8196 MB * Оперативна память DDR3 8GB 1667 MHz   Апаратна частина клієнтського компютера:   * Локальна мережа , вихід в Інтернет. * Процесор Intel Pentium II 800 MHz * Вінчестер 4 GB * Відеокарта 64 Mb * Оперативна пам’ять DDR 128MB * Мережева карта: Atheros AR5B97 * Монітор * Клавіатура * Мишка * Принтер   Вимоги до мережі:  Швидкодія Інтернету повинна бути не меншою 1 Мбіт/сек.  Вимоги до офісу:   * Температура повітря,­­­­0­С – 15 – 30 * Вологість повітря, % – 40-90 |
| Програмний інтерфейс | * Сумісність з ОС WINDOWS.   Програмна чаcтина сервера:   * Microsoft SQL server 2008 SP1. |
| Взаємодія людини з системою | Для роботи користувача з програмою достатньо: - встановити програму на ОС WINDOWS;  - мати вихід в Інтернет чи локальну мережу.  Також є можливість здійснення замовлення за допомогою web-браузера. Для цього достатньо мати вихід у Інтернет чи локальну мережу. |
| Безпека | База даних написана мовою SQL і розміщена на сервері.  Для захисту використовують стандартні методи програми MS SQL Server, такі як :   * Правила безпеки; * Обмеження доступу до файлів SQL Server; * Захист клієнтів; * Антивірус; * Захист від атаки «SQL Injection»   При неправильному завершенні роботи втрати даних не буде, оскільки клієнт працює напряму з базою. Всі зміни здійснені в клієнті одразу передаються базі даних. Втратитися можуть лише ті зміни, які були в процесі виконання. |
| Гнучкість | Клієнтські програми працюють практично на всіх версіях MS Windows.  Програма передбачає роботу як через локальну мережу, так і через Інтернет  Є можливість проведення тестувань або опитувань думки за допомогою Web-браузера. Таким чином клієнтська сторона є незалежною від операційної системи.  Інтерфейси користувачів передбачають налаштування під власні потреби. |
| Ресурси | Обмеження строго по бюджету  3 000 грн. |

*4.3 Концептуальна модель*

1. Результат логування. .

1. Позначення місця виклику;
2. Прокладання шляху водію автомобіля до місця події;
3. Редагування пріоритетність викликів;

1. Запит на логування.

1. Запит на позначення місця виклику;
2. Запит на прокладання шляху водію автомобіля до місця події;
3. Запит на редагування пріоритетність викликів;

Оператор

1. Успішне логування.

2. Успішне додавання персональної інформації

3. Успішне додавання членів сім’ї та лікаря.

1. Запит на логування.

2. Запит на .додавання персональної інформації

3. Запит на додавання членів сім’ї та лікаря.

Пацієнт

**4. Тестування роботи системи**

Як відомо, програм без помилок не існує. Практика доводить, що винуватцями помилок у програмах найчастіше бувають самі програмісти. Один із загальних законів практичного програмування полягає в тому, що жодна програма не дає бажаних результатів при першій спробі трансляції та виконання.

Існують два типи програмних помилок:

*синтаксичні помилки* - виникають через порушення правил мови програмування. Такі помилки зазвичай виявляються під час компіляції. Можуть бути виключені порівняно легко. Навіть якщо не переглядати текст програми можна бути впевненим, що компілятор на стадії трансляції знайде помилки і видасть відповідні попередження. Фактично пошук помилок здійснює компілятор, а їхнє виправлення - програміст;

*семантичні (логічні) помилки -* ті, що призводять до некоректних обчислень або помилок під час виконання (run-time error). Семантичні помилки усувають зазвичай за допомогою виконання програми з ретельно підібраними перевірочними даними, для яких відома правильна відповідь.

У широкому сенсі, тестування – це одна з технік контролю якості (Quality Control), яка включає планування, складання тестів, безпосередньо виконання тестування і аналіз отриманих результатів.

Загалом, тестування програмного забезпечення (Software Testing) - це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись. Техніка тестування також включає як процес пошуку помилок бо інших дефектів, так і випробування програмних складових з метою оцінки.

Метою тестування даного проекту є перевірка усіх функціональних можливостей програмного продукту, а також коректність роботи програмного продукту у різних браузерах та при використанні моніторів з різним розширенням.

Потрібно оцінити:

* відповідність вимогам, якими керувалися розробники
* правильна відповідь для усіх можливих вхідних даних
* виконання функцій за прийнятний час
* практичність
* сумісність з програмним забезпеченням та операційними системами
* відповідність задачам замовника.

Оскільки число можливих тестів навіть для нескладних програмних компонент практично нескінченне, тому стратегія тестування полягає в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів. Як результат програмне забезпечення (ПЗ) тестується стандартним виконанням програми з метою виявлення багів (помилок або інших дефектів).

Виконане мною тестування включало такі етапи:

* Планування робіт (Test Management)
* Проектування тестів (Test Design)
* Виконання тестування (Test Execution)
* Аналіз отриманих результатів (Test Analysis).

Тестування проводиться у три рівні:

1) Модульне тестування полягає в перевірці працездатності та коректності роботи модулів програми окремо один від одного.

2) Інтеграційне тестування проводиться до тих пір, поки тестовані на попередньому кроці окремі модулі програми, не почнуть працювати як система.

3) На останньому рівні системного тестування інтегрована система перевіряється на відповідність всім вимогам. Крім того, системне тестування ПЗ повинно гарантувати, що програма працює так, як очікувалося, а також, що її не можна знищити чи перевести систему в неробочий стан. Системне інтеграційне тестування перевіряє, чи система інтегрується в будь-яку зовнішню систему.

Існує багато підходів до тестування програмного забезпечення, але ефективне тестування складних продуктів - це по суті дослідницький процес, а не тільки створення і виконання рутинної процедури.

Тестування- процес також ітераційний. Після виявлення та виправлення кожної помилки обов'язково слід повторити тести, щоб переконатися у працездатності програми. Більше того, для ідентифікації причини виявленої проблеми може знадобитися проведення спеціального додаткового тестування. При цьому потрібно завжди пам'ятати про фундаментальний висновок, зроблений професором Едсжером Дейкстри у 1972 році: "Тестування програм може служити доказом наявності помилок, але ніколи не доведе їхню відсутність!".

Слід зазначити, що якість не є абсолютною величиною. Це суб’єктивне поняття. Тому тестування як процес своєчасного виявлення дефектів не може повністю забезпечити коректність програмного забезпечення. Воно тільки порівнює стан та поведінку продукту зі специфікацією.

У результаті аналізу підсумків проведених тестувань та верифікації можна зробити висновок, що наш програмний продукт відповідає усім вимогам, потрібні функції реалізовано правильно, інтерфейс зручний та простий у використанні, системи захищена, якісна та надійна.

На етапі було проведено тестування проекту згідно плану розробленого на етапі реалізації.

Було розглянуто основні види тестувань і з них обрано найоптимальніші для даної системи.

* **Метод прозорої скриньки** був обраний, оскільки він дозволяє визначити джерело помилки безпосередньо в програмному коді.
* **Метод чорної скриньки** також був обраний, оскільки він базовим та поростим, і не вимагає заглиблення у принципи функціонування сиситеми.
* **Статичні тести** проводились програмістами в процесі розробки системи. Такі тести базуються на аналізі коду, що є ефективним у випадку невеликих за об’ємом коду систем. Також такі тести є економічно вигідними оскільки виконуються безпосередньо програмістами без участі тестерів.

Останній пункт тестування – це перевірка сумісності. На цьому кроці система тестувалася на сумісність з різними операційними системами і програмним забезпеченням. В результаті тестування було підтверджено коректну роботу системи на всіх версіях ОС(як для ПК, так і для телефонних пристроїв та планшетів). Також система є доступною для використання у основних браузерах (Google Chrome, IE, Opera, Mozilla Firefox, Safari, а також міні-версії вищеперерахованих браузерів для мобільних пристроїв).

**5. Приклади реалізації**

Продукт розробляється на мові програмування **Java** із застосуванням ***Visual Beginners All-purpose Symbolic Instruction Code Script***(VBScript, або Visual BASIC Script, візуальний символічний універсальний командний код для початківців) - мова, що інтерпретується, вбудовується в літі-документ з метою включення до складу Web-сторінки інтерактивних елементів. Visual Basic Scripting Edition є компонентом Windows Script Host. Він широко використовується при створенні скриптів в операційних системах сімейства Windows. VBScript було створено компанією Microsoft з метою заміни застарілої пакетної мови. VBScript є спрощеною версією синтаксису Visual Basic, зокрема не підтримується типізація: усі змінні мають тип Variant.

**ВИСНОВКИ**

При проходженні практики на ПрАТ «Видавництво «Вільна Україна»» я ознайомився з технологічними процесами керування інформаційними потоками, організацією зберігання даних, створення баз даних і доступу до даних. Вивчив послідовності методів і засобів обробки даних, основи конфігурування технічних засобів, інформаційних систем та мереж, систем проектування та керування. Опанував технології розробки, впровадження та супроводу програмного забезпечення. Оволодів сучасними технологіями оброблення інформації та застосування їх для виконання практичних занять. Набув технічних, проектувальних та виконавських навичок. Набув навичок групової роботи та роботи в колективі спеціалістів. Виконав дане мені індивідуальне завдання .В процесі роботи я розглянув різні уже створені успішні продукти даної тематики, виділив позитивні і негативні моменти таких засобів.

# Список літератури

1. Doodle. [В Інтернеті] http://doodle.com/ru/.

2. CIVS. [В Інтернеті] http://civs.cs.cornell.edu/.

3. Demopanel. [В Інтернеті] http://demopanel.freehost.com.ua/pols.php.

4. VNS. [В Інтернеті] http://vns.lp.edu.ua/.

5. 6. Технология разработки программного обеспечения Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Виснадул, 2008.

6. Дронов В. А. - HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов.

7. Підручник з Umbrello UML Modeller.

**Додаток А**

Фрагмент коду

addConditionalDependencies(pkgs);  
requirejs(pkgs, function () {  
wixBiSession.packagesLoaded = Date.now();  
if (prevError) {  
var err = JSON.parse(prevError);  
err.errn = 'REQUIREJS\_RETRY\_ERROR';  
err.errc = 111025;  
err.severity = 10;  
requirejs.onError(JSON.parse(prevError));  
}  
function buildFunctionParametersObject(\_pkgs, args) {  
return \_pkgs.reduce(function (result, pkg, index) {  
result[pkg] = args[index];  
return result;  
}, {});  
}  
function initConditionalDependencies(\_pkgs) {  
if (\_pkgs.qaAutomation) {  
\_pkgs.qaAutomation.init(window, siteModel);  
}  
if (\_pkgs.tpaIntegration) {  
\_pkgs.tpaIntegration.init(window);  
}  
}

var p = buildFunctionParametersObject(pkgs, arguments);  
initConditionalDependencies(p);  
var ajaxHandler = getAjaxHandler(p.zepto);  
p.utils.ajaxLibrary.register(ajaxHandler);  
p.utils.ajaxLibrary.enableJsonpHack();  
// Wait for DOM to be ready before accessing it, e.g. getElementById  
p.zepto(function () {  
if (siteModel.wixData) {  
var siteStructureNode = document.getElementById('SITE\_STRUCTURE');  
siteModel.wixHtmlRaw = siteStructureNode.outerHTML;  
siteModel.wixAnchors = window.anchors || {};  
siteStructureNode.parentNode.removeChild(siteStructureNode);  
}  
siteModel.requestModel = {  
userAgent: window.navigator.userAgent,  
cookie: document.cookie,  
storage: p.utils.storage(window)  
};  
siteModel.currentUrl = p.utils.urlUtils.parseUrl(location.href);  
siteModel.forceMobileView = window.forceMobileView;  
callback(p, ajaxHandler);  
});  
}, function (err) {  
if (!config.urlArgs) {  
if (err.requireModules) {  
err.requireModules.forEach(function (module) {  
requirejs.undef(module);  
});  
}  
config.urlArgs = 'c=' + Date.now();  
requirejs.config(config);  
requirejs(pkgs, function () {  
err.errn = 'REQUIREJS\_RETRY\_ERROR';  
err.errc = 111025;  
err.severity = 10;  
requirejs.onError(err);  
}, function (error) {  
requirejs.onError(error);  
});  
}  
});  
}

function getDSConfig(configs) {  
var activeConfig = configs.fullFunctionality;  
activeConfig.origin = queryUtil.getParameterByName('dsOrigin');  
return activeConfig;  
}  
function fixPages(p, pagesData) {  
var \_ = p.lodash;  
var pageIds = \_(pagesData).keys().pull('masterPage').value();  
return \_.mapValues(pagesData, function (data, pageId) {  
// don't fix pages from the server  
if (window.pagesData && window.pagesData[pageId]) {  
return data;  
}  
return p.utils.dataFixer.fix(data, pageIds.slice());  
});  
}  
function buildRenderedSite(p, siteData, callback) {  
var \_ = p.lodash;  
p.core.renderer.renderSite(siteData, function (renderedReact) {  
if (window.rendered) {  
window.rendered.forceUpdate();  
} else {  
var a = document.getElementById("SITE\_CONTAINER").children[0];  
window.rendered = p.react.render(renderedReact, document.getElementById("SITE\_CONTAINER"));  
if (p.qaAutomation) {  
\_.set(window, 'testApi.domSelectors', p.qaAutomation.getDomSelectors(p.react));  
window.testApi.domSelectors.setSearchRoot(window.rendered);  
}  
var b = document.getElementById("SITE\_CONTAINER").children[0];  
window.sssr.success = (a === b);  
window.sssr.clientSideRender = {  
sinceInitialTimestamp: (Date.now() - window.wixBiSession.initialTimestamp),  
performanceNow: performanceNow()  
};